WHITESTEIN
Technologies

Living Systems® Process Suite

Scaffolding Library

Living Systems Process Suite Documentation

3.1
Tue Jan 12 2021

Whitestein Technologies AG | Hinterbergstrasse 20 | CH-6330 Cham
Tel +41 44-256-5000 | Fax +41 44-256-5001 | http://www.whitestein.com

Contents

Chapter 1

Creating CRUD Components

To prototype CRUD components in your forms quickly, use the scaffolding framework provided by the *scaffolding* module in the Scaffolding Library: it allows you to create form components that enable the front-end user to create, read, update, and delete objects of a particular data type.

> **Important:** The Scaffolding library resources are not optimized and hence not intended for production environment.

The capabilities are provided by the form component *GenericView*. By default the component renders as a CRUD table with object of a record type; however you can change the component definition in the component properties to render as another component.

1.1 Generic View

The *Generic View* component is a custom form component that creates and renders a component, by default a CRUD table, over instances of a particular data type.

In the default CRUD table, the columns represent the data type structure and the rows the data type instances: For example, if you have a Record Book with the fields `ISBN` and `title`, and you use it as the data type of the Generic View component, the rendered table will have the columns ISBN and title. If you use a Record instance, the table will in addition contain a row with the book's ISBN and title. From the table you will be able to create a new book instance, and delete and modify the book instances.

The Generic View component has the following properties:

Object Object or a list of Objects that are populating the form component and defining its structure

If not overridden by the Record options property, the fields of the underlying data type are used as CRUD table columns and every row represents one object.

Editable Whether the values are read-only or can be edited

If false, the table is read-only.

Object type If the component is editable and the Object expression returns an empty Collection, the system assumes the Collection items are of the data type defined by this property.

It is recommended to define the data type property since if the collection returned by the Object parameter is empty, the front-end layer of the system considers the collection data type to be `Null`.

Recursion Depth The number of relationships of the displayed data type (Record) that is displayed directly in the component of the top data type (Record)

If applied, the related data types are displayed as nested components which you can expand directly in the parent component. Otherwise, the data is displayed on a new page and the depth is indicated only in the breadcrumb.

Submit Button The component rendered instead of the default Submit button

Note that you can use a layout component to insert, for example, multiple buttons to where the Submit button is located by default.

Record options Custom Record rendering

If defined, the system does not use the generic CRUD table but the form defined in this map to render the Record whenever it appears in the Generic View.

```
[
  Author -> new RecordOptions(
    linkDisplayName -> {o:Object-> cast(o, Author).name},
    recordCustomForm -> {o:Object -> new OutputText(content ->{->"anonymous"})}
  ),
  Book -> new RecordOptions(
    propertyVisibilityDefault -> false,
    propertyOptions -> [
      Book.name -> new PropertyOptions(
        propertyVisibility -> true
      ),
      Book.authors -> new PropertyOptions(
        propertyVisibility -> true
      )
    ]
  )
]
```

1.2 Recursion Depth

When using Records as data for a Generic View component and the Record type has a Relationship to another Record, the Record instance is rendered in its own component (the CRUD table by default) which is displayed in a popup window on request. To display such objects directly in the parent component, define the Recursive property on your Generic View. If the Record instance is "within" the reach of the recursion, the component of the Record is rendered as a nested component that can be expanded. If this is not the case, the Record instance is displayed in a pop-up dialog box after clicking the view icon on the parent Record instance.

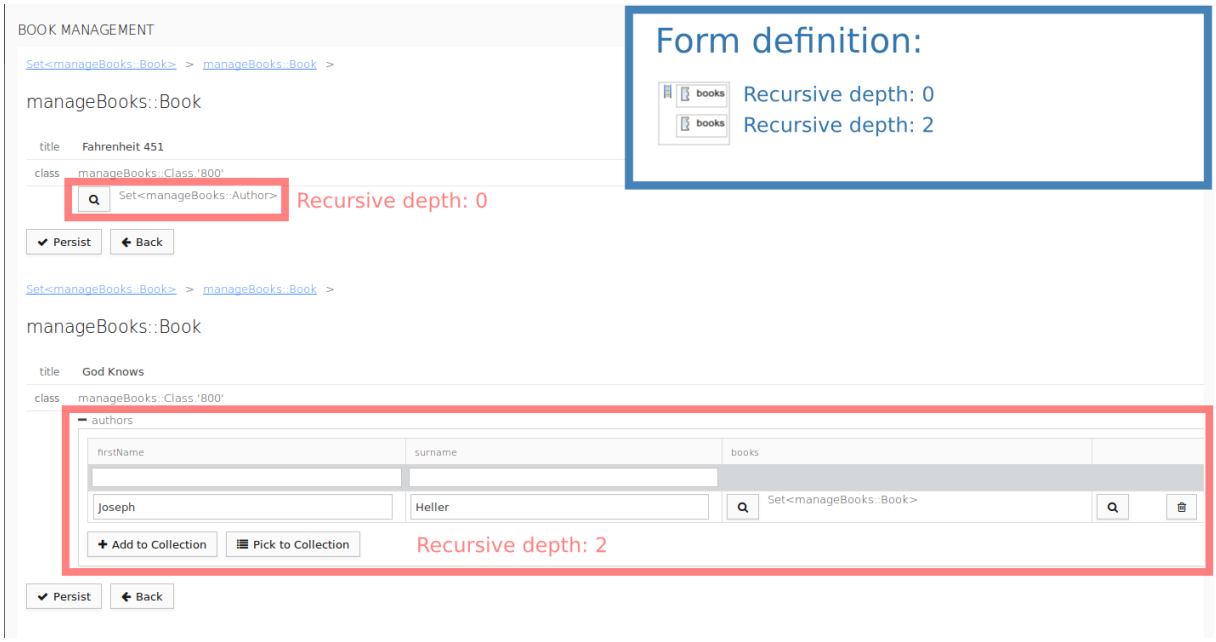

Figure 1.1 Document with Generic Views

The top Generic View has the Recursive depth property set to 0 while the bottom Generic View has the Recursive depth set to 2.

> **Note:** In the background the system always loads the data within the reach of one relationship, so that when requested the data is already available.

1.3 Using Generic View

Before you can make use of the library, you need to import it into your Project and Module:

1. In the GO-BPMN Explorer, right-click your Project, click Add Library; in the popup select Select a built-in library and select the Scaffolding Library in the Library drop-down box.
2. In the GO-BPMN Explorer, click the module. In the Imports tab of the Properties view, click Add. In the pop-up, select the scaffolding module.

To create a form with the scaffolding component, do the following:

1. Create or open a form definition.
2. Insert the Generic View custom component.

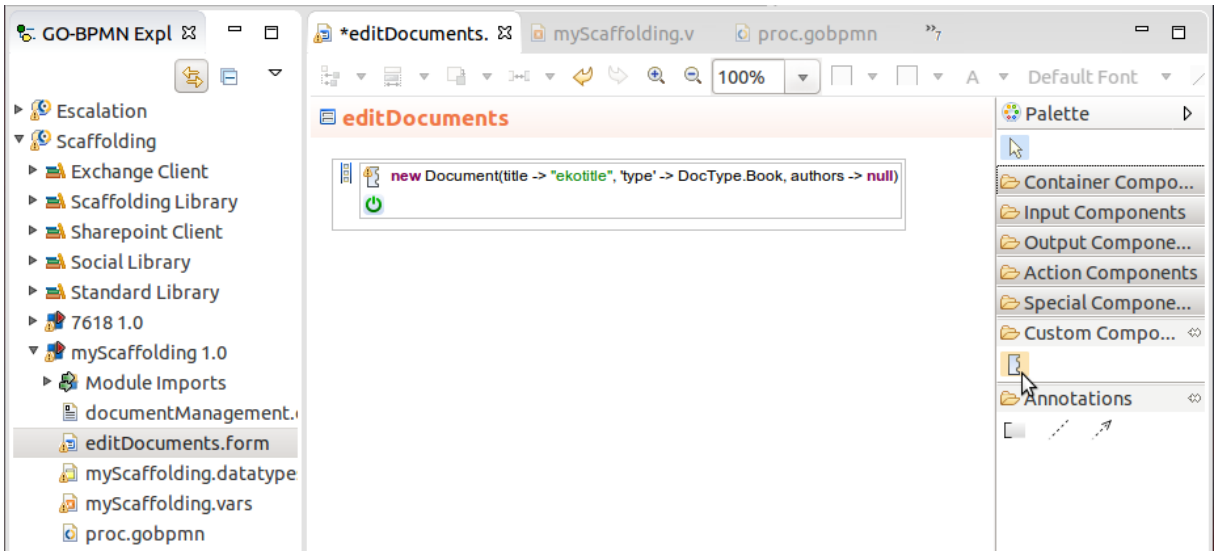

Figure 1.2 Generic view in palette

3. Define the required Generic View properties:
   - object: expression that returns an object or a collection of objects that you want to work with in the component
   - editable: whether the rows are editable
   - object type: data type applied if the object expression returns an empty collection as the data type of the collection objects
4. Define the optional properties:
   - recursion depth: the depth of the data types (Records) displayed directly in the component
   - submit button: component rendered instead of the Submit button

     Note that this expression has no impact on the **OK & Persist** button, which is displayed when creating a new instance in a Collection of shared Records from the Generic View.
   - record options: custom rendering of the Records
5. Run preview of the form on the Embedded Server if applicable: right-click the form definition and go to **Run As** > **Form Preview**.