COMPEX COMMERCE

Process Specification

DialogDSL

Version 1.0

Date: 30.10.15

Contents

PS_DialogDSL_1_0	Created	Checked	Released
Date			
Signature			

1 Introduction

1.1 Vaadin

Vaadin is a web application framework for Java. In contrast to Javascript libraries and browser-plugin based solutions, Vaadin features a complete stack that includes a robust server-side programming model as well as client-side development tools based on GWT and HTML5.

More information:

https://vaadin.com/home

1.2 Vaaclipse

Vaaclipse is a framework for building web applications using Eclipse 4 Platform and Vaadin. It allows to use the power of the Eclipse 4 in web development. Vaaclipse moves the Eclipse Platform to Web using the rich web capabilities of Vaadin. You create your web application using Eclipse 4 features such as Eclipse Workbench, Application Model, Dependency Injection. You provide your own application parts using Vaadin widget library.

2 DialogDSL

DialogDSL generates the vaddin ui.

The main semantic elements of the DialogDSL are:

- package" - the root element that contains all the other elements. A model can contain multiple packages.
- "import" declarations - used to import external models or even Java classes.
- "dialog" - define the dialog configurations, e.g. ui view configuration, handler type, etc..
- "view" - define the ui view for this dialog.
- "handler" - define the handler type for this dialog, e.g. default, new, save, cancel, delete.

2.1 Syntax

2.1.1 Package definition

▶ Syntax:

```
package <package name>  {
    import <import models/class name>
     ...
    dialog <dialog name> view <ui model name>[handler {
           . . .
    }]
    refreshingView <refreshingView String>
     ...
}
```

Note:

ui model is defined in a *xx.uimodel* file, which using Dto models for the data exchanging with database. See the following example.

▶ Example:

```
xx.dialog:
package de.compex.foodmart.views {
    import de.compex.foodmart.uimodels.*
    . . .
    dialog Employees view EmployeeDialog handler {
        default new save cancel delete
    }
    refreshingView "EmployeesTable"
     . . .
}
xx.uimodel:
package de.compex.foodmart.uimodels

import de.compex.foodmart.dtos.*

ideview EmployeeDialog {

    datasource dsEmployee : MemployeeDto

    verticalLayout {
        horizontalLayout {
            autowire source dsEmployee
        }
    }
}
```

2.1.2 dialog

- Syntax:

```
dialog <dialog name> view <ui model name>[handler {
           <dialog handler>
    }]
    refreshingView <refreshing View String>
```

- Simplified syntax (see 2.2):

```
dialog <dialog name> autobind <dto name> ...
```

Generate a *<dialogname>+Vaaclipse.java* file, in which a java class named *<dialogname>+Vaaclipse* extended from java class `AbstractHybridVaaclipseView` is defined. In this class, dialog and dialog configurations are defined.

For each handler, generate a *<dialogname>+ <UPPER HANDLERNAME>+ Handler.java*, in which a java class *<dialogname>+ <UPPER HANDLERNAME>+ Handler* is defined. The methods canExecute and execute+*<handlername>* are defined in this file, and they will be used as handlers by an Eclipse 4 application.

☞ **Notes:**

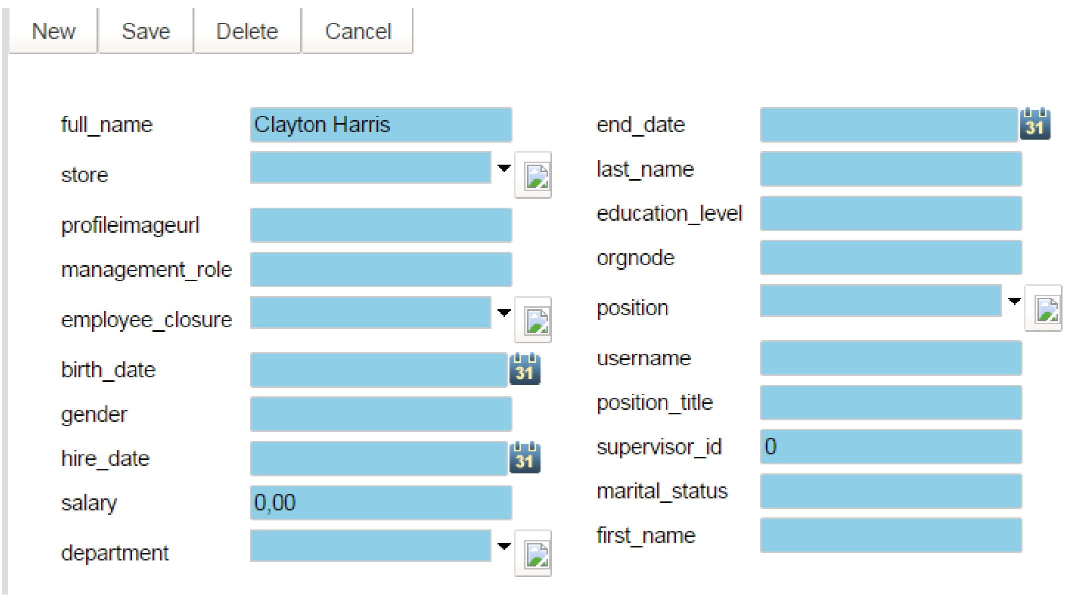
dialog handler is including *default*, *new*, *save*, *cancel* and *delete*.

refreschingView is automatically set `eventBroker.send(EventBrokerConstants.REFRESH_VIEW+"refresching View String", null);` with every handleEvent in *subscribe()*.

- Example:

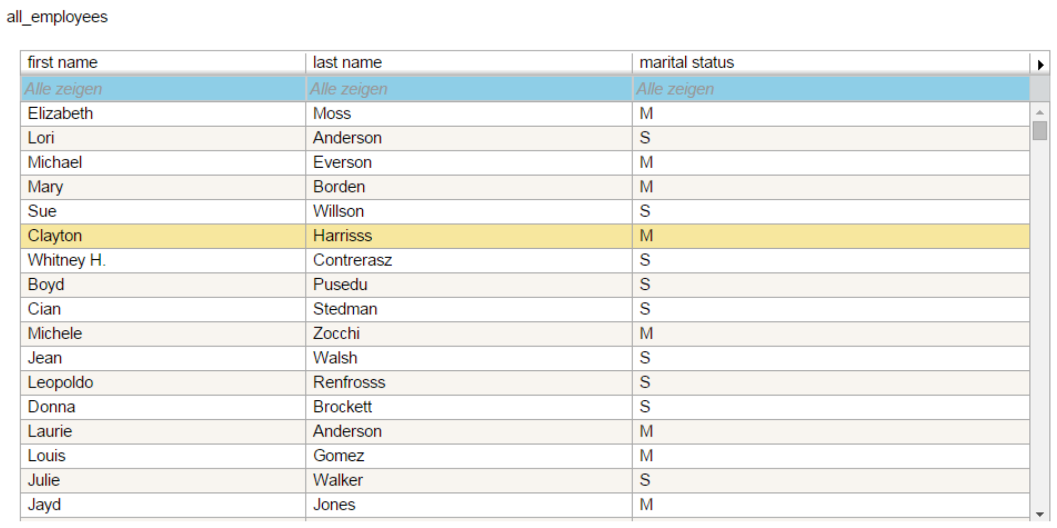
```
dialog Employees view EmployeeDialog handler {
    default new save cancel delete
    }
    refreshingView "EmployeesTable"
```

- Employees Dialog:



full_name	Clayton Harris	end_date	
store		last_name	
profileimageurl		education_level	
management_role		orgnode	
employee_closure		position	
birth_date		username	
gender		position_title	
hire_date		supervisor_id	0
salary	0,00	marital_status	
department		first_name	

- Employees Table:


all_employees

first name	last name	marital status
Alle zeigen	Alle zeigen	Alle zeigen
Elizabeth	Moss	M
Lori	Anderson	S
Michael	Everson	M
Mary	Borden	M
Sue	Willson	S
Clayton	Harrisss	M
Whitney H.	Contrerasz	S
Boyd	Pusedu	S
Cian	Stedman	S
Michele	Zocchi	M
Jean	Walsh	S
Leopoldo	Renfrosss	S
Donna	Brockett	S
Laurie	Anderson	M
Louis	Gomez	M
Julie	Walker	S
Jayd	Jones	M

When we select a row in employees table, the details of table will be automatically showed in employees dialog, and then, we could modify all details of this record in database using this dialog.

2.2 Extension: Direct use of DTOs

The DialogDSL has been extended to allow for a simpler use of DTOs. Using this new syntax, it is no longer necessary to reference a datasource from a ui model since DTOs can now be addressed directly. See the following example:

- Example:

```
dialog Products autobinding ProductDto toolbar Dialog
```

This extension works as follows: A transient ui model is generated directly in the DialogDSL. This ui model is never serialized to disk.

In the lifecycle of the dialog model, this ui model is automatically converted to an ECView model and written to disk as an .ecview file.

To achieve this, a DialogDslDerivedStateComputer has been created that generates the transient ui model “on the fly” and adds it to the resource as a “derived state”.

In a second step, the DialogDslStateComputer calls the UiModelDerivedStateComputerx and passes the transient ui model to it. Thus, the (transient) ui model is translated to an ECView model.

In a last step, the DialogModelGenerator reads the ECView model from the resource, serializes it using XMLResource and writes it to disk.

☞ **As an added benefit, the respective models can immediately be found in the XtextIndex (CTRL+SHIFT+F3) since both ui model and ECView model are generated before the ResourceDescriptions are created.**