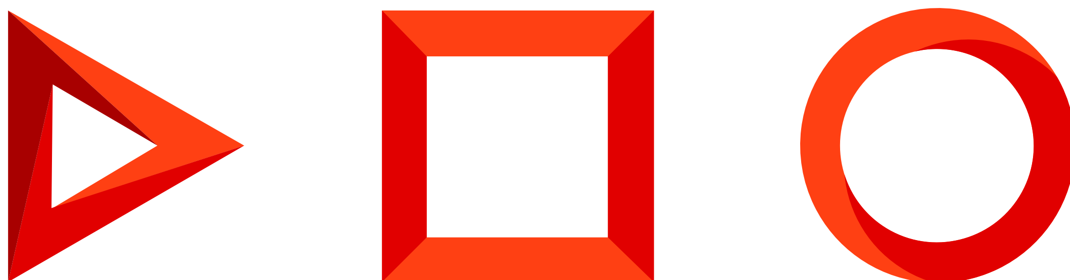
Creatio Academy

Data sources

Data handling

Version 8.0



This documentation is provided under restrictions on use and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this documentation, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

© 2023 Creatio. All rights reserved.

Table of Contents

Data handling **4**
DataStorage layer 4
DataAccessObject layer 5
BusinessObject layer 6

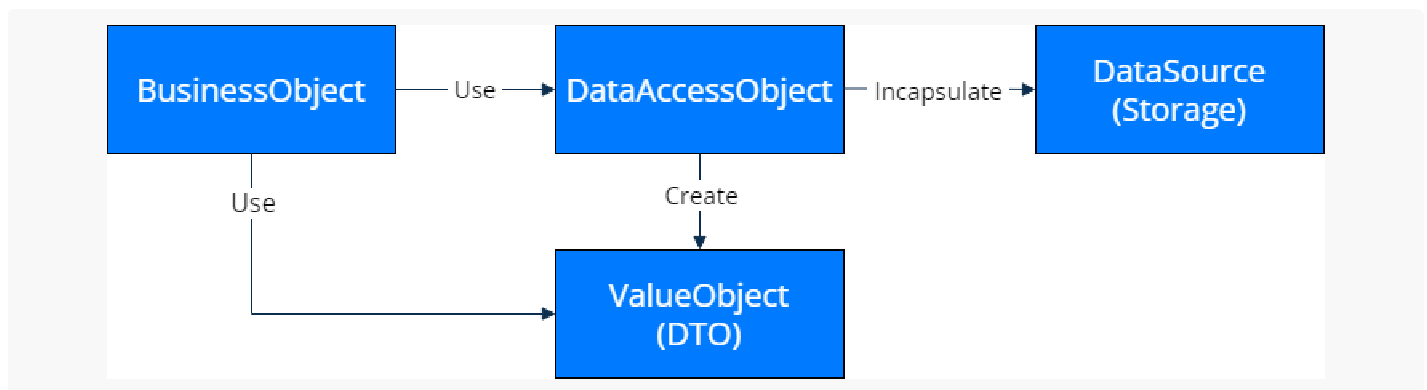
© 2023 Creatio. All rights reserved.

Data handling

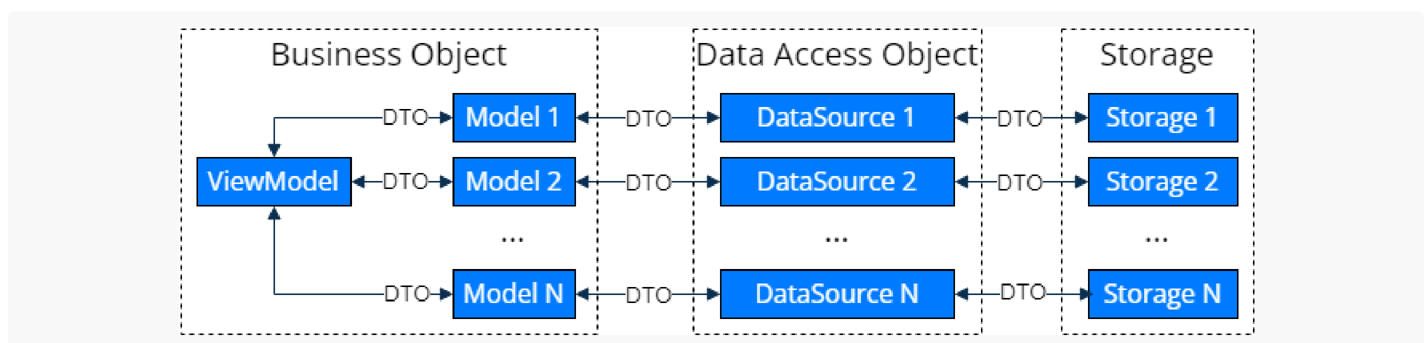
Data handling in Creatio 8.x is based on the **DAO (Data Access Object)** pattern. Learn more about the DAO pattern in Wikipedia.

The DAO pattern has the following **layers**:

- `BusinessObject`. Implements the business logic.
- `DataAccessObject`. Enables Creatio to access data. The business logic and data source are separated from each other. The DataAccessObject layer can create a **DTO (Data Transfer Object)**. The DTO can use the `BusinessObject` layer to save DTOs and return them to the DataAccessObject layer. Learn more about the DTO pattern in Wikipedia.
- `DataStorage`. Organizes various data repositories.



View the detailed data storage diagram in the figure below.



DataStorage layer

The `DataStorage` layer is linked to a particular data source.

The **types** of storage are as follows:

- Creatio database
- web service

© 2023 Creatio. All rights reserved.

- file system

In Creatio 8.0 Atlas, you can access only the Creatio database.

`DataAccessObject` layer

The `DataAccessObject` layer contains data sources (Data Source). Each data source is linked to the corresponding model within the business logic layer.

The `DataAccessObject` layer performs the following **functions**:

- Enable users to perform CRUD operations (`load` , `save` , `insert` , `delete`).
- Provide permissions for data operations (`canEdit` , `canCreate` , `canDelete`).
- Provide data structure (`getSchema`).

Data schema describes the structure of data managed by Creatio.

Data schema includes the following **components:**

- name
- title
- attributes

Important structural elements of attributes are validators. Learn more about validators in a separate article: Freedom UI page customization basics.

Learn more about the **classes** that describe the data schema and attribute structure below.

Data schema structure

```
export class DataSchema {
    public name: string;
    public caption: LocalizableString;
    public attributes: DataSchemaAttribute[];
    public primaryAttributeName?: string;
    public primaryDisplayAttributeName?: string;
}
```

Attribute schema structure

```
export class DataSchemaAttribute {
    public name: string;
    public caption: string;
    public path: string;
    public dataValueType: DataValueType;
    public validators: DataSchemaValidatorConfig;
    public defaultValue: JsonData;
```

© 2023 Creatio. All rights reserved.

```
    public isValueCloneable: boolean;
    public attributeType: DataSchemaAttributeType;
    public referenceSchemaName?: string;
}
```

Attention. Creatio version 8.0 Atlas implements `EntityDataSource` that lets you manage Creatio database.

`BusinessObject` layer

The `BusinessObject` layer contains the view model (`View model`), which, in turn, can contain several models (`Model`). Learn more about the `View model` and `Model` layers in a separate article: Creatio front-end architecture.

Model

The `modelConfig` section of the client schema describes data models. The model contains the data source description. Creatio version 8.0 Atlas implements `EntityDataSource` that lets you manage the Creatio database.

Use the data source to perform various **data actions**, for example:

- Add new and existing data source elements to the canvas of the Freedom UI Designer.
- Edit the data source elements.
- Bind the controls to the data source elements on the canvas.

View an example that registers a model in the client schema below.

Example that registers a model in the client schema

```
modelConfig: /**SCHEMA_MODEL_CONFIG*/{
    "dataSources": {
        /* Unique names of data sources. */
        "ContactDS": {
            /* Data source type. */
            "type": "crt.EntityDataSource",
            "config": {
                /* Data schema code. */
                "entitySchemaName": "Contact"
            }
        }
    }
}/**SCHEMA_MODEL_CONFIG*/
```

Learn more about the client schema structure in a separate article: Client Schema.

© 2023 Creatio. All rights reserved.

View model

The **view model** handles the `ViewModel` layer that contains the business logic of the interaction between the `View` and `Model` layers.

Configure the view model in the `viewModelConfig` section of the client schema. For each view model attribute, specify the corresponding model attribute in the `modelConfig` schema section.

View an example that describes a view model in the client schema below.

Example that describes a view model in the client schema

```
viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
    "attributes": {
        /* Bind the ContactName view model attribute to the Name attribute of the ContactDS mode
        "ContactName": {
            "modelConfig": {
                "path": "ContactDS.Name"
            }
        }
    }
}/**SCHEMA_VIEW_MODEL_CONFIG*/
```

The view model can have the following **attribute types**:

- simple type columns
- collection columns
- direct link columns

Attributes for columns that contain simple data types

Simple data types can include the following data source elements:

- string
- number
- boolean
- date/time

To create an **attribute for a column that contains a simple data type**:

1. Register the data source in the schema of the Freedom UI page.

   Register the data source

   ```
   modelConfig: /**SCHEMA_MODEL_CONFIG*/{
       "dataSources": {
           "ContactDS": {
               "type": "crt.EntityDataSource",
   ```

© 2023 Creatio. All rights reserved.

```
            "config": {
                "entitySchemaName": "Contact",
            }
        }
    }
}/**SCHEMA_MODEL_CONFIG*/
```

2. Create an attribute in the `attributes` property of the `viewModelConfig` schema section.

Create an attribute

```
viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
    "attributes": {
        "StringAttribute_jghoo32": {
            "modelConfig": {
                "path": "ContactDS.Name"
            }
        }
    }
}/**SCHEMA_VIEW_MODEL_CONFIG*/,
```

Attention. Use a unique attribute name.

The base attribute property is `modelConfig`. This property describes the link to the data source element using the `path` property. The `path` property value must have the following format: `[Data Source Name].[Column Code]`. For this example, the `path` property value is `ContactDS.Name`.

Attributes for collections

The view model lets you bind attributes that are data collections. To bind a collection type attribute, set the `isCollection` attribute property to `true`.

Example that binds a collection type attribute to a data model

```
viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
    "attributes": {
        /* Bind a collection type attribute of the AccountList view model to the AccountDS model
        "AccountList": {
            "isCollection": true,
            "modelConfig": {
                "path": "AccountDS",
                "pagingConfig": {},
                "sortingConfig": {},
                "filterAttributes": {}
```

© 2023 Creatio. All rights reserved.

```
            },
            "viewModelConfig": {
                "attributes": {
                    "value": {
                        "modelConfig": {
                            "path": "AccountDS.Id"
                        }
                    },
                    "displayValue": {
                        "modelConfig": {
                            "path": "AccountDS.Name"
                        }
                    }
                }
            },
            "embeddedModel": {
                "name": "AccountDS",
                "config": {
                    "type": "crt.EntityDataSource",
                    "config": {
                        "entitySchemaName": "Account"
                    }
                }
            }
        }
    }
}/**SCHEMA_VIEW_MODEL_CONFIG*/
```

The advanced settings of the collection type attribute let you configure data pagination and sorting.

Example of advanced settings of a collection type attribute

```
viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
    "attributes": {
        "AccountList": {
            "isCollection": true,
            "modelConfig": {
                "path": "AccountDS",
                /* Set up pagination.*/
                "pagingConfig": {
                    "rowsOffset"?: "<OffsetAttributeName>",
                    "rowCount": 5 | "<RowCountAttributeName>"
                },
                /* Set up collection sorting.*/
                "sortingConfig": {
                    "attributeName": "AccountsSorting",
                    "default": [{
```

© 2023 Creatio. All rights reserved.

```
                    "columnName": "Name",
                    "direction": "Asc"
                }]
            }
        },
        "viewModelConfig": {...}
    }
  }
}/**SCHEMA_VIEW_MODEL_CONFIG*/
```

Attributes for direct link columns

Direct link columns are columns that contain objects linked to the current data source. For example, the `[Contact]` object contains the `[Account]` property, which is also an object. Register the attribute in the `[Name]` column of the linked `[Account]` object.

To create an attribute for a direct link column:

1. Register the data source in the page schema.

   Example that registers a data source

   ```
   modelConfig: /**SCHEMA_MODEL_CONFIG*/{
       "dataSources": {
           "ContactDS": {
               "type": "crt.EntityDataSource",
               "config": {
                   "entitySchemaName": "Contact",
               }
           }
       }
   }/**SCHEMA_MODEL_CONFIG*/
   ```

2. Create a page schema attribute in the `attributes` property of the `viewModelConfig` schema section.

   Example that creates an attribute

   ```
   viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
       "attributes": {
           "AccountName": {
               "modelConfig": {
                   "path": "ContactDS.AccountName"
               }
           }
       }
   }/**SCHEMA_VIEW_MODEL_CONFIG*/,
   ```

© 2023 Creatio. All rights reserved.

The `path` property stores not the path to the column, but the path to the data source attribute that will be created on the next step. The `path` property value must have the following format: `[Data Source Name].[Data Source Attribute Name]`, for example, `ContactDS.AccountName`.

3. Create an attribute in the `modelConfig` schema section.

Example that creates a data source attribute

```
modelConfig: /**SCHEMA_MODEL_CONFIG*/{
    "dataSources": {
        "ContactDS": {
            "type": "crt.EntityDataSource",
            "config": {
                "entitySchemaName": "Contact",
                "attributes": {
                    "AccountName": {
                        "path": "Account.Name",
                        "type": "ForwardReference"
                    }
                }
            }
        }
    }
}/**SCHEMA_MODEL_CONFIG*/,
```

Attention. The name of the data source attribute must match the attribute name specified in the `path` property of the `viewModelConfig` schema section on step 2.

You can create a direct link attribute that has multiple nesting levels. For example, you can display the city name of the account linked to the current contact. In this case, the data source attribute can look like this:

Example that creates an attribute that has multiple nesting levels

```
"dataSources": {
    "ContactDS": {
        "type": "crt.EntityDataSource",
        "config": {
            "entitySchemaName": "Contact",
            "attributes": {
                "AccountCityName": {
                    "path": "Account.City.Name",
                    "type": "ForwardReference"
                },
            }
        }
    }
```

© 2023 Creatio. All rights reserved.

```
}
```

Embedded model

The **Embedded Model** is a data model embedded into the view model without being connected to the data source.

Creatio lets you use the embedded model for various **collection type elements**, for example:

- index of list values
- drop-down list generated from a lookup

Creatio does not display embedded data models in the Freedom UI Designer, nor does it let you bind embedded models to controls on the canvas.

Example that sets up an embedded model

```
viewModelConfig: /**SCHEMA_VIEW_MODEL_CONFIG*/{
    "attributes": {
        "LookupAttributeCityList": {
            "isCollection": true,
            "modelConfig": {
                "path": "EmbeddedDS"
            },
            "embeddedModel" {
                "name": "EmbeddedDS",
                "config": {
                    "type": "crt.EntityDataSource",
                    "config": {
                        "entitySchemaName": "City"
                    }
                }
            }
            "viewModelConfig": {
                "attributes": {
                    "value": {
                        "modelConfig": {
                            "path": "EmbeddedDS.Id"
                        }
                    },
                    "displayValue": {
                        "modelConfig": {
                            "path": "EmbeddedDS.Name"
                        }
                    }
                }
            }
        }
```

© 2023 Creatio. All rights reserved.

```
    }
}/**SCHEMA_VIEW_MODEL_CONFIG*/,
```

© 2023 Creatio. All rights reserved.