Creatio Academy

Creatio IDE

Source code (C#)

Version 8.0

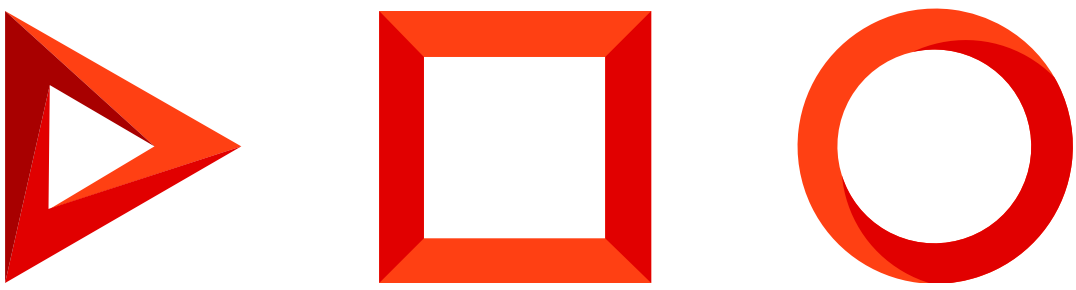


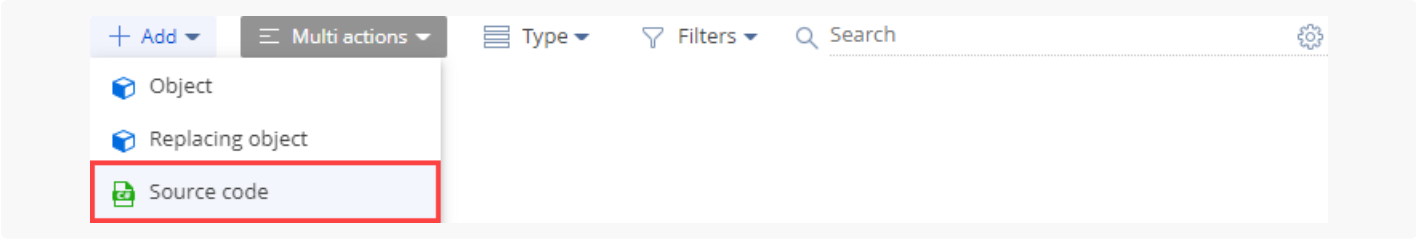
Table of Contents

Source code (C#)

Beginner

A **configuration element** of the [*Source code*] type is an entity that implements the business logic. The element lets you add, delete, and format the C# source code of new functionality. The **purpose** of the configuration element of the [*Source code*] type is to enable Creatio back-end development.

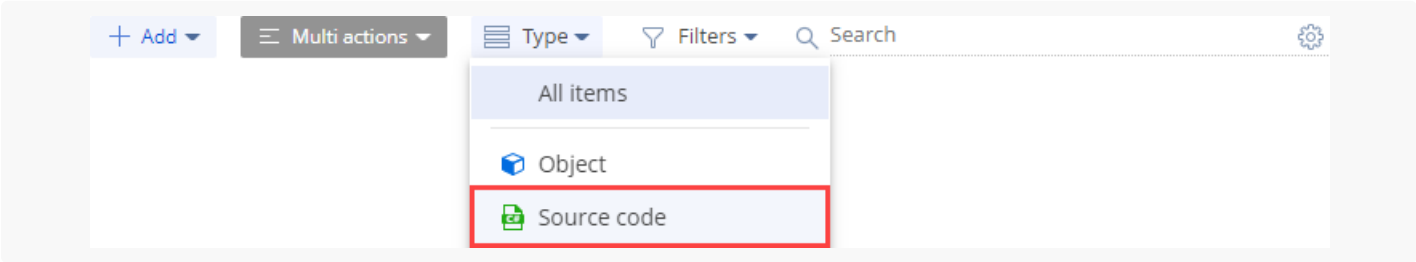
The items of the [*Add*] drop-down list in the toolbar of the [*Configuration*] section workspace represent the source code schema you can add in Creatio IDE.



Learn more about configuration element types in a separate article: Operations in Creatio IDE.

The [*Source code*] type schema in the [*Type*] drop-down list in the toolbar of the [*Configuration*] section workspace represents the configuration element of the [*Source code*] type. A **schema** is the basis of Creatio configuration.

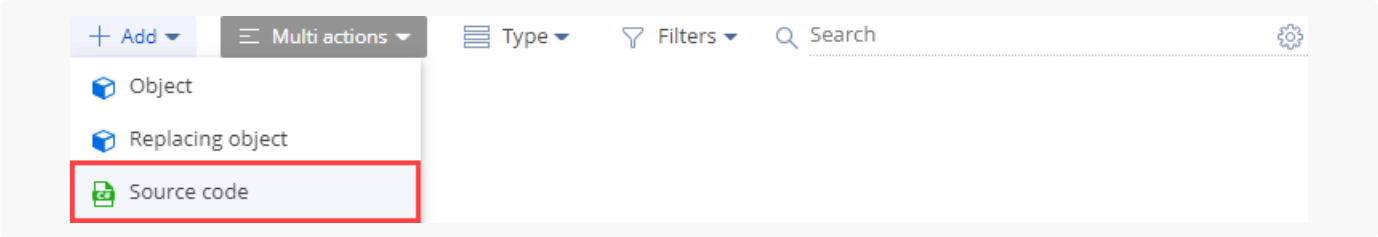
View the **type** of the source code schema in the figure below.



Learn more about configuration element types in a separate article: Operations in Creatio IDE.

Implement the source code

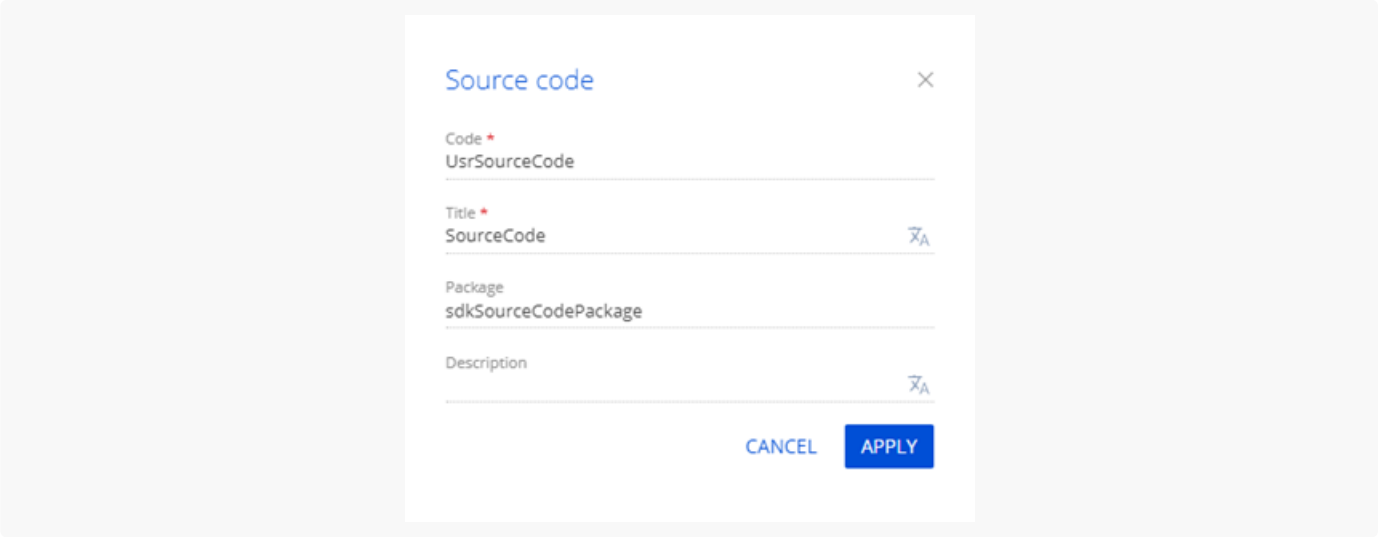
1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Source code*] on the section list toolbar.



3. Fill out the schema properties in the Source Code Designer.

   The **main schema properties** are as follows:

   - [*Code*] is the schema name. Required. Starts with the prefix specified in the [*Prefix for object name*] (`SchemaNamePrefix` code) system setting, `Usr` by default. Can contain Latin characters and digits. When a configuration element schema is created, the prefix specified in the [*Prefix for object name*] (`SchemaNamePrefix` code) system setting is added to the current field automatically. Creatio checks for the prefix and whether it matches the system setting value when saving the schema properties. If the prefix is missing or does not match, the user receives a corresponding notification.
   - [*Title*] is the localizable schema title. Required. The title of the configuration element schema is generated automatically and matches the value of the [*Code*] property without the prefix.
   - [*Package*] is the custom package where you create the schema. The property is populated automatically and non-editable.
   - [*Description*] is the localizable column description.



   Click [*Apply*] to apply the properties.

   The **properties area** of the Source Code Designer lets you:

   - edit the main schema properties (button)
   - specify the additional schema properties (button)

   The **additional schema properties** are as follows: [*Localizable strings*]

4. Add the source code in the Source Code Designer. The name of the class declared in the source code must match the schema name in the [*Code*] property.

Creatio displays the type of the error or warning – if any – to the left of the row number. Hover over the error type to view a tooltip with the error description.

Creatio IDE lets you create a replacing class in the Source Code Designer. To do this, follow the guide in a different section: Implement a replacing class.

5. Click [*Save*] on the Source Code Designer's toolbar to save the changes to Creatio metadata temporarily.
6. Click [*Publish*] on the Source Code Designer's toolbar to apply the changes to the database level.

Implement a replacing class

The class replacement principle, including the creation and use of replaced class instances in the configuration, has unique features.

To **implement a replacing class**:

1. Create the class to inherit the replaced class.
2. Add the `[Override]` attribute to the class. Learn more about the attribute in a separate article: `[Override]` attribute.
3. Implement the functionality that distinguishes the replacing class from the replaced class. For example, implement properties and methods that expand the functionality of the replaced class, method overloads of the replaced class, etc.

   - Add the `override` modifier to the replacing class's properties and methods.
   - Add the `virtual` modifier to the replaced class's properties and methods to replace.

   In a base class, you can replace only virtual methods or implement abstract methods. The replacing properties and methods declared without the `override` keyword are unavailable before the compilation. The Ninject open-source dependency injection framework binds and injects type dependencies only during the execution.

Learn more about replacing configuration elements in a separate article: Replace configuration elements.