

COMPEX COMMERCE

# **Process Specification**

## **DialogDSL**

**Version 1.0**

Date: 30.10.15

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>1.1</b>	<b>Vaadin</b>	<b>4</b>
<b>1.2</b>	<b>Vaaclipse</b>	<b>4</b>
<b>2</b>	<b>DialogDSL</b>	<b>4</b>
<b>2.1</b>	<b>Syntax</b>	<b>5</b>
2.1.1	Package definition	5
2.1.2	dialog	6

All rights are reserved by Compex Systemhaus GmbH. In particular, duplications, translations, microfilming, saving and processing in electronic systems are protected by copyright. Use of this manual is only authorized with the permission of Compex Systemhaus GmbH. Infringements of the law shall be punished in accordance with civil and penal laws.

We have taken utmost care in putting together texts and images. Nevertheless, the possibility of errors cannot be completely ruled out. The Figures and information in this manual are only given as approximations unless expressly indicated as binding. Amendments to the manual due to amendments to the standard software remain reserved. Please note that the latest amendments to the manual can be accessed through our helpdesk at any time. The contractually agreed regulations of the licensing and maintenance of the standard software shall apply with regard to liability for any errors in the documentation. Guarantees, particularly guarantees of quality or durability can only be assumed for the manual insofar as its quality or durability are expressly stipulated as guaranteed.

If you would like to make a suggestion, the Compex Team would be very pleased to hear from you.

© 2015 Compex Systemhaus GmbH

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 Vaclipse

Vaclipse 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. Vaclipse 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

### 2.1.1

## Syntax

### 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 : MempleadoDto

    verticalLayout {
        horizontalLayout {
            autore 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>+Vaclipse.java` file, in which a java class named `<dialogname>+Vaclipse` extended from java class `AbstractHybridVaclipseView` 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*.

**refreshingView** is automatically set

```
eventBroker.send(EventBrokerConstants.REFRESH_VIEW+"refreshing View String", null);
```

with every `handleEvent` in `subscribe()`.

► Example:

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

■ Employees Dialog:

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



## ■ 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	Harriss	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.**