OTRS
Open Technology
Real Services

Documentation

OTRS 7 - Developer Manual

Build Date:

2018-08-16

OTRS 7 - Developer Manual

A dokumentum forráskódja megtalálható a githubon a doc-developer tárolóban. A hozzájárulásokat mindennél jobban köszönjük. Segíthet a saját nyelvére való fordításban is a Transifex oldalon.

Tartalom

Az ábrák listája

A táblázatok listája

1. fejezet - Kezdeti lépések

Az OTRS egy többplatformos webes alkalmazás-keretrendszer, amelyet eredetileg hibajegyrendszernek fejlesztettek. Különböző webkiszolgálókat és adatbázisokat támogat.

Ez a kézikönyv azt mutatja be, hogy hogyan fejleszthet saját OTRS modulokat és alkalmazásokat az OTRS stílus útmutatók alapján.

1. Fejlesztői környezet

Az OTRS bővítőmodulok írásának megkönnyítéséhez egy fejlesztői környezet létrehozása szükséges. Az OTRS és a további nyilvános modulok forráskódja megtalálható a githubon.

1.1. A keretrendszer letöltése

Először is létre kell hozni egy könyvtárat, amelyben eltárolhatók a modulok. Ezután lépje be az új könyvtárba a parancssor használatával, és töltse le azokat a következő parancs használatával:

```
# a git master ágnál
shell> git clone git@github.com:OTRS/otrs.git -b master
# egy adott ágnál, például OTRS 3.3
shell> git clone git@github.com:OTRS/otrs.git -b rel-3_3
```

Töltse le a `module-tools` modult is (a githubról) a fejlesztői környezetéhez. Ez számos hasznos eszközt tartalmaz:

```
shell> git clone git@github.com:OTRS/module-tools.git
```

Állítsa be az OTRS rendszert az adminisztrációs kézikönyvben olvasható telepítési utasítások szerint.

1.2. Hasznos eszközök

Létezik két modul, amely erősen ajánlott az OTRS fejlesztésnél: OTRSCodePolicy és Fred.

Az OTRSCodePolicy egy kódolási minőségellenőrző, amely kikényszeríti a közös kódolási szabványok használatát az OTRS fejlesztőcsapatnál is. Erősen ajánlott ennek használata, ha hozzájárulásokat tervez készíteni. Használhatja önálló tesztelő parancsfájlként, vagy akár regisztrálhatja egy olyan git véglegesítés horogként, amely minden alkalommal lefut, amikor egy véglegesítést készít. A részletekért nézze meg a modul dokumentációját.

A Fred egy kicsi fejlesztést segítő modul, amelyet ténylegesen telepíthet vagy hozzákapcsolhat (a lent leírt módon) a fejlesztői rendszeréhez. Számos hasznos modult szerepeltet, amelyet bekapcsolhat, mint például egy SQL naplózóként vagy egy szabványos hibakimenet konzolként. Néhány további részletet találhat a modul dokumentációjában.

Mellesleg ezek az eszközök szintén nyílt forrásúak, és nagyon örülnénk bármilyen továbbfejlesztésnek, amellyel hozzájárul.

1.3. Bővítőmodulok hozzákapcsolása

Egyértelmű szétválasztás szükséges az OTRS és a modulok között a megfelelő fejlesztéshez. Különösen egy git klón használatakor kritikus az egyértelmű szétválasztás. Annak érdekében, hogy megkönnyítse az OTRS-nek a fájlokhoz történő hozzáférést,

linkeket kell létrehozni. Ez megtehető a könyvtármodul eszközök tárolójában lévő parancsfájllal. Példa: a Naptár modul hozzákapcsolása:

```
shell> ~/src/module-tools/link.pl ~/src/Calendar/ ~/src/otrs/
```

Amikor új fájlok kerülnek hozzáadásara, akkor azokat a fent bemutatott módon kell hozzákapcsolni.

Amint a hozzákapcsolás befejeződött, újra kell építeni a rendszerbeállításokat a modul regisztrálásához az OTRS-be. A további SQL vagy Perl-kódot is végre kell hajtani a modulból. Példa:

```
shell> ~/src/otrs/bin/otrs.Console.pl Maint::Config::Rebuild
shell> ~/src/module-tools/DatabaseInstall.pl -m Calendar.sopm -a install
shell> ~/src/module-tools/CodeInstall.pl -m Calendar.sopm -a install
```

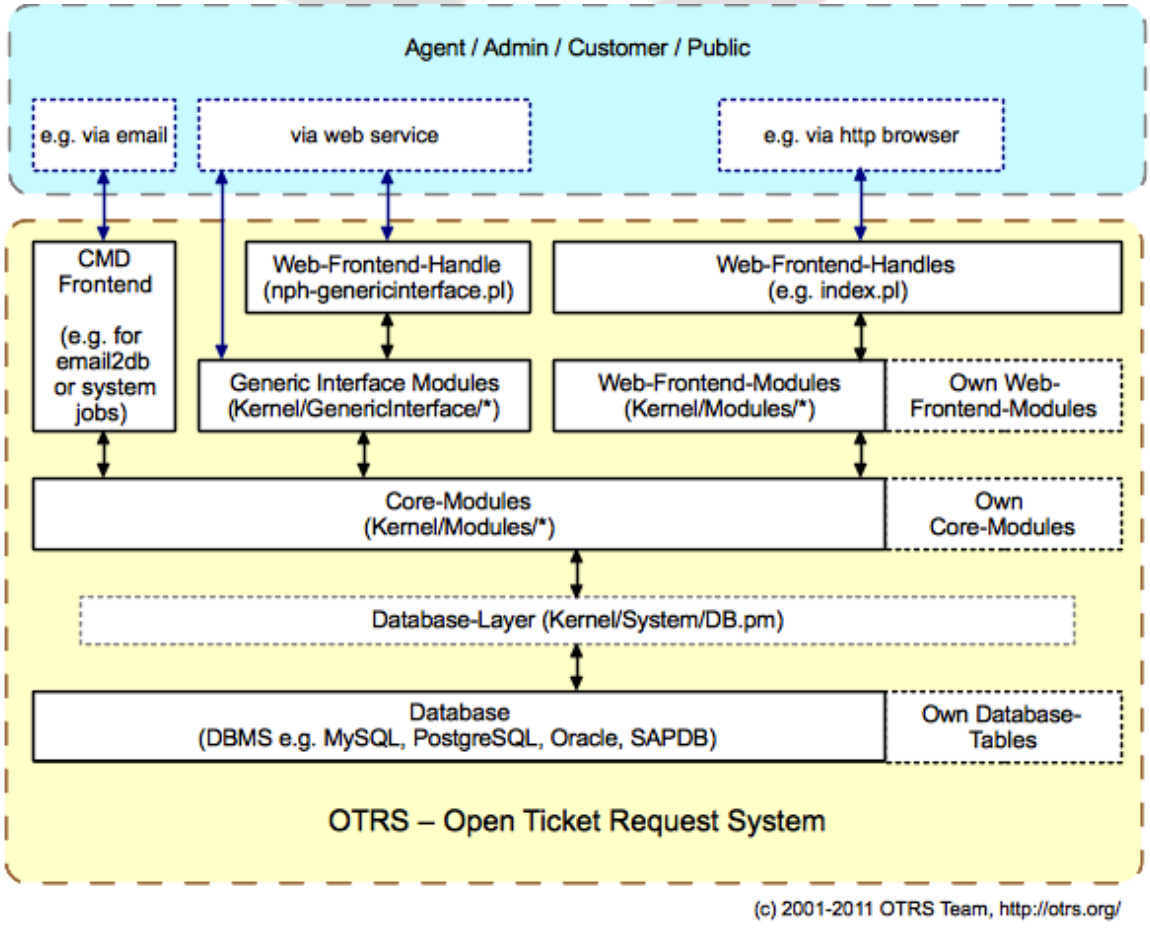
A kapcsolatoknak az OTRS-ből történő eltávolításához adja ki a következő parancsot:

```
shell> ~/src/module-tools/remove_links.pl ~/src/otrs/
```

2. Szerkezeti áttekintés

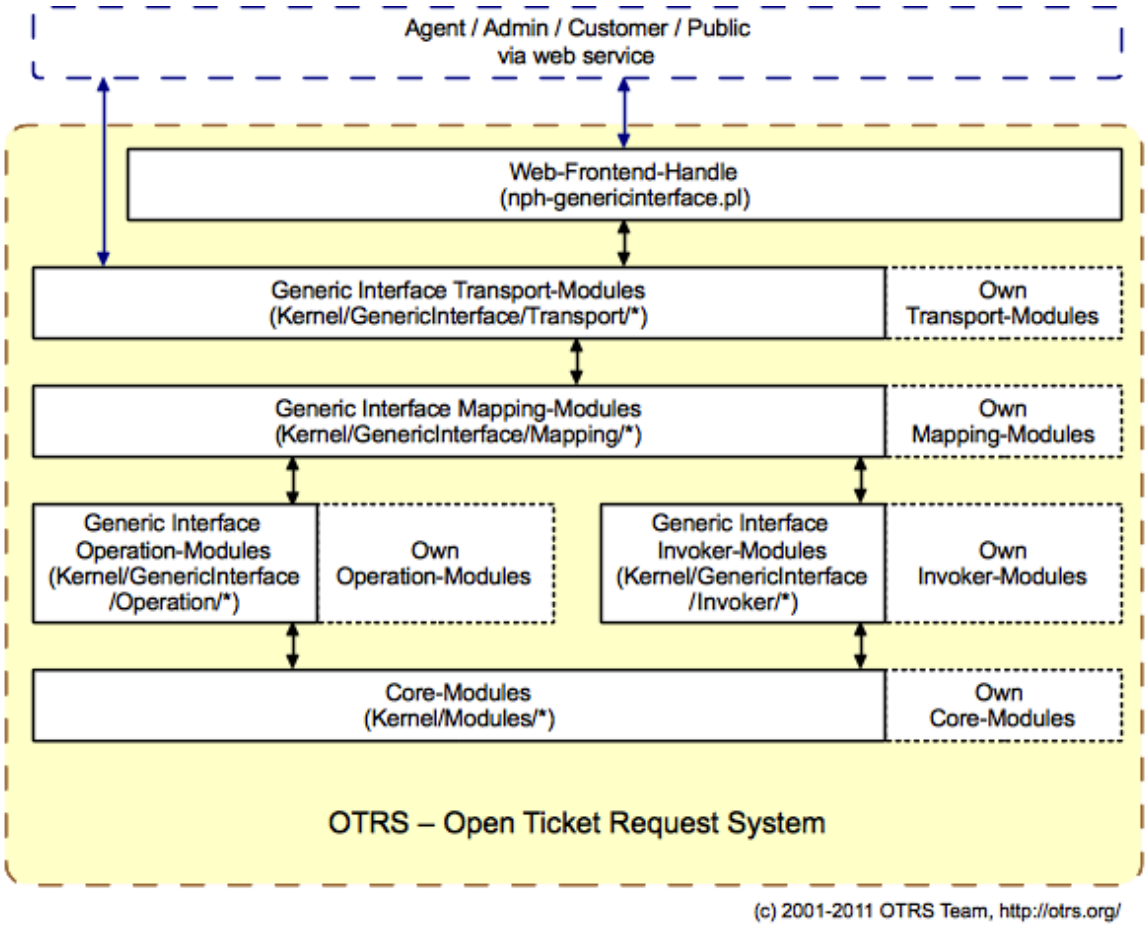
Az OTRS keretrendszer moduláris. A következő kép az OTRS alapvető rétegszerkezetét jeleníti meg.

1.1. ábra - Az OTRS szerkezete



Az OTRS 3.1-ben bevezetett OTRS általános felület folytatja az OTRS modularitását. A következő kép az általános felület alapvető rétegszerkezetét jeleníti meg.

1.2. ábra - Az általános felület szerkezete



2.1. Könyvtárak

Könyvtár	Leírás
bin/	parancssori eszközök
bin/cgi-bin/	webes kezelő
bin/fcgi-bin/	fast cgi webes kezelő
Kernel	alkalmazás kódbázisa
Kernel/Config/	beállítófájlok
Kernel/Config/Files	beállítófájlok
Kernel/GenericInterface/	az általános felület API-ja
Kernel/GenericInterface/Invoker/	meghívó modulok az általános felülethez
Kernel/GenericInterface/Mapping/	leképező modulok az általános felülethez, például: Simple
Kernel/GenericInterface/Operation/	műveleti modulok az általános felülethez
Kernel/GenericInterface/Transport/	átviteli modulok az általános felülethez, például: „HTTP SOAP”
Kernel/Language	nyelvi fordítási fájlok
Kernel/Scheduler/	az ütemező fájljai

Könyvtár	Leírás
Kernel/Scheduler/TaskHandler	kezelőmodulok az ütemező feladatokhoz, például: GenericInterface
Kernel/System/	alapmodulok, például: Log, Ticket...
Kernel/Modules/	előtétprogram modulok, például: QueueView...
Kernel/Output/HTML/	HTML sablonok
var/	változó adatok
var/log	naplófájlok
var/cron/	cron fájlok
var/httpd/htdocs/	htdocs könyvtár az index.html fájllal
var/httpd/htdocs/skins/Agent/	az ügyintéző felületéhez elérhető felszínek
var/httpd/htdocs/skins/Customer/	az ügyfélfelülethez elérhető felszínek
var/httpd/htdocs/js/	JavaScript fájlok
scripts/	egyéb fájlok
scripts/test/	egységteszt fájlok
scripts/test/sample/	egységteszt példaadat fájlok

2.2. Fájlok

.pl = Perl

.pm = Perl-modul

.tt = Template::Toolkit sablonfájlok

.dist = Fájlok alapértelmezett sablonjai

.yaml vagy .yml = A webszolgáltatás beállításaihoz használt YAML-fájlok

2.3. Alapmodulok

Az alapmodulok az `$OTRS_HOME/Kernel/System/*` alatt találhatók. Ez a réteg a logikai munkához van. Az alapmodulok a rendszer rutinjai kezeléséhez használhatók, mint például „jegy zárolása” és „jegy létrehozása”. Néhány fő alapmodul a következő:

- `Kernel::System::Config` (beállítási lehetőségek eléréséhez)
- `Kernel::System::Log` (naplózáshoz az OTRS naplózó háttérprogramjába)
- `Kernel::System::DB` (az adatbázis háttérprogram eléréséhez)
- `Kernel::System::Auth` (felhasználói hitelesítés ellenőrzéséhez)
- `Kernel::System::User` (felhasználók kezeléséhez)
- `Kernel::System::Group` (csoportok kezeléséhez)
- `Kernel::System::Email` (e-mailek küldéséhez)

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.4. Előtétprogram kezelés

A böngésző, a webkiszolgáló és az előtétprogram modulok közti felület. Egy előtétprogram modul használható HTTP-hivatkozáson keresztül.

http://localhost/otrs/index.pl?Action=Module

2.5. Előtétprogram modulok

Az előtétprogram modulok az `$OTRS_HOME/Kernel/Modules/*.pm` alatt találhatók. Két nyilvános függvény van ebben - `new()` és `run()` - amelyek az előtétprogram kezelésből érhetők el (például `index.pl`).

A `new()` előtétprogram modul objektumok létrehozásához használható. Az előtétprogram kezelés biztosítja a használt előtétprogram modulokat az alapvető keretrendszer objektumokkal. Ezek például a következők: `ParamObject` (webes űrlapparaméterek lekéréséhez), `DBObject` (meglévő adatbázis-kapcsolatok használatához), `LayoutObject` (sablonok és egyéb HTML elrendezés függvények használatához), `ConfigObject` (konfigurációs beállítások hozzáféréséhez), `LogObject` (a keretrendszer naplózó rendszerének használatához), `UserObject` (a felhasználói függvények lekéréséhez a jelenlegi felhasználótól), `GroupObject` (a csoportfüggvények lekéréséhez).

Az alapmodulokról további információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.6. CMD előtétprogram

A CMD (Command - Parancs) előtétprogram a webes előtétprogram kezeléshez és a webes előtétprogram modulokhoz együtt hasonló (csak a `LayoutObject` nélkül), és az alapmodulokat használja néhány műveletnél a rendszeren.

2.7. Általános felület modulok

Az általános felület modulok az `$OTRS_HOME/Kernel/GenericInterface/*` alatt találhatók. Az általános felület modulok egy webszolgáltatás végrehajtásának egyes részei kezeléséhez használhatók a rendszeren. Az általános felület fő moduljai a következők:

- `Kernel::GenericInterface::Transport` (kölcsönhatásba lépéshez távoli rendszerekkel)
- `Kernel::GenericInterface::Mapping` (adatok átalakításához egy szükséges formátumba)
- `Kernel::GenericInterface::Requester` (az OTRS mint kérelmezőként való használatához a webszolgáltatásnál)
- `Kernel::GenericInterface::Provider` (az OTRS mint kiszolgálóként való használatához a webszolgáltatásnál)
- `Kernel::GenericInterface::Operation` (szolgáltató műveletek végrehajtásához)
- `Kernel::GenericInterface::Invoker` (kérelmező műveletek végrehajtásához)
- `Kernel::GenericInterface::Debugger` (webszolgáltatás kommunikációjának követéséhez naplóbejegyzések használatával)

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.7.1. Általános felület meghívó modulok

Az általános felület meghívó modulok az `$OTRS_HOME/Kernel/GenericInterface/Invoker/*` alatt találhatók. Minden egyes meghívót egy `Controller` elnevezésű mappa tartalmaz. Ez a megközelítés nem csak a belső osztályoknak és metódusoknak segít

egy névteret meghatározni, hanem a fájlneveknek is. Például az `$OTRS_HOME/Kernel/GenericInterface/Invoker/Test/` az összes Teszt-típusú meghívó vezérlője.

Az általános felület meghívó modulok háttérprogramként használhatók kérések létrehozásához a távoli rendszereknél műveletek végrehajtásához.

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.7.2. Általános felület leképező modulok

Az általános felület leképező modulok az `$OTRS_HOME/Kernel/GenericInterface/Mapping/*` alatt találhatók. Ezek a modulok adatok (kulcsok és értékek) átalakításához használhatók az egyik formátumról egy másikra.

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.7.3. Általános felület műveleti modulok

Az általános felület műveleti modulok az `$OTRS_HOME/Kernel/GenericInterface/Operation/*` alatt találhatók. Minden egyes műveletet egy `Controller` elnevezésű mappa tartalmaz. Ez a megközelítés nem csak a belső osztályoknak és metódusoknak segít egy névteret meghatározni, hanem a fájlneveknek is. Például az `$OTRS_HOME/Kernel/GenericInterface/Operation/Ticket/` az összes Ticket-típusú művelet vezérlője.

Az általános felület műveleti modulok háttérprogramként használhatók egy távoli rendszer által kért műveletek végrehajtásához.

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.7.4. Általános felület átviteli modulok

Az általános felület átviteli modulok az `$OTRS_HOME/Kernel/GenericInterface/Operation/*` alatt találhatók. Minden egyes átviteli modult egy olyan elnevezésű könyvtárba kell elhelyezni, ahogy a használt hálózati protokollt hívják. Például a „HTTP SOAP” átviteli modul a `$OTRS_HOME/Kernel/GenericInterface/Transport/HTTP/SOAP.pm` fájlban található.

Az általános felület átviteli modulok adatok küldéséhez használhatók egy távoli rendszer számára, valamint adatok fogadásához tőle.

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.8. Ütemező feladatkezelő modulok

Az ütemező feladatkezelő modulok az `$OTRS_HOME/Kernel/Scheduler/TaskHandler/*` alatt találhatók. Ezek a modulok aszinkron feladatok végrehajtásához használhatók. Például a `GenericInterface` feladatkezelő általános felület kéréseket hajt végre a távoli rendszerekhez az apache folyamaton kívül. Ez abban segíti a rendszert, hogy fogékonyabb legyen, megelőzve a lehetséges teljesítményproblémákat.

További információkért nézze meg a http://otrs.github.io/doc/ oldalt.

2.9. Adatbázis

Az adatbázis-felület különböző adatbázisokat támogat.

Az OTRS adatmodelljéhez nézze meg a `/doc` könyvtárban lévő fájlokat. Alternatívaként megnézheti az adatmodellt a githubon is.

2. fejezet - OTRS belsőségek - hogyan működik

1. Beállítási mechanizmus

OTRS comes with a dedicated mechanism to manage configuration options via a graphical interface (System Configuration). This section describes how it works internally and how you can provide new configuration options or change existing default values.

1.1. `Defaults.pm`: az OTRS alapértelmezett beállításai

The default configuration file of OTRS is `Kernel/Config/Defaults.pm`. This file is needed for operation of freshly installed systems without a deployed XML configuration and should be left untouched as it is automatically updated on framework updates.

1.2. Automatikusan előállított beállítófájlok

A `Kernel/Config/Files` mappában néhány automatikusan előállított beállítófájl található:

`ZZZAAuto.pm`

Perl cache of the XML configuration's current values (default or modified by user)

`ZZZACL.pm`

Perl cache of ACL configuration from database

`ZZZProcessManagement.pm`

Perl cache of ProcessManagement configuration from database

These files are a Perl representation of the current system configuration. They should never be manually changed as they are overwritten by OTRS.

1.3. XML Configuration Files

In OTRS, configuration options that the administrator can configure via SysConfig are provided via XML files with a special format. To convert old XML's you can use 'otrs.Console.pl Dev::Tools::Migrate::ConfigXMLStructure' command. The file `Kernel/Config/Files/ZZZAAuto.pm` is a cached Perl version of the XML that contains all settings with their current value. It can be (re-)generated with `bin/otrs.Console.pl Maint::Config::Rebuild`.

Note:`$OTRS_HOME/Kernel/Config/Files/ZZZAuto.pm` does not exist any more, it has been merged into `$OTRS_HOME/Kernel/Config/Files/ZZZAAuto.pm`.

Az egyes XML beállítófájloknak a következő elrendezésük van:

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="2.0" init="Changes">

    <!--  settings will be here -->
```

```
</otrs_config>
```

Attributes of the `otrs_config` element

`init`

The global `init` attribute describes where the config options should be loaded. There are different levels available and will be loaded/overloaded in the following order: `Framework` (for framework settings e. g. session option), `Application` (for application settings e. g. ticket options), `Config` (for extensions to existing applications e. g. ITSM options) and Changes (for custom development e. g. to overwrite framework or ticket options).

The configuration items are written as `Setting` elements with a `Description`, a `Navigation` group (for the tree-based navigation in the GUI) and the `Value` that it represents. Here's an example:

```
<Setting Name="Ticket::Hook" Required="1" Valid="1">
    <Description Translatable="1">The identifier for a ticket, e.g. Ticket#, Call#,
 MyTicket#. The default is Ticket#.</Description>
    <Navigation>Core::Ticket</Navigation>
    <Value>
        <Item ValueType="String" ValueRegex="">Ticket#</Item>
    </Value>
</Setting>
```

Attributes of the `Setting` element

`Required`

If this is set to 1, the config setting cannot be disabled.

`Valid`

Determines if the config setting is active (1) or inactive (0) by default.

`ConfigLevel`

Ha az opcionális `ConfigLevel` attribútum be van állítva, akkor a beállítási változót esetleg nem szerkesztheti az adminisztrátor a saját beállítási szintjétől függően. A `ConfigLevel` beállítási változó állítja be az adminisztrátor szakmai tapasztalatának szintjét. Lehet 100 (Szakértő), 200 (Speciális) vagy 300 (Kezdő). Iránymutatásként, hogy mely beállítási szintet kell egy lehetőséghez megadni, az az ajánlott, hogy az összes olyan lehetőségnek, amelyet külső interakció beállításával kell megtenni (mint például Sendmail, LDAP, SOAP és egyebek), legalább 200 (Speciális) beállítási szintet kell kapnia.

`Invisible`

If set to 1, the config setting is not shown in the GUI.

`Readonly`

If set to 1, the config setting cannot be changed in the GUI.

`UserModificationPossible`

If `UserModificationPossible` is set to 1, administrators can enable user modifications of this setting (in user preferences). Please note that this feature requires the **OTRS Business Solution™**.

`UserModificationActive`

If `UserModificationActive` is set to 1, user modifications of this setting is enabled (in user preferences). You should use this attribute together with `UserModificationPossible`.

`UserPreferencesGroup`

Use `UserPreferencesGroup` attribute to define under which group config variable belongs (in the UserPreferences screen). You should use this attribute together with `UserModificationPossible`.

Guidelines for placing settings in the right `Navigation` nodes

- Only create new nodes if neccessary. Avoid nodes with only very few settings if possible.
- On the first tree level, no new nodes should be added.
- Don't place new settings in `Core` directly. This is reserved for a few important global settings.
- `Core::*` can take new groups that contain settings that cover the same topic (like `Core::Email`) or relate to the same entity (like `Core::Queue`).
- All event handler registrations go to `Core::Event`.
- Don't add new direct child nodes within `Frontend`. Global front end settings go to `Frontend::Base`, settings only affecting a part of the system go to the respective `Admin`, `Agent`, `Customer` or `Public` sub nodes.
- Front end settings that only affect one screen should go to the relevant screen (`View`) node (create one if needed), for example `AgentTicketZoom` related settings go to `Frontend::Agent::View::TicketZoom`. If there are module layers within one screen with groups of related settings, they would also go to a sub group here (e. g. `Frontend::Agent::View::TicketZoom::MenuModule` for all ticket zoom menu module registrations).
- All global Loader settings go to `Frontend::Base::Loader`, screen specific Loader settings to `Frontend::*::ModuleRegistration::Loader`.

1.3.1. Structure of `Value` elements

`Value` elements hold the actual configuration data payload. They can contain single values, hashes or arrays.

1.3.1.1. `Item`

An `Item` element holds one piece of data. The optional `ValueType` attribute determines which kind of data and how it needs to be presented to the user in the GUI. If no `ValueType` is specified, it defaults to `String`.

Please see below for a definition of the different value types.

```
<Setting Name="Ticket::Hook" Required="1" Valid="1">
    <Description Translatable="1">The identifier for a ticket, e.g. Ticket#, Call#,
 MyTicket#. The default is Ticket#.</Description>
    <Navigation>Core::Ticket</Navigation>
    <Value>
        <Item ValueType="String" ValueRegex="">Ticket#</Item>
    </Value>
</Setting>
```

1.3.1.2. Array

Ezzel a beállítási elemmel tömbök jeleníthetők meg.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Array>
            <Item Translatable="1">Value 1</Item>
            <Item Translatable="1">Value 2</Item>
            ...
        </Array>
    </Value>
</Setting>
```

1.3.1.3. Hash

Ezzel a beállítási elemmel kivonatok jeleníthetők meg.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Hash>
            <Item Key="One" Translatable="1">First</Item>
            <Item Key="Two" Translatable="1">Second</Item>
            ...
        </Hash>
    </Value>
</Setting>
```

It's possible to have nested array/hash elements (like hash of arrays, array of hashes, array of hashes of arrays, ...). Below is an example array of hashes.

```
<Setting Name="ExampleAoH">
    ...
    <Value>
        <Array>
            <DefaultItem>
                <Hash>
                    <Item></Item>
                </Hash>
            </DefaultItem>
            <Item>
                <Hash>
                    <Item Key="One">1</Item>
                    <Item Key="Two">2</Item>
                </Hash>
            </Item>
            <Item>
                <Hash>
                    <Item Key="Three">3</Item>
                    <Item Key="Four">4</Item>
                </Hash>
            </Item>
        </Array>
    </Value>
</Setting>
```

1.3.2. Value Types

The XML config settings support various types of configuration variables.

1.3.2.1. Szöveg

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="String" ValueRegex=""></Item>
    </Value>
</Setting>
```

A config element for numbers and single-line strings. Checking the validity with a regular expression is possible (optional). This is the default `ValueType`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="String" ValueRegex="" Translatable="1">Value</Item>
    </Value>
</Setting>
```

Az opcionális `Translatable` attribútum jelöli meg ezt a beállítást fordíthatóként, amely azt fogja eredményezni, hogy fel lesz véve az OTRS fordítási fájljaiba. Ezt az attribútumot bármely címkére el lehet helyezni (lásd lent is).

1.3.2.2. Jelszó

A config element for passwords. It's still stored as plain text in the XML, but it's masked in the GUI.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Password">Secret</Item>
    </Value>
</Setting>
```

1.3.2.3. PerlModule

A config element for Perl module. It has a `ValueFilter` attribute, which filters possible values for selection. In the example below, user can select Perl module `Kernel::System::Log::SysLog` or `Kernel::System::Log::File`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="PerlModule" ValueFilter="Kernel/System/Log/
*.pm">Kernel::System::Log::SysLog</Item>
    </Value>
</Setting>
```

1.3.2.4. Szövegdoboz

Egy beállítási elem többsoros szöveghez.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Textarea"></Item>
    </Value>
</Setting>
```

1.3.2.5. Kiválasztás

This config element offers preset values as a pull-down menu. The SelectedID or SelectedValue attributes can pre-select a default value.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Select" SelectedID="Queue">
            <Item ValueType="Option" Value="Queue" Translatable="1">Queue</Item>
            <Item ValueType="Option" Value="SystemAddress" Translatable="1">System address</
Item>
        </Item>
    </Value>
</Setting>
```

1.3.2.6. Jelölőnégyzet

This config element checkbox has two states: 0 or 1.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Checkbox">0</Item>
    </Value>
</Setting>
```

1.3.2.7. Dátum

This config element contains a date value.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Date">2016-02-02</Item>
    </Value>
</Setting>
```

1.3.2.8. DateTime

This config element contains a date and time value.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="DateTime">2016-12-08 01:02:03</Item>
    </Value>
</Setting>
```

1.3.2.9. Könyvtár

This config element contains a directory.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Directory">/etc</Item>
    </Value>
</Setting>
```

1.3.2.10. Fájl

This config element contains a file path.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="File">/etc/hosts</Item>
    </Value>
</Setting>
```

1.3.2.11. Entity

This config element contains a value of a particular entity. `ValueEntityType` attribute defines the entity type. Supported entities: `DynamicField`, `Queue`, `Priority`, `State` and `Type`. Consistency checks will ensure that only valid entities can be configured and that entities used in the configuration cannot be set to invalid. Also, when an entity is renamed, all referencing config settings will be updated.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="Entity" ValueEntityType="Queue">Junk</Item>
    </Value>
</Setting>
```

1.3.2.12. TimeZone

This config element contains a time zone value.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="TimeZone">UTC</Item>
    </Value>
</Setting>
```

1.3.2.13. VacationDays

This config element contains definitions for vacation days which are repeating each year. Following attributes are mandatory: `ValueMonth`, `ValueDay`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="VacationDays">
            <DefaultItem ValueType="VacationDays"></DefaultItem>
            <Item ValueMonth="1" ValueDay="1" Translatable="1">New Year's Day</Item>
            <Item ValueMonth="5" ValueDay="1" Translatable="1">International Workers' Day</
Item>
            <Item ValueMonth="12" ValueDay="24" Translatable="1">Christmas Eve</Item>
        </Item>
    </Value>
</Setting>
```

1.3.2.14. VacationDaysOneTime

This config element contains definitions for vacation days which occur only once. Following attributes are mandatory: `ValueMonth`, `ValueDay` and `ValueYear`.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="VacationDaysOneTime">
            <Item ValueYear="2004" ValueMonth="1" ValueDay="1">test</Item>
        </Item>
    </Value>
</Setting>
```

1.3.2.15. WorkingHours

This config element contains definitions for working hours.

```
<Setting Name="SettingName">
    ...
    <Value>
        <Item ValueType="WorkingHours">
            <Item ValueType="Day" ValueName="Mon">
                <Item ValueType="Hour">8</Item>
                <Item ValueType="Hour">9</Item>
            </Item>
            <Item ValueType="Day" ValueName="Tue">
                <Item ValueType="Hour">8</Item>
                <Item ValueType="Hour">9</Item>
            </Item>
        </Item>
    </Value>
</Setting>
```

1.3.2.16. Frontend Registration

Modul regisztráció az ügyintézői felülethez:

```
<Setting Name="SettiFrontend::Module###AgentModuleName">
    ...
    <Value>
        <Item ValueType="FrontendRegistration">
            <Hash>
                <Item Key="Group">
                    <Array>
                    </Array>
                </Item>
                <Item Key="GroupRo">
                    <Array>
                    </Array>
                </Item>
                <Item Key="Description" Translatable="1">Phone Call.</Item>
                <Item Key="Title" Translatable="1">Phone-Ticket</Item>
                <Item Key="NavBarName">Ticket</Item>
            </Hash>
        </Item>
    </Value>
</Setting>
```

1.3.3. DefaultItem in Array and Hash

The new XML structure allows us to create complex structures. Therefore we need `DefaultItem` entries to describe the structure of the Array/Hash. If it's not provided, system considers that you want simple Array/Hash with scalar values. `DefaultItem` is used as a template when adding new elements, so it can contain additional attributes, like `ValueType`, and it can define default values.

Here are few examples:

1.3.3.1. Array of Array with Select items

```
<Array>
    <DefaultItem>
        <Array>
            <DefaultItem ValueType="Select" SelectedID='option-2'>
                <Item ValueType="Option" Value="option-1">Option 1</Item>
                <Item ValueType="Option" Value="option-2">Option 2</Item>
            </DefaultItem>
        </Array>
    </DefaultItem>
    ...
</Array>
```

1.3.3.2. Hash of Hash with Date items

```
<Hash>
    <DefaultItem>
        <Hash>
            <DefaultItem ValueType="Date">2017-01-01</DefaultItem>
        </Hash>
    </DefaultItem>
    ...
</Hash>
```

1.4. Hozzáférés a beállítási lehetőségekhez futási időben

Olvashatja és írhatja (egy kérésnél) a beállítási lehetőségeket a `Kernel::Config` alapmodulon keresztül.

Ha egy beállítási lehetőséget szeretne olvasni:

```
my $ConfigOption = $Kernel::OM->Get('Kernel::Config')->Get('Prefix::Option');
```

Ha meg szeretne változtatni egy beállítási lehetőséget futási időben, és csak ennél az egy kérésnél/folyamatnál:

```
$Kernel::OM->Get('Kernel::Config')->Set(
    Key => 'Prefix::Option'
    Value => 'Valami új érték',
);
```

2. Adatbázis mechanizmus

Az OTRS olyan adatbázis réteggel érkezik, amely különböző adatbázisokat támogat.

2.1. Hogyan működik

Az adatbázis rétegnek (`Kernel::System::DB`) két bemeneti lehetősége van: SQL és XML.

2.1.1. SQL

Az SQL felületet kell használni a normál adatbázis-műveleteknél (SELECT, INSERT, UPDATE, ...). Úgy használható mint egy normál Perl DBI felület.

2.1.1.1. INSERT/UPDATE/DELETE

```
$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "INSERT INTO table (name, id) VALUES ('Valamilyen név', 123)",
);

$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "UPDATE table SET name = 'Valamilyen név', id = 123",
);

$Kernel::OM->Get('Kernel::System::DB')->Do(
    SQL=> "DELETE FROM table WHERE id = 123",
);
```

2.1.1.2. SELECT

```
my $SQL = "SELECT id FROM table WHERE tn = '123'";

$Kernel::OM->Get('Kernel::System::DB')->Prepare(SQL => $SQL, Limit => 15);

while (my @Row = $Kernel::OM->Get('Kernel::System::DB')->FetchrowArray()) {
    $Id = $Row[0];
}
return $Id;
```

Megjegyzés

Vigyázzon arra, hogy a `Limit` megadását paraméterként használja, és ne az SQL utasításban, mert nem minden adatbázis támogatja a LIMIT kulcsszót az SQL lekérdezésekben.

```
my $SQL = "SELECT id FROM table WHERE tn = ? AND group = ?";

$Kernel::OM->Get('Kernel::System::DB')->Prepare(
    SQL   => $SQL,
    Limit => 15,
    Bind  => [ $Tn, $Group ],
);

while (my @Row = $Kernel::OM->Get('Kernel::System::DB')->FetchrowArray()) {
    $Id = $Row[0];
}
return $Id;
```

Megjegyzés

Használja a `Bind` attribútumot, ahol csak tudja, különösen a hosszú utasításoknál. Ha a `Bind` attribútumot használja, akkor nincs szükség a `Quote()` függvényre.

2.1.1.3. QUOTE

Szöveg:

```
my $QuotedString = $Kernel::OM->Get('Kernel::System::DB')->Quote("Ez egy probléma!");
```

Egész:

```
my $QuotedInteger = $Kernel::OM->Get('Kernel::System::DB')->Quote('123', 'Integer');
```

Szám:

```
my $QuotedNumber = $Kernel::OM->Get('Kernel::System::DB')->Quote('21.35', 'Number');
```

Megjegyzés

A `Bind` attribútumot használja a `Quote()` függvény helyett, ahol csak tudja.

2.1.2. XML

Az XML felületet kell használni INSERT, CREATE TABLE, DROP TABLE és ALTER TABLE utasításoknál. Mivel ez a szintaxis adatbázisról adatbázisra eltérő, ezért ennek használata gondoskodik arról, hogy olyan alkalmazásokat írjon, amelyek az összesnél használhatók.

Megjegyzés

Az `<Insert>` szintaxis megváltozott a 2.2 és újabb verziókban. Az értékeket mostantól a címketartalomban használják (többé nem egy attribútumban).

2.1.2.1. INSERT

```
<Insert Table="some_table">
    <Data Key="id">1</Data>
    <Data Key="description" Type="Quote">exploit</Data>
</Insert>
```

2.1.2.2. CREATE TABLE

Possible data types are: `BIGINT`, `SMALLINT`, `INTEGER`, `VARCHAR` (Size=1-1000000), `DATE` (Format: `yyyy-mm-dd hh:mm:ss`) and `LONGBLOB`.

```
<TableCreate Name="calendar_event">
    <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement="true" Type="BIGINT"/>
    <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
    <Column Name="content" Required="false" Size="250" Type="VARCHAR"/>
    <Column Name="start_time" Required="true" Type="DATE"/>
    <Column Name="end_time" Required="true" Type="DATE"/>
    <Column Name="owner_id" Required="true" Type="INTEGER"/>
    <Column Name="event_status" Required="true" Size="50" Type="VARCHAR"/>
    <Index Name="calendar_event_title">
        <IndexColumn Name="title"/>
    </Index>
    <Unique Name="calendar_event_title">
        <UniqueColumn Name="title"/>
    </Unique>
    <ForeignKey ForeignTable="users">
        <Reference Local="owner_id" Foreign="id"/>
    </ForeignKey>
</TableCreate>
```

`LONGBLOB` columns need special treatment. Their content needs to be `Base64` transcoded if the database driver does not support the feature `DirectBlob`. Please see the following example:

```
my $Content = $StorableContent;
if ( !$DBObject->GetDatabaseFunction('DirectBlob') ) {
    $Content = MIME::Base64::encode_base64($StorableContent);
}
```

Similarly, when reading from such a column, the content must not automatically be decoded as UTF-8 by passing the `Encode => 0` flag to `Prepare()`:

```
return if !$DBObject->Prepare(
    SQL => '
        SELECT content_type, content, content_id, content_alternative, disposition, filename
        FROM article_data_mime_attachment
        WHERE id = ?',
    Bind   => [ \$AttachmentID ],
    Encode => [ 1, 0, 0, 0, 1, 1 ],
);

while ( my @Row = $DBObject->FetchrowArray() ) {

    $Data{ContentType} = $Row[0];

    # Decode attachment if it's e. g. a postgresql backend.
    if ( !$DBObject->GetDatabaseFunction('DirectBlob') ) {
        $Data{Content} = decode_base64( $Row[1] );
    }
    else {
        $Data{Content} = $Row[1];
    }
    $Data{ContentID}          = $Row[2] || '';
    $Data{ContentAlternative} = $Row[3] || '';
    $Data{Disposition}        = $Row[4];
    $Data{Filename}           = $Row[5];
}
```

2.1.2.3. DROP TABLE

```
<TableDrop Name="calendar_event"/>
```

2.1.2.4. ALTER TABLE

A következő az oszlopok hozzáadásának, megváltoztatásának és eldobásának példáját jeleníti meg.

```
<TableAlter Name="calendar_event">
    <ColumnAdd Name="test_name" Type="varchar" Size="20" Required="true"/>

    <ColumnChange NameOld="test_name" NameNew="test_title" Type="varchar" Size="30"
 Required="true"/>

    <ColumnChange NameOld="test_title" NameNew="test_title" Type="varchar" Size="100"
 Required="false"/>

    <ColumnDrop Name="test_title"/>

    <IndexCreate Name="index_test3">
        <IndexColumn Name="test3"/>
    </IndexCreate>

    <IndexDrop Name="index_test3"/>

    <UniqueCreate Name="uniq_test3">
        <UniqueColumn Name="test3"/>
    </UniqueCreate>

    <UniqueDrop Name="uniq_test3"/>
</TableAlter>
```

A következő egy olyan példát jelenít meg, hogy hogyan nevezhető át egy tábla.

```
<TableAlter NameOld="calendar_event" NameNew="calendar_event_new"/>
```

2.1.2.5. Kód az XML feldolgozásához

```
my @XMLARRAY = @{$Self->ParseXML(String => $XML)};

my @SQL = $Kernel::OM->Get('Kernel::System::DB')->SQLProcessor(
    Database => \@XMLARRAY,
);
push(@SQL, $Kernel::OM->Get('Kernel::System::DB')->SQLProcessorPost());

for (@SQL) {
    $Kernel::OM->Get('Kernel::System::DB')->Do(SQL => $_);
}
```

2.2. Adatbázis-meghajtók

Az adatbázis-meghajtók az `$OTRS_HOME/Kernel/System/DB/*.pm` alatt találhatók.

2.3. Támogatott adatbázisok

- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server (csak külső adatbázis-kapcsolatokhoz, nem OTRS adatbázisként)

3. Naplózó mechanizmus

3.1. Rendszernapló

OTRS comes with a system log backend that can be used for application logging and debugging.

The Log object can be accessed and used via the `ObjectManager` like this:

```
$Kernel::OM->Get('Kernel::System::Log')->Log(
    Priority => 'error',
    Message  => 'Need something!',
);
```

Depending on the configured log level via `MinimumLogLevel` option in SysConfig, logged message will either be saved or not, based on their *Priority* flag.

If `error` is set, just errors are logged. With debug, you get all logging messages. The order of log levels is:

- `debug`
- `info`
- `notice`
- `error`

The output of the system log can be directed to either a syslog daemon or log file, depending on the configured `LogModule` option in SysConfig.

3.2. Communication Log

In addition to System Log, OTRS provides specialized logging backend for any communication related logging. Since OTRS 6, system comes with dedicated tables and frontends to track and display communication logs for easier debugging and operational overview.

To take advantage of the new system, first create a non-singleton instance of communication log object:

```
my $CommunicationLogObject = $Kernel::OM->Create(
    'Kernel::System::CommunicationLog',
    ObjectParams => {
        Transport   => 'Email',     # Transport log module
        Direction   => 'Incoming',  # Incoming|Outgoing
        AccountType => 'POP3',      # Mail account type
        AccountID   => 1,           # Mail account ID
    },
);
```

When you have a communication log object instance, you can start an object log for logging individual messages. There are two object logs currently implemented: `Connection` and `Message`.

`Connection` object log should be used for logging any connection related messages (for example: authenticating on server or retrieving incoming messages).

Simply, start the object log by declaring its type, and you can use it immediately:

```
$CommunicationLogObject->ObjectLogStart(
    ObjectLogType => 'Connection',
);

$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Debug',                               # Trace, Debug, Info, Notice,
 Warning or Error
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Open connection to 'host.example.com' (user-1).",
);
```

The communication log object instance handles the current started object logs, so you don't need to remember and bring them around everywhere, but it also means that you can only start one object per type.

If you encounter an unrecoverable error, you can choose to close the object log and mark it as failed:

```
$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Error',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => 'Something went wrong!',
);

$CommunicationLogObject->ObjectLogStop(
    ObjectLogType => 'Connection',
    Status        => 'Failed',
);
```

In turn, you can mark the communication log as failure as well:

```
$CommunicationLogObject->CommunicationStop(
    Status => 'Failed',
);
```

Otherwise, stop the object log and in turn communication log as success:

```
$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Connection',
    Priority      => 'Debug',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Connection to 'host.example.com' closed.",
);

$CommunicationLogObject->ObjectLogStop(
    ObjectLogType => 'Connection',
    Status        => 'Successful',
);

$CommunicationLogObject->CommunicationStop(
    Status => 'Successful',
);
```

Message object log should be used for any log entries regarding specific messages and their processing. It is used in a similar way, just make sure to start it before using it:

```
$CommunicationLogObject->ObjectLogStart(
    ObjectLogType => 'Message',
);

$CommunicationLogObject->ObjectLog(
    ObjectLogType => 'Message',
    Priority      => 'Error',
    Key           => 'Kernel::System::MailAccount::POP3',
    Value         => "Could not process message. Raw mail saved (report it on http://
bugs.otrs.org/)!",
);

$CommunicationLogObject->ObjectLogStop(
    ObjectLogType => 'Message',
    Status        => 'Failed',
);

$CommunicationLogObject->CommunicationStop(
    Status => 'Failed',
);
```

You also have the possibility to link the log object and later lookup the communications for a certain object type and ID:

```
$CommunicationLogObject->ObjectLookupSet(
    ObjectLogType    => 'Message',
    TargetObjectType => 'Article',
    TargetObjectID   => 2,
);

my $LookupInfo = $CommunicationLogObject->ObjectLookupGet(
    TargetObjectType => 'Article',
    TargetObjectID   => 2,
);
```

You should make sure to always stop communication and flag it as failed, if any log object failed as well. This will allow administrators to see failed communications in the overview, and take any action if needed.

It's important to preserve the communication log for duration of a single process. If your work is spanning over multiple modules and any of them can benefit from logging, make sure to pass the existing communication log instance around so all methods can use the same one. With this approach, you will make sure any log entries spawned for the same process are contained in a single communication.

If passing the communication log instance is not an option (async tasks!), you can also choose to recreate the communication log object in the same state as in previous step. Just get the communication ID and pass it to the new code, and then create the instance with this parameter supplied:

```
# Get communication ID in parent code.
my $CommunicationID = $CommunicationLogObject->CommunicationIDGet();

# Somehow pass communication ID to child code.
# ...

# Recreate the instance in child code by using same communication ID.
my $CommunicationLogObject = $Kernel::OM->Create(
    'Kernel::System::CommunicationLog',
    ObjectParams => {
        CommunicationID => $CommunicationID,
    },
);
```

You can then continue to use this instance as previously stated, start any object logs if needed, adding entries and setting status in the end.

If you need to retrieve the communication log data or do something else with it, please also take a look at `Kernel::System::CommunicationLog::DB.pm`

4. Date and Time

OTRS comes with its own package to handle date and time which ensures correct calculation and storage of date and time.

4.1. Bevezetés

Date and time are represented by an object of `Kernel::System::DateTime`. Every `DateTime` object holds its own date, time and time zone information. In contrast to the now deprecated `Kernel::System::Time` package, this means that you can and should create a `DateTime` object for every date/time you want to use.

4.2. Creation of a DateTime object

The object manager of OTRS has been extended by a `Create` method to support packages for which more than one instance can be created:

```
my $DateTimeObject = $Kernel::OM->Create(
    'Kernel::System::DateTime',
    ObjectParams => {
        TimeZone => 'Europe/Berlin'
    },
);
```

The example above will create a `DateTime` object for the current date and time in time zone Europe/Berlin. There are more options to create a `DateTime` object (time components, string, time stamp, cloning), see POD of `Kernel::System::DateTime`.

Megjegyzés

You will get an error if you try to retrieve a DateTime object via $Kernel::OM->Get('Kernel::System::DateTime').

4.3. Time zones

Time offsets in hours (+2, -10, etc.) have been replaced by time zones (Europe/Berlin, America/New_York, etc.). The conversion between time zones is completely encapsulated within a DateTime object. If you want to convert to another time zone, simply use the following code:

```
$DateTimeObject->ToTimeZone( TimeZone => 'Europe/Berlin' );
```

There is a new SysConfig option OTRSTimeZone. This setting defines the time zone that OTRS uses internally to store date and time within the database.

Megjegyzés

You have to ensure to convert a DateTime object to the OTRS time zone before it gets stored in the database (there's a convenient method for this: ToOTRSTimeZone()). An exception could be that you explicitly want a database column to hold a date/time in a specific time zone. But be aware that the database itself won't provide time zone information by itself when retrieving it.

Megjegyzés

TimeZoneList() of Kernel::System::DateTime provides a list of available time zones.

4.4. Method summary

The Kernel::System::DateTime package provides the following methods (this is only a selection, see source code for details).

4.4.1. Object creation methods

A DateTime object can be created either via the object manager's Create() method or by cloning another DateTime object with its Clone() method.

4.4.2. Get method

With Get() all data of a DateTime object will be returned as a hash (date and time components including day name, etc. as well as time zone).

4.4.3. Set method

With Set() you can either change certain components of the DateTime object (year, month, day, hour, minute, second) or you can set a date and time based on a given string ('2016-05-24 23:04:12'). Note that you cannot change the time zone with this method.

4.4.4. Time zone methods

To change the time zone of a DateTime object use method ToTimeZone() or as a shortcut for converting to OTRS time zone ToOTRSTimeZone().

To retrieve the configured OTRS time zone or user default time zone, always use method `OTRSTimeZoneGet()` or `UserDefaultTimeZoneGet()`. Never retrieve these manually via `Kernel::Config`.

4.4.5. Comparison operators and methods

`Kernel::System::DateTime` uses operator overloading for comparisons. So you can simply compare two `DateTime` objects with <, <=, ==, !=, >= and >. `Compare()` is usable in Perl's sort context as it returns -1, 0 or 1.

4.5. Deprecated package Kernel::System::Time

The now deprecated package `Kernel::System::Time` has been extended to fully support time zones instead of time offsets. This has been done to ensure that existing code works without (bigger) changes.

However, there is a case in which you have to change existing code. If you have code that uses the old time offsets to calculate a new date/time or a difference, you have to migrate this code to use the new `DateTime` object.

Example (old code):

```
my $TimeObject     = $Kernel::OM->Get('Kernel::System::Time'); # Assume a time offset of 0
 for this time object
my $SystemTime     = $TimeObject->TimeStamp2SystemTime( String => '2004-08-14 22:45:00' );
my $UserTimeZone   = '+2'; # normally retrieved via config or param
my $UserSystemTime = $SystemTime + $UserTimeZone * 3600;
my $UserTimeStamp  = $TimeObject->SystemTime2TimeStamp( SystemTime => $UserSystemTime );
```

Example (new code):

```
my $DateTimeObject = $Kernel::OM->Create('Kernel::System::DateTime'); # This implicitly sets
 the configured OTRS time zone
my $UserTimeZone   = 'Europe/Berlin'; # normally retrieved via config or param
$DateTimeObject->ToTimeZone( TimeZone => $UserTimeZone );
my $SystemTime     = $DateTimeObject->ToEpoch(); # note that the epoch is independent from
 the time zone, it's always calculated for UTC
my $UserTimeStamp = $DateTimeObject->ToString();
```

5. Felszínek

Az OTRS 3.0-ás verziója óta az OTRS látható megjelenése „felszínekkel” szabályozható.

Egy felszín CSS-fájlok és képfájlok halmaza, amelyek együtt azt vezérlik, hogy a grafikus felhasználói felület hogyan jelenjen meg a felhasználónak. A felszínek nem változtatják meg az OTRS által előállított HTML tartalmát (ez az, amit a „témák” csinálnak), hanem azt szabályozzák, hogy az hogyan jelenjen meg. A modern CSS szabványok segítségével lehetőség van a megjelenítés teljes megváltoztatására (például pérbeszédablakok egyes részeinek áthelyezésére, elemek elrejtésére, stb.).

5.1. Felszín alapok

Az összes felszín az `$OTRS_HOME/var/httpd/htdocs/skins/$SKIN_TYPE/$SKIN_NAME` mappában van. Kétféle típusú felszín létezik: ügyintézői és ügyfél felszín.

Az ügyintézők mindegyike egyénileg választhatja ki, hogy melyik telepített ügyintézői felszínt szeretnék „viselni”.

Az ügyfélfelületnél egy felszínt globálisan kell kiválasztani a `Loader::Customer::SelectedSkin` konfigurációs beállítással. Az összes ügyfél ezt a felszínt fogja látni.

5.2. Hogyan töltődnek be a felszínek

Fontos megjegyezni, hogy *mindig* az „alapértelmezett” felszín fog *először* betöltődni. Ha az ügyintéző egy másik felszínt választott az „alapértelmezett” helyett, akkor a másik felszín csak az alapértelmezett felszín *után* lesz betöltve. A „betöltésen” itt azt értjük, hogy az OTRS olyan címkéket fog elhelyezni a HTML tartalmában, amelyek a CSS-fájlok betöltését idézik elő a böngészőnél. Nézzünk egy példát erre:

```
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css-cache/
CommonCSS_179376764084443c181048401ae0e2ad.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/ivory/css-cache/
CommonCSS_e0783e0c2445ad9cc59c35d6e4629684.css" />
```

Itt tisztán látható, hogy az alapértelmezett felszín töltődik be először, majd ezután az ügyintéző által megadott egyéni felszín. Ebben a példában a bekapcsolt betöltő (a `Loader::Enabled` 1-re állítva) eredményét látjuk, amely begyűjti az összes CSS-fájlt, összefűzi és minimalizálja azokat, majd egyetlen nagy egységként szolgálja ki a böngészőnek. Ezzel sávszélességet spórol, és csökkenti a HTTP-kérések számát is. Nézzük meg ugyanezt a példát kikapcsolt betöltővel:

```
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Reset.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Default.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Header.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.OverviewControl.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.OverviewSmall.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.OverviewMedium.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.OverviewLarge.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Footer.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Grid.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Form.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Table.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Widget.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.WidgetMenu.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.TicketDetail.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Tooltip.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Dialog.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/Core.Print.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/
Core.Agent.CustomerUser.GoogleMaps.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/default/css/
Core.Agent.CustomerUser.OpenTicket.css" />
<link rel="stylesheet" href="/otrs-web/skins/Agent/ivory/css/Core.Dialog.css" />
```

Itt jobban láthatjuk az egyes fájlokat, amelyek a felszínekből jönnek.

Különböző típusú CSS-fájlok vannak: közös fájlok, amelyeket mindig be kell tölteni, és „modul specifikus” fájlok, amelyek csak az OTRS keretrendszeren belüli speciális moduloknál vannak betöltve.

Továbbá lehetséges olyan CSS-fájlok megadása, amelyeket csak IE7 vagy IE8 böngészőknél kell betölteni (az ügyfélfelület esetén IE6 böngészőnél is). Ez szerencsétlen ugyan, de ezeknél a böngészőknél nem volt lehetséges egy modern grafikus felhasználói felület kifejlesztése a hozzájuk elkészített speciális CSS nélkül.

A CSS-fájltípusokra vonatkozó részletekért nézze meg a betöltőről szóló szakaszt.

Minden egyes HTML-oldal előállításához a betöltő először az alapértelmezett felszínből fogja az összes beállított CSS-fájlt venni, és ezután az egyes fájlok kinézetéhez, ha az egy egyéni felszínben is elérhető (ha egy egyéni felszín ki lett választva), majd betölti azokat az alapértelmezett fájlok után.

Ez azt jelenti, hogy az egyéni felszínekben lévő CSS-fájloknak ugyanolyan nevűeknek kell lenniük mint az alapértelmezett felszínekben, és hogy egy egyéni felszínnek nem kell az alapértelmezett felszín összes fájljával rendelkeznie. Ez a nagy előnye az alapértelmezett felszín elsőként való betöltésének: egy egyéni felszínben az összes alapértelmezett CSS-szabály jelen van, és csak azokat szükséges megváltoztatni, amelyeknek eltérő megjelenítést kell eredményezniük. Ez gyakran egyetlen fájllal elvégezhető, mint a fenti példában látható.

Ennek másik hatása, hogy figyelmesnek kell lennie az egyéni felszínekben lévő összes olyan alapértelmezett CSS-szabály felülírásánál, amelyeken változtatni szeretne. Nézzünk egy példát:

```
.Header h1 {
    font-weight: bold;
    color: #000;
}
```

Ez speciális címsorokat határoz meg a `.Header` elemen belül félkövér, fekete szöveggel. Ha most azt szeretné megváltoztatni, hogy a felszínben más színnel és normál szöveggel jelenjen meg, akkor nem elég ezt írni:

```
.Header h1 {
    color: #F00;
}
```

Ugyanis az eredeti `font-weight` szabály még mindig alkalmazva lesz. Határozottan felül kell írnia:

```
.Header h1 {
    font-weight: normal;
    color: #F00;
}
```

5.3. Új felszín létrehozása

Ebben a szakaszban egy új ügyintézői felszínt fogunk létrehozni, amely lecseréli az alapértelmezett (fehér) OTRS háttérszínt egy egyéni (világos szürke) vállalati színre és az alapértelmezett logót egy egyénire. Azt is be fogjuk állítani, hogy ez a felszín legyen az, amelyet az összes ügyintéző alapértelmezetten látni fog.

Csak három egyszerű lépést kell megtennünk a cél eléréséhez:

- a felszínfájlok létrehozását
- az új logó beállítását és
- a felszín megismertetését az OTRS rendszerrel.

Kezdjük az új felszínünkhöz szükséges fájlok létrehozásával. Először is létre kell hoznunk egy új mappát ehhez a felszínhez (ezt `custom` néven fogjuk hívni). Ez a mappa a következő lesz: `$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom`.

Ebben el kell helyeznünk az új CSS-fájlt egy új `css` könyvtárban, amely az új felszín megjelenését fogja meghatározni. Ezt `Core.Default.css` néven fogjuk hívni (emlékezzen arra, hogy ugyanolyan névvel kell rendelkeznie mint az „alapértelmezett” felszínben lévő fájlok egyike). Ez a CSS-fájlhoz szükséges kód:

```
body {
    background-color: #c0c0c0; /* nem túl szép, de a célnak megfelel */
}
```

Most következik a második lépés egy új logó hozzáadásával, és az új felszín megismertetésével az OTRS rendszer számára. Ehhez először el kell helyeznünk az egyéni logónkat (például `logo.png`) egy új könyvtárban (például `img`) a saját felszín könyvtárunkban. Ezután létre kell hoznunk egy új `$OTRS_HOME/Kernel/Config/Files/CustomSkin.xml` beállítófájlt, amely tartalmazni fogja a szükséges beállításokat az alábbiak szerint:

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="1.0" init="Changes">
    <ConfigItem Name="AgentLogo" Required="0" Valid="1">
        <Description Translatable="1">
          Az ügyintézői felület fejlécében megjelenített logó. A képre mutató
          URL-nek a felszín képkönyvtárától relatív URL-nek kell lennie.
        </Description>
        <Group>Framework</Group>
        <SubGroup>Frontend::Agent</SubGroup>
        <Setting>
            <Hash>
                <Item Key="URL">skins/Agent/custom/img/logo.png</Item>
                <Item Key="StyleTop">13px</Item>
                <Item Key="StyleRight">75px</Item>
                <Item Key="StyleHeight">67px</Item>
                <Item Key="StyleWidth">244px</Item>
            </Hash>
        </Setting>
    </ConfigItem>
    <ConfigItem Name="Loader::Agent::Skin###100-custom" Required="0" Valid="1">
        <Description Translatable="1">Egyéni felszín a fejlesztői kézikönyvhöz.</
Description>
        <Group>Framework</Group>
        <SubGroup>Frontend::Agent</SubGroup>
        <Setting>
            <Hash>
                <Item Key="InternalName">custom</Item>
                <Item Key="VisibleName">Egyéni</Item>
                <Item Key="Description">Egyéni felszín a fejlesztői kézikönyvhöz.</Item>
                <Item Key="HomePage">www.azencegem.hu</Item>
            </Hash>
        </Setting>
    </ConfigItem>
</otrs_config>
```

A beállítás aktívvá tételéhez el kell navigálnunk az OTRS adminisztrációs területén lévő rendszerbeállítás modulra (alternatív esetben lefuttathatja az `$OTRS_HOME/bin/otrs.Console.pl Maint::Config::Rebuild` parancsfájlt). Ez újra elő fogja állítani az XML beállítófájlok Perl gyorsítótárát azért, hogy az új felszínünk most már ismert legyen, és kiválasztható legyen a rendszeren. Ennek alapértelmezett felszínné tételéhez, amelyet az új ügyintézők azelőtt láthatnak, mielőtt a saját felszínválasztásukat megtennék, szerkessze a `Loader::Agent::DefaultSelectedSkin` rendszerbeállítási paramétert, és állítsa „Egyéni” értékre.

Következtetésképpen: egy új felszín létrehozásához az OTRS-ben el kellett helyeznünk az új logófájlt, és létre kellett hoznunk egy CSS-fájlt és egy XML-fájlt, amely három új fájlt eredményezett:

```
$OTRS_HOME/Kernel/Config/Files/CustomSkin.xml
$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom/img/custom-logo.png
$OTRS_HOME/var/httpd/htdocs/skins/Agent/custom/css/Core.Header.css
```

6. A CSS és JavaScript „betöltő”

Az OTRS 3.0-val kezdve az OTRS-ben lévő CSS és JavaScript kód hatalmas mennyiségre nőtt. Hogy képes legyen kielégíteni mind a fejlesztői szempontokat (jó karbantarthatóság különálló fájlok nagy mennyiségével), mind a teljesítmény problémákat (kevés HTTP-kérés megtétele és minimalizált tartalom kiszolgálása felesleges üres karakterek és dokumentáció nélkül), foglalkozni kellett ezzel. A célok eléréséhez kitalálták a betöltőt.

6.1. Hogyan működik

Leegyszerűsítve, a betöltő

- minden egyes kérésnél pontosan meghatározza, hogy mely CSS és JavaScript fájlok szükségesek a kliens oldalhoz a jelenlegi alkalmazásmodulnál
- összegyűjti az összes ide vonatkozó adatot
- minimalizálja az adatokat a felesleges üres karakterek és dokumentáció eltávolításával
- kiszolgálja a kliens oldalnak mindössze néhány HTTP-kérésben a sok egyedüli helyett, lehetővé téve a kliensnek, hogy a gyorstárazza ezeket a töredékeket a böngésző gyorsítótárába
- végrehajtja ezeket a feladatokat egy jól teljesítő módon az OTRS gyorsítótárazó mechanizmusait használva.

Természetesen van egy kicsivel részletesebb magyarázat is, de ennek elegendőnek kell lennie első áttekintésként.

6.2. Alapvető működés

A `Loader::Enabled::CSS` és a `Loader::Enabled::JavaScript` konfigurációs beállításokkal kapcsolható be és ki a betöltő a CSS-t és a JavaScriptet illetőleg (alapértelmezetten be van kapcsolva).

Figyelem

Az Internet Explorer böngészőben lévő megjelenítési problémák miatt a betöltőt nem lehet kikapcsolni a CSS-fájlokhoz ennél a kliens böngészőnél (a konfigurációs beállítás felül lesz bírálva). A 8-as verzióig az Internet Explorer nem tud 32 CSS-fájlnál többet kezelni egy oldalon.

Ha többet szeretne megtudni arról, hogy a betöltő hogyan működik, akkor kapcsolja ki az OTRS telepítésében a fent említett konfigurációs beállításokkal. Most nézze meg annak az alkalmazásmodulnak a forráskódját, amelyet jelenleg használ ezen az OTRS rendszeren (természetesen egy újratöltés után). Látni fogja, hogy számos CSS-fájl töltődött be az oldal <head> szakaszában, és sok JavaScript fájl van az oldal alján közvetlenül a lezáró </body> elem előtt.

Ehhez hasonlóan számos egyedülálló fájlban lévő olvashatóan formázott tartalom megléte sokkal egyszerűbbé teszi a fejlesztést, és akár egyáltalán lehetségessé téve azt. Azonban ennek megvan az a hátránya, hogy nagyszámú HTTP-kérést (a hálózati

késleltetésnek nagy hatása van) és felesleges tartalmakat (üres karaktereket és dokumentációt) szükséges átvinni a kliensnek.

A betöltő megoldja ezt a problémát a fenti rövid leírásban felvázolt lépések végrehajtásával. Kapcsolja be ismét a betöltőt, és most töltse újra az oldalt. Most azt láthatja, hogy csak nagyon kevés CSS és JavaScript címke van a HTML kódban ehhez hasonlóan:

```
<script type="text/javascript" src="/otrs30-dev-web/js/js-cache/
CommonJS_d16010491cbd4faaaeb740136a8ecbfd.js"></script>

<script type="text/javascript" src="/otrs30-dev-web/js/js-cache/
ModuleJS_b54ba9c085577ac48745f6849978907c.js"></script>
```

Mi történt most? Ennél az oldalnál a HTML kódot előállító eredeti kérés közben a betöltő előállította ezt a két fájlt (vagy kivette azokat a gyorsítótárból), és betette a látható `<script>` címkébe azon az oldalon, amely ezekhez a fájlokhoz van kapcsolva, ahelyett, hogy az összes ide vonatkozó JavaScript fájlt különállóan kapcsolta volna hozzá (amint a bekapcsolt betöltő nélkül láthatta).

A CSS szakasz egy kicsivel bonyolultabbnak tűnik:

```
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/default/css-
cache/CommonCSS_00753c78c9be7a634c70e914486bfbad.css" />

<!--[if IE 7]>
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/default/css-
cache/CommonCSS_IE7_59394a0516ce2e7359c255a06835d31f.css" />
<![endif]-->

<!--[if IE 8]>
    <link rel="stylesheet" type="text/css" href="/otrs30-dev-web/skins/Agent/default/css-
cache/CommonCSS_IE8_ff58bd010ef0169703062b6001b13ca9.css" />
<![endif]-->
```

Ennek az az oka, hogy az Internet Explorer 7 és 8 esetén speciális bánásmód szükséges az alapértelmezett CSS mellett a webes szabványtechnológiák hiányos támogatásuk miatt. Ezért van néhány normál CSS-fájlunk, amelyek minden böngészőben betöltődnek, és néhány speciális CSS-fájl az úgynevezett „feltételes megjegyzéseken” belül, amely azt idézi elő, hogy *csak* az Internet Explorer 7/8 töltse be azokat. Az összes többi böngésző figyelmen kívül fogja hagyni.

Most felvázoltuk, hogy a betöltő hogyan működik. Nézzük meg, hogy hogyan hasznosíthatja azt a saját OTRS kiterjesztéseiben a betöltőhöz történő konfigurációs adatok hozzáadásával, azt mondva neki, hogy további vagy alternatív CSS vagy JavaScript tartalmat töltsön be.

6.3. A betöltő beállítása: JavaScript

Hogy képes legyen helyesen működni, a betöltőnek tudnia kell, hogy mely tartalmat kell betöltenie egy bizonyos OTRS alkalmazásmodulnál. Először olyan JavaScript fájlokat fog keresni, amelyeket *mindig* be kell tölteni, és ezután keres olyan speciális fájlokat, amelyek csak a jelenlegi alkalmazásmodulnál fontosak.

6.3.1. Közös JavaScript

A betöltendő JavaScript fájlok listája a `Loader::Agent::CommonJS` (az ügyintézői felülethez) és a `Loader::Customer::CommonJS` (az ügyfélfelülethez) konfigurációs beállításokban állítható be.

Ezek a beállítások kivonatokként vannak tervezve azért, hogy az OTRS kiterjesztések hozzáadhassák a saját kivonatkulcsaikat a további betöltendő tartalomhoz. Nézzünk egy példát:

```
<Setting Name="Loader::Agent::CommonJS###000-Framework" Required="1" Valid="1">
    <Description Translatable="1">List of JS files to always be loaded for the agent
 interface.</Description>
    <Navigation>Frontend::Base::Loader</Navigation>
    <Value>
        <Array>
            <Item>thirdparty/jquery-3.2.1/jquery.js</Item>
            <Item>thirdparty/jquery-browser-detection/jquery-browser-detection.js</Item>

            ...

            <Item>Core.Agent.Header.js</Item>
            <Item>Core.UI.Notification.js</Item>
            <Item>Core.Agent.Responsive.js</Item>
        </Array>
    </Value>
</Setting>
```

Ez azon JavaScript fájlok listája, amelyeket mindig be kell tölteni az OTRS ügyintézői felületénél.

Olyan új tartalom hozzáadásához, amelyet mindig be kellene tölteni az ügyintézői felületen, egyszerűen adjon hozzá egy XML beállítófájlt egy másik kivonatbejegyzéssel:

```
<Setting Name="Loader::Agent::CommonJS###000-Framework" Required="1" Valid="1">
    <Description Translatable="1">List of JS files to always be loaded for the agent
 interface.</Description>
    <Navigation>Frontend::Base::Loader</Navigation>
    <Value>
        <Array>
            <Item>thirdparty/jquery-3.2.1/jquery.js</Item>
        </Array>
    </Value>
</Setting>
```

Egyszerű, nemde?

6.3.2. Modulspecifikus JavaScript

Nem minden JavaScript használható az OTRS összes alkalmazásmoduljánál. Ezért lehetséges modulspecifikus JavaScript fájlok megadása. Amikor egy bizonyos modult használnak (mint például AgentDashboard), akkor ennek a modulnak a modulspecifikus JavaScript fájlja is be lesz töltve. A beállítás az XML beállításokban lévő előtétprogram modul regisztrációban történik. Ismét egy példa:

```
<Setting Name="Loader::Module::AgentDashboard###001-Framework" Required="0" Valid="1">
    <Description Translatable="1">Loader module registration for the agent interface.</
Description>
    <Navigation>Frontend::Agent::ModuleRegistration::Loader</Navigation>
    <Value>
        <Hash>
            <Item Key="CSS">
                <Array>
                    <Item>Core.Agent.Dashboard.css</Item>

                    ...

                </Array>
            </Item>
```

```
            <Item Key="JavaScript">
                <Array>
                    <Item>thirdparty/momentjs-2.18.1/moment.min.js</Item>
                    <Item>thirdparty/fullcalendar-3.4.0/fullcalendar.min.js</Item>
                    <Item>thirdparty/d3-3.5.6/d3.min.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/nvd3.min.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSLineChart.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSMultiBarChart.js</Item>
                    <Item>thirdparty/nvd3-1.7.1/models/OTRSStackedAreaChart.js</Item>
                    <Item>thirdparty/canvg-1.4/rgbcolor.js</Item>
                </Array>
            </Item>
        </Hash>
    </Value>
</Setting>
```

It is possible to put a `<Item Key="JavaScript">` tag in the frontend module registrations which may contain `<Array>` and one tag `<Item>` for each JavaScript file that is supposed to be loaded for this application module.

Most már rendelkezik az összes olyan információval, amely annak a módszernek a beállításhoz szükséges, hogy a betöltő kezelje a JavaScript kódot.

6.4. A betöltő beállítása: CSS

A betöltő a CSS-fájlokat nagyon hasonlóan kezeli a JavaScript fájlokhoz, ahogy az előző szakaszban le van írva, és a beállítások kiterjesztése is ugyanolyan módon működik.

6.4.1. Közös CSS

The way common CSS is handled is very similar to the way common JavaScript is loaded.

7. Sablonozó mechanizmus

Belsőleg az OTRS egy sablonozó mechanizmust használ a HTML oldalak (és egyéb tartalom) dinamikus előállításához, miközben szétválasztva tartja a program logikáját (Perl) és a megjelenítést (HTML). Tipikusan egy előtétprogram modul egy saját sablonfájlt fog használni, át fog adni néhány adatot annak, és vissza fogja adni a megjelenített eredményt a felhasználónak.

A sablonfájlok itt találhatók: `$OTRS_HOME/Kernel/Output/HTML/Standard/*.tt`

Az OTRS a Template::Toolkit megjelenítő motorra támaszkodik. A teljes Template::Toolkit szintaxis használható az OTRS sablonokban. Ez a szakasz néhány példa használati esetet és OTRS kiterjesztést mutat be a Template::Toolkit szintaxishoz.

7.1. Sablonparancsok

7.1.1. Dinamikus adatok beszúrása

A sablonokba dinamikus adatokat kell beszúrni, idézni, stb. Ez a szakasz sorolja fel a fontos parancsokat ennek elvégzéséhez.

7.1.1.1. `[% Data.Name %]`

Ha az alkalmazásmodul adatparamétereket ad meg a sablonoknak, akkor ezeket az adatokat ki lehet íratni a sablonra. A `[% Data.Name %]` a legegyszerűbb, de a legveszélyesebb is. Azt az adatparamétert fogja további feldolgozás nélkül beszúrni a sablonba úgy ahogy van, amely neve `Name`.

Figyelem

A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet. Sose írasson ki olyan adatokat, amelyeket egy felhasználó adott meg, anélkül, hogy idézné azokat a HTML környezetben. A felhasználó például egyszerűen beszúrhat egy `<script>` címkét, és az kiíródhat az OTRS által előállított HTML oldalon.

Amikor csak lehetséges, használjon `[% Data.Name | html %]` (HTML-ben) vagy `[% Data.Name | uri %]` (hivatkozásokban) paramétert helyette.

Példa: Amikor HTML kimenetet állítunk elő az alkalmazásban, akkor HTML idézés nélkül kell kiíratnunk azt a sablonba, mint például a `<select>` elemeket, amelyeket a `Layout::BuildSelection()` függvény állít elő az OTRS-ben.

```
<label for="Dropdown">Példa legördülő</label>
[% Data.DropdownString"]
```

Ha speciális karaktereket tartalmazó, összetett nevű adatbejegyzései vannak, akkor nem használhatja a pont (`.`) jelölést az adathoz való hozzáféréshez. Az `item()` függvényt használja helyette: `[% Data.item('Összetett-név') %]`.

7.1.1.2. [% Data.Name | html %]

Ennek a parancsnak ugyanaz a funkciója mint az előzőnek, de HTML idézést hajt végre az adatokon, amint beszúrásra kerülnek a sablonba.

```
A szerző neve [% Data.Name | html %].
```

Lehetséges az érték legnagyobb hosszának megadása is. Ha például egy változónak csak 9 karakterét szeretné megjeleníteni (az eredmény „ValamiNév[...]” lesz), akkor használja a következőt:

```
A szerző nevének első 20 karaktere: [% Data.Name | truncate(20) | html %].
```

7.1.1.3. [% Data.Name | uri %]

Ez a parancs URL-kódolást hajt végre az adatokon, amint az beszúrásra kerül a sablonba. Ezt kell használni az URL-ek egyedülálló paraméternevei vagy értékei kiíratásánál a biztonsági problémák megakadályozásához. Nem használható teljes URL-eknél, mert ki fogja maszkolni például az = karaktert is.

```
<a href="[% Env("Baselink") %];Location=[% Data.File | uri %]">[% Data.File | truncate(110)
 | html %]</a>
```

7.1.1.4. [% Data.Name | JSON %]

Ez a parancs JavaScript JSON szövegként írat ki egy szöveget vagy más adatszerkezetet.

```
var Text = [% Data.Text | JSON %];
```

Vegye figyelembe, hogy a szűrőjelölés csak egyszerű szövegeknél fog működni. Összetett adatok JSON szövegként való kiíratásához függvényként használja azt:

```
var TreeData = [% JSON(Data.TreeData) %];
```

7.1.1.5. [% Env() %]

A LayoutObject által szolgáltatott környezeti változókat szúrja be. Néhány példa:

```
A jelenlegi felhasználó neve: [% Env("UserFullname") %]

Néhány egyéb gyakori előre meghatározott változó:

[% Env("Action") %] --> a jelenlegi művelet
[% Env("Baselink") %] --> az alaphivatkozás --> index.pl?SessionID=...
[% Env("CGIHandle") %] --> a jelenlegi CGI-kezelő, például index.pl
[% Env("SessionID") %] --> a jelenlegi munkamenet-azonosító
[% Env("Time") %] --> a jelenlegi idő, például Thu Dec 27 16:00:55 2001
[% Env("UserFullname") %] --> például Kovács János
[% Env("UserIsGroup[admin]") %] = Igen
[% Env("UserIsGroup[users]") %] = Igen --> felhasználócsoportok (hasznos saját
 hivatkozásoknál)
[% Env("UserLogin") %] --> például mgg@x11.org
```

Figyelem

A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet. Sose írasson ki olyan adatokat, amelyeket egy felhasználó adott meg, anélkül, hogy idézné azokat a HTML környezetben. A felhasználó például egyszerűen beszúrhat egy <script> címkét, és az kiíródhat az OTRS által előállított HTML oldalon.

Ne felejtse el a | html szűrőt hozzáadni, ahol az helyénvaló.

7.1.1.6. [% Config() %]

Beállítási változókat szúr be a sablonba. Nézzünk egy példa Kernel/Config.pm fájlt:

```
[Kernel/Config.pm]
    # FQDN
    # (A rendszer teljesen minősített tartományneve.)
    $Self->{FQDN} = 'otrs.example.com';
    # AdminEmail
    # (A rendszer adminisztrátorának e-mail címe.)
    $Self->{AdminEmail} = 'admin@example.com';
[...]
```

Változók kiíratásához ebből fájlból a sablonba a következőt használja:

```
A gépnév „$Config{"FQDN"}”
Az adminisztrátori e-mail cím „[% Config("AdminEmail") %]”
```

Figyelem

A hiányzó HTML idézés miatt ez biztonsági problémákat eredményezhet.

Ne felejtse el a | html szűrőt hozzáadni, ahol az helyénvaló.

7.1.2. Honosítási parancsok

7.1.2.1. [% Translate() %]

Lefordít egy szöveget a felhasználó által kiválasztott jelenlegi nyelve. Ha nem található fordítás, akkor az eredeti szöveget fogja használni.

```
Ezen szöveg lefordítása: [% Translate("Help") | html %]
```

Lefordíthat dinamikus adatokat is a `Translate` szűrőként való használatával:

```
Adatok lefordítása az alkalmazásból: [% Data.Type | Translate | html %]
```

Egy vagy több paramétert (%s) is megadhat a szövegen belül, amelyeket dinamikus adatokkal kell kicserélni:

```
Ezen szöveg lefordítása és a megadott adatok beszúrása: [% Translate("Change %s settings",
 Data.Type) | html %]
```

A JavaScriptben lévő szövegek is lefordíthatók és feldolgozhatók a JSON szűrővel.

```
var Text = [% Translate("Change %s settings", Data.Type) | JSON %];
```

7.1.2.2. [% Localize() %]

Outputs data according to the current language/locale.

Különböző kulturális területeken különböző egyezményt használnak a dátum és idő formázásához. Például ami Németországban 01.02.2010 formátum, annak az USA-ban 02/01/2010 formátumban kellene lennie. A [% Localize() %] függvény elvonatkoztatja ezt a sablontól. Nézzünk egy példát:

```
[% Data.CreateTime | Localize("TimeLong") %]
# Eredmény a US English területi beállításnál:
06/09/2010 15:45:41
```

Először is az adatok a Data segítségével kerülnek beszúrásra az alkalmazásmodulból. Itt mindig egy ISO UTC időbéleget (2010-06-09 15:45:41) kell átadni adatként a [% Localize() %] függvénynek. Ezután lesz kiíratva a jelenlegi területi beállítás dátum és idő meghatározása szerint.

A [% Localize() %] függvénynek átadott adatoknak UTC formátumban kell lenniük. Ha időzóna-eltolás van meghatározva a jelenlegi ügyintézőnél, akkor az alkalmazva lesz az UTC időbélyegen a kimenet előállítása előtt.

There are different possible date/time output formats: `TimeLong` (full date/time), `TimeShort` (no seconds) and `Date` (no time).

```
[% Data.CreateTime | Localize("TimeLong") %]
# Result for US English locale:
06/09/2010 15:45:41

[% Data.CreateTime | Localize("TimeShort") %]
# Result for US English locale:
06/09/2010 15:45

[% Data.CreateTime | Localize("Date") %]
# Result for US English locale:
06/09/2010
```

Also the output of human readable file sizes is available as an option `Localize('Filesize')` (just pass the raw file size in bytes).

```
[% Data.Filesize | Localize("Filesize") %]
# Result for US English locale:
23 MB
```

7.1.2.3. [% ReplacePlaceholders() %]

Replaces placeholders (%s) in strings with passed parameters.

In certain cases, you might want to insert HTML code in translations, instead of placeholders. On the other hand, you also need to take care of sanitization, since translated strings should not be trusted as-is. For this, you can first translate the string, pass it through the HTML filter and finally replace placeholders with static (safe) HTML code.

```
[% Translate("This is %s.") | html | ReplacePlaceholders('<strong>bold text</strong>') %]
```

Number of parameters to ReplacePlaceholders() filter should match number of placeholders in the original string.

You can also use [% ReplacePlaceholders() %] in function format, in case you are not translating anything. In this case, first parameter is the target string, and any found placeholders in it are substituted with subsequent parameters.

```
[% ReplacePlaceholders("This string has both %s and %s.", '<strong>bold text</strong>,
 '<em>italic text</em>') %]
```

7.1.3. Sablonfeldolgozó parancsok

7.1.3.1. Megjegyzés

Azok a sorok, amelyek # karakterrel kezdődnek az elején, nem lesznek láthatók a HTML kimeneten. Ez használható a sablonkód magyarázatához, vagy annak egyes részei letiltásához is.

```
# ez a szakasz átmenetileg le van tiltva
# <div class="AsBlock">
#     <a href="...">hivatkozás</a>
# </div>
```

7.1.3.2. [% InsertTemplate("Copyright.tt") %]

Figyelem

Felhívjuk a figyelmét, hogy az InsertTemplate parancs azért lett hozzáadva, hogy jobb visszafelé kompatibilitást nyújtson a régi DTL rendszerhez. Ez esetleg elavulttá válhat az OTRS jövőbeli verzióiban, és később eltávolításra kerülhet. Ha nem használ blokk parancsokat a felvett sablonjában, akkor nincs szüksége az InsertTemplate parancsra, és használhatja helyette a szabványos Template::Toolkit szintaxist, úgymint INCLUDE/PROCESS.

Felvesz egy másik sablonfájlt a jelenlegibe. A felvett fájl is tartalmazhat sablonparancsokat.

```
# a Copyright.tt felvétele
[% InsertTemplate("Copyright") %]
```

Felhívjuk a figyelmét, hogy ez nem ugyanaz mint a Template::Toolkit `[% INCLUDE %]` parancsa, amely csak feldolgozza a hivatkozott sablont. Az `[% InsertTemplate() %]` tulajdonképpen hozzáadja a hivatkozott sablon tartalmát a jelenlegi sablonhoz azért, hogy együtt legyenek feldolgozhatók. Ez lehetővé teszi a beágyazott sablon számára, hogy ugyanazon környezethez vagy adatokhoz férjen hozzá mint a fő sablon.

7.1.3.3. `[% RenderBlockStart %]` / `[% RenderBlockEnd %]`

Figyelem

Vegye figyelembe, hogy a blokk parancsok azért lettek hozzáadva, hogy jobb visszafelé kompatibilitást nyújtsanak a régi DTL rendszerhez. Ezek esetleg elavulttá válhatnak az OTRS jövőbeli verzióiban, és később eltávolításra kerülhetnek. Azt javasoljuk, hogy blokk parancsok használata nélkül fejlesszen bármilyen új kódot. Használhatja a szabványos Template::Toolkit szintaxist a feltételes sablonkimenethez, mint például `IF/ELSE`, `LOOP`-ok és egyéb hasznos dolgok.

Ezzel a paranccsal lehet megadni egy sablonfájl részeit blokként. Ezt a blokkot határozottan ki kell tölteni egy függvényhívással az alkalmazásból, hogy megjelenjen az előállított kimeneten. Az alkalmazás meghívhatja a blokkot 0-szor (nem fog megjelenni a kimeneten), illetve 1 vagy többször (esetleg mindegyiket a sablonnak átadott adatparaméterek különböző halmazával).

Egy gyakori használati eset egy táblázat kitöltése dinamikus adatokkal:

```
<table class="DataTable">
    <thead>
        <tr>
            <th>[% Translate("Name") | html %]</th>
            <th>[% Translate("Type") | html %]</th>
            <th>[% Translate("Comment") | html %]</th>
            <th>[% Translate("Validity") | html %]</th>
            <th>[% Translate("Changed") | html %]</th>
            <th>[% Translate("Created") | html %]</th>
        </tr>
    </thead>
    <tbody>
[% RenderBlockStart("NoDataFoundMsg") %]
        <tr>
            <td colspan="6">
                [% Translate("No data found.") | html %]
            </td>
        </tr>
[% RenderBlockEnd("NoDataFoundMsg") %]
[% RenderBlockStart("OverviewResultRow") %]
        <tr>
            <td><a class="AsBlock" href="[% Env("Baselink") %]Action=[% Env("Action")
 %];Subaction=Change;ID=[% Data.ID | uri %]">[% Data.Name | html %]</a></td>
            <td>[% Translate(Data.TypeName) | html %]</td>
            <td title="[% Data.Comment | html %]">[% Data.Comment | truncate(20) | html %]</
td>
            <td>[% Translate(Data.Valid) | html %]</td>
            <td>[% Data.ChangeTime | Localize("TimeShort") %]</td>
            <td>[% Data.CreateTime | Localize("TimeShort") %]</td>
        </tr>
[% RenderBlockEnd("OverviewResultRow") %]
    </tbody>
</table>
```

A körülvevő táblázat a fejléccel mindig elő lesz állítva. Ha nem található adat, akkor a `NoDataFoundMsg` blokk egyszer lesz meghívva egy olyan táblázatot eredményezve, amelynek egy adatsora van a „Nem található adat.” üzenettel.

Ha találhatók adatok, akkor minden egyes sornál egy függvényhívás történik az `OverViewResultRow` blokknál (minden alkalommal átadva az adatokat ehhez a bizonyos sorhoz) egy olyan táblázatot eredményezve, amelynek annyi sora van, ahány eredmény található.

Nézzük meg, hogyan vannak meghívva a blokkok az alkalmazásmodulból:

```
my %List = $Kernel::OM->Get('Kernel::System::State)->StateList(
    UserID => 1,
    Valid  => 0,
);

# ha van bármilyen állapot, akkor azok meg fognak jelenni
if (%List) {

    # érvényes lista beszerzése
    my %ValidList = $Kernel::OM->Get('Kernel::System::Valid')->ValidList();
    for my $ListKey ( sort { $List{$a} cmp $List{$b} } keys %List ) {

        my %Data = $Kernel::OM->Get('Kernel::System::State)->StateGet( ID => $ListKey );
        $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Block(
            Name => 'OverviewResultRow',
            Data => {
                Valid => $ValidList{ $Data{ValidID} },
                %Data,
            },
        );
    }
}

# egyébként egy „Nem található adat” üzenet jelenik meg
else {
    $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Block(
        Name => 'NoDataFoundMsg',
        Data => {},
    );
}
```

Figyelje meg, hogy a blokkoknak hogyan kell átadniuk mind a nevüket, mind egy opcionális adathalmazt különálló paraméterekként a blokkfüggvény hívásnak. Az adatbeszúró parancsoknak egy blokkon belül mindig az ezen blokk blokkfüggvény hívásához megadott adatokra van szükségük, nem az általános sablonmegjelenítő híváshoz.

A részletekért nézze meg a `Kernel::Output::HTML::Layout` dokumentációját az otrs.github.io/doc oldalon.

7.1.4. `[% WRAPPER JSOnDocumentComplete %]...[% END %]`

Megjelöli azt a JavaScript kódot, amelyet azután kell lefuttatni, miután az összes CSS, JavaScript és egyéb külső tartalom betöltődött, és az alapvető JavaScript előkészítés befejeződött. Vessünk egy pillantást ismét egy példára:

```
<form action="[% Env("CGIHandle") %]" method="post" enctype="multipart/form-data"
 name="MoveTicketToQueue" class="Validate PreventMultipleSubmits" id="MoveTicketToQueue">
    <input type="hidden" name="Action"       value="[% Env("Action") %]"/>
    <input type="hidden" name="Subaction"    value="MoveTicket"/>

    ...

    <div class="Content">
        <fieldset class="TableLike FixedLabel">
            <label class="Mandatory" for="DestQueueID"><span class="Marker">*</span> [%
 Translate("New Queue") | html %]:</label>
            <div class="Field">
```

```
                [% Data.MoveQueuesStrg %]
                <div id="DestQueueIDError" class="TooltipErrorMessage" ><p>[%
 Translate("This field is required.") | html %]</p></div>
                <div id="DestQueueIDServerError" class="TooltipErrorMessage"><p>[%
 Translate("This field is required.") | html %]</p></div>
[% WRAPPER JSOnDocumentComplete %]
<script type="text/javascript">
    $('#DestQueueID').bind('change', function (Event) {
        $('#NoSubmit').val('1');
        Core.AJAX.FormUpdate($('#MoveTicketToQueue'), 'AJAXUpdate', 'DestQueueID',
 ['NewUserID', 'OldUserID', 'NewStateID', 'NewPriorityID' [% Data.DynamicFieldNamesStrg
 %]]);
    });
</script>
[% END %]
                    </div>
                    <div class="Clear"></div>
```

Ez a kódrészlet egy kicsi űrlapot hoz létre, és rátesz egy onchange kezelőt a `<select>` elemre, amely aktivál egy AJAX-alapú űrlapfrissítést.

Miért van szükség a JavaScript kód körbezárására a `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkban? Az OTRS 3.0-tól kezdve a JavaScript betöltést teljesítmény okok miatt áthelyezték az oldal lábrészébe. Ez azt jelenti, hogy az oldal `<body>` részén belül még nincsenek JavaScript programkönyvtárak betöltve. A `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkal lehet biztos abban, hogy ez a JavaScript áthelyezésre kerül a végső HTML dokumentumnak egy olyan részébe, ahol csak akkor kerül végrehajtásra, miután a teljes külső JavaScript és CSS tartalom sikeresen be lett töltve és elő lett készítve.

A `[% WRAPPER JSOnDocumentComplete %]...[% END %]` blokkon belül használhatja a `<script>` címkéket a JavaScript kód körbezárásához, de ezt nem kell megtennie. Előnyös lehet, mert engedélyezni fogja a helyes szintaxis-kiemelést az olyan integrált fejlesztői környezetekben, amelyek támogatják azt.

7.2. Egy sablonfájl használata

Rendben, de tulajdonképpen hogyan kell egy sablonfájlt feldolgozni és az eredményt előállítani? Ez igazán egyszerű:

```
# render AdminState.tt
$Output .= $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Output(
    TemplateFile => 'AdminState',
    Data         => \%Param,
);
```

Az előtétprogram modulokban a `Kernel::Output::HTML::Layout` objektum `Output()` függvénye lesz meghívva (miután az összes szükséges blokk meg lett hívva ebben a sablonban) a végső kimenet előállításához. Adatparaméterek opcionális halmaza is átadásra kerül a sablonnak minden olyan adatbeszúró parancsnál, amelyek nincsenek egy blokk belsejében.

8. Saját témák létrehozása

You can create your own themes so as to use the layout you like in the OTRS web frontend. To create custom themes, you should customize the output templates to your needs. More information on the syntax and structure of output templates can be found in the templating section.

Példaként hajtsa végre a következő lépéseket egy új „Vállalat” nevű téma létrehozásához:

1. Create a directory called `Kernel/Output/HTML/Templates/Company` and copy all files that you like to change from `Kernel/Output/HTML/Templates/Standard` into the new folder.

   Fontos

   Only copy over the files you're planning to change. OTRS will automatically get the missing files from the Standard theme. This will make upgrading at a later stage much easier.

2. Customize the files in the directory `Kernel/Output/HTML/Templates/Company` and change the layout to your needs.

3. To activate the new theme, add them in SysConfig under Frontend::Themes.

Now the new theme should be usable. You can select it via your personal preferences.

Figyelem

Ne változtassa meg az OTRS-sel szállított témafájlokat, mivel azok a változtatások el fognak veszni egy frissítés után. Csak a fent leírt lépések végrehajtásával hozzon létre saját témákat.

9. Honosítási és fordítási mechanizmus

Négy lépés szükséges a szoftver lefordításához és honosításához: a honosítható szövegek megjelölése a forrásfájlokban, a fordítási adatbázis/fájl előállítása, maga a fordítási folyamat, és a lefordított adatok használata a kódon belül.

9.1. Lefordítható szövegek megjelölése a forrásfájlokban

A Perl-kódban az összes lefordítandó literál szöveg automatikusan meg van jelölve a fordításhoz: a `$LanguageObject->Translate('My string %s', $Data)` fogja megjelölni a 'My string %s' szöveget a fordításhoz. Ha arra van szüksége, hogy a kódban megjelölje a szövegeket, de még NE fordítsa le azokat, akkor használhatja a `Kernel::Langauge::Translatable()` NOOP metódust.

```
package MyPackage;

use strict;
use warnings;

use Kernel::Language (qw(Translatable));

...

my $UntranslatedString = Translatable('My string %s');
```

A sablonfájlokban az összes olyan literál szöveg automatikusan meg van jelölve a kigyűjtéshez, amelyek a `Translate()`-címkével vannak körbezárva: `[% Translate('My string %s', Data.Data )%]`.

A rendszerbeállítás és az adatbázis XML-fájlokban a `Translatable="1"` attribútummal jelölheti meg a szövegeket a kigyűjtéshez.

```
# Adatbázis XML
<Insert Table="groups">
    <Data Key="id" Type="AutoIncrement">1</Data>
    ...
    <Data Key="comments" Type="Quote" Translatable="1">Group for default access.</Data>
    ...
</Insert>

# Rendszerbeállítás XML
<Setting>
    <Option SelectedID="0">
        <Item Key="0" Translatable="1">No</Item>
        <Item Key="1" Translatable="1">Yes</Item>
    </Option>
</Setting>
```

9.2. Lefordítható szövegek összegyűjtése a fordítási adatbázisba

Az `otrs.Console.pl Dev::Tools::TranslationsUpdate` konzolparancs használható az összes lefordítható szöveg kigyűjtéséhez a forrásfájlokból. Ezek össze lesznek gyűjtve, és ki lesznek írva a fordítási fájlokba.

Az OTRS keretrendszernél és az összes olyan kiterjesztés modulnál, amelyek szintén a Transifex szolgáltatását használják a fordítások kezeléséhez, .pot és .po fájlok lesznek kiírva. Ezeket a fájlokat használják a lefordítható szövegek feltöltéséhez a Transifexre, illetve a fordítások letöltéséhez onnan.

De az OTRS-nek sebességi okok miatt a fordításokra Perl-fájlokban van szüksége. Az `otrs.Console.pl Dev::Tools::TranslationsUpdate` parancs ezeket a fájlokat is elő fogja állítani. Két különböző fordítási gyorsítótár fájltípus létezik, amelyek a következő sorrendben kerülnek felhasználásra. Ha egy szó vagy mondat újra meg van adva egy fordítási fájlban, akkor a legutolsó meghatározást fogja használni.

1. Alapértelmezett keretrendszer fordítási fájl

   `Kernel/Language/$Language.pm`

2. Egyéni fordítási fájl

   `Kernel/Language/$Language_Custom.pm`

9.2.1. Alapértelmezett keretrendszer fordítási fájl

Az alapértelmezett keretrendszer fordítási fájl tartalmazza az alapvető fordításokat. Az alábbi egy alapértelmezett keretrendszer fordítási fájl példája.

Formátum:

```
package Kernel::Language::hu;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub Data {
    my $Self = shift;

    # $$START$$
```

```
    # lehetséges karakterkészletek
    $Self->{Charset} = ['utf-8', 'iso-8859-2', ];
    # date formats (%A=WeekDay;%B=LongMonth;%T=Time;%D=Day;%M=Month;%Y=Jear;)
    $Self->{DateFormat}          = '%Y-%M-%D %T';
    $Self->{DateFormatLong}      = '%Y. %B %D. %A %T';
    $Self->{DateFormatShort}     = '%Y-%M-%D';
    $Self->{DateInputFormat}     = '%Y-%M-%D';
    $Self->{DateInputFormatLong} = '%Y-%M-%D - %T';

    $Self->{Translation} = {
    # Template: AAABase
    'Yes' => 'Igen',
    'No' => 'Nem',
    'yes' => 'igen',
    'no' => 'nem',
    'Off' => 'Ki',
    'off' => 'ki',
    };
    # $$STOP$$
    return 1;
}

1;
```

9.2.2. Egyéni fordítási fájl

Az egyéni fordítási fájl kerül beolvasásra legutoljára, és így annak fordítása, amely használva lesz. Ha saját megfogalmazást szeretne hozzáadni a telepítéshez, akkor hozza létre ezt a fájlt a nyelvéhez.

Formátum:

```
package Kernel::Language::xx_Custom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub Data {
    my $Self = shift;

    # $$START$$

    # saját fordítások
    $Self->{Translation}->{'Lock'} = 'Lala';
    $Self->{Translation}->{'Unlock'} = 'Lulu';

    # $$STOP$$
    return 1;
}

1;
```

9.3. Maga a fordítási folyamat

Az OTRS a Transifex szolgáltatását használja a fordítási folyamat kezeléséhez. A részletekért nézze meg ezt a szakaszt.

9.4. A kódból lefordított adatok használata

Használhatja a `$LanguageObject->Translate()` metódust a szövegek lefordításához futási időben a Perl-kódból, és a `Translate()`-címkét a sablonokban.

3. fejezet - Hogyan bővíthető az OTRS

1. Egy új OTRS előtétprogram modul írása

Ebben a fejezetben egy új OTRS modul írása van szemléltetve egy egyszerű kis program alapján. A szükséges előkövetelmény egy olyan OTRS fejlesztői környezet, amely a hasonló nevű fejezetben van megadva.

1.1. Mit szeretnénk írni

Szeretnénk írni egy olyan kis OTRS modult, amely a „Hello World” szöveget jeleníti meg, amikor előhívják. Mindenek előtt fel kell építenünk a `/Hello World` könyvtárat a modulhoz a fejlesztői könyvtárban. Ebben a könyvtárban létrehozható az OTRS-ben meglévő összes könyvtár. Minden modulnak legalább a következő könyvtárakat kell tartalmaznia:

```
Kernel
Kernel/System
Kernel/Modules
Kernel/Output/HTML/Templates/Standard
Kernel/Config
Kernel/Config/Files
Kernel/Config/Files/XML/
Kernel/Language
```

1.2. Alapértelmezett beállítófájl

The creation of a module registration facilitates the display of the new module in OTRS. Therefore we create a file `/Kernel/Config/Files/XML/HelloWorld.xml`. In this file, we create a new config element. The impact of the various settings is described in the chapter 'Config Mechanism'.

```
<?xml version="1.0" encoding="UTF-8" ?>
<otrs_config version="2.0" init="Application">
    <Setting Name="Frontend::Module###AgentHelloWorld" Required="1" Valid="1">
        <Description Translatable="1">FrontendModuleRegistration for HelloWorld module.</
Description>
        <Navigation>Frontend::Agent::ModuleRegistration</Navigation>
        <Value>
            <Item ValueType="FrontendRegistration">
                <Hash>
                    <Item Key="Group">
                        <Array>
                            <Item>users</Item>
                        </Array>
                    </Item>
                    <Item Key="GroupRo">
                        <Array>
                        </Array>
                    </Item>
                    <Item Key="Description" Translatable="1">HelloWorld.</Item>
                    <Item Key="Title" Translatable="1">HelloWorld</Item>
                    <Item Key="NavBarName">HelloWorld</Item>
                </Hash>
            </Item>
        </Value>
    </Setting>
```

```
    <Setting Name="Loader::Module::AgentHelloWorld###002-Filename" Required="0" Valid="1">
        <Description Translatable="1">Loader module registration for the agent interface.</
Description>
        <Navigation>Frontend::Agent::ModuleRegistration::Loader</Navigation>
        <Value>
            <Hash>
                <Item Key="CSS">
                    <Array>
                    </Array>
                </Item>
                <Item Key="JavaScript">
                    <Array>
                    </Array>
                </Item>
            </Hash>
        </Value>
    </Setting>
    <Setting Name="Frontend::Navigation###AgentHelloWorld###002-Filename" Required="0"
 Valid="1">
        <Description Translatable="1">Main menu item registration.</Description>
        <Navigation>Frontend::Agent::ModuleRegistration::MainMenu</Navigation>
        <Value>
            <Array>
                <DefaultItem ValueType="FrontendNavigation">
                    <Hash>
                    </Hash>
                </DefaultItem>
                <Item>
                    <Hash>
                        <Item Key="Group">
                            <Array>
                                <Item>users</Item>
                            </Array>
                        </Item>
                        <Item Key="GroupRo">
                            <Array>
                            </Array>
                        </Item>
                        <Item Key="Description" Translatable="1">HelloWorld.</Item>
                        <Item Key="Name" Translatable="1">HelloWorld</Item>
                        <Item Key="Link">Action=AgentHelloWorld</Item>
                        <Item Key="LinkOption"></Item>
                        <Item Key="NavBar">HelloWorld</Item>
                        <Item Key="Type">Menu</Item>
                        <Item Key="Block"></Item>
                        <Item Key="AccessKey"></Item>
                        <Item Key="Prio">8400</Item>
                    </Hash>
                </Item>
            </Array>
        </Value>
    </Setting>
</otrs_config>
```

1.3. Előtétprogram modul

A hivatkozások létrehozása és a rendszerbeállítások végrehajtása után megjelenik egy új modul „HelloWorld” néven. Előhívásakor egy hibaüzenet jelenik meg, mivel az OTRS még nem találja a hozzá tartozó előtétprogram modult. Ez a következő dolog, amit létre kell hozni. Ehhez hozzuk létre az alábbi fájlt:

```
# --
# Kernel/Modules/AgentHelloWorld.pm - frontend module
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
```

```
# --

package Kernel::Modules::AgentHelloWorld;

use strict;
use warnings;

# Frontend modules are not handled by the ObjectManager.
our $ObjectManagerDisabled = 1;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless ($Self, $Type);

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;
    my %Data = ();

    my $HelloWorldObject = $Kernel::OM->Get('Kernel::System::HelloWorld');
    my $LayoutObject     = $Kernel::OM->Get('Kernel::Output::HTML::Layout');

    $Data{HelloWorldText} = $HelloWorldObject->GetHelloWorldText();

    # build output
    my $Output = $LayoutObject->Header(Title => "HelloWorld");
    $Output   .= $LayoutObject->NavigationBar();
    $Output   .= $LayoutObject->Output(
        Data         => \%Data,
        TemplateFile => 'AgentHelloWorld',
    );
    $Output   .= $LayoutObject->Footer();

    return $Output;
}

1;
```

1.4. Alapmodul

Ezután hozzunk létre egy fájlt a /HelloWorld/Kernel/System/HelloWorld.pm alapmodulhoz az alábbi tartalommal:

```
# --
# Kernel/System/HelloWorld.pm - core module
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::HelloWorld;

use strict;
use warnings;

# list your object dependencies (e.g. Kernel::System::DB) here
our @ObjectDependencies = (
    # 'Kernel::System::DB',
);

=head1 NAME
```

```
HelloWorld - Little "Hello World" module

=head1 DESCRIPTION

Little OTRS module that displays the text 'Hello World' when called up.

=head2 new()

Create an object. Do not use it directly, instead use:

    my $HelloWorldObject = $Kernel::OM->Get('Kernel::System::HelloWorld');

=cut

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless ($Self, $Type);

    return $Self;
}

=head2 GetHelloWorldText()

Return the "Hello World" text.

    my $HelloWorldText = $HelloWorldObject->GetHelloWorldText();

=cut

sub GetHelloWorldText {
    my ( $Self, %Param ) = @_;

    # Get the DBObject from the central object manager
    # my $DBObject = $Kernel::OM->Get('Kernel::System::DB');

    my $HelloWorld = $Self->_FormatHelloWorldText(
        String => 'Hello World',
    );

    return $HelloWorld;
}

=begin Internal:

=cut

=head2 _FormatHelloWorldText()

Format "Hello World" text to uppercase

    my $HelloWorld = $Self->_FormatHelloWorldText(
        String => 'Hello World',
    );

sub _FormatHelloWorldText{
    my ( $Self, %Param ) = @_;

    my $HelloWorld = uc $Param{String};

    return $HelloWorld;

}

=end Internal:

1;
```

1.5. Sablonfájl

Az utolsó hiányzó dolog, mielőtt az új modul futtatható lenne, a hozzá tartozó HTML sablon. Ezért hozzuk létre az alábbi fájlt:

```
# --
# Kernel/Output/HTML/Templates/Standard/AgentHelloWorld.tt - overview
# Copyright (C) (year) (name of author) (email of author)
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
<h1>[% Translate("Overview") | html %]: [% Translate("HelloWorld") %]</h1>
<p>
    [% Data.HelloWorldText | Translate() | html %]
</p>
```

A modul most már működik, és a meghívásakor megjeleníti a „Hello World” szöveget.

1.6. Nyelvi fájl

Ha a „Hello World!” szöveget le kell fordítani például magyarra, akkor létrehozhat egy fordítási fájlt ehhez a nyelvhez a `HelloWorld/Kernel/Language/hu_AgentHelloWorld.pm` helyre. Példa:

```
package Kernel::Language::hu_AgentHelloWorld;

use strict;
use warnings;

sub Data {
    my $Self = shift;

    $Self->{Translation}->{'Hello World!'} = 'Helló, Világ!';

    return 1;
}
1;
```

1.7. Összefoglaló

A fent megadott példa azt mutatja be, hogy nem túl bonyolult egy új modult írni az OTRS-hez. Fontos azonban meggyőződni arról, hogy a modul és a fájlnév egyedi legyen, és ennélfogva ne ütközzenek a keretrendszerrel vagy egyéb kiterjesztő modulokkal. Amikor egy modul elkészült, akkor egy OPM csomagot kell előállítani belőle (lásd a csomagkészítés fejezetet).

2. Az OTRS modulrétegek erejének használata

Az OTRS nagyszámú úgynevezett „modulréteggel” rendelkezik, amely nagyon egyszerűvé teszi a rendszer kibővítését a meglévő kód foltozása nélkül. Egy példa erre a számelőállító mechanizmus a jegyeknél. Ez egy csatlakoztatható modulokkal rendelkező „modulréteg”, és ha szeretné, hozzáadhatja a saját egyéni számelőállító moduljait is. Nézzük meg részletesen a különböző rétegeket!

2.1. Hitelesítés és felhasználókezelés

2.1.1. Ügyintézői hitelesítő modul

Számos ügyintézői hitelesítő modul létezik (DB, LDAP és HTTPBasicAuth), amelyek az OTRS keretrendszerrel érkeznek. Lehetőség van saját hitelesítő modulok fejlesztésére is. Az ügyintézői hitelesítő modulok a `Kernel/System/Auth/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. Ezt követően egy egyszerű ügyintézői hitelesítő modul példája található. Mentse el a `Kernel/System/Auth/Simple.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `GetOption()` és `Auth()`. Adja vissza az uid-t, és ezután a hitelesítés rendben van.

2.1.1.1. Kódpélda

A felületosztály neve `Kernel::System::Auth`. A példa ügyintézői hitelesítés hívható `Kernel::System::Auth::CustomAuth` néven. Lent találhat egy példát.

```
# --
# Kernel/System/Auth/CustomAuth.pm - provides the CustomAuth authentication
# based on Martin Edenhofer's Kernel::System::Auth::DB
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# ID: CustomAuth.pm,v 1.1 2010/05/10 15:30:34 fk Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Auth::CustomAuth;

use strict;
use warnings;

use Authen::CustomAuth;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

    # get config
    $Self->{Die} = $Self->{ConfigObject}->Get( 'AuthModule::CustomAuth::Die' .
 $Param{Count} );

    # get user table
    $Self->{CustomAuthHost} = $Self->{ConfigObject}->Get( 'AuthModule::CustomAuth::Host' .
 $Param{Count} )
        || die "Need AuthModule::CustomAuth::Host$Param{Count}.";
    $Self->{CustomAuthSecret}
        = $Self->{ConfigObject}->Get( 'AuthModule::CustomAuth::Password' . $Param{Count} )
        || die "Need AuthModule::CustomAuth::Password$Param{Count}.";

    return $Self;
}

sub GetOption {
```

```
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{What} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need What!" );
        return;
    }

    # module options
    my %Option = ( PreAuth => 0, );

    # return option
    return $Option{ $Param{What} };
}

sub Auth {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{User} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need User!" );
        return;
    }

    # get params
    my $User       = $Param{User}      || '';
    my $Pw         = $Param{Pw}        || '';
    my $RemoteAddr = $ENV{REMOTE_ADDR} || 'Got no REMOTE_ADDR env!';
    my $UserID     = '';
    my $GetPw      = '';

    # just in case for debug!
    if ( $Self->{Debug} > 0 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: '$User' tried to authenticate with Pw: '$Pw' ($RemoteAddr)",
        );
    }

    # just a note
    if ( !$User ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "No User given!!! (REMOTE_ADDR: $RemoteAddr)",
        );
        return;
    }

    # just a note
    if ( !$Pw ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication without Pw!!! (REMOTE_ADDR:
 $RemoteAddr)",
        );
        return;
    }

    # Create a RADIUS object
    my $CustomAuth = Authen::CustomAuth->new(
        Host   => $Self->{CustomAuthHost},
        Secret => $Self->{CustomAuthecret},
    );
    if ( !$CustomAuth ) {
        if ( $Self->{Die} ) {
            die "Can't connect to $Self->{CustomAuthHost}: $@";
        }
        else {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Can't connect to $Self->{CustomAuthHost}: $@",
            );
```

```
            return;
        }
    }
    my $AuthResult = $CustomAuth->check_pwd( $User, $Pw );

    # login note
    if ( defined($AuthResult) && $AuthResult == 1 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication ok (REMOTE_ADDR: $RemoteAddr).",
        );
        return $User;
    }

    # just a note
    else {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User authentication with wrong Pw!!! (REMOTE_ADDR:
 $RemoteAddr)"
        );
        return;
    }
}

1;
```

2.1.1.2. Beállítási példa

Szükség van az egyéni ügyintézői hitelesítés modul bekapcsolására. Ezt a lenti Perl beállítás használatával lehet megtenni. Nem ajánlott az XML beállítás használata, mert kizárhatja magát a rendszerbeállításokon keresztül.

```
            $Self->{'AuthModule'} = 'Kernel::System::Auth::CustomAuth';
```

2.1.1.3. Használati eset példa

Egy hitelesítési megvalósítás hasznos példája lehet egy SOAP háttérprogram.

2.1.1.4. Kiadási elérhetőség

Név	Kiadás
DB	1.0
HTTPBasicAuth	1.2
LDAP	1.0
RADIUS	1.3

2.1.2. Hitelesítés szinkronizációs modul

Létezik egy LDAP hitelesítés szinkronizációs modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját hitelesítés modulok fejlesztésére is. A hitelesítés szinkronizációs modulok a `Kernel/System/Auth/Sync/*.pm` alatt találhatók. A beállításaikkal kapcsolatban további információkért nézze meg az adminisztrációs kézikönyvet. A következőkben egy hitelesítés szinkronizációs modul példája található. Mentse el a `Kernel/System/Auth/Sync/CustomAuthSync.pm` fájlba. Mindössze két függvényre van szüksége: `new()` és `Sync()`. Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

2.1.2.1. Kódpélda

A felületosztály neve `Kernel::System::Auth`. A példa ügyintézői hitelesítés hívható `Kernel::System::Auth::Sync::CustomAuthSync` néven. Lent találhat egy példát.

```
# --
# Kernel/System/Auth/Sync/CustomAuthSync.pm - provides the CustomAuthSync
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: CustomAuthSync.pm,v 1.9 2010/03/25 14:42:45 martin Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Auth::Sync::CustomAuthSync;

use strict;
use warnings;
use Net::LDAP;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject UserObject GroupObject EncodeObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

...

    return $Self;
}

sub Sync {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(User)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }
...
    return 1;
}
```

2.1.2.2. Beállítási példa

Be kell kapcsolnia az egyéni szinkronizációs modult. Ezt a lenti Perl beállítás használatával lehet megtenni. Nem ajánlott az XML beállítás használata, mert az lehetővé teheti, hogy kizárja magát a rendszerbeállításokon keresztül.

```
$Self->{'AuthSyncModule'} = 'Kernel::System::Auth::Sync::LDAP';
```

2.1.2.3. Használati eset példák

Hasznos szinkronizációs megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

2.1.2.4. Kiadási elérhetőség

Név	Kiadás
LDAP	2.4

2.1.2.5. Ellenjavaslatok és figyelmeztetések

Ne feledje, hogy a szinkronizáció a `Kernel::System::Auth` hitelesítési osztály része volt a 2.4-es keretrendszer előtt.

2.1.3. Ügyfél hitelesítő modul

Számos ügyfél hitelesítő modul létezik (DB, LDAP és HTTPBasicAuth), amelyek az OTRS keretrendszerrel érkeznek. Lehetőség van saját hitelesítő modulok fejlesztésére is. Az ügyfél hitelesítő modulok a `Kernel/System/CustomerAuth/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. Ezt követően egy egyszerű ügyfél hitelesítő modul példája található. Mentse el a `Kernel/System/CustomerAuth/Simple.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `GetOption()` és `Auth()`. Adja vissza az uid-t, és ezután a hitelesítés rendben van.

2.1.3.1. Kódpélda

A felületosztály neve `Kernel::System::CustomerAuth`. A példa ügyfél hitelesítés hívható `Kernel::System::CustomerAuth::CustomAuth` néven. Lent találhat egy példát.

```
# --
# Kernel/System/CustomerAuth/CustomAuth.pm - provides the custom Authentication
# based on Martin Edenhofer's Kernel::System::Auth::DB
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: CustomAuth.pm,v 1.11 2009/09/22 15:16:05 mb Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::CustomerAuth::CustomAuth;

use strict;
use warnings;

use Authen::CustomAuth;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(LogObject ConfigObject DBObject)) {
        $Self->{$_} = $Param{$_} || die "No $_!";
    }

    # Debug 0=off 1=on
    $Self->{Debug} = 0;

    # get config
    $Self->{Die}
```

```
        = $Self->{ConfigObject}->Get( 'Customer::AuthModule::CustomAuth::Die' .
 $Param{Count} );

    # get user table
    $Self->{CustomAuthHost}
        = $Self->{ConfigObject}->Get( 'Customer::AuthModule::CustomAuth::Host' .
 $Param{Count} )
        || die "Need Customer::AuthModule::CustomAuth::Host$Param{Count} in Kernel/
Config.pm";
    $Self->{CustomAuthSecret}
        = $Self->{ConfigObject}->Get( 'Customer::AuthModule::CustomAuth::Password' .
 $Param{Count} )
        || die "Need Customer::AuthModule::CustomAuth::Password$Param{Count} in Kernel/
Config.pm";

    return $Self;
}

sub GetOption {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{What} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need What!" );
        return;
    }

    # module options
    my %Option = ( PreAuth => 0, );

    # return option
    return $Option{ $Param{What} };
}

sub Auth {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    if ( !$Param{User} ) {
        $Self->{LogObject}->Log( Priority => 'error', Message => "Need User!" );
        return;
    }

    # get params
    my $User       = $Param{User}      || '';
    my $Pw         = $Param{Pw}        || '';
    my $RemoteAddr = $ENV{REMOTE_ADDR} || 'Got no REMOTE_ADDR env!';
    my $UserID     = '';
    my $GetPw      = '';

    # just in case for debug!
    if ( $Self->{Debug} > 0 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: '$User' tried to authentificate with Pw:
 '$Pw' ($RemoteAddr)",
        );
    }

    # just a note
    if ( !$User ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "No User given!!! (REMOTE_ADDR: $RemoteAddr)",
        );
        return;
    }

    # just a note
    if ( !$Pw ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
```

```
            Message  => "User: $User Authentication without Pw!!! (REMOTE_ADDR:
 $RemoteAddr)",
        );
        return;
    }

    # Create a custom object
    my $CustomAuth = Authen::CustomAuth->new(
        Host   => $Self->{CustomAuthHost},
        Secret => $Self->{CustomAuthSecret},
    );
    if ( !$CustomAuth ) {
        if ( $Self->{Die} ) {
            die "Can't connect to $Self->{CustomAuthHost}: $@";
        }
        else {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Can't connect to $Self->{CustomAuthHost}: $@",
            );
            return;
        }
    }
    my $AuthResult = $CustomAuth->check_pwd( $User, $Pw );

    # login note
    if ( defined($AuthResult) && $AuthResult == 1 ) {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User Authentication ok (REMOTE_ADDR: $RemoteAddr).",
        );
        return $User;
    }

    # just a note
    else {
        $Self->{LogObject}->Log(
            Priority => 'notice',
            Message  => "User: $User Authentication with wrong Pw!!! (REMOTE_ADDR:
 $RemoteAddr)"
        );
        return;
    }
}

1;
```

2.1.3.2. Beállítási példa

Szükség van az egyéni ügyfél hitelesítő modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="AuthModule" Required="1" Valid="1">
    <Description Lang="en">Module to authenticate customers.</Description>
    <Description Lang="hu">Egy modul az ügyfelek hitelesítéséhez.</Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::CustomerAuthAuth</SubGroup>
    <Setting>
        <Option Location="Kernel/System/CustomerAuth/*.pm"
 SelectedID="Kernel::System::CustomerAuth::CustomAuth"></Option>
    </Setting>
</ConfigItem>
```

2.1.3.3. Használati eset példa

Hasznos hitelesítés megvalósítás lehet egy SOAP háttérprogram.

2.1.3.4. Kiadási elérhetőség

Név	Kiadás
DB	1.0
HTTPBasicAuth	1.2
LDAP	1.0
RADIUS	1.3

2.2. Beállítások

2.2.1. Ügyfél-felhasználó beállítások modul

Létezik egy DB ügyfél-felhasználó beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját ügyfél-felhasználó beállítási modulok fejlesztésére is. Az ügyfél-felhasználó beállítási modulok a `Kernel/System/CustomerUser/Preferences/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy ügyfél-felhasználó beállítások modul példája található. Mentse el a `Kernel/System/CustomerUser/Preferences/Custom.pm` helyre. Mindössze négy függvényre van szüksége: `new()`, `SearchPreferences()`, `SetPreferences()` és `GetPreferences()`.

2.2.1.1. Kódpélda

A felületosztály neve `Kernel::System::CustomerUser`. A példa ügyfél-felhasználó beállítások hívhatók `Kernel::System::CustomerUser::Preferences::Custom` néven. Lent találhat egy példát.

```
# --
# Kernel/System/CustomerUser/Preferences/Custom.pm - some customer user functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: Custom.pm,v 1.20 2009/10/07 20:41:50 martin Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::CustomerUser::Preferences::Custom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Object (qw(DBObject ConfigObject LogObject)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    # preferences table data
    $Self->{PreferencesTable} = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}-
>{Table}
        || 'customer_preferences';
    $Self->{PreferencesTableKey}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->{TableKey}
```

```
        || 'preferences_key';
    $Self->{PreferencesTableValue}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->{TableValue}
        || 'preferences_value';
    $Self->{PreferencesTableUserID}
        = $Self->{ConfigObject}->Get('CustomerPreferences')->{Params}->{TableUserID}
        || 'user_id';

    return $Self;
}

sub SetPreferences {
    my ( $Self, %Param ) = @_;

    my $UserID = $Param{UserID} || return;
    my $Key    = $Param{Key}    || return;
    my $Value = defined( $Param{Value} ) ? $Param{Value} : '';

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . " $Self->{PreferencesTableUserID} = ? AND $Self->{PreferencesTableKey} = ?",
        Bind => [ \$UserID, \$Key ],
    );

    $Value .= 'Custom';

    # insert new data
    return if !$Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->{PreferencesTableUserID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue}) "
            . " VALUES (?, ?, ?)",
        Bind => [ \$UserID, \$Key, \$Value ],
    );

    return 1;
}

sub GetPreferences {
    my ( $Self, %Param ) = @_;

    my $UserID = $Param{UserID} || return;
    my %Data;

    # get preferences

    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->{PreferencesTableUserID} = ?",
        Bind => [ \$UserID ],
    );
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

sub SearchPreferences {
    my ( $Self, %Param ) = @_;

    my %UserID;
    my $Key   = $Param{Key}   || '';
    my $Value = $Param{Value} || '';

    # get preferences
    my $SQL = "SELECT $Self->{PreferencesTableUserID}, $Self->{PreferencesTableValue} "
        . " FROM "
        . " $Self->{PreferencesTable} "
        . " WHERE "
        . " $Self->{PreferencesTableKey} = '"
```

```
        . $Self->{DBObject}->Quote($Key) . "'" . " AND "
        . " LOWER($Self->{PreferencesTableValue}) LIKE LOWER('"
        . $Self->{DBObject}->Quote( $Value, 'Like' ) . "')";

    return if !$Self->{DBObject}->Prepare( SQL => $SQL );
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $UserID{ $Row[0] } = $Row[1];
    }

    # return data
    return %UserID;
}

1;
```

2.2.1.2. Beállítási példa

Szükség van az egyéni ügyfél-felhasználó beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="CustomerPreferences" Required="1" Valid="1">
    <Description Lang="en">Parameters for the customer preference table.</Description>
    <Description Lang="hu">Paraméterek az ügyfél beállításainak táblájához.</Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::Customer::Preferences</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::System::CustomerUser::Preferences::Custom</Item>
            <Item  Key="Params">
                <Hash>
                    <Item Key="Table">customer_preferences</Item>
                    <Item Key="TableKey">preferences_key</Item>
                    <Item Key="TableValue">preferences_value</Item>
                    <Item Key="TableUserID">user_id</Item>
                </Hash>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.2.1.3. Használati eset példa

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy LDAP háttérprogram.

2.2.1.4. Kiadási elérhetőség

Név	Kiadás
DB	2.3

2.2.2. Várólista beállítások modul

Létezik egy DB várólista beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját várólista beállítási modulok fejlesztésére is. A várólista beállítási modulok a `Kernel/System/Queue/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy várólista beállítások modul példája található. Mentse el a `Kernel/System/Queue/PreferencesCustom.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `QueuePreferencesSet()` és `QueuePreferencesGet()`. Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

2.2.2.1. Kódpélda

A felületosztály neve `Kernel::System::Queue`. A példa várólista beállítások hívhatók `Kernel::System::Queue::PreferencesCustom` néven. Lent találhat egy példát.

```
# --
# Kernel/System/Queue/PreferencesCustom.pm - some user functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: PreferencesCustom.pm,v 1.5 2009/02/16 11:47:34 tr Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Queue::PreferencesCustom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}        = 'queue_preferences';
    $Self->{PreferencesTableKey}     = 'preferences_key';
    $Self->{PreferencesTableValue}   = 'preferences_value';
    $Self->{PreferencesTableQueueID} = 'queue_id';

    return $Self;
}

sub QueuePreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(QueueID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableQueueID} = ? AND $Self->{PreferencesTableKey} = ?",
        Bind => [ \$Param{QueueID}, \$Param{Key} ],
    );

    $Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->{PreferencesTableQueueID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue}) "
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{QueueID}, \$Param{Key}, \$Param{Value} ],
```

```
    );
}

sub QueuePreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(QueueID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }

    # check if queue preferences are available
    if ( !$Self->{ConfigObject}->Get('QueuePreferences') ) {
        return;
    }

    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->{PreferencesTableQueueID} = ?",
        Bind => [ \$Param{QueueID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

1;
```

2.2.2.2. Beállítási példa

Szükség van az egyéni várólista beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="Queue::PreferencesModule" Required="1" Valid="1">
    <Description Lang="en">Default queue preferences module.</Description>
    <Description Lang="hu">Alapértelmezett várólista beállítások modul.</Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Queue::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::Queue::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

2.2.2.3. Használati eset példák

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

2.2.2.4. Kiadási elérhetőség

Név	Kiadás
PreferencesDB	2.3

2.2.3. Szolgáltatás beállítások modul

Létezik egy DB szolgáltatás beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját szolgáltatás beállítási modulok fejlesztésére is. A szolgáltatás

beállítási modulok a Kernel/System/Service/*.pm alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy szolgáltatás beállítások modul példája található. Mentse el a Kernel/System/Service/PreferencesCustom.pm helyre. Mindössze három függvényre van szüksége: new(), ServicePreferencesSet() és ServicePreferencesGet(). Adjon vissza 1-et, és ezután a szinkronizáció rendben van.

2.2.3.1. Kódpélda

A felületosztály neve Kernel::System::Service. A példa szolgáltatás beállítások hívhatók Kernel::System::Service::PreferencesCustom néven. Lent találhat egy példát.

```
# --
# Kernel/System/Service/PreferencesCustom - some user functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: PreferencesCustom.pm,v 1.2 2009/02/16 11:47:34 tr Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Service::PreferencesCustom;

use strict;
use warnings;

use vars qw(@ISA $VERSION);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}          = 'service_preferences';
    $Self->{PreferencesTableKey}       = 'preferences_key';
    $Self->{PreferencesTableValue}     = 'preferences_value';
    $Self->{PreferencesTableServiceID} = 'service_id';

    return $Self;
}

sub ServicePreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(ServiceID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableServiceID} = ? AND $Self->{PreferencesTableKey} = ?",
        Bind => [ \$Param{ServiceID}, \$Param{Key} ],
    );
```

```
$Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->{PreferencesTableServiceID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue}) "
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{ServiceID}, \$Param{Key}, \$Param{Value} ],
    );
}

sub ServicePreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(ServiceID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }

    # check if service preferences are available
    if ( !$Self->{ConfigObject}->Get('ServicePreferences') ) {
        return;
    }

    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->{PreferencesTableServiceID}
 = ?",
        Bind => [ \$Param{ServiceID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

1;
```

2.2.3.2. Beállítási példa

Szükség van az egyéni szolgáltatás beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="Service::PreferencesModule" Required="1" Valid="1">
    <Description Lang="en">Default service preferences module.</Description>
    <Description Lang="hu">Alapértelmezett szolgáltatás beállítások modul.</Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Service::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::Service::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

2.2.3.3. Használati eset példa

Hasznos beállítások megvalósítás lehet egy SOAP vagy egy RADIUS háttérprogram.

2.2.3.4. Kiadási elérhetőség

Név	Kiadás
PreferencesDB	2.4

2.2.4. SLA beállítások modul

Létezik egy DB SLA beállítások modul, amely az OTRS keretrendszerrel érkezik. Lehetőség van saját SLA beállítási modulok fejlesztésére is. Az SLA beállítási modulok a `Kernel/System/SLA/*.pm` alatt találhatók. Ezek beállításáról további információkért nézze meg az adminisztrátori kézikönyvet. A következőkben egy SLA beállítások modul példája található. Mentse el a `Kernel/System/SLA/PreferencesCustom.pm` helyre. Mindössze három függvényre van szüksége: `new()`, `SLAPreferencesSet()` és `SLAPreferencesGet()`. Győződjön meg arról, hogy a függvény 1-et adjon vissza.

2.2.4.1. Kódpélda

A felületosztály neve `Kernel::System::SLA`. A példa SLA beállítások hívhatók `Kernel::System::SLA::PreferencesCustom` néven. Lent találhat egy példát.

```
# --
# Kernel/System/SLA/PreferencesCustom.pm - some user functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::SLA::PreferencesCustom;

use strict;
use warnings;

use vars qw(@ISA);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for (qw(DBObject ConfigObject LogObject)) {
        $Self->{$_} = $Param{$_} || die "Got no $_!";
    }

    # preferences table data
    $Self->{PreferencesTable}      = 'sla_preferences';
    $Self->{PreferencesTableKey}   = 'preferences_key';
    $Self->{PreferencesTableValue} = 'preferences_value';
    $Self->{PreferencesTableSLAID} = 'sla_id';

    return $Self;
}

sub SLAPreferencesSet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(SLAID Key Value)) {
        if ( !defined( $Param{$_} ) ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
```

```
        }
    }

    # delete old data
    return if !$Self->{DBObject}->Do(
        SQL => "DELETE FROM $Self->{PreferencesTable} WHERE "
            . "$Self->{PreferencesTableSLAID} = ? AND $Self->{PreferencesTableKey} = ?",
        Bind => [ \$Param{SLAID}, \$Param{Key} ],
    );

$Self->{PreferencesTableValue} .= 'PreferencesCustom';

    # insert new data
    return $Self->{DBObject}->Do(
        SQL => "INSERT INTO $Self->{PreferencesTable} ($Self->{PreferencesTableSLAID}, "
            . " $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue}) "
            . " VALUES (?, ?, ?)",
        Bind => [ \$Param{SLAID}, \$Param{Key}, \$Param{Value} ],
    );
}

sub SLAPreferencesGet {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for (qw(SLAID)) {
        if ( !$Param{$_} ) {
            $Self->{LogObject}->Log( Priority => 'error', Message => "Need $_!" );
            return;
        }
    }

    # check if SLA preferences are available
    if ( !$Self->{ConfigObject}->Get('SLAPreferences') ) {
        return;
    }

    # get preferences
    return if !$Self->{DBObject}->Prepare(
        SQL => "SELECT $Self->{PreferencesTableKey}, $Self->{PreferencesTableValue} "
            . " FROM $Self->{PreferencesTable} WHERE $Self->{PreferencesTableSLAID} = ?",
        Bind => [ \$Param{SLAID} ],
    );
    my %Data;
    while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
        $Data{ $Row[0] } = $Row[1];
    }

    # return data
    return %Data;
}

1;
```

2.2.4.2. Beállítási példa

Szükség van az egyéni SLA beállítások modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="SLA::PreferencesModule" Required="1" Valid="1">
    <Description Translatable="1">Default SLA preferences module.</Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::SLA::Preferences</SubGroup>
    <Setting>
        <String Regex="">Kernel::System::SLA::PreferencesCustom</String>
    </Setting>
</ConfigItem>
```

2.2.4.3. Használati eset példa

Hasznos beállítások megvalósítás lehet további értékek tárolása az SLA-khoz.

2.2.4.4. Kiadási elérhetőség

Név	Kiadás
PreferencesDB	2.4

2.3. Egyéb alapfüggvények

2.3.1. Naplózás modul

Létezik egy globális naplózó felület az OTRS-hez, amely a saját naplózó háttérprogramok létrehozásának lehetőségét biztosítja.

Egy saját naplózó háttérprogram írása olyan egyszerű, mint újra megvalósítani a `Kernel::System::Log::Log()` metódust.

2.3.1.1. Kódpélda: `Kernel::System::Log::CustomFile`

Ebben a kis példában írni fogunk egy kicsi fájl naplózó háttérprogramot, amely hasonlóan működik mint a `Kernel::System::Log::File`, de egy szöveget fűz minden naplóbejegyzés elé.

```
# --
# Kernel/System/Log/CustomFile.pm - file log backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Log::CustomFile;

use strict;
use warnings;

umask "002";

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
    for (qw(ConfigObject EncodeObject)) {
        if ( $Param{$_} ) {
            $Self->{$_} = $Param{$_};
        }
        else {
            die "Got no $_!";
        }
    }

    # get logfile location
    $Self->{LogFile} = '/var/log/CustomFile.log';

    # set custom prefix
    $Self->{CustomPrefix} = 'CustomFileExample';

    # Fixed bug# 2265 - For IIS we need to create a own error log file.
```

```
    # Bind stderr to log file, because IIS do print stderr to web page.
    if ( $ENV{SERVER_SOFTWARE} && $ENV{SERVER_SOFTWARE} =~ /^microsoft\-iis/i ) {
        if ( !open STDERR, '>>', $Self->{LogFile} . '.error' ) {
            print STDERR "ERROR: Can't write $Self->{LogFile}.error: $!";
        }
    }

    return $Self;
}

sub Log {
    my ( $Self, %Param ) = @_;

    my $FH;

    # open logfile
    if ( !open $FH, '>>', $Self->{LogFile} ) {

        # print error screen
        print STDERR "\n";
        print STDERR " >> Can't write $Self->{LogFile}: $! <<\n";
        print STDERR "\n";
        return;
    }

    # write log file
    $Self->{EncodeObject}->SetIO($FH);
    print $FH '[' . localtime() . ']';
    if ( lc $Param{Priority} eq 'debug' ) {
        print $FH "[Debug][$Param{Module}][$Param{Line}] $Self->{CustomPrefix}
 $Param{Message}\n";
    }
    elsif ( lc $Param{Priority} eq 'info' ) {
        print $FH "[Info][$Param{Module}]  $Self->{CustomPrefix} $Param{Message}\n";
    }
    elsif ( lc $Param{Priority} eq 'notice' ) {
        print $FH "[Notice][$Param{Module}] $Self->{CustomPrefix} $Param{Message}\n";
    }
    elsif ( lc $Param{Priority} eq 'error' ) {
        print $FH "[Error][$Param{Module}][$Param{Line}] $Self->{CustomPrefix}
 $Param{Message}\n";
    }
    else {

        # print error messages to STDERR
        print STDERR
            "[Error][$Param{Module}] $Self->{CustomPrefix} Priority: '$Param{Priority}' not
 defined! Message: $Param{Message}\n";

        # and of course to logfile
        print $FH
            "[Error][$Param{Module}] $Self->{CustomPrefix} Priority: '$Param{Priority}' not
 defined! Message: $Param{Message}\n";
    }

    # close file handle
    close $FH;
    return 1;
}

1;
```

2.3.1.2. Beállítási példa

Az egyéni naplózómodul bekapcsolásához az adminisztrátor beállíthatja kézzel a meglévő `LogModule` konfigurációs elemet a `Kernel::System::Log::CustomFile` osztályhoz. Ennek automatikus megvalósításához megadhat egy XML beállítófájlt, amely felülbírálja az alapértelmezett beállítást.

```
<ConfigItem Name="LogModule" Required="1" Valid="1">
    <Description Translatable="1">Set Kernel::System::Log::CustomFile as default logging
 backend.</Description>
    <Group>Framework</Group>
    <SubGroup>Core::Log</SubGroup>
    <Setting>
        <Option Location="Kernel/System/Log/*.pm"
 SelectedID="Kernel::System::Log::CustomFile"></Option>
    </Setting>
</ConfigItem>
```

2.3.1.3. Használati eset példák

Hasznos naplózó háttérprogram lehet egy webszolgáltatásba vagy egy titkosított fájlba történő naplózás.

2.3.1.4. Ellenjavaslatok és figyelmeztetések

Ne feledje, hogy a `Kernel::System::Log` a `Log()` metóduson kívül egyéb metódusokkal is rendelkezik, amelyeket nem lehet újra megvalósítani, például az osztott memóriaszakaszokkal történő munkához tartozó kód és a naplóadatok gyorsítótárazása.

2.3.2. Kimenetszűrő

A kimenetszűrők lehetővé teszik a HTML módosítását röptében. A bevált gyakorlat a kimenetszűrők használata a `.tt` fájlok közvetlen módosítása helyett. Három jó ok létezik erre. Amikor ugyanazt az átdolgozást kell alkalmazni számos előtétprogram modulon, akkor az átdolgozást csak egyszer kell megvalósítani. A második előnye, hogy amikor az OTRS-t frissítik, akkor megvan az esély arra, hogy a szűrőt nem kell frissíteni, ha a hozzá tartozó minta nem változott. Amikor két kiterjesztés ugyanazt a fájlt módosítja, akkor ütközés lép fel a második csomag telepítése során. Ez az ütközés feloldható két kimenetszűrő használatával, amelyek ugyanazt az előtétprogram modult módosítják.

Három különböző fajta kimenetszűrő létezik. Ezek a HTML tartalom előállításának különböző szakaszaiban aktívak.

2.3.2.1. `FilterElementPost`

Ezek a szűrők lehetővé teszik egy sablon kimenetének módosítást, miután az megjelenítésre került.

A tartalom lefordításához futtathatja közvetlenül a `$LayoutObject->Translate()` függvényt. Ha egyéb sablonszolgáltatásokra van szüksége, akkor egyszerűen határozzon meg egy kis sablonfájlt a kimenetszűrőhöz, és használja azt a tartalom megjelenítéséhez, mielőtt beültetné azt a fő adatokba. Néhány esetben hasznos lehet a jQuery DOM műveletek használata is a képernyőn lévő tartalom sorrendjének megváltoztatásához vagy cseréjéhez a reguláris kifejezések használata helyett. Ebben az esetben láthatatlan tartalomként kellene beültetnie az új kódot valahova az oldalba (például a `Hidden` osztállyal), majd ezután áthelyezni a jQuery használatával a megfelelő helyre a DOM-ban, és megjeleníteni azt.

Az utó-kimenetszűrők használatának megkönnyítéséhez létezik egy mechanizmus is a HTML megjegyzéshorgok lekéréséhez bizonyos sablonoknál vagy blokkoknál. Hozzáadhatja a modulbeállító XML-be a következőhöz hasonlóan:

```
<ConfigItem
Name="Frontend::Template::GenerateBlockHooks###100-OTRSBusiness-ContactWithData"
Required="1" Valid="1">
    <Description Translatable="1">Generate HTML comment hooks for
the specified blocks so that filters can use them.</Description>
```

```
    <Group>OTRSBusiness</Group>
    <SubGroup>Core</SubGroup>
    <Setting>
        <Hash>
            <Item Key="AgentTicketZoom">
                <Array>
                    <Item>CustomerTable</Item>
                </Array>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

Ez azt fogja okozni, hogy az `AgentTicketZoom.tt` fájlban lévő `CustomerTable` blokk át lesz alakítva a HTML megjegyzésekben minden alkalommal, amikor megjelenítésre kerül:

```
<!--HookStartCustomerTable-->
... blokk kimenet ...
<!--HookEndCustomerTable-->
```

Ezzel a mechanizmussal minden csomag csak azokat a blokkhorgokat kérheti, amelyekre szüksége van, és következetesen kerülnek megjelenítésre. Ezek a HTML megjegyzések használhatók ezután a kimenetszűrőben az egyszerű reguláris kifejezés illesztéshez.

2.3.2.2. `FilterContent`

Ez a fajta szűrő lehetővé teszi a teljes HTML kimenet feldolgozását a kérésnél közvetlenül azelőtt, hogy kiküldésre kerül a böngészőnek. Ez globális átalakításokhoz használható.

2.3.2.3. `FilterText`

Ez a fajta kimenetszűrő egy bővítmény a `Kernel::Output::HTML::Layout::Ascii2HTML()` metódushoz, és csak akkor aktív, amikor a `LinkFeature` paraméter 1-re van állítva. Így a `FilterText` kimenetszűrők jelenleg csak az egyszerű szöveges bejegyzések törzsének megjelenítésénél aktívak. Az egyszerű szöveges bejegyzéseket a bejövő nem HTML levelek állítják elő, illetve amikor az OTRS úgy van beállítva, hogy ne használja a Rich Text szolgáltatást az előtétprogramon.

2.3.2.4. Kódpélda

Lásd a `TemplateModule` csomagot.

2.3.2.5. Beállítási példa

Lásd a `TemplateModule` csomagot.

2.3.2.6. Használati esetek

2.3.2.6.1. További jegyattribútumok megjelenítése az `AgentTicketZoom` képernyőn

Ez egy `FilterElementPost` kimenetszűrővel valósítható meg.

2.3.2.6.2. A szolgáltatásválasztás megjelenítése többszintű menüként

Használjon egy `FilterElementPost` szűrőt ehhez a szolgáltatáshoz. A választható szolgáltatások listája a feldolgozott sablonkimenetből dolgozható fel. A többszintű választás a szolgáltatáslistából építhető fel, és szúrható be a sablontartalomba. Egy `FilterElementPost` kimenetszűrőt kell használni ehhez.

2.3.2.6.3. Hivatkozások létrehozása az egyszerű szöveges bejegyzés törzseiben

Egy biotechnológiai vállalat IPI00217472 formátumú génneveket használ az egyszerű szöveges bejegyzésekben. Egy `FilterText` kimenetszűrő használható a szekvencia-adatbázisra mutató hivatkozások létrehozásához a génneveknél, például http://srs.ebi.ac.uk/srsbin/cgi-bin/wgetz?-e+[IPI-acc:IPI00217472]+-vn+2 formában.

2.3.2.6.4. Az aktív tartalom megtiltása

Van egy olyan tűzfalszabály, amely megtiltja az összes aktív tartalmat. Azért, hogy elkerüljük a tűzfal visszautasítását, az <applet> HTML-címke kiszűrhető egy `FilterContent` kimenetszűrővel.

2.3.2.7. Ellenjavaslatok és figyelmeztetések

Minden `FilterElementPost` kimenetszűrő felépítésre és futtatásra kerül minden olyan beállított sablonnál, amely szükséges az aktuális kéréshez. Így a kimenetszűrő alacsony teljesítménye vagy a szűrők nagy száma komolyan csökkentheti a teljesítményt.

2.3.2.8. Bevált gyakorlatok

A rugalmasság növelésének érdekében az érintett sablonok listáját be kell állítani a rendszerbeállításokban.

2.3.2.9. Kiadási elérhetőség

A kimenetszűrők az OTRS 2.4-es verziójától érhetők el. A `FilterElementPre` típus eldobásra került az OTRS 5-tel.

2.3.3. Statisztikák modul

A belső statisztikamoduloknak két különböző típusa létezik - dinamikus és statikus. Ez a szakasz azt írja le, hogy az ilyen statisztikamodulok hogyan fejleszthetők.

2.3.3.1. Dinamikus statisztikák

A statikus statisztikamodulokkal ellentétben a dinamikus statisztikák beállíthatók az OTRS webes felületén keresztül. Ebben a szakaszban egy egyszerű statisztikamodul kerül fejlesztésre. Minden egyes dinamikus statisztikamodulnak meg kell valósítania ezeket a szubrutinokat:

- `new`
- `GetObjectName`
- `GetObjectAttributes`
- `ExportWrapper`
- `ImportWrapper`

Továbbá a modulnak meg kell valósítania vagy a `GetStatElement`, vagy a `GetStatTable` rutint. És ha az eredménytábla fejlécsorát is meg kell változtatni, akkor egy úgynevezett `GetHeaderLine` szubrutint is fejleszteni kell.

2.3.3.1.1. Kódpélda

Ebben a szakaszban egy minta statisztikamodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Kernel/System/Stats/Dynamic/DynamicStatsTemplate.pm - all advice functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Dynamic::DynamicStatsTemplate;

use strict;
use warnings;

use Kernel::System::Queue;
use Kernel::System::State;
use Kernel::System::Ticket;
```

Ez egy gyakori sabloncsomag, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva. Ezután a szükséges modulok használatának megadása következik a use kulcsszóval.

```
sub new {
    my ( $Type, %Param ) = @_;

    # új kivonat lefoglalása az objektumhoz
    my $Self = {};
    bless( $Self, $Type );

    # a szükséges objektumok ellenőrzése
    for my $Object (
        qw(DBObject ConfigObject LogObject UserObject TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Object} = $Param{$Object} || die "Nincs $Object!";
    }

    # a létrehozott szükséges objektumok
    $Self->{QueueObject}    = Kernel::System::Queue->new( %{$Self} );
    $Self->{TicketObject}   = Kernel::System::Ticket->new( %{$Self} );
    $Self->{StateObject}    = Kernel::System::State->new( %{$Self} );

    return $Self;
}
```

A new a statisztikamodul konstruktora. Ez hozza létre az osztály új példányát. A kódolási irányelveknek megfelelően az ebben a modulban szükséges más osztályok objektumait is a new konstruktorban kell létrehozni. A 27-29. sorban van létrehozva a statisztikák modul objektuma. A 31-37. sorban azt ellenőrzik, hogy az ebben a kódban szükséges objektumok - vagy más objektumok létrehozásánál, vagy ebben a modulban - át vannak-e adva. Ezután a többi objektum kerül létrehozásra.

```
sub GetObjectName {
    my ( $Self, %Param ) = @_;

    return 'Minta statisztikák';
}
```

A GetObjectName visszaad egy nevet a statisztikák modulhoz. Ez az a címke, amely a lenyíló menüben jelenik meg a beállításokban, valamint a meglévő statisztikák listájában (az „objektum” oszlopban).

```
sub GetObjectAttributes {
```

```
    my ( $Self, %Param ) = @_;

    # állapotlista lekérése
    my %StateList = $Self->{StateObject}->StateList(
        UserID => 1,
    );

    # várólisták listájának lekérése
    my %QueueList = $Self->{QueueObject}->GetAllQueues();

    # a jelenlegi idő lekérése a 3830. hiba javításához
    my $TimeStamp = $Self->{TimeObject}->CurrentTimestamp();
    my ($Date) = split /\s+/, $TimeStamp;
    my $Today = sprintf "%s 23:59:59", $Date;

    my @ObjectAttributes = (
        {
            Name             => 'Állapot',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'StateIDs',
            Block            => 'MultiSelectField',
            Values           => \%StateList,
        },
        {
            Name             => 'Létrehozva várólistában',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreatedQueueIDs',
            Block            => 'MultiSelectField',
            Translation      => 0,
            Values           => \%QueueList,
        },
        {
            Name             => 'Létrehozás ideje',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreateTime',
            TimePeriodFormat => 'DateInputFormat',    # 'DateInputFormatLong',
            Block            => 'Time',
            TimeStop         => $Today,
            Values           => {
                TimeStart => 'TicketCreateTimeNewerDate',
                TimeStop  => 'TicketCreateTimeOlderDate',
            },
        },
    );

    return @ObjectAttributes;
}
```

Ebben a minta statisztikák modulban három olyan attribútumot szeretnénk szolgáltatni, amelyből a felhasználó választhat: a várólisták listáját, az állapotok listáját és egy idő legördülőt. A legördülőben megjelenített értékek lekéréséhez szükséges néhány művelet. Ebben az esetben a `StateList` és a `GetAllQueues` kerül meghívásra.

Ezután az attribútumok listája kerül létrehozásra. Minden egyes attribútum egy kivonathivatkozáson keresztül van meghatározva. Ezeket a kulcsokat használhatja:

- `Name`

  A címke a webes felületen.

- `UseAsXvalue`

  Ez az attribútum használható az X-tengelyen.

- `UseAsValueSeries`

  Ez az attribútum használható az Y-tengelyen.

- `UseAsRestriction`

  Ez az attribútum használható a korlátozásokhoz.

- `Element`

  A HTML mező neve.

- `Block`

  A blokknév a sablonfájlban (például `<OTRS_HOME>/Kernel/Output/HTML/Standard/AgentStatsEditXaxis.tt`).

- `Values`

  Az attribútumban megjelenített értékek.

Tipp: Ha telepíti ezt a mintát, és beállít egy statisztikát néhány várólistával (mondjuk „A várólista” és „B várólista”), akkor ezek a várólisták az egyetlenek, amelyek láthatóak lesznek a felhasználónak, amikor elindítja a statisztikát. Néha egy dinamikus legördülő vagy többválasztós mező szükséges. Ebben az esetben beállíthatja a `SelectedValues` kulcsot az attribútum meghatározásában:

```
        {
            Name             => 'Létrehozva várólistában',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreatedQueueIDs',
            Block            => 'MultiSelectField',
            Translation      => 0,
            Values           => \%QueueList,
            SelectedValues   => [ @SelectedQueues ],
        },
```

```
sub GetStatElement {
    my ( $Self, %Param ) = @_;

    # jegyek keresése
    return $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'COUNT',
        Permission => 'ro',
        Limit      => 100_000_000,
        %Param,
    );
}
```

A `GetStatElement` kerül meghívásra minden egyes cellánál az eredménytáblában. Így annak számszerű értéknek kell lennie. Ebben a mintában egy egyszerű jegykeresést hajt végre. A `%Param` kivonat tartalmaz információkat a „jelenlegi” X-értékről és Y-értékről, valamint bármely korlátozásról. Így egy olyan cellánál, amelynek össze kell számolnia a „nyitott” állapotban lévő létrehozott jegyeket a „Misc” várólistánál, az átadott paraméter kivonat valahogy így néz ki:

```
    'CreatedQueueIDs' => [
        '4'
    ],
```

```
    'StateIDs' => [
        '2'
    ]
```

Ha a „cellánkénti” számítást el kellene kerülni, akkor a `GetStatTable` egy alternatíva. A `GetStatTable` visszaadja a sorok listáját, amely ezentúl egy tömbhivatkozások tömbje. Ez ugyanahhoz az eredményhez vezet mint a `GetStatElement` használata.

```
sub GetStatTable {
    my ( $Self, %Param ) = @_;

    my @StatData;

    for my $StateName ( keys %{ $Param{TableStructure} } ) {
        my @Row;
        for my $Params ( @{ $Param{TableStructure}->{$StateName} } ) {
            my $Tickets = $Self->{TicketObject}->TicketSearch(
                UserID     => 1,
                Result     => 'COUNT',
                Permission => 'ro',
                Limit      => 100_000_000,
                %{$Params},
            );

            push @Row, $Tickets;
        }

        push @StatData, [ $StateName, @Row ];
    }

    return @StatData;
}
```

A `GetStatTable` az összes olyan információt lekéri a statisztikák lekérdezéssel kapcsolatban, amelyek szükségesek. Az átadott paraméterek információkat tartalmaznak az attribútumokról (`Restrictions`, olyan attribútumok, amelyek az X/Y-tengelynél vannak használva) és a táblaszerkezetről. A táblaszerkezet egy olyan kivonathivatkozás, ahol a kulcsok az Y-tengely értékei, és azok értékei kivonathivatkozások a `GetStatElement` szubrutinokhoz használt paraméterekkel.

```
    'Restrictions' => {},
    'TableStructure' => {
        'closed successful' => [
            {
                'CreatedQueueIDs' => [
                    '3'
                ],
                'StateIDs' => [
                    '2'
                ]
            },
        ],
        'closed unsuccessful' => [
            {
                'CreatedQueueIDs' => [
                    '3'
                ],
                'StateIDs' => [
                    '3'
                ]
            },
        ],
    },
    'ValueSeries' => [
        {
            'Block' => 'MultiSelectField',
```

```
            'Element' => 'StateIDs',
            'Name' => 'Állapot',
            'SelectedValues' => [
                '5',
                '3',
                '2',
                '1',
                '4'
            ],
            'Translation' => 1,
            'Values' => {
                '1' => 'new',
                '10' => 'closed with workaround',
                '2' => 'closed successful',
                '3' => 'closed unsuccessful',
                '4' => 'open',
                '5' => 'removed',
                '6' => 'pending reminder',
                '7' => 'pending auto close+',
                '8' => 'pending auto close-',
                '9' => 'merged'
            }
        }
    ],
    'XValue' => {
        'Block' => 'MultiSelectField',
        'Element' => 'CreatedQueueIDs',
        'Name' => 'Létrehozva várólistában',
        'SelectedValues' => [
            '3',
            '4',
            '1',
            '2'
        ],
        'Translation' => 0,
        'Values' => {
            '1' => 'Postmaster',
            '2' => 'Raw',
            '3' => 'Junk',
            '4' => 'Misc'
        }
    }
```

Néha a táblázat fejléceit meg kell változtatni. Ebben az esetben egy `GetHeaderLine` nevű szubrutint kell megvalósítani. Ennek a szubrutinnak egy tömbhivatkozást kell visszaadnia az oszlopfejlécekkel mint elemekkel. Ez információkat kap az átadott X-értékekkel kapcsolatban.

```
sub GetHeaderLine {
    my ( $Self, %Param ) = @_;

    my @HeaderLine = ('');
    for my $SelectedXValue ( @{ $Param{XValue}->{SelectedValues} } ) {
        push @HeaderLine, $Param{XValue}->{Values}->{$SelectedXValue};
    }

    return \@HeaderLine;
}
```

```
sub ExportWrapper {
    my ( $Self, %Param ) = @_;

    # azonosítók átalakítása a használt helyesíráshoz
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
```

```
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq 'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $Self->{QueueObject}->QueueLookup( QueueID => $ID-
>{Content} );
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq 'CreatedStateIDs' ) {
                my %StateList = $Self->{StateObject}->StateList( UserID => 1 );
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $StateList{ $ID->{Content} };
                }
            }
        }
    }
    return \%Param;
}
```

A beállított statisztikák exportálhatók XML-formátumba. De ahogy a várólistáknál, ahol ugyanazok a várólistanevek rendelkezhetnek különböző azonosítókkal a különböző OTRS példányoknál, különösen fájdalmas lehet az azonosítók exportálása (a statisztikák ekkor rossz számokat számítanának ki). Ezért egy exportálási átalakítót kell írni, hogy az azonosítók helyett neveket használjon. Ezt a statisztikák modul minden egyes „dimenziójánál” el kell végezni (X-tengely, Y-tengely és korlátozások).

Az `ImportWrapper` fordítva működik - átalakítja a nevet az azonosítóra abban a példányban, ahova a beállítások importálásra kerülnek.

Ez egy minta exportálás:

```
<?xml version="1.0" encoding="utf-8"?>

<otrs_stats>
<Cache>0</Cache>
<Description>Minta statisztikák modul</Description>
<File></File>
<Format>CSV</Format>
<Format>Print</Format>
<Object>DeveloperManualSample</Object>
<ObjectModule>Kernel::System::Stats::Dynamic::DynamicStatsTemplate</ObjectModule>
<ObjectName>Sample Statistics</ObjectName>
<Permission>stats</Permission>
<StatType>dynamic</StatType>
<SumCol>0</SumCol>
<SumRow>0</SumRow>
<Title>Sample 1</Title>
<UseAsValueSeries Element="StateIDs" Fixed="1">
<SelectedValues>removed</SelectedValues>
<SelectedValues>closed unsuccessful</SelectedValues>
<SelectedValues>closed successful</SelectedValues>
<SelectedValues>new</SelectedValues>
<SelectedValues>open</SelectedValues>
</UseAsValueSeries>
<UseAsXvalue Element="CreatedQueueIDs" Fixed="1">
<SelectedValues>Junk</SelectedValues>
<SelectedValues>Misc</SelectedValues>
<SelectedValues>Postmaster</SelectedValues>
<SelectedValues>Raw</SelectedValues>
</UseAsXvalue>
<Valid>1</Valid>
</otrs_stats>
```

Most, hogy az összes szubrutin elmagyarázásra került, itt a teljes minta statisztikák modul.

```
# --
# Kernel/System/Stats/Dynamic/DynamicStatsTemplate.pm - all advice functions
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Dynamic::DynamicStatsTemplate;

use strict;
use warnings;

use Kernel::System::Queue;
use Kernel::System::State;
use Kernel::System::Ticket;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # check needed objects
    for my $Object (
        qw(DBObject ConfigObject LogObject UserObject TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    # created needed objects
    $Self->{QueueObject}    = Kernel::System::Queue->new( %{$Self} );
    $Self->{TicketObject}   = Kernel::System::Ticket->new( %{$Self} );
    $Self->{StateObject}    = Kernel::System::State->new( %{$Self} );

    return $Self;
}

sub GetObjectName {
    my ( $Self, %Param ) = @_;

    return 'Sample Statistics';
}

sub GetObjectAttributes {
    my ( $Self, %Param ) = @_;

    # get state list
    my %StateList = $Self->{StateObject}->StateList(
        UserID => 1,
    );

    # get queue list
    my %QueueList = $Self->{QueueObject}->GetAllQueues();

    # get current time to fix bug#3830
    my $TimeStamp = $Self->{TimeObject}->CurrentTimestamp();
    my ($Date) = split /\s+/, $TimeStamp;
    my $Today = sprintf "%s 23:59:59", $Date;

    my @ObjectAttributes = (
        {
            Name             => 'State',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
```

```
            UseAsRestriction => 1,
            Element          => 'StateIDs',
            Block            => 'MultiSelectField',
            Values           => \%StateList,
        },
        {
            Name             => 'Created in Queue',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreatedQueueIDs',
            Block            => 'MultiSelectField',
            Translation      => 0,
            Values           => \%QueueList,
        },
        {
            Name             => 'Create Time',
            UseAsXvalue      => 1,
            UseAsValueSeries => 1,
            UseAsRestriction => 1,
            Element          => 'CreateTime',
            TimePeriodFormat => 'DateInputFormat',    # 'DateInputFormatLong',
            Block            => 'Time',
            TimeStop         => $Today,
            Values           => {
                TimeStart => 'TicketCreateTimeNewerDate',
                TimeStop  => 'TicketCreateTimeOlderDate',
            },
        },
    );

    return @ObjectAttributes;
}

sub GetStatElement {
    my ( $Self, %Param ) = @_;

    # search tickets
    return $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'COUNT',
        Permission => 'ro',
        Limit      => 100_000_000,
        %Param,
    );
}

sub ExportWrapper {
    my ( $Self, %Param ) = @_;

    # wrap ids to used spelling
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq 'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $Self->{QueueObject}->QueueLookup( QueueID => $ID-
>{Content} );
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq 'CreatedStateIDs' ) {
                my %StateList = $Self->{StateObject}->StateList( UserID => 1 );
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    $ID->{Content} = $StateList{ $ID->{Content} };
```

```
                }
            }
        }
    }
    return \%Param;
}

sub ImportWrapper {
    my ( $Self, %Param ) = @_;

    # wrap used spelling to ids
    for my $Use (qw(UseAsValueSeries UseAsRestriction UseAsXvalue)) {
        ELEMENT:
        for my $Element ( @{ $Param{$Use} } ) {
            next ELEMENT if !$Element || !$Element->{SelectedValues};
            my $ElementName = $Element->{Element};
            my $Values      = $Element->{SelectedValues};

            if ( $ElementName eq 'QueueIDs' || $ElementName eq 'CreatedQueueIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;
                    if ( $Self->{QueueObject}->QueueLookup( Queue => $ID->{Content} ) ) {
                        $ID->{Content}
                            = $Self->{QueueObject}->QueueLookup( Queue => $ID->{Content} );
                    }
                    else {
                        $Self->{LogObject}->Log(
                            Priority => 'error',
                            Message  => "Import: Can' find the queue $ID->{Content}!"
                        );
                        $ID = undef;
                    }
                }
            }
            elsif ( $ElementName eq 'StateIDs' || $ElementName eq 'CreatedStateIDs' ) {
                ID:
                for my $ID ( @{$Values} ) {
                    next ID if !$ID;

                    my %State = $Self->{StateObject}->StateGet(
                        Name  => $ID->{Content},
                        Cache => 1,
                    );
                    if ( $State{ID} ) {
                        $ID->{Content} = $State{ID};
                    }
                    else {
                        $Self->{LogObject}->Log(
                            Priority => 'error',
                            Message  => "Import: Can' find state $ID->{Content}!"
                        );
                        $ID = undef;
                    }
                }
            }
        }
    }
    return \%Param;
}

1;
```

2.3.3.1.2. Beállítási példa

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_config version="1.0" init="Config">
    <ConfigItem Name="Stats::DynamicObjectRegistration###DynamicStatsTemplate" Required="0"
 Valid="1">
```

```
        <Description Lang="en">Here you can decide if the common stats module may generate
 stats about the number of default tickets a requester created.</Description>
        <Group>Framework</Group>
        <SubGroup>Core::Stats</SubGroup>
        <Setting>
            <Hash>
                <Item Key="Module">Kernel::System::Stats::Dynamic::DynamicStatsTemplate</
Item>
            </Hash>
        </Setting>
    </ConfigItem>
</otrs_config>
```

2.3.3.1.3. Használati eset példák

Használati esetek.

2.3.3.1.4. Ellenjavaslatok és figyelmeztetések

Ha nagyon sok cellája van az eredménytáblázatban és a `GetStatElement` meglehetősen összetett, akkor a kérés eltarthat egy ideig.

2.3.3.1.5. Kiadási elérhetőség

A dinamikus statisztikamodulok az OTRS 2.0 óta érhetők el.

2.3.3.2. Statikus statisztikák

A következő bekezdések a statikus statisztikákat írják le. A statikus statisztikákat nagyon könnyű létrehozni, mivel ezeknek a moduloknak csak három szubrutint kell megvalósítaniuk.

- new
- `Param`
- Run

2.3.3.2.1. Kódpélda

A következő bekezdések a statikus statisztikákban szükséges szubrutinokat mutatják be.

```
sub new {
    my ( $Type, %Param ) = @_;

    # új kivonat lefoglalása az objektumhoz
    my $Self = {%Param};
    bless( $Self, $Type );

    # az összes szükséges objektum ellenőrzése
    for my $Needed (
        qw(DBObject ConfigObject LogObject
        TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Needed} = $Param{$Needed} || die "Nincs $Needed";
    }

    # a szükséges objektumok létrehozása
    $Self->{TypeObject}   = Kernel::System::Type->new( %{$Self} );
    $Self->{TicketObject} = Kernel::System::Ticket->new( %{$Self} );
    $Self->{QueueObject}  = Kernel::System::Queue->new( %{$Self} );

    return $Self;
}
```

A new hozza létre a statikus statisztikák osztályának egy új példányát. Először létrehoz egy új objektumot, és azután ellenőrzi a szükséges objektumokat.

```
sub Param {
    my $Self = shift;

    my %Queues = $Self->{QueueObject}->GetAllQueues();
    my %Types  = $Self->{TypeObject}->TypeList(
        Valid => 1,
    );

    my @Params = (
        {
            Frontend  => 'Type',
            Name      => 'TypeIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Types,
        },
        {
            Frontend  => 'Queue',
            Name      => 'QueueIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Queues,
        },
    );

    return @Params;
}
```

A `Param` metódus biztosítja az összes olyan paraméter és attribútum listáját, amelyek kiválaszthatók egy statikus statisztika létrehozásához. Megkap néhány átadott paramétert: az értékeket egy kérésben szolgáltatott statisztikák attribútumaihoz, a statisztikák formátumát és az objektum nevét (a modul nevét).

A paramétereknek és az attribútumoknak kivonathivatkozásoknak kell lenniük ezekkel a kulcs-érték párokkal:

- `Frontend`

  A címke a webes felületen.

- `Name`

  A HTML mező neve.

- `Data`

  Az attribútumban megjelenített értékek.

Egyéb paraméterek is használhatók a `LayoutObjectBuildSelection` metódusánál, ahogy az a `Size` és `Multiple` paraméterekkel történik ebben a minta modulban.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # a szükséges dolgok ellenőrzése
    for my $Needed (qw(TypeIDs QueueIDs)) {
        if ( !$Param{$Needed} ) {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Szükséges: $Needed!",
            );
            return;
        }
```

```
    }

    # a jelentés címének beállítása
    my $Title = 'Jegyek várólistánként';

    # táblázat címsorok
    my @HeadData = (
        'Jegyszám',
        'Várólista',
        'Típus',
    );

    my @Data;
    my @TicketIDs = $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'ARRAY',
        Permission => 'ro',
        %Param,
    );

    for my $TicketID ( @TicketIDs ) {
        my %Ticket = $Self->{TicketObject}->TicketGet(
            UserID => 1,
            TicketID => $TicketID,
        );
        push @Data, [ $Ticket{TicketNumber}, $Ticket{Queue}, $Ticket{Type} ];
    }

    return ( [$Title], [@HeadData], @Data );
}
```

Tulajdonképpen a Run metódus állítja elő a táblázat adatait a statisztikákhoz. Megkapja az ennél a statisztikánál átadott attribútumokat. Ebben a mintában a `%Param` paraméterben egy `TypeIDs` kulcs és egy `QueueIDs` kulcs létezik (lásd a `Param` metódusban lévő attribútumokat), és ezek értékei tömbhivatkozások. A visszaadott adatok három részből állnak: két tömbhivatkozásból és egy tömbből. Az első tömbhivatkozásban a statisztika címe van eltárolva, a második tömbhivatkozás tartalmazza a táblázatban lévő oszlopok címsorait. És ezután a táblázattörzs adatai következnek.

```
# --
# Kernel/System/Stats/Static/StaticStatsTemplate.pm
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Stats::Static::StaticStatsTemplate;

use strict;
use warnings;

use Kernel::System::Type;
use Kernel::System::Ticket;
use Kernel::System::Queue;

=head1 NAME

StaticStatsTemplate.pm - the module that creates the stats about tickets in a queue

=head1 SYNOPSIS

All functions

=head1 PUBLIC INTERFACE

=over 4
```

```
=cut

=item new()

create an object

    use Kernel::Config;
    use Kernel::System::Encode;
    use Kernel::System::Log;
    use Kernel::System::Main;
    use Kernel::System::Time;
    use Kernel::System::DB;
    use Kernel::System::Stats::Static::StaticStatsTemplate;

    my $ConfigObject = Kernel::Config->new();
    my $EncodeObject = Kernel::System::Encode->new(
        ConfigObject => $ConfigObject,
    );
    my $LogObject    = Kernel::System::Log->new(
        ConfigObject => $ConfigObject,
    );
    my $MainObject = Kernel::System::Main->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
    );
    my $TimeObject = Kernel::System::Time->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
    );
    my $DBObject = Kernel::System::DB->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
        MainObject   => $MainObject,
    );
    my $StatsObject = Kernel::System::Stats::Static::StaticStatsTemplate->new(
        ConfigObject => $ConfigObject,
        LogObject    => $LogObject,
        MainObject   => $MainObject,
        TimeObject   => $TimeObject,
        DBObject     => $DBObject,
        EncodeObject => $EncodeObject,
    );

=cut

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    # check all needed objects
    for my $Needed (
        qw(DBObject ConfigObject LogObject
        TimeObject MainObject EncodeObject)
        )
    {
        $Self->{$Needed} = $Param{$Needed} || die "Got no $Needed";
    }

    # create needed objects
    $Self->{TypeObject}   = Kernel::System::Type->new( %{$Self} );
    $Self->{TicketObject} = Kernel::System::Ticket->new( %{$Self} );
    $Self->{QueueObject}  = Kernel::System::Queue->new( %{$Self} );

    return $Self;
}

=item Param()

Get all parameters a user can specify.
```

```
    my @Params = $StatsObject->Param();

=cut

sub Param {
    my $Self = shift;

    my %Queues = $Self->{QueueObject}->GetAllQueues();
    my %Types  = $Self->{TypeObject}->TypeList(
        Valid => 1,
    );

    my @Params = (
        {
            Frontend  => 'Type',
            Name      => 'TypeIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Types,
        },
        {
            Frontend  => 'Queue',
            Name      => 'QueueIDs',
            Multiple  => 1,
            Size      => 3,
            Data      => \%Queues,
        },
    );

    return @Params;
}

=item Run()

generate the statistic.

    my $StatsInfo = $StatsObject->Run(
        TypeIDs  => [
            1, 2, 4
        ],
        QueueIDs => [
            3, 4, 6
        ],
    );

=cut

sub Run {
    my ( $Self, %Param ) = @_;

    # check needed stuff
    for my $Needed (qw(TypeIDs QueueIDs)) {
        if ( !$Param{$Needed} ) {
            $Self->{LogObject}->Log(
                Priority => 'error',
                Message  => "Need $Needed!",
            );
            return;
        }
    }

    # set report title
    my $Title = 'Tickets per Queue';

    # table headlines
    my @HeadData = (
        'Ticket Number',
        'Queue',
        'Type',
    );
```

```
    my @Data;
    my @TicketIDs = $Self->{TicketObject}->TicketSearch(
        UserID     => 1,
        Result     => 'ARRAY',
        Permission => 'ro',
        %Param,
    );

    for my $TicketID ( @TicketIDs ) {
        my %Ticket = $Self->{TicketObject}->TicketGet(
            UserID => 1,
            TicketID => $TicketID,
        );
        push @Data, [ $Ticket{TicketNumber}, $Ticket{Queue}, $Ticket{Type} ];
    }

    return ( [$Title], [@HeadData], @Data );
}

1;
```

2.3.3.2.2. Beállítási példa

Nincs szükség beállításokra. Közvetlenül telepítés után a modul elérhető egy statisztika létrehozásához ennél a modulnál.

2.3.3.2.3. Használati eset példák

Használati esetek.

2.3.3.2.4. Ellenjavaslatok és figyelmeztetések

Ellenjavaslatok és figyelmeztetések a statikus statisztikáknál.

2.3.3.2.5. Kiadási elérhetőség

A statikus statisztikamodulok az OTRS 1.3 óta érhetők el.

2.3.3.2.6. Régi statikus statisztikák használata

A szabványos 1.3-as és 2.0-ás OTRS verziók már megkönnyítették a statisztikák előállítását. Az OTRS 1.3-as és 2.0-ás verzióinak különféle statisztikái, amelyek különlegesen lettek kifejlesztve, hogy kielégítsék az ügyfelek követelményeit, használhatók az újabb verziókban is.

A fájlokat pusztán csak át kell helyezni a `Kernel/System/Stats/` útvonalról a `Kernel/System/Stats/Static/` könyvtárba. Továbbá a megfelelő parancsfájl csomagnevét `::Static` névre kell módosítani.

A következő példa azt mutatja be, hogy hogyan kell az első útvonalat módosítani.

```
    package Kernel::System::Stats::AccountedTime;
```

```
    package Kernel::System::Stats::Static::AccountedTime;
```

2.3.4. Jegyszám előállító modulok

A jegyszám előállítókat elkülönülő azonosítók létrehozásához használják az új jegyekhez, amelyeket jegyszámnak neveznek. Bármilyen metódus lehetséges a számok karakterláncainak létrehozásához, de a józan ész határain belül kell maradnia az eredményül kapott szöveg hosszával kapcsolatban (irányelv: 5-10).

Egy jegyszám létrehozásakor győződjön meg arról, hogy az eredmény megkapta-e a SystemID rendszerbeállítási változót előtagként annak érdekében, hogy engedélyezze a jegyszámok felismerését a bejövő e-mail válaszoknál. Egy jegyszám előállító modulnak a következő két függvényre van szüksége: TicketCreateNumber() és GetTNByString().

A TicketCreateNumber() metódus paraméterek nélkül kerül meghívásra, és az új jegyszámot adja vissza.

A GetTNByString() metódus egy olyan szöveg paraméterrel kerül meghívásra, amely a feldolgozandó szöveget tartalmazza a jegyszámnál, és visszaadja a jegyszámot, ha megtalálta.

2.3.4.1. Kódpélda

Nézze meg a Kernel/System/Ticket/Number/UserRandom.pm fájlt a TemplateModule csomagban.

2.3.4.2. Beállítási példa

Nézze meg a Kernel/Config/Files/TicketNumberGenerator.xml fájlt a TemplateModule csomagban.

2.3.4.3. Használati esetek

2.3.4.3.1. A jegyszámoknak egy bizonyos sémát kell követniük.

Akkor kell majd egy új jegyszám előállítót létrehoznia, ha az alapértelmezett modulok nem biztosítják azt a jegyszám sémát, amelyet használni szeretne.

2.3.4.4. Ellenjavaslatok és figyelmeztetések

Ragaszkodnia kell a meglévő jegyszám előállítókban használt GetTNByString() kódjához, hogy megelőzze a jegyszám feldolgozással kapcsolatos problémákat. A TicketCreateNumber() metódusban lévő hurok felismeréséhez használt rutint is érintetlenül kell hagynia a kettőzött jegyszámok megelőzéséhez.

2.3.4.5. Kiadási elérhetőség

A jegyszám előállítók az OTRS 1.1 óta lettek elérhetők az OTRS-ben.

2.3.5. Jegyesemény modul

A jegyesemény modulok közvetlenül azután futnak le, amikor egy jegyművelet megtörténik. Megegyezés szerint ezek a modulok a Kernel/System/Ticket/Event könyvtárban találhatók. Egy jegyesemény modulnak mindössze két függvényre van szüksége: new() és Run(). A Run() metódus legalább az Event, a UserID és a Data paramétereket fogadja. A Data a jegy adatait tartalmazó kivonathivatkozás, és a bejegyzésre vonatkozó események esetében a bejegyzés adatait is tartalmazza.

2.3.5.1. Kódpélda

Nézze meg a Kernel/System/Ticket/Event/EventModulePostTemplate.pm fájlt a TemplateModule csomagban.

2.3.5.2. Beállítási példa

Nézze meg a Kernel/Config/Files/EventModulePostTemplate.xml fájlt a TemplateModule csomagban.

2.3.5.3. Használati esetek

2.3.5.3.1. Egy jegyet fel kell oldani egy áthelyezés művelet után.

Ez a szabványos funkció a `Kernel::System::Ticket::Event::ForceUnlock` jegyesemény modullal lett megvalósítva. Amikor erre a funkcióra nincs szükség, akkor az kikapcsolható a `Ticket::EventModulePost###910-ForceUnlockOnMove` rendszerbeállítási bejegyzés beállításának törlésével.

2.3.5.3.2. További tisztítóművelet végrehajtása egy jegy törlésekor.

Egy személyre szabott OTRS tarthat nem szabványos adatokat további adatbázistáblákban. Amikor egy jegyet törölnek, akkor ezeket a további adatokat is törölni kell. Ez a funkcionalitás elérhető egy olyan jegyesemény modullal, amely a `TicketDelete` eseményekre figyel.

2.3.5.3.3. Az új jegyeket közzé kell tenni a Twitteren.

Egy `TicketCreate` eseményre figyelő jegyesemény modul képes üzeneteket kiküldeni a Twitterre.

2.3.5.4. Ellenjavaslatok és figyelmeztetések

Nincsenek ismert ellenjavaslatok.

2.3.5.5. Kiadási elérhetőség

A jegyesemények az OTRS 2.0 óta lettek elérhetők az OTRS-ben.

Ticket events available in OTRS 6.0:

- TicketCreate
- TicketDelete
- TicketTitleUpdate
- TicketUnlockTimeoutUpdate
- TicketQueueUpdate
- TicketTypeUpdate
- TicketServiceUpdate
- TicketSLAUpdate
- TicketCustomerUpdate
- TicketPendingTimeUpdate
- TicketLockUpdate
- TicketArchiveFlagUpdate
- TicketStateUpdate
- TicketOwnerUpdate
- TicketResponsibleUpdate
- TicketPriorityUpdate
- HistoryAdd
- HistoryDelete

- TicketAccountTime
- TicketMerge
- TicketSubscribe
- TicketUnsubscribe
- TicketFlagSet
- TicketFlagDelete
- EscalationResponseTimeNotifyBefore
- EscalationUpdateTimeNotifyBefore
- EscalationSolutionTimeNotifyBefore
- EscalationResponseTimeStart
- EscalationUpdateTimeStart
- EscalationSolutionTimeStart
- EscalationResponseTimeStop
- EscalationUpdateTimeStop
- EscalationSolutionTimeStop
- NotificationNewTicket
- NotificationFollowUp
- NotificationLockTimeout
- NotificationOwnerUpdate
- NotificationResponsibleUpdate
- NotificationAddNote
- NotificationMove
- NotificationPendingReminder
- NotificationEscalation
- NotificationEscalationNotifyBefore
- NotificationServiceUpdate

Article events available in OTRS 6.0:

- ArticleCreate
- ArticleUpdate
- ArticleSend
- ArticleBounce
- ArticleAgentNotification
- ArticleCustomerNotification
- ArticleAutoResponse
- ArticleFlagSet

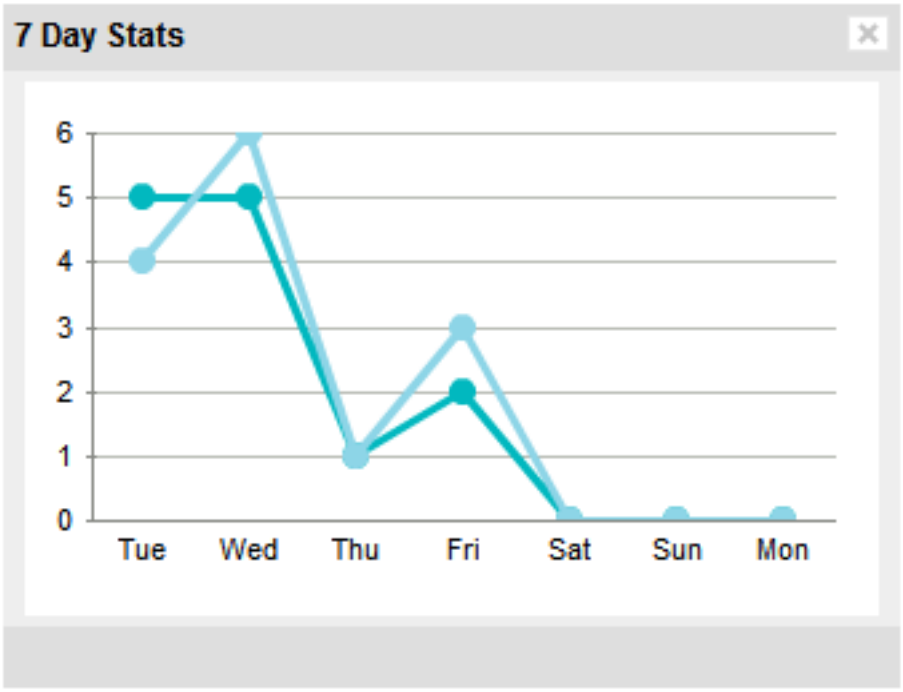
- ArticleFlagDelete
- ArticleAgentNotification
- ArticleCustomerNotification

2.4. Előtétprogram modulok

2.4.1. Vezérlőpult modul

Vezérlőpult modul statisztikák megjelenítéséhez vonaldiagram formájában.

3.1. ábra - Vezérlőpult felületi elem




```
# --
# Kernel/Output/HTML/DashboardTicketStatsGeneric.pm - message of the day
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::DashboardTicketStatsGeneric;

use strict;
use warnings;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    # get needed objects
    for (
        qw(Config Name ConfigObject LogObject DBObject LayoutObject ParamObject TicketObject
 UserID)
        )
```

```
    {
        die "Got no $_!" if !$Self->{$_};
    }

    return $Self;
}

sub Preferences {
    my ( $Self, %Param ) = @_;

    return;
}

sub Config {
    my ( $Self, %Param ) = @_;

    my $Key = $Self->{LayoutObject}->{UserLanguage} . '-' . $Self->{Name};
    return (
        %{ $Self->{Config} },
        CacheKey => 'TicketStats' . '-' . $Self->{UserID} . '-' . $Key,
    );

}

sub Run {
    my ( $Self, %Param ) = @_;

    my %Axis = (
        '7Day' => {
            0 => { Day => 'Sun', Created => 0, Closed => 0, },
            1 => { Day => 'Mon', Created => 0, Closed => 0, },
            2 => { Day => 'Tue', Created => 0, Closed => 0, },
            3 => { Day => 'Wed', Created => 0, Closed => 0, },
            4 => { Day => 'Thu', Created => 0, Closed => 0, },
            5 => { Day => 'Fri', Created => 0, Closed => 0, },
            6 => { Day => 'Sat', Created => 0, Closed => 0, },
        },
    );

    my @Data;
    my $Max = 1;
    for my $Key ( 0 .. 6 ) {

        my $TimeNow = $Self->{TimeObject}->SystemTime();
        if ($Key) {
            $TimeNow = $TimeNow - ( 60 * 60 * 24 * $Key );
        }
        my ( $Sec, $Min, $Hour, $Day, $Month, $Year, $WeekDay )
            = $Self->{TimeObject}->SystemTime2Date(
            SystemTime => $TimeNow,
            );

        $Data[$Key]->{Day} = $Self->{LayoutObject}->{LanguageObject}->Get(
            $Axis{'7Day'}->{$WeekDay}->{Day}
        );

        my $CountCreated = $Self->{TicketObject}->TicketSearch(

            # cache search result 20 min
            CacheTTL => 60 * 20,

            # tickets with create time after ... (ticket newer than this date) (optional)
            TicketCreateTimeNewerDate => "$Year-$Month-$Day 00:00:00",

            # tickets with created time before ... (ticket older than this date) (optional)
            TicketCreateTimeOlderDate => "$Year-$Month-$Day 23:59:59",

            CustomerID => $Param{Data}->{UserCustomerID},
            Result     => 'COUNT',

            # search with user permissions
            Permission => $Self->{Config}->{Permission} || 'ro',
```

```
            UserID => $Self->{UserID},
        );
        $Data[$Key]->{Created} = $CountCreated;
        if ( $CountCreated > $Max ) {
            $Max = $CountCreated;
        }

        my $CountClosed = $Self->{TicketObject}->TicketSearch(

            # cache search result 20 min
            CacheTTL => 60 * 20,

            # tickets with create time after ... (ticket newer than this date) (optional)
            TicketCloseTimeNewerDate => "$Year-$Month-$Day 00:00:00",

            # tickets with created time before ... (ticket older than this date) (optional)
            TicketCloseTimeOlderDate => "$Year-$Month-$Day 23:59:59",

            CustomerID => $Param{Data}->{UserCustomerID},
            Result     => 'COUNT',

            # search with user permissions
            Permission => $Self->{Config}->{Permission} || 'ro',
            UserID => $Self->{UserID},
        );
        $Data[$Key]->{Closed} = $CountClosed;
        if ( $CountClosed > $Max ) {
            $Max = $CountClosed;
        }
    }

    @Data = reverse @Data;
    my $Source = $Self->{LayoutObject}->JSONEncode(
        Data => \@Data,
    );

    my $Content = $Self->{LayoutObject}->Output(
        TemplateFile => 'AgentDashboardTicketStats',
        Data         => {
            %{ $Self->{Config} },
            Key    => int rand 99999,
            Max    => $Max,
            Source => $Source,
        },
    );

    return $Content;
}

1;
```

Ezen modul használatához adja hozzá a következőket a `Kernel/Config.pm` fájlhoz, és indítsa újra a webkiszolgálóját (ha a `mod_perl` modult használja).

```
<ConfigItem Name="DashboardBackend###0250-TicketStats" Required="0" Valid="1">
    <Description Lang="en">Parameters for the dashboard backend. "Group" are used to
 restricted access to the plugin (e. g. Group: admin;group1;group2;). "Default" means if the
 plugin is enabled per default or if the user needs to enable it manually. "CacheTTL" means
 the cache time in minutes for the plugin.</Description>
    <Description Lang="hu">Paraméterek a vezérlőpult háttérprogramhoz. A
 „Csoport” használható a hozzáférés korlátozásához a bővítményre (például Csoport:
 admin;csoport1;csoport2;). Az „Alapértelmezett” azt jelenti, hogy a bővítmény
 alapértelmezetten engedélyezve van, vagy hogy a felhasználónak kézzel kell engedélyeznie
 azt. A „CacheTTL” a bővítmény gyorsítótár lejárati időtartamát jelenti percben.</
Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Agent::Dashboard</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::DashboardTicketStatsGeneric</Item>
```

```
            <Item Key="Title">7 Day Stats</Item>
            <Item Key="Created">1</Item>
            <Item Key="Closed">1</Item>
            <Item Key="Permission">rw</Item>
            <Item Key="Block">ContentSmall</Item>
            <Item Key="Group"></Item>
            <Item Key="Default">1</Item>
            <Item Key="CacheTTL">45</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.4.1.1. Ellenjavaslatok és figyelmeztetések

A napok vagy az önálló sorok túlzott száma teljesítmény-csökkenéshez vezethet.

2.4.1.2. Kiadási elérhetőség

A 2.4.0-ás verziótól.

2.4.2. Értesítési modul

Az értesítési modulokat egy értesítés megjelenítéséhez használják a fő navigáció alatt. Megírhatja és regisztrálhatja a saját értesítési modulját. Jelenleg 5 jegymenü van az OTRS keretrendszerben.

- `AgentOnline`
- `AgentTicketEscalation`
- `CharsetCheck`
- `CustomerOnline`
- `UIDCheck`

2.4.2.1. Kódpélda

Az értesítési modulok a `Kernel/Output/HTML/TicketNotification*.pm` alatt találhatók. Ezt követően egy értesítőmodul példája található. Mentse el a `Kernel/Output/HTML/TicketNotificationCustom.pm` fájlba. Mindössze két függvényre van szüksége: `new()` és `Run()`.

```
# --
# Kernel/Output/HTML/NotificationCustom.pm
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::NotificationCustom;

use strict;
use warnings;

use Kernel::System::Custom;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
```

```
    for my $Object (qw(ConfigObject LogObject DBObject LayoutObject TimeObject UserID)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }
    $Self->{CustomObject} = Kernel::System::Custom->new(%Param);
    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;

    # get session info
    my %CustomParam      = ();
    my @Customs    = $Self->{CustomObject}->GetAllCustomIDs();
    my $IdleMinutes = $Param{Config}->{IdleMinutes} || 60 * 2;
    for (@Customs) {
        my %Data = $Self->{CustomObject}->GetCustomIDData( CustomID => $_, );
        if (
            $Self->{UserID} ne $Data{UserID}
            && $Data{UserType} eq 'User'
            && $Data{UserLastRequest}
            && $Data{UserLastRequest} + ( $IdleMinutes * 60 ) > $Self->{TimeObject}-
>SystemTime()
            && $Data{UserFirstname}
            && $Data{UserLastname}
            )
        {
            $CustomParam{ $Data{UserID} } = "$Data{UserFirstname} $Data{UserLastname}";
            if ( $Param{Config}->{ShowEmail} ) {
                $CustomParam{ $Data{UserID} } .= " ($Data{UserEmail})";
            }
        }
    }
    for ( sort { $CustomParam{$a} cmp $CustomParam{$b} } keys %CustomParam ) {
        if ( $Param{Message} ) {
            $Param{Message} .= ', ';
        }
        $Param{Message} .= "$CustomParam{$_}";
    }
    if ( $Param{Message} ) {
        return $Self->{LayoutObject}->Notify( Info => 'Custom Message: %s", "' .
 $Param{Message} );
    }
    else {
        return '';
    }
}

1;
```

2.4.2.2. Beállítási példa

Szükség van az egyéni értesítési modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni. Lehetnek további paraméterek is a beállítás kivonatában az értesítési moduljánál.

```
<ConfigItem Name="Frontend::NotifyModule###3-Custom" Required="0" Valid="0">
    <Description Lang="en">Module to show custom message in the agent interface.</
Description>
    <Description Lang="hu">Egy modul egyéni üzenet megjelenítéséhez az ügyintézői
 felületen.</Description>
    <Group>Framework</Group>
    <SubGroup>Frontend::Agent::ModuleNotify</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::NotificationCustom</Item>
            <Item Key="Key1">1</Item>
            <Item Key="Key2">2</Item>
        </Hash>
```

```
    </Setting>
</ConfigItem>
```

2.4.2.3. Használati eset példa

Hasznos jegymenü megvalósítás lehet egy hivatkozás egy külső eszközre, ha a paraméterek (például `FreeTextField`) be lettek állítva.

2.4.2.4. Kiadási elérhetőség

Név	Kiadás
NotificationAgentOnline	2.0
NotificationAgentTicketEscalation	2.0
NotificationCharsetCheck	1.2
NotificationCustomerOnline	2.0
NotificationUIDCheck	1.2

2.4.3. Jegymenü modul

A jegymenü modulokat egy további hivatkozás megjelenítéséhez használják a jegy fölött lévő menüben. Megírhatja és regisztrálhatja a saját jegymenü moduljait. Négy jegymenü létezik (Általános, Zárolás, Felelős és Jegymegfigyelő), amely az OTRS keretrendszerrel érkezik. További információkért nézzen bele az OTRS adminisztrációs kézikönyvébe.

2.4.3.1. Kódpélda

A jegymenü modulok a `Kernel/Output/HTML/TicketMenu*.pm` fájlokban találhatók. A következőkben egy jegymenü modul példája található. Mentse el a `Kernel/Output/HTML/TicketMenuCustom.pm` helyre. Mindössze két függvényre van szüksége: `new()` és `Run()`.

```
# --
# Kernel/Output/HTML/TicketMenuCustom.pm
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# Id: TicketMenuCustom.pm,v 1.17 2010/04/12 21:34:06 martin Exp $
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::TicketMenuCustom;

use strict;
use warnings;

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless( $Self, $Type );

    # get needed objects
    for my $Object (qw(ConfigObject LogObject DBObject LayoutObject UserID TicketObject)) {
        $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
    }

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;
```

```
    # check needed stuff
    if ( !$Param{Ticket} ) {
        $Self->{LogObject}->Log(
            Priority => 'error',
            Message  => 'Need Ticket!'
        );
        return;
    }

    # check if frontend module registered, if not, do not show action
    if ( $Param{Config}->{Action} ) {
        my $Module = $Self->{ConfigObject}->Get('Frontend::Module')->{ $Param{Config}-
>{Action} };
        return if !$Module;
    }

    # check permission
    my $AccessOk = $Self->{TicketObject}->Permission(
        Type     => 'rw',
        TicketID => $Param{Ticket}->{TicketID},
        UserID   => $Self->{UserID},
        LogNo    => 1,
    );
    return if !$AccessOk;

    # check permission
    if ( $Self->{TicketObject}->CustomIsTicketCustom( TicketID => $Param{Ticket}-
>{TicketID} ) ) {
        my $AccessOk = $Self->{TicketObject}->OwnerCheck(
            TicketID => $Param{Ticket}->{TicketID},
            OwnerID  => $Self->{UserID},
        );
        return if !$AccessOk;
    }

    # check acl
    return
        if defined $Param{ACL}->{ $Param{Config}->{Action} }
            && !$Param{ACL}->{ $Param{Config}->{Action} };

    # if ticket is customized
    if ( $Param{Ticket}->{Custom} eq 'lock' ) {

        # if it is locked for somebody else
        return if $Param{Ticket}->{OwnerID} ne $Self->{UserID};

        # show custom action
        return {
            %{ $Param{Config} },
            %{ $Param{Ticket} },
            %Param,
            Name        => 'Custom',
            Description => 'Custom to give it back to the queue!',
            Link        => 'Action=AgentTicketCustom;Subaction=Custom;TicketID=
$QData{"TicketID"}',
        };
    }

    # if ticket is customized
    return {
        %{ $Param{Config} },
        %{ $Param{Ticket} },
        %Param,
        Name        => 'Custom',
        Description => 'Custom it to work on it!',
        Link        => 'Action=AgentTicketCustom;Subaction=Custom;TicketID=
$QData{"TicketID"}',
    };
}

1;
```

2.4.3.2. Beállítási példa

Szükség van az egyéni jegymenü modul bekapcsolására. Ezt a lenti XML beállítás használatával lehet megtenni. Lehetnek további paraméterek is a beállítás kivonatában a jegymenü moduljánál.

```
<ConfigItem Name="Ticket::Frontend::MenuModule###110-Custom" Required="0" Valid="1">
    <Description Lang="en">Module to show custom link in menu.</Description>
    <Description Lang="hu">Egy modul egyéni hivatkozás megjelenítéséhez a menüben.</
Description>
    <Group>Ticket</Group>
    <SubGroup>Frontend::Agent::Ticket::MenuModule</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::TicketMenuCustom</Item>
            <Item Key="Name">Custom</Item>
            <Item Key="Action">AgentTicketCustom</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.4.3.3. Használati eset példa

Hasznos jegymenü megvalósítás lehet egy hivatkozás egy külső eszközre, ha a paraméterek (például `FreeTextField`) be lettek állítva.

2.4.3.4. Ellenjavaslatok és figyelmeztetések

A jegymenü egy olyan URL-re irányít, amely kezelhető. Ha ezt a kérést az OTRS keretrendszeren keresztül szeretné kezelni, akkor meg kell írnia a saját előtétprogram modulját.

2.4.3.5. Kiadási elérhetőség

Név	Kiadás
TicketMenuGeneric	2.0
TicketMenuLock	2.0
TicketMenuResponsible	2.1
TicketMenuTicketWatcher	2.4

2.5. Általános felület modulok

2.5.1. Hálózati átvitel

A hálózati átvitelt használják az információk küldésének és fogadásának módszereként az OTRS és egy távoli rendszer között. Az általános felület beállításai lehetővé teszik egy webszolgáltatásnak, hogy különböző hálózati átviteli modulokat használjon a szolgáltatónál és a kérelmezőnél, de a leggyakoribb forgatókönyv az, hogy ugyanazt az átviteli modult használják mindkettőnél.

OTRS mint szolgáltató:

Az OTRS arra használja a hálózati átviteli modulokat, hogy lekérje az adatokat a távoli rendszertől, valamint lekérje a végrehajtandó műveleteket. A művelet végrehajtása után az OTRS ismét azokat használja a válasz visszaküldéséhez a távoli rendszernek.

OTRS mint kérelmező:

Az OTRS arra használja a hálózati átviteli modulokat, hogy kérelmeket küldjön a távoli rendszernek egy távoli művelet végrehajtásához a szükséges adatok mellett. Az OTRS várakozik a távoli rendszer válaszára, és visszaküldi azt a kérelmező modulnak.

Mindkét irányban a hálózati átviteli modulok foglalkoznak a távoli rendszer formátumában lévő adatokkal. Nem ajánlott semmilyen adatátalakítás végrehajtása sem ezekben a modulokban, mivel a leképező réteg felelős a kommunikáció során szükséges bármilyen adatátalakítás végrehajtásáért. Egy kivétel erre az olyan adatátalakítás, amely kifejezetten az átvitelnél szükséges, például XML vagy JSON és Perl átalakítások.

2.5.1.1. Átviteli háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új átviteli háttérprogramot kifejleszteni. Minden egyes átviteli háttérprogramnak meg kell valósítania ezeket a szubrutinokat:

- new
- `ProviderProcessRequest`
- `ProviderGenerateResponse`
- `RequesterPerformRequest`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk mindkét irányban helyesen kommunikálni egy távoli rendszerrel. Az összes átviteli háttérprogramot az átviteli modul kezeli (`Kernel/GenericInterface/Transport.pm`).

Jelenleg az általános felület megvalósítja a HTTP SOAP és a HTTP REST átviteleket. Ha a tervezett webszolgáltatás használhat HTTP SOAP vagy HTTP SOAP átviteleket, akkor nincs szükség egy új hálózati átviteli modul létrehozására, hanem ahelyett azt ajánljuk, hogy vessen egy pillantást a HTTP SOAP vagy HTTP REST konfigurációira a beállításaik ellenőrzéséhez, valamint hogy hogyan hangolhatók a távoli rendszernek megfelelően.

2.5.1.1.1. Kódpélda

Abban az esetben, ha a biztosított hálózati átvitelek nem illeszkednek a webszolgáltatás igényeire, akkor ebben a szakaszban egy minta hálózati átviteli modul van bemutatva, és minden egyes szubrutin elmagyarázásra kerül. Normális esetben az átviteli modulok CPAN modulokat használnak háttérprogramokként. Például a HTTP SOAP átviteli modulok a `SOAP::Lite` modult használják háttérprogramként.

Ennél a példánál egy egyéni csomagot használnak az adatok visszaadásához anélkül, hogy valódi hálózati kérést intéznének egy távoli rendszerhez, ehelyett ez az egyéni modul működik visszacsatolási felületként.

```
# --
# Kernel/GenericInterface/Transport/HTTP/Test.pm - GenericInterface network transport
 interface for testing
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Transport::HTTP::Test;

use strict;
use warnings;

use HTTP::Request::Common;
use LWP::UserAgent;
use LWP::Protocol;

# prevent 'Used once' warning for Kernel::OM
```

```
use Kernel::System::ObjectManager;

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva. Az átvitelek nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    for my $Needed (qw( DebuggerObject TransportConfig)) {
        $Self->{$Needed} = $Param{$Needed} || return {
            Success      => 0,
            ErrorMessage => "Nincs $Needed!"
        };
    }

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub ProviderProcessRequest {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
            Success      => 0,
            ErrorMessage => "HTTP állapotkód: 500",
            Data         => {},
        };
    }

    my $ParamObject = $Kernel::OM->Get('Kernel::System::Web::Request');

    my %Result;
    for my $ParamName ( $ParamObject->GetParamNames() ) {
        $Result{$ParamName} = $ParamObject->GetParam( Param => $ParamName );
    }

    # különleges kezelés az üres POST kérésnél
    if ( scalar keys %Result == 1 && exists $Result{POSTDATA} && !$Result{POSTDATA} ) {
        %Result = ();
    }

    if ( !%Result ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Nem található kért adat.',
        );
    }

    return {
        Success   => 1,
        Data      => \%Result,
        Operation => 'test_operation',
    };
}
```

A `ProviderProcessRequest` függvény megkapja a kérést a távoli kiszolgálótól (ebben az esetben ugyanaz az OTRS), és kibontja az adatokat és a műveletet a kérésből a végrehajtáshoz. Ennél a példánál a művelet mindig a `test_operation`.

Annak a módja, ahogy ez a függvény feldolgozza a kérést az adatok és a művelet nevének lekéréséhez, az teljes egészében a megvalósítandó protokolltól és azon külső moduloktól függ, amelyekhez használják azokat.

```
sub ProviderGenerateResponse {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
            Success      => 0,
            ErrorMessage => 'A tesztválasz előállítása sikertelen.',
        };
    }

    my $Response;

    if ( !$Param{Success} ) {
        $Response
            = HTTP::Response->new( 500 => ( $Param{ErrorMessage} || 'Internal Server
 Error' ) );
        $Response->protocol('HTTP/1.0');
        $Response->content_type("text/plain; charset=UTF-8");
        $Response->date(time);
    }
    else {

        # egy kérésszöveg előállítása az adatokból
        my $Request
            = HTTP::Request::Common::POST( 'http://testhost.local/', Content =>
 $Param{Data} );

        $Response = HTTP::Response->new( 200 => "OK" );
        $Response->protocol('HTTP/1.0');
        $Response->content_type("text/plain; charset=UTF-8");
        $Response->add_content_utf8( $Request->content() );
        $Response->date(time);
    }

    $Self->{DebuggerObject}->Debug(
        Summary => 'HTTP-válasz küldése',
        Data    => $Response->as_string(),
    );

    # a válasz elküldése a kliensnek most
    print STDOUT $Response->as_string();

    return {
        Success => 1,
    };
}
```

Ez a függvény visszaküldi a választ a távoli rendszernek a kért művelethez.

Ennél a bizonyos példánál minden esetben egy szabványos sikeres HTTP-választ adunk vissza (200) vagy nem (500) a szükséges adatok mellett.

```
sub RequesterPerformRequest {
    my ( $Self, %Param ) = @_;

    if ( $Self->{TransportConfig}->{Config}->{Fail} ) {

        return {
```

```
            Success      => 0,
            ErrorMessage => "HTTP állapotkód: 500",
            Data         => {},
        };
    }

    # egyéni protokollkezelő használata a valódi hálózati kérések kiküldésének elkerüléséhez
    LWP::Protocol::implementor(
        testhttp => 'Kernel::GenericInterface::Transport::HTTP::Test::CustomHTTPProtocol'
    );
    my $UserAgent = LWP::UserAgent->new();
    my $Response = $UserAgent->post( 'testhttp://localhost.local/', Content =>
 $Param{Data} );

    return {
        Success => 1,
        Data    => {
            ResponseContent => $Response->content(),
        },
    };
}
```

Ez az egyetlen olyan függvény, amelyet az OTRS mint kérelmező használ. Elküldi a kérést a távoli rendszernek, és várakozik annak válaszára.

Ennél a példánál egy egyéni protokollkezelőt használunk a valódi hálózatra történő kérésküldés elkerüléséhez. Ez az egyéni protokoll az alábbiakban van megadva.

```
package Kernel::GenericInterface::Transport::HTTP::Test::CustomHTTPProtocol;

use base qw(LWP::Protocol);

sub new {
    my $Class = shift;

    return $Class->SUPER::new(@_);
}

sub request {    ## nem kritikus
    my $Self = shift;

    my ( $Request, $Proxy, $Arg, $Size, $Timeout ) = @_;

    my $Response = HTTP::Response->new( 200 => "OK" );
    $Response->protocol('HTTP/1.0');
    $Response->content_type("text/plain; charset=UTF-8");
    $Response->add_content_utf8( $Request->content() );
    $Response->date(time);

    #print $Request->as_string();
    #print $Response->as_string();

    return $Response;
}
```

Ez a kód ahhoz az egyéni protokollhoz van, amelyet használunk. Ez a megközelítés csak gyakorlásnál vagy olyan tesztelési környezeteknél hasznos, ahol a távoli rendszerek nem érhetők el.

Egy új modul kifejlesztéséhez nem ajánljuk ezen megközelítés használatát, egy valódi protokollt kell megvalósítani.

2.5.1.1.2. Beállítási példa

Szükség van ezen hálózati átviteli modul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Transport::Module###HTTP::Test" Required="0" Valid="1">
    <Description Translatable="1">GenericInterface module registration for the transport
 layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Transport::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Protocol">HTTP</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceTransportHTTPTest</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.5.2. Leképezés

A leképezést adatok átalakításához használják az OTRS és a távoli rendszer között, illetve fordítva. Ezek az adatok kulcs => érték párokként vannak ábrázolva. Leképező modulok fejleszthetők ki nem csak az értékek, hanem a kulcsok átalakításához is.

Például:

Erről	**Erre**
Prio => Warning	PriorityID => 3

A leképező réteg nem feltétlenül szükséges, egy webszolgáltatás teljesen kihagyhatja azt a webszolgáltatás beállításaitól, valamint a meghívók és műveletek megvalósításának módjától függően. De ha egy kis átalakítás szükséges, akkor erősen ajánlott egy meglévő leképezőmodul használata, vagy egy új létrehozása.

A leképező modulok egynél több alkalommal is meghívhatók egy normál kommunikáció közben. Vessen egy pillantást a következő példákra.

OTRS mint szolgáltató példa:

1. A távoli rendszer elküldi a kérést az adatokkal a távoli rendszer formátumában
2. Az adatok leképezésre kerülnek a távoli rendszer formátumáról az OTRS formátumára
3. Az OTRS végrehajtja a műveletet, és visszaadja a választ az OTRS formátumában
4. Az adatok leképezésre kerülnek az OTRS formátumáról a távoli rendszer formátumára
5. A válasz a távoli rendszer formátumában lévő adatokkal elküldésre kerül a távoli rendszernek

OTRS mint kérelmező példa:

1. Az OTRS előkészíti a kérést a távoli rendszerhez az OTRS formátumában lévő adatokkal
2. Az adatok leképezésre kerülnek az OTRS formátumáról a távoli rendszer formátumára
3. A kérés elküldésre kerül a távoli rendszernek, amely végrehajtja a műveletet, és visszaküldi a választ az OTRS-nek a távoli rendszer formátumában lévő adatokkal
4. Az adatok (ismét) leképezésre kerülnek a távoli rendszer formátumáról az OTRS formátumára
5. Az OTRS feldolgozza a választ

2.5.2.1. Leképező háttérprogram

Az általános felület biztosít egy *Simple* nevű leképezőmodult. Ezzel a modullal a legtöbb adatátalakítás (beleértve a kulcs és érték leképezést) elvégezhető, és szabályokat is

meghatároz az alapértelmezett leképezések kezeléséhez mind a kulcsoknál, mind az értékeknél.

Ezért erősen valószínű, hogy nem lesz szüksége egy egyéni leképezőmodul kifejlesztésére. A folytatás előtt nézze meg a *Simple* leképezőmodult (`Kernel/GenericInterface/Mapping/Simple.pm`) és annak internetes dokumentációját.

Ha a *Simple* leképezőmodul nem felel meg az igényeinek, akkor meg fogjuk mutatni, hogy hogyan lehet kifejleszteni egy új leképező háttérprogramot. Minden egyes leképező háttérprogramnak meg kell valósítania ezeket a szubrutinokat:

- new
- Map

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk az adatok leképezésére a kommunikációban, amelyet vagy a kérelmező, vagy a szolgáltató kezel. Az összes leképező háttérprogramot a leképezőmodul kezeli (`Kernel/GenericInterface/Mapping.pm`).

2.5.2.1.1. Kódpélda

Ebben a szakaszban egy minta leképezőmodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Kernel/GenericInterface/Mapping/Test.pm - GenericInterface test data mapping backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Mapping::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsHashRefWithData IsStringWithData);

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

Felveszünk egy `VariableCheck` modult is bizonyos ellenőrzések végrehajtásához néhány változón. A leképezések nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    # új kivonat lefoglalása az objektumhoz
    my $Self = {};
    bless( $Self, $Type );

    # a szükséges paraméterek ellenőrzése
    for my $Needed (qw(DebuggerObject MappingConfig)) {
        if ( !$Param{$Needed} ) {

            return {
                Success      => 0,
                ErrorMessage => "Nincs $Needed!"
            };
        }
```

```
        $Self->{$Needed} = $Param{$Needed};
    }

    # leképezési beállítás ellenőrzése
    if ( !IsHashRefWithData( $Param{MappingConfig} ) ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Nincs MappingConfig objektum kivonathivatkozásként tartalommal!',
        );
    }

    # beállítás ellenőrzése - ha van leképezési beállításunk, akkor annak
    # nem üres kivonathivatkozásnak kell lennie
    if (
        defined $Param{MappingConfig}->{Config}
        && !IsHashRefWithData( $Param{MappingConfig}->{Config} )
        )
    {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Van MappingConfig adatokkal, de az adat nem kivonathivatkozás
 tartalommal!',
        );
    }

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Map {
    my ( $Self, %Param ) = @_;

    # adatok ellenőrzése - csak meghatározatlant vagy kivonathivatkozást fogad el
    if ( defined $Param{Data} && ref $Param{Data} ne 'HASH' ) {

        return $Self->{DebuggerObject}->Error(
            Summary => 'Van adat, de az nem kivonathivatkozás a leképezésteszt
 háttérprogramban!'
        );
    }

    # visszatérés, ha az adat üres
    if ( !defined $Param{Data} || !%{ $Param{Data} } ) {

        return {
            Success => 1,
            Data    => {},
        };
    }

    # ha nincs beállítás, akkor az azt jelenti, hogy egyszerűen visszaadjuk a bemeneti
 adatot
    if (
        !defined $Self->{MappingConfig}->{Config}
        || !defined $Self->{MappingConfig}->{Config}->{TestOption}
        )
    {

        return {
            Success => 1,
            Data    => $Param{Data},
        };
    }

    # a TestOption formátum ellenőrzése
    if ( !IsStringWithData( $Self->{MappingConfig}->{Config}->{TestOption} ) ) {
```

```
        return $Self->{DebuggerObject}->Error(
            Summary => 'Nincs TestOption szövegként értékkel!',
        );
    }

    # adatok feldolgozása a beállítások szerint
    my $ReturnData = {};
    if ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'ToUpper' ) {
        $ReturnData = $Self->_ToUpper( Data => $Param{Data} );
    }
    elsif ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'ToLower' ) {
        $ReturnData = $Self->_ToLower( Data => $Param{Data} );
    }
    elsif ( $Self->{MappingConfig}->{Config}->{TestOption} eq 'Empty' ) {
        $ReturnData = $Self->_Empty( Data => $Param{Data} );
    }
    else {
        $ReturnData = $Param{Data};
    }

    # az eredmény visszaadása
    return {
        Success => 1,
        Data    => $ReturnData,
    };
}
```

A Map függvény az egyes leképezőmodulok fő része. Fogadja a leképezési beállításokat (szabályokat) és az eredeti formátumban lévő adatokat (vagy az OTRS vagy a távoli rendszer formátumában lévőket), és átalakítja azokat egy új formátumra még akkor is, ha az adatok szerkezete megváltozhat a leképezési folyamat során.

Ebben a bizonyos példában három szabály van az értékek leképezéséhez. Ezek a szabályok a leképezési beállítások `TestOption` kulcsában vannak beállítva, és a következők: `ToUpper`, `ToLower` és `Empty`.

- `ToUpper`: nagybetűsre alakít át minden egyes adatértéket.
- `ToLower`: kisbetűsre alakít át minden egyes adatértéket.
- `Empty`: egy üres szövegre alakít át minden egyes adatértéket.

Ebben a példában nem lettek adatkulcs átalakítások megvalósítva.

```
sub _ToUpper {
    my ( $Self, %Param ) = @_;

    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = uc $Param{Data}->{$Key};
    }

    return $ReturnData;
}

sub _ToLower {
    my ( $Self, %Param ) = @_;

    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = lc $Param{Data}->{$Key};
    }

    return $ReturnData;
}

sub _Empty {
    my ( $Self, %Param ) = @_;
```

```
    my $ReturnData = {};
    for my $Key ( sort keys %{ $Param{Data} } ) {
        $ReturnData->{$Key} = '';
    }

    return $ReturnData;
}
```

Ezek azok a segédfüggvények, amelyek ténylegesen végrehajtják a szövegátalakításokat.

2.5.2.1.2. Beállítási példa

Szükség van ezen leképezőmodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Mapping::Module###Test" Required="0" Valid="1">
    <Description Translatable="1">GenericInterface module registration for the mapping
 layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Mapping::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="ConfigDialog"></Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.5.3. Meghívó

A meghívót arra használják, hogy egy kérést hozzon létre az OTRS-ből egy távoli rendszerhez. Az általános ügyintéző ezen része felelős a szükséges feladatok végrehajtásáért az OTRS oldalán, illetve a szükséges adatok begyűjtéséért a kérés felépítésének érdekében.

2.5.3.1. Meghívó háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új meghívót kifejleszteni. Minden egyes meghívónak meg kell valósítania ezeket a szubrutinokat:

- new
- `PrepareRequest`
- `HandleResponse`

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk végrehajtani egy kérést a kéréskezelő használatával (`Kernel/GenericInterface/Requester.pm`).

2.5.3.1.1. Kódpélda

Ebben a szakaszban egy minta meghívómodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Kernel/GenericInterface/Invoker/Test.pm - GenericInterface test data Invoker backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

```
package Kernel::GenericInterface::Invoker::Test::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsString IsStringWithData);

# prevent 'Used once' warning for Kernel::OM
use Kernel::System::ObjectManager;

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva. A meghívók nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    # új kivonat lefoglalása az objektumhoz
    my $Self = {};
    bless( $Self, $Type );

    # a szükséges paraméterek ellenőrzése
    if ( !$Param{DebuggerObject} ) {
        return {
            Success      => 0,
            ErrorMessage => "Nem sikerült lekérni a hibakezelő objektumot!"
        };
    }

    $Self->{DebuggerObject} = $Param{DebuggerObject};

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub PrepareRequest {
    my ( $Self, %Param ) = @_;

    # szükségünk van egy jegyszámra
    if ( !IsStringWithData( $Param{Data}->{TicketNumber} ) ) {
        return $Self->{DebuggerObject}->Error( Summary => 'Nem kaptunk jegyszámot' );
    }

    my %ReturnData;

    $ReturnData{TicketNumber} = $Param{Data}->{TicketNumber};

    # a művelet ellenőrzése
    if ( IsStringWithData( $Param{Data}->{Action} ) ) {
        $ReturnData{Action} = $Param{Data}->{Action} . 'Test';
    }

    # a rendszeridő kérésének ellenőrzése
    if ( IsStringWithData( $Param{Data}->{GetSystemTime} ) && $Param{Data}-
>{GetSystemTime} ) {
        $ReturnData{SystemTime} = $Kernel::OM->Get('Kernel::System::Time')->SystemTime();
    }

    return {
        Success => 1,
        Data    => \%ReturnData,
```

```
    };
}
```

A PrepareRequest függvényt használják a kérésbe küldendő összes szükséges adat kezeléséhez és összegyűjtéséhez. Itt fogadhatunk adatokat a kéréskezelőtől, használhatjuk azokat, kiterjeszthetjük azokat, új adatokat állíthatunk elő, és ezután átvihetjük az eredményeket a leképező réteghez.

Ennél a példánál azt várjuk, hogy kapunk egy jegyszámot. Ha nem, akkor az Error() hibakeresési metódust használjuk, amely létrehoz egy bejegyzést a hibakeresési naplóban, és visszaad egy szerkezetet is a Success paraméterrel 0-ként, és egy hibaüzenetet az átadott Summary értékként.

Ez a példa hozzáfűzi a „Test” szót is az Action paraméterhez, és ha a GetSystemTime kérve volt, akkor ki fogja tölteni a SystemTime paramétert az aktuális rendszeridővel. A kód ezen része azért van, hogy előkészítse az elküldendő adatokat. Egy valódi meghívónál itt kell elvégezni néhány hívást az alapmodulokhoz (Kernel/System/*.pm).

Ha a kérést a PrepareRequest függvény bármely része közben le kell állítani a hibakeresési naplóba való bejegyzés előállítása és hibajelzése nélkül, akkor a következő kód használható:

```
# a kérelmező kommunikációjának leállítása
return {
    Success           => 1,
    StopCommunication => 1,
};
```

Ennek használatával a kérelmező meg foga érteni, hogy a kérést nem szabad folytatni (nem kerül elküldésre a leképező réteghez, és nem kerül elküldésre a hálózati átvitelhez sem). A kérelmező nem fog hibát küldeni a hibakeresési naplóba, hanem csak csendben le fog állni.

```
sub HandleResponse {
    my ( $Self, %Param ) = @_;

    # ha hiba volt a válaszban, akkor továbbítja
    if ( !$Param{ResponseSuccess} ) {
        if ( !IsStringWithData( $Param{ResponseErrorMessage} ) ) {

            return $Self->{DebuggerObject}->Error(
                Summary => 'Hiba volt a válaszban, de nincs válasz hibaüzenet!',
            );
        }

        return {
            Success      => 0,
            ErrorMessage => $Param{ResponseErrorMessage},
        };
    }

    # szükségünk van egy jegyszámra
    if ( !IsStringWithData( $Param{Data}->{TicketNumber} ) ) {

        return $Self->{DebuggerObject}->Error( Summary => 'Nem kaptunk jegyszámot!' );
    }

    # a jegyszám előkészítése
    my %ReturnData = (
        TicketNumber => $Param{Data}->{TicketNumber},
    );

    # a művelet ellenőrzése
    if ( IsStringWithData( $Param{Data}->{Action} ) ) {
```

```
        if ( $Param{Data}->{Action} !~ m{ \A ( .*? ) Test \z }xms ) {

            return $Self->{DebuggerObject}->Error(
                Summary => 'Kaptunk műveletet, nem az nem megfelelő formátumú!',
            );
        }
        $ReturnData{Action} = $1;
    }

    return {
        Success => 1,
        Data    => \%ReturnData,
    };
}
```

A `HandleResponse` függvényt használják az előző kérésből származó adatok fogadásához és feldolgozásához, amelyet a távoli rendszernek készítettek. Ezeket az adatokat már átadta a leképező réteg, hogy átalakítsa azokat a távoli rendszer formátumáról az OTRS formátumára (ha szükséges).

Ennél a bizonyos példánál ismét ellenőrzi a jegyszámot, és azt is ellenőrzi, hogy a művelet a „Test” szóval végződik-e (amint az a `PrepareRequest` függvényben történt).

Megjegyzés

Ez a meghívó csak tesztelésekhez van, egy valódi meghívó ellenőrizni fogja, hogy a válasz a távoli rendszer által leírt formátumban volt-e, és végrehajthat néhány műveletet, mint például: egy másik meghívó meghívása, egy hívás végrehajtása egy alapmodulhoz, az adatbázis frissítése, hiba küldése, stb.

2.5.3.1.2. Beállítási példa

Szükség van ezen meghívómodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Invoker::Module###Test::Test" Required="0" Valid="1">
    <Description Translatable="1">GenericInterface module registration for the invoker
 layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Invoker::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Controller">Test</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceInvokerDefault</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.5.4. Művelet

A műveletet egy tevékenység végrehajtásához használják az OTRS-en belül. Ezt a tevékenységet a távoli rendszer kéri, és tartalmazhat különleges paramétereket azért, hogy helyesen végrehajtsa a tevékenységet. A tevékenység végrehajtása után az OTRS elküld egy meghatározott megerősítést a távoli rendszernek.

2.5.4.1. Műveleti háttérprogram

Ezután be fogjuk mutatni, hogy hogyan kell egy új műveletet kifejleszteni. Minden egyes műveletnek meg kell valósítania ezeket a szubrutinokat:

- new
- Run

Meg kell valósítanunk ezen metódusok mindegyikét azért, hogy képesek legyünk végrehajtani a szolgáltató által kezelt műveletet (`Kernel/GenericInterface/Provider.pm`).

2.5.4.1.1. Kódpélda

Ebben a szakaszban egy minta műveletmodul lesz megjelenítve, és minden szubrutin elmagyarázásra kerül.

```
# --
# Kernel/GenericInterface/Operation/Test/Test.pm - GenericInterface test operation backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::GenericInterface::Operation::Test::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(IsHashRefWithData);

our $ObjectManagerDisabled = 1;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

Felveszünk egy `VariableCheck` modult is bizonyos ellenőrzések végrehajtásához néhány változón. A műveletek nem példányosíthatók az objektumkezelővel.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    # a szükséges objektumok ellenőrzése
    for my $Needed (qw(DebuggerObject)) {
        if ( !$Param{$Needed} ) {
            return {
                Success      => 0,
                ErrorMessage => "Nincs $Needed!"
            };
        }

        $Self->{$Needed} = $Param{$Needed};
    }

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint az objektumkezelő által nem kezelt más osztályoknak csak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # adatok ellenőrzése - csak meghatározatlant vagy kivonathivatkozást fogad el
    if ( defined $Param{Data} && ref $Param{Data} ne 'HASH' ) {

        return $Self->{DebuggerObject}->Error(
```

```
            Summary => 'Van adat, de az nem kivonathivatkozás a műveletteszt
 háttérprogramban!'
        );
    }

    if ( defined $Param{Data} && $Param{Data}->{TestError} ) {

        return {
            Success      => 0,
            ErrorMessage => "Hibaüzenet a következő hibakódhoz: $Param{Data}->{TestError}",
            Data         => {
                ErrorData => $Param{Data}->{ErrorData},
            },
        };
    }

    # adatok másolása
    my $ReturnData;

    if ( ref $Param{Data} eq 'HASH' ) {
        $ReturnData = \%{ $Param{Data} };
    }
    else {
        $ReturnData = undef;
    }

    # az eredmény visszaadása
    return {
        Success => 1,
        Data    => $ReturnData,
    };
}
```

A Run függvény az egyes műveletek fő része. Fogadja az összes belsőleg leképezett adatot a távoli rendszertől, amelyre a szolgáltatónak szüksége van a művelet végrehajtásához, végrehajtja a műveletet, és visszaadja az eredményt a szolgáltatónak a külső leképezéshez, valamint visszaszállítja a távoli rendszerhez.

Ez a bizonyos példa ugyanúgy adja vissza az adatokat, ahogy azok a távoli rendszertől jönnek, hacsak a `TestError` paraméter át nincs adva. Ebben az esetben egy hibát ad vissza.

2.5.4.1.2. Beállítási példa

Szükség van ezen műveletmodul regisztrálására, hogy elérhető legyen az OTRS grafikus felhasználói felületén. Ezt a lenti XML beállítás használatával lehet megtenni.

```
<ConfigItem Name="GenericInterface::Operation::Module###Test::Test" Required="0" Valid="1">
    <Description Translatable="1">GenericInterface module registration for the operation
 layer.</Description>
    <Group>GenericInterface</Group>
    <SubGroup>GenericInterface::Operation::ModuleRegistration</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Name">Test</Item>
            <Item Key="Controller">Test</Item>
            <Item Key="ConfigDialog">AdminGenericInterfaceOperationDefault</Item>
        </Hash>
    </Setting>
</ConfigItem>
```

2.5.4.1.3. Unit Test Example

Unit Test for Generic Interface operations does not differs from other unit tests but it is needed to consider testing locally, but also simulating a remote connection. It is a good practice to test both separately since results could be slightly different.

To learn more about unit tests, please take a look to the Unit Test Chapter.

The following is just the starting point for a unit test:

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

## no critic (Modules::RequireExplicitPackage)
use strict;
use warnings;
use utf8;

use vars (qw($Self));

use Kernel::GenericInterface::Debugger;
use Kernel::GenericInterface::Operation::Test::Test;

use Kernel::System::VariableCheck qw(:all);

# Skip SSL certificate verification (RestoreDatabase must not be used in this test).
$Kernel::OM->ObjectParamAdd(
    'Kernel::System::UnitTest::Helper' => {
        SkipSSLVerify => 1,
    },
);
my $Helper = $Kernel::OM->Get('Kernel::System::UnitTest::Helper');

# get a random number
my $RandomID = $Helper->GetRandomNumber();

# create a new user for current test
my $UserLogin = $Helper->TestUserCreate(
    Groups => ['users'],
);
my $Password = $UserLogin;

my $UserID = $Kernel::OM->Get('Kernel::System::User')->UserLookup(
    UserLogin => $UserLogin,
);

# set web-service name
my $WebserviceName = '-Test-' . $RandomID;

# create web-service object
my $WebserviceObject = $Kernel::OM->Get('Kernel::System::GenericInterface::Webservice');
$Self->Is(
    'Kernel::System::GenericInterface::Webservice',
    ref $WebserviceObject,
    "Create web service object",
);

my $WebserviceID = $WebserviceObject->WebserviceAdd(
    Name   => $WebserviceName,
    Config => {
        Debugger => {
            DebugThreshold => 'debug',
        },
        Provider => {
            Transport => {
                Type => '',
            },
        },
    },
    ValidID => 1,
    UserID  => 1,
);
```

```
$Self->True(
    $WebserviceID,
    "Added Web Service",
);

# get remote host with some precautions for certain unit test systems
my $Host = $Helper->GetTestHTTPHostname();

my $ConfigObject = $Kernel::OM->Get('Kernel::Config');

# prepare web-service config
my $RemoteSystem =
    $ConfigObject->Get('HttpType')
    . '://'
    . $Host
    . '/'
    . $ConfigObject->Get('ScriptAlias')
    . '/nph-genericinterface.pl/WebserviceID/'
    . $WebserviceID;

my $WebserviceConfig = {
    Description =>
        'Test for Ticket Connector using SOAP transport backend.',
    Debugger => {
        DebugThreshold => 'debug',
        TestMode       => 1,
    },
    Provider => {
        Transport => {
            Type   => 'HTTP::SOAP',
            Config => {
                MaxLength => 10000000,
                NameSpace => 'http://otrs.org/SoapTestInterface/',
                Endpoint  => $RemoteSystem,
            },
        },
        Operation => {
            Test => {
                Type => 'Test::Test',
            },
        },
    },
    Requester => {
        Transport => {
            Type   => 'HTTP::SOAP',
            Config => {
                NameSpace => 'http://otrs.org/SoapTestInterface/',
                Encoding  => 'UTF-8',
                Endpoint  => $RemoteSystem,
            },
        },
        Invoker => {
            Test => {
                Type => 'Test::TestSimple'
                ,    # requester needs to be Test::TestSimple in order to simulate a request
 to a remote system
            },
        },
    },
};

# update web-service with real config
# the update is needed because we are using
# the WebserviceID for the Endpoint in config
my $WebserviceUpdate = $WebserviceObject->WebserviceUpdate(
    ID      => $WebserviceID,
    Name    => $WebserviceName,
    Config  => $WebserviceConfig,
    ValidID => 1,
    UserID  => $UserID,
);
$Self->True(
```

```
    $WebserviceUpdate,
    "Updated Web Service $WebserviceID - $WebserviceName",
);

# debugger object
my $DebuggerObject = Kernel::GenericInterface::Debugger->new(
    DebuggerConfig => {
        DebugThreshold => 'debug',
        TestMode       => 1,
    },
    WebserviceID      => $WebserviceID,
    CommunicationType => 'Provider',
);
$Self->Is(
    ref $DebuggerObject,
    'Kernel::GenericInterface::Debugger',
    'DebuggerObject instantiate correctly',
);

# define test cases
my @Tests = (
    {
        Name           => 'Test case name',
        SuccessRequest => 1,                    # 1 or 0
        RequestData    => {

            # ... add test data
        },
        ExpectedReturnLocalData => {
            Data => {

                # ... add expected local results
            },
            Success => 1,                       # 1 or 0
        },
        ExpectedReturnRemoteData => {
            Data => {

                # ... add expected remote results
            },
            Success => 1,                       # 1 or 0
        },
        Operation => 'Test',
    },

    # ... add more test cases
);

TEST:
for my $Test (@Tests) {

    # create local object
    my $LocalObject = "Kernel::GenericInterface::Operation::Test::$Test->{Operation}"->new(
        DebuggerObject => $DebuggerObject,
        WebserviceID   => $WebserviceID,
    );

    $Self->Is(
        "Kernel::GenericInterface::Operation::Test::$Test->{Operation}",
        ref $LocalObject,
        "$Test->{Name} - Create local object",
    );

    my %Auth = (
        UserLogin => $UserLogin,
        Password  => $Password,
    );
    if ( IsHashRefWithData( $Test->{Auth} ) ) {
        %Auth = %{ $Test->{Auth} };
    }

    # start requester with our web-service
```

```
    my $LocalResult = $LocalObject->Run(
        WebserviceID => $WebserviceID,
        Invoker      => $Test->{Operation},
        Data         => {
            %Auth,
            %{ $Test->{RequestData} },
        },
    );

    # check result
    $Self->Is(
        'HASH',
        ref $LocalResult,
        "$Test->{Name} - Local result structure is valid",
    );

    # create requester object
    my $RequesterObject = $Kernel::OM->Get('Kernel::GenericInterface::Requester');
    $Self->Is(
        'Kernel::GenericInterface::Requester',
        ref $RequesterObject,
        "$Test->{Name} - Create requester object",
    );

    # start requester with our web-service
    my $RequesterResult = $RequesterObject->Run(
        WebserviceID => $WebserviceID,
        Invoker      => $Test->{Operation},
        Data         => {
            %Auth,
            %{ $Test->{RequestData} },
        },
    );

    # check result
    $Self->Is(
        'HASH',
        ref $RequesterResult,
        "$Test->{Name} - Requester result structure is valid",
    );

    $Self->Is(
        $RequesterResult->{Success},
        $Test->{SuccessRequest},
        "$Test->{Name} - Requester successful result",
    );

    # ... add tests for the results
}

# delete web service
my $WebserviceDelete = $WebserviceObject->WebserviceDelete(
    ID     => $WebserviceID,
    UserID => $UserID,
);
$Self->True(
    $WebserviceDelete,
    "Deleted Web Service $WebserviceID",
);

# also delete any other added data during the this test, since RestoreDatabase must not be
 used.

1;
```

2.5.4.1.4. WSDL Extension Example

WSDL files contain the definitions of the web services and its operations for SOAP messages, in case we will extend development/webservices/GenericTickeConnectorSOAP.wsdl in some places:

Port Type:

```
   <wsdl:portType name="GenericTicketConnector_PortType">
       <!-- ... -->
       <wsdl:operation name="Test">
           <wsdl:input message="tns:TestRequest"/>
           <wsdl:output message="tns:TestResponse"/>
       </wsdl:operation>
   <!-- ... -->
```

Binding:

```
   <wsdl:binding name="GenericTicketConnector_Binding"
type="tns:GenericTicketConnector_PortType">
       <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
       <!-- ... -->
       <wsdl:operation name="Test">
           <soap:operation soapAction="http://www.otrs.org/TicketConnector/Test"/>
           <wsdl:input>
               <soap:body use="literal"/>
           </wsdl:input>
           <wsdl:output>
               <soap:body use="literal"/>
           </wsdl:output>
       </wsdl:operation>
       <!-- ... -->
   </wsdl:binding>
```

Type:

```
   <wsdl:types>
       <xsd:schema targetNamespace="http://www.otrs.org/TicketConnector/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <!-- ... -->
           <xsd:element name="Test">
               <xsd:complexType>
                   <xsd:sequence>
                           <xsd:element minOccurs="0" name="Param1" type="xsd:string"/>
                           <xsd:element minOccurs="0" name="Param2"
type="xsd:positiveInteger"/>
                   </xsd:sequence>
               </xsd:complexType>
           </xsd:element>
           <xsd:element name="TestResponse">
               <xsd:complexType>
                   <xsd:sequence>
                       <xsd:element maxOccurs="unbounded" minOccurs="1" name="Attribute1"
type="xsd:string"/>
                   </xsd:sequence>
               </xsd:complexType>
           </xsd:element>
       <!-- ... -->
       </xsd:schema>
   </wsdl:types>
```

Message:

```
   <!-- ... -->
   <wsdl:message name="TestRequest">
       <wsdl:part element="tns:Test" name="parameters"/>
   </wsdl:message>
   <wsdl:message name="TestResponse">
       <wsdl:part element="tns:TestResponse" name="parameters"/>
   </wsdl:message>
```

```
    <!-- ... -->
```

2.5.4.1.5. WADL Extension Example

WADL files contain the definitions of the web services and its operations for REST interface, add a new resource to `development/webservices/GenericTickeConnectorREST.wadl`.

```
  <resources base="http://localhost/otrs/nph-genericinterface.pl/Webservice/
GenericTicketConnectorREST">
    <!-- ... -->
    <resource path="Test" id="Test">
      <doc xml:lang="en" title="Test"/>
        <param name="Param1" type="xs:string" required="false" default="" style="query"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <param name="Param2" type="xs:string" required="false" default="" style="query"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <method name="GET" id="GET_Test">
          <doc xml:lang="en" title="GET_Test"/>
          <request/>
          <response status="200">
            <representation mediaType="application/json; charset=UTF-8"/>
          </response>
        </method>
      </resource>
    </resource>
    <!-- ... -->
  </resources>
```

2.5.4.1.6. Web Service SOAP Extension Example

Web services can be imported into OTRS by a YAML with a predefined structure in this case we will extend `development/webservices/GenericTickeConnectorSOAP.yml` for a SOAP web service.

```
Provider:
  Operation:
    # ...
    Test:
      Description: This is only a test
      MappingInbound: {}
      MappingOutbound: {}
      Type: Test::Test
```

2.5.4.1.7. Web Service REST Extension Example

Web services can be imported into OTRS by a YAML with a predefined structure in this case we will extend `development/webservices/GenericTickeConnectorREST.yml` for a REST web service.

```
Provider:
  Operation:
    # ...
    Test:
      Description: This is only a test
      MappingInbound: {}
      MappingOutbound: {}
      Type: Test::Test
  # ...
  Transport:
    Config:
      # ...
      RouteOperationMapping:
        # ..
```

```
        Test:
          RequestMethod:
          - GET
          Route: /Test
```

2.6. Démon és ütemező

2.6.1. OTRS démon

Az OTRS démon egy elkülönített folyamat, amely segít az OTRS-nek bizonyos műveleteket aszinkron módon és a webszolgáltatás folyamattól leválasztva végrehajtani, de ugyanazt az adatbázist megosztva.

2.6.1.1. OTRS démonmodulok

A `bin/otrs.Daemon.pl` OTRS démon fő célja, hogy meghívja (démonizálja) a rendszerbeállításokban lévő összes regisztrált démonmodult.

Minden egyes démonmodulnak meg kell valósítania egy közös API-t annak érdekében, hogy az OTRS démon helyesen tudja meghívni, és félig állandó folyamat legyen a rendszeren. Az állandó folyamat megnövelheti a méretét és memóriahasználatát az idő múlásával, és normális esetben nem válaszolnak a beállításokban lévő változásokra. Ezért kell a démonmoduloknak megvalósítaniuk egy eldobási mechanizmust, hogy leállíthatók és újra meghívhatók legyenek időről időre, felszabadítva a rendszer erőforrásait és újraolvasva a beállításokat.

Egy démonmodul lehet mindenre jó megoldás bizonyos feladat végrehajtásánál, de lehet olyan eset is, amikor egy megoldás különböző démonmodulokat igényel az összetettsége miatt. Pontosan ez az eset az OTRS ütemező démonjával, amely fel van osztva számos démonmodulra, beleértve a feladatkezeléshez és a feladatvégrehajtáshoz szükséges néhány démonmodult.

Nem szükséges mindig új démonmodult létrehozni bizonyos feladatok végrehajtásához. Általában az OTRS ütemező démon elboldogul ezek jelentős részével – akár ha egy olyan OTRS függvényről van szó, amelyet rendszeresen végre kell hajtani (CRON-szerűen), vagy ha egy OTRS esemény aktiválta azt – az OTRS ütemezőnek képesnek kell lenni kezelnie mindenféle beállítás nélkül, vagy egy új ütemező feladatelvégző modul hozzáadásával.

2.6.1.1.1. Új démonmodul létrehozása

Az összes démonmodulnak regisztrálva kell lennie a rendszerbeállításokban azért, hogy a fő OTRS démon meg tudja hívni azokat.

2.6.1.1.1.1. Démonmodul regisztrációs kódpélda

```
<Setting Name="DaemonModules###TestDaemon" Required="1" Valid="1">
    <Description Translatable="1">The daemon registration for the scheduler generic agent
 task manager.</Description>
    <Navigation>Daemon::ModuleRegistration</Navigation>
    <Value>
        <Hash>
            <Item Key="Module">Kernel::System::Daemon::DaemonModules::TestDaemon</Item>
        </Hash>
    </Value>
</Setting>
```

2.6.1.1.1.2. Démonmodul kódpélda

A következő kód egy olyan démonmodult valósít meg, amely megjeleníti a rendszeridőt 2 másodpercenként.

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Daemon::DaemonModules::TestDaemon;

use strict;
use warnings;
use utf8;

use Kernel::System::VariableCheck qw(:all);

use parent qw(Kernel::System::Daemon::BaseDaemon);

our @ObjectDependencies = (
    'Kernel::Config',
    'Kernel::System::Cache',
    'Kernel::System::DB',
);
```

This is common header that can be found in most OTRS modules. The class/package name is declared via the package keyword.

Ebben az esetben a BaseDaemon osztályból származtatunk le, és az objektumkezelő függőségei be vannak állítva.

```
sub new {
    my ( $Type, %Param ) = @_;

    # Új kivonat lefoglalása az objektumhoz.
    my $Self = {};
    bless $Self, $Type;

    # Objektumok lekérése a konstruktorban a teljesítmény megtartásáért.
    $Self->{ConfigObject} = $Kernel::OM->Get('Kernel::Config');
    $Self->{CacheObject}  = $Kernel::OM->Get('Kernel::System::Cache');
    $Self->{DBObject}     = $Kernel::OM->Get('Kernel::System::DB');

    # Letiltás a memóriagyorsítótárban, hogy fürtözhető legyen.
    $Self->{CacheObject}->Configure(
        CacheInMemory  => 0,
        CacheInBackend => 1,
    );

    $Self->{SleepPost} = 2;          # 2 másodperc alvás minden hurok után
    $Self->{Discard}   = 60 * 60;    # eldobás minden órában

    $Self->{DiscardCount} = $Self->{Discard} / $Self->{SleepPost};

    $Self->{Debug}      = $Param{Debug};
    $Self->{DaemonName} = 'Daemon: TestDaemon';

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. Néhány felhasznált objektum is itt lesz létrehozva. Erősen ajánlott a memóriába történő gyorsítótárazás letiltása a démonmodulokban, különösen akkor, ha az OTRS fürtözött környezetben fut.

Azért, hogy ez a démonmodul minden második másodpercben végrehajtható legyen, egy alvási idő meghatározása szükséges annak megfelelően, egyébként azonnal végrehajtásra kerül, amint lehetséges.

A démonmodul frissítése időről időre azért szükséges, hogy meghatározható legyen, mikor kell eldobni.

A következő függvényeknél (PreRun, Run és PostRun) ha azok hamis értékkel térnek vissza, akkor a fő OTRS démon el fogja dobni az objektumot, és egy újat hoz létre, amint lehetséges.

```
sub PreRun {
    my ( $Self, %Param ) = @_;

    # Annak ellenőrzése, hogy az adatbázis elérhető-e.
    return 1 if $Self->{DBObject}->Ping();

    sleep 10;

    return;
}
```

A PreRun metódus a fő démonmodul metódusa előtt kerül végrehajtásra, és a célja néhány teszt elvégzése a valódi műfelet előtt. Ebben az esetben az adatbázis ellenőrzése készen van (mindig javasolt), egyébként 10 másodpercet alszik. Ez azért szükséges, hogy megvárja az adatbázis-kapcsolat ismételt felépítését.

```
sub Run {
    my ( $Self, %Param ) = @_;

    print "Jelenlegi idő: " . localtime . "\n";

    return 1;
}
```

A Run metódus az, ahol a fő démonmodul kódja található. Ebben az esetben csak az aktuális időt írja ki.

```
sub PostRun {
    my ( $Self, %Param ) = @_;
    sleep $Self->{SleepPost};
    $Self->{DiscardCount}--;

    if ( $Self->{Debug} ) {
        print "  $Self->{DaemonName} eldobásainak száma: $Self->{DiscardCount}\n";
    }

    return if $Self->{DiscardCount} <= 0;

    return 1;
}
```

A PostRun metódus használható az alvások végrehajtásához (annak megakadályozásához, hogy a démonmodul túl gyakran legyen végrehajtva), valamint az objektum biztonságos eldobásának kezeléséhez is. Egyéb műveletek is elvégezhetők itt, mint például ellenőrzés vagy tisztítás.

```
sub Summary {
    my ( $Self, %Param ) = @_;

    my %Summary = (
        Header => 'Tesztdémon összegzés:',
        Column => [
            {
                Name        => 'SomeColumn',
```

```
                DisplayName => 'Valamilyen oszlop',
                Size        => 15,
            },
            {
                Name        => 'AnotherColumn',
                DisplayName => 'Egy másik oszlop',
                Size        => 15,
            },
            # ...
        ],
        Data => [
            {
                SomeColumn    => '1. valamilyen adat',
                AnotherColumn => '1. másik adat',
            },
            {
                SomeColumn    => '2. valamilyen adat',
                AnotherColumn => '2. másik adat',
            },
            # ...
        ],
        NoDataMesssage => '',
    );

    return \%Summary;
}
```

A `Summary` metódust a `Maint::Daemon::Summary` konzolparancs hívja meg, és `Header`, `Column`, `Data` és `NoDataMessages` kulcsokat kell visszaadnia. A `Column` és a `Data` kulcsoknak tömböknek vagy kivonatoknak kell lenniük. Arra használható, hogy hasznos információkat jelenítsen meg arról, amit a démonmodul éppen csinál, vagy ami eddig történt. Ez a metódus elhagyható.

```
1;
```

Fájl vége.

2.6.2. OTRS ütemező

Az OTRS ütemező a démonmodulok és a feladatelvégzők együttese, amelyek együtt futnak azért, hogy az összes szükséges OTRS feladatot aszinkron módon végrehajtsák a webkiszolgáló folyamatából.

2.6.2.1. OTRS ütemező feladatkezelők

A `SchedulerCronTaskManager` kiolvassa a regisztrált cron-feladatokat az OTRS rendszerbeállításaiból, és meghatározza a helyes időt a végrehajtandó feladat létrehozásához.

A `SchedulerFutureTaskManager` ellenőrzi azokat a feladatokat, amelyek úgy vannak beállítva, hogy csak egy alkalommal fussanak a jövőben, és beállítja, hogy a feladat időben kerüljön végrehajtásra. Például amikor egy általános felület meghívónak nem sikerül elérnie a távoli kiszolgálót, akkor ütemezni tudja magát, hogy 5 perccel később újra fusson.

A `SchedulerGenericAgentTaskManager` folyamatosan olvassa a `GenericAgent` feladatot, amely rendszeres időközönkénti futáshoz van beállítva, és annak megfelelően állítja be azok végrehajtását.

Amikor ezek a feladatkezelők nem elegendőek, akkor egy új démonmodul hozható létre. Egy feladat regisztrálásához a `Run()` metódusuk egy bizonyos pontján meg kell hívni

a TaskAdd() függvényt a chedulerDB objektumból, és amint regisztrálva lett, akkor a SchedulerTaskWorker végrehajtja a következő szabad időszeletben.

2.6.2.2. OTRS ütemező feladatelvégzők

A SchedulerTaskWorker az aszinkron végrehajtó használatával végrehajtja az előző feladatkezelő által tervezett összes feladatot, és még azokat is, amelyek közvetlenül a kódból jönnek.

Annak érdekében, hogy az összes feladatot végrehajtsa, a SchedulerTaskWorker meghív egy háttérprogram-modult (feladatelvégzőt) az adott feladat végrehajtásához. Az elvégző modult a feladat típusa határozza meg. Ha új feladattípus kerül hozzáadásra, akkor az új feladatelvégzőt is igényel.

2.6.2.2.1. Új ütemező feladatelvégző létrehozása

A Kernel/System/Daemon/DaemonModules/SchedulerTaskWorker mappa alatt elhelyezett összes fájl lehet potenciális feladatelvégző, és azok nem igényelnek semmilyen regisztrációt a rendszerbeállításokban.

2.6.2.2.1.1. Ütemező feladatelvégző kódpélda

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::Daemon::DaemonModules::SchedulerTaskWorker::TestWorker;

use strict;
use warnings;

use parent qw(Kernel::System::Daemon::DaemonModules::BaseTaskWorker);

our @ObjectDependencies = (
    'Kernel::System::Log',
);
```

This is common header that can be found in most OTRS modules. The class/package name is declared via the package keyword.

Ebben az esetben a BaseTaskWorker osztályból származtatunk le, és az objektumkezelő függőségei be vannak állítva.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {};
    bless( $Self, $Type );

    $Self->{Debug}      = $Param{Debug};
    $Self->{WorkerName} = 'Worker: Test';

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát.

```
sub Run {
    my ( $Self, %Param ) = @_;
```

```
    # Feladatparaméterek ellenőrzése.
    my $CheckResult = $Self->_CheckTaskParams(
        %Param,
        NeededDataAttributes => [ 'NeededAtrribute1', 'NeededAtrribute2' ],
        DataParamsRef        => 'HASH', # vagy 'ARRAT'
    );

    # Végrehajtás leállítása, ha hiba található a paraméterekben.
    return if !$CheckResult;

    my $Success;
    my $ErrorMessage;

    if ( $Self->{Debug} ) {
        print "    $Self->{WorkerName} végrehajtja a feladatot: $Param{TaskName}\n";
    }

    do {

        # A szabványos hiba lokalizálása.
        local *STDERR;

        # A szabványos hiba átirányítása egy változóba.
        open STDERR, ">>", \$ErrorMessage;

        $Success = $Kernel::OM->Get('Kernel::System::MyPackage')->Run(
            Param1 => 'someparam',
        );
    };

    if ( !$Success ) {

        $ErrorMessage ||= "$Param{TaskName} végrehajtása sikertelen egy hibaüzenettel!";

        $Self->_HandleError(
            TaskName     => $Param{TaskName},
            TaskType     => 'Test',
            LogMessage   => "Hiba történt a(z) $Param{TaskName} végrehajtásakor:
 $ErrorMessage",
            ErrorMessage => "$ErrorMessage",
        );
    }

    return $Success;
}
```

A Run a fő metódus. Egy `_CheckTaskParams()` hívás az alaposztályból megspórol néhány kódsort. A feladat végrehajtása a szabványos hibakimenet megszerzése közben nagyon jó gyakorlat, mivel az OTRS ütemező normális esetben felügyelet nélkül fut, és az összes hiba egy változóba történő mentése lehetővé fogja tenni a későbbi feldolgozást. A `_HandleError()` közös felületet biztosít a hibaüzenetek e-mailben történő küldéséhez a rendszerbeállításokban megadott címzetteknek.

```
1;
```

Fájl vége.

2.7. Dinamikus mezők

2.7.1. Áttekintés

A dinamikus mezők olyan egyéni mezők, amelyek hozzáadhatók egy képernyőhöz, hogy javítsák és információkat adjanak hozzá egy objektumhoz (például egy jegyhez vagy egy bejegyzéshez).

A dinamikus mezők a jegy és a bejegyzés szabad mezőinek (`TikcetFreeText`, `TicketFreeKey`, `TicketFreeTime`, `ArticleFreeText`, `ArticleFreeKey` és `ArticleFreeTime`) evolúciója az OTRS régebbi verzióiból.

Az OTRS 3.1-es verziójától a régi szabad mezőket lecserélték az új dinamikus mezőkre. A korábbi verziókról történő frissítéskor a jobb visszafelé kompatibilitáshoz és az adatmegőrzéshez egy költöztető parancsfájlt fejlesztettek ki a meglévő szabad mezők dinamikus mezőkre való átalakításához, és azok értékeinek áthelyezéséhez az adatbázisban lévő *ticket* és *article* táblákból az új dinamikus mezők tábláiba.

Megjegyzés

Minden szabad mezőket használó egyéni fejlesztést át kell írni az új dinamikus mezők kódszerkezetére, különben többé nem fognak működni. Emiatt nagyon fontos tudni azt, hogy az OTRS 3.0-nak csak a frissített telepítései rendelkeznek a dinamikus mezőkre átalakított régi szabad mezőkkel, az OTRS új vagy tiszta telepítéseinek nincsenek „eredetileg” meghatározott dinamikus mezői, és az egyéni fejlesztés által szükséges összes dinamikus mezőt hozzá kell adni.

A jegyenkénti vagy bejegyzésenkénti mezők számának korlátozása eltávolításra került. Ez azt jelenti, hogy egy jegy vagy egy bejegyzés annyi mezővel rendelkezhet, amennyi szükséges. És mostantól lehetséges a dinamikus mezők keretrendszerének használata egyéb objektumoknál is ahelyett, hogy csak a jegynél vagy a bejegyzésnél lenne használható.

Az új dinamikus mezők ugyanazokat az adattípusokat tudják eltárolni mint a szabad mezők (szöveg és dátum/idő), és ugyanúgy határozhatók meg mint azok (egysoros beviteli mező, legördülő és dátum/idő), de a dinamikus mezők túlmennek ezen, ugyanis egy új egész szám adattípus került hozzáadásra, valamint új lehetőségek is az olyan mezők meghatározásához, mint például többsoros beviteli mezők, jelölőnégyzetek, többválasztós mezők és (idő nélküli) dátum mezők. Minden egyes mezőtípus saját adattípust határoz meg.

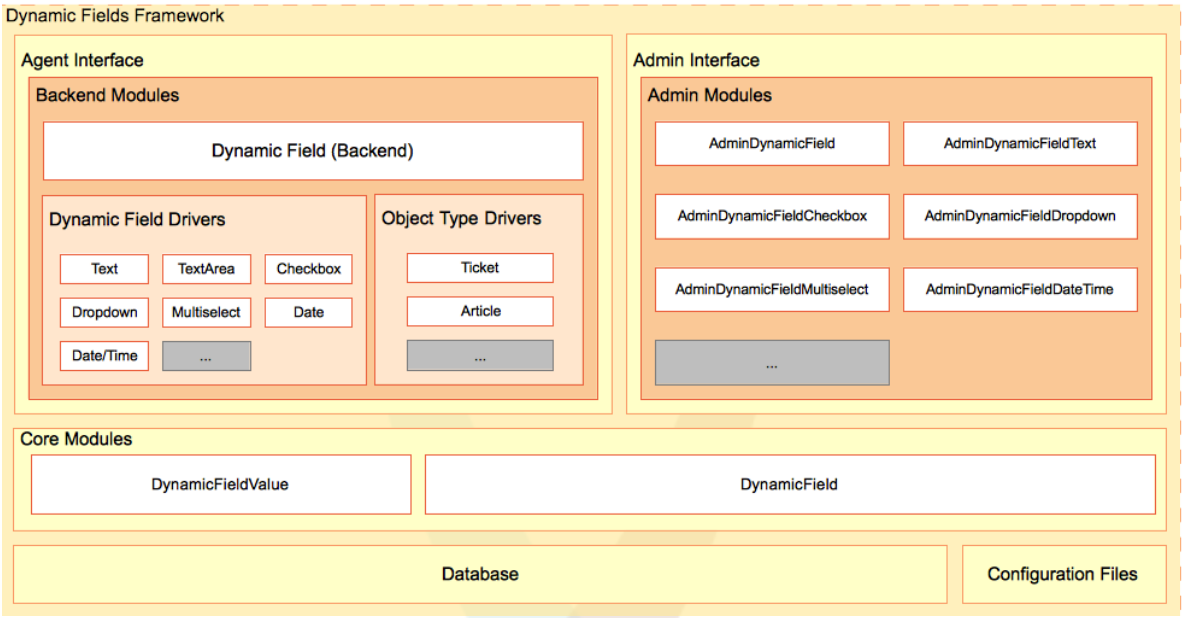
A moduláris tervezésének köszönhetően az egyes dinamikus mezőtípusok egy keretrendszerhez tartozó bővítményként láthatók, és ez a bővítmény lehet egy szabványos OTRS csomag is a dinamikus mezők elérhető típusainak kiterjesztéséhez, vagy akár a jelenlegi dinamikus mező további függvényekkel való kiterjesztéséhez.

2.7.2. Dinamikus mezők keretrendszer

Az új dinamikus mezők létrehozása előtt szükséges megérteni azok keretrendszerét, és hogy az OTRS képernyők hogyan lépnek kölcsönhatásba azokkal, valamint a mögöttes API-t.

A következő kép a dinamikus mezők keretrendszer szerkezetét mutatja be.

3.2. ábra - Dinamikus mezők szerkezete



2.7.2.1. Dinamikus mező háttérprogram modulok

2.7.2.1.1. Dinamikus mező (háttérprogram)

Az előtétprogram modulokban normális esetben a `BackendObject` nevű objektum a közvetítő az előtétprogram modulok és az egyes konkrét dinamikus mező megvalósítás vagy illesztőprogram között. Ez határoz meg egy általános közbenső API-t az összes dinamikus mező illesztőprogramhoz, és az egyes illesztőprogramok felelőssége a közbenső API megvalósítása a mező sajátos szükségleteihez.

A dinamikus mező háttérprogram az összes illesztőprogram fő vezérlője. Ebben a modulban minden egyes függvény felelős a szükséges paraméterek ellenőrzéséért, és ugyanazon függvény meghívásáért az adott illesztőprogramban a kapott dinamikus mező beállítási paraméter szerint.

Ez a modul felelős bizonyos függvények meghívásáért is minden egyes objektumtípus delegáltnál (úgymint jegy vagy bejegyzés). Például egy előzmény bejegyzés hozzáadásához vagy egy esemény elsütéséhez.

Ez a modul az `$OTRS_HOME/Kernel/System/DynamicField/Backend.pm` fájlban található.

2.7.2.1.2. Dinamikus mező illesztőprogramok

Egy dinamikus mező illesztőprogram a dinamikus mező megvalósítása. Minden egyes illesztőprogramnak meg kell valósítania a háttérprogramban meghatározott összes kötelező függvényt (van néhány olyan függvény, amely egy viselkedéstől függ, és nem szükséges megvalósítani azokat, ha a dinamikus mező nem rendelkezik azzal a bizonyos viselkedéssel).

Egy illesztőprogram felelős annak ismeretéért, hogy hogyan kérje le a saját értékét vagy értékeit egy webkérésből vagy egy profilból (mint például egy keresési profilból). Szükséges tudnia a HTML kódot is a szerkesztő vagy megjelenítő képernyőkön lévő mező megjelenítéséhez, vagy hogy hogyan lépjen kölcsönhatásba a statisztikák modullal, többek között a függvényekkel.

Ezek a modulok az `$OTRS_HOME/Kernel/System/DynamicField/Driver/*.pm` fájlokban találhatók.

Létezik néhány alap illesztőprogram, úgymint `Base.pm`, `BaseText.pm`, `BaseSelect.pm` és `BaseDateTime.pm`, amely gyakori függvényeket valósít meg bizonyos illesztőprogramokhoz (például a `TextArea.pm` illesztőprogram a `BaseText.pm` fájlt használja, amely a `Base.pm` fájlt használja, ekkor a `TextArea` csak azon függvények megvalósítását igényli, amelyek hiányoznak a `Base.pm` és `BateText.pm` fájlokból, vagy azokat, amelyek különleges esetek).

A következő az illesztőprogramok öröklődési fája:

- Base.pm
  - BaseText.pm
    - Text.pm
    - TextArea.pm
  - BaseSelect.pm
    - Dropdown.pm
    - Multiselect.pm
  - BaseDateTime.pm
    - DateTime.pm
    - Date.pm
  - Checkbox.pm

2.7.2.1.3. Objektumtípus delegált

Egy objektumtípus delegált felelős bizonyos függvények végrehajtásáért a dinamikus mezőhöz kapcsolt objektumon. Ezeket a függvényeket a háttérprogram objektum aktiválja, amint szükség van rájuk.

Ezek a modulok az `$OTRS_HOME/Kernel/System/DynamicField/ObjectType/*.pm` fájlokban találhatók.

2.7.2.2. Dinamikus mezők adminisztrátori moduljai

A dinamikus mezők kezeléséhez (hozzáadás, szerkesztés és felsorolás) már egy csomó modul van kifejlesztve. Van egy bizonyos fő modul (`AdminDynamicField.pm`), amely megjeleníti a meghatározott dinamikus mezők listáját, és más modulokon belülről hívják meg új dinamikus mezők létrehozásához vagy a meglévők módosításához.

Normális esetben egy dinamikus mező illesztőprogramnak saját adminisztrátori modulra van szüksége (adminisztrátori párbeszédablak) a tulajdonságai meghatározásához. Ez a párbeszédablak esetleg eltérhet a többi illesztőprogramtól. De ez nem kötelező, az illesztőprogramok megoszthatják az adminisztrátori párbeszédablakokat, ha szükséges információkat biztosíthatnak az összes olyan illesztőprogramhoz, amelyek hozzájuk vannak kapcsolva, nem számít, hogy eltérő típusból származnak. Ami kötelező, hogy minden egyes illesztőprogramnak hozzákapcsolva kell lennie egy adminisztrátori párbeszédablakhoz (például a szöveg és a szövegterület illesztőprogramok megosztják az `AdminDynamicFieldText.pm` adminisztrátori párbeszédablakot, és a dátum és a dátum/idő illesztőprogramok megosztják az `AdminDynamicFieldDateTime.pm` adminisztrátori párbeszédablakot).

Az adminisztrátori párbeszédablakok a normál OTRS adminisztrátori modulszabályokat és szerkezetet követik. De a szabványosításhoz az összes beállítás közös részének az összes

dinamikus mezőnél ugyanolyan megjelenésűnek kell lennie az összes adminisztrátori párbeszédablaknál.

Ezek a modulok az `$OTRS_HOME/Kernel/Modules/*.pm` fájlokban találhatók.

> **Megjegyzés**
>
> Minden adminisztrátori párbeszédablaknak szüksége van a neki megfelelő HTML sablonfájlra (`.tt`).

2.7.2.3. Dinamikus mezők alapmoduljai

Ezek a modulok olvassák és írják a dinamikus mezők információit az adatbázistáblákban.

2.7.2.3.1. `DynamicField.pm` alapmodul

Ez a modul felelős a dinamikus mező meghatározások kezeléséért. Ez biztosítja az alap API-t a hozzáadáshoz, megváltoztatáshoz, törléshez, felsoroláshoz és a dinamikus mezők lekéréséhez. Ez a modul az `$OTRS_HOME/Kernel/System/DynamicField.pm` fájlban található.

2.7.2.3.2. `DynamicFieldValue.pm` alapmodul

Ez a modul felelős a dinamikus mező értékeinek olvasásáért és írásáért az űrlapon és az adatbázisban. Ezt a modult erősen használják az illesztőprogramok, és az `$OTRS_HOME/Kernel/System/DynamicFieldValue.pm` fájlban található.

2.7.2.4. Dinamikus mezők adatbázistáblái

Két tábla van az adatbázisban a dinamikus mező információinak tárolásához:

dynamic_field: a `DynamicField.pm` alapmodul használja, és a dinamikus mező meghatározásokat tárolja.

dynamic_field_value: a `DynamicFieldValue.pm` alapmodul használja a dinamikus mező értékeinek mentéséhez minden egyes dinamikus mező és minden egyes objektumtípus példánynál.

2.7.2.5. Dinamikus mezők beállítófájljai

A háttérprogram modulnak szüksége van egy módra megtudni azt, hogy mely illesztőprogramok léteznek, mivel az illesztőprogramok mennyisége egyszerűen kiterjeszthető. Ezek kezelésének legegyszerűbb módja a rendszerbeállítás használata, ahol a dinamikus mező illesztőprogramok és az objektumtípus illesztőprogramok információi eltárolhatók és kiterjeszthetők.

A fő adminisztrátori modulnak is szükséges tudnia ezeket az információkat az elérhető dinamikus mező illesztőprogramokról a hozzájuk kapcsolt adminisztrátori párbeszédablakok használatához, a dinamikus mezők létrehozásához vagy módosításához.

Az előtétprogram moduloknak szükségük van a rendszerbeállítások olvasására megtudni azt, hogy mely dinamikus mezők vannak bekapcsolva az egyes képernyőknél, és melyek kötelezőek. Például a `Ticket::Frontend::AgentTicketPhone###DynamicField` tárolja az aktív, kötelező és inaktív dinamikus mezőket az új telefonos jegy képernyőnél.

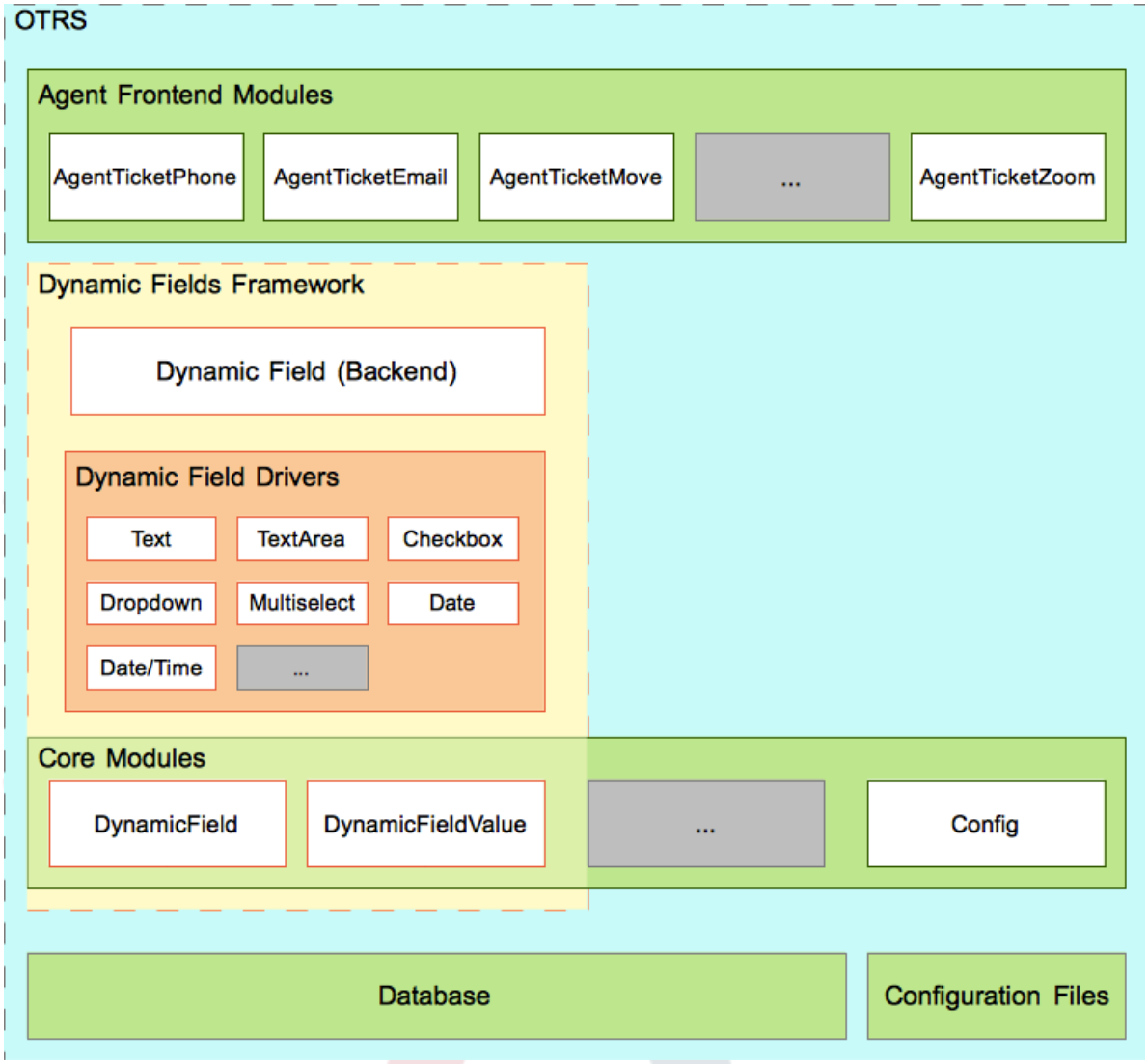
2.7.3. Dinamikus mező kölcsönhatása az előtétprogram modulokkal

Ismerve azt, hogy az előtétprogram modulok hogyan lépnek kölcsönhatásba a dinamikus mezőkkel, nem feltétlenül szükséges a dinamikus mezők kiterjesztése a jegy vagy bejegyzés objektumokhoz, mivel már elő van készítve az összes olyan képernyő, amely

dinamikus mezőket tud használni. De egyéni fejlesztések esetén vagy a dinamikus mezők más objektumokhoz történő kiterjesztéséhez nagyon hasznos tudni, hogy a dinamikus mezők keretrendszere hogyan érhető el egy előtétprogram modulból.

A következő kép egy egyszerű példáját mutatja be annak, hogy a dinamikus mezők hogyan lépnek kölcsönhatásba az OTRS keretrendszer többi részével.

3.3. ábra - Dinamikus mezők kölcsönhatása



Az első lépés, hogy az előtétprogram modul beolvassa a beállított dinamikus mezőket. Például az `AgentTicketNote` modulnak be kell olvasnia a `Ticket::Frontend::AgentTicketNote###DynamicField` beállítást. Ez a beállítás használható szűrőparaméterként a `DynamicFieldListGet()` dinamikus mező alapmodul függvénynél. A képernyő tárolhatja ennek a függvénynek az eredményeit, hogy meglegyen az aktivált dinamikus mezők listája ennél a bizonyos képernyőnél.

Ezután a képernyőnek meg kell próbálnia lekérni az értékeket a webkérésből. Erre a célra használhatja az `EditFieldValueGet()` háttérprogram-objektum függvényt, és használhatja ezeket az értékeket az ACL-ek aktiválásához. A háttérprogram-objektum minden egyes illesztőprogramot használni fog a különleges műveletek végrehajtásához az összes függvénynél.

A folytatáshoz a képernyőnek le kell kérnie a HTML-t minden egyes mezőhöz annak megjelenítéséhez. Az `EditFieldRender()` háttérprogram-objektum függvény

használható ezen művelet és az ACL-ek korlátozásának végrehajtásához, valamint a webkérésből származó értékek átadhatók ennek a függvénynek azért, hogy jobb eredményeket kapjon. Egy elküldés esetén a képernyő használhatja az `EditFieldValueValidate()` háttérprogram-objektum függvényt is a kötelező mezők ellenőrzéséhez.

Megjegyzés

A többi képernyő használhatja a `DisplayFieldRender()` függvényt az `EditFieldRender()` helyett, ha a képernyő csak a mezőértéket jeleníti meg, és ilyen esetben nincs szükség értékellenőrzésre.

A dinamikus mező értékének tárolásához szükséges az objektumazonosító lekérése. Ennél a példánál ha a dinamikus mező hozzá van kapcsolva egy jegy objektumhoz, akkor a képernyőnek már rendelkeznie kell a jegyazonosítóval, egyébként ha a mező hozzá van kapcsolva egy bejegyzés objektumhoz azért, hogy beállítsa a mező értékét, akkor először a bejegyzés létrehozása szükséges. A háttérprogram-objektumból a `ValueSet()` függvény használható a dinamikus mező értékének beállításához.

Összefoglalva, az előtétprogram moduloknak nem szükséges tudniuk, hogy az egyes dinamikus mezők hogyan működnek belsőleg azért, hogy lekérjék vagy beállítsák az értékeiket vagy megjelenítsék azokat. Egyszerűen csak meg kell hívnia a háttérprogram-objektum modult, és általános módon kell használnia a mezőket.

2.7.4. Hogyan lehet kiterjeszteni a dinamikus mezőket

Számos módszer létezik a dinamikus mezők kiterjesztésére. A következő szakaszok meg fogják próbálni a leggyakoribb forgatókönyveket bemutatni.

2.7.4.1. Egy új dinamikus mező típus létrehozása (a jegy vagy bejegyzés objektumokhoz)

Egy új dinamikus mező típus létrehozásához a következők szükségesek:

- Hozzon létre egy dinamikus mező illesztőprogramot

  Ez az új mező fő modulja.

- Hozzon létre vagy használjon egy meglévő adminisztrátori párbeszédablakot

  Egy kezelőfelület megszerzéséhez, és a konfigurációs beállításainak megadásához.

- Hozzon létre egy beállítófájlt

  Az új mező regisztrálásához a háttérprogramban (vagy a keretrendszerben lévő új adminisztrátori párbeszédablakokban, ha szükséges), valamint hogy képes legyen példányokat vagy azt létrehozni.

2.7.4.2. Egy új dinamikus mező típus létrehozása (egyéb objektumokhoz)

Egy új dinamikus mező típus létrehozásához más objektumoknál a következők szükségesek:

- Hozzon létre egy dinamikus mező illesztőprogramot

  Ez az új mező fő modulja.

- Hozzon létre egy objektumtípus delegáltat

Ez akkor is szükséges, ha a „másik objektum” nem igényel semmilyen különleges adatkezelést a függvényeiben (például egy érték beállítása után). Az összes objektumtípus delegáltnak meg kell valósítania azokat a függvényeket, amelyeket a háttérprogram igényel.

Vessen egy pillantást a jelenlegi objektumtípus delegáltakra ugyanazon függvények megvalósításához még akkor is, ha azok csak egy sikeres értéket adnak vissza a „másik objektumnál”.

- Hozzon létre vagy használjon egy meglévő adminisztrátori párbeszédablakot

  Egy kezelőfelület megszerzéséhez, és a konfigurációs beállításainak megadásához.

- Valósítsa meg a dinamikus mezőket az előtétprogram modulokban

  Hogy képes legyen használni a dinamikus mezőket.

- Hozzon létre egy beállítófájlt

  Az új mező regisztrálásához a háttérprogramban (vagy a keretrendszerben lévő új adminisztrátori párbeszédablakokban, ha szükséges), valamint hogy képes legyen példányokat vagy azt létrehozni. És végezze el a szükséges beállításokat az új képernyőkön történő megjelenítéshez, elrejtéshez vagy a dinamikus mezők kötelezőként való megjelenítéséhez.

2.7.4.3. Egy új csomag létrehozása a dinamikus mezők használatához

Egy csomag létrehozásához a meglévő dinamikus mezők használata érdekében a következők szükségesek:

- Valósítsa meg a dinamikus mezőket az előtétprogram modulokban

  Hogy képes legyen használni a dinamikus mezőket.

- Hozzon létre egy beállítófájlt

  Hogy lehetőséget adjon a végfelhasználónak az új képernyőkön történő megjelenítéshez, elrejtéshez vagy a dinamikus mezők kötelezőként való megjelenítéséhez.

2.7.4.4. A háttérprogram és az illesztőprogramok funkcionalitásainak kiterjesztése

Lehetséges lehet, hogy a háttérprogram objektum nem rendelkezik egy szükséges függvénnyel az egyéni fejlesztésekhez, vagy az is előfordulhat, hogy megvan ugyan a szükséges függvénye, de a visszatérési formátum nem felel meg az egyéni fejlesztés szükségleteinek, vagy hogy egy új viselkedés az új vagy a régi függvények végrehajtását igényli.

A legegyszerűbb mód ennek elvégzéséhez a jelenlegi mezőfájlok kiterjesztése. Ehhez egy olyan új háttérprogram kiterjesztésfájlt szükséges létrehozni, amely meghatározza az új függvényeket, és olyan illesztőprogram kiterjesztéseket is létre kell hozni, amelyek megvalósítják ezeket az új függvényeket minden egyes mezőnél. Ezeknek az új illesztőprogramoknak csak az új függvényeket kell majd megvalósítaniuk, mivel az eredeti illesztőprogramok törődnek a szabványos függvényekkel. Ezen új fájlok egyikének sincs szüksége konstruktorra, mivel ezek egy alapként lesznek betöltve a háttérprogram objektumhoz és az illesztőprogramokhoz.

Az egyetlen korlátozás, hogy a függvényeket eltérően kell elnevezni a háttérprogramnál és az illesztőprogramnál lévőknél, különben felül fognak íródni a jelenlegi objektumokkal.

Tegye az új háttérprogram kiterjesztést a DynamicField könyvtárba (például /$OTRS_HOME/Kernel/System/DynamicField/NewPackageBackend.pm és az illesztőprogramjait a /$OTRS_HOME/Kernel/System/DynamicField/Driver/NewPackage*.pm fájlokba).

Az új viselkedéseknek csak egy kis beállítás szükséges a kiterjesztések beállítófájljában.

Az új háttérprogram függvények létrehozásához a következők szükségesek:

- Hozzon létre egy új háttérprogram kiterjesztés modult

  Csak az új függvények meghatározásához.

- Hozza létre a dinamikus mezők illesztőprogram kiterjesztéseit

  Csak az új függvények megvalósításához.

- Valósítsa meg az új dinamikus mezők függvényeit az előtétprogram modulokban

  Hogy képes legyen használni az új dinamikus mezők függvényeit.

- Hozzon létre egy beállítófájlt

  Az új háttérprogram és az illesztőprogramok kiterjesztéseinek és viselkedéseinek regisztrálásához.

2.7.4.5. Egyéb kiterjesztések

Egyéb kiterjesztések lehetnek a fenti példák kombinációi.

2.7.5. Egy új dinamikus mező létrehozása

A folyamat bemutatásához egy új „jelszó” dinamikus mező lesz létrehozva. Ez az új dinamikus mező típus egy új jelszómezőt fog megjeleníteni a jegy és a bejegyzés objektumokhoz. Mivel nagyon hasonló egy szöveg dinamikus mezőhöz, ezért a Base és a BaseText illesztőprogramokat fogjuk használni alapként ezen új mező felépítéséhez.

Megjegyzés

Az új jelszómező megvalósítása csak oktatási célokra van, nem biztosít semmilyen biztonsági szintet, és nem ajánlott termelési rendszereknél.

A dinamikus mező létrehozásához négy fájlt fogunk létrehozni: egy beállítófájlt (XML) a modulok regisztrálásához, egy adminisztrátori párbeszédablak modult (Perl) a mezőlehetőségek beállításához, egy sablonmodult az adminisztrátori párbeszédablakhoz és egy dinamikus mező illesztőprogramot (Perl).

Fájlszerkezet:

```
$HOME (például /opt/otrs/)
|
...
|--/Kernel/
|   |--/Config/
|   |   |--/Files/
|   |   |   |DynamicFieldPassword.xml
...
|   |--/Modules/
|   |   |AdminDynamicFieldPassword.pm
...
|   |--/Output/
```

```
|   |   |--/HTML/
|   |   |   |--/Standard/
|   |   |   |   |AdminDynamicFieldPassword.tt
...
|   |--/System/
|   |   |--/DynamicField/
|   |   |   |--/Driver/
|   |   |   |   |Password.pm
...
```

2.7.5.1. Dinamikus mező jelszó fájlok

2.7.5.1.1. Dinamikus mező beállítófájl példa

A beállítófájlokat használják a dinamikus mező típusok (illesztőprogram) és az objektumtípus illesztőprogramok regisztrálásához a `BackendObject` számára. Ezek szabványos regisztrációkat is tárolnak az adminisztrátori modulokhoz a keretrendszerben.

2.7.5.1.1.1. Kódpélda:

Ebben a szakaszban a jelszó dinamikus mezőhöz egy beállítófájl van megjelenítve és elmagyarázva.

```
<?xml version="1.0" encoding="utf-8"?>
<otrs_config version="1.0" init="Application">
```

Ez a normál fejléc egy beállítófájlhoz.

```
    <ConfigItem Name="DynamicFields::Driver###Password" Required="0" Valid="1">
        <Description Translatable="1">Dinamikus mező háttérprogram regisztráció.</
Description>
        <Group>DynamicFieldPassword</Group>
        <SubGroup>DynamicFields::Backend::Registration</SubGroup>
        <Setting>
            <Hash>
                <Item Key="DisplayName" Translatable="1">Jelszó</Item>
                <Item Key="Module">Kernel::System::DynamicField::Driver::Password</Item>
                <Item Key="ConfigDialog">AdminDynamicFieldPassword</Item>
            </Hash>
        </Setting>
    </ConfigItem>
```

Ez a beállítás regisztrálja a jelszó dinamikus mező illesztőprogramot a háttérprogram modulhoz, így az felvehető az elérhető dinamikus mezők típusainak listájába. A saját adminisztrátori párbeszédablakát is meghatározza a `ConfigDialog` kulcsban. Ezt a kulcsot a fő dinamikus mező adminisztrátori modul használja ennek az új dinamikus mező típusának a kezeléséhez.

```
    <ConfigItem Name="Frontend::Module###AdminDynamicFieldPassword" Required="0" Valid="1">
        <Description Translatable="1">Előtétprogram-modul regisztráció az ügyintézői
felülethez.</Description>
        <Group>DynamicFieldPassword</Group>
        <SubGroup>Frontend::Admin::ModuleRegistration</SubGroup>
        <Setting>
            <FrontendModuleReg>
                <Group>admin</Group>
                <Description>Admin</Description>
                <Title Translatable="1">Dinamikus mezők szöveg háttérprogram grafikus
felület</Title>
                <Loader>
                    <JavaScript>Core.Agent.Admin.DynamicField.js</JavaScript>
                </Loader>
            </FrontendModuleReg>
```

```
        </Setting>
    </ConfigItem>
```

Ez egy szabványos modulregisztráció a jelszó adminisztrátori párbeszédablakhoz az adminisztrátori felületen.

```
</otrs_config>
```

Egy beállítófájl szabványos lezárása.

2.7.5.1.2. Dinamikus mező adminisztrátori párbeszédablak példa

Az adminisztrátori párbeszédablakok szabványos adminisztrátori modulok a dinamikus mezők kezeléséhez (hozzáadás vagy szerkesztés).

2.7.5.1.2.1. Kódpélda:

Ebben a szakaszban a jelszó dinamikus mezőhöz egy adminisztrátori párbeszédablak van megjelenítve és elmagyarázva.

```
# --
# Kernel/Modules/AdminDynamicFieldPassword.pm - provides a dynamic fields password config
 view for admins
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Modules::AdminDynamicFieldPassword;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::System::Valid;
use Kernel::System::CheckItem;
use Kernel::System::DynamicField;
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

```
sub new {
    my ( $Type, %Param ) = @_;

    my $Self = {%Param};
    bless( $Self, $Type );

    for (qw(ParamObject LayoutObject LogObject ConfigObject)) {
        if ( !$Self->{$_} ) {
            $Self->{LayoutObject}->FatalError( Message => "Nincs $_!" );
        }
    }

    # további objektumok létrehozása
    $Self->{ValidObject} = Kernel::System::Valid->new( %{$Self} );

    $Self->{DynamicFieldObject} = Kernel::System::DynamicField->new( %{$Self} );

    # beállított objektumtípusok lekérése
    $Self->{ObjectTypeConfig} = $Self->{ConfigObject}->Get('DynamicFields::ObjectType');

    # a mezők beállításának lekérése
```

```
    $Self->{FieldTypeConfig} = $Self->{ConfigObject}->Get('DynamicFields::Backend') || {};

    $Self->{DefaultValueMask} = '****';
    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint más osztályoknak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

```
sub Run {
    my ( $Self, %Param ) = @_;

    if ( $Self->{Subaction} eq 'Add' ) {
        return $Self->_Add(
            %Param,
        );
    }
    elsif ( $Self->{Subaction} eq 'AddAction' ) {

        # kihívási token ellenőrzése az írási művelethez
        $Self->{LayoutObject}->ChallengeTokenCheck();

        return $Self->_AddAction(
            %Param,
        );
    }
    if ( $Self->{Subaction} eq 'Change' ) {

        return $Self->_Change(
            %Param,
        );
    }
    elsif ( $Self->{Subaction} eq 'ChangeAction' ) {

        # kihívási token ellenőrzése az írási művelethez
        $Self->{LayoutObject}->ChallengeTokenCheck();

        return $Self->_ChangeAction(
            %Param,
        );
    }

    return $Self->{LayoutObject}->ErrorScreen(
        Message => "Meghatározatlan alművelet.",
    );
}
```

A Run a webkérés által meghívott alapértelmezett függvény. Megpróbáljuk ezt a függvényt annyira egyszerűvé tenni, amennyire csak lehetséges, és lehetővé tenni a segítő függvényeknek a „kemény” munka elvégzését.

```
sub _Add {
    my ( $Self, %Param ) = @_;

    my %GetParam;
    for my $Needed (qw(ObjectType FieldType FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed );
        if ( !$Needed ) {

            return $Self->{LayoutObject}->ErrorScreen(
                Message => "Szükséges: $Needed",
            );
        }
    }

    # az objektumtípus és a mezőtípus megjelenített nevének lekérése
```

```
    my $ObjectTypeName = $Self->{ObjectTypeConfig}->{ $GetParam{ObjectType} }->{DisplayName}
 || '';
    my $FieldTypeName  = $Self->{FieldTypeConfig}->{ $GetParam{FieldType} }->{DisplayName}
  || '';

    return $Self->_ShowScreen(
        %Param,
        %GetParam,
        Mode           => 'Add',
        ObjectTypeName => $ObjectTypeName,
        FieldTypeName  => $FieldTypeName,
    );
}
```

Az _Add függvény is nagyon egyszerű, csak lekér néhány paramétert a webkérésből, és meghívja a _ShowScreen() függvényt. Normális esetben ezt a függvényt nem szükséges módosítani.

```
sub _AddAction {
    my ( $Self, %Param ) = @_;

    my %Errors;
    my %GetParam;

    for my $Needed (qw(Name Label FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed );
        if ( !$GetParam{$Needed} ) {
            $Errors{ $Needed . 'ServerError' }        = 'ServerError';
            $Errors{ $Needed . 'ServerErrorMessage' } = 'Ez a mező kötelező.';
        }
    }

    if ( $GetParam{Name} ) {

        # annak ellenőrzése, hogy a név csak betűket és számokat tartalmaz-e
        if ( $GetParam{Name} !~ m{\A ( ?: [a-zA-Z] | \d )+ \z}xms ) {

            # kiszolgálóhiba hibaosztály hozzáadása
            $Errors{NameServerError} = 'ServerError';
            $Errors{NameServerErrorMessage} =
                'A mező nem csak ASCII betűket és számokat tartalmaz.';
        }

        # annak ellenőrzése, hogy a név kettőzött-e
        my %DynamicFieldsList = %{
            $Self->{DynamicFieldObject}->DynamicFieldList(
                Valid      => 0,
                ResultType => 'HASH',
                )
        };

        %DynamicFieldsList = reverse %DynamicFieldsList;

        if ( $DynamicFieldsList{ $GetParam{Name} } ) {

            # kiszolgálóhiba hibaosztály hozzáadása
            $Errors{NameServerError}        = 'ServerError';
            $Errors{NameServerErrorMessage} = 'Már létezik egy másik ugyanilyen nevű mező.';
        }
    }

    if ( $GetParam{FieldOrder} ) {

        # annak ellenőrzése, hogy a mezősorrend számszerű és pozitív-e
        if ( $GetParam{FieldOrder} !~ m{\A ( ?: \d )+ \z}xms ) {

            # kiszolgálóhiba hibaosztály hozzáadása
            $Errors{FieldOrderServerError}        = 'ServerError';
            $Errors{FieldOrderServerErrorMessage} = 'A mezőnek számnak kell lennie.';
        }
```

```
    }

    for my $ConfigParam (
        qw(
        ObjectType ObjectTypeName FieldType FieldTypeName DefaultValue ValidID ShowValue
        ValueMask
        )
        )
    {
        $GetParam{$ConfigParam} = $Self->{ParamObject}->GetParam( Param => $ConfigParam );
    }

    # kijavíthatatlan hibák
    if ( !$GetParam{ValidID} ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Szükséges: ValidID",
        );
    }

    # visszatérés a hozzáadás képernyőre, ha hibák vannak
    if (%Errors) {

        return $Self->_ShowScreen(
            %Param,
            %Errors,
            %GetParam,
            Mode => 'Add',
        );
    }

    # bizonyos beállítások elvégzése
    my $FieldConfig = {
        DefaultValue => $GetParam{DefaultValue},
        ShowValue    => $GetParam{ShowValue},
        ValueMask    => $GetParam{ValueMask} || $Self->{DefaultValueMask},
    };

    # egy új mező létrehozása
    my $FieldID = $Self->{DynamicFieldObject}->DynamicFieldAdd(
        Name       => $GetParam{Name},
        Label      => $GetParam{Label},
        FieldOrder => $GetParam{FieldOrder},
        FieldType  => $GetParam{FieldType},
        ObjectType => $GetParam{ObjectType},
        Config     => $FieldConfig,
        ValidID    => $GetParam{ValidID},
        UserID     => $Self->{UserID},
    );

    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Nem sikerült létrehozni az új mezőt",
        );
    }

    return $Self->{LayoutObject}->Redirect(
        OP => "Action=AdminDynamicField",
    );
}
```

Az `_AddAction` függvény kéri le a beállítási paramétereket egy új dinamikus mezőből, és ellenőrzi, hogy a dinamikus mező neve csak betűket és számokat tartalmaz-e. Ez a függvény képes ellenőrizni bármilyen egyéb paramétert is.

A `Name`, `Label`, `FieldOrder`, `Validity` közös paraméterek az összes dinamikus mezőnél, és ezek kötelezők. Minden egyes dinamikus mezőnek megvan a saját különleges beállítása, amelynek tartalmaznia kell legalább a `DefaultValue` paramétert. Ebben az esetben a `ShowValue` és a `ValueMask` paraméterekkel is rendelkezik a jelszómezőnél.

Ha a mező rendelkezik egy rögzített listájú értékek tárolásának képességével, akkor azokat a `PossibleValues` paraméterben kell tárolni a különleges beállítási kivonaton belül.

Mint más adminisztrátori modulokban, ha egy paraméter nem érvényes, akkor ez a függvény visszatér a hozzáadás képernyőre, kiemelve a hibás űrlapmezőket.

Ha az összes paraméter helyes, akkor létrehoz egy új dinamikus mezőt.

```
sub _Change {
    my ( $Self, %Param ) = @_;

    my %GetParam;
    for my $Needed (qw(ObjectType FieldType)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed );
        if ( !$Needed ) {

            return $Self->{LayoutObject}->ErrorScreen(
                Message => "Szükséges: $Needed",
            );
        }
    }

    # az objektumtípus és a mezőtípus megjelenített nevének lekérése
    my $ObjectTypeName = $Self->{ObjectTypeConfig}->{ $GetParam{ObjectType} }->{DisplayName}
 || '';
    my $FieldTypeName  = $Self->{FieldTypeConfig}->{ $GetParam{FieldType} }->{DisplayName}
  || '';

    my $FieldID = $Self->{ParamObject}->GetParam( Param => 'ID' );

    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Azonosító szükséges",
        );
    }

    # dinamikus mező adatainak lekérése
    my $DynamicFieldData = $Self->{DynamicFieldObject}->DynamicFieldGet(
        ID => $FieldID,
    );

    # érvényes dinamikus mező beállítások ellenőrzése
    if ( !IsHashRefWithData($DynamicFieldData) ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Nem sikerült lekérni az adatokat a dinamikus mezőhöz: $FieldID",
        );
    }

    my %Config = ();

    # beállítások kibontása
    if ( IsHashRefWithData( $DynamicFieldData->{Config} ) ) {
        %Config = %{ $DynamicFieldData->{Config} };
    }

    return $Self->_ShowScreen(
        %Param,
        %GetParam,
        %${DynamicFieldData},
        %Config,
        ID             => $FieldID,
        Mode           => 'Change',
        ObjectTypeName => $ObjectTypeName,
        FieldTypeName  => $FieldTypeName,
    );
}
```

A _Change függvény nagyon hasonló az _Add függvényhez, de mivel ezt a függvényt egy meglévő mező szerkesztéséhez használják, ellenőriznie kell a `FieldID` paramétert, és be kell gyűjtenie a jelenlegi dinamikus mező adatait.

```
sub _ChangeAction {
    my ( $Self, %Param ) = @_;

    my %Errors;
    my %GetParam;

    for my $Needed (qw(Name Label FieldOrder)) {
        $GetParam{$Needed} = $Self->{ParamObject}->GetParam( Param => $Needed );
        if ( !$GetParam{$Needed} ) {
            $Errors{ $Needed . 'ServerError' }        = 'ServerError';
            $Errors{ $Needed . 'ServerErrorMessage' } = 'Ez a mező kötelező.';
        }
    }

    my $FieldID = $Self->{ParamObject}->GetParam( Param => 'ID' );
    if ( !$FieldID ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Azonosító szükséges",
        );
    }

    if ( $GetParam{Name} ) {

        # annak ellenőrzése, hogy a név kisbetűs-e
        if ( $GetParam{Name} !~ m{\A ( ?: [a-zA-Z] | \d )+ \z}xms ) {

            # kiszolgálóhiba hibaosztály hozzáadása
            $Errors{NameServerError} = 'ServerError';
            $Errors{NameServerErrorMessage} =
                'A mező nem csak ASCII betűket és számokat tartalmaz.';
        }

        # annak ellenőrzése, hogy a név kettőzött-e
        my %DynamicFieldsList = %{
            $Self->{DynamicFieldObject}->DynamicFieldList(
                Valid      => 0,
                ResultType => 'HASH',
                )
        };

        %DynamicFieldsList = reverse %DynamicFieldsList;

        if (
            $DynamicFieldsList{ $GetParam{Name} } &&
            $DynamicFieldsList{ $GetParam{Name} } ne $FieldID
            )
        {

            # kiszolgálóhiba hibaosztály hozzáadása
            $Errors{NameServerError}        = 'ServerError';
            $Errors{NameServerErrorMessage} = 'Már létezik egy másik ugyanilyen nevű mező.';
        }
    }

    if ( $GetParam{FieldOrder} ) {

        # annak ellenőrzése, hogy a mezősorrend számszerű és pozitív-e
        if ( $GetParam{FieldOrder} !~ m{\A ( ?: \d )+ \z}xms ) {

            # add server error error class
            $Errors{FieldOrderServerError}        = 'ServerError';
            $Errors{FieldOrderServerErrorMessage} = 'A mezőnek számnak kell lennie.';
        }
    }

    for my $ConfigParam (
```

```
        qw(
        ObjectType ObjectTypeName FieldType FieldTypeName DefaultValue ValidID ShowValue
        ValueMask
        )
        )
    {
        $GetParam{$ConfigParam} = $Self->{ParamObject}->GetParam( Param => $ConfigParam );
    }

    # kijavíthatatlan hibák
    if ( !$GetParam{ValidID} ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Szükséges: ValidID",
        );
    }

    # a dinamikus mező adatainak lekérése
    my $DynamicFieldData = $Self->{DynamicFieldObject}->DynamicFieldGet(
        ID => $FieldID,
    );

    # érvényes dinamikus mező beállítások ellenőrzése
    if ( !IsHashRefWithData($DynamicFieldData) ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Nem sikerült lekérni az adatokat a dinamikus mezőhöz: $FieldID",
        );
    }

    # visszatérés a változtatás képernyőhöz, ha hibák vannak
    if (%Errors) {

        return $Self->_ShowScreen(
            %Param,
            %Errors,
            %GetParam,
            ID   => $FieldID,
            Mode => 'Change',
        );
    }

    # különleges beállítások elvégzése
    my $FieldConfig = {
        DefaultValue => $GetParam{DefaultValue},
        ShowValue    => $GetParam{ShowValue},
        ValueMask    => $GetParam{ValueMask},
    };

    # dinamikus mező frissítése (a FieldType és az ObjectType nem változtatható meg; régi
értékek használata)
    my $UpdateSuccess = $Self->{DynamicFieldObject}->DynamicFieldUpdate(
        ID         => $FieldID,
        Name       => $GetParam{Name},
        Label      => $GetParam{Label},
        FieldOrder => $GetParam{FieldOrder},
        FieldType  => $DynamicFieldData->{FieldType},
        ObjectType => $DynamicFieldData->{ObjectType},
        Config     => $FieldConfig,
        ValidID    => $GetParam{ValidID},
        UserID     => $Self->{UserID},
    );

    if ( !$UpdateSuccess ) {

        return $Self->{LayoutObject}->ErrorScreen(
            Message => "Nem sikerült frissíteni a mezőt: $GetParam{Name}",
        );
    }

    return $Self->{LayoutObject}->Redirect(
        OP => "Action=AdminDynamicField",
```

```
    );
}
```

A _ChangeAction() nagyon hasonló az _AddAction() függvényhez, de át van írva egy meglévő mező frissítéséhez egy új létrehozása helyett.

```
sub _ShowScreen {
    my ( $Self, %Param ) = @_;

    $Param{DisplayFieldName} = 'New';

    if ( $Param{Mode} eq 'Change' ) {
        $Param{ShowWarning}      = 'ShowWarning';
        $Param{DisplayFieldName} = $Param{Name};
    }

    # fejléc
    my $Output = $Self->{LayoutObject}->Header();
    $Output .= $Self->{LayoutObject}->NavigationBar();

    # az összes mező lekérése
    my $DynamicFieldList = $Self->{DynamicFieldObject}->DynamicFieldListGet(
        Valid => 0,
    );

    # a sorrendszámok listájának lekérése (már rendezve van).
    my @DynamicfieldOrderList;
    for my $Dynamicfield ( @{$DynamicFieldList} ) {
        push @DynamicfieldOrderList, $Dynamicfield->{FieldOrder};
    }

    # hozzáadáskor létre kell hoznunk egy további sorrendszámot az új mezőhöz
    if ( $Param{Mode} eq 'Add' ) {

        # az utolsó elem lekérése a sorrendlistából, és 1 hozzáadása
        my $LastOrderNumber = $DynamicfieldOrderList[-1];
        $LastOrderNumber++;

        # ezen új sorrendszám hozzáadása a lista végéhez
        push @DynamicfieldOrderList, $LastOrderNumber;
    }

    my $DynamicFieldOrderSrtg = $Self->{LayoutObject}->BuildSelection(
        Data          => \@DynamicfieldOrderList,
        Name          => 'FieldOrder',
        SelectedValue => $Param{FieldOrder} || 1,
        PossibleNone  => 0,
        Class         => 'W50pc Validate_Number',
    );

    my %ValidList = $Self->{ValidObject}->ValidList();

    # az ellenőrzés kiválasztás létrehozása
    my $ValidityStrg = $Self->{LayoutObject}->BuildSelection(
        Data         => \%ValidList,
        Name         => 'ValidID',
        SelectedID   => $Param{ValidID} || 1,
        PossibleNone => 0,
        Translation  => 1,
        Class        => 'W50pc',
    );

    # a beállítási mezőre jellemző beállítások meghatározása
    my $DefaultValue = ( defined $Param{DefaultValue} ? $Param{DefaultValue} : '' );

    # az érték megjelenítése választás létrehozása
    my $ShowValueStrg = $Self->{LayoutObject}->BuildSelection(
        Data => [ 'No', 'Yes' ],
        Name => 'ShowValue',
        SelectedValue => $Param{ShowValue} || 'No',
```

```
        PossibleNone  => 0,
        Translation   => 1,
        Class         => 'W50pc',
    );

    # kimenet előállítása
    $Output .= $Self->{LayoutObject}->Output(
        TemplateFile => 'AdminDynamicFieldPassword',
        Data         => {
            %Param,
            ValidityStrg          => $ValidityStrg,
            DynamicFieldOrderSrtg => $DynamicFieldOrderSrtg,
            DefaultValue          => $DefaultValue,
            ShowValueStrg         => $ShowValueStrg,
            ValueMask             => $Param{ValueMask} || $Self->{DefaultValueMask},
        },
    );

    $Output .= $Self->{LayoutObject}->Footer();

    return $Output;
}

1;
```

A _ShowScreen függvényt használják egy sablonból történő HTML elemek és blokkok beállítására és meghatározására az adminisztrátori párbeszédablak HTML kódjának előállításához.

2.7.5.1.3. Dinamikus mező sablon az adminisztrátori párbeszédablak példához

A sablon az a hely, ahol a párbeszédablak HTML kódja el van tárolva.

2.7.5.1.3.1. Kódpélda:

Ebben a szakaszban a jelszó dinamikus mezőhöz egy adminisztrátori párbeszédablak sablon van megjelenítve és elmagyarázva.

```
# --
# AdminDynamicFieldPassword.tt - provides HTML form for AdminDynamicFieldPassword
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban.

```
<div class="MainBox ARIARoleMain LayoutFixedSidebar SidebarFirst">
    <h1>[% Translate("Dynamic Fields") | html %] - [% Translate(Data.ObjectTypeName) |
 html %]: [% Translate(Data.Mode) | html %] [% Translate(Data.FieldTypeName) | html %] [%
 Translate("Mező") | html %]</h1>

    <div class="Clear"></div>

    <div class="SidebarColumn">
        <div class="WidgetSimple">
            <div class="Header">
                <h2>[% Translate("Műveletek") | html %]</h2>
            </div>
            <div class="Content">
                <ul class="ActionList">
                    <li>
                        <a href="[% Env("Baselink") %]Action=AdminDynamicField"
 class="CallForAction"><span>[% Translate("Ugrás vissza az áttekintőhöz") | html %]</span></
a>
```

```
                    </li>
                </ul>
            </div>
        </div>
    </div>
```

A kód ezen része rendelkezik a fő dobozzal és a műveletek oldalsávval is. Nincs szükség módosításokra ebben a szakaszban.

```
    <div class="ContentColumn">
        <form action="[% Env("CGIHandle") %]" method="post" class="Validate
 PreventMultipleSubmits">
            <input type="hidden" name="Action" value="AdminDynamicFieldPassword" />
            <input type="hidden" name="Subaction" value="[% Data.Mode | html %]Action" />
            <input type="hidden" name="ObjectType" value="[% Data.ObjectType | html %]" />
            <input type="hidden" name="FieldType" value="[% Data.FieldType | html %]" />
            <input type="hidden" name="ID" value="[% Data.ID | html %]" />
```

A kód ezen szakaszában van meghatározva a párbeszédablak jobboldali része. Figyelje meg, hogy a rejtett `Action` beviteli mező értékének egyeznie kell az adminisztrátori párbeszédablak nevével.

```
            <div class="WidgetSimple">
                <div class="Header">
                    <h2>[% Translate("Általános") | html %]</h2>
                </div>
                <div class="Content">
                    <div class="LayoutGrid ColumnsWithSpacing">
                        <div class="Size1of2">
                            <fieldset class="TableLike">
                                <label class="Mandatory" for="Name"><span class="Marker">*</
span> [% Translate("Név") | html %]:</label>
                                <div class="Field">
                                    <input id="Name" class="W50pc [% Data.NameServerError |
 html %] [% Data.ShowWarning | html %]  Validate_Alphanumeric" type="text" maxlength="200"
 value="[% Data.Name | html %]" name="Name"/>
                                    <div id="NameError" class="TooltipErrorMessage"><p>[%
 Translate("Ez a mező kötelező, és az értéke csak betű és szám karakter lehet.") | html %]</
p></div>
                                    <div id="NameServerError"
 class="TooltipErrorMessage"><p>[% Translate(Data.NameServerErrorMessage) | html %]</p></
div>
                                    <p class="FieldExplanation">[% Translate("Egyedinek kell
 lennie, és csak betű és szám karaktereket fogad el.") | html %]</p>
                                    <p class="Warning Hidden">[% Translate("Az érték
 megváltoztatása kézi módosításokat fog igényelni a rendszeren.") | html %]</p>
                                </div>
                                <div class="Clear"></div>

                                <label class="Mandatory" for="Label"><span
 class="Marker">*</span> [% Translate("Címke") | html %]:</label>
                                <div class="Field">
                                    <input id="Label" class="W50pc [% Data.LabelServerError
 | html %] Validate_Required" type="text" maxlength="200" value="[% Data.Label | html %]"
 name="Label"/>
                                    <div id="LabelError" class="TooltipErrorMessage"><p>[%
 Translate("Ez a mező kötelező.") | html %]</p></div>
                                    <div id="LabelServerError"
 class="TooltipErrorMessage"><p>[% Translate(Data.LabelServerErrorMessage) | html %]</p></
div>
                                    <p class="FieldExplanation">[% Translate("Ez azokon a
 képernyőkön megjelenítendő név, ahol a mező aktív.") | html %]</p>
                                </div>
                                <div class="Clear"></div>

                                <label class="Mandatory" for="FieldOrder"><span
 class="Marker">*</span> [% Translate("Mezősorrend") | html %]:</label>
```

```
                                <div class="Field">
                                    [% Data.DynamicFieldOrderSrtg %]
                                    <div id="FieldOrderError"
class="TooltipErrorMessage"><p>[% Translate("Ez a mező kötelező, és csak számot
tartalmazhat.") | html %]</p></div>
                                    <div id="FieldOrderServerError"
class="TooltipErrorMessage"><p>[% Translate(Data.FieldOrderServerErrorMessage) | html %]</
p></div>
                                    <p class="FieldExplanation">[% Translate("Ez az a
sorrend, amelyben ez a mező meg fog jelenni a képernyőkön, ahol aktív.") | html %]</p>
                                </div>
                                <div class="Clear"></div>
                            </fieldset>
                        </div>
                        <div class="Size1of2">
                            <fieldset class="TableLike">
                                <label for="ValidID">[% Translate("Érvényesség") | html
%]:</label>
                                <div class="Field">
                                    [% Data.ValidityStrg %]
                                </div>
                                <div class="Clear"></div>

                                <div class="SpacingTop"></div>
                                <label for="FieldTypeName">[% Translate("Mezőtípus") | html
%]:</label>
                                <div class="Field">
                                    <input id="FieldTypeName" readonly="readonly"
class="W50pc" type="text" maxlength="200" value="[% Data.FieldTypeName | html %]"
name="FieldTypeName"/>
                                    <div class="Clear"></div>
                                </div>

                                <div class="SpacingTop"></div>
                                <label for="ObjectTypeName">[% Translate("Objektumtípus") |
html %]:</label>
                                <div class="Field">
                                    <input id="ObjectTypeName" readonly="readonly"
class="W50pc" type="text" maxlength="200" value="[% Data.ObjectTypeName | html %]"
name="ObjectTypeName"/>
                                    <div class="Clear"></div>
                                </div>
                            </fieldset>
                        </div>
                    </div>
                </div>
            </div>
```

Ez az első felületi elem tartalmazza a szokásos űrlapattribútumokat a dinamikus mezőkhöz. A többi dinamikus mezővel való következetességért javasolt a kód ezen részének változatlanul hagyása.

```
            <div class="WidgetSimple">
                <div class="Header">
                    <h2>[% Translate(Data.FieldTypeName) | html %] [%
Translate("Mezőbeállítások") | html %]</h2>
                </div>
                <div class="Content">
                    <fieldset class="TableLike">

                        <label for="DefaultValue">[% Translate("Alapértelmezett érték") |
html %]:</label>
                        <div class="Field">
                            <input id="DefaultValue" class="W50pc" type="text"
maxlength="200" value="[% Data.DefaultValue | html %]" name="DefaultValue"/>
                            <p class="FieldExplanation">[% Translate("Ez az alapértelmezett
érték ehhez a mezőhöz.") | html %]</p>
                        </div>
                        <div class="Clear"></div>
```

```
                        <label for="ShowValue">[% Translate("Érték megjelenítése") | html
%]:</label>
                        <div class="Field">
                            [% Data.ShowValueStrg %]
                            <p class="FieldExplanation">
                                [% Translate("A mezőérték felfedéséhez a nem szerkesztői
képernyőkön (például jegynagyítás képernyő)") | html %]
                            </p>
                        </div>
                        <div class="Clear"></div>

                        <label for="ValueMask">[% Translate("Rejtett értékmaszk") | html
%]:</label>
                        <div class="Field">
                            <input id="ValueMask" class="W50pc" type="text" maxlength="200"
value="[% Data.ValueMask | html %]" name="ValueMask"/>
                            <p class="FieldExplanation">
                                [% Translate("Ez az alternatív érték annak megjelenítéséhez,
ha az Érték megjelenítése „Nem” értékre van állítva (Alapértelmezett: **** ).") | html %]
                            </p>
                        </div>
                        <div class="Clear"></div>

                    </fieldset>
                </div>
            </div>
```

A második felületi elem a dinamikus mezőre jellemző űrlapattribútumokkal rendelkezik. Ez az a hely, ahol az új attribútumok beállíthatók, és használhatnak JavaScript és AJAX technológiákat, hogy egyszerűbbé és barátságosabbá tegyék a végfelhasználó számára.

```
            <fieldset class="TableLike">
                <div class="Field SpacingTop">
                    <button type="submit" class="Primary" value="[% Translate("Save") | html
%]">[% Translate("Mentés") | html %]</button>
                    [% Translate("vagy") | html %]
                    <a href="[% Env("Baselink") %]Action=AdminDynamicField">[%
Translate("Mégse") | html %]</a>
                </div>
                <div class="Clear"></div>
            </fieldset>
        </form>
    </div>
</div>
[% WRAPPER JSOnDocumentComplete %]
<script type="text/javascript">//<![CDATA[
$('.ShowWarning').bind('change keyup', function (Event) {
    $('p.Warning').removeClass('Hidden');
});

Core.Agent.Admin.DynamicField.ValidationInit();
//]]></script>
[% END %]
```

A fájl utolsó része tartalmazza a „Mentés” gombot és a „Mégse” hivatkozást, valamint az egyéb szükséges JavaScript kódot.

2.7.5.1.4. Dinamikus mező illesztőprogram példa

Az illesztőprogram *maga a* dinamikus mező. Számos olyan függvényt tartalmaz, amelyet széles körben használnak az OTRS keretrendszerben. Egy illesztőprogram örökölhet néhány függvényt az alaposztályokból, például a `TextArea` illesztőprogram a `Base.pm` és a `BaseText.pm` fájlokból örökli a függvények legnagyobb részét, és csak azokat a függvényeket valósítja meg, amelyek eltérő logikát vagy eredményeket igényelnek. A jelölőnégyzet mező illesztőprogram csak a `Base.pm` fájlból örököl, mivel az összes többi függvény nagyon eltérő bármely más alap illesztőprogramtól.

Megjegyzés

Nézze meg a /Kernel/System/DynmicField/Backend.pm modul Perl On-line Dokumentációját (POD) az összes attribútum listája és a lehetséges visszatérési adatok megismeréséhez az egyes függvényeknél.

2.7.5.1.4.1. Kódpélda:

Ebben a szakaszban a jelszó dinamikus mező illesztőprogram van bemutatva és elmagyarázva. Ez az illesztőprogram örököl néhány függvényt a Base.pm és a BaseText.pm fájlokból, és csak azokat a függvényeket valósítja meg, amelyek eltérő eredményeket igényelnek.

```
# --
# Kernel/System/DynamicField/Driver/Password.pm - Driver for DynamicField Password backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::Driver::Password;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::System::DynamicFieldValue;

use base qw(Kernel::System::DynamicField::Driver::BaseText);

our @ObjectDependencies = (
    'Kernel::Config',
    'Kernel::System::DynamicFieldValue',
    'Kernel::System::Main',
);
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva. Figyelje meg, hogy a BaseText osztályt használják alaposztályként.

```
sub new {
    my ( $Type, %Param ) = @_;

    # új kivonat lefoglalása az objektumhoz
    my $Self = {};
    bless( $Self, $Type );

    # mezőviselkedések beállítása
    $Self->{Behaviors} = {
        'IsACLReducible'               => 0,
        'IsNotificationEventCondition' => 1,
        'IsSortable'                   => 0,
        'IsFiltrable'                  => 0,
        'IsStatsCondition'             => 1,
        'IsCustomerInterfaceCapable'   => 1,
    };

    # a dinamikus mező háttérprogram egyéni kiterjesztéseinek lekérése
    my $DynamicFieldDriverExtensions
        = $Kernel::OM->Get('Kernel::Config')-
>Get('DynamicFields::Extension::Driver::Password');

    EXTENSION:
    for my $ExtensionKey ( sort keys %{$DynamicFieldDriverExtensions} ) {
```

```
        # érvénytelen kiterjesztések kihagyása
        next EXTENSION if !IsHashRefWithData( $DynamicFieldDriverExtensions-
>{$ExtensionKey} );

        # egy kiterjesztés beállítás gyors elérésének létrehozása
        my $Extension = $DynamicFieldDriverExtensions->{$ExtensionKey};

        # annak ellenőrzése, hogy a kiterjesztésnek van-e új modulja
        if ( $Extension->{Module} ) {

            # annak ellenőrzése, hogy a modul betölthető-e
            if (
                !$Kernel::OM->Get('Kernel::System::Main')->RequireBaseClass( $Extension-
>{Module} )
                )
            {
                die "Nem lehet betölteni a dinamikus mezők háttérprogram modulját:"
                    . " $Extension->{Module}! $@";
            }
        }

        # annak ellenőrzése, hogy a kiterjesztés tartalmaz-e további viselkedéseket
        if ( IsHashRefWithData( $Extension->{Behaviors} ) ) {

            %{ $Self->{Behaviors} } = (
                %{ $Self->{Behaviors} },
                %{ $Extension->{Behaviors} }
            );
        }
    }

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát. A kódolási irányelvek szerint más osztályoknak azon objektumait kell a new konstruktorban létrehozni, amelyek ebben a modulban szükségesek.

Fontos a viselkedéseket helyesen meghatározni, mivel a mező lehet használható vagy lehet nem használható bizonyos képernyőkön. Azokat a függvényeket esetleg nem szükséges megvalósítani, amelyek olyan viselkedésektől függenek, amelyek nem aktívak ennél a bizonyos mezőnél.

Megjegyzés

Az illesztőprogramokat kizárólag a `BackendObject` hozza létre, és nem közvetlenül bármely egyéb modulból.

```
sub EditFieldRender {
    my ( $Self, %Param ) = @_;

    # beállítások átvétele a mezőbeállításokból
    my $FieldConfig = $Param{DynamicFieldConfig}->{Config};
    my $FieldName   = 'DynamicField_' . $Param{DynamicFieldConfig}->{Name};
    my $FieldLabel  = $Param{DynamicFieldConfig}->{Label};

    my $Value = '';

    # a mezőérték beállítása vagy alapértelmezett
    if ( $Param{UseDefaultValue} ) {
        $Value = ( defined $FieldConfig->{DefaultValue} ? $FieldConfig->{DefaultValue} :
 '' );
    }
    $Value = $Param{Value} if defined $Param{Value};

    # a dinamikus mező értékének kibontása a webkérésből
    my $FieldValue = $Self->EditFieldValueGet(
        %Param,
```

```
    );

    # értékek beállítása a paraméterobjektumból, ha létezik
    if ( defined $FieldValue ) {
        $Value = $FieldValue;
    }

    # osztály ellenőrzése és beállítása, ha szükséges
    my $FieldClass = 'DynamicFieldText W50pc';
    if ( defined $Param{Class} && $Param{Class} ne '' ) {
        $FieldClass .= ' ' . $Param{Class};
    }

    # mező beállítása kötelezőként
    $FieldClass .= ' Validate_Required' if $Param{Mandatory};

    # hiba CSS osztály beállítása
    $FieldClass .= ' ServerError' if $Param{ServerError};

    my $HTMLString = <<"EOF";
<input type="password" class="$FieldClass" id="$FieldName" name="$FieldName"
 title="$FieldLabel" value="$Value" />
EOF

    if ( $Param{Mandatory} ) {
        my $DivID = $FieldName . 'Error';

        # kliensoldali ellenőrzéshez
        $HTMLString .= <<"EOF";
    <div id="$DivID" class="TooltipErrorMessage">
        <p>
            \$Text{"Ez a mező kötelező."}
        </p>
    </div>
EOF
    }

    if ( $Param{ServerError} ) {

        my $ErrorMessage = $Param{ErrorMessage} || 'Ez a mező kötelező.';
        my $DivID = $FieldName . 'ServerError';

        # kiszolgálóoldali ellenőrzéshez
        $HTMLString .= <<"EOF";
    <div id="$DivID" class="TooltipErrorMessage">
        <p>
            \$Text{"$ErrorMessage"}
        </p>
    </div>
EOF
    }

    # az EditLabelRender meghívása a közös illesztőprogramon
    my $LabelString = $Self->EditLabelRender(
        %Param,
        DynamicFieldConfig => $Param{DynamicFieldConfig},
        Mandatory          => $Param{Mandatory} || '0',
        FieldName          => $FieldName,
    );

    my $Data = {
        Field => $HTMLString,
        Label => $LabelString,
    };

    return $Data;
}
```

Ez a függvény felelős a mező és annak címkéje HTML ábrázolásának létrehozásáért, és olyan szerkesztő képernyőkön használják, mint például az `AgentTicketPhone`, `AgentTicketNote`, stb.

```
sub DisplayValueRender {
    my ( $Self, %Param ) = @_;

    # a HTMLOutput beállítása alapértelmezettként, ha nincs megadva
    if ( !defined $Param{HTMLOutput} ) {
        $Param{HTMLOutput} = 1;
    }

    my $Value;
    my $Title;

    # annak ellenőrzése, hogy a mező be van-e állítva a jelszó megjelenítéséhez vagy sem
    if (
        defined $Param{DynamicFieldConfig}->{Config}->{ShowValue}
        && $Param{DynamicFieldConfig}->{Config}->{ShowValue} eq 'Yes'
        )
    {

        # a nyers Title és Value szövegek lekérése a mezőértékből
        $Value = defined $Param{Value} ? $Param{Value} : '';
        $Title = $Value;
    }
    else {

        # a maszk megjelenítése az érték helyett
        $Value = $Param{DynamicFieldConfig}->{Config}->{ValueMask} || '';
        $Title = 'A mező értéke rejtve van.'
    }

    # HTMLOutput átalakítások
    if ( $Param{HTMLOutput} ) {
        $Value = $Param{LayoutObject}->Ascii2Html(
            Text => $Value,
            Max => $Param{ValueMaxChars} || '',
        );

        $Title = $Param{LayoutObject}->Ascii2Html(
            Text => $Title,
            Max => $Param{TitleMaxChars} || '',
        );
    }
    else {
        if ( $Param{ValueMaxChars} && length($Value) > $Param{ValueMaxChars} ) {
            $Value = substr( $Value, 0, $Param{ValueMaxChars} ) . '...';
        }
        if ( $Param{TitleMaxChars} && length($Title) > $Param{TitleMaxChars} ) {
            $Title = substr( $Title, 0, $Param{TitleMaxChars} ) . '...';
        }
    }

    # visszatérési szerkezet létrehozása
    my $Data = {
        Value => $Value,
        Title => $Title,
    };

    return $Data;
}
```

A `DisplayValueRender()` függvény egyszerű szövegként adja vissza a mező értékét és annak feliratát (mindkettő lefordítható). Ennél a bizonyos példánál azt ellenőrizzük, hogy a jelszót meg kell-e mutatni, vagy egy beállítási paraméter által előre meghatározott maszkot kell-e megjeleníteni a dinamikus mezőben.

```
sub ReadableValueRender {
    my ( $Self, %Param ) = @_;

    my $Value;
```

```
    my $Title;

    # annak ellenőrzése, hogy a mező be van-e állítva a jelszó megjelenítéséhez vagy sem
    if (
        defined $Param{DynamicFieldConfig}->{Config}->{ShowValue}
        && $Param{DynamicFieldConfig}->{Config}->{ShowValue} eq 'Yes'
        )
    {

        # a nyers Title és Value szövegek lekérése a mezőértékből
        $Value = $Param{Value} // '';
        $Title = $Value;
    }
    else {

        # a maszk megjelenítése az érték helyett
        $Value = $Param{DynamicFieldConfig}->{Config}->{ValueMask} || '';
        $Title = 'A mező értéke rejtve van.'
    }

    # szövegek levágása, ha szükséges
    if ( $Param{ValueMaxChars} && length($Value) > $Param{ValueMaxChars} ) {
        $Value = substr( $Value, 0, $Param{ValueMaxChars} ) . '...';
    }
    if ( $Param{TitleMaxChars} && length($Title) > $Param{TitleMaxChars} ) {
        $Title = substr( $Title, 0, $Param{TitleMaxChars} ) . '...';
    }

    # visszatérési szerkezet létrehozása
    my $Data = {
        Value => $Value,
        Title => $Title,
    };

    return $Data;
}
```

Ez a függvény hasonló a `DisplayValueRender()` függvényhez, de olyan helyeken használják, ahol nincs `LayoutObject`.

2.7.5.1.4.2. Egyéb függvények:

A következők olyan egyéb függvények, amelyek talán szükségesek lehetnek, ha egy új dinamikus mező nem örököl más osztályokból. Ezen függvények teljes kódjának megtekintéséhez nézzen bele közvetlenül a `Kernel/System/DynamicField/Driver/Base.pm` és a `Kernel/System/DynamicField/Driver/BaseText.pm` fájlokba.

```
sub ValueGet { ... }
```

Ez a függvény lekéri az értéket a mezőből egy adott objektumnál. Ebben az esetben az első szövegértéket adjuk vissza, mivel a mező egyszerre csak egy szövegértéket tárol.

```
sub ValueSet { ... }
```

A `ValueSet()` függvényt egy dinamikus mező érték tárolásához használják. Ebben az esetben a mező csak egy szöveg típusú értéket tárol. Más mezők tárolhatnak egynél több értéket is a `ValueText`, a `ValueDateTime` vagy a `ValueInt` formátumnál.

```
sub ValueDelete { ... }
```

Ezt a függvényt egy mező értékének törléséhez használják, amely egy bizonyos objektumazonosítóhoz van csatolva. Például ha egy objektum példánynak törlésére

készülnek, akkor nincs oka a mezőérték tárolásának az adatbázisban annál a bizonyos objektumpéldánynál.

```
sub AllValuesDelete { ... }
```

Az `AllValuesDelete()` függvényt az összes érték törléséhez használják egy bizonyos dinamikus mezőből. Ez a függvény nagyon hasznos, amikor egy dinamikus mező törlésre fog kerülni.

```
sub ValueValidate { ... }
```

A `ValueValidate()` függvényt annak ellenőrzéséhez használják, hogy az érték megegyezik-e a típusának.

```
sub SearchSQLGet { ... }
```

Ezt a függvényt a `TicketSearch` alapmodul használja egy jegy kereséséhez szükséges belső lekérdezés felépítéséhez, ezt a mezőt alapul véve keresési paraméterként.

```
sub SearchSQLOrderFieldGet { ... }
```

A `SearchSQLOrderFieldGet` szintén egy segítő a `TicketSearch` modulhoz. A `$Param{TableAlias}` paramétert meg kell tartani, és a `value_text` lecserélhető a `value_date` vagy a `value_int` paraméterrel a mezőtől függően.

```
sub EditFieldValueGet { ... }
```

Az `EditFieldValueGet()` egy olyan függvény, amelyet az OTRS szerkesztő képernyőin használnak, és a célja a mező értékének lekérése vagy egy sablonból (mint például az általános ügyintéző profilból), vagy egy webkérésből. Ez a függvény megkapja a webkérést a `$Param{ParamObject}` paraméterben, amely az előtétprogram-modul vagy a képernyő `ParamObject` objektumának egy másolata.

Két visszatérési formátum létezik ennél a függvénynél. A normál, amely egyszerűen a nyers érték, vagy egy szerkezet, amely a mezőnév => mezőérték páros. Például egy dátum dinamikus mező normális esetben a dátumot szövegként adja vissza, és ha egy szerkezetet kell visszaadnia, akkor a kivonatban egy párost ad vissza a dátum minden egyes részéhez.

Ha az eredménynek egy szerkezetnek kell lennie, akkor normális esetben ezt arra használják, hogy az értékét egy sablonban tárolja, mint például egy általános ügyintéző profilban. Például egy dátummező számos HTML összetevőt használ a mező felépítéséhez, mint például a „használt” jelölőnégyzetet és kiválasztókat az évhez, hónaphoz, naphoz, stb.

```
sub EditFieldValueValidate { ... }
```

Ennek a függvénynek biztosítania kell legalább egy metódust az ellenőrzéshez, ha a mező üres, és hibát kell visszaadnia, ha a mező üres és kötelező, de végrehajthat további ellenőrzéseket is a másfajta mezőknél, mint például ha a kiválasztott lehetőség érvényes, vagy ha egy dátumnak csak a múltban kell lennie, stb. Egy egyéni hibaüzenetet is biztosíthat.

```
sub SearchFieldRender { ... }
```

Ezt a függvényt a jegykeresés párbeszédablak használja, és hasonló a `EditFieldRander()` függvényhez, de normális esetben egy keresési képernyőn kisebb változtatásokat kell elvégezni az összes mezőnél. Például ebben az esetben egy HTML szöveges beviteli mező használunk egy jelszóbeviteli mező helyett. Más mezőkben (például legördülő mező) többválasztósként jelenik meg azért, hogy egyszerre több érték keresését tegye lehetővé a felhasználónak.

```
sub SearchFieldValueGet { ... }
```

Nagyon hasonló az `EditFieldValueGet()` függvényhez, de eltérő név előtagot használ a keresési párbeszédablak képernyőhöz átírva.

```
sub SearchFieldParameterBuild { ... }
```

A `SearchFieldParameterBuild()` függvényt is a jegykeresés párbeszédablak használja a helyes operátor és érték beállításához, hogy elvégezze a keresést ezen a mezőn. Azt is visszaadja, hogy az értéket hogyan kell megjeleníteni a használt keresési attribútumokban a találatok oldalon.

```
sub StatsFieldParameterBuild { ... }
```

Ezt a függvényt a statisztikák moduljai használják. Tartalmazza a mező meghatározást a statisztikák formátumában. A rögzített értékekkel rendelkező mezőknél tartalmazza az összes lehetséges értéket is, és ha azok lefordíthatók, akkor nézze meg a `BaseSelect` illesztőprogram kódját példaként arra, hogy azokat hogyan kell megvalósítani.

```
sub StatsSearchFieldParameterBuild { ... }
```

A `StatsSearchFieldParameterBuild()` nagyon hasonló a `SearchFieldParameterBuild()` függvényhez. A különbség az, hogy a `SearchFieldParameterBuild()` a keresési profilból kapja meg az értéket, és ez pedig közvetlenül a paramétereiből kapja meg az értéket.

Ezt a függvényt a statisztikák modul használja.

```
sub TemplateValueTypeGet { ... }
```

A `TemplateValueTypeGet()` függvényt annak megismeréséhez használják, hogy a dinamikus mező értékei hogyan vannak eltárolva egy olyan profilnál, amelyet le kell kérni (SCALAR vagy ARRAY formában), és meghatározza a mező helyes nevét is a profilban.

```
sub RandomValueSet { ... }
```

Ezt a függvényt az `otrs.FillDB.pl` parancsfájl használja, hogy feltöltse az adatbázist néhány teszt és véletlenszerű adattal. A függvény által beszúrt érték nem igazán fontos. Az egyetlen megkötés az, hogy az értéknek kompatibilisnek kell lennie a mezőérték típusával.

```
```

```
sub ObjectMatch { ... }
```

Az értesítési modulok használják. Ez a függvény 1-et ad vissza, ha a mező jelen van a `$Param{ObjectAttributes}` paraméterben, és ha megegyezik a megadott értékkel.

2.7.6. Egy dinamikus mező funkcionalitás kiterjesztés létrehozása

Ezen folyamat bemutatásához a *Foo* függvénynél egy új dinamikus mező funkcionalitás kiterjesztés lesz hozzáadva a háttérprogram objektumhoz, valamint a szövegmező illesztőprogramba.

A kiterjesztés létrehozásához három fájlt fogunk létrehozni: egy beállítófájlt (XML) a modulok regisztrálásához, egy háttérprogram kiterjesztést (Perl) az új függvény meghatározásához, és egy szövegmező illesztőprogram kiterjesztést (Perl), amely megvalósítja az új függvényt a szövegmezőknél.

Fájlszerkezet:

```
$HOME (például /opt/otrs/)
|
...
|--/Kernel/
|   |--/Config/
|   |   |--/Files/
|   |   |   |DynamicFieldFooExtension.xml
...
|   |--/System/
|   |   |--/DynamicField/
|   |   |   FooExtensionBackend.pm
|   |   |   |--/Driver/
|   |   |   |   |FooExtensionText.pm
...
```

2.7.6.1. Dinamikus mező Foo kiterjesztés fájlok

2.7.6.1.1. Dinamikus mező kiterjesztés beállítófájl példa

A beállítófájlokat használják a kiterjesztések regisztrálásához a háttérprogramnál és az illesztőprogramoknál, valamint az egyes illesztőprogramok új viselkedéseihez.

> **Megjegyzés**
>
> Ha egy illesztőprogramot kiterjesztenek egy új függvénnyel, akkor a háttérprogramnak is szüksége lesz egy kiterjesztésre ahhoz a függvényhez.

2.7.6.1.1.1. Kódpélda:

Ebben a szakaszban a Foo kiterjesztéshez egy beállítófájl van megjelenítve és elmagyarázva.

```
<?xml version="1.0" encoding="utf-8"?>
<otrs_config version="1.0" init="Application">
```

Ez a normál fejléc egy beállítófájlhoz.

```
    <ConfigItem Name="DynamicFields::Extension::Backend###100-Foo" Required="0" Valid="1">
        <Description Translatable="1">Dynamic Fields Extension.</Description>
        <Group>DynamicFieldFooExtension</Group>
```

```
        <SubGroup>DynamicFields::Extension::Registration</SubGroup>
        <Setting>
            <Hash>
                <Item Key="Module">Kernel::System::DynamicField::FooExtensionBackend</Item>
            </Hash>
        </Setting>
    </ConfigItem>
```

Ez a beállítás regisztrálja a kiterjeszt a Backend objektumban. A modul alaposztályként lesz betöltve a Backend objektumból.

```
    <ConfigItem Name="DynamicFields::Extension::Driver::Text###100-Foo" Required="0"
 Valid="1">
        <Description Translatable="1">Dynamic Fields Extension.</Description>
        <Group>DynamicFieldFooExtension</Group>
        <SubGroup>DynamicFields::Extension::Registration</SubGroup>
        <Setting>
            <Hash>
                <Item Key="Module">Kernel::System::DynamicField::Driver::FooExtensionText</
Item>
                <Item Key="Behaviors">
                    <Hash>
                        <Item Key="Foo">1</Item>
                    </Hash>
                </Item>
            </Hash>
        </Setting>
    </ConfigItem>
```

Ez egy kiterjesztés regisztrációja a `Text` dinamikus mező illesztőprogramban. A modul alaposztályként lesz betöltve az illesztőprogramban. Figyeljen arra is, hogy új viselkedések is megadhatók. Ezek a kiterjesztett viselkedések lesznek hozzáadva azokhoz a viselkedésekhez, amelyekkel az illesztőprogram eredetileg rendelkezik, ebből adódóan a `HasBehavior()` hívásával azt ellenőrizve, hogy ezeknél az új viselkedéseknél teljesen átlátszó legyen.

```
</otrs_config>
```

Egy beállítófájl szabványos lezárása.

2.7.6.1.2. Dinamikus mező háttérprogram kiterjesztés példa

A háttérprogram kiterjesztések átláthatóan lesznek betöltve magába a háttérprogramba egy alaposztályként. A háttérprogramból az összes meghatározott objektum és tulajdonság elérhető lesz a kiterjesztésben.

Megjegyzés

A háttérprogram kiterjesztésben meghatározott összes új függvényt meg kell valósítani egy illesztőprogram kiterjesztésben.

2.7.6.1.2.1. Kódpélda:

Ebben a szakaszban a háttérprogramhoz készített Foo kiterjesztés van megjelenítve és elmagyarázva. A kiterjesztés csak a `Foo()` függvényt határozza meg.

```
# --
# Kernel/System/DynamicField/FooExtensionBackend.pm - Extension for DynamicField backend
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
```

```
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::FooExtensionBackend;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);

=head1 NAME

Kernel::System::DynamicField::FooExtensionBackend

=head1 SYNOPSIS

DynamicFields Extension for Backend

=head1 PUBLIC INTERFACE

=over 4

=cut
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

```
=item Foo()

Tesztelő függvény: 1-et ad vissza, ha a függvény elérhető egy dinamikus mező
 illesztőprogramnál.

    my $Success = $BackendObject->Foo(
        DynamicFieldConfig   => $DynamicFieldConfig,      # a dinamikus mező teljes
 beállítása
    );

Returns:
    $Success = 1;       # vagy undef

=cut

sub Foo {
    my ( $Self, %Param ) = @_;

    # a szükséges dolgok ellenőrzése
    for my $Needed (qw(DynamicFieldConfig)) {
        if ( !$Param{$Needed} ) {
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'error',
                Message  => "Szükséges: $Needed!",
            );

            return;
        }
    }

    # a DynamicFieldConfig ellenőrzése (általánosan)
    if ( !IsHashRefWithData( $Param{DynamicFieldConfig} ) ) {
        $Kernel::OM->Get('Kernel::System::Log')->Log(
            Priority => 'error',
            Message  => "A mező beállítása érvénytelen",
        );

        return;
    }

    # a DynamicFieldConfig ellenőrzése (belsőleg)
    for my $Needed (qw(ID FieldType ObjectType)) {
```

```
        if ( !$Param{DynamicFieldConfig}->{$Needed} ) {
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'error',
                Message  => "$Needed szükséges ebben: DynamicFieldConfig!",
            );

            return;
        }
    }

    # a dinamikus mezőre jellemző háttérprogram beállítása
    my $DynamicFieldBackend = 'DynamicField' . $Param{DynamicFieldConfig}->{FieldType} .
 'Object';

    if ( !$Self->{$DynamicFieldBackend} ) {
        $Kernel::OM->Get('Kernel::System::Log')->Log(
            Priority => 'error',
            Message  => "A háttérprogram érvénytelen: $Param{DynamicFieldConfig}-
>{FieldType}!",
        );

        return;
    }

    # annak ellenőrzése, hogy a függvény elérhető-e
    return if !$Self->{$DynamicFieldBackend}->can('Foo');

    # a HasBehavior meghívása az adott háttérprogramnál
    return $Self->{$DynamicFieldBackend}->Foo(%Param);
}
```

A `Foo()` függvényt kizárólag tesztelési célokra használják. Először leellenőrzi a dinamikus mező beállításait, majd azt ellenőrzi, hogy a dinamikus mező illesztőprogram (típus) létezik-e, és betöltésre került-e. Annak megakadályozásához, hogy egy olyan illesztőprogramnál történjen függvényhívás, ahol nincs meghatározva, először azt ellenőrzi, hogy az illesztőprogram képes-e végrehajtani a függvényt, majd végrehajtja a függvényt az illesztőprogramban az összes paramétert átadva.

Megjegyzés

Lehetséges annak a lépésnek a kihagyása is, amely azt próbálja, hogy az illesztőprogram képes-e végrehajtani a függvényt. Ennek elvégzéséhez szükséges egy mechanizmus megvalósítása az előtétprogram modulban egy különleges viselkedés megköveteléséhez a dinamikus mezőnél, és csak azután hívja meg a függvényt a háttérprogram objektumban.

2.7.6.1.3. Dinamikus mező illesztőprogram kiterjesztés példa

Az illesztőprogram kiterjesztések átláthatóan lesznek betöltve magába az illesztőprogramba egy alaposztályként. Az illesztőprogramból az összes meghatározott objektum és tulajdonság elérhető lesz a kiterjesztésben.

Megjegyzés

Az illesztőprogram kiterjesztésben megvalósított összes új függvényt meg kell határozni egy háttérprogram kiterjesztésben, mivel minden függvény a háttérprogram objektumból kerül meghívásra.

2.7.6.1.3.1. Kódpélda:

Ebben a szakaszban a szövegmező illesztőprogramhoz készített Foo kiterjesztés van megjelenítve és elmagyarázva. A kiterjesztés csak a `Foo()` függvényt valósítja meg.

```
```

```
# --
# Kernel/System/DynamicField/Driver/FooExtensionText.pm - Extension for DynamicField Text
 Driver
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::DynamicField::Driver::FooExtensionText;

use strict;
use warnings;

=head1 NAME

Kernel::System::DynamicField::Driver::FooExtensionText

=head1 SYNOPSIS

DynamicFields Text Driver Extension

=head1 PUBLIC INTERFACE

This module extends the public interface of L<Kernel::System::DynamicField::Backend>.
Please look there for a detailed reference of the functions.

=over 4

=cut
```

Ez egy gyakori fejléc, amely megtalálható a szokásos OTRS modulokban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

```
sub Foo {
    my ( $Self, %Param ) = @_;
    return 1;
}
```

A Foo() függvénynek nincs különleges logikája. Csak tesztelésre van, és mindig 1-et ad vissza.

2.8. E-mail kezelés

2.8.1. Jegy levelezési modul

A levelezési modulokat a levelezési folyamat során használják. Kétféle levelezési modul létezik: PostMasterPre (egy e-mail feldolgozása után használják) és PostMasterPost (azután használják, amikor egy e-mail feldolgozásra került, és az adatbázisban van).

Ha saját levelezési szűrőket szeretne létrehozni, akkor egyszerűen hozza létre a saját modulját. Ezek a modulok a Kernel/System/PostMaster/Filter/*.pm fájlokban találhatók. Az alapértelmezett modulokért nézze meg az adminisztrátori kézikönyvet. Mindössze két függvényre van szüksége: new() és Run().

A következőkben egy példaszerű modul található az e-mailek egyeztetéséhez és az X-OTRS fejlécek beállításához (további információkért nézze meg a doc/X-OTRS-Headers.txt fájlt).

Kernel/Config/Files/MyModule.xml:

```
```

```
<ConfigItem Name="PostMaster::PreFilterModule###1-Example" Required="0" Valid="1">
    <Description Translatable="1">Példamodul a bejövő üzenetek szűréséhez és
 manipulálásához.</Description>
    <Group>Ticket</Group>
    <SubGroup>Core::PostMaster</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::System::PostMaster::Filter::Example</Item>
            <Item Key="Match">
                <Hash>
                    <Item Key="From">noreply@</Item>
                </Hash>
            </Item>
            <Item Key="Set">
                <Hash>
                    <Item Key="X-OTRS-Ignore">yes</Item>
                </Hash>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

És a tényleges szűrőkód a `Kernel/System/PostMaster/Filter/Example.pm` fájlban:

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::PostMaster::Filter::Example;

use strict;
use warnings;

our @ObjectDependencies = (
    'Kernel::System::Log',
);

sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {};
    bless ($Self, $Type);

    $Self->{Debug} = $Param{Debug} || 0;

    return $Self;
}

sub Run {
    my ( $Self, %Param ) = @_;
    # get config options
    my %Config = ();
    my %Match = ();
    my %Set = ();
    if ($Param{JobConfig} && ref($Param{JobConfig}) eq 'HASH') {
        %Config = %{$Param{JobConfig}};
        if ($Config{Match}) {
            %Match = %{$Config{Match}};
        }
        if ($Config{Set}) {
            %Set = %{$Config{Set}};
        }
        }
        # match 'Match => ???' stuff
        my $Matched = '';
```

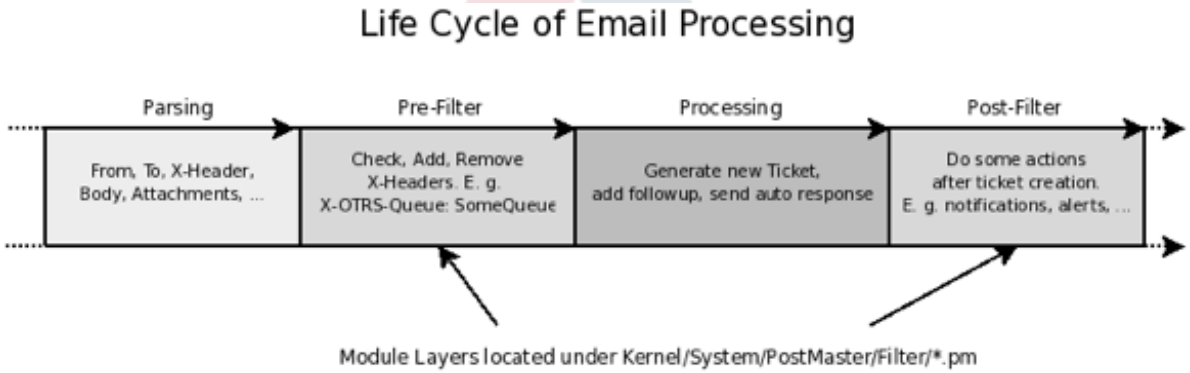
```
        my $MatchedNot = 0;
        for (sort keys %Match) {
        if ($Param{GetParam}->{$_} && $Param{GetParam}->{$_} =~ /$Match{$_}/i) {
            $Matched = $1 || '1';
            if ($Self->{Debug} > 1) {
                $Kernel::OM->Get('Kernel::System::Log')->Log(
                    Priority => 'debug',
                    Message => "'$Param{GetParam}->{$_}' =~ /$Match{$_}/i matched!",
                );
            }
        }
        else {
            $MatchedNot = 1;
            if ($Self->{Debug} > 1) {
                $Kernel::OM->Get('Kernel::System::Log')->Log(
                    Priority => 'debug',
                    Message => "'$Param{GetParam}->{$_}' =~ /$Match{$_}/i matched NOT!",
                );
            }
        }
        }
        # should I ignore the incoming mail?
        if ($Matched && !$MatchedNot) {
        for (keys %Set) {
            if ($Set{$_} =~ /\[\*\*\*\]/i) {
                $Set{$_} = $Matched;
            }
            $Param{GetParam}->{$_} = $Set{$_};
            $Kernel::OM->Get('Kernel::System::Log')->Log(
                Priority => 'notice',
                Message => "Set param '$_' to '$Set{$_}' (Message-ID: $Param{GetParam}-
>{'Message-ID'}) ",
            );
        }
    }

    return 1;
}

1;
```

A következő kép az e-mail feldolgozási folyamatát mutatja be.

3.4. ábra - E-mail feldolgozási folyamat



2.9. Process Management

2.9.1. Process Management

Since OTRS 7 Process Management can use script modules to perform Activities (script tasks) and/or Sequence Flow Actions (know before as Transition Actions). This modules are located in `Kernel::System::ProcessManagement::Modules`.

2.9.1.1. Process Management Modules

The Process Management Modules are scripts written in Perl language to perform certain action or actions over the process ticket like set a dynamic field or change a queue etc. Or any other part of the system like create new tickets.

By default Modules uses a set of key value pairs to use them as parameters for the action e.g. to change queue of the process ticket, the queue or queue id and its corresponding value is needed.

Some scripts might require more than a simple key value pairs as parameters or its configuration might need to have a more user friendly GUI, in such cases OTRS provides some configuration field types that can be also extended if needed.

Current field types:

- Dropdown

  Shows a drop down list with predefined values.

- KeyValueList

  Shows a list of simple key value pairs (text inputs). Pairs can be added or deleted.

- MultiLanguageRichText

  Shows a rich text editor associated to a system language, also shows a language selector to add rich text fields for the proper selected language.

- Recipients

  Shows a multi select field pre-filled with agents to be used as email recipients, also displays a free input field to be used to specify external email addresses to be added to the recipients list.

- RichText

  Shows a single rich text field.

2.9.1.1.1. Creating a New Process Management Module

The following is an example of how to create a Process Management module, it includes a section where all possible fields are defined as a reference, to create a new module only one field type is needed but consider that by convention the parameter UserID is used to impersonate all the actions in the module with another user than the one that triggers the action, then it is a good practice to always include the `KeyValueList` field type along with any other needed field.

2.9.1.1.1.1. Process Management Module Code Example

The following code implements a dummy process management module that can be used in script task activities or sequence flow actions.

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::System::ProcessManagement::Modules::Test;
```

```
use strict;
use warnings;
use utf8;

use Kernel::System::VariableCheck qw(:all);

use parent qw(Kernel::System::ProcessManagement::Modules::Base);

our @ObjectDependencies = ( );
```

Ez egy gyakori fejléc, amely megtalálható a legtöbb OTRS modulban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

In this case we are inheriting from Base class, and the object manager dependencies are set.

```
sub new {
    my ( $Type, %Param ) = @_;

    # Allocate new hash for object.
    my $Self = {};
    bless( $Self, $Type );

    $Self->{Config} = {
        Description => 'Description for the module.',
        ConfigSets  => [
            {
                Type        => 'Dropdown',
                Description => 'Description for Dropdown.',
                Mandatory   => 1,
                Config      => {
                    Data => {
                        Internal1 => 'Display 1',
                        Internal2 => 'Display 2',
                    },
                    Name         => 'Test-DropDown',
                    Multiple     => 1,
                    PossibleNone => 1,
                    Sort         => 'AlphanumericValue',
                    Translation  => 1,
                },
            },
            {
                Type        => 'KeyValueList',
                Description => 'Description of the Key/Value pairs',
                Defaults    => [
                    {
                        Key       => 'Test-Param1',
                        Value     => 'Hello',
                        Mandatory => 1,
                    },
                    {
                        Key       => 'Test-Param2',
                        Mandatory => 1,
                    },
                    {
                        Key       => 'Test-Param3',
                        Value     => 'World',
                        Mandatory => 0,
                    },
                ],
            },
            {
                Type        => 'MultiLanguageRichText',
                Description => "Description for Mutli-Language Rich Text.",
                Defaults    => [
                    {
                        Key   => 'en_Subject',
```

```
                        Value => 'Hello',
                    },
                    {
                        Key   => 'en_Body',
                        Value => 'World',
                    },

                ],
            },
            {
                Type        => 'Recipients',
                Description => "Description for Recipients."
            },
            {
                Type        => 'RichText',
                Description => "Description for Rich Text.",
                Defaults    => [
                    {
                        Value     => 'Hello World',
                        Mandatory => 1,
                    },
                ],
            },
        ],
    };

    return $Self;
}
```

The constructor `new()` creates a new instance of the class. The configuration fields are defined here and they are set in `$Self->{Config}`.

The configuration has two main entries:

- `Description`:

  Is is used to to explain the administrators what does the module do and/or considerations for its configuration.

- `ConfigSets`:

  This is just a container for the actual configuration fields

All configuration fields requires a Type: that defines the kind of field and they could also have an internal `Description`: to be used a the title of the field widget, if its not defined a default description is used.

Each field defines its configuration parameters and capabilities, the following is a small reference for the fields provided my OTRS out of the box.

- `Dropdown`:
  - `Mandatory`:

    Used to define if a value is required to be set.

  - `Config`:

    Holds the information to display the drop-down field

    - `Data`:

      Simple hash that defines the options for the Dropdown, the keys are used internally, and the values are the options that the user see in the screen.

    - `Name`:

The name of the parameter.

- `Multiple`:

  To define if only one or multiple values can be selected.

- `PossibleNone`:

  Defines if the list of values offer an empty value or not.

- `Sort`:

  Defines how the options will be sorted when the field is render, the possible values are: `AlphanumericValue`, `NumericValue>`, `AlphanumericKey` and `NumericKey`.

- `Translation`:

  Set if the display values should be translated.

- `KeyValueList`:

  - `Defaults`:

    Array of Hashes that holds the default configuration for its key value pairs.

    - Key:

      The name of a parameter.

    - `Value`:

      The default value of the parameter (optional).

    - `Mandatory`:

      Mandatory parameters can not be renamed or removed (optional).

- `MultiLanguageRichText`:

  - `Defaults`:

    Array of Hashes that holds the default configuration each language and field part.

    - Key:

      Its composed by language such as en or es_MX, followed by a '_' (underscore character) and then `Subject` or Body for the corresponding part of the field.

    - `Value`:

      The default value of the field part (optional).

- `Recipients`:

  No further configuration is provided for this kind of field.

- `RichText`:

  - `Defaults`:

    Array of Hashes that holds the default configuration field (only the first element is used).

- `Value`:

  The default value of the field.

- `Mandatory`:

  Used to define if a value is required to be set.

```
sub Run {
    my ( $Self, %Param ) = @_;

    # Define a common message to output in case of any error.
    my $CommonMessage = "Process: $Param{ProcessEntityID} Activity:
 $Param{ActivityEntityID}";

    # Add SequenceFlowEntityID to common message if available.
    if ( $Param{SequenceFlowEntityID} ) {
        $CommonMessage .= " SequenceFlow: $Param{SequenceFlowEntityID}";
    }

    # Add SequenceFlowActionEntityID to common message if available.
    if ( $Param{SequenceFlowActionEntityID} ) {
        $CommonMessage .= " SequenceFlowAction: $Param{SequenceFlowActionEntityID}";
    }

    # Check for missing or wrong params.
    my $Success = $Self->_CheckParams(
        %Param,
        CommonMessage => $CommonMessage,
    );
    return if !$Success;

    # Override UserID if specified as a parameter in the TA config.
    $Param{UserID} = $Self->_OverrideUserID(%Param);

    # Use ticket attributes if needed.
    $Self->_ReplaceTicketAttributes(%Param);
    $Self->_ReplaceAdditionalAttributes(%Param);

    # Get module configuration.
    my $ModuleConfig = $Param{Config};

    # Add module logic here!

    return 1;
}
```

The Run method Is the main part of the module. First sets a Common Message that can be used in error logs or any other purpose, for consistency its highly recommended to use it as described above.

Next step is to check if the global parameters was sent correctly.

By Convention all modules should be able to override the current UserID is one is provided in the parameters (if any), this passed UserID should be used in any function call that requires it.

User defined attribute values can use current ticket values by using OTRS smart tags, _ReplaceTicketAttributes is used for normal text attributes, while _ReplaceAdditionalAttributes for rich texts. For more complex parameters it might need customized functions to replace this smart tags

The following is the proper logic of the module.

If everything was OK it must return 1.

2.9.1.1.2. Creating a New Process Management Module Configuration Field

The following is an example of how to create a Process Management module configuration field, this field can be used by any Process Management module after its configuration.

2.9.1.1.2.1. Process Management Module Configuration Field Code Example

The following code implements a simple input process management module configuration field (Test). The name of the field and its default value can be set trough a process management module ConfigSets.

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Output::HTML::ProcessManagement::ModuleConfiguration::Test;

use strict;
use warnings;

use Kernel::System::VariableCheck qw(:all);
use Kernel::Language qw(Translatable);

our @ObjectDependencies = (
    'Kernel::Output::HTML::Layout',
    'Kernel::System::Web::Request',
);
```

Ez egy gyakori fejléc, amely megtalálható a legtöbb OTRS modulban. Az osztály/csomag neve a package kulcsszón keresztül van deklarálva.

```
sub new {
    my ( $Type, %Param ) = @_;

    # allocate new hash for object
    my $Self = {%Param};
    bless( $Self, $Type );

    return $Self;
}
```

A new konstruktor hozza létre az osztály új példányát.

Every configuration field requires to implement at least 2 main methods `Render` and `GetParams`.

```
sub Render {
    my ( $Self, %Param ) = @_;

    my $ConfigSet = $Param{ConfigSet} // {};
    my $EntityConfig = $Param{EntityData}->{Config}->{Config}->{ConfigTest} // {};

    my %Data;

    $Data{Description} = $ConfigSet->{Description} || 'Config Parameters (Test)';
    $Data{Name} = $ConfigSet->{Config}->{Name} // 'Test';
    $Data{Value} = $EntityConfig->{ $ConfigSet->{Config}->{Name} } // $ConfigSet-
>{Defaults}->[0]->{Value} // '';

    return {
```

```
        Success => 1,
        HTML    => $Kernel::OM->Get('Kernel::Output::HTML::Layout')->Output(
            TemplateFile => 'ProcessManagement/ModuleConfiguration/Test',
            Data         => \%Data,
        ),
    };
}
```

Render method is responsible to create the required HTML for the field.

In This example it first localize some parameters for more easy read and maintain.

The following lines sets the data to display, the field widget title Description is gather from the ConfigSet if defined, otherwise it uses a default text. Similar to the field Name, for the Valueit first checks if the activity or sequence flow action already have an stored value, if not it tries to use the default value from the ConfigSet, or use empty otherwise.

At the end it returns a structure with the HTML code from a template filled with the gathered data.

```
sub GetParams {
    my ( $Self, %Param ) = @_;

    my %GetParams;
    my $Config = $Param{ConfigSet}->{Config} // 'Test';

    $GetParams{ $Config->{Name} } = $Kernel::OM->Get('Kernel::System::Web::Request')-
>GetParam( Param => $Config->{Name} );

    return \%GetParams;
}
```

For this example the GetParams method is very straight forward, it get the name of the field from the ConfigSet or use a default, and gets the value from the web request.

2.9.1.1.2.2. Process Management Module Configuration Field Template Example

The following code implements a basic HTML template for the Test Process Management module configuration field.

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

This is common header that can be found in most OTRS modules.

```
<div id="TestConfig" class="WidgetSimple Expanded">
    <div class="Header">
        <div class="WidgetAction Toggle">
            <a href="#" title="[% Translate("Show or hide the content") | html %]"><i
 class="fa fa-caret-right"></i><i class="fa fa-caret-down"></i></a>
        </div>
        <h2 for="Config">[% Translate(Data.Description) | html %]</h2>
    </div>
    <div class="Content" id="TestParams">
        <fieldset class="TableLike">
            <label for="[% Data.Name | html %]">[% Data.Name | html %]</label>
```

```
            <div class="Field">
                <input type="text" value="[% Data.Value | html %]" name="[% Data.Name | html
 %]" id="[% Data.Name | html %]" />
            </div>
        </fieldset>
    </div>
</div>
```

The template shows a simple text input element with its associated label.

4. fejezet - Hogyan tehetők közzé az OTRS kiterjesztések

1. Csomagkezelés

Az OPM (OTRS csomagkezelő) egy mechanizmus az OTRS keretrendszerhez való szoftvercsomagok terjesztésére HTTP-n, FTP-n vagy fájlfeltöltésen keresztül.

Például az OTRS projekt OTRS modulokat kínál OTRS csomagokban az Internetes tárolókon vagy az FTP-kiszolgálóinkon keresztül, mint például naptár, fájlkezelő vagy webes levelező. A csomagok az adminisztrátori felületen keresztül kezelhetők (telepítés, frissítés vagy eltávolítás).

1.1. Csomagterjesztés

Ha egy internetes OPM tárolót szeretne létrehozni, akkor egyszerűen mondja meg az OTRS keretrendszernek a `Package::RepositoryList` rendszerbeállítási lehetőség bekapcsolásával, hogy hol van annak a helye, és adja meg az új helyet itt. Ezután egy új választási lehetősége lesz a csomagkezelőben.

A tárolójában hozzon létre egy index fájlt az OPM csomagokhoz. Az OTRS egyszerűen beolvassa ezt az index fájlt, és tudni fogja, hogy mely csomagok érhetők el.

```
shell> bin/otrs.Console.pl Dev::Package::RepositoryIndex /útvonal/a/tárolóhoz/ > /útvonal/a/
tárolóhoz/otrs.xml
```

1.2. Csomagparancsok

A következő OPM parancsokat használhatja az adminisztrátori felületen vagy a `bin/otrs.PackageManager.pl` parancsfájllal az adminisztrátori feladatok kezeléséhez az OPM csomagoknál.

1.2.1. Telepítés

OPM csomagok telepítése.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor:

```
shell> bin/otrsConsole.pl Admin::Package::Install /útvonal/ehhez/csomag.opm
```

1.2.2. Eltávolítás

OPM csomagok eltávolítása.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor:

```
shell> bin/otrsConsole.pl Admin::Package::Uninstall /útvonal/ehhez/csomag.opm
```

1.2.3. Frissítés

OPM csomagok frissítése.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor:

```
shell> bin/otrsConsole.pl Admin::Package::Upgrade /útvonal/ehhez/csomag.opm
```

1.2.4. Felsorolás

Az összes OPM csomag felsorolása.

- Web: http://localhost/otrs/index.pl?Action=AdminPackageManager
- Parancssor:

```
shell> bin/otrsConsole.pl Admin::Package::List
```

2. Csomagkészítés

Ha egy OPM csomagot (`.opm`) szeretne létrehozni, akkor létre kell hoznia egy specifikációs fájl (`.sopm`), amely a csomag tulajdonságait tartalmazza.

2.1. Csomagspecifikációs fájl

Az OPM csomag XML alapú. A `.sopm` fájlt egy szöveg- vagy egy XML-szerkesztővel hozhatja létre és szerkesztheti. Ez metaadatokat, egy fájllistát és adatbázis-beállításokat tartalmaz.

2.1.1. <Name>

A csomag neve (kötelező).

```
<Name>Naptár</Name>
```

2.1.2. <Version>

A csomag verziója (kötelező).

```
<Version>1.2.3</Version>
```

2.1.3. <Framework>

A megcélzott keretrendszer verziója (a 3.2.x jelentése például 3.2.1 vagy 3.2.2) (kötelező).

```
<Framework>3.2.x</Framework>
```

Több alkalommal is használható.

```
<Framework>3.0.x</Framework>
<Framework>3.1.x</Framework>
<Framework>3.2.x</Framework>
```

2.1.4. <Vendor>

A csomag gyártója (kötelező).

```
<Vendor>OTRS AG</Vendor>
```

2.1.5. <URL>

A gyártó URL-e (kötelező).

```
<URL>https://otrs.com/</URL>
```

2.1.6. <License>

A csomag licence (kötelező).

```
<License>GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007</License>
```

2.1.7. <ChangeLog>

A csomag változásnaplója (elhagyható).

```
<ChangeLog Version="1.1.2" Date="2013-02-15 18:45:21">Néhány funkció hozzáadva.</ChangeLog>
<ChangeLog Version="1.1.1" Date="2013-02-15 16:17:51">Új csomag.</ChangeLog>
```

2.1.8. <Description>

A csomag leírása különböző nyelveken (kötelező).

```
<Description Lang="en">A web calendar.</Description>
<Description Lang="hu">Egy webes naptár.</Description>
```

2.1.9. Csomagműveletek

A csomag lehetséges műveletei a telepítés után. Ha ezen műveletek valamelyike nincs meghatározva a csomagnál, akkor lehetségesként fogja tekinteni.

```
<PackageIsVisible>1</PackageIsVisible>
<PackageIsDownloadable>0</PackageIsDownloadable>
<PackageIsRemovable>1</PackageIsRemovable>
```

A special package action is PackageAllowDirectUpdate. Only if it is defined on the package and set to true, a package can be upgraded from a lower major version (earlier than the last one) to the latest version. (e.g. a package for OTRS 5 updated to OTRS 7).

```
<PackageAllowDirectUpdate>1</PackageAllowDirectUpdate>
```

2.1.10. <BuildHost>

Ezt az OPM automatikusan ki fogja tölteni.

```
<BuildHost>?</BuildHost>
```

2.1.11. <BuildDate>

Ezt az OPM automatikusan ki fogja tölteni.

```
<BuildDate>?</BuildDate>
```

2.1.12. <PackageRequired>

Csomagok, amelyeket előzetesen telepíteni kell (elhagyható). Ha a PackageRequired használatban van, akkor a szükséges csomag egy verzióját meg kell adni.

```
<PackageRequired Version="1.0.3">ValamilyenMasikCsomag</PackageRequired>
<PackageRequired Version="5.3.2">ValamilyenMasikCsomag2</PackageRequired>
```

2.1.13. <ModuleRequired>

Perl-modulok, amelyeket előzetesen telepíteni kell (elhagyható).

```
<ModuleRequired Version="1.03">Encode</ModuleRequired>
<ModuleRequired Version="5.32">MIME::Tools</ModuleRequired>
```

2.1.14. <OS>

A szükséges operációs rendszer (elhagyható).

```
<OS>linux</OS>
<OS>darwin</OS>
<OS>mswin32</OS>
```

2.1.15. <Filelist>

Ez a csomagban lévő fájlok listája (elhagyható).

```
<Filelist>
    <File Permission="644" Location="Kernel/Config/Files/Calendar.pm"/>
    <File Permission="644" Location="Kernel/System/CalendarEvent.pm"/>
    <File Permission="644" Location="Kernel/Modules/AgentCalendar.pm"/>
    <File Permission="644" Location="Kernel/Language/de_AgentCalendar.pm"/>
</Filelist>
```

2.1.16. <DatabaseInstall>

Adatbázis-bejegyzések, amelyeket létre kell hozni, amikor a csomagot telepítik (elhagyható).

```
<DatabaseInstall>
    <TableCreate Name="calendar_event">
    <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement="true" Type="BIGINT"/>
    <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
    <Column Name="content" Required="false" Size="250" Type="VARCHAR"/>
    <Column Name="start_time" Required="true" Type="DATE"/>
    <Column Name="end_time" Required="true" Type="DATE"/>
    <Column Name="owner_id" Required="true" Type="INTEGER"/>
    <Column Name="event_status" Required="true" Size="50" Type="VARCHAR"/>
    </TableCreate>
</DatabaseInstall>
```

Választhat <DatabaseInstall Type="post"> vagy <DatabaseInstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.17. <DatabaseUpgrade>

Információk arról, hogy mely műveleteket kell végrehajtani egy frissítés esetén (elhagyható). Például ha egy korábban telepített csomag verziója 1.3.4 alatt van (mondjuk 1.2.6), akkor végre lesz hajtva a meghatározott művelet:

```
<DatabaseUpgrade>
    <TableCreate Name="calendar_event_involved" Version="1.3.4">
        <Column Name="event_id" Required="true" Type="BIGINT"/>
        <Column Name="user_id" Required="true" Type="INTEGER"/>
    </TableCreate>
</DatabaseUpgrade>
```

Választhat <DatabaseUpgrade Type="post"> vagy <DatabaseUpgrade Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.18. <DatabaseReinstall>

Információk arról, hogy mely műveleteket kell végrehajtani, ha a csomagot újratelepítik (elhagyható).

```
<DatabaseReinstall></DatabaseReinstall>
```

Választhat <DatabaseReinstall Type="post"> vagy <DatabaseReinstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.19. <DatabaseUninstall>

A végrehajtandó műveletek a csomag eltávolításakor (elhagyható).

```
<DatabaseUninstall>
    <TableDrop Name="calendar_event" />
```

```
</DatabaseUninstall>
```

Választhat <DatabaseUninstall Type="post"> vagy <DatabaseUninstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.20. <IntroInstall>

Egy telepítés előtti („pre”) vagy utáni („post”) bevezető megjelenítéséhez a telepítési párbeszédablakban.

```
<IntroInstall Type="post" Lang="hu" Title="Valamilyen cím"><![CDATA[
Valamilyen HTML formátumú információ…
]]></IntroInstall>
```

Használhatja a Format attribútumot is annak meghatározásához, hogy „html” (amely alapértelmezett) vagy „plain” (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy <pre></pre> címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

2.1.21. <IntroUninstall>

Egy eltávolítás előtti („pre”) vagy utáni („post”) bevezető megjelenítéséhez az eltávolítási párbeszédablakban.

```
<IntroUninstall Type="post" Lang="hu" Title="Valamilyen cím"><![CDATA[
Valamilyen HTML formátumú információ…
]]></IntroUninstall>
```

Használhatja a Format attribútumot is annak meghatározásához, hogy „html” (amely alapértelmezett) vagy „plain” (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy <pre></pre> címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

2.1.22. <IntroReinstall>

Egy újratelepítés előtti („pre”) vagy utáni („post”) bevezető megjelenítéséhez az újratelepítési párbeszédablakban.

```
<IntroReinstall Type="post" Lang="hu" Title="Valamilyen cím"><![CDATA[
Valamilyen HTML formátumú információ…
]]></IntroReinstall>
```

Használhatja a Format attribútumot is annak meghatározásához, hogy „html” (amely alapértelmezett) vagy „plain” (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy <pre></pre> címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

2.1.23. <IntroUpgrade>

Egy frissítés előtti („pre”) vagy utáni („post”) bevezető megjelenítéséhez a frissítési párbeszédablakban.

```
<IntroUpgrade Type="post" Lang="hu" Title="Valamilyen cím"><![CDATA[
Valamilyen HTML formátumú információ…
```

```
]]></IntroUpgrade>
```

Használhatja a Format attribútumot is annak meghatározásához, hogy „html” (amely alapértelmezett) vagy „plain” (egyszerű szöveg) tartalmat szeretne használni. Az utóbbi automatikusan egy <pre></pre> címkét használ, amikor a bevezető megjelenik (a tartalom új sorai és üres karakterei megtartásához).

2.1.24. <CodeInstall>

A végrehajtandó Perl-kód, amikor a csomagot telepítik (elhagyható).

```
<CodeInstall><![CDATA[
# log example
$Kernel::OM->Get('Kernel::System::Log')->Log(
    Priority => 'notice',
    Message => "Valamilyen üzenet!",
);
# adatbázis példa
$Kernel::OM->Get('Kernel::System::DB')->Do(SQL => "VALAMILYEN SQL");
]]></CodeInstall>
```

Választhat <CodeInstall Type="post"> vagy <CodeInstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.25. <CodeUninstall>

A végrehajtandó Perl-kód, amikor a csomagot eltávolítják (elhagyható). A csomag eltávolításának előtti („pre”) vagy utáni („post”) idejében.

```
<CodeUninstall><![CDATA[
...
]]></CodeUninstall>
```

Választhat <CodeUninstall Type="post"> vagy <CodeUninstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.26. <CodeReinstall>

A végrehajtandó Perl-kód, amikor a csomagot újratelepítik (elhagyható).

```
<CodeReinstall><![CDATA[
...
]]></CodeReinstall>
```

Választhat <CodeReinstall Type="post"> vagy <CodeReinstall Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.27. <CodeUpgrade>

A végrehajtandó Perl-kód, amikor a csomagot frissítik (a version címkétől függően), (elhagyható). Például ha egy korábban telepített csomag verziója 1.3.4 alatt van (mondjuk 1.2.6), akkor végre lesz hajtva a meghatározott művelet:

```
```

```
<CodeUpgrade Version="1.3.4"><![CDATA[
...
]]></CodeUpgrade>
```

Választhat <CodeUpgrade Type="post"> vagy <CodeUpgrade Type="pre"> típust is a végrehajtás idejének külön-külön történő meghatározásához (a post az alapértelmezett). További információkért nézze meg a csomagéletciklus szakaszt.

2.1.28. <PackageMerge>

Ez a címke jelzi, hogy egy csomag egyesítve lett egy másik csomaggal. Ebben az esetben az eredeti csomagot el kell távolítani a fájlrendszerről és a csomagok adatbázisából, de az összes adatot meg kell tartani. Tegyük fel, hogy az ElsoCsomag egyesítve lett a MasodikCsomag nevű csomaggal. Ekkor a MasodikCsomag.sopm fájlnak ezt kell tartalmaznia:

```
<PackageMerge Name="MergeOne" TargetVersion="2.0.0"></PackageMerge>
```

Ha az ElsoCsomag is tartalmaz adatbázis-szerkezetet, akkor meg kell győződnünk arról, hogy az a csomag legfrissebb elérhető verziójánál volt, hogy következetes állapot legyen az adatbázisban a csomag egyesítése után. A TargetVersion attribútum csak ennyi csinál: jelzi az ElsoCsomag utolsó ismert verzióját abban az időpontban, amikor a MasidikCsomag létrejött. Ez főleg azért van, hogy leállítsa a frissítési folyamatot, ha a felhasználó rendszerén megtalálható az ElsoCsomag egy olyan verziója, amely *újabb* a TargetVersion attribútumban megadottnál, mivel ekkor ez problémákhoz vezethet.

Továbbá lehetőség van a szükséges adatbázis és kódfrissítési címkék hozzáadására az ElsoCsomag nevű csomagnál annak biztosításához, hogy az megfelelően kerül frissítésre a TargetVersion verzióra az egyesítés *előtt* - a következetlenségi problémák elkerüléséhez. Itt látható, hogy ennek hogyan kellene kinéznie:

```
<PackageMerge Name="MergeOne" TargetVersion="2.0.0">
  <DatabaseUpgrade Type="merge">
      <TableCreate Name="merge_package">
          <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement="true"
 Type="INTEGER"/>
          <Column Name="description" Required="true" Size="200" Type="VARCHAR"/>
      </TableCreate>
  </DatabaseUpgrade>
</PackageMerge>
```

Amint láthatja, ebben az esetben a Type="merge" attribútumot kell beállítani. Ezek a szakaszok csak akkor lesznek végrehajtva, ha lehetséges egy csomagegyesítés.

2.1.29. Feltételek: IfPackage és IfNotPackage

Ezek az attribútumok hozzáadhatók a szabályos Database* és Code* szakaszokhoz. Ha ezek jelen vannak, akkor a szakasz csak akkor lesz végrehajtva, ha egy másik csomag létezik vagy nem létezik a helyi csomagtárolóban.

```
<DatabaseInstall IfPackage="ValamilyenCsomag">
    ...
</DatabaseInstall>
```

vagy

```
```

```
<CodeUpgrade IfNotPackage="MasikCsomag">
    ...
</CodeUpgrade>
```

Ezek az attribútumok beállíthatók a PackageMerge címkéken belüli Database* és Code* szakaszokban is.

2.2. Példa .sopm

Ez egy példa specifikációs fájl kinézete a fenti címkék egy részével.

```
<?xml version="1.0" encoding="utf-8" ?>
<otrs_package version="1.0">
    <Name>Calendar</Name>
    <Version>0.0.1</Version>
    <Framework>3.2.x</Framework>
    <Vendor>OTRS AG</Vendor>
    <URL>https://otrs.com/</URL>
    <License>GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007</License>
    <ChangeLog Version="1.1.2" Date="2013-02-15 18:45:21">Added some feature.</ChangeLog>
    <ChangeLog Version="1.1.1" Date="2013-02-15 16:17:51">New package.</ChangeLog>
    <Description Lang="en">A web calendar.</Description>
    <Description Lang="de">Ein Web Kalender.</Description>
    <IntroInstall Type="post" Lang="en" Title="Thank you!">Thank you for choosing the
 Calendar module.</IntroInstall>
    <IntroInstall Type="post" Lang="de" Title="Vielen Dank!">Vielen Dank fuer die Auswahl
 des Kalender Modules.</IntroInstall>
    <BuildDate>?</BuildDate>
    <BuildHost>?</BuildHost>
    <Filelist>
        <File Permission="644" Location="Kernel/Config/Files/Calendar.pm"></File>
        <File Permission="644" Location="Kernel/System/CalendarEvent.pm"></File>
        <File Permission="644" Location="Kernel/Modules/AgentCalendar.pm"></File>
        <File Permission="644" Location="Kernel/Language/de_AgentCalendar.pm"></File>
        <File Permission="644" Location="Kernel/Output/HTML/Standard/AgentCalendar.tt"></
File>
        <File Permission="644" Location="Kernel/Output/HTML/NotificationCalendar.pm"></File>
        <File Permission="644" Location="var/httpd/htdocs/images/Standard/calendar.png"></
File>
    </Filelist>
    <DatabaseInstall>
        <TableCreate Name="calendar_event">
            <Column Name="id" Required="true" PrimaryKey="true" AutoIncrement="true"
 Type="BIGINT"/>
            <Column Name="title" Required="true" Size="250" Type="VARCHAR"/>
            <Column Name="content" Required="false" Size="250" Type="VARCHAR"/>
            <Column Name="start_time" Required="true" Type="DATE"/>
            <Column Name="end_time" Required="true" Type="DATE"/>
            <Column Name="owner_id" Required="true" Type="INTEGER"/>
            <Column Name="event_status" Required="true" Size="50" Type="VARCHAR"/>
        </TableCreate>
    </DatabaseInstall>
    <DatabaseUninstall>
        <TableDrop Name="calendar_event"/>
    </DatabaseUninstall>
</otrs_package>
```

2.3. Csomagösszeállítás

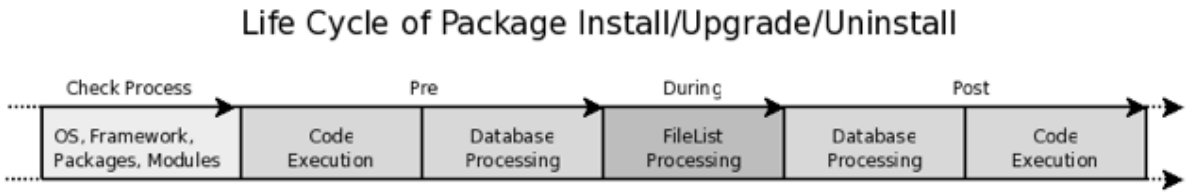
Egy .opm csomag összeállításához a specifikációs opm fájlból.

```
shell> bin/otrs.Console.pl Dev::Package::Build /útvonal/ehhez/példa.sopm /tmp
Building package...
Done.
shell>
```

2.4. Csomagéletciklus - telepítés, frissítés, eltávolítás

A következő kép azt mutatja be lépésről lépésre, hogy egy csomag telepítési, frissítési vagy eltávolítási életciklusa hogyan működik a háttérprogramban.

4.1. ábra - Csomagéletciklus



3. Csomagátírás

Az OTRS minden új hibajavító vagy fő verziójával át kell írnia a csomagjait, és meg kell győződnie arról, hogy azok továbbra is működnek az OTRS API-val.

3.1. From OTRS 6 to 7

This section lists changes that you need to examine when porting your package from OTRS 6 to 7.

3.1.1. Sessions always require cookies

Starting from OTRS 7, sessions always require cookies to be enabled. Therefore, the `SessionIDCookie` value was removed from `LayoutObject`, Templates and JavaScript. It is no longer necessary to append session variables to URLs or HTTP request payloads.

3.1.2. The `groups` table was renamed

Due to a change in MySQL 8, the table `groups` had to be renamed to `groups_table`. If you use this table directly in any SQL statements, they will need to be adapted. For more information, see bug#13866.

3.1.3. New external interface

The existing customer (`customer.pl`) and public (`public.pl`) interfaces were replaced by a new REST backend (`Kernel/WebApp`) and a modern `Vue.js` based frontend application. This means that all related code has to be ported and/or rewritten.

There is one special case for "public" frontend modules that don't serve an HTML application. These can be ported rather easily to the new REST backend (see also the REST API docs). See for example `Kernel/WebApp/Controller/API/Public/Package/Repository.pm`. This example also shows how endpoints can support both new REST-like URLs but at the same time the legacy URLs based on the `/otrs/customer.pl?Action=MyAction` routes at the same time.

For some important URLs in the customer interface that are linked from legacy systems, redirect controllers may need to be provided to make sure the old URLs keep working.

3.1.4. Changes in Process Management

The migration script `scripts/DBUpdate-to-7.pl` will upgrade all processes that are already in the database. Manual action is only needed to make use of any new features that OTRS 7 provides.

3.1.4.1. New Activity Types

Since OTRS 7 is capable of managing more activity types, all existing activities now become 'User Task' activities. To update a task definition on a YAML file, please add `Type: UserTask` with the same indentation as `Name` or `ID`.

3.1.4.2. Renamed Process Management component `Transition` to `SequenceFlow`

This process component was renamed to be more aligned with BPMN. Files, classes and methods has been renamed accordingly. Customized files needs to be updated following the new conventions.

It is recommended to update any shipped process definitions to use the new name schema by replacing `Transition` with `SequenceFlow` and `Transitions` with `SequenceFlows`.

3.1.4.3. Renamed Process Management component `TransitionAction` to `SequenceFlowAction`

This process component does not exists in BPMN but has to be also renamed to be consistent with the other changes. Files, classes and methods has been renames accordingly. Customized files needs to be updated following the new conventions.

It is recommended to update any shipped process definitions to use the new name schema by replacing `TransitionAction` with `SequenceFlowAction` and `TransitionActions` with `SequenceFlowActionss`.

3.1.4.4. Renamed Process Management component `TransitionActionModules` to `ProcessManagementModules`

This process components are not only used by Sequence Flow Actions but also for 'Script Tasks' activities and has been moved from `Kernel/System/ProcessManagement/TransitionAction` to `Kernel/System/ProcessManagement/Modules`.

The new process management modules can offer more field types and options to present the parameters to the process designer, please follow the instructions in the Process Management Modules documentation to learn more about this new feature and how to improve existing modules.

It is needed to update any shipped process definitions to remove `Kernel::System::ProcessManagement::TransitionAction` from the `Module:` on all SequenceFlowAction, so for example: `Module: Kernel::System::ProcessManagement::TransitionAction::TicketLockSet` should become `Module: TicketLockSet`.

3.1.5. Changes in the `LayoutObject`

There are changes in `Kernel/Output/HTML/Layout.pm` which are necessary to properly render content using Mojolicious real-time web framework.

3.1.5.1. Not shown/empty tables in screens

Please make sure to check every screen which produces table-like output (e.g. `Kernel/Modules/AgentTicketStatusView.pm`). If the list of e.g. tickets is empty or even not

shown at all, check if the parameter `Output => 1` is used in creating the output for the page.

3.1.5.2. Encoding issues in legacy frontend modules

If you are getting into trouble with broken characters like umlauts, it could be that the content which is meant to be shown is rendered by the `Print()` method. To fix this, please switch the code from using the `Print()` method to the normal way of returning the complete response from the frontend module.

3.2. From OTRS 5 to 6

This section lists changes that you need to examine when porting your package from OTRS 5 to 6.

3.2.1. Date and time calculation

In OTRS 6, a new module for date and time calculation was added: `Kernel::System::DateTime`. The module `Kernel::System::Time` is now deprecated and should not be used for new code anymore.

The main advantage of the new `Kernel::System::DateTime` module is the support for real time zones like `Europe/Berlin` instead of time offsets in hours like +2. Note that also the old `Kernel::System::Time` module has been improved to support time zones. Time offsets have been completely dropped. This means that any code that uses time offsets for calculations has to be ported to use the new `DateTime` module instead. Code that doesn't fiddle around with time offsets itself can be left untouched in most cases. You just have to make sure that upon creation of a `Kernel::System::Time` object a valid time zone will be given.

Here's an example for porting time offset code to time zones:

```
my $TimeObject     = $Kernel::OM->Get('Kernel::System::Time'); # Assume a time offset of 0
 for this time object
my $SystemTime     = $TimeObject->TimeStamp2SystemTime( String => '2004-08-14 22:45:00' );
my $UserTimeZone   = '+2'; # normally retrieved via config or param
my $UserSystemTime = $SystemTime + $UserTimeZone * 3600;
my $UserTimeStamp  = $TimeObject->SystemTime2TimeStamp( SystemTime => $UserSystemTime );
```

Code using the new `Kernel::System::DateTime` module:

```
my $DateTimeObject = $Kernel::OM->Create('Kernel::System::DateTime'); # This implicitly sets
 the configured OTRS time zone
my $UserTimeZone   = 'Europe/Berlin'; # normally retrieved via config or param
$DateTimeObject->ToTimeZone( TimeZone => $UserTimeZone );
my $SystemTime    = $DateTimeObject->ToEpoch(); # note that the epoch is independent from
 the time zone, it's always calculated for UTC
my $UserTimeStamp = $DateTimeObject->ToString();
```

Please note that the returned time values with the new `Get()` function in the `Kernel::System::DateTime` module are without leading zero instead of the old `SystemTime2Date()` function in the `Kernel::System::Time` module. In the new `Kernel::System::DateTime` module the function `Format()` returns the date/time as string formatted according to the given format.

3.2.2. Adding the drag & drop multiupload

For OTRS 6, a multi attachment upload functionality was added. To implement the multi attachment upload in other extensions it is necessary to remove the attachment part

from the template file, also the JSOnDocumentComplete parts (AttachmentDelete and AttachmentUpload). Please keep in mind, in some cases the JavaScript parts are already outsourced in Core.Agent.XXX files.

Megjegyzés

Please note that this is currently only applicable for places where it actually makes sense to have the possibility to upload multiple files (like AgentTicketPhone, AgentTicketCompose, etc.). This is not usable out of the box for admin screens.

To include the new multi attachment upload in the template, replace the existing input type="file" with the following code in your .tt template file:

```
<label>[% Translate("Attachments") | html %]:</label>
<div class="Field">
[% INCLUDE "FormElements/AttachmentList.tt" %]
</div>
<div class="Clear"></div>
```

It is also necessary to remove the IsUpload variable and all other IsUpload parts from the Perl module. Code parts like following are not needed anymore:

```
my $IsUpload = ( $ParamObject->GetParam( Param => 'AttachmentUpload' ) ? 1 : 0 );
```

Additional to that, the Attachment Layout Block needs to be replaced:

```
$LayoutObject->Block(
    Name => 'Attachment',
    Data => $Attachment,
);
```

Replace it with this code:

```
push @{ $Param{AttachmentList} }, $Attachment;
```

If the module where you want to integrate multi upload supports standard templates, make sure to add a section to have a human readable file size format right after the attachments of the selected template have been loaded (see e.g. AgentTicketPhone for reference):

```
for my $Attachment (@TicketAttachments) {
    $Attachment->{Filesize} = $LayoutObject->HumanReadableDataSize(
        Size => $Attachment->{Filesize},
    );
}
```

When adding selenium unit tests for the modules you ported, please take a look at Selenium/Agent/MultiAttachmentUpload.t for reference.

3.2.3. Improvements to administration screens

3.2.3.1. Add breadcrumbs to administration screens

In OTRS 6, all admin modules should have a breadcrumb. The breadcrumb only needs to be added on the .tt template file and should be placed right after the h1 headline on top

of the file. Additionally, the headline should receive the class `InvisibleText` to make it only *visible* for screen readers.

```
<div class="MainBox ARIARoleMain LayoutFixedSidebar SidebarFirst">
    <h1 class="InvisibleText">[% Translate("Name of your module") | html %]</h1>
[% BreadcrumbPath = [
        {
            Name => Translate('Name of your module'),
        },
    ]
%]
[% INCLUDE "Breadcrumb.tt" Path = BreadcrumbPath %]
...
```

Please make sure to add the correct breadcrumb for all levels of your admin module (e.g. `Subactions`):

```
[% BreadcrumbPath = [
        {
            Name => Translate('Module Home Screen'),
            Link => Env("Action"),
        },
        {
            Name => Translate("Some Subaction"),
        },
    ]
%]

[% INCLUDE "Breadcrumb.tt" Path = BreadcrumbPath %]
```

3.2.3.2. Add *Save* and *Save and finish* buttons to administration screens

Admin modules in OTRS 6 should not only have a *Save* button, but also a *Save and finish* button. *Save* should leave the user on the same edit page after saving, *Save and finish* should lead back to the overview of the entity the user is currently working on. Please see existing OTRS admin screens for reference.

```
<div class="Field SpacingTop SaveButtons">
    <button class="Primary CallForAction" id="SubmitAndContinue" type="submit" value="[%
 Translate("Save") | html %]"><span>[% Translate("Save") | html %]</span></button>
    [% Translate("or") | html %]
    <button class="Primary CallForAction" id="Submit" type="submit" value="[%
 Translate("Save") | html %]"><span>[% Translate("Save and finish") | html %]</span></
button>
    [% Translate("or") | html %]
    <a href="[% Env("Baselink") %]Action=[% Env("Action") %]"><span>[% Translate("Cancel") |
 html %]</span></a>
</div>
```

3.2.4. Migrate configuration files

3.2.4.1. XML configuration file format

OTRS 6 uses a new XML configuration file format and the location of configuration files moved from `Kernel/Config/Files` to `Kernel/Config/Files/XML`. To convert existing XML configuration files to the new format and location, you can use the following tool that is part of the OTRS framework:

```
```

```
bin/otrs.Console.pl Dev::Tools::Migrate::ConfigXMLStructure --source-directory Kernel/
Config/Files
Migrating configuration XML files...
Kernel/Config/Files/Calendar.xml -> Kernel/Config/Files/XML/Calendar.xml... Done.
Kernel/Config/Files/CloudServices.xml -> Kernel/Config/Files/XML/CloudServices.xml... Done.
Kernel/Config/Files/Daemon.xml -> Kernel/Config/Files/XML/Daemon.xml... Done.
Kernel/Config/Files/Framework.xml -> Kernel/Config/Files/XML/Framework.xml... Done.
Kernel/Config/Files/GenericInterface.xml -> Kernel/Config/Files/XML/GenericInterface.xml...
 Done.
Kernel/Config/Files/ProcessManagement.xml -> Kernel/Config/Files/XML/
ProcessManagement.xml... Done.
Kernel/Config/Files/Ticket.xml -> Kernel/Config/Files/XML/Ticket.xml... Done.

Done.
```

3.2.4.2. Perl configuration file format

OTRS 6 speeds up configuration file loading by dropping support for the old configuration format (1) that just used sequential Perl code and had to be run by `eval` and instead enforcing the new package-based format (1.1) for Perl configuration files. OTRS 6+ can only load files with this format, please make sure to convert any custom developments to it (see `Kernel/Config/Files/ZZZ*.pm` for examples). Every Perl configuration file needs to contain a package with a `Load()` method.

In the past, Perl configuration files were sometimes misused as an autoload mechanism to override code in existing packages. This is not necessary any more as OTRS 6 features a dedicated `Autoload` mechanism. Please see `Kernel/Autoload/Test.pm` for a demonstration on how to use this mechanism to add a method in an existing file.

3.2.5. Perldoc structure changed

The structure of POD in Perl files was slightly improved and should be adapted in all files. POD is now also enforced to be syntactically correct.

What was previously called `SYNOPSIS` is now changed to `DESCRIPTION`, as a synopsis typically provides a few popular code usage examples and not a description of the module itself. An additional synopsis can be provided, of course. Here's how an example:

```
=head1 NAME

Kernel::System::ObjectManager - Central singleton manager and object instance generator

=head1 SYNOPSIS

# In toplevel scripts only!
local $Kernel::OM = Kernel::System::ObjectManager->new();

# Everywhere: get a singleton instance (and create it, if needed).
my $ConfigObject = $Kernel::OM->Get('Kernel::Config');

# Remove singleton objects and all their dependencies.
$Kernel::OM->ObjectsDiscard(
    Objects            => ['Kernel::System::Ticket', 'Kernel::System::Queue'],
);

=head1 DESCRIPTION

The ObjectManager is the central place to create and access singleton OTRS objects (via
 C<L</Get()>>)
as well as create regular (unmanaged) object instances (via C<L</Create()>>).
```

In case the `DESCRIPTION` does not add any value to the line in the `NAME` section, it should be rewritten or removed altogether.

The second important change is that functions are now documented as =head2 instead of the previously used =item.

```
=head2 Get()

Retrieves a singleton object, and if it not yet exists, implicitly creates one for you.

my $ConfigObject = $Kernel::OM->Get('Kernel::Config');

# On the second call, this returns the same ConfigObject as above.
my $ConfigObject2 = $Kernel::OM->Get('Kernel::Config');

=cut

sub Get { ... }
```

These changes lead to an improved online API documentation as can be seen in the ObjectManager documentation for OTRS 5 and OTRS 6.

3.2.6. Improvements to templating and working with JavaScript

3.2.6.1. JavaScript removed from templates

With OTRS 6, all JavaScript - especially located in JSOnDocumentComplete blocks - is removed from template files and moved to JavaScript files instead. Only in very rare conditions JavaScript needs to be placed within template files. For all other occurrences, place the JS code in module-specific JavaScript files. An Init() method within such a JavaScript file is executed automatically on file load (for the initialization of event bindings etc.) if you register the JavaScript file at the OTRS application. This is done by executing Core.Init.RegisterNamespace(TargetNS, 'APP_MODULE'); at the end of the namespace declaration within the JavaScript file.

3.2.6.2. Template files for rich text editor removed

Along with the refactoring of the JavaScript within template files (see above), the template files for the rich text editor (RichTextEditor.tt and CustomerRichTextEditor.tt) were removed as they are no longer necessary.

Typically, these template files were included in the module-specific template files within a block:

```
[% RenderBlockStart("RichText") %]
[% InsertTemplate("RichTextEditor.tt") %]
[% RenderBlockEnd("RichText") %]
```

This is no longer needed and can be removed. Instead of calling this block from the Perl module, it is now necessary to set the needed rich text parameters there. Instead of:

```
$LayoutObject->Block(
    Name => 'RichText',
    Data => \%Param,
);
```

you now have to call:

```
$LayoutObject->SetRichTextParameters(
```

```
    Data => \%Param,
);
```

Same rule applies for customer interface. Remove RichText blocks from `CustomerRichTextEditor.tt` and apply following code instead:

```
$LayoutObject->CustomerSetRichTextParameters(
    Data => \%Param,
);
```

3.2.6.3. Translations in JavaScript files

Adding translatable strings in JavaScript was quite difficult in OTRS. The string had to be translated in Perl or in the template and then sent to the JavaScript function. With OTRS 6, translation of strings is possible directly in the JavaScript file. All other workarounds, especially blocks in the templates only for translating strings, should be removed.

Instead, the new JavaScript translation namespace `Core.Language` should be used to translate strings directly in the JS file:

```
Core.Language.Translate('The string to translate');
```

It is also possible to handover JS variables to be replaced in the string directly:

```
Core.Language.Translate('The %s to %s', 'string', 'translate');
```

Every `%s` is replaced by the variable given as extra parameter. The number of parameters is not limited.

3.2.6.4. Handover data from Perl to JavaScript

To achieve template files without JavaScript code, some other workarounds had to be replaced with an appropriate solution. Besides translations, also the handover of data from Perl to JavaScript has been a problem in OTRS. The workaround was to add a JavaScript block in the template in which JavaScript variables were declared and filled with template tags based on data handed over from Perl to the template.

The handover process of data from Perl to JavaScript is now much easier in OTRS 6. To send specific data as variable from Perl to JavaScript, one only has to call a function on Perl-side. The data is than automatically available in JavaScript.

In Perl you only have to call:

```
$Self->{LayoutObject}->AddJSData(
    Key   => 'KeyToBeAvailableInJS',
    Value => $YourData,
);
```

The `Value` parameter is automatically converted to a JSON object and can also contain complex data.

In JavaScript you can get the data with:

```
Core.Config.Get('KeyToBeAvailableInJS');
```

This replaces all workarounds which need to be removed when porting a module to OTRS 6, because JavaScript in template files is now only allowed in very rare conditions (see above).

3.2.6.5. HTML templates for JavaScript

OTRS 6 exposes new JavaScript template API via `Core.Template` class. You can use it in your JavaScript code in a similar way as you use `TemplateToolkit` from Perl code.

Here's an example for porting existing `jQuery` based code to new template API:

```
var DivID = 'MyDiv',
    DivText = 'Hello, world!';

$('<div />').addClass('CSSClass')
    .attr('id', DivID)
    .text(DivText)
    .appendTo('body');
```

First, make sure to create a new template file under `Kernel/Output/JavaScript/Templates/Standard` folder. In doing this, you should keep following in mind:

- Create a subfolder with name of your `Module`.
- You may reuse any existing subfolder structure but only if it makes sense for your component (e.g. `Agent/MyModule/` or `Agent/Admin/MyModule/`).
- Use `.html.tmpl` as extension for template file.
- Name templates succinctly and clearly in order to avoid confusion (i.e. good: `Agent/MyModule/SettingsDialog.html.tmpl`, bad: `Agent/SettingsDialogTemplate.html.tmpl`).

Then, add your HTML to the template file, making sure to use placeholders for any variables you might need:

```
<div id="{{ DivID }}" class="CSSClass">
    {{ DivText | Translate }}
</div>
```

Then, just get rendered HTML by calling `Core.Template.Render` method with template path (without extension) and object containing variables for replacement:

```
var DivHTML = Core.Template.Render('Agent/MyModule/SettingsDialog', {
    DivID: 'MyDiv',
    DivText: 'Hello, world!'
});

$(DivHTML).appendTo('body');
```

Internally, `Core.Template` uses Nunjucks engine for parsing templates. Essentially, any valid Nunjucks syntax is supported, please see their documentation for more information.

Here are some tips:

- You can use `| Translate` filter for string translation to current language.

- All {{ VarName }} variable outputs are HTML escaped by default. If you need to output some existing HTML, please use | safe filter to bypass escaping.
- Use | urlencode for encoding URL parameters.
- Complex structures in replacement object are supported, so feel free to pass arrays or hashes and iterate over them right from template. For example, look at {% for %} syntax in Nunjucks documentation.

3.2.7. Checking user permissions

Before OTRS 6, user permissions were stored in the session and passed to the LayoutObject as attributes, which were then in turn accessed to determine user permissions like if ($LayoutObject->{'UserIsGroup[admin]'}) { ... }.

With OTRS 6, permissions are no longer stored in the session and also not passed to the LayoutObject. Please switch your code to calling PermissionCheck() on Kernel::System::Group (for agents) or Kernel::System::CustomerGroup (for customers). Here's an example:

```
my $HasPermission = $Kernel::OM->Get('Kernel::System::Group')->PermissionCheck(
UserID    => $UserID,
GroupName => $GroupName,
Type      => 'move_into',
);
```

3.2.8. Ticket API changes

3.2.8.1. TicketGet()

For OTRS 6, all extensions need to be checked and ported from $Ticket{SolutionTime} to $Ticket{Closed} if TicketGet() is called with the Extended parameter (see bug#11872).

Additionally, the database column ticket.create_time_unix was removed, and likewise the value CreateTimeUnix from the TicketGet() result data. Please use the value Created (database column ticket.create_time) instead.

3.2.8.2. LinkObject Events

In OTRS 6, old ticket-specific LinkObject events have been dropped:

- TicketSlaveLinkAdd
- TicketSlaveLinkDelete
- TicketMasterLinkDelete

Any event handlers listening on these events should be ported to two new events instead:

- LinkObjectLinkAdd
- LinkObjectLinkDelete

These new events will be triggered any time a link is added or deleted by LinkObject, regardless of the object type. Data parameter will contain all information your event handlers might need for further processing, e.g.:

`SourceObject`

Name of the link source object (e.g. `Ticket`).

`SourceKey`

Key of the link source object (e.g. `TicketID`).

`TargetObject`

Name of the link target object (e.g. `FAQItem`).

`TargetKey`

Key of the link target object (e.g. `FAQItemID`).

`Type`

Type of the link (e.g. `ParentChild`).

`State`

State of the link (`Valid` or `Temporary`).

With these new events in place, any events specific for custom `LinkObject` module implementations can be dropped, and all event handlers ported to use them instead. Since source and target object names are provided in the event itself, it would be trivial to make them run only in specific situations.

To register your event handler for these new events, make sure to add a registration in the configuration, for example:

```
<!-- OLD STYLE -->
<ConfigItem Name="LinkObject::EventModulePost###1000-SampleModule" Required="0" Valid="1">
    <Description Translatable="1">Event handler for sample link object module.</Description>
    <Group>Framework</Group>
    <SubGroup>Core::Event::Package</SubGroup>
    <Setting>
        <Hash>
            <Item Key="Module">Kernel::System::LinkObject::Event::SampleModule</Item>
            <Item Key="Event">(LinkObjectLinkAdd|LinkObjectLinkDelete)</Item>
            <Item Key="Transaction">1</Item>
        </Hash>
    </Setting>
</ConfigItem>

<!-- NEW STYLE -->
<Setting Name="LinkObject::EventModulePost###1000-SampleModule" Required="0" Valid="1">
    <Description Translatable="1">Event handler for sample link object module.</Description>
    <Navigation>Core::Event::Package</Navigation>
    <Value>
        <Hash>
            <Item Key="Module">Kernel::System::LinkObject::Event::SampleModule</Item>
            <Item Key="Event">(LinkObjectLinkAdd|LinkObjectLinkDelete)</Item>
            <Item Key="Transaction">1</Item>
        </Hash>
    </Value>
</Setting>
```

3.2.9. Article API changes

In OTRS 6, changes to Article API have been made, in preparations for new *Omni Channel* infrastructure.

3.2.9.1. Meta Article API

Article object now provides top-level article functions that do not involve back-end related data.

Following methods related to articles have been moved to `Kernel::System::Ticket::Article` object:

- `ArticleFlagSet()`
- `ArticleFlagDelete()`
- `ArticleFlagGet()`
- `ArticleFlagsOfTicketGet()`
- `ArticleAccountedTimeGet()`
- `ArticleAccountedTimeDelete()`
- `ArticleSenderTypeList()`
- `ArticleSenderTypeLookup()`
- `SearchStringStopWordsFind()`
- `SearchStringStopWordsUsageWarningActive()`

If you are referencing any of these methods via `Kernel::System::Ticket` object in your code, please switch to Article object and use it instead. For example:

```
    my $ArticleObject = $Kernel::OM->Get('Kernel::System::Ticket::Article');

    my %ArticleSenderTypeList = $ArticleObject->ArticleSenderTypeList();
```

New `ArticleList()` method is now provided by the article object, and can be used for article listing and locating. This method implements filters and article numbering and returns article meta data only as an ordered list. For example:

```
my @Articles = $ArticleObject->ArticleList(
    TicketID             => 123,
    CommunicationChannel => 'Email',           # optional, to limit to a certain
 CommunicationChannel
    SenderType           => 'customer',        # optional, to limit to a certain article
 SenderType
    IsVisibleForCustomer => 1,                 # optional, to limit to a certain visibility
    OnlyFirst            => 1,                 # optional, only return first match, or
    OnlyLast             => 1,                 # optional, only return last match
);
```

Following methods related to articles have been dropped all-together. If you are using any of them in your code, please evaluate possibility of alternatives.

- `ArticleFirstArticle()` (use `ArticleList( OnlyFirst => 1)` instead)
- `ArticleLastCustomerArticle()` (use `ArticleList( SenderType => 'customer', OnlyLast => 1)` or similar)
- `ArticleCount()` (use `ArticleList()` instead)

- `ArticlePage()` (reimplemented in `AgentTicketZoom`)
- `ArticleTypeList()`
- `ArticleTypeLookup()`
- `ArticleIndex()` (use `ArticleList()` instead)
- `ArticleContentIndex()`

To work with article data please use new article backend API. To get correct backend object for an article, please use:

- `BackendForArticle(%Article)`
- `BackendForChannel( ChannelName => $ChannelName )`

`BackendForArticle()` returns the correct back end for a given article, or the invalid back end, so that you can always expect a back end object instance that can be used for chain-calling.

```
my $ArticleBackendObject = $ArticleObject->BackendForArticle( TicketID => 42, ArticleID =>
 123 );
```

`BackendForChannel()` returns the correct back end for a given communication channel.

```
my $ArticleBackendObject = $ArticleObject->BackendForChannel( ChannelName => 'Email' );
```

3.2.9.2. Article Backend API

All other article data and related methods have been moved to separate backends. Every communication channel now has a dedicated backend API that handles article data and can be used to manipulate it.

OTRS 6 Free ships with some default channels and corresponding backends:

- Email (equivalent to old `email` article types)
- Phone (equivalent to old phone article types)
- Internal (equivalent to old `note` article types)
- Chat (equivalent to old `chat` article types)

Megjegyzés

While chat article backend is available in OTRS 6 Free, it is only utilized when system has a valid **OTRS Business Solution™** installed.

Article data manipulation can be managed via following backend methods:

- `ArticleCreate()`
- `ArticleUpdate()`
- `ArticleGet()`
- `ArticleDelete()`

All of these methods have dropped article type parameter, which must be substituted for `SenderType` and `IsVisibleForCustomer` parameter combination. In addition, all these methods now also require `TicketID` and `UserID` parameters.

Megjegyzés

Since changes in article API are system-wide, any code using the old API must be ported for OTRS 6. This includes any web service definitions which leverage these methods directly via GenericInterface for example. They will need to be re-assessed and adapted to provide all required parameters to the new API during requests and manage subsequent responses in new format.

Please note that returning hash of `ArticleGet()` has changed, and some things (like ticket data) might be missing. Utilize parameters like `DynamicFields => 1` and `RealNames => 1` to get more information.

In addition, attachment data is not returned any more, please use combination of following methods from the article backends:

- `ArticleAttachmentIndex()`
- `ArticleAttachment()`

Note that `ArticleAttachmentIndex()` parameters and behavior has changed. Instead of old strip parameter use combination of new `ExcludePlainText`, `ExcludeHTMLBody` and `ExcludeInline`.

As an example, here is how to get all article and attachment data in the same hash:

```
my @Articles = $ArticleObject->ArticleList(
    TicketID => $TicketID,
);

ARTICLE:
for my $Article (@Articles) {

    # Make sure to retrieve backend object for this specific article.
    my $ArticleBackendObject = $ArticleObject->BackendForArticle( %{$Article} );

    my %ArticleData = $ArticleBackendObject->ArticleGet(
        %{$Article},
        DynamicFields => 1,
        UserID        => $UserID,
    );
    $Article = \%ArticleData;

    # Get attachment index (without attachments).
    my %AtmIndex = $ArticleBackendObject->ArticleAttachmentIndex(
        ArticleID => $Article->{ArticleID},
        UserID    => $UserID,
    );
    next ARTICLE if !%AtmIndex;

    my @Attachments;
    ATTACHMENT:
    for my $FileID ( sort keys %AtmIndex ) {
        my %Attachment = $ArticleBackendObject->ArticleAttachment(
            ArticleID => $Article->{ArticleID},
            FileID    => $FileID,
            UserID    => $UserID,
        );
        next ATTACHMENT if !%Attachment;

        $Attachment{FileID} = $FileID;
        $Attachment{Content} = encode_base64( $Attachment{Content} );
```

```
        push @Attachments, \%Attachment;
    }

    # Include attachment data in article hash.
    $Article->{Atms} = \@Attachments;
}
```

3.2.9.3. Article Search Index

To make article indexing more generic, article backends now provide information necessary for properly indexing article data. Index will be created similar to old `StaticDB` mechanism and stored in a dedicated article search table.

Since now every article backend can provide search on arbitrary number of article fields, use `BackendSearchableFieldsGet()` method to get information about them. This data can also be used for forming requests to `TicketSearch()` method. Coincidentally, some `TicketSearch()` parameters have changed their name to also include article backend information, for example:

Old parameter	New parameter
`From`	`MIMEBase_From`
`To`	`MIMEBase_To`
`Cc`	`MIMEBase_Cc`
`Subject`	`MIMEBase_Subject`
`Body`	`MIMEBase_Body`
`AttachmentName`	`MIMEBase_AttachmentName`

Additionally, article search indexing will be done in an async call now, in order to off-load index calculation to a separate task. While this is fine for production systems, it might create new problems in certain situations, e.g. unit tests. If you are manually creating articles in your unit test, but expect it to be searchable immediately after created, make sure to manually call the new `ArticleSearchIndexBuild()` method on article object.

3.2.10. SysConfig API changes

Note that in OTRS 6 SysConfig API was changed, so you should check if the methods are still existing. For example, `ConfigItemUpdate()` is removed. To replace it you should use combination of the following methods:

- `SettingLock()`
- `SettingUpdate()`
- `ConfigurationDeploy()`

In case that you want to update a configuration setting during a `CodeInstall` section of a package, you could use `SettingsSet()`. It does all previously mentioned steps and it can be used for multiple settings at once.

Megjegyzés

Do not use `SettingSet()` in the SysConfig GUI itself.

```
my $Success = $SysConfigObject->SettingsSet(
    UserID   => 1,                                      # (required) UserID
```

```
    Comments => 'Deployment comment',                   # (optional) Comment
    Settings => [                                       # (required) List of settings to
 update.
        {
            Name                   => 'Setting::Name',  # (required)
            EffectiveValue         => 'Value',          # (optional)
            IsValid                => 1,                # (optional)
            UserModificationActive => 1,                # (optional)
        },
        ...
    ],
);
```

3.2.11. `LinkObject` API changes

Note that `LinkObject` was slightly modified in the OTRS 6 and methods `LinkList()` and `LinkKeyList()` might return different result if `Direction` parameter is used. Consider changing `Direction`.

Old code:

```
my $LinkList = $LinkObject->LinkList(
    Object    => 'Ticket',
    Key       => '321',
    Object2   => 'FAQ',
    State     => 'Valid',
    Type      => 'ParentChild',
    Direction => 'Target',
    UserID    => 1,
);
```

New code:

```
my $LinkList = $LinkObject->LinkList(
    Object    => 'Ticket',
    Key       => '321',
    Object2   => 'FAQ',
    State     => 'Valid',
    Type      => 'ParentChild',
    Direction => 'Source',
    UserID    => 1,
);
```

3.2.12. Communication Log support for additional PostMaster Filters

As part of email handling improvements for OTRS 6, a new logging mechanism was added to OTRS 6, exclusively used for incoming and outgoing communications. All PostMaster filters were enriched with this new Communication Log API, which means any additional filters coming with packages should also leverage the new log feature.

If your package implements additional PostMaster filters, make sure to get acquainted with API usage instructions. Also, you can get an example of how to implement this logging mechanism by looking the code in the `Kernel::System::PostMaster::NewTicket`.

3.2.13. Process MailQueue for unit tests

As part of email handling improvements for OTRS 6, all emails are now sent asynchronously, that means they are saved in a queue for future processing.

To the unit tests that depend on emails continue to work properly is necessary to force the processing of the email queue.

Make sure to start with a clean queue:

```
                my $MailQueueObject = $Kernel::OM->Get('Kernel::System::MailQueue');
                $MailQueueObject->Delete();
```

If for some reason you can't clean completely the queue, e.g. selenium unit tests, just delete the items created during the tests:

```
                my $MailQueueObject = $Kernel::OM->Get('Kernel::System::MailQueue');
                my %MailQueueCurrentItems = map { $_->{ID} => $_ } @{ $MailQueueObject-
>List() || [] };

                my $Items = $MailQueueObject->List();
                MAIL_QUEUE_ITEM:
                for my $Item ( @{$Items} ) {
                    next MAIL_QUEUE_ITEM if $MailQueueCurrentItems{ $Item->{ID} };
                    $MailQueueObject->Delete(
                        ID => $Item->{ID},
                    );
                }
```

Process the queue after the code that you expect to send emails:

```
                my $MailQueueObject = $Kernel::OM->Get('Kernel::System::MailQueue');
                my $QueueItems      = $MailQueueObject->List();
                for my $Item ( @{$QueueItems} ) {
                    $MailQueueObject->Send( %{$Item} );
                }
```

Or process only the ones created during the tests:

```
                my $MailQueueObject = $Kernel::OM->Get('Kernel::System::MailQueue');
                my $QueueItems      = $MailQueueObject->List();
                MAIL_QUEUE_ITEM:
                for my $Item ( @{$QueueItems} ) {
                    next MAIL_QUEUE_ITEM if $MailQueueCurrentItems{ $Item->{ID} };
                    $MailQueueObject->Send( %{$Item} );
                }
```

Depending on your case, you may need to clean the queue after or before processing it.

3.2.14. Widget Handling in Ticket Zoom Screen

The widgets in the ticket zoom screen have been improved to work in a more generic way. With OTRS 6, it is now possible to add new widgets for the ticket zoom screen via the SysConfig. It is possible to configure the used module, the location of the widget (e.g. Sidebar) and if the content should be loaded synchronously (default) or via AJAX.

Here is an example configuration for the default widgets:

```
<Setting Name="Ticket::Frontend::AgentTicketZoom###Widgets###0100-TicketInformation"
 Required="0" Valid="1">
    <Description Translatable="1">AgentTicketZoom widget that displays ticket data in the
 side bar.</Description>
    <Navigation>Frontend::Agent::View::TicketZoom</Navigation>
```

```
    <Value>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::TicketZoom::TicketInformation</Item>
            <Item Key="Location">Sidebar</Item>
        </Hash>
    </Value>
</Setting>
<Setting Name="Ticket::Frontend::AgentTicketZoom###Widgets###0200-CustomerInformation"
 Required="0" Valid="1">
    <Description Translatable="1">AgentTicketZoom widget that displays customer information
 for the ticket in the side bar.</Description>
    <Navigation>Frontend::Agent::View::TicketZoom</Navigation>
    <Value>
        <Hash>
            <Item Key="Module">Kernel::Output::HTML::TicketZoom::CustomerInformation</Item>
            <Item Key="Location">Sidebar</Item>
            <Item Key="Async">1</Item>
        </Hash>
    </Value>
</Setting>
```

Megjegyzés

With this change, the template blocks in the widget code have been removed, so you should check if you use the old widget blocks in some output filters via `Frontend::Template::GenerateBlockHooks` functionality, and implement it in the new fashion.

3.3. OTRS 4-ről 5-re

Ez a szakasz azokat a változtatásokat sorolja fel, amelyeket meg kell vizsgálnia, amikor átírja a csomagját az OTRS 4-ről 5-re.

3.3.1. Átszerkesztett `Kernel/Output/HTML`

In OTRS 5, `Kernel/Output/HTML` was restructured. All Perl modules (except `Layout.pm`) were moved to subdirectories (one for every module layer). Template (theme) files were also moved from `Kernel/Output/HTML/Standard` to `Kernel/Output/HTML/Templates/Standard`. Please perform this migration also in your code.

3.3.2. Elő-kimenetszűrők

Az OTRS 5-tel többé nincs támogatás a `pre` kimenetszűrőkhöz. Ezek a szűrők azelőtt változtatták meg a sablon tartalmát, mielőtt az feldolgozásra került volna, és potenciálisan rossz teljesítményproblémákhoz vezethettek, ugyanis a sablonokat többé nem lehetett gyorstárazni, és minden alkalommal fel kellett dolgozni és le kellett fordítani.

Egyszerűen váltson a `pre` kimenetszűrőről a `post` kimenetszűrőre. A tartalom lefordításához futtathatja közvetlenül a `$LayoutObject->Translate()` függvényt. Ha egyéb sablonszolgáltatásokra van szüksége, akkor egyszerűen határozzon meg egy kis sablonfájlt a kimenetszűrőhöz, és használja azt a tartalom megjelenítéséhez, mielőtt beültetné azt a fő adatokba. Néhány esetben hasznos lehet a jQuery DOM műveletek használata is a képernyőn lévő tartalom sorrendjének megváltoztatásához vagy cseréjéhez a reguláris kifejezések használata helyett. Ebben az esetben láthatatlan tartalomként kellene beültetnie az új kódot valahova az oldalba (például a `Hidden` osztállyal), majd ezután áthelyezni a jQuery használatával a megfelelő helyre a DOM-ban, és megjeleníteni azt.

Az utó-kimenetszűrők használatának megkönnyítéséhez létezik egy új mechanizmus is a HTML megjegyzéshorgok lekéréséhez bizonyos sablonoknál vagy blokkoknál. Hozzáadhatja a modulbeállító XML-be a következőhöz hasonlóan:

```
<ConfigItem
Name="Frontend::Template::GenerateBlockHooks###100-OTRSBusiness-ContactWithData"
Required="1" Valid="1">
    <Description Translatable="1">Generate HTML comment hooks for
the specified blocks so that filters can use them.</Description>
    <Group>OTRSBusiness</Group>
    <SubGroup>Core</SubGroup>
    <Setting>
        <Hash>
            <Item Key="AgentTicketZoom">
                <Array>
                    <Item>CustomerTable</Item>
                </Array>
            </Item>
        </Hash>
    </Setting>
</ConfigItem>
```

Ez azt fogja okozni, hogy az `AgentTicketZoom.tt` fájlban lévő `CustomerTable` blokk át lesz alakítva a HTML megjegyzésekben minden alkalommal, amikor megjelenítésre kerül:

```
<!--HookStartCustomerTable-->
... blokk kimenet ...
<!--HookEndCustomerTable-->
```

Ezzel a mechanizmussal minden csomag csak azokat a blokkhorgokat kérheti, amelyekre szüksége van, és következetesen kerülnek megjelenítésre. Ezek a HTML megjegyzések használhatók ezután a kimenetszűrőben az egyszerű reguláris kifejezés illesztéshez.

3.3.3. IE 8 és IE 9

Az IE 8 és IE 9 támogatást eldobták. Eltávolíthat minden kerülőmegoldást a kódjából, amelyet ezekhez a platformokhoz készített, valamint az összes olyan régi <CSS_IE7> vagy <CSS_IE8> betöltő címkét, amely még esetleg megbújik az XML beállítófájljaiban.

3.3.4. Általános felület API változás a „Ticket” csatlakozóban

A `TicketGet()` művelet másképpen ad vissza dinamikus mező adatokat a jegyből és a bejegyzésből mint az OTRS 4-ben. Mostantól ezek tisztán el vannak választva a többi statikus jegy- és bejegyzésmezőktől - innentől kezdve csoportosítva vannak egy `DynamicField` nevű listába. Eszerint alakítson át minden olyan alkalmazást, amely ezt a műveletet használja.

```
# changed from:

Ticket => [
{
    TicketNumber       => '20101027000001',
    Title              => 'some title',
    ...
    DynamicField_X     => 'value_x',
},
]

# to:

Ticket => [
{
    TicketNumber       => '20101027000001',
    Title              => 'some title',
```

```
    ...
    DynamicField => [
        {
            Name  => 'some name',
            Value => 'some value',
        },
    ],
},
]
```

3.3.5. Előnézeti függvények a dinamikus statisztikákban

The new statistics GUI provides a preview for the current configuration. This must be implemented in the statistic modules and usually returns fake / random data for speed reasons. So for any dynamic (matrix) statistic that provides the method `GetStatElement()` you should also add a method `GetStatElementPreview()`, and for every dynamic (table) statistic that provides `GetStatTable()` you should accordingly add `GetStatTablePreview()`. Otherwise the preview in the new statistics GUI will not work for your statistics. You can find example implementations in the default OTRS statistics.

3.3.6. Eldobott HTML nyomtatás

Until OTRS 5, the Perl module `PDF::API2` was not present on all systems. Therefore a fallback HTML print mode existed. With OTRS 5, the module is now bundled and HTML print was dropped. `$LayoutObject->PrintHeader()` and `PrintFooter()` are not available any more. Please remove the HTML print fallback from your code and change it to generate PDF if necessary.

3.3.7. Továbbfejlesztett fordítási szöveg kinyerés

Until OTRS 5, translatable strings could not be extracted from Perl code and Database XML definitions. This is now possible and makes dummy templates like `AAA*.tt` obsolete. Please see this section for details.

3.4. OTRS 3.3-ról 4-re

Ez a szakasz azokat a változtatásokat sorolja fel, amelyeket meg kell vizsgálnia, amikor átírja a csomagját az OTRS 3.3-ról 4-re.

3.4.1. Új objektumkezelés

Up to OTRS 4, objects used to be created both centrally and also locally and then handed down to all objects by passing them to the constructors. With OTRS 4 and later versions, there is now an `ObjectManager` that centralizes singleton object creation and access.

This will require you first of all to change all top level Perl scripts (.pl files only!) to load and provide the `ObjectManager` to all OTRS objects. Let's look at `otrs.CheckDB.pl` from OTRS 3.3 as an example:

```
use strict;
use warnings;

use File::Basename;
use FindBin qw($RealBin);
use lib dirname($RealBin);
use lib dirname($RealBin) . '/Kernel/cpan-lib';
use lib dirname($RealBin) . '/Custom';
```

```
use Kernel::Config;
use Kernel::System::Encode;
use Kernel::System::Log;
use Kernel::System::Main;
use Kernel::System::DB;

# a szokásos objektumok létrehozása
my %CommonObject = ();
$CommonObject{ConfigObject} = Kernel::Config->new();
$CommonObject{EncodeObject} = Kernel::System::Encode->new(%CommonObject);
$CommonObject{LogObject}    = Kernel::System::Log->new(
    LogPrefix    => 'OTRS-otrs.CheckDB.pl',
    ConfigObject => $CommonObject{ConfigObject},
);
$CommonObject{MainObject} = Kernel::System::Main->new(%CommonObject);
$CommonObject{DBObject}   = Kernel::System::DB->new(%CommonObject);
```

Láthatjuk, hogy rengeteg kódot használnak a csomagok betöltéséhez és a gyakori objektumok létrehozásához, amelyet a parancsfájlban át kell adni a használandó OTRS objektumoknak. Az OTRS 4-gyel ez egy kicsit máshogy néz ki:

```
use strict;
use warnings;

use File::Basename;
use FindBin qw($RealBin);
use lib dirname($RealBin);
use lib dirname($RealBin) . '/Kernel/cpan-lib';
use lib dirname($RealBin) . '/Custom';

use Kernel::System::ObjectManager;

# create common objects
local $Kernel::OM = Kernel::System::ObjectManager->new(
'Kernel::System::Log' => {
    LogPrefix => 'OTRS-otrs.CheckDB.pl',
},
);

# get database object
my $DBObject = $Kernel::OM->Get('Kernel::System::DB');
```

Az új kód egy kicsivel rövidebb mint a régi. Többé nem szükséges az összes csomag betöltése, elég csak az `ObjectManager` objektumot. Azután a `$Kernel::OM->Get('Sajat::Perl::Csomag')` használható az objektumok példányainak lekéréséhez, amelyeket csak egyszer kell létrehozni. A `LogPrefix` beállítás vezérli azokat a naplóüzeneteket, amelyeket a `Kernel::System::Log` ír ki, így az szintén elhagyható.

From this example you can also deduce the general porting guide when it comes to accessing objects: don't store them in `$Self` any more (unless needed for specific reasons). Just fetch and use the objects on demand like `$Kernel::OM->Get('Kernel::System::Log')->Log(...)`. This also has the benefit that the Log object will only be created if something must be logged. Sometimes it could also be useful to create local variables if an object is used many times in a function, like `$DBObject` in the example above.

There's not much more to know when porting packages that should be loadable by the `ObjectManager`. They should declare the modules they use (via `$Kernel::OM->Get()`) like this:

```
our @ObjectDependencies = (
'Kernel::Config',
'Kernel::System::Log',
```

```
'Kernel::System::Main',
);
```

Az `@ObjectDependencies` meghatározás szükséges az `ObjectManager` objektumhoz a helyes sorrend megtartásához az objektumok megsemmisítésekor.

Nézzük meg a `Valid.pm` fájlt az OTRS 3.3-ból és a 4-ből, hogy lássuk a különbséget. A régi:

```
package Kernel::System::Valid;

use strict;
use warnings;

use Kernel::System::CacheInternal;

...

sub new {
my ( $Type, %Param ) = @_;

# allocate new hash for object
my $Self = {};
bless( $Self, $Type );

# check needed objects
for my $Object (qw(DBObject ConfigObject LogObject EncodeObject MainObject)) {
    $Self->{$Object} = $Param{$Object} || die "Got no $Object!";
}

$Self->{CacheInternalObject} = Kernel::System::CacheInternal->new(
    %{$Self},
    Type => 'Valid',
    TTL  => 60 * 60 * 24 * 20,
);

return $Self;
}

...

sub ValidList {
my ( $Self, %Param ) = @_;

# read cache
my $CacheKey = 'ValidList';
my $Cache = $Self->{CacheInternalObject}->Get( Key => $CacheKey );
return %{$Cache} if $Cache;

# get list from database
return if !$Self->{DBObject}->Prepare( SQL => 'SELECT id, name FROM valid' );

# fetch the result
my %Data;
while ( my @Row = $Self->{DBObject}->FetchrowArray() ) {
    $Data{ $Row[0] } = $Row[1];
}

# set cache
$Self->{CacheInternalObject}->Set( Key => $CacheKey, Value => \%Data );

return %Data;
}
```

Az új:

```
package Kernel::System::Valid;
```

```
use strict;
use warnings;

our @ObjectDependencies = (
'Kernel::System::Cache',
'Kernel::System::DB',
'Kernel::System::Log',
);

...

sub new {
my ( $Type, %Param ) = @_;

# allocate new hash for object
my $Self = {};
bless( $Self, $Type );

$Self->{CacheType} = 'Valid';
$Self->{CacheTTL}  = 60 * 60 * 24 * 20;

return $Self;
}

...

sub ValidList {
my ( $Self, %Param ) = @_;

# read cache
my $CacheKey = 'ValidList';
my $Cache    = $Kernel::OM->Get('Kernel::System::Cache')->Get(
    Type => $Self->{CacheType},
    Key  => $CacheKey,
);
return %{$Cache} if $Cache;

# get database object
my $DBObject = $Kernel::OM->Get('Kernel::System::DB');

# get list from database
return if !$DBObject->Prepare( SQL => 'SELECT id, name FROM valid' );

# fetch the result
my %Data;
while ( my @Row = $DBObject->FetchrowArray() ) {
    $Data{ $Row[0] } = $Row[1];
}

# set cache
$Kernel::OM->Get('Kernel::System::Cache')->Set(
    Type  => $Self->{CacheType},
    TTL   => $Self->{CacheTTL},
    Key   => $CacheKey,
    Value => \%Data
);

return %Data;
}
```

You can see that the dependencies are declared and the objects are only fetched on demand. We'll talk about the `CacheInternalObject` in the next section.

3.4.2. Eltávolított `CacheInternalObject`

Mivel a `Kernel::System::Cache` mostantól képes a memóriában is gyorstárazni, a `Kernel::System::CacheInternal` eldobásra került. Nézze meg az előző példát, hogy a kódot hogyan kell átköltöztetni: a globális Cache objektumot kell használnia, és át kell adnia a Type beállítást a `Get()`, `Set()`, `Delete()` és `CleanUp()` függvények minden egyes

hívásához. A TTL paraméter mostantól elhagyható, és alapértelmezetten 20 nap, így csak akkor kell megadnia a `Get()` függvényben, ha eltérő TTL értékre van szüksége.

Figyelem

Különösen fontos a Type paraméter hozzáadása a `CleanUp()` függvényhez, mivel különben nem csak a jelenlegi gyorsítótártípus, hanem a teljes gyorsítótár törölve lehet.

3.4.3. Áthelyezett ütemező háttérprogram fájlok

Az ütemező háttérprogram fájljai áthelyezésre kerültek a `Kernel/Scheduler` mappából a `Kernel/System/Scheduler` mappába. Ha valamilyen egyéni feladatkezelő moduljai vannak, akkor azokat is át kell helyeznie.

3.4.4. Kódszakaszok frissítése az SOPM fájlokban

Code tags in SOPM files have to be updated. Please do not use `$Self` any more. In the past this was used to get access to OTRS objects like the `MainObject`. Please use the `ObjectManager` now. Here is an example for the old style:

```
<CodeInstall Type="post">

# define function name
my $FunctionName = 'CodeInstall';

# create the package name
my $CodeModule = 'var::packagesetup::' . $Param{Structure}->{Name}->{Content};

# load the module
if ( $Self->{MainObject}->Require($CodeModule) ) {

# create new instance
my $CodeObject = $CodeModule->new( %{$Self} );

if ($CodeObject) {

    # start method
    if ( !$CodeObject->$FunctionName(%{$Self}) ) {
        $Self->{LogObject}->Log(
            Priority => 'error',
            Message  => "Could not call method $FunctionName() on $CodeModule.pm."
        );
    }
}

# error handling
else {
    $Self->{LogObject}->Log(
        Priority => 'error',
        Message  => "Could not call method new() on $CodeModule.pm."
    );
}
}

</CodeInstall>
```

Most ezt a következővel kell helyettesíteni:

```
<CodeInstall Type="post"><![CDATA[
$Kernel::OM->Get('var::packagesetup::SajatCsomag')->CodeInstall();
]]></CodeInstall>
```

3.4.5. Új sablonmotor

With OTRS 4, the DTL template engine was replaced by Template::Toolkit. Please refer to the Templating section for details on how the new template syntax looks like.

Ezek azok a változtatások, amelyet alkalmaznia kell, amikor a meglévő DTL sablonokat az új Template::Toolkit szintaxisra alakítja át:

4.1. táblázat - Sablonváltoztatások az OTRS 3.3-ról 4-re

DTL címke	Template::Toolkit címke
$Data{"Name"}	[% Data.Name %]
$Data{"Complex-Name"}	[% Data.item("Complex-Name") %]
$QData{"Name"}	[% Data.Name | html %]
$QData{"Name", "$Length"}	[% Data.Name | truncate($Length) | html %]
$LQData{"Name"}	[% Data.Name | uri %]
$Quote{"Szöveg", "$Length"}	nem lehet közvetlenül lecserélni, lásd a lenti példákat
$Quote{"$Config{"Name"}"}	[% Config("Name") | html %]
$Quote{"$Data{"Name"}", "$Length"}	[% Data.Name | truncate($Length) | html %]
$Quote{"$Data{"Content"}","$QData{"MaxLength"}"}	[% Data.Name | truncate(Data.MaxLength) | html %]
$Quote{"$Text{"$Data{"Content"}"}","$QData{"MaxLength"}"}	[% Translate(Data.Content, Data.MaxLength) | truncate(Data.MaxLength) | html %]
$Config{"Name"}	[% Config("Name") %]
$Env{"Name"}	[% Env("Name") %]
$QEnv{"Name"}	[% Env("Name") | html %]
$Text{"Szöveg %s helykitöltőkkel", "String"}	[% Translate("Szöveg %s helykitöltőkkel", "String") | html %]
$Text{"Szöveg dinamikus %s helykitöltőkkel", "$QData{Name}"}	[% Translate("Szöveg dinamikus %s helykitöltőkkel", Data.Name) | html %]
'$JSText{"Szöveg dinamikus %s helykitöltőkkel", "$QData{Name}"}'	[% Translate("Szöveg dinamikus %s helykitöltőkkel", Data.Name) | JSON %]
"$JSText{"Szöveg dinamikus %s helykitöltőkkel", "$QData{Name}"}"	[% Translate("Szöveg dinamikus %s helykitöltőkkel", Data.Name) | JSON %]
$TimeLong{"$Data{"CreateTime"}"}	[% Data.CreateTime | Localize("TimeLong") %]
$TimeShort{"$Data{"CreateTime"}"}	[% Data.CreateTime | Localize("TimeShort") %]
$Date{"$Data{"CreateTime"}"}	[% Data.CreateTime | Localize("Date") %]
<-- dtl:block:Name -->...<-- dtl:block:Name -->	[% RenderBlockStart("Name") %]...[% RenderBlockEnd("Name") %]
<-- dtl:js_on_document_complete -->...<-- dtl:js_on_document_complete -->	[% WRAPPER JSOnDocumentComplete %]...[% END %]

DTL címke	Template::Toolkit címke
<!-- dtl:js_on_document_complete_placeholder -->	[% PROCESS JSOnDocumentCompleteInsert %]
$Include{"Copyright"}	[% InsertTemplate("Copyright") %]

Létezik egy `bin/otrs.MigrateDTLtoTT.pl` segítő parancsfájl is, amely automatikusan át fogja írni önnek a DTL-fájlokat a Template::Toolkit szintaxisra. Sikertelen lehet, ha hibák találhatók a DTL-jében, ezért először javítsa ki ezeket, és azután futtassa újra a parancsfájlt.

Van még további néhány dolog, amelyet tudomásul kell vennie a kód átírásakor az új sablonmotorra:

- Az összes nyelvi fájlnak mostantól rendelkeznie kell a `use utf8;` kikötéssel.
- A `Layout::Get()` mostantól elavult. Használja a `Layout::Translate()` függvényt helyette.
- A Perl-kódban a `$Text{""}` összes előfordulását mostantól le kell cserélni a `Layout::Translate()` hívásaival.

  Ez azért van, mert a DTL-ben nem volt különválasztás a sablon és az adatok között. Ha DTL-címkék voltak beszúrva valamilyen adat részeként, akkor a motornak továbbra is fel kellene dolgozni azokat. Ez többé nincs a Template::Toolkit esetén, mert itt a sablon és az adatok szigorú különválasztása van.

  Tipp: ha valamikor interpolálnia kell a címkéket az adatokban, akkor ehhez használhatja az `Interpolate` szűrőt (`[% Data.Name | Interpolate %]`). Ez nem ajánlott biztonsági és teljesítménybeli okok miatt!
- Hasonló okból a `dtl:js_on_document_complete` által körbezárt dinamikusan beágyazott JavaScript sem fog működni többé. Használja a `Layout::AddJSOnDocumentComplete()` függvényt ahelyett, hogy ezt sablonadatként ágyazná be.

  Erre találhat egy példát a `Kernel/System/DynamicField/Driver/BaseSelect.pm` fájlban.
- Legyen óvatos a `pre` kimenetszűrőkkel (a `Frontend::Output::FilterElementPre` objektumban beállítottakkal). Ezek továbbra is működnek, de meg fogják akadályozni a sablont, hogy gyorstárazza azokat. Ez komoly teljesítményproblémákhoz vezethet. Határozottan ne legyen egyetlen olyan `pre` kimenetszűrője sem, amely az összes sablonnal dolgozik, hanem korlátozza azokat bizonyos sablonokra a konfigurációs beállításokon keresztül.

  The `post` output filters (`Frontend::Output::FilterElementPost`) don't have such strong negative performance effects. However, they should also be used carefully, and not for all templates.

3.4.6. Új FontAwesome verzió

Az OTRS 4-gyel egy új verzióra frissítettük a FontAwesome betűkészletet is. Ennek következtében az ikonok CSS-osztályai megváltoztak. Miközben a korábbi ikonok egy `icon-{ikonnév}` szerű sémával voltak meghatározva, ezt mostantól a `fa fa-{ikonnév}` formában kell megadni.

Ezen változtatás miatt meg kell győződnie arról, hogy frissítette-e az összes olyan egyéni előtétprogram-modul regisztrációit, amelyek ikonokat használnak (például a felső

navigációs sávnál) az új séma használatához. Ez igaz az olyan sablonoknál is, ahol ikonelemeket használ, mint például `<i class="icon-{ikonnév}"></i>`.

3.4.7. Egységtesztek

Az OTRS 4-gyel az egységtesztekben a `$Self` többé nem szolgáltat olyan gyakori objektumokat, mint például a `MainObject`. Mindig a `$Kernel::OM->Get('...')` függvényt használja ezen objektumok lekéréséhez.

3.4.8. Egyéni jegy előzmény típusok

Ha bármilyen egyéni jegy előzmény típusokat használ, akkor két lépést kell elvégeznie, hogy azok helyesen legyenek megjelenítve az OTRS 4+ `AgentTicketHistory` képernyőjén.

Először regisztrálnia kell az egyéni jegy előzmény típusait a rendszerbeállításokon keresztül. Ez így nézhet ki:

```
<ConfigItem Name="Ticket::Frontend::HistoryTypes###100-MyCustomModule" Required="1"
 Valid="1">
<Description Translatable="1">Controls how to display the ticket history entries as readable
 values.</Description>
<Group>Ticket</Group>
<SubGroup>Frontend::Agent::Ticket::ViewHistory</SubGroup>
<Setting>
    <Hash>
        <Item Key="MyCustomType" Translatable="1">Added information (%s)</Item>
    </Hash>
</Setting>
</ConfigItem>
```

A második lépés az egyéni jegy előzmény típusnál biztosított angol szöveg lefordítása a nyelvi fájljaiban, ha szükséges. Ennyi!

Ha érdeklődik a részletek iránt, akkor nézze meg ezt a véglegesítést azon változtatásokkal kapcsolatos további információkról, amelyek az OTRS-ben történtek.

5. fejezet - Közreműködés az OTRS-ben

Ez a fejezet azt fogja bemutatni, hogy hogyan működhet közre az OTRS keretrendszerben azért, hogy a többi felhasználó is képes legyen hasznot húzni a munkájából.

1. Hozzájárulások küldése

Az OTRS és a további nyilvános modul forráskódja megtalálható a githubon. Innen eljuthat az összes elérhető tároló felsorolásához. Leírja a jelenleg aktív ágakat is, és hogy hova kell kerülniük a hozzájárulásoknak (stabil vagy fejlesztői ágak).

Erősen ajánlott a fejlesztői környezet fejezetben bemutatott OTRSCodePolicy OTRS kódminőség-ellenőrző használata, még mielőtt elküldené a hozzájárulásait. Ha a kódját nem érvényesíti ezzel az eszközzel, akkor valószínűleg nem fogják elfogadni.

A legegyszerűbb módja a hozzájárulások elküldésének az OTRS fejlesztőcsapata számára egy „pull request” létrehozása a githubon. Vessen egy pillantást a githubon lévő utasításokra, különösen a tároló elágaztatására és a beolvasztási kérések küldésére.

Az alap munkafolyamatnak így kellene kinéznie:

- Regisztráljon a githubon, ha még nincs fiókja.
- Ágaztassa el azt a tárolót, amelynél közre szeretne működni, és váltson át arra az ágra, amelybe a változtatásokat be kell tenni.
- Hozzon létre egy új fejlesztői ágat a javításhoz/szolgáltatáshoz/hozzájáruláshoz az aktuális ág alapján.
- A változtatások befejezése és a véglegesítésük után küldje be az ágat a githubra.
- Hozzon létre egy beolvasztási kérést. Az OTRS fejlesztőcsapata értesülni fog erről, ellenőrizni fogják a beolvasztási kérését, és beolvasszák azt, vagy valamilyen visszajelzést adnak a lehetséges továbbfejlesztésekről.

Ez esetleg bonyolultan hangzik, de miután beállította ezt a munkafolyamatot, látni fogja, hogy a hozzájárulások elvégzése hihetetlenül egyszerű.

2. Az OTRS fordítása

Az OTRS keretrendszer különböző nyelvek használatát teszi lehetővé az előtétprogramon. A fordításokat többnyire az OTRS felhasználói készítik el és tartják karban, ezért az *Ön* segítsége is szükséges.

2.1. Egy meglévő fordítás frissítése

Az OTRS 4-től kezdve az OTRS grafikus felhasználói felületének és a nyilvános kiterjesztőmoduloknak az összes fordítását a Transifex oldalán keresztül kezelik. Az OTRS projekt a https://www.transifex.com/otrs/OTRS/ címen található a Transifex oldalán.

Az OTRS grafikus felhasználói felületének, egy kiterjesztőmodulnak vagy egy kézikönyvnek a fordításában történő közreműködéshez regisztráljon egy ingyenes fordítófiókot a Transifex oldalán. Ezután csatlakozhat egy nyelvi csoporthoz, és elkezdheti

a fordítás frissítését. Nincs szükség további szoftverekre vagy fájlokra. Az OTRS fejlesztői időről időre le fogják tölteni a fordításokat az OTRS forráskód tárolóiba, így nem kell semmit sem elküldenie sehova.

2.2. Egy új előtétprogram-fordítás hozzáadása

Ha új nyelvre szeretné lefordítani az OTRS keretrendszert, akkor javasolhat egy új nyelvi fordítást a Transifex OTRS projekt oldalán. Miután azt elfogadták, azonnal elkezdheti a fordítást.

3. A dokumentáció fordítása

Az OTRS adminisztrátori kézikönyv a Transifex oldalán keresztül fordítható az OTRS fordításáról szóló szakaszban leírtak szerint. A Transifex oldalon csatlakozhat egy nyelvi csoporthoz a meglévő fordítás javításához, vagy akár javasolhat egy új nyelvet is, amelyre az adminisztrátori kézikönyv fordítása elkészülhet.

Fontos, hogy az előállított XML szerkezete sértetlen maradjon. Tehát ha az eredeti szöveg `Edit <filename>Kernel/Config.pm</filename>`, akkor a magyar fordításnak `<filename>Kernel/Config.pm</filename> szerkesztése` kell lennie, sértetlenül megtartva az XML címkéket. A forrásszövegben átalakítva lévő szokásos < és > jeleknek átalakítva kell lenniük a fordításokban is (mint például `&lt;valaki@pelda.hu&gt;`). A parancsfájlokat és a példákat általában nem kell lefordítani (vagyis ebben az esetben egyszerűen átmásolhatja a forrásszöveget a fordítási szövegmezőbe).

4. Kódolási stílus irányelvek

Az OTRS projekt következetes fejlesztésének megtartása érdekében irányelveket fektettünk le a stílusra vonatkozóan a különböző programnyelvekhez.

4.1. Perl

4.1.1. Formázás

4.1.1.1. Üres karakterek

TABULÁTOR: 4 szóközt használunk. Példa a zárójelekre:

```
if ($Feltétel) {
    Izé();
}
else {
    Bigyó();
}

while ($Feltétel == 1) {
    Izé();
}
```

4.1.1.2. A sorok hossza

A sorok általában nem lehetnek hosszabbak 120 karakternél, hacsak ez különleges okok miatt nem szükséges.

4.1.1.3. Szóközök és zárójelek

A jobb olvashatóság érdekében szóközöket használunk a kulcsszavak és a nyitó zárójelek között.

```
if ()...
for ()...
```

Ha csak egy egyedülálló változó van, akkor a zárójelek belül szóközök nélkül veszik körbe a változót.

```
if ($Feltétel) { ... }

# e helyett

if ( $Feltétel ) { ... }
```

Ha a feltétel nem csak egy egyedülálló változó, akkor szóközöket használunk a zárójelek és a feltétel között. És továbbra is szóköz van a kulcsszó (például `if`) és a nyitó zárójel között.

```
if ( $Feltétel && $ABC ) { ... }
```

Ne feledje, hogy a Perl beépített függvényeinél nem használunk zárójeleket:

```
chomp $Variable;
```

4.1.1.4. Forráskód fejléc és karakterkódolás

Csatolja hozzá a következő fejlécet minden egyes forrásfájlhoz. A forrásfájlok UTF-8 karakterkódolással vannak elmentve.

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

A végrehajtható fájloknak (`*.pl`) különleges fejlécük van.

```
#!/usr/bin/perl
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
```

```
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --
```

4.1.2. A Perl nyelv használata

4.1.2.1. Vezérlési folyamat

4.1.2.1.1. Feltételek

A feltételek meglehetősen összetettek lehetnek, és lehetnek „láncolt” feltételek is (logikai „és” vagy „vagy” operátorral összekapcsolva). Az OTRS kódolásakor tisztában kell lennie számos helyzettel.

A bevált Perl gyakorlatok azt mondják, hogy a magas precedenciájú operátorokat (&& és `||`) nem kellene keverni az alacsony precedenciájú operátorokkal (and és `or`). A zűrzavar elkerülése érdekében mindig a magas precedenciájú operátorokat használjuk.

```
if ( $Condition1 && $Condition2 ) { ... }

# e helyett

if ( $Condition and $Condition2 ) { ... }
```

Ez azt jelenti, hogy tisztában kell lennie a buktatókkal. Néha zárójeleket kell használnia, hogy világossá tegye, mit szeretne.

Ha hosszú feltételei vannak (a sor 120 karakternél hosszabb), akkor több sorra kell tördelnie azt. Továbbá a feltételek kezdete egy új sorban van (nem az `if` sorában).

```
if (
    $Feltétel1
    && $Feltétel2
    )
{ ... }

# e helyett:

if ( $Feltétel1
    && $Feltétel2
    )
{ ... }
```

Jegyezze meg azt is, hogy a jobboldali zárójel egyedül áll a sorban, valamint a baloldali kapcsos zárójel szintén új sorban van, és ugyanolyan behúzással rendelkezik mint az `if`. Az operátorok egy új sor elején vannak! A következő példák bemutatják, hogyan kell ezt csinálni...

```
if (
    $XMLHash[0]->{otrs_stats}[1]{StatType}[1]{Content}
    && $XMLHash[0]->{otrs_stats}[1]{StatType}[1]{Content} eq 'static'
    )
```

```
{ ... }

if ( $TemplateName eq 'AgentTicketCustomer' ) {
    ...
}

if (
    ( $Param{Section} eq 'Xaxis' || $Param{Section} eq 'All' )
    && $StatData{StatType} eq 'dynamic'
    )
{ ... }

if (
    $Self->{TimeObject}->TimeStamp2SystemTime( String => $Cell->{TimeStop} )
    > $Self->{TimeObject}->TimeStamp2SystemTime(
        String => $ValueSeries{$Row}{$TimeStop}
    )
    || $Self->{TimeObject}->TimeStamp2SystemTime( String => $Cell->{TimeStart} )
    < $Self->{TimeObject}->TimeStamp2SystemTime(
        String => $ValueSeries{$Row}{$TimeStart}
    )
    )
{ ... }
```

4.1.2.1.2. Hátul álló `if`

Általánosan azért használunk „hátul álló `if`” utasításokat, hogy csökkentsük a szintek számát. De ne használjuk többsoros utasításoknál, és csak akkor megengedett, amikor visszatérési utasításokat hoz magával a függvény, vagy egy ciklus befejezéséhez, illetve a következő iterációra való ugráshoz.

Ez helyes:

```
next ITEM if !$ItemId;
```

Ez hibás:

```
return $Self->{LogObject}->Log(
    Priority => 'error',
    Message  => 'ItemID szükséges!',
) if !$ItemId;
```

Ez kevésbé karbantartható ennél:

```
if( !$ItemId ) {
    $Self->{LogObject}->Log( ... );
    return;
}
```

Ez helyes:

```
for my $Needed ( 1 .. 10 ) {
    next if $Needed == 5;
    last  if $Needed == 9;
}
```

Ez hibás:

```
my $Var = 1 if $Something == 'Yes';
```

4.1.2.2. Néhány beépített Perl szubrutin használatának korlátozása

A Perl néhány beépített szubrutinját nem lehet használni semmilyen helyen:

- Ne használja a `die` és `exit` szubrutinokat a `.pm` fájlokban.
- Ne használja a `Dumper` függvényt a kiadott fájlokban.
- Ne használja a `print` utasítást a `.pm` fájlokban.
- Ne használja a `require` kulcsszót, inkább használja a `Main::Require()` metódust.
- Use the functions of the `DateTimeObject` instead of the builtin functions like `time()`, `localtime()`, etc.

4.1.2.3. Reguláris kifejezések

For regular expressions *in the source code*, we always use the `m//` operator with curly braces as delimiters. We also use the modifiers `x`, `m` and `s` by default. The `x` modifier allows you to comment your regex and use spaces to visually separate logical groups.

```
$Date =~ m{ \A \d{4} - \d{2} - \d{2} \z }xms
$Date =~ m{
    \A      # a szöveg kezdete
    \d{4} - # év
    \d{2} - # hónap
    [^\n]   # minden, kivéve az új sort
    #..
}xms;
```

Mivel a szóköznek többé nincs különleges jelentése, ezért egy egyedüli karakterosztályt kell használnia egy egyedülálló szóköz illesztéséhez (`[ ]`). Ha akármennyi szóközre szeretne illeszteni, akkor azt a `\s` használatával teheti meg.

A reguláris kifejezésben a pont (`.`) tartalmazza az új sort (minthogy az `s` módosító nélküli reguláris kifejezésben a pont azt jelenti, hogy „minden, kivéve az új sor”). Ha bármire szeretne illeszteni az új sort kivéve, akkor a tagadott egyedüli karakterosztályt kell használnia (`[^\n]`).

```
$Text =~ m{
    Teszt
    [ ]    # itt szóköznek kell lennie a „Teszt” és a „Regex” között
    Regex
}xms;
```

An exception to the convention above applies to all cases where regular expressions are not written statically in the code but instead are *supplied by users* in one form or another (for example via SysConfig or in a PostMaster filter configuration). Any evaluation of such a regular expression has to be done without any modifiers (e.g. `$Variable =~ m{$Regex}`) in order to match the expectation of (mostly inexperienced) users and also to be backwards compatible.

If modifiers are strictly necessary for user supplied regular expressions, it is always possible to use embedded modifiers (e.g. `(?:(?i)SmAlL oR lArGe)`). For details, please see perlretut.

Usage of the `r` modifier is encouraged, e.g. if you need to extract part of a string into another variable. This modifier keeps the matched variable intact and instead provides the substitution result as a return value.

Example: Use this...

```
my $NewText = $Text =~ s{
    \A
    Prefix
    (
        Text
    )
}
{NewPrefix$1Postfix}xmsr;
```

instead of this...

```
my $NewText = $Text;
$NewText =~ s{
    \A
    Prefix
    (
        Text
    )
}
{NewPrefix$1Postfix}xms;
```

If you want to match for start and end of a *string*, you should generally use \A and \z instead of the more generic ^ and $ unless you really need to match start or end of *lines* within a multiline string.

```
$Text =~ m{
    \A      # beginning of the string
    Content # some string
    \z      # end of the string
}xms;

$MultilineText =~ m{
    \A                      # beginning of the string
    .*
    (?: \n Content $ )+      # one or more lines containing the same string
    .*
    \z                      # end of the string
}xms;
```

Usage of named capture groups is also encouraged, particularly for multi-matches. Named capture groups are easier to read/understand, prevent mix-ups when matching more than one capture group and allow extension without accidentally introducing bugs.

Example: Use this...

```
$Contact =~ s{
    \A
    [ ]*
    (?'TrimmedContact'
        (?'FirstName' \w+ )
        [ ]+
```

```
        (?'LastName' \w+ )
    )
    [ ]+
    (?'Email' [^ ]+ )
    [ ]*
    \z
}
{$+{TrimmedContact}}xms;
my $FormattedContact = "$+{LastName}, $+{FirstName} ($+{Email})";
```

instead of this...

```
$Contact =~ s{
    \A
    [ ]*
    (
        ( \w+ )
        [ ]+
        ( \w+ )
    )
    [ ]+
    ( [^ ]+ )
    [ ]*
    \z
}
{$1}xms;
my $FormattedContact = "$3, $2 ($4)";
```

4.1.2.4. Elnevezés

A neveket és a megjegyzéseket angolul kell írni. A változókat, objektumokat és metódusokat leíró főnevekkel vagy főnévi igenevekkel írjuk úgy, hogy az első betű nagybetűs legyen (CamelCase).

Names should be as descriptive as possible. A reader should be able to say what is meant by a name without digging too deep into the code. E.g. use `$ConfigItemID` instead of `$ID`. Examples: `@TicketIDs`, `$Output`, `StateSet()`, etc.

4.1.2.5. Változók

4.1.2.5.1. Deklaráció

Ha több változója van, akkor deklarálhatja azokat egyetlen sorban, ha azok „összetartoznak”:

```
my ( $Minute, $Hour, $Year );
```

Egyébként tördelje azokat külön sorokba:

```
my $Minute;
my $ID;
```

Ne állítson be `undef` vagy `''` kezdeti értéket a deklarációban, ugyanis ez elrejtheti a hibákat a kódban.

```
my $Variable = undef;
```

```
# ugyanaz mint

my $Variable;
```

Akkor állíthat be egy változót '' értékre, ha szövegeket szeretne összefűzni:

```
my $SqlStatement = '';
for my $Part (@Parts) {
    $SqlStatement .= $Part;
}
```

Egyébként „előkészítetlen” figyelmeztetést kaphat.

4.1.2.6. Szubrutinok

4.1.2.6.1. Paraméterek kezelése

A szubrutinoknak átadott paraméterek lekéréséhez az OTRS normális esetben a %Param kivonatot használja (nem a %Params kivonatot). Ez jobban olvasható kódot eredményez, mivel minden esetben tudjuk, hogy amikor %Param kivonatot használjuk a szubrutin kódokban, akkor paraméterkivonat került átadásra a szubrutinnak.

Csak néhány kivételnél kell a paraméterek szabályos listáját használni. Így el szeretnénk kerülni az ehhez hasonlókat:

```
sub TestSub {
    my ( $Self, $Param1, $Param2 ) = @_;
}
```

Inkább ezt szeretnénk használni:

```
sub TestSub {
    my ( $Self, %Param ) = @_;
}
```

Ennek számos előnye van: nem kell megváltoztatnunk a kódot a szubrutinban, amikor egy új paramétert kell átadni, és egy elnevezett paraméterekkel rendelkező függvény hívása sokkal olvashatóbb.

4.1.2.6.2. Több elnevezett paraméter

Ha egy függvényhívás egynél több elnevezett paramétert igényel, akkor tördelje azokat több sorba:

```
    $Self->{LogObject}->Log(
        Priority => 'error',
        Message  => "Need $Needed!",
    );
```

E helyett:

```
    $Self->{LogObject}->Log( Priority => 'error', Message  => "Need $Needed!", );
```

4.1.2.6.3. `return` utasítások

A szubrutinoknak rendelkezniük kell egy `return` utasítással. Az explicit `return` utasítás előnyben részesített az implicit módszernél (az utolsó utasítás eredménye a szubrutinban), mivel ez tisztázza, hogy mit ad vissza a szubrutin.

```
sub TestSub {
    ...
    return; # undef visszaadása, de nem az utolsó utasítás eredménye
}
```

4.1.2.6.4. Explicit visszatérési értékek

Az explicit visszatérési értékek azt jelentik, hogy nem kell egy `return` utasítást tenni egy szubrutinhívást követően.

```
return $Self->{DBObject}->Do( ... );
```

A következő példa jobb, mivel ez explicit módon megmondja, hogy mi kerül visszaadásra. A fenti példával az olvasó nem tudja, hogy mi a visszatérési érték, mivel nem tudhatja, hogy a `Do()` mit ad vissza.

```
return if !$Self->{DBObject}->Do( ... );
return 1;
```

Ha egy szubrutin eredményét hozzárendeli egy változóhoz, akkor egy „jó” változónév jelzi, hogy mi lett visszaadva:

```
my $SuccessfulInsert = $Self->{DBObject}->Do( ... );
return $SuccessfulInsert;
```

4.1.2.7. Csomagok

4.1.2.7.1. `use` utasítások

A `use strict` és `use warnings` utasításoknak kell az első két „use”-nak lennie a modulban. Ez helyes:

```
package Kernel::System::ITSMConfigItem::History;

use strict;
use warnings;

use Kernel::System::User;
use Kernel::System::DateTime;
```

Ez hibás:

```
package Kernel::System::ITSMConfigItem::History;

use Kernel::System::User;
```

```
use Kernel::System::DateTime;

use strict;
use warnings;
```

4.1.2.7.2. Objektumok és azok lefoglalása

Az OTRS-ben sok objektum érhető el. De nem kell minden egyes objektumot használnia minden fájlban az előtétprogram/háttérprogram elválasztásának megtartásához.

- Ne használja a `LayoutObject` objektumot az alapmodulokban (`Kernel/System`).
- Ne használja a `ParamObject` objektumot az alapmodulokban (`Kernel/System`).
- Ne használja a `DBObject` objektumot az előtétprogram modulokban (`Kernel/Modules`).

4.1.3. Jó dokumentáció írása

4.1.3.1. Perldoc

4.1.3.1.1. Documenting backend modules

'NAME' section

This section should include the module name, ' - ' as separator and a brief description of the module purpose.

```
=head1 NAME

Kernel::System::MyModule - Functions to read from and write to files
```

'SYNOPSIS' section

This section should give a short usage example of commonly used module functions.

Usage of this section is optional.

```
=head1 SYNOPSIS

my $Object = $Kernel::OM->Get('Kernel::System::MyModule');

Read data

    my $FileContent = $Object->Read(
        File => '/tmp/testfile',
    );

Write data

    $Object->Write(
        Content => 'my file content',
        File    => '/tmp/testfile',
    );
```

'DESCRIPTION' section

This section should give more in-depth information about the module if deemed necessary (instead of having a long 'NAME' section).

Usage of this section is optional.

```
=head1 DESCRIPTION

This module does not only handle files.

It is also able to:
- brew coffee
- turn lead into gold
- bring world peace
```

'PUBLIC INTERFACE' section

This section marks the begin of all functions that are part of the API and therefore meant to be used by other modules.

```
=head1 PUBLIC INTERFACE
```

'PRIVATE FUNCTIONS' section

This section marks the begin of private functions.

Functions below are not part of the API, to be used only within the module and therefore not considered stable.

It is advisable to use this section whenever one or more private functions exist.

```
=head1 PRIVATE FUNCTIONS
```

4.1.3.1.2. Szubrutinok dokumentálása

A szubrutinokat mindig dokumentálni kell. A dokumentum tartalmaz egy általános leírást arról, hogy mit csinál a szubrutin, egy minta szubrutinhívást, és hogy mit ad vissza a szubrutin. Ezeknek ebben a sorrendben kell lenniük. Egy minta dokumentáció így néz ki:

```
=head2 LastTimeObjectChanged()

Calculates the last time the object was changed. It returns a hash reference with
    information about the object and the time.

    my $Info = $Object->LastTimeObjectChanged(
        Param => 'Value',
    );

This returns something like:

    my $Info = {
        ConfigItemID    => 1234,
        HistoryType     => 'foo',
        LastTimeChanged => '08.10.2009',
    };

=cut
```

Lemásolhat és beilleszthet egy `Data::Dumper` kimenetet a visszatérési értékekhez.

4.1.3.2. Kódmagyarázatok

Általánosságban meg kell próbálni olvashatóan és önmagát magyarázóan írni a kódot, amennyire csak lehetséges. Ne írjon megjegyzést annak magyarázásához, hogy a nyilvánvaló kód mit csinál, mert az szükségtelen megkettőzés. A jó megjegyzéseknek azt

kell elmagyarázniuk, hogy *miért* van valami a kódban, mik a lehetséges mellékhatások és bármi egyéb, amely különleges lehet vagy szokatlanul bonyolult a kóddal kapcsolatban.

Ragaszkodjon a következő irányelvekhez:

Tegye a kódot annyira olvashatóvá, hogy ne legyen szükség magyarázatra, ha ez lehetséges.

Mindig vonzóbb a kódot úgy írni, hogy nagyon olvasható és önmagát magyarázó legyen, például pontos változónevekkel és függvénynevekkel.

Don't say what the code says (DRY -> Don't repeat yourself).

Ne ismételjen (nyilvánvaló) kódot a magyarázatokban.

```
# HIBÁS:

# beállítási objektum lekérése
my $ConfigObject = $Kernel::OM->Get('Kernel::Config');
```

Azt dokumentálja, hogy a kód *miért* van ott, és ne azt, hogy hogyan működik.

Általában a kódmagyarázatoknak a kód *célját* kellene elmagyarázniuk, és nem azt, hogy részletesen hogyan működik. Lehetnek kivételek különösen bonyolult kódnál, de ebben az esetben egy átszerkesztés lenne dicséretes, hogy olvashatóbb legyen.

Dokumentálja a buktatókat.

Mindent dokumentálni kell, ami nem világos, furfangos vagy amit összerakott a fejlesztés során.

Használjon teljes soros mondatszerű magyarázatokat az algoritmus bekezdéseinek dokumentálásához.

Mindig teljes mondatokat használjon (első betűt nagybetűvel írva és központozással). Egy mondat következő sorait be kell húzni.

```
# Annak ellenőrzése, hogy meg lett-e adva objektumnév.
if ( !$_[1] ) {
    $_[0]->_DieWithError(
        Error => "Hiba: hiányzó paraméter (objektumnév)",
    );
}

# Az objektum rögzítése, amelyet lekérni készülünk, hogy potenciálisan jobb
#   hibaüzenetet készíthessünk.
# Utasításmódosító „if”-nek kell lennie, különben a „local” helyi lesz az
#   „if”-blokk hatóköréhez képest.
local $CurrentObject = $_[1] if !$CurrentObject;
```

Használjon rövid sorvégi magyarázatokat a részletes információk hozzáadásához.

Ez lehet vagy teljes mondat (nagy kezdőbetű és központozás), vagy csak egy kifejezés (kis kezdőbetű és nincs központozás).

```
$BuildMode = oct $Param{Mode};   # oktális *típusról*, nem oktális *típusra*

# vagy

$BuildMode = oct $Param{Mode};   # Átalakítás oktális *típusról*, nem oktális *típusra*.
```

4.1.4. Adatbázis kölcsönhatás

4.1.4.1. SQL-utasítások deklarációja

Ha nincs esély az SQL-utasítás megváltoztatására, akkor azt a `Prepare` függvényben kell használni. Ennek az az oka, hogy az SQL-utasítás és a kötési paraméterek közelebb vannak egymáshoz.

Az SQL-utasítást egy összefűzések nélküli, pontosan behúzott szövegként kell megírni úgy, mint például ezt:

```
return if !$Self->{DBObject}->Prepare(
    SQL => '
        SELECT art.id
        FROM article art, article_sender_type ast
        WHERE art.ticket_id = ?
            AND art.article_sender_type_id = ast.id
            AND ast.name = ?
        ORDER BY art.id',
    Bind => [ \$Param{TicketID}, \$Param{SenderType} ],
);
```

Ezt könnyű olvasni és módosítani, és az üres karaktereket jól tudják kezelni a támogatott DBMS-ek. Az automatikusan előállított SQL-kódnál (mint a `TicketSearch` modulban) ez a behúzás nem szükséges.

4.1.4.2. Visszatérés hibák esetén

Valahányszor adatbázis-függvényeket használ, kezelnie kell a hibákat. Ha valami elromlik, az visszakerül a szubrutinból:

```
return if !$Self->{DBObject}->Prepare( ... );
```

4.1.4.3. Korlát használata

Használja a `Limit => 1` korlátozást, ha csak egyetlen sort vár visszatérésként.

```
$Self->{DBObject}->Prepare(
    SQL   => 'SELECT id FROM users WHERE username = ?',
    Bind  => [ \$Username ],
    Limit => 1,
);
```

4.1.4.4. A `while` ciklus használata

Mindig használja a `while` ciklust még akkor is, ha csak egyetlen sort vár visszatérésként, mivel néhány adatbázis nem szabadítja fel az utasításkezelőt, és ez furcsa hibákhoz vezethet.

4.2. JavaScript

4.2.1. Böngészőkezelés

Az összes JavaScript betöltődik minden böngészőben (nincsenek böngésző trükközések a sablonfájlokban). A kód felelős annak eldöntéséért, hogy ki kell hagynia vagy végre kell hajtania saját magának bizonyos részeit az egyes böngészőkben.

4.2.2. Könyvtárszerkezet

Könyvtárszerkezet a `js/` mappán belül:

```
* js
    * thirdparty              # harmadik féltől származó függvénykönyvtárak,
        * ckeditor-3.0.1      #   amelyek mindig tartalmaznak verziószámot a
        * jquery-1.3.2        #   könyvtáron belül
    * Core.Agent.*            # az ügyintézői felületre jellemző dolgok
    * Core.Customer.*         # ügyfélfelület
    * Core.*                  # közös API
```

4.2.2.1. Harmadik féltől származó kód

Minden harmadik féltől származó modul saját alkönyvtárat kap: „modulnév”-„verziószám” (például ckeditor-3.0.1, jquery-1.3.2). Ezen belül a fájlneveknek nem kell verziószámot vagy előtagot tartalmaznia (hibás: `jquery/jquery-1.4.3.min.js`, helyes: `jquery-1.4.3/jquery.js`).

4.2.3. Változók

- A változóneveket CamelCase jelölésrendszerben kell írni, akárcsak a Perlben.
- A jQuery objektumot tartalmazó változókat `$` karakterrel kell kezdeni, például: `$Tooltip`.

4.2.4. Függvények

- A függvényneveket CamelCase jelölésrendszerben kell írni, akárcsak a Perlben.

4.2.5. Névterek

- Ez a rész még nincs megírva...

4.2.6. Kódmagyarázatok

A Perl-kód magyarázási irányelvei a JavaScriptre is vonatkoznak.

- Egysoros megjegyzéseket `//` karakterekkel kell készíteni.
- Hosszabb megjegyzéseket `/* ... */` karakterekkel kell készíteni.
- Ha megjegyzésre állítja a JavaScript kód egyes részeit, akkor csak a `//` karaktereket használja, ugyanis a `/* ... */` használata problémákat okozhat a reguláris kifejezéseknél a kódban.

4.2.7. Eseménykezelés

- Always use `$.on()` instead of the event-shorthand methods of jQuery for better readability (wrong: `$SomeObject.click(...)`, right: `$SomeObject.on('click', ...)`.
- If you `$.on()` events, make sure to `$.off()` them beforehand, to make sure that events will not be bound twice, should the code be executed another time.
- Make sure to use `$.on()` with namespacing, such as `$.on('click.<Name>')`.

4.3. HTML

- Use HTML 5 notation. Don't use self-closing tags for non-void elements (such as div, span, etc.).

- Use proper intendation. Elements which contain other non-void child elements should not be on the same level as their children.
- Don't use HTML elements for layout reasons (e.g. using `br` elements for adding space to the top or bottom of other elements). Use the proper CSS classes instead.
- Don't use inline CSS. All CSS should either be added by using predefined classes or (if necessary) using JavaScript (e.g. for showing/hiding elements).
- Don't use JavaScript in TT templates. All needed JavaScript should be part of the proper library for a certain frontend module or of a proper global library. If you need to pass JavaScript data to the frontend, use `$LayoutObject->AddJSData()`.

4.4. CSS

- A legkisebb felbontás 1024×768 képpont.
- Az elrendezés folyékony, amely azt jelenti, hogy ha a képernyő szélesebb, akkor a helyet fel fogja használni.
- Az abszolút méretmeghatározásokat képpontban (px) kell megadni, hogy következetes kinézetet kapjon a legtöbb platformon és böngészőben.
- A dokumentáció CSSDOC használatával készül (nézze meg a CSS-fájlokat példaként). Az összes logikai blokknak rendelkeznie kell egy CSSDOC megjegyzéssel.

4.4.1. Szerkezet

- Az objektumorientált CSS megközelítést követjük. Lényegében ez azt jelenti, hogy az elrendezés különböző általános építőkockák egyesítésével érhető el egy bizonyos látványterv megvalósításához.
- Ahol csak lehetséges, nem szabad a modulra jellemző látványtervet használni. Például ezért nem dolgozunk azonosítókkal a body elemen sem, ha az elkerülhető.

4.4.2. Stílus

- Az összes meghatározásnak ugyanabban a sorában van a { karakter mint a kiválasztó, az összes szabály szabályonként egy sorban van meghatározva, a meghatározások egyetlen } karaktert tartalmazó sorral végződnek. Nézze meg a következő példát:

```
#Selector {
    width: 10px;
    height: 20px;
    padding: 4px;
}
```

- A : és szabály értéke között van egy szóköz.
- Minden szabály 4 szóközzel van behúzva.
- Ha több kiválasztó van megadva, akkor vesszővel válassza el azokat, és mindegyiket tegye külön sorba:

```
#Selector1,
#Selector2,
#Selector3 {
    width: 10px;
```

```
}
```

- Ha a szabályok egyesíthetők, akkor egyesítse azokat (például egyesítse a `background-position`, `background-image`, stb. szabályokat a `background` szabályba).
- A szabályoknak logikai sorrendben kell lenniük egy meghatározáson belül (az összes színre jellemző szabály együtt, az összes pozicionáló szabály együtt, stb.).
- Az összes azonosító és név CamelCase jelölésrendszerben van írva:

```
<div class="NavigationBar" id="AdminMenu"></div>
```

5. Felhasználó felület tervezése

5.1. Nagybetűs írás

Ez a szakasz azt mutatja be, hogy az angol felhasználói felület különböző részeit hogyan kell nagybetűsen írni. További információkért érdemes átnézni ezt a hasznos oldalt.

- A címsorok (h1-h6) és a címek (nevek, mint például `Queue View`) „címstílusú” nagy kezdőbetűs írásmódban vannak, amely azt jelenti, hogy az összes első betű nagybetűvel lesz írva (néhány kivétellel, mint például „this”, „and”, „or”, stb.).

  Példák: `Action List`, `Manage Customer-Group Relations`.

- Az egyéb szerkezeti elemek (mint például gombok, címkék, lapok, menüpontok) „mondatstílusú” nagy kezdőbetűs írásmódban vannak (csak a kifejezés első betűje van nagybetűvel írva), de nincs lezáró pont hozzáadva a kifejezés mondatként való befejezéséhez.

  Példák: `First name`, `Select queue refresh time`, `Print this ticket`.

- A leíró szövegek és a buboréksúgó tartalmak teljes mondatként vannak írva.

  Példa: `This value is required.`

- A fordításoknál ellenőrizni kell, hogy címstílusú nagybetűs írás megfelelő-e a célnyelven is. Lehet, hogy meg kell változtatni mondatstílusú nagybetűs írásra vagy valami másra.

6. Akadálymentesítési útmutató

Ez a dokumentum hivatott elmagyarázni az akadálymentesítési problémákkal kapcsolatos alapokat, és irányelveket ad az OTRS-ben való közreműködéshez.

6.1. Akadálymentesítési alapok

6.1.1. Mi az akadálymentesítés?

Az akadálymentesítés egy általános kifejezés annak leírásához, hogy egy termék, eszköz, szolgáltatás vagy környezet milyen mértékben érhető el a lehető legtöbb ember által. Az akadálymentesítés tekinthető úgyis mint „képesség a hozzáféréshez”, és néhány rendszer vagy dolog lehetséges előnyeként. Az akadálymentesítés gyakran a fogyatékkal rendelkező emberekre és a dolgokhoz való hozzáférésük jogára összpontosít, gyakran kisegítő technológiák használatán keresztül.

A webfejlesztés környezetében akadálymentesítéskor a sérült embereknek a webes felületekhez való teljes hozzáférés engedélyezésén van a hangsúly. Például az emberek ezen csoportja részlegesen látássérült vagy teljesen vak embereket tartalmazhat. Míg az előbbiek továbbra is képesek részlegesen használni a grafikus felhasználói felületet, addig az utóbbiaknak teljes mértékben a kisegítő technológiákra kell támaszkodniuk, az olyan szoftverekre, amelyek felolvassák számukra a képernyőt (képernyőolvasók).

6.1.2. Miért fontos ez az OTRS-nél?

A sérült felhasználóknak történő hozzáférés engedélyezése az OTRS rendszerekhez önmagában indokolt cél. Tiszteletet tanúsít.

Továbbá az akadálymentesítési szabványok teljesítése egyre inkább fontossá válik az állami szektorban (kormányzati intézményeknél) és a nagyvállalatoknál, amely mindkettő az OTRS célcsoportjához tartozik.

6.1.3. Hogyan tudok sikeresen dolgozni az akadálymentesítési problémákon akkor is, ha nem vagyok fogyatékos?

Ez nagyon egyszerű. Tegyen úgy, mintha vak lenne.

Ne tegye a következőket:

- ne használja az egeret
- ne nézzen a képernyőre

Ezután próbálja meg az OTRS-t csak egy képernyőolvasó és a billentyűzet segítségével használni. Ez ötletet adhat arra, hogy mit fog érezni egy vak ember.

6.1.4. Rendben, de nincs képernyőolvasóm!

Amíg a kereskedelmi képernyőolvasók - mint például a JAWS (talán a legismertebb) - nagyon drágák lehetnek, léteznek nyílt forrású képernyőolvasók, amelyeket telepíthet és használhat:

- NVDA, egy képernyőolvasó Windows alá.
- ORCA, egy képernyőolvasó Gnome/Linux alá.

Most már nincs többé mentsége. ;)

6.2. Akadálymentesítési szabványok

Ez a szakasz csak hivatkozásként szolgál, nem kell magát a szabványokat áttanulmányoznia ahhoz, hogy képes legyen akadálymentesítési problémákon dolgozni az OTRS-ben. Megpróbáljuk kigyűjteni a fontos irányelveket ebben a dokumentumban.

6.2.1. Webtartalom akadálymentesítési irányelvek (WCAG)

Ez a W3C szabvány általános irányelveket ad ahhoz, hogyan hozhatók létre akadálymentes weboldalak.

- WCAG 2.0

- Hogyan tegyen eleget a WCAG 2.0 szabványnak
- A WCAG 2.0 megértése

A WCAG az akadálymentesítési támogatás különböző szintjeivel rendelkezik. Jelenleg az A szint támogatását tervezzük, ugyanis az AA és az AAA olyan kérdésekkel foglalkozik, amely nem tűnik fontosnak az OTRS-nél.

6.2.2. Akadálymentes gazdag internetes alkalmazások (WAI-ARIA) 1.0

Ez a szabvány foglalkozik a statikus tartalom eltolásából eredő különleges problémákkal a dinamikus webalkalmazásoknál. Olyan kérdésekkel foglalkozik, mint például hogyan értesülhet a felhasználó az AJAX kérésekből eredményezett felhasználói felület változásairól.

- WAI-ARIA 1.0

6.3. Megvalósítási irányelvek

6.3.1. Alternatívák biztosítása a nem szöveges tartalomhoz

Cél: *a felhasználónak megjelenő összes nem szöveges tartalomnak legyen szöveges alternatívája, amely ugyanazt a célt szolgálja.* (WCAG 1.1.1)

Nagyon fontos megérteni, hogy a képernyőolvasók csak a szöveges információkat és az elérhető metaadatokat tudják megjeleníteni a felhasználónak. Hogy egy példát mondjunk, amikor egy képernyőolvasó a `<a href="#" class="CloseLink"></a>` hivatkozást látja, akkor csak a „hivatkozást” tudja felolvasni a felhasználónak, nem a hivatkozás célját. Egy apró fejlesztéssel akadálymentessé válhat: `<a href="#" class="CloseLink" title="Felületi elem bezárása"></a>`. Ebben az esetben a felhasználó a „hivatkozás felületi elem bezárása” szöveget hallhatja, íme!

Fontos, hogy a szöveget mindig a leginkább „beszédes” módon fogalmazza meg. Egyszerűen képzelje el, hogy csak ennyi információval rendelkezik. Segíteni fog önnek? Képes megérteni a célját pusztán hallás után?

Kövesse ezeket a szabályokat, amikor az OTRS-en dolgozik:

- *Szabály*: Ahol csak lehetséges, használjon beszédes szövegeket, és fogalmazzon meg valódi, érthető és pontos mondatokat. A „Felületi elem bezárása” sokkal jobb mint a „Bezárás”, mert az utóbbi fölösleges.
- *Szabály*: A hivatkozásoknak mindig legyen vagy olyan szöveges tartalma, amelyet a képernyőolvasók kimondanak (`<a href="#" >A bejegyzés törlése</a>`), vagy `title` attribútuma (`<a href="#" title="Felületi elem bezárása"></a>`).
- *Szabály*: A képeknek mindig legyen alternatív szövege, amely felolvasható a felhasználónak (`<img src="house.png" alt="Fénykép egy házról" />`).

6.3.2. A navigáció könnyűvé tétele

Cél: *lehetővé tenni a felhasználónak, hogy könnyen navigáljon az aktuális oldalon és az egész alkalmazásban.*

A `title` címke az első dolog, amit a felhasználó hall a képernyőolvasótól, amikor megnyit egy weboldalt. Az OTRS-nél mindig csak egyetlen `h1` elem van az oldalon az aktuális

oldalt jelezve (a `title` elemből vett információ egy részét tartalmazza). Ez a navigációs információ segít megérteni a felhasználónak, hogy éppen hol van, és mi az aktuális oldal célja.

- *Szabály*: Mindig pontos címet adjon az oldalnak, amely lehetővé teszi a felhasználónak annak megértését, hogy jelenleg éppen hol van.

A képernyőolvasók képesek a HTML beépített dokumentumszerkezetét használni (címsorok h1 és h6 között) egy dokumentum szerkezetének meghatározásához, és lehetővé tenni a felhasználónak, hogy egyik szakaszról a másik szakaszra ugorjon. Azonban ez nem elegendő egy dinamikus webalkalmazás szerkezetének kifejezéséhez. Ez az, amiért az ARIA számos olyan „iránypont” szerepet határoz meg, amelyek megadhatók az elemeknek, hogy jelezzék a navigációs jelentésüket.

A HTML dokumentumok érvényességének megtartásához a `role` attribútumok (ARIA iránypont szerepek) nincsenek közvetlenül a forráskódba beszúrva, hanem olyan osztályok által kerülnek beszúrásra, amelyeket később az `OTRS.UI.Accessibility` osztályban lévő JavaScript függvények fognak használni a megfelelő `role` attribútumok beszúrásához a csomópontba.

- *Szabály*: Használjon WAI-ARIA iránypont szabályokat a tartalom szerkezetbe foglalásához a képernyőolvasók számára.
  - Reklámcsík: `<div class="ARIARoleBanner"></div>` helyett `<div class="ARIARoleBanner" role="banner"></div>` lesz
  - Navigáció: `<div class="ARIARoleNavigation"></div>` helyett `<div class="ARIARoleNavigation" role="navigation"></div>` lesz
  - Keresőfunkció: `<div class="ARIARoleSearch"></div>` helyett `<div class="ARIARoleSearch" role="search"></div>` lesz
  - Fő alkalmazásterület: `<div class="ARIARoleMain"></div>` helyett `<div class="ARIARoleMain" role="main"></div>` lesz
  - Lábléc: `<div class="ARIARoleContentinfo"></div>` helyett `<div class="ARIARoleContentinfo" role="contentinfo"></div>` lesz

A `<form<` elemeken belüli navigációnál szükséges a fogyatékos felhasználónak tudnia, hogy mi az egyes beviteli elemek célja. Ezt el lehet érni a szabványos HTML `<label>` elemek használatával, amelyek kapcsolatot hoznak létre a címke és az űrlap elem között.

Amikor egy beviteli elem megkapja a fókuszt, akkor a képernyőolvasó általában fel fogja olvasni a hozzá kapcsolt címkét, így a felhasználó hallhatja annak pontos célját. További előnye a látó felhasználóknak, hogy rákattinthatnak a címkére, és a beviteli elem meg fogja kapni a fókuszt (különösen jelölőnégyzeteknél hasznos például).

- *Szabály*: Adjon meg `<label>` elemeket az *összes* űrlapelem (`input`, `select`, `textarea`) mezőhöz.

  Példa: `<label for="date">Dátum:</label><input type="text" name="date" id="date"/>`

6.3.3. A kölcsönhatás lehetővé tétele

Cél: *lehetővé tenni a felhasználónak, hogy az összes kölcsönhatást végrehajtsa pusztán a billentyűzet használatával.*

Miközben technikailag lehetséges kölcsönhatásokat létrehozni JavaScript segítségével tetszőleges HTML elemen, ezt korlátozni kell azon elemekre, amelyekkel a felhasználó

kölcsönhatásba léphet a billentyűzet használatával. Különösen azt kell lehetővé tenni számukra, hogy fókuszt adjanak az elemnek, és kölcsönhatásba lépjenek vele. Például egy felületi elemet ki-be kapcsoló nyomógombot nem egy JavaScript `onclick` eseményfigyelővel összekötött span elem használatával kellene megoldani, hanem egy a címkének kellene lennie (vagy tartalmaznia kellene), hogy világossá tegye a képernyőolvasónak, hogy ez az elem kölcsönhatást okozhat.

- *Szabály*: A kölcsönhatásoknál mindig olyan elemeket használjon, amelyek megkaphatják a fókuszt, mint például a, `input`, `select` és `button`.

- *Rule*: Győződjön meg arról, hogy a felhasználó mindig képes-e azonosítani a kölcsönhatás természetét (nézze meg a szabályokat a nem szöveges tartalommal és az űrlapelemek címkézésével kapcsolatban).

Cél: *Tudatni a felhasználóval a dinamikus változtatásokat.*

Az akadálymentesítési problémák egy különleges területe a dinamikus változtatások a felhasználói felületen, amelyeket JavaScript vagy AJAX hívások okoznak. A képernyőolvasó nem fog beszélni a felhasználónak a változtatásokról különleges óvintézkedések nélkül. Ez egy bonyolult téma, és még nem lehet itt teljesen elmagyarázni.

- *Szabály*: Mindig használja az `OTRS.Validate` érvényesítő keretrendszert az űrlap érvényesítéséhez.

  Ez biztosítani fogja, hogy a hiba buboréksúgókat felolvassa a képernyőolvasó. Ily módon a vak felhasználó a) megtudja azt az elemet, amelynek hibája van, és b) kap egy szöveget, amely leírja a hibát.

- *Szabály*: Használja az `OTRS.UI.Accessibility.AudibleAlert()` függvényt, hogy értesítse a felhasználót az egyéb fontos felhasználói felületet érintő változtatásokról.

- *Szabály*: Használja az `OTRS.UI.Dialog` keretrendszert a kizárólagos párbeszédablakok létrehozásához. Ezek már optimalizálva vannak az akadálymentesítéshez.

6.3.4. Általános képernyőolvasó optimalizálások

Cél: *segíteni a képernyőolvasókat a munkájukban.*

- *Szabály*: Minden egyes oldalnak azonosítania kell a saját fő nyelvét azért, hogy a képernyőolvasók ki tudják választani a megfelelő beszédszintetizátor motort.

  Példa: `<html lang="hu">...</html>`

7. Egységtesztek

OTRS provides a test suite which can be used to develop and run unit tests for all system related code.

7.1. Egy tesztfájl létrehozása

The test files are stored in .t files under `scripts/test/*.t`. For example, let's take a look at the file `scripts/test/Calendar.t` for the Calendar class.

Every test file should ideally instantiate unit test helper object at the start, so it can benefit from some built-in methods provided by it:

```
# --
# Copyright (C) 2001-2018 OTRS AG, https://otrs.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

use strict;
use warnings;
use utf8;

use vars (qw($Self));

$Kernel::OM->ObjectParamAdd(
    'Kernel::System::UnitTest::Helper' => {
        RestoreDatabase => 1,
    },
);
my $Helper = $Kernel::OM->Get('Kernel::System::UnitTest::Helper');
```

By providing `RestoreDatabase` parameter to helper constructor, any database statement executed during the unit test will be rolled back at the end, making sure no permanent change has been done.

Like any other test suite, OTRS provides assertion methods which can be used to test conditions. For example, this is how we create a test user and test that it has been indeed created:

```
my $UserLogin = $Helper->TestUserCreate();
my $UserID = $UserObject->UserLookup( UserLogin => $UserLogin );

$Self->True(
    $UserID,
    "Test user $UserID created"
);
```

Please consult API section below for complete list of assertion methods.

It's always good practice to create random data in unit tests, which can help distinguish it from previously added data. Use random methods from API to get the strings and include them in your parameters:

```
my $RandomID = $Helper->GetRandomID();

# create test group
my $GroupName = 'test-calendar-group-' . $RandomID;
my $GroupID   = $GroupObject->GroupAdd(
    Name    => $GroupName,
    ValidID => 1,
    UserID  => 1,
);

$Self->True(
    $GroupID,
    "Test group $GroupID created"
);
```

Good developers make their unit test easy to maintain. Consider putting all test cases in an array and then iterate over them with some code. This will provide an easy way to extend the test later:

```
#
```

```
# Tests for CalendarCreate()
#
my @Tests = (
    {
        Name    => 'CalendarCreate - No params',
        Config  => {},
        Success => 0,
    },
    {
        Name   => 'CalendarCreate - All required parameters',
        Config => {
            CalendarName => "Calendar-$RandomID",
            Color        => '#3A87AD',
            GroupID      => $GroupID,
            UserID       => $UserID,
        },
        Success => 1,
    },
    {
        Name   => 'CalendarCreate - Same name',
        Config => {
            CalendarName => "Calendar-$RandomID",
            Color        => '#3A87AD',
            GroupID      => $GroupID,
            UserID       => $UserID,
        },
        Success => 0,
    },
);

for my $Test (@Tests) {

    # make the call
    my %Calendar = $CalendarObject->CalendarCreate(
        %{ $Test->{Config} },
    );

    # check data
    if ( $Test->{Success} ) {
        for my $Key (qw(CalendarID GroupID CalendarName Color CreateTime CreateBy ChangeTime
 ChangeBy ValidID)) {
            $Self->True(
                $Calendar{$Key},
                "$Test->{Name} - $Key exists",
            );
        }

        KEY:
        for my $Key ( sort keys %{ $Test->{Config} } ) {
            next KEY if $Key eq 'UserID';

            $Self->IsDeeply(
                $Test->{Config}->{$Key},
                $Calendar{$Key},
                "$Test->{Name} - Data for $Key",
            );
        }
    }
    else {
        $Self->False(
            $Calendar{CalendarID},
            "$Test->{Name} - No success",
        );
    }
}
```

7.2. Előfeltételek a teszteléshez

To be able to run the unit tests, you need to have all optional environment dependencies (Perl modules and other modules) installed, except those for different database backends

than what you are using. Run `bin/otrs.CheckEnvironment.pl` to verify your module installation.

Szüksége van egy teljes képzésű tartományneven (FQDN) futó OTRS webes előtétprogram egy példányára, amely az OTRS helyi `Config.pm` fájljában be van állítva. Ennek az OTRS példánynak ugyanazt az adatbázist kell használnia, amelyek az egységtesztekhez vannak beállítva.

7.3. Tesztelés

To run your tests, just use `bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar` to use `scripts/test/Calendar.t`.

```
shell:/opt/otrs> bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar.t:
+-------------------------------------------------------------------+
.....................................................................................................
=====================================================================
yourhost ran tests in 2s for OTRS 6.0.x git
All 97 tests passed.
shell:/opt/otrs>
```

You can even run several tests at once, just supply additional test arguments to the command:

```
shell:/opt/otrs> bin/otrs.Console.pl Dev::UnitTest::Run --test Calendar --test Appointment
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar.t:
+-------------------------------------------------------------------+
.....................................................................................................
+-------------------------------------------------------------------+
/opt/otrs/scripts/test/Calendar/Appointment.t:
+-------------------------------------------------------------------+
.....................................................................................................................
=====================================================================
yourhost ran tests in 5s for OTRS 6.0.x git
All 212 tests passed.
shell:/opt/otrs>
```

If you execute `bin/otrs.Console.pl Dev::UnitTest::Run` without any argument, it will run all tests found in the system. Please note that this can take some time to finish.

Provide `--verbose` argument in order to see messages about successful tests too. Any errors encountered during testing will be displayed regardless of this switch, provided they are actually raised in the test.

7.4. Unit Test API

OTRS provides API for unit testing that was used in the previous example. Here we'll list the most important functions, please also see the online API reference of `Kernel::System::UnitTest`.

`True()`

Ez a függvény azt teszteli, hogy a megadott skalár érték igaz érték-e a Perlben.

```
$Self->True(
    1,
```

```
    'Scalar 1 is always evaluated as true'
);
```

`False()`

Ez a függvény azt teszteli, hogy a megadott skalár érték hamis érték-e a Perlben.

```
$Self->False(
    0,
    'Scalar 0 is always evaluated as false'
);
```

`Is()`

Ez a függvény azt teszteli, hogy a megadott skalár változók egyenlők-e.

```
$Self->Is(
    $A,
    $B,
    'Tesztnév',
);
```

`IsNot()`

This function tests whether the given scalar variables are unequal.

```
$Self->IsNot(
    $A,
    $B,
    'Test Name'
);
```

`IsDeeply()`

Ez a függvény összetett adatszerkezeteket hasonlít össze az egyenlőséghez. `$A` és `$B` hivatkozás kell legyen.

```
$Self->IsDeeply(
    $A,
    $B,
    'Test Name'
);
```

`IsNotDeeply()`

Ez a függvény összetett adatszerkezeteket hasonlít össze a nem egyenlőséghez. `$A` és `$B` hivatkozás kell legyen.

```
$Self->IsNotDeeply(
    $A,
    $B,
    'Test Name'
);
```

Besides this, unit test helper object also provides some helpful methods for common test conditions. For full reference, please see the online API reference of `Kernel::System::UnitTest::Helper`.

`GetRandomID()`

This function creates a random ID that can be used in tests as a unique identifier. It is guaranteed that within a test this function will never return a duplicate.

Megjegyzés

Please note that these numbers are not really random and should only be used to create test data.

```
my $RandomID = $Helper->GetRandomID();
# $RandomID = 'test6326004144100003';
```

`TestUserCreate()`

This function creates a test user that can be used in tests. It will be set to invalid automatically during the destructor. It returns the login name of the new user, the password is the same.

```
my $TestUserLogin = $Helper->TestUserCreate(
    Groups   => ['admin', 'users'],          # optional, list of groups to add this user
 to (rw rights)
    Language => 'de',                        # optional, defaults to 'en' if not set
);
```

`FixedTimeSet()`

This functions makes it possible to override the system time as long as this object lives. You can pass an optional time parameter that should be used, if not, the current system time will be used.

Megjegyzés

All calls to methods of `Kernel::System::Time` and `Kernel::System::DateTime` will use the given time afterwards.

```
$HelperObject->FixedTimeSet(366475757);         # with Timestamp
$HelperObject->FixedTimeSet($DateTimeObject);   # with previously created DateTime
 object
$HelperObject->FixedTimeSet();                  # set to current date and time
```

`FixedTimeUnset()`

This functions restores the regular system time behavior.

`FixedTimeAddSeconds()`

This functions adds a number of seconds to the fixed system time which was previously set by `FixedTimeSet()`. You can pass a negative value to go back in time.

`ConfigSettingChange()`

This functions temporarily changes a configuration setting system wide to another value, both in the current instance of the `ConfigObject` and also in the system configuration on disk. This will be reset when the `Helper` object is destroyed.

Megjegyzés

Please note that this will not work correctly in clustered environments.

```
$Helper->ConfigSettingChange(
    Valid => 1,            # (optional) enable or disable setting
    Key   => 'MySetting',  # setting name
    Value => { ... } ,     # setting value
);
```

`CustomCodeActivate()`

This function will temporarily include custom code in the system. For example, you may use this to redefine a subroutine from another class. This change will persist for remainder of the test. All code will be removed when the Helper object is destroyed.

Megjegyzés

Please note that this will not work correctly in clustered environments.

```
$Helper->CustomCodeActivate(
    Code => q^
use Kernel::System::WebUserAgent;
package Kernel::System::WebUserAgent;
use strict;
use warnings;
{
    no warnings 'redefine';
    sub Request {
        my $JSONString = '{"Results":{},"ErrorMessage":"","Success":1}';
        return (
            Content => \$JSONString,
            Status  => '200 OK',
        );
    }
}
1;^,
    Identifier => 'News',   # (optional) Code identifier to include in file name
);
```

`ProvideTestDatabase()`

This function will provide a temporary database for the test. Please first define test database settings in `Kernel/Config.pm`, i.e:

```
$Self->{TestDatabase} = {
    DatabaseDSN  => 'DBI:mysql:database=otrs_test;host=127.0.0.1;',
    DatabaseUser => 'otrs_test',
    DatabasePw   => 'otrs_test',
};
```

The method call will override global database configuration for duration of the test, i.e. temporary database will receive all calls sent over system `DBObject`.

All database contents will be automatically dropped when the `Helper` object is destroyed.

This method returns `undef` in case the test database is not configured. If it is configured, but the supplied XML cannot be read or executed, this method will `die()` to interrupt the test with an error.

```
$Helper->ProvideTestDatabase(
    DatabaseXMLString => $XML,      # (optional) OTRS database XML schema to execute
                                    # or
```

```
    DatabaseXMLFiles => [           # (optional) List of XML files to load and execute
        '/opt/otrs/scripts/database/otrs-schema.xml',
        '/opt/otrs/scripts/database/otrs-initial_insert.xml',
    ],
);
```

A. függelék - További erőforrások

otrs.com

Az OTRS weboldala a forráskóddal, dokumentációval és hírekkel a www.otrs.com címen érhető el. Itt a hivatalos szakmai szolgáltatásokkal és az OTRS adminisztrátorképzési szemináriumokkal kapcsolatos információkat is megtalálja az OTRS csoporttól, az OTRS készítőjétől.

Internetes API könyvtár

Az OTRS fejlesztői API dokumentáció elérhető a Perl és a JavaScript programnyelvekhez.

Fejlesztői levelezőlista

Az OTRS fejlesztői levelezőlista a http://lists.otrs.org/ címen érhető el.