

Living Systems® Process Suite

Scaffolding Library

Living Systems Process Suite Documentation

3.2
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>

Copyright © 2007-2021 Whitestein Technologies AG
All rights reserved.

Copyright © 2007-2021 Whitestein Technologies AG.

This document is part of the Living Systems® Process Suite product, and its use is governed by the corresponding license agreement. All rights reserved.

Whitestein Technologies, Living Systems, and the corresponding logos are registered trademarks of Whitestein Technologies AG. Java and all Java-based trademarks are trademarks of Oracle and/or its affiliates. Other company, product, or service names may be trademarks or service marks of their respective holders.

Contents

1	Creating CRUD Components	1
1.1	Generic View	1
1.2	Recursion Depth	3

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.

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.

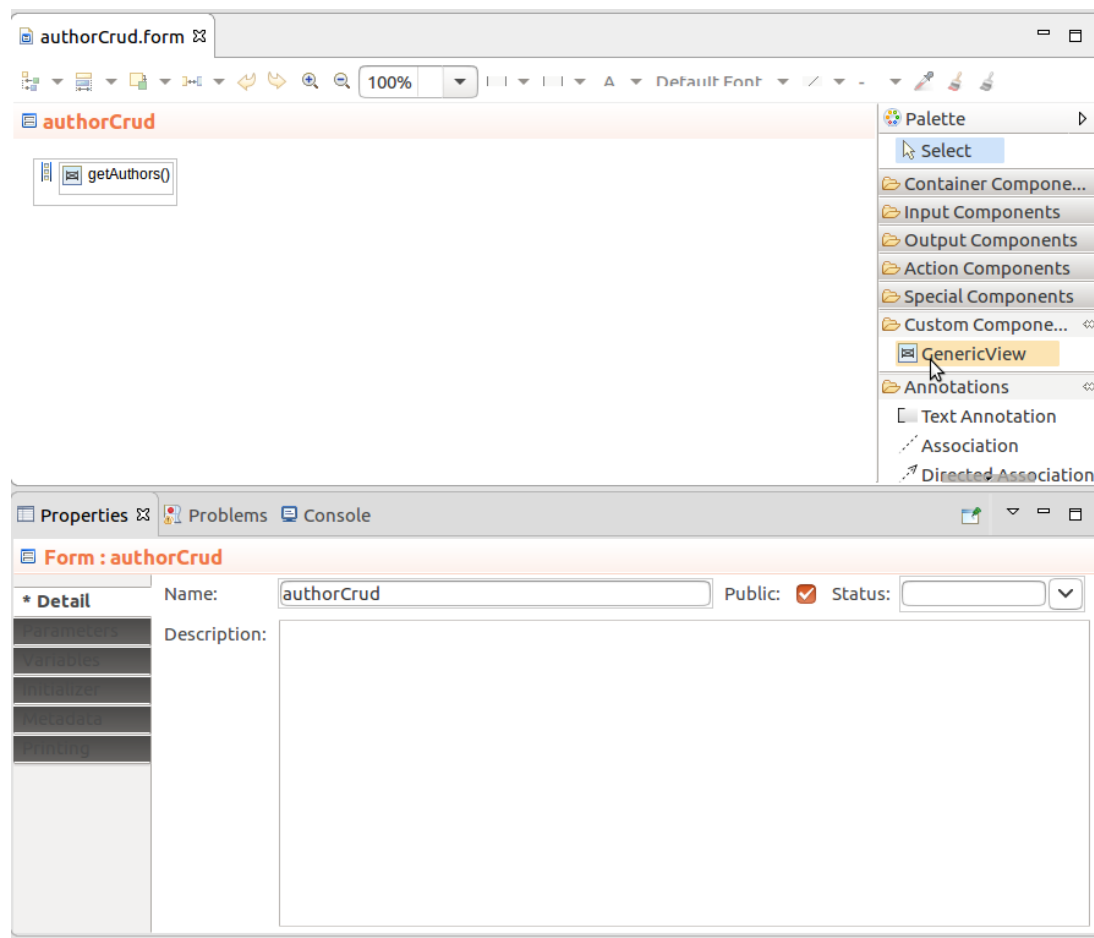


Figure 1.1 Generic view in palette

3. Define the required Generic View properties:

Object Collection of Objects that are the content of the form component

Each object is represented by a table row.

For object of a simple type, their value is rendered in a sole column; for objects of a record type, the value of each field is renderer in its own column; Note that this can be overridden by the *record options* property.

Editable If true, the table is editable: if the table contains different types of objects, the table remains read-only.

Object type If the Object expression returns an empty Collection, the table is considered to operate over this data type, hence if the table is editable and you click *Create new instance*, the table creates a new instance of this type).

Recursion Depth The relationship depth from the table record that is displayed directly in the component of the top data type: the related data types are displayed as sub-node components. They can be expanded 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 by default (a CRUD table by default). 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.

BOOK MANAGEMENT

[Set<manageBooks:Book>](#) > [manageBooks:Book](#) >

manageBooks::Book

title Fahrenheit 451

class manageBooks:Class:800'

Q Set<manageBooks:Author> Recursive depth: 0

[Set<manageBooks:Book>](#) > [manageBooks:Book](#) >

manageBooks::Book

title God Knows

class manageBooks:Class:800'

authors

firstName	surname	books
Joseph	Heller	Add to Collection Pick to Collection Set<manageBooks:Book> Q X

Form definition:

books

Recursive depth: 0

books

Recursive depth: 2

Figure 1.2 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.

