Creatio Academy

Back-end development

Automatic data binding

Version 8.0

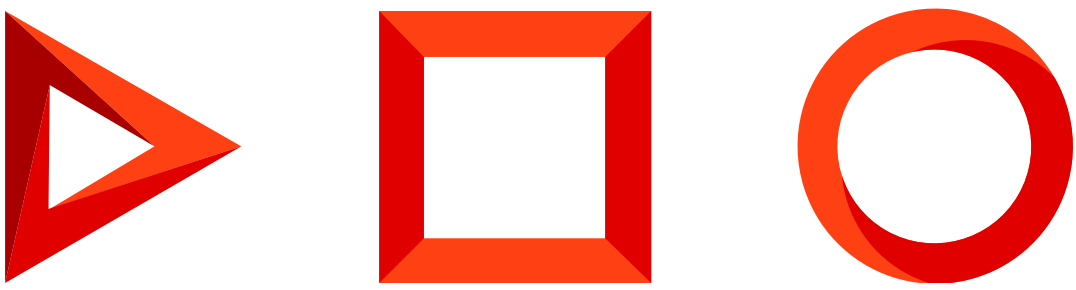


Table of Contents

Automatic data binding

Advanced

Use the **automatic data binding** mechanism on the data of objects with a complex object model. Automatic data binding is available for product catalog in Financial Services Creatio product lineup.

Structure of automatic data binding

View the **components** of automatic data binding in the table below.

Components of automatic data binding

Name	Purpose	Classes and interfaces
Service	Generates data bindings based on the set structure Creatio generates data bindings in the background.	`Service`. The interface that calls automatic data binding from the front-end.
Controller	Accesses the `Obtainer` (obtains the structure of bound records) and `Manufacturer` (generates the data bindings) classes.	`Controller`. The class that manages the creation of data bindings.
Binding generator	Wraps the `PackageElementUtilities` core class. Provides a convenient interface to execute CRUD operations on data bindings. Learn more about data operations in separate articles: Data access through ORM, Access data directly.	`Manufacturer`. A proxy class that interacts with the controller, `PackageElementUtilities` class, and `PackageValidator` class.
Structure obtainer	Returns the structure of the main entity's bound records to the controller.	`Obtainer`. Provides flexible modification of the structure of the main entity's bound objects when creating data bindings. Uses the `Strategy` template to define the required implementation.

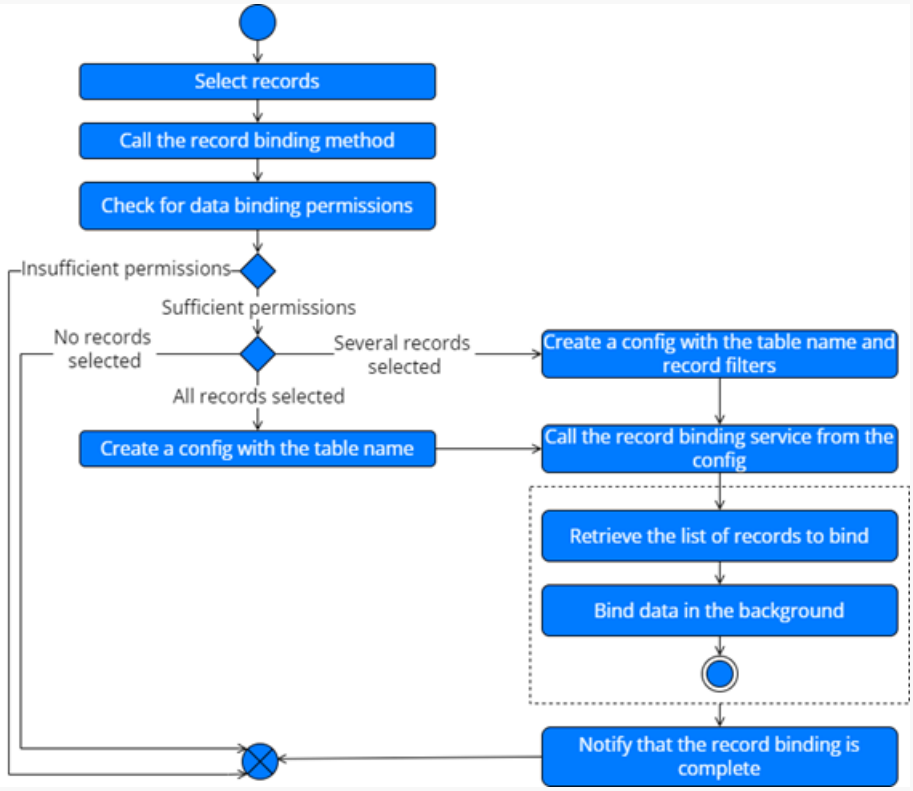
Creatio implements the **base structure**, within which only the current entity and its columns (except for system columns) are returned. View the properties of the base structure in the table below.

Properties of the base structure of bound entities

Property	Description
`EntitySchemaName`	The name of the entity.
`FilterColumnPathes`	The array of paths to the main entity's primary column. If you use filters, join them using the `OR` operator.
`IsBundle`	The way to save the records. The **ways** to save the records are as follows: • Save the records to a single binding. • Create a binding per each record bound to the main entity record via the filter.
`Columns`	The list of columns to bind.
`InnerStructures`	The list of nested objects' structures. For example, details connected to the main entity.
`DependsStructures`	The list of structures on which the current structure depends. For example, lookups to which the record in the current structure links via lookup fields.

Operating procedure of automatic data binding

View the **operating procedure** of automatic data binding in the figure below.



Creatio starts processing the structure from the main entity. The structure is processed for each instance of the `SchemaDataBindingStructure` structure class.

The **structure processing procedure** is as follows:

1. Creatio recursively processes each structure from the list of structures contained in the `DependsStructures` property. This is done to bind the objects to which the current object links before binding the main entity.
2. Creatio adds the binding to the main entity record by the filter in the `FilterColumnPathes` property.
3. Creatio recursively processes each structure from the list of structures contained in the `InnerStructures` property. Since the structures reference the main entity, bind them after binding the main entity.

The following **naming patterns** are available:

- `agb_EntitySchemaName` if the `IsBundle` property is `true`. `EntitySchemaName` is the name of the current binding's entity.
- `agb_EntitySchemaName_PrimaryEntityId` if the `IsBundle` property is `false`. `PrimaryEntityId` is the ID of the main entity record to which the records in the current binding are bound.

Attention. We recommend against renaming automatically created bindings. Automatic updates are available only for the bindings whose names match the specified template.

Add a custom structure for the object binding

1. Create a [*Source code*] schema. Learn more about creating schemas in a separate article: Develop

configuration elements.

2. Create a service class.

   a. Add the `Terrasoft.Configuration` namespace in the Schema Designer.

   b. Create a class. Name the class according to the following template: `[EntityName]SchemaDataBindingStructureObtainer`. For example, `ProductSchemaDataBindingStructureObtainer`. Creatio will search for classes based on the template when defining the relevant obtainer implementation (the `Obtainer` component).

   c. Specify the `BaseSchemaDataBindingStructureObtainer` class as a parent class.

   d. Add a custom implementation of the `ObtainStructure()` interface method. Create and return the needed entity structure in this method.

   `ObtainStructure()` method

   ```
   public override SchemaDataBindingStructure ObtainStructure(UserConnection userConnection)
   ```

   Attention. If you need to modify a structure implementation from a non-editable package, create a class in a custom package and use the `[Override]` attribute. Learn more about the attribute in a separate article: `[Override]` attribute.

3. Publish the source code schema.

Call automatic data binding

You can call automatic data binding from the front-end and back-end.

Call automatic data binding from the front-end

1. Call the service that generates bindings.
2. Specify the binding generation settings.

   You can specify binding generation settings in the following **ways**:

   - Pass the parameters.
     - package for which to generate the binding
     - name of the entity
     - list of IDs to bind
   - Pass the filter.

View the example that calls automatic data binding from the front-end using the parameters below.

Example that calls automatic data binding from the front-end using parameters

```
ServiceHelper.callService("SchemaDataBindingService",
    "GenerateBindings",
    callback,
    {
        "schemaName": this.entitySchemaName,
        "sysPackageUId": sysPackageUId
        "recordIds": ["...", "..."]
    },
    this
);
```

View the example that calls automatic data binding from the front-end using a filter below.

Example that calls automatic data binding from the front-end using a filter

```
ServiceHelper.callService("SchemaDataBindingService",
    "GenerateBindingsByFilter",
    callback,
    {
        "schemaName": this.entitySchemaName,
        "sysPackageUId": sysPackageUId
        "filterConfig": ...
    },
    this
);
```

Attention. Creatio runs automatic data binding called from the front-end in the background. Once the binding is complete, Creatio will display a notification message in the communication panel. The message will contain the number of successfully and unsuccessfully bound product records.

Call automatic data binding from the back-end

1. Create an instance of the `SchemaDataBindingController` class.
2. Call the `GenerateBindings` method. Pass the following to the method:

   - package ID
   - entity name
   - list of IDs to bind

View the example that calls automatic data binding from the back-end below.

Example that calls automatic data binding from the back-end

```
var dataBindingController = ClassFactory.Get<IDataBindingController>(
    new ConstructorArgument("userConnection", UserConnection));
var result = dataBindingController.GenerateBindings(schemaName, recordIds, sysPackageUId);
```