Creatio Academy

Localizable resources

Version 8.0

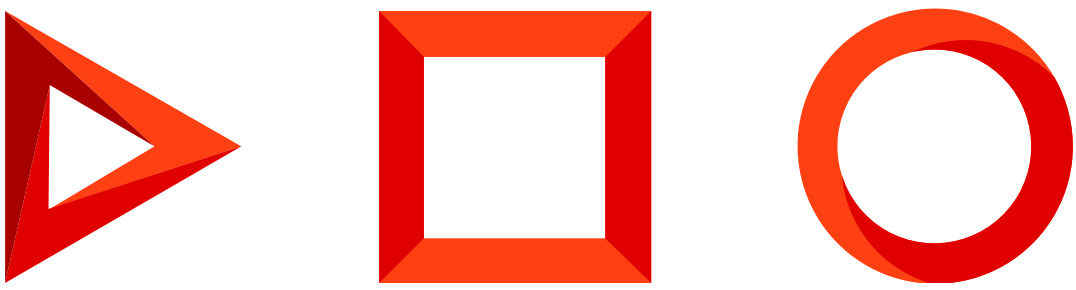


Table of Contents

Localizable resources basics

Advanced

Localizable resources are resources required to display Creatio UI in the user's profile language. Localizable resources include images and localizable strings.

Localizable resources are a part of Creatio configuration resources. Creatio resources are contained in packages and bound to the base schema. When you request resources from a specific schema, Creatio collects them based on package hierarchy, levels, and positions.

Localizable resources utilize the concepts of primary and additional language.

The **primary language** (primary language culture) is the default Creatio UI language.

The **additional language** (additional language culture) is the Creatio UI language that was changed in the user profile and is different from the primary language.

To **change the primary language**, activate the new language. Learn more about activating the UI language in a separate article: Manage UI languages.

The Section and Detail Wizards, Process and Case Designers, and the [*Translation*] section use every language installed in Creatio. In other cases, only activated languages (i. e., languages for which the [*Active*] checkbox is selected) are available. This separation decreases the task execution time, for example, logging in, opening the record page, etc. Learn more in the user documentation: Manage UI languages, Localize UI via the [*Translation*] section.

Creatio provides the following localizable resource **types**:

- simple localizable resources
- bound localizable resources

Display localizable resources

Creatio displays localizable resources based on the following:

- display mode
- display method

Display modes of localizable resources

Creatio implements the following **display modes of localizable resources**:

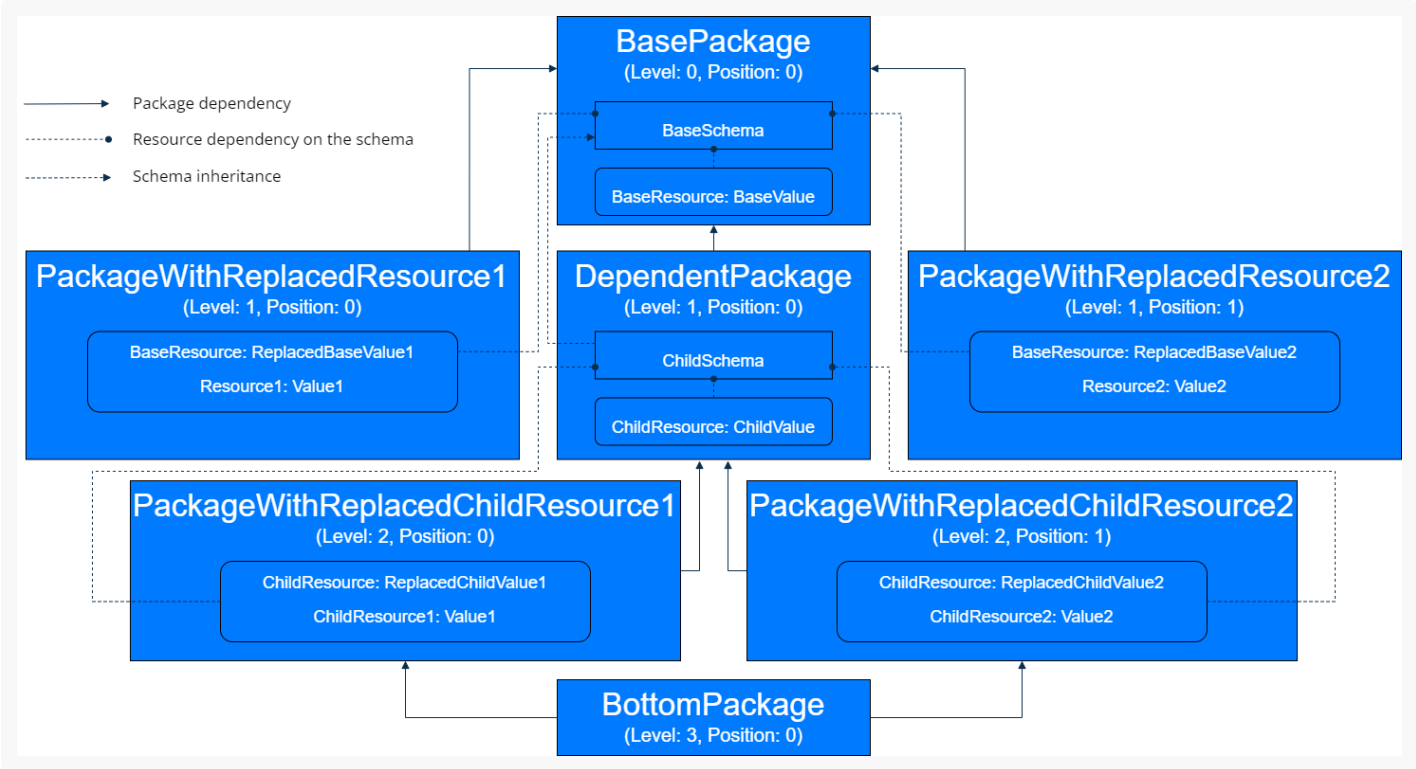
- design-time mode
- run-time mode

The package hierarchy is applied when displaying localizable resources.

Design-time mode

The **purpose** of design-time mode is to display localizable resources in Designers and Wizards. In this mode, the localizable resource hierarchy of configuration element schemas is built only up to the level of the package that contains the scheme with the requested resources. Creatio builds the hierarchy based on the localizable package resources that are added to the hierarchy via direct connections.

The example mechanism that displays localizable resources is provided below. The example uses the `ChildResource: ChildValue` resource of the `ChildSchema` schema in the `DependentPackage` package. View the package hierarchy in the figure below.



The **resulting resources** for the `ChildSchema` schema in design-time mode are as follows:

- `BaseResource: BaseValue`
- `ChildResource: ChildValue`

In design-time mode, resources do not depend on the following:

- resources of the `PackageWithReplacedResource1` and `PackageWithReplacedResource2` packages because these packages are not involved in the hierarchy
- resources of the `PackageWithReplacedChildResource1` and `PackageWithReplacedChildResource2` packages because these packages are lower in the hierarchy compared to the requested schema

If a start package (the package from which the resource collection starts) is specified for the requested schema, Creatio generates the resulting set of resources up to the level of the specified package.

The **resulting resources** for the `ChildSchema` schema up to the level of the `BottomPackage` package in design-time mode are as follows:

- `BaseResource: BaseValue`

- `ChildResource: ReplacedChildValue2`
- `ChildResource1: Value1`
- `ChildResource2: Value2`

The `ChildResource` value is changed to `ReplacedChildValue2` because the original resource was replaced in packages that are one level below in the hierarchy (`Level 2`). The position and name of the package are considered. Package that has a higher position value is prioritized. Packages that have the same position value are sorted in alphabetical order.

Run-time mode

The **purpose** of run-time mode is to display localizable resources in Creatio sections, excluding Designers and Wizards. **Unlike** run-time mode, design-time mode includes package resources that are not part of the hierarchy in the resulting resource index when requesting a schema.

The **resulting resources** for the `ChildSchema` schema in run-time mode are as follows:

- `BaseResource: ReplacedBaseValue2`
- `Resource1: Value1`
- `Resource2: Value2`
- `ChildResource: ReplacedChildValue2`
- `ChildResource1: Value1`
- `ChildResource2: Value2`

Display methods of localizable resources

Creatio implements the following **display methods of localizable resources**:

- if a **primary language is active** in the user profile, Creatio displays the localizable resource value in the primary language
- if an **additional language is activated** in the user profile and the **localizable resource value in this language exists**, Creatio displays the localizable resource value in the additional language.
- if an **additional language is activated** in the user profile and the **localizable resource value in this language does not exist**, Creatio displays the localizable resource value in the primary language

The following **classes** implement the mechanism that displays localizable resources:

- `Terrasoft.Common.LocalizableString`. Manages localizable strings.
- `Terrasoft.Common.LocalizableImage`. Manages localizable images.

To retrieve the localizable resource values, use the following **properties and methods**:

- The `Value` property. Returns the localizable object value for the current language. If Creatio cannot find the localizable object value for the current language, the value for the primary language is returned.
- The `HasValue` property. Returns a flag that specifies whether the localizable object has a value for the current language. If Creatio cannot find the localizable object value for the current language, the value for the primary

language is returned.

- The `GetCultureValue()` method. Returns the localizable object value for the current language. If Creatio cannot find the localizable object value for the current language, the value for the primary language is returned.
- The `HasCultureValue()` method. Returns a flag that specifies whether the localizable object has a value for the requested language regardless of the primary language.

View the description of the `Terrasoft.Common.LocalizableString` class in the .NET class library. View The description of the `Terrasoft.Common.LocalizableImage` class in the .NET class library.

Store localizable resources

The storing features depend on the localizable resource type.

Store simple localizable resources

Creatio can store simple localizable resources in the following **repositories**:

- **Database** Stores resources required for Creatio operation. The main repository for localizable resources.
- **SVN repository** Stores resources that must be installed into Creatio or transferred among .development environments. Pre-export the localizable resources to the SVN repository.

Store simple localizable resources in the database

The `[SysLocalizableValue]` database table stores localizable resources in text format as "key-value" pairs. Each record is bound to a package and base schema ID.

View the primary columns of the `[SysLocalizableValue]` database table below.

Primary columns of the `[SysLocalizableValue]` database table

Column	Description
`[Id]`	Record ID.
`[SysPackageId]`	Package ID.
`[SysSchemaId]`	Base schema ID. Populated only for configuration resources.
`[ResourceManager]`	Resource manager name. Populated only for core resources.
`[SysCultureId]`	Language culture ID.
`[ResourceType]`	Resource type.
`[IsChanged]`	Flag that specifies whether the user changed the localizable resource.
`[Key]`	Localizable resource key.
`[Value]`	String resource value.
`[ImageData]`	Graphic resource value.

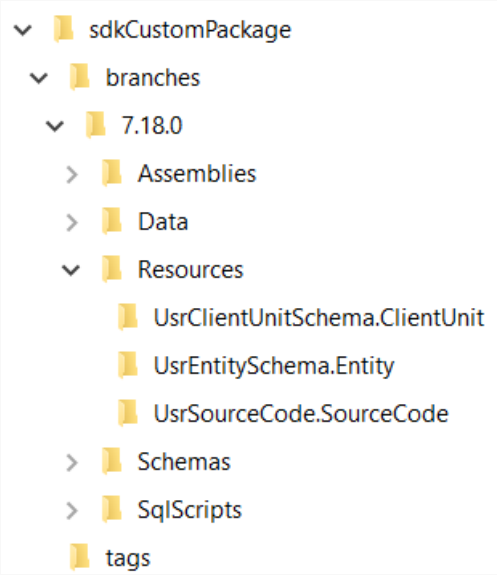
If an additional language culture is activated in the user profile, Creatio adds a corresponding record to the `[SysLocalizableValue]` database table when you add a localizable resource. Users that have other language cultures can access the added localizable resource using the **mechanism that duplicates the resource to the primary language culture**. The purpose of the mechanism is to create a similar localizable resource record that has a link to the primary language culture. If the value of the current localizable resource is not specified for other language cultures, the value of the primary language culture is displayed.

The `[SysPackageResourceChecksum]` database table stores a **checksum** for each set of schema resources in the package – schema – culture chain. The checksum lets you identify resource changes when the package is updated. The checksum separates schemas from resources, which lets you create translation packages.

Store simple localizable resources in the SVN repository

The `Resources` directory of the SVN repository stores localizable package resources. Use the directory to create a translation package. A **translation package** is a package that contains only localizable resources and does not contain configuration element schemas. A translation package can contain localizable resources for a schema in a different package.

To store localizable schema resources that have the same name but different managers, for example, `Entity` and `ClientUnit`, the schema Manager name without the `SchemaManager` prefix are added to the localizable resources names of the package. Creatio stores the localizable resources in exported schemas as *.xml files.



Store bound localizable resources

The repositories for bound localizable resources and simple localizable resources are similar.

Store bound localizable resources in the database

The `[SysPackageDataLcz]` database table stores bound localizable resources.

View the primary columns of the `[SysPackageDataLcz]` database table below.

Primary columns of the `[SysPackageDataLcz]` database table

Column	Description
`[Id]`	Record ID.
`[SysPackageSchemaDataId]`	Binding ID from the `[SysPackageSchemaData]` database table.
`[SysCultureId]`	Language culture ID.
`[Data]`	Bound localizable data.

The storage of bound localizable resources has the following **special features**:

- If a schema **contains bound localizable resources**, Creatio saves data to the `[SysPackageDataLcz]` database table.
- If a schema **does not contain bound localizable resources**, Creatio saves data to the `[SysPackageSchemaData]` database table.

A record of the `[SysPackageDataLcz]` database table contains the following **data**:

- link to the corresponding record ID from the `[SysPackageSchemaData]` database table

- link to the corresponding language culture ID from the `[SysCulture]` database table

For example, if the Creatio instance uses English and Ukrainian language cultures, each record of the `[SysPackageSchemaData]` database table corresponds to two records of the `[SysPackageDataLcz]` database table.

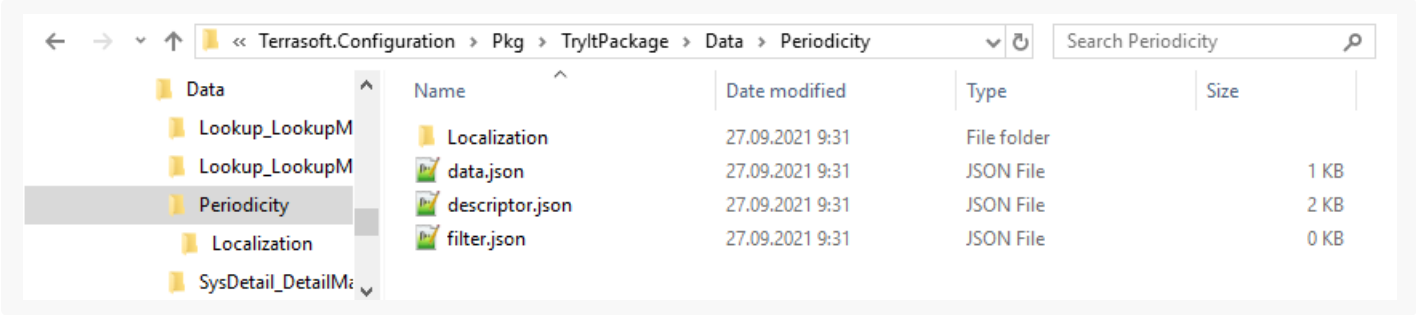
Store bound localizable resources in the SVN repository

The `Data` directory of the SVN repository stores bound localizable resources.

`Data` **directory structure**:

- The `data.json` file. Non-localizable resources.
- The `Localization` directory. Bound localizable resources. The directory contains the corresponding files for the language cultures. The file names follow this pattern: `data.LanguageCultureCode.json`, for example, `data.en-US.json`.

View an example that displays the structure of the stored bound localizable resources below. The example uses the `Periodicity` schema in the `TryItPackage` package.



Install bound localizable resources

The installation of bound localizable resources has the following **special features**:

- If the schema **does not contain bound localizable resources**, Creatio installs the resources into the primary table of the `Entity` object.
- If the schema **contains bound localizable resources** (i. e., the `[SysPackageDataLcz]` database table contains the appropriate records), Creatio installs the resources into the primary database table of the corresponding schema and into the localization table.

The template for the localization table name is as follows: `[SysPrimaryTableNameLcz]`.

For example, the installation order of the bound localizable resources for the `ContactType` scheme is as follows:

- non-localizable data is installed into the `[ContactType]` database table
- localizable data is installed into the `[ContactType]` and `[SysContactTypeLcz]` database tables

Note. Creatio does not add another `Sys` prefix for the localization table that matches the system table (i. e., the name starts with the `Sys` prefix). For example, the localization table for the `[SysTestSchema]` database table is named `[SysTestSchemaLcz]`, not `[SysSysTestSchemaLcz]`.

Operations with localizable resources

Advanced

To execute operations with localizable resources, you can use the following **tools**:

- Creatio IDE
- database
- SVN version control system
- file system

Execute operations with localizable resources using Creatio IDE

You can execute the following **operations with localizable resources** using Creatio IDE:

- add a localizable column
- add a localizable string

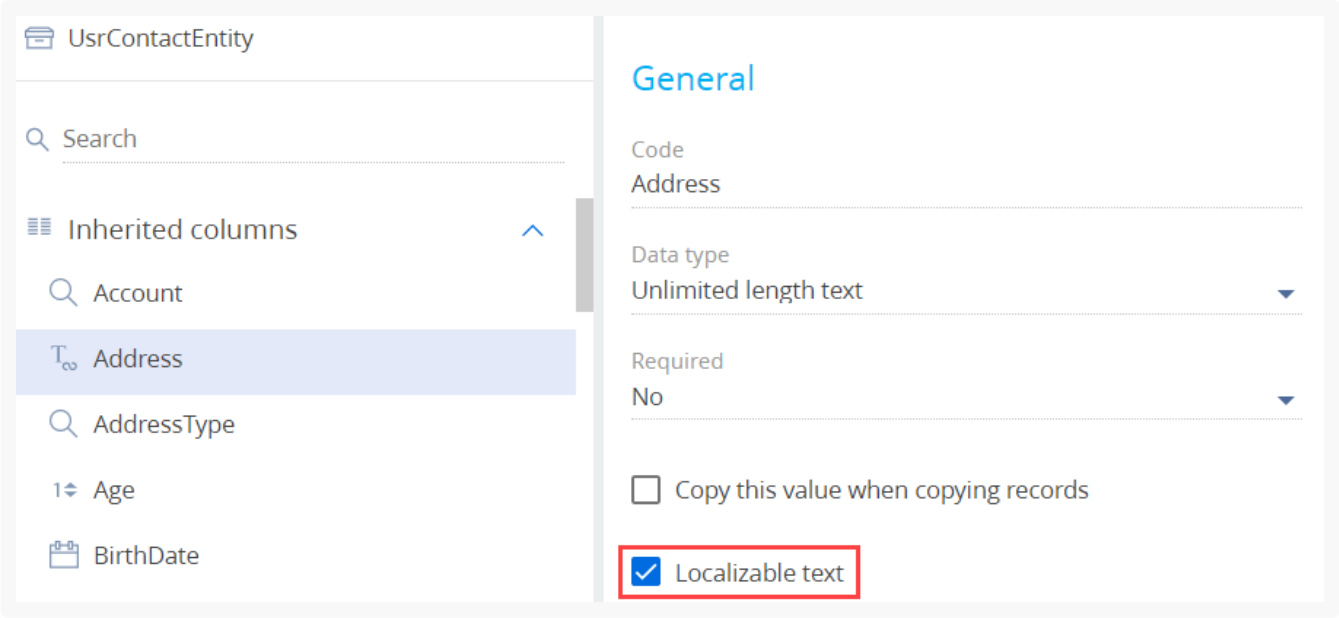
Add a localizable column

Localizable columns let you display object data in multiple languages in Creatio UI. The localizable column value depends on the language selected in the user profile. You can configure a localizable column in the Object Designer. The database stores the column value.

To **add a localizable column**:

1. Add the column to localize.
   a. Open the [*Configuration*] section and select a custom package to add the schema.
   b. Click [*Add*] → [*Object*] or [*Replacing object*] on the section list toolbar.

   c. Fill out the schema properties in the Object Designer.
   d. Add the columns to localize if needed or select an existing object column.
   e. Select the [*Localizable text*] checkbox in the [*General*] property block of the corresponding column.



Note. You can localize the following column types:

- [*Text (50 characters)*]
- [*Text (250 characters)*]
- [*Text (500 characters)*]
- [*Unlimited length text*]

j. Click [*Save*] then [*Publish*] on the Object Designer toolbar.

2. Translate the localizable column value. To do this, follow the instruction in the user documentation: Localize UI via the [*Translation*] section.

As a result, the column will contain values in different languages. Creatio UI uses a value that depends on the language selected in the user profile.

Add a localizable string

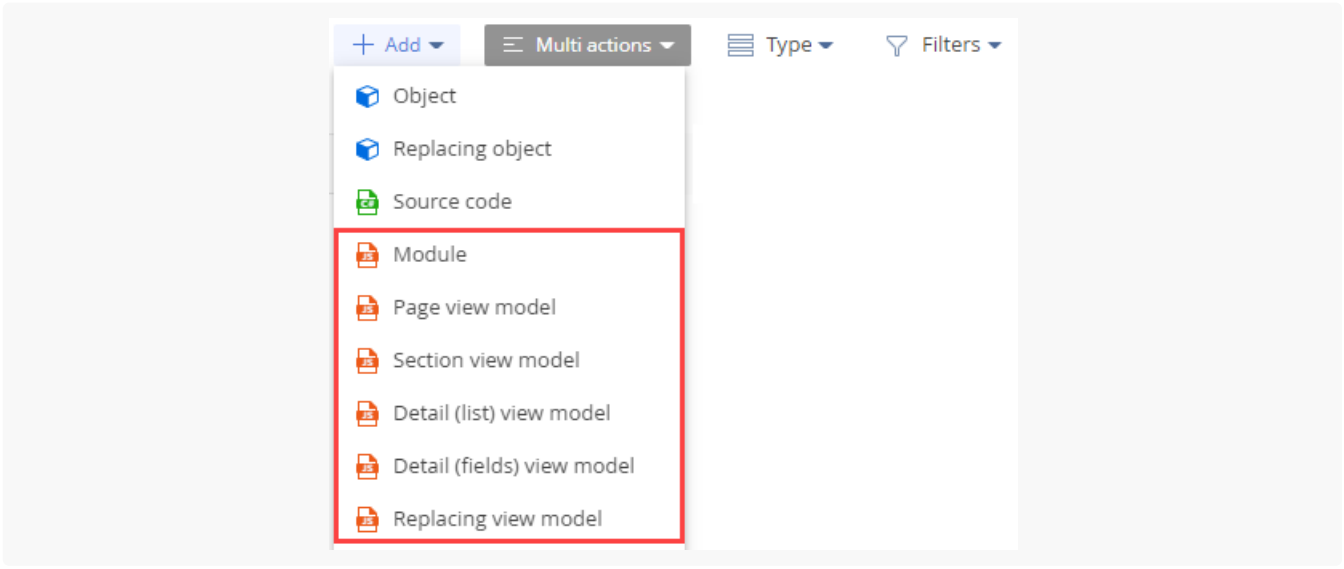
The localizable string lets you display module data in multiple languages in Creatio UI. The localizable string value depends on the language selected in the user profile. You can configure a localizable string in the Module Designer and Source Code Designer. The database stores the string value.

To **add a localizable string**:

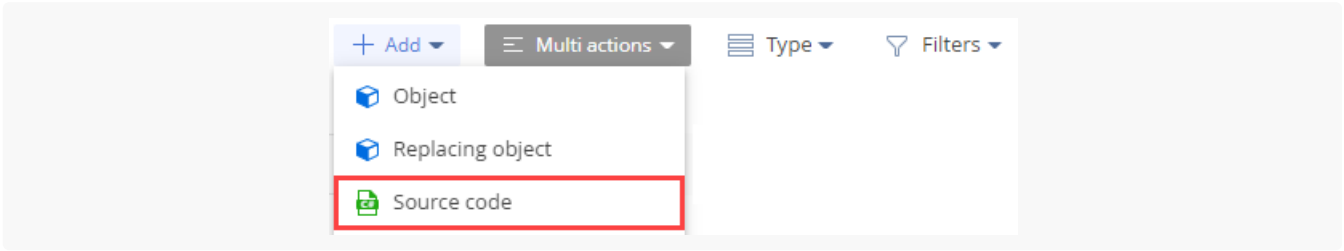
1. Open the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] on the section list toolbar and select the schema type.

- Select one of the following options if you want to **localize the Client Module string**:
  - [*Module*]
  - [*Page view model*]
  - [*Section view model*]

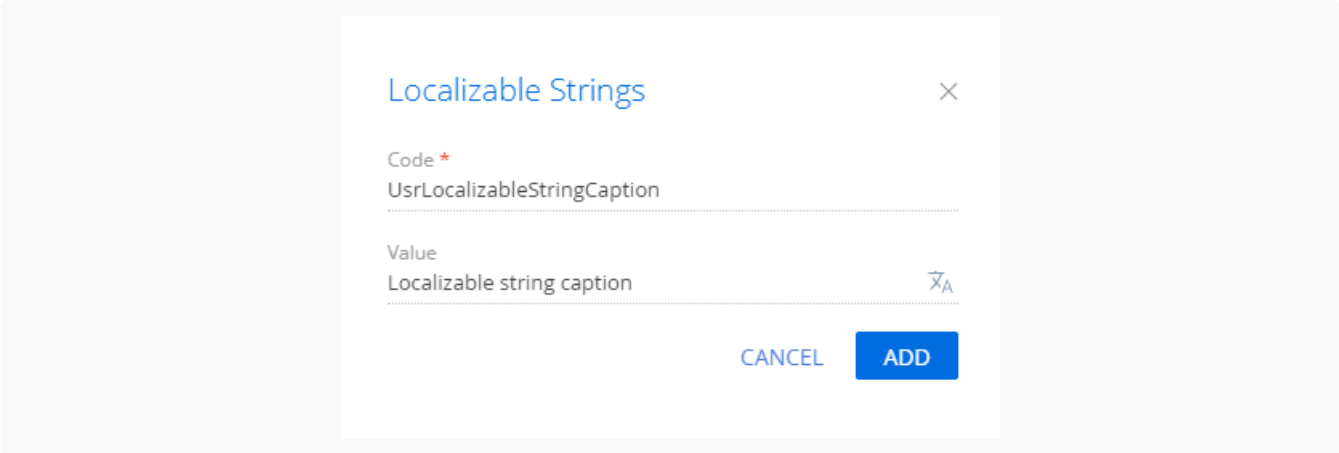
- [*Detail (list) view model*]
- [*Detail (fields) view model*]
- [*Replacing view model*]



- If you want to **localize the Source Code string**, select [*Source code*].



3. Fill out the schema properties in the Designer.
4. Add the string to localize.
   a. Click the + button in the context menu of the [*Localizable strings*] node.
   b. Fill out the localizable string properties:
      - Enter the localizable string name in the [*Code*] property. Required. Must start with the prefix specified in the [*Prefix for object name*] ([*SchemaNamePrefix*] code) system setting, `Usr` by default.
      - Enter the localizable string value in the [*Value*] property. By default, Creatio saves the value for the language selected in the user profile. Click the button to set the localizable string values in different languages.



e. Click [*Add*] to add a localizable string.

5. Click [*Save*] then [*Publish*] on the Designer toolbar.

As a result, the string will contain values in different languages. Creatio UI uses a value that depends on the language selected in the user profile.

Execute operations with localizable resources using the database

You can execute the following **operations with localizable resources** using the database:

- retrieve localizable resources from the database
- update localizable resources in the database
- save localizable resources to the database
- disable localizable resources in the database

Retrieve localizable resources from the database

The following **classes** implement the mechanism that retrieves localizable resources from the database:

- `Terrasoft.Core.Entities.EntitySchemaQuery`. Retrieves data from the database via the ORM model. Supports multilingual data by default.
- `Terrasoft.Core.Entities.Entity`. Works with the database entity.

Learn more about retrieving data from the database using the `Terrasoft.Core.Entities.EntitySchemaQuery` and `Terrasoft.Core.Entities.Entity` classes in a separate article: Data access through ORM.

The **rules to generate a selection** of multilingual data are as follows:

- if a **primary language is active** in the user profile, the data selection is generated from the primary database table.
- if an **additional language is active** in the user profile and the **value of the localizable resource exists** in the `[SysLocalizableValue]` database table, the data selection is generated from the `[SysLocalizableValue]` database table.
- if an **additional language is active** in the user profile and the **value of the localizable resource does**

not exist in the `[SysLocalizableValue]` database table, the data selection is generated from the primary database table.

To generate a data selection, you can use `views` that let you retrieve data from localizable columns. Configure localizable views to generate a selection of localizable data using a view.

To **configure localizable views**:

1. Create an object schema for the view. Use the following schema name template: `UsrVwObjectSchemaName`.

   The schema name must contain the following required elements:

   - First prefix specified in the [*Prefix for object name*] ([*SchemaNamePrefix*] code) system setting, `Usr` by default.
   - Second `Vw` prefix (short for `View`) that indicates that the schema is a view in the database.

2. Configure a localizable column. Learn more about configuring localizable columns in a separate guide: Add a localizable column.
3. Add a localization view to the database.

Update localizable resources in the database

Update localizable resources when the localizable resource value in the `[SysLocalizableValue]` database table is changed.

To **update the localizable resources in the database** for the corresponding schema, modify the `[IsChanged]` column value of the `[SysPackageResourceChecksum]` database table using an SQL query. Otherwise, a localizable resource conflict will occur when the package is updated in Creatio. The conflict cannot be detected, which will cause the localizable resource value to be lost.

Attention. Do not add data to the `[SysLocalizableValue]` database table using a direct SQL query because this does not add information about the added localizable resources to the metadata of the corresponding schema. To **add localizable resources**, use Creatio IDE, SVN version control system, or file system.

Save localizable resources to the database

To **save localizable resources**, use the `Entity.SetColumnValue()` method. The method can accept parameters of the `string` and `LocalizableString` types.

Save localizable resources using parameters of the `string` type

The **special features of saving localizable resources** using parameters of the `string` type are as follows:

- if a user that has an additional language **adds a record**, Creatio saves data to the primary table of the `Entity` object and localization table of the `Entity` object
- if a user that has an additional language **modifies a record**, Creatio saves data to the localization table of the `Entity` object

- if a user that has a primary language **adds or modifies a record**, Creatio saves data to the primary table of the `Entity` object

Creatio generates the localization table name based on the following template: `[SysPrimaryTableNameLcz]`.

View the example that saves localizable resources using a parameter of the `string` type for a user that has a primary language (German) below.

Example that saves localizable resources using a parameter of the `string` type

```
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];
EntitySchema schema = userConnection.EntitySchemaManager.FindInstanceByName("AccountType");
Entity entity = schema.CreateEntity(userConnection);

/* Set the default column values. */
entity.SetDefColumnValues();

/* Set the [Name] column value. */
entity.SetColumnValue("Name", "Neuer kunde");

/* Save. */
entity.Save();
var name = String.Format("Name: {0}", entity.GetTypedColumnValue<string>("Name"));
return name;
```

If the user that has the primary language (German) runs the code above, Creatio executes an SQL query to the primary `[AccountType]` database table. "`Neuer kunde`" is specified in the `@P2` parameter.

SQL query

```
exec sp_executesql N'
INSERT INTO [dbo].[AccountType]([Id], [Name], [CreatedOn], [CreatedById], [ModifiedOn], [Modifie
VALUES(@P1, @P2, @P3, @P4, @P5, @P6, @P7, @P8)',N'@P1 uniqueidentifier,@P2 nvarchar(12),@P3 date
```

View the example that saves localizable resources using a parameter of the `string` type for a user that has an additional language, for example, French, below.

Example that saves localizable resources using a parameter of the `string` type

```
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];
EntitySchema schema = userConnection.EntitySchemaManager.FindInstanceByName("AccountType");
Entity entity = schema.CreateEntity(userConnection);
entity.SetDefColumnValues();
entity.SetColumnValue("Name", "Nouveau Client");
entity.Save();
```

```
var name = String.Format("Name: {0}", entity.GetTypedColumnValue<string>("Name"));
return name;
```

If the user that has an additional language, for example, French, runs the code above, Creatio executes the following SQL queries:

1. SQL query to the primary `[AccountType]` database table. The query is similar to the query for the primary localization, but "Nouveau Client" is specified in the `@P2` parameter.

   SQL query

   ```
   exec sp_executesql N'
   INSERT INTO [dbo].[AccountType]([Id], [Name], [CreatedOn], [CreatedById], [ModifiedOn], [Modi
   VALUES(@P1, @P2, @P3, @P4, @P5, @P6, @P7, @P8)',N'@P1 uniqueidentifier,@P2 nvarchar(12),@P3 d
   ```

2. SQL query to the `[SysAccountTypeLcz]` localization table. "Nouveau Client" is specified in the `@P5` parameter.

   SQL query

   ```
   exec sp_executesql N'
   INSERT INTO [dbo].[SysAccountTypeLcz]([Id], [ModifiedOn], [RecordId], [SysCultureId], [Name])
   ```

Therefore, the primary `[AccountType]` table will contain a value that does not match the primary localization. To prevent that, save localizable resources using parameters of the `LocalizableString` type.

Save localizable resources using parameters of the `LocalizableString` type

View the example that saves localizable resources using a parameter of the `LocalizableString` type below.

Example that saves localizable resources using a parameter of the `LocalizableString` type

```
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];
EntitySchema schema = userConnection.EntitySchemaManager.FindInstanceByName("AccountType");
Entity entity = schema.CreateEntity(userConnection);
entity.SetDefColumnValues();

/* Create a localizable string that contains localized values for different language cultures. *
var localizableString = new LocalizableString();
localizableString.SetCultureValue(new CultureInfo("de-DE"), "Neuer Kunde de-DE");
localizableString.SetCultureValue(new CultureInfo("en-US"), "New Customer en-US");

/* Set the column value using the localizable string. */
entity.SetColumnValue("Name", localizableString);
entity.Save();
```

```
/* Display the result in the current language culture of the user. */
var name = String.Format("Name: {0}", entity.GetTypedColumnValue<string>("Name"));
return name;
```

Regardless of the language selected in the user profile, if the user runs the code above, Creatio executes the following SQL queries:

1. SQL query to the primary `[AccountType]` database table "`Neuer Kunde de-DE`" is specified in the `@P2` parameter.

   SQL query

   ```
   exec sp_executesql N'
   INSERT INTO [dbo].[AccountType]([Id], [Name], [CreatedOn], [CreatedById], [ModifiedOn], [Modi
   VALUES(@P1, @P2, @P3, @P4, @P5, @P6, @P7, @P8)',N'@P1 uniqueidentifier,@P2 nvarchar(18),@P3 d
   ```

2. SQL query to the `[SysAccountTypeLcz]` localization table. "`New Customer en-US`" is specified in the `@P5` parameter.

   SQL query

   ```
   exec sp_executesql N'
   INSERT INTO [dbo].[SysAccountTypeLcz]([Id], [ModifiedOn], [RecordId], [SysCultureId], [Name])
   VALUES(@P1, @P2, @P3, @P4, @P5)',N'@P1 uniqueidentifier,@P2 datetime2(7),@P3 uniqueidentifier
   ```

If the user that has an additional language executes the code above and the localizable string value does not exist for the primary language, Creatio adds the record that contains additional language value to the primary `[AccountType]` table.

Turn off localizable resources in the database

To **turn off localizable resources**, set the `UseLocalization` property of the `EntitySchemaQuery` class instance to `false`. Turning off localizable resources does not remove localizable resources from database tables and does not depend on the language in the user profile.

Example that turns off receiving localizable resources

```
/* The user connection. */
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];

/* Generate a query. */
var esqResult = new EntitySchemaQuery(userConnection.EntitySchemaManager, "City");

/* Add a column to the query. */
esqResult.AddColumn("Name");
```

```
/* Turn off the mechanism that selects localized data. */
esqResult.UseLocalization = false;

/* Execute a database query and retrieve the resulting object collection. */
var entities = esqResult.GetEntityCollection(userConnection);

/* Retrieve the query text. */
var s = esqResult.GetSelectQuery(userConnection).GetSqlText();

/* Return the result. */
return s;
```

Regardless of the language selected in the user profile, if the user runs the code above, Creatio executes an SQL query to the primary `[City]` database table.

SQL query

```
SELECT
    [City].[Name] [Name]
FROM
    [dbo].[City] [City] WITH(NOLOCK)
```

Execute operations with localizable resources using SVN version control system

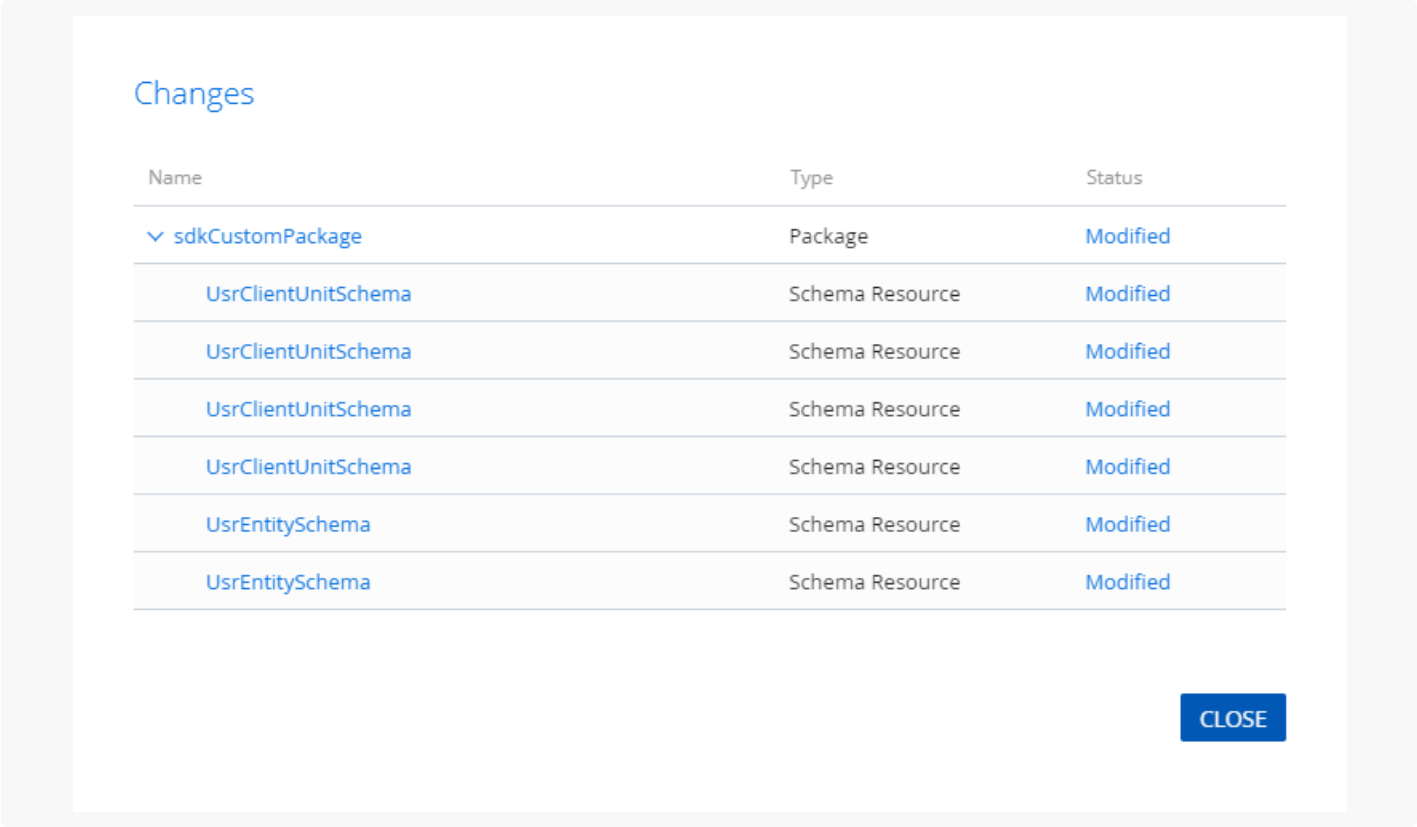
You can execute the following **operations with localizable resources** using SVN version control system:

- update localizable resources from the SVN repository
- commit localizable resources to the SVN repository

Learn more about working with SVN in a separate article: Version control in Subversion.

Update localizable resources from the SVN repository

To **update localizable resources from the SVN repository**, update the package from the SVN repository. Learn more about updating packages in a separate article: Version control in Creatio IDE.

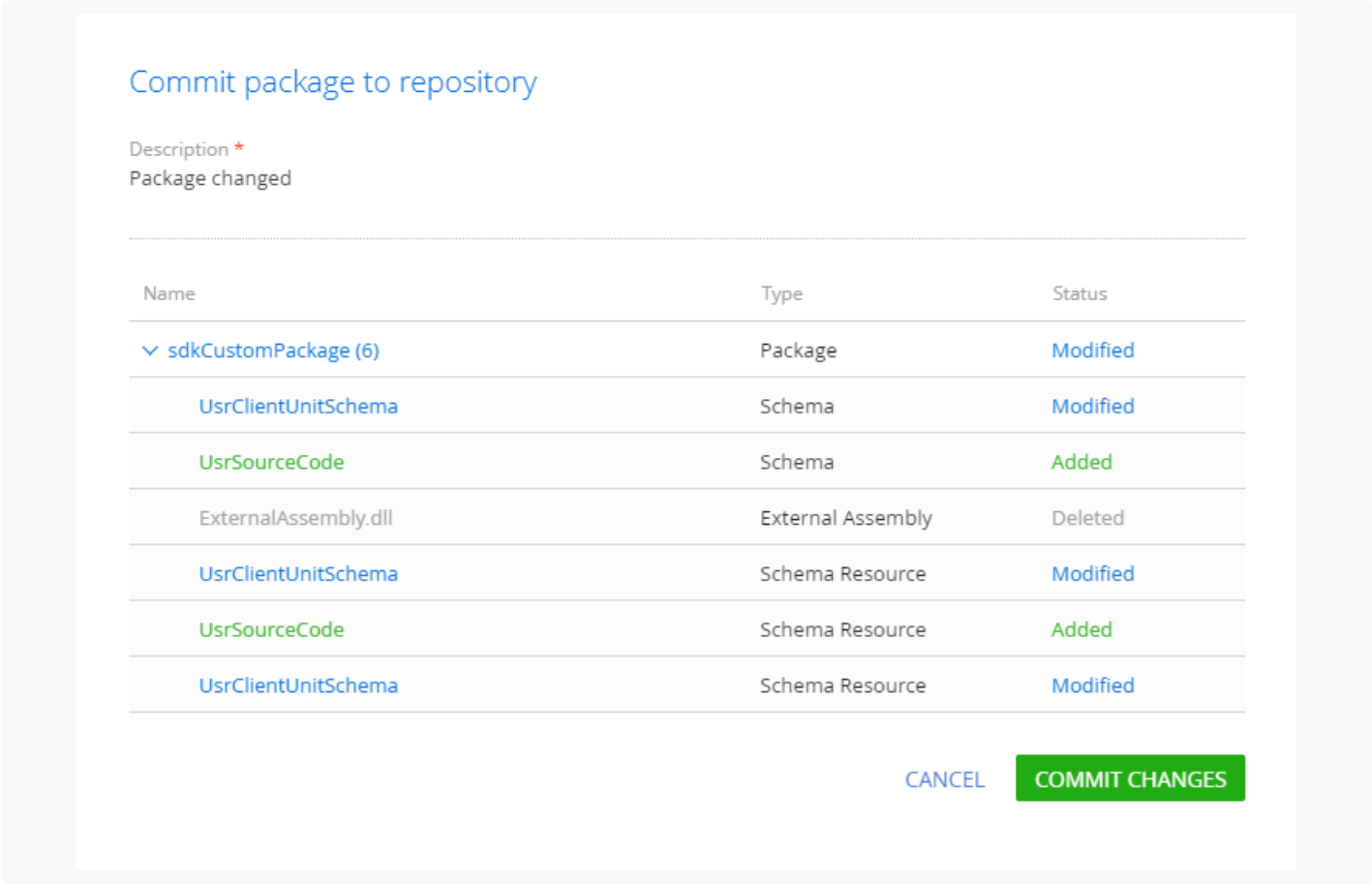


The available **localizable resource statuses** are as follows:

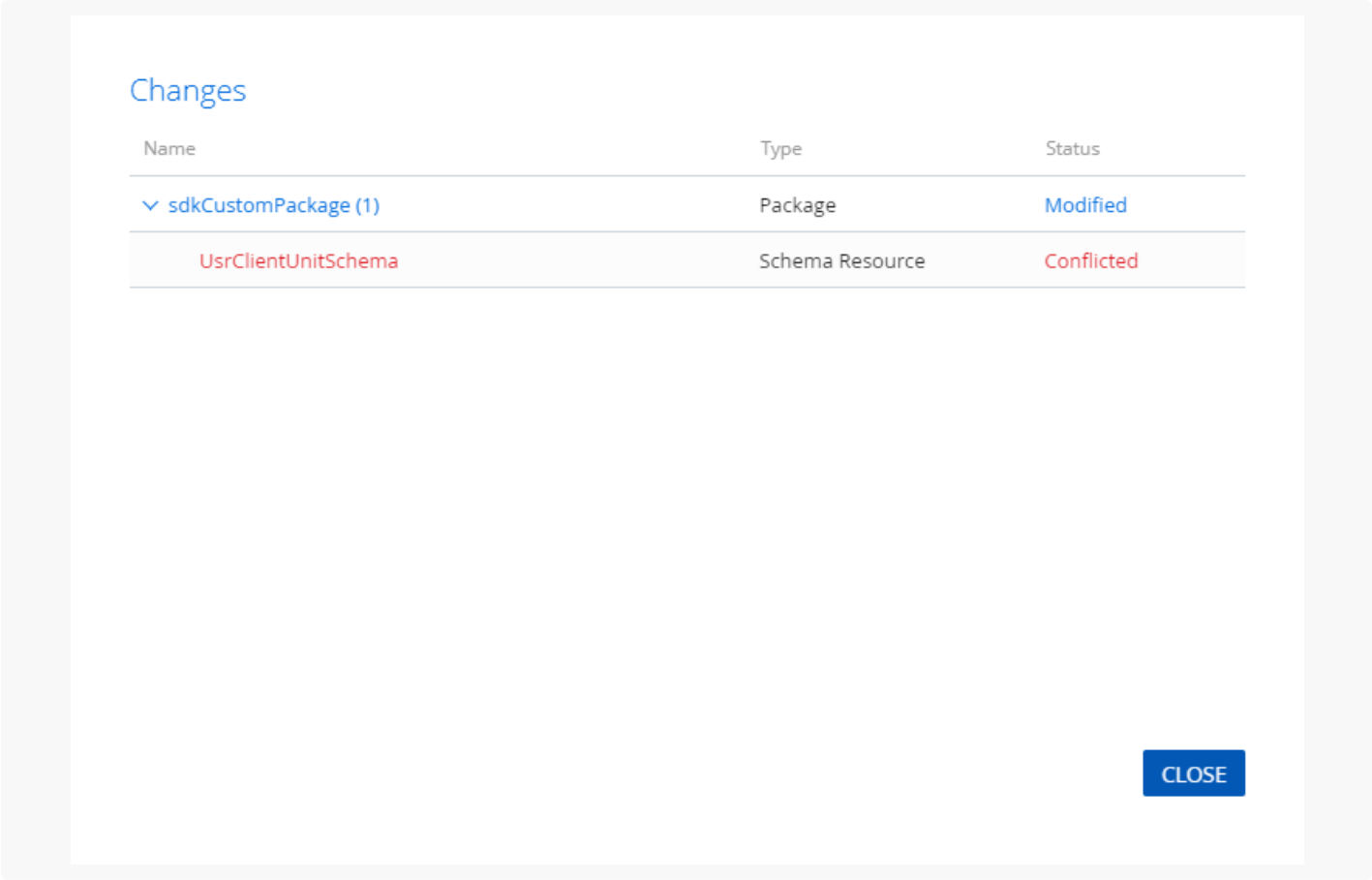
- [*modified*]. The localizable resource was changed.
- [*added*]. The localizable resource was added.
- [*deleted*]. The localizable resource was deleted.
- [*conflicted*]. Multiple developers worked with the localizable resource simultaneously. One developer committed localizable resource changes to the SVN repository.

Commit localizable resources to the SVN repository

To **commit localizable resources to the SVN repository**, commit the package to the SVN repository. Learn more about committing packages in a separate article: Version control in Creatio IDE.



You can lock a package in the SVN repository using Creatio IDE. Locking a package prevents the [*conflicted*] status of the localizable resource because multiple users cannot work with the package simultaneously. For example, a developer modifies the localizable package resources without pre-locking them. Another developer can modify the same localizable resources and commit them to the SVN repository of the current package. In this case, when you commit the package to SVN and update the package, Creatio IDE displays a list of localizable resources in the [*conflicted*] status. I. e., the version and content of the localizable resources changed by the first developer do not match those of the resources committed to the SVN repository. The localizable resource changes committed to the SVN repository will be lost when you commit changes. Resolve such conflicts manually using SVN clients, for example, TortoiseSVN.



Execute operations with localizable resources using the file system

You can modify localizable resources using the file system.

To **modify localizable resources from the file system**:

1. Configure Creatio to work in the file system. Learn more in a separate article: External IDEs.
2. Export localizable resources to the file system using the SVN client.
3. Modify the localizable resources.
4. Commit the changes to the SVN repository.

Attention. The localizable resource value corresponds to a single record from the `[SysLocalizableValue]` database table. This record contains links to the appropriate IDs (`[SysPackageId]` , `[SysSchemaId]` , `[SysCultureId]`) and key (`[Key]`). Therefore, if you commit a resource whose status is [*conflicted*], Creatio writes the last value to the table.

Retrieve localizable resources from the database

Advanced

Example 1

Example. Retrieve localizable resources for an arbitrary language culture using the `Terrasoft.Core.Entities.EntitySchemaQuery` class.

The `Terrasoft.Core.Entities.EntitySchemaQuery` class generates a data selection for an arbitrary language culture, i. e., a language culture that differs from the primary language culture in Creatio and additional language culture of the user.

To **retrieve localizable resources for an arbitrary language culture**:

1. Call the `SetLocalizationCultureId(Guid cultureId)` method in the `Terrasoft.Core.Entities.EntitySchemaQuery` class instance before you retrieve data.
2. Pass the language culture ID whose data you want to retrieve to the `SetLocalizationCultureId(Guid cultureId)` method.

Generate a data selection for an arbitrary language culture

```
/* The user connection. */
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];

/* Retrieve the ID of the needed language culture, for example, Italian. */
var sysCulture = new SysCulture(userConnection);
if (!sysCulture.FetchPrimaryInfoFromDB("Name", "it-IT")) {
    /* Error, no record found. */
    return "No culture found";
}
Guid italianCultureId = sysCulture.Id;

/* Generate a query. */
var esqResult = new EntitySchemaQuery(userConnection.EntitySchemaManager, "City");

/* Add a column to the query. */
esqResult.AddColumn("Name");

/* Set the needed localization. */
esqResult.SetLocalizationCultureId(italianCultureId);

/* Execute a database query and retrieve the resulting object collection. */
var entities = esqResult.GetEntityCollection(userConnection);

/* Retrieve the query text. */
var s = esqResult.GetSelectQuery(userConnection).GetSqlText();

/* Return the result. */
```

```
return s;
```

When you run the code above, Creatio executes the SQL query. The `@P1` parameter takes the value of the record ID (`[Id]`) stored in the `italianCultureId` variable.

SQL query

```
SELECT
    ISNULL([SysCityLcz].[Name], [City].[Name])[Name]
FROM
    [dbo].[City] [City] WITH(NOLOCK)
    LEFT OUTER JOIN [dbo].[SysCityLcz] [SysCityLcz] WITH(NOLOCK) ON ([SysCityLcz].[RecordId] = [
    AND [SysCityLcz].[SysCultureId] = @P1)
```

Example 2

Example. Retrieve the value of the [*Name*] localizable column in the [*AccountType*] object schema using the `FetchFromDB()` method of the `Terrasoft.Core.Entities.Entity` class.

The `FetchFromDB()` methods of the `Terrasoft.Core.Entities.Entity` class let you retrieve multilingual data. View the example that uses one of the `FetchFromDB()` method overloads for a user that has a primary (English) and an additional (for example, French) language culture below. For example, you can use these methods in custom web service methods.

Example that uses the `FetchFromDB()` method

```
/* The user connection. */
var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];

/* Retrieve the [Account type] schema. */
EntitySchema schema = userConnection.EntitySchemaManager.FindInstanceByName("AccountType");

/* Create the Entity (object) instance. */
Entity entity = schema.CreateEntity(userConnection);

/* The collection of column names to generate a selection. */
List<string> columnNamesToFetch = new List<string> {
    "Name",
    "Description"
};

/* Retrieve object data whose [Name] column value is "Customer." */
entity.FetchFromDB("Name", "Customer", columnNamesToFetch);
```

```
/* Generate and send a response. */
var name = String.Format("Name: {0}", entity.GetTypedColumnValue<string>("Name"));
return name;
```

If the user that has the primary language culture (English) runs the code above, Creatio executes an SQL query to the primary `[AccountType]` database table. The `@P1` parameter contains the "`Customer`" value that defines the corresponding record of the `[AccountType]` primary database table.

SQL query

```
exec sp_executesql N'
SELECT
    [AccountType].[Name] [Name],
    [AccountType].[Description] [Description]
FROM
    [dbo].[AccountType] [AccountType] WITH(NOLOCK)
WHERE
    [AccountType].[Name] = @P1',N'@P1 nvarchar(6)',@P1=N'Customer'
```

If the user that has an additional language culture (for example, French) runs the code above, Creatio executes an SQL query to the `[SysAccountTypeLcz]` localization table. The `@P1` parameter contains the "`Customer`" value that defines the corresponding record of the `[AccountType]` primary database table. The `@P2` parameter contains the ID value (`[SysCultureId]`) of the additional language culture from the `[SysCulture]` table. The value defines the corresponding record of the `[SysAcountTypeLcz]` localization table.

SQL query

```
exec sp_executesql N'
SELECT
    ISNULL([SysAccountTypeLcz].[Name], [AccountType].[Name]) [Name],
    ISNULL([SysAccountTypeLcz].[Description], [AccountType].[Description]) [Description]
FROM
    [dbo].[AccountType] [AccountType] WITH(NOLOCK)
    LEFT OUTER JOIN [dbo].[SysAccountTypeLcz] [SysAccountTypeLcz] WITH(NOLOCK) ON ([SysAccountTy
    AND [SysAccountTypeLcz].[SysCultureId] = @P2)
WHERE
    [AccountType].[Name] = @P1',N'@P1 nvarchar(6),@P2 uniqueidentifier',@P1=N'Customer',@P2='A54
```

As a result, `Name` variable will correspond to "`Customer`" for a user that has an English language culture and "`Client`" for a user that has a French language culture.

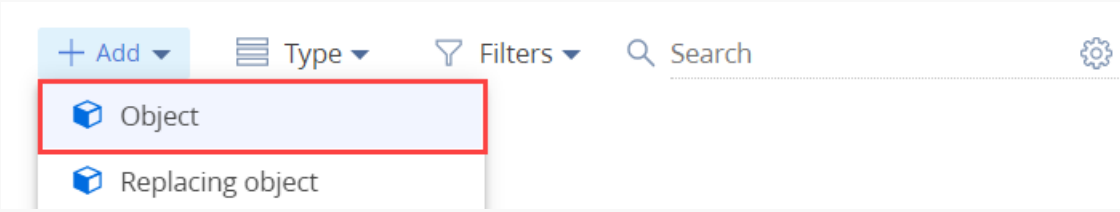
Add a localizable column

Medium

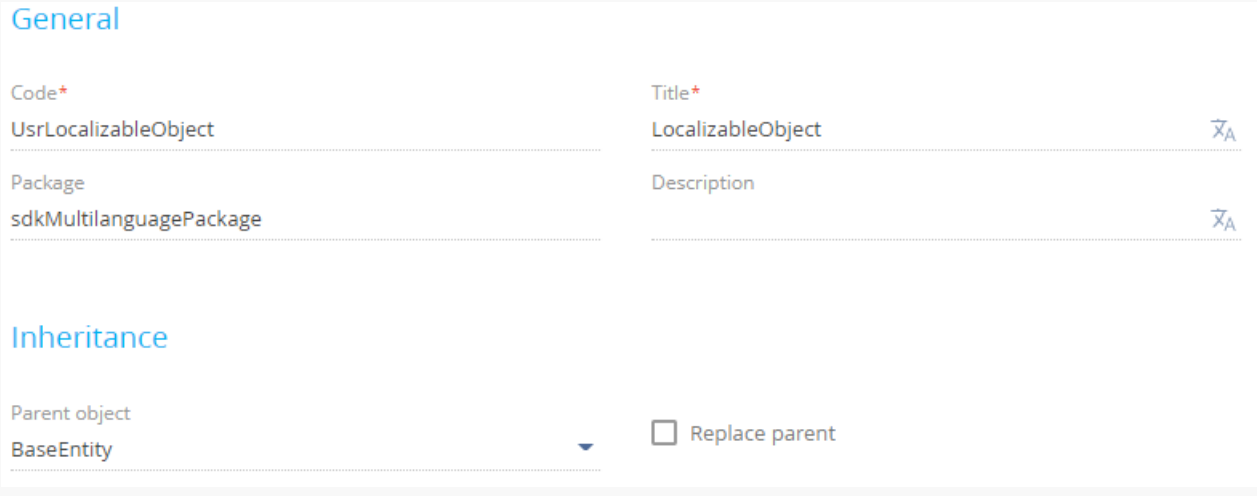
Example. Add a localizable column.

1. Create an object

1. Open the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Object*] in the section list toolbar.

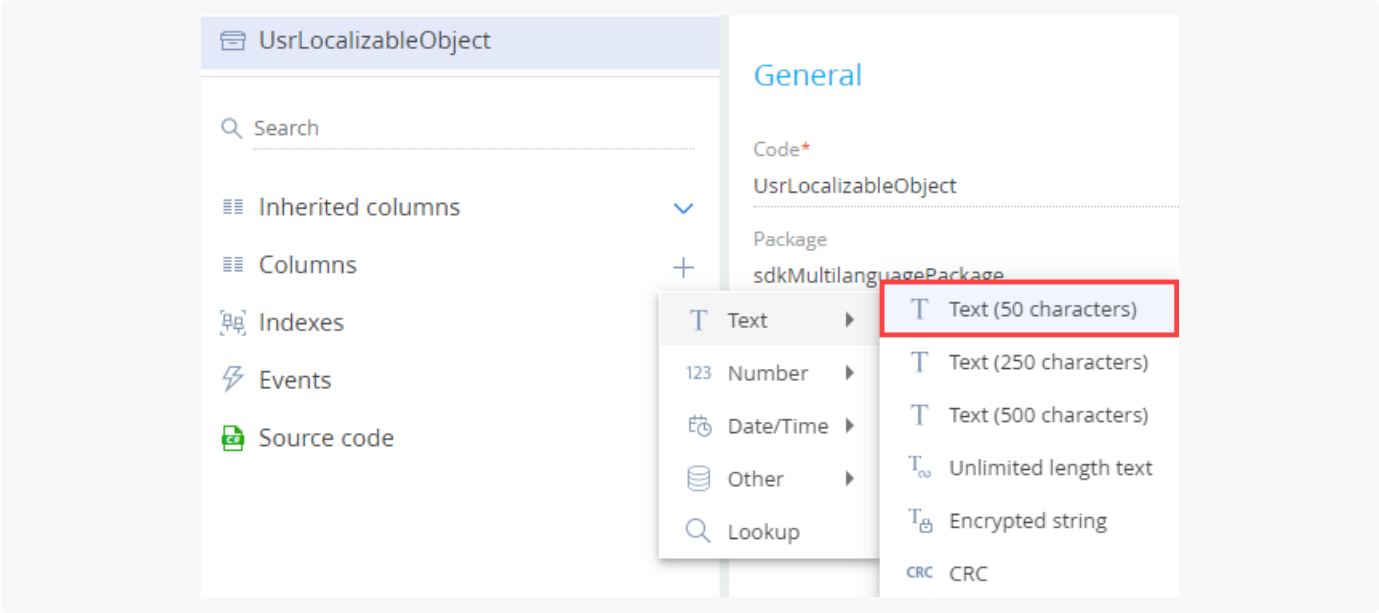


3. Fill out the schema properties in the Object Designer.
   - Set [*Code*] to "`UsrLocalizableObject`."
   - Set [*Title*] to "`LocalizableObject`."
   - Select "`BaseEntity`" in the [*Parent object*] property.



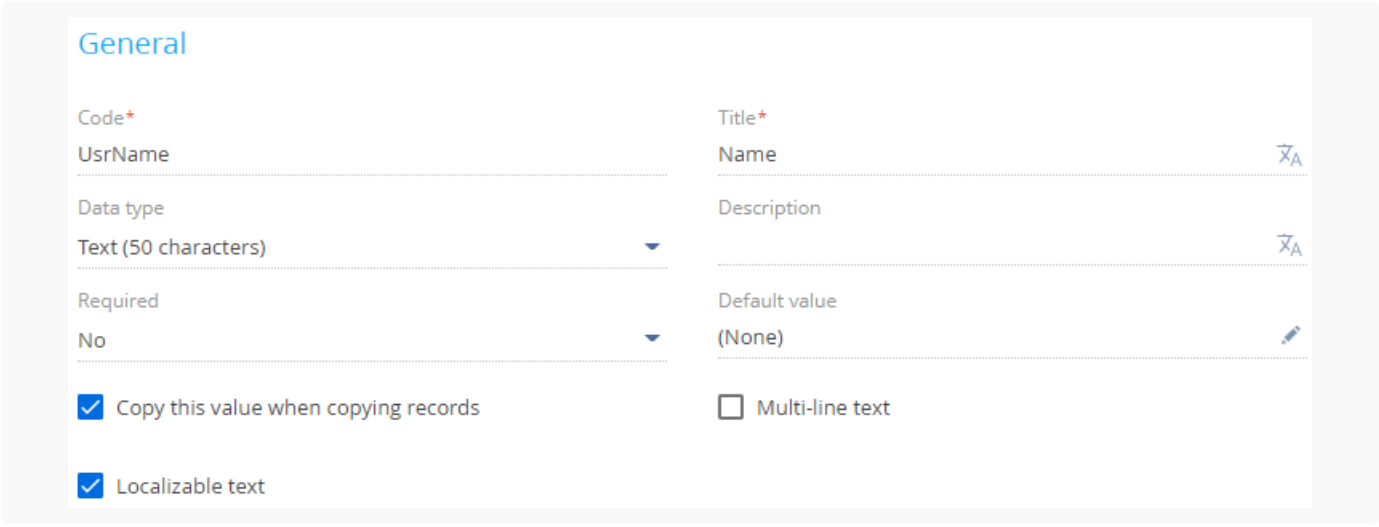
2. Add a localizable column

1. Click + in the context menu of the[*Columns*] node of the object structure. This opens a menu.
2. Hold the pointer over [*Text*] → click [*Text (50 characters)*] in the menu.



3. Fill out the column properties in the Object Designer.

   - Set [*Code*] to "`UsrName`."
   - Set [*Title*] to "`Name`."
   - Select the [*Localizable text*] checkbox.



4. Click [*Save*] and then [*Publish*] on the Object Designer toolbar.

Outcome of the example

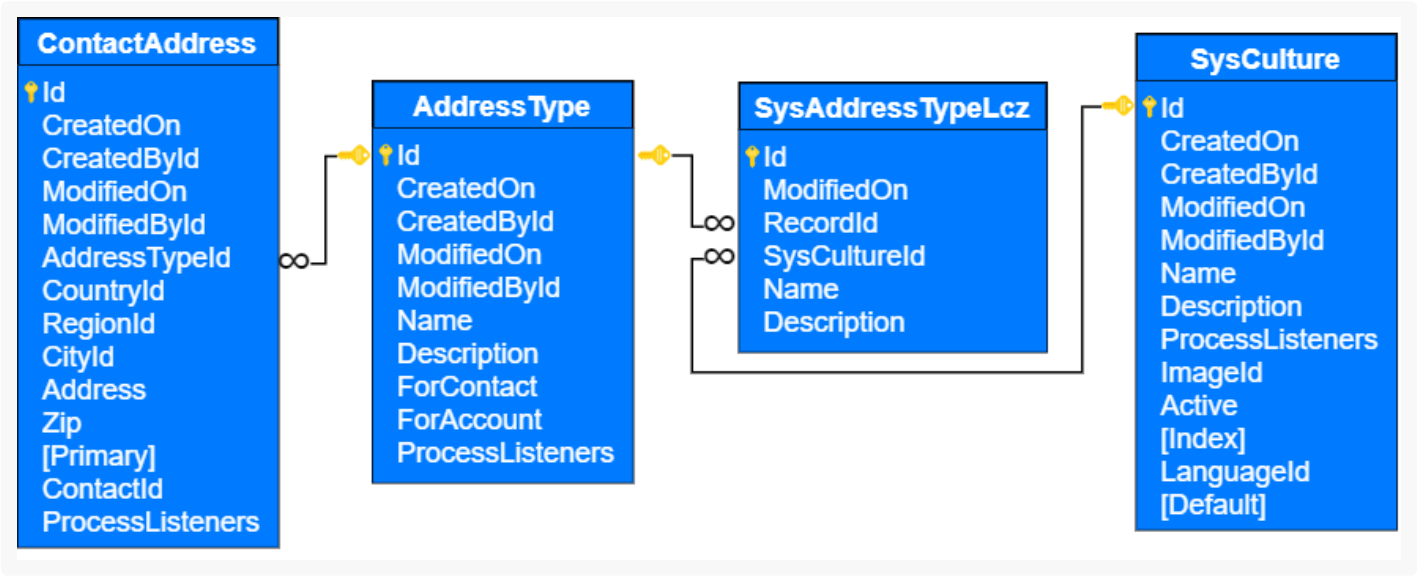
As a result, the `[SysUsrLocalizableObjectLcz]` localization table will be created in the database. The table will store localized data for all localizable columns of the object.

Localize the view in the database

Advanced

Example. Create a view that generates a collection. The collection must comprise localizable values of contact addresses (the `[Name]` column of the `[AddressType]` database table) and values of contact addresses (the `[Address]` column of the `[ContactAddress]` database table).

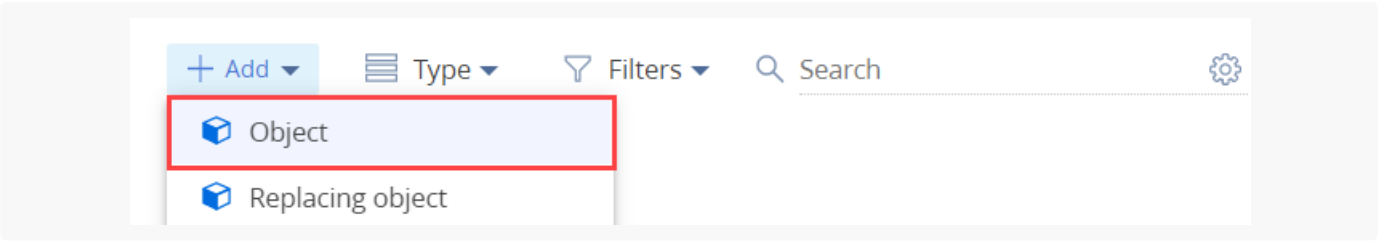
Creatio includes the [*ContactAddress*] object schema. The schema column links to the [*AddressType*] lookup that contains the [*Name*] localizable column. View the table structure and relationships in the figure below.



- The `[ContactAddress]` database table contains the index of contact address values. Linked to the `[AddressType]` table via the `[AddressTypeId]` column.
- The `[AddressType]` database table contains the index of contact address types. The `[Name]` table column contains the index of address type values in the primary language. The `[SysAddressTypeLcz]` table contains values in additional languages.
- The `[SysAddressTypeLcz]` database system table contains the index of localizable values of contact address types. Generated automatically. Linked to the `[AddressType]` table via the `[RecordId]` column and to the `[SysCulture]` table via the `[SysCultureId]` column. The `[Name]` table column contains the index of localizable values of contact address types for the language culture that is specified in the `[SysCultureId]` column of the current table.
- The `[SysCulture]` system database table contains the index of language cultures.

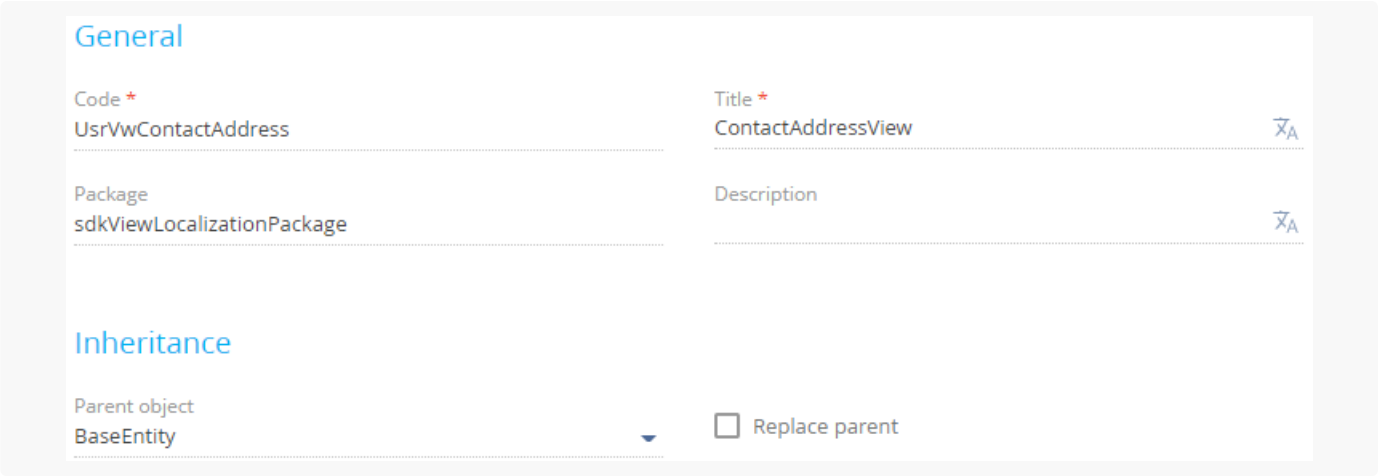
1. Create a view object schema

1. Open the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Object*] in the section list toolbar.



3. Fill out the schema properties in the Object Designer.

   - Set [*Code*] to "`UsrVwContactAddress`."
   - Set [*Title*] to "`ContactAddressView`."
   - Select "`BaseEntity`" in the [*Parent object*] property.



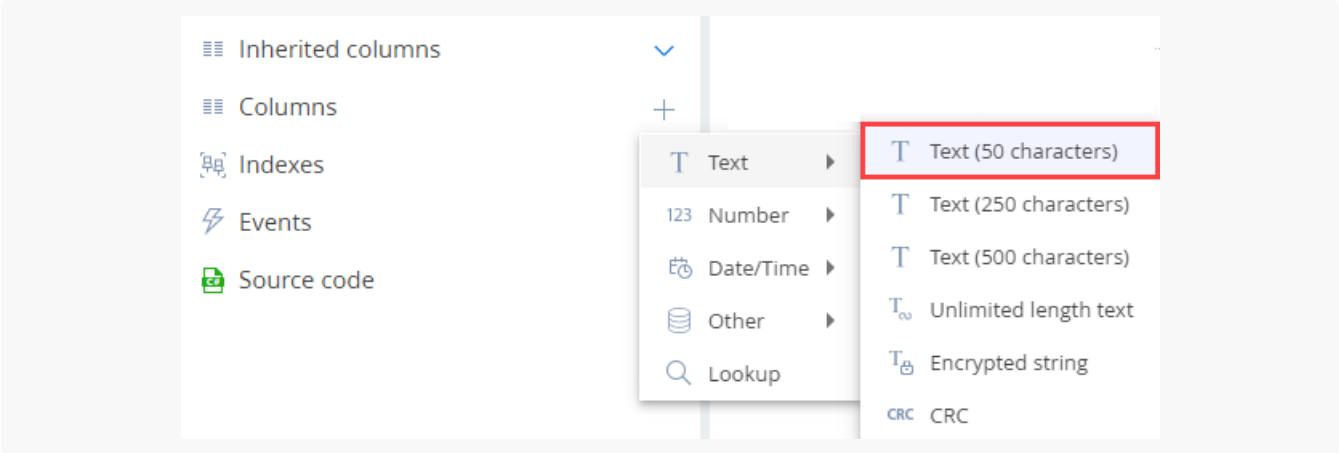
4. Select the [*Represent Structure of Database View*] checkbox in the [*Behavior*] property block.

2. Add columns

1. Add a column that contains the index of contact address values in the primary language.

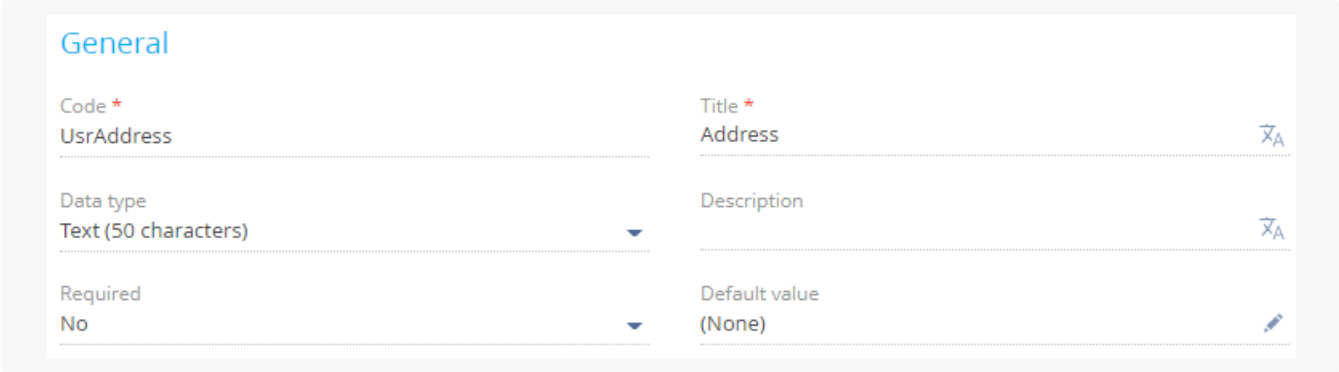
   a. Click + next to the [*Columns*] node of the object structure. This opens a menu.

   b. Hold the pointer over [*Text*] → click [*Text (50 characters)*] in the menu.



c. Fill out the column properties in the Object Designer.

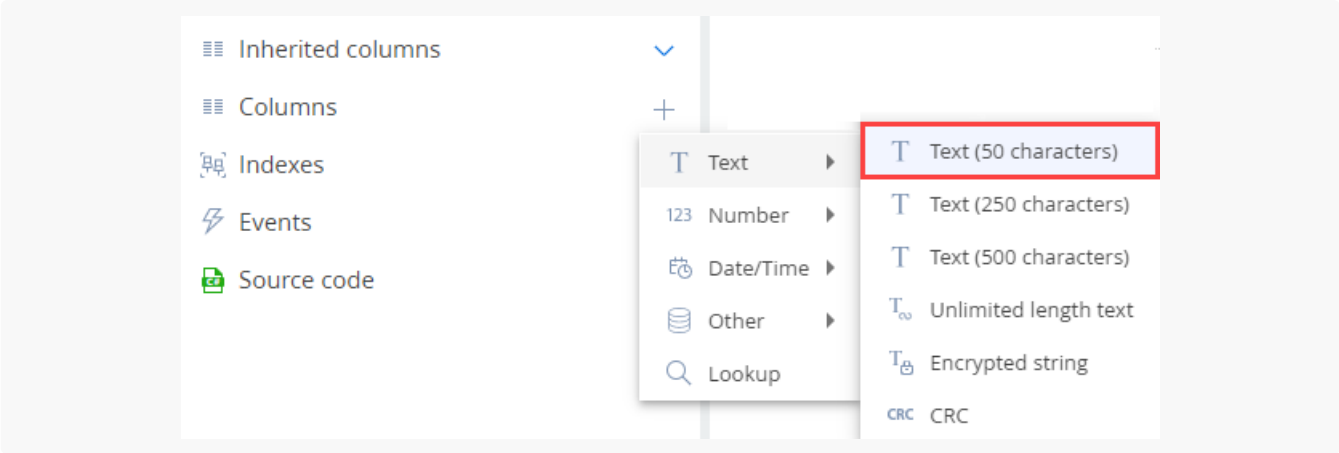
- Set [*Code*] to "`UsrAddress`."
- Set [*Title*] to "`Address`."



2. Add a column that contains the index of contact address types in an additional language.

a. Click + next to the [*Columns*] node of the object structure. This opens a menu.

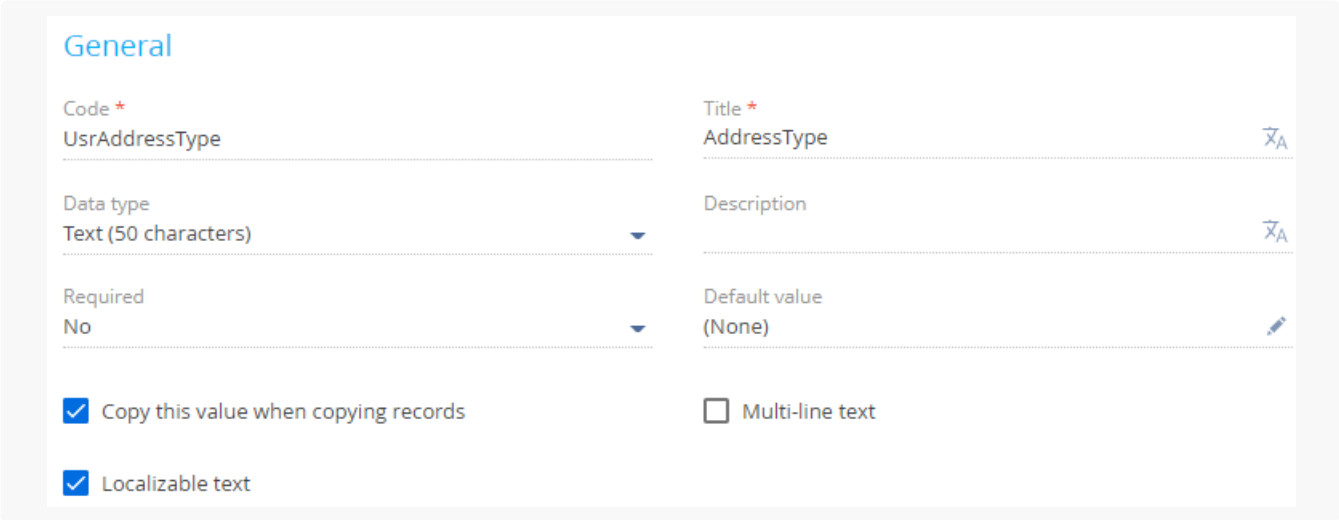
b. Hold the pointer over [*Text*] → click [*Text (50 characters)*] in the menu.



c. Fill out the column properties in the Object Designer.

- Set [*Code*] to "`UsrAddressType`."

- Set [*Title*] to "AddressType."
- Select the [*Localizable text*] checkbox.



g. Click [*Save*] then [*Publish*] on the Object Designer toolbar.

3. Create views in the database

1. Create the `[UsrVwContactAddress]` view in the database. Execute the following SQL query to do this.

SQL query

```
-- The view name must match the table name.
CREATE VIEW dbo.UsrVwContactAddress
AS
SELECT
    ContactAddress.Id,
    -- View columns must match the schema column names.
    ContactAddress.Address AS UsrAddress,
    AddressType.Name AS UsrAddressType
FROM ContactAddress
INNER JOIN AddressType ON ContactAddress.AddressTypeId = AddressType.Id;
```

2. Create the `[SysUsrVwContactAddressLcz]` localizable view in the database. Execute the following SQL query to do this.

SQL query

```
-- The view name must match the localizable table name.
CREATE VIEW dbo.SysUsrVwContactAddressLcz
AS
SELECT
    SysAddressTypeLcz.Id,
```

```
    ContactAddress.id AS RecordId,
    SysAddressTypeLcz.SysCultureId,
    -- View columns must match the schema column names.
    SysAddressTypeLcz.Name AS UsrAddressType
FROM ContactAddress
INNER JOIN AddressType ON ContactAddress.AddressTypeId = AddressType.Id
INNER JOIN SysAddressTypeLcz ON AddressType.Id = SysAddressTypeLcz.RecordId;
```

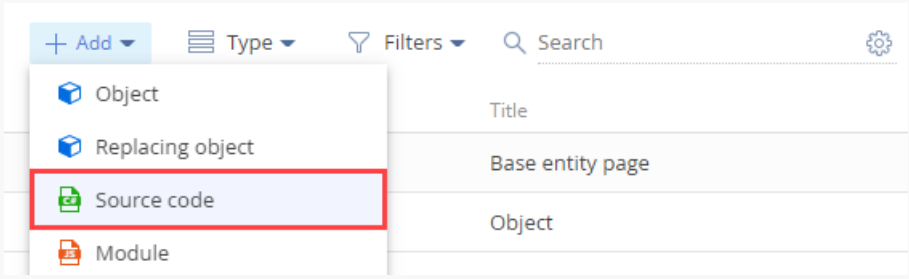
As a result, when Creatio retrieves data from the `[UsrAddressType]` column of the `[UsrVwContactAddress]` view using `EntitySchemaQuery`, the correct values for different languages will be displayed.

Outcome of the example

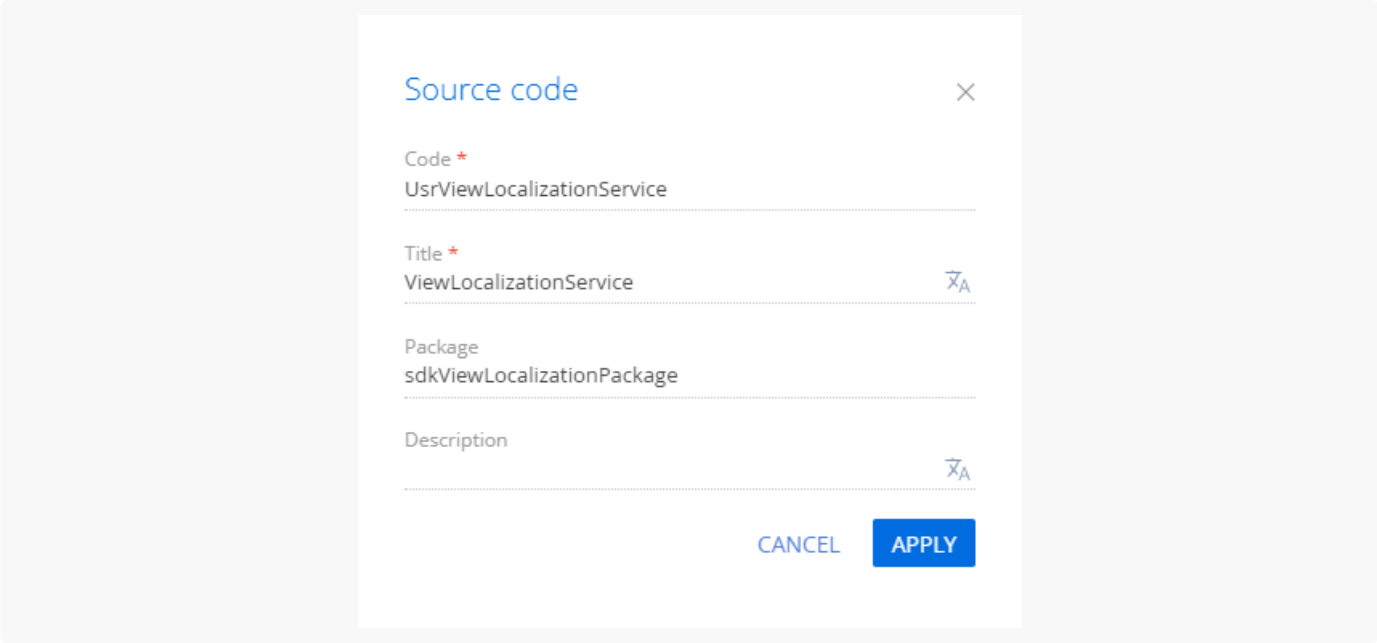
Create a custom web service that uses cookie-based authentication to verify the outcome of the example.

1. Create a [*Source code*] schema

1. Open the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Source code*] on the section list toolbar.



3. Fill out the schema properties in the Schema Designer.
   - Set [*Code*] to "`UsrViewLocalizationService`."
   - Set [*Title*] to "`UsrViewLocalizationService`."



Click [*Apply*] to apply the changes.

2. Create a service class

1. Add the `Terrasoft.Configuration` namespace in the Schema Designer.
2. Add the namespaces whose data types to utilize in the class using the `using` directive.
3. Add a class name that matches the schema name (the [*Code*] property).
4. Specify the `System.Web.SessionState.IReadOnlySessionState` class as a parent class.
5. Add the `[ServiceContract]` and `[AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Required)]` attributes to the class.

3. Implement the class method

1. Implement a method that returns the index of values of contact address types and contact addresses from the created `[UsrVwContactAddress]` non-localizable view. Add the `public string GetNonLocalizableView()` class method that implements the endpoint of the custom web service. The method executes database queries using `EntitySchemaQuery`.
2. Implement a method that returns the index of localizable values of contact address types and contact addresses from the created `[UsrVwContactAddress]` localizable view. Add the `public string GetLocalizableView()` class method that implements the endpoint of the custom web service. The method executes database queries using `EntitySchemaQuery`.

View the source code of the `UsrViewLocalizationService` custom web service below.

`UsrViewLocalizationService`

```
namespace Terrasoft.Configuration
{
    using System.ServiceModel;
    using System.ServiceModel.Web;
    using System.ServiceModel.Activation;
    using System.Web;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;
    using System;
    using System.Collections.Generic;

    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Requ
    public class UsrViewLocalizationService : System.Web.SessionState.IReadOnlySessionState
    {

        [OperationContract]
        [WebInvoke(Method = "GET", UriTemplate = "GetNonLocalizableView")]
        public string GetNonLocalizableView()
        {
            var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];
            var esqResult = new EntitySchemaQuery(userConnection.EntitySchemaManager, "UsrVwCont
            esqResult.AddColumn("UsrAddress");
            esqResult.AddColumn("UsrAddressType");
            var entities = esqResult.GetEntityCollection(userConnection);
            var s = "";
            foreach (var item in entities)
            {
                s += item.GetTypedColumnValue<string>("UsrAddressType") + ": ";
                s += item.GetTypedColumnValue<string>("UsrAddress") + "; ";
            }
            return s;
        }

        [OperationContract]
        [WebInvoke(Method = "GET", UriTemplate = "GetLocalizableView")]
        public string GetLocalizableView()
        {
            var userConnection = (UserConnection)HttpContext.Current.Session["UserConnection"];
            var sysCulture = new SysCulture(userConnection);
            if (!sysCulture.FetchPrimaryInfoFromDB("Name", "en-us"))
            {
                return "No culture found";
            }
            Guid CultureId = sysCulture.Id;

            var esqResult = new EntitySchemaQuery(userConnection.EntitySchemaManager, "UsrVwCont
            esqResult.AddColumn("UsrAddress");
            esqResult.AddColumn("UsrAddressType");
```

```
            esqResult.SetLocalizationCultureId(CultureId);

            var entities = esqResult.GetEntityCollection(userConnection);
            var s = "";
            foreach (var item in entities)
            {
                s += item.GetTypedColumnValue<string>("UsrAddressType") + ": ";
                s += item.GetTypedColumnValue<string>("UsrAddress") + "; ";
            }
            return s;
        }
    }
}
```

Click [*Save*] then [*Publish*] on the Designer's toolbar.

Outcome of the custom web service

As a result, Creatio will add the custom `UsrViewLocalizationService` web service that has the `GetLocalizableView` and `GetNonLocalizableView` endpoints.

Log in to Creatio and access the `GetLocalizableView` endpoint of the web service from the browser.

Request string

```
http://mycreatio.com/0/rest/UsrViewLocalizationService/GetLocalizableView
```

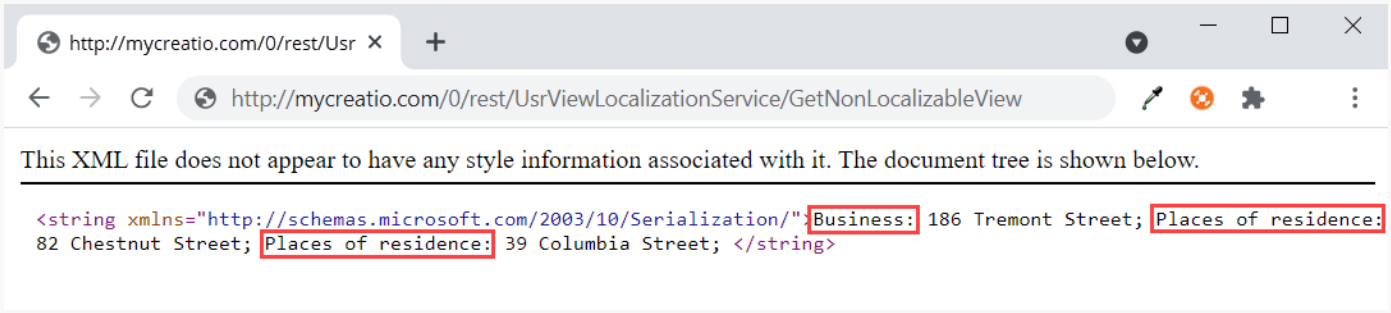
As a result, you will receive a selection that contains the localizable values of contact address types and contact addresses.

Access the `GetNonLocalizableView` endpoint of the web service from the browser.

Request string

```
http://mycreatio.com/0/rest/UsrViewLocalizationService/GetNonLocalizableView
```

As a result, you will receive a collection that contains the non-localizable values of contact address types and contact addresses.



LocalizableValue<T> class C#

Advanced

The `Terrasoft.Common` namespace.

The `Terrasoft.Common.LocalizableValue<T>` class is a base class for the `Terrasoft.Common.LocalizableString` (displays localizable strings) and `Terrasoft.Common.LocalizableImage` (displays localizable images) classes that work with localizable resources.

The `Terrasoft.Common.LocalizableValue<T>` class is a template for localizable values of different types. It also provides methods that work with them.

Note. View the full list of the properties, methods, and implemented interfaces of the `LocalizableValue<T>` class in the .NET class reference.

Properties

`Value` `T`

Returns and sets the localizable value based on the current language culture.

`HasValue` `bool`

Returns the flag that determines whether a localizable value of the current type exists for the current language culture.

`CultureValues` `IDictionary<CultureInfo,T>`

Returns a localizable value lookup of the current instance for available language cultures.

Methods

`void ClearCultureValue(CultureInfo culture)`

Deletes the localizable value for available language cultures.

Parameters

`culture`	A language culture.

`T GetCultureValue(CultureInfo culture, bool throwIfNoManager)`

Retrieves a localizable value of the specified type for available language cultures. Depending on the `throwIfNoManager` parameter value, the method can throw an exception of the `ItemNotFoundException` type if the resource manager is not set for that localizable value.

Parameters

`culture`	A language culture.
`throwIfNoManager`	Flag that determines whether to call the `ItemNotFoundException` exception.

`T GetCultureValueWithFallback(CultureInfo culture, bool throwIfNoManager)`

Retrieves a localizable value of the specified type for available language cultures. If no localizable resource value is found for the specified language culture, returns the value for the primary language culture. Depending on the `throwIfNoManager` parameter value, the method can throw an exception of the `ItemNotFoundException` type if the resource manager is not set for that localizable value.

Parameters

`culture`	A language culture.
`throwIfNoManager`	Flag that determines whether to call the `ItemNotFoundException` exception.

`bool HasCultureValue(CultureInfo culture)`

Determines whether a localizable value exists for the specified language culture.

Parameters

`culture`	A language culture.

`void LoadCultureValues()`

Loads an index of localizable values of the specified type for all cultures defined in the global resource storage.

`void SetCultureValue(CultureInfo culture, T value)`

Sets the specified localizable value for the specified language culture.

Parameters

`culture`	A language culture.
`value`	A localizable value.