

# VAADIN PROGRAMMING COOKBOOK

Hot Recipes for Vaadin Development

**vaadin** } >

**JESUS BOADAS**



**Java Code Geeks**  
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

# **Vaadin Programming Cookbook**

---

# Contents

<b>1</b>	<b>Vaadin Architecture Tutorial</b>	<b>1</b>
1.1	The tools . . . . .	1
1.2	Introduction . . . . .	1
1.3	Prerequisites . . . . .	1
1.4	Set up the project . . . . .	2
1.5	The Architecture . . . . .	3
1.5.1	The layers . . . . .	3
1.5.1.1	Client Widgets . . . . .	4
1.5.1.2	Web Server Layer . . . . .	4
1.5.1.3	Persistence Layer . . . . .	4
1.5.2	Our project explained . . . . .	5
1.5.2.1	Persistence . . . . .	5
1.5.2.2	Web Server Layer . . . . .	5
1.5.2.3	Client Tier . . . . .	8
1.6	Complete Source Code . . . . .	9
1.7	Running the example . . . . .	11
1.8	Results . . . . .	11
1.9	Download the Source Code . . . . .	12
<b>2</b>	<b>Vaadin Example Application</b>	<b>13</b>
2.1	The tools . . . . .	13
2.2	Introduction . . . . .	13
2.3	Prerequisites . . . . .	13
2.4	Set up the project . . . . .	14
2.5	Coding the example . . . . .	15
2.6	The complete source code . . . . .	21
2.7	Running the example . . . . .	25
2.8	Results . . . . .	26
2.9	Download the Source Code . . . . .	26

---

<b>3</b>	<b>Vaadin Best Practices</b>	<b>27</b>
3.1	The tools	27
3.2	Introduction	27
3.3	Prerequisites	27
3.4	Set up the project	27
3.5	The example	29
3.5.1	Make a design	29
3.5.2	Annotations	30
3.5.3	Navigator	30
3.5.3.1	Layout & content	30
3.5.3.2	Navigator views	31
3.5.3.3	Menu listeners	31
3.5.3.4	Menu	31
3.5.3.5	Navigator initial page	32
3.5.3.6	Welcome page	32
3.5.4	Validate user input	33
3.5.4.1	Input form	33
3.5.4.2	Name field validator	33
3.5.4.3	Surname field validator	33
3.5.4.4	Age field validator	34
3.5.4.5	Age field validator	34
3.5.4.6	Validation process	34
3.5.4.7	Clear fields	35
3.5.5	Use containers in fields	35
3.5.5.1	Property sets	36
3.5.5.2	Field groups	36
3.5.6	Separate the UI from the data	36
3.5.7	Deploy on https	38
3.6	The complete source code	38
3.6.1	VaadinbestpracticesUI.java	38
3.6.2	WelcomePage.java	39
3.6.3	InputPage.java	40
3.6.4	DataPage.java	42
3.6.5	DataBean.java	43
3.7	Running the example	43
3.8	Results	43
3.8.1	Welcome view	43
3.8.2	Input view	44
3.8.3	Input view	45
3.9	Download the Source Code	46

---

---

<b>4</b>	<b>Vaadin Visual Designer Example</b>	<b>47</b>
4.1	The tools . . . . .	47
4.2	Introduction . . . . .	47
4.3	Prerequisites . . . . .	47
4.4	Set up the project . . . . .	47
4.4.1	Get the Visual Designer . . . . .	47
4.4.2	Create the project . . . . .	49
4.5	Coding the example . . . . .	57
4.6	The complete source code . . . . .	61
4.7	Running the example . . . . .	63
4.8	Results . . . . .	64
4.9	Download the Source Code . . . . .	64
<b>5</b>	<b>Vaadin Data Binding Example</b>	<b>65</b>
5.1	The tools . . . . .	65
5.2	Introduction . . . . .	65
5.3	Prerequisites . . . . .	65
5.4	Set up the project . . . . .	65
5.5	Coding the example . . . . .	67
5.6	The complete source code . . . . .	72
5.7	Running the example . . . . .	75
5.8	Results . . . . .	75
5.9	Download the Source Code . . . . .	77
<b>6</b>	<b>Vaadin Rest Example</b>	<b>78</b>
6.1	Introduction . . . . .	78
6.2	Prerequisites . . . . .	78
6.3	Set up the server project . . . . .	78
6.3.1	Download Jersey . . . . .	78
6.3.2	Create the server project . . . . .	79
6.3.3	Copy Jersey Files . . . . .	80
6.4	Coding the server . . . . .	81
6.5	The server web.xml . . . . .	82
6.6	Create the client project . . . . .	83
6.7	Coding the client project . . . . .	84
6.7.1	Client Class to access the server . . . . .	84
6.7.2	Vaadin UI . . . . .	86
6.8	The complete source code . . . . .	87
6.8.1	The server source code . . . . .	87

---

---

6.8.2	The client Source Code . . . . .	88
6.9	Running the example . . . . .	91
6.10	Results . . . . .	92
6.10.1	Get the response . . . . .	92
6.10.2	Get the plain text . . . . .	93
6.10.3	Get the XML . . . . .	94
6.10.4	Get the JSON . . . . .	95
6.11	Download the Source Code . . . . .	95
<b>7</b>	<b>Vaadin Container Example</b>	<b>96</b>
7.1	The tools . . . . .	96
7.2	Introduction . . . . .	96
7.3	Prerequisites . . . . .	96
7.4	Set up the project . . . . .	97
7.5	Coding the example . . . . .	98
7.5.1	MyBean BeanContainer . . . . .	98
7.5.2	MySubBean BeanContainer . . . . .	99
7.5.3	The UI . . . . .	100
7.5.3.1	Random name generator . . . . .	100
7.5.3.2	The layout . . . . .	101
7.5.3.3	First BeanContainer . . . . .	101
7.5.3.4	Second BeanContainer . . . . .	101
7.5.3.5	Employee Table . . . . .	101
7.5.3.6	Doe's Table . . . . .	102
7.5.3.7	Dynamic input . . . . .	102
7.5.3.8	Click listener . . . . .	102
7.5.3.9	Find Duplicates . . . . .	103
7.6	The complete source code . . . . .	103
7.7	Running the example . . . . .	107
7.8	Results . . . . .	107
7.9	Download the Source Code . . . . .	111
<b>8</b>	<b>Vaadin Custom Component Example</b>	<b>112</b>
8.1	The tools . . . . .	112
8.2	Introduction . . . . .	112
8.3	Prerequisites . . . . .	112
8.4	Create the project . . . . .	112
8.5	Create the Vaadin custom component . . . . .	116
8.6	Review your project . . . . .	118
8.7	Compile the widgetset . . . . .	120
8.8	Run The project . . . . .	120
8.9	Application output . . . . .	123
8.10	Get the source code . . . . .	123

---

---

<b>9</b>	<b>Vaadin Server Push Example</b>	<b>124</b>
9.1	The tools . . . . .	124
9.2	Introduction . . . . .	124
9.3	Prerequisites . . . . .	124
9.4	Set up the project . . . . .	125
9.5	Coding the example . . . . .	126
9.5.1	The quote generator class . . . . .	126
9.5.2	Initial preparations . . . . .	127
9.5.3	First Thread . . . . .	127
9.5.4	Second Thread . . . . .	128
9.5.5	The init method . . . . .	128
9.6	The complete source code . . . . .	129
9.7	Running the example . . . . .	131
9.8	Results . . . . .	131
9.9	Download the Source Code . . . . .	134

---

Copyright (c) Exelixis Media P.C., 2016

All rights reserved. Without limiting the rights under copyright reserved above, no part of this publication may be reproduced, stored or introduced into a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), without the prior written permission of the copyright owner.

---



# Preface

Vaadin is an open source web framework for rich Internet applications. In contrast to JavaScript libraries and browser-plugin based solutions, it features a server-side architecture, which means that the majority of the logic runs on the servers. Ajax technology is used at the browser-side to ensure a rich and interactive user experience. On the client-side Vaadin is built on top of and can be extended with Google Web Toolkit.

Vaadin uses Java as the programming language for creating web content. The framework incorporates event-driven programming and widgets, which enables a programming model that is closer to GUI software development than traditional web development with HTML and JavaScript. (Source: <https://en.wikipedia.org/wiki/Vaadin>)

In this ebook, we provide a compilation of Vaadin programming examples that will help you kick-start your own projects. We cover a wide range of topics, from Architecture and Best Practices, to Data Binding and Custom Components. With our straightforward tutorials, you will be able to get your own projects up and running in minimum time.

---

## About the Author

Jesus is a self taught programmer who began programming back in 1991 using an IBM A10 mainframe with Pascal an Assembler IBM 360/70 emulator and Turbo C on a X86 PC. Since that he has been working for the banking industry with emerging technologies like Fox Pro, Visual Fox Pro, Visual Basic, Visual Cpp, Borland Cpp.

Lately he has moved out to the Airline industry, leading designing and programming in-house web applications with Flex, Actionscript, PHP, Python and Rails and in the last 7 years he has focused all his work in Java, working on Linux servers using GlassFish, TomCat, Apache and MySql.

---

## Chapter 1

# Vaadin Architecture Tutorial

The design is the most important part of a program, because a bad design produces bad software. A solid rock design increases the chance of getting good results, of course you still need good programmers but it's easier to correct clumsy code with a good design. Anyway how do you define code quality or clumsy code? If the code works and no bugs arise in the final product, could it be improved?. Is the language we are using the best to solve my problem?. Am i using the right platform to deploy my software?.

All these questions are the nightmare of software architects. The design produces the architecture of the software and the amount of parts my system has. The architecture refers to the high level structures of a software system, it's a discipline. Each structure has software elements. The architecture of a software system is similar to the architecture in buildings. Where the design is the blueprint of the building, the architecture is the technique used to build and each part of the building is a piece of software. In the architecture of the software it's recommended for a lot of architects to have a rigid division of concerns like Model-View-Presenter, Model-View-Controller, etc. But in the end if you are the architect of the system, all of these choices are up to you, just build your software in the best way you can and live with the consequences. In my own experience in pro of the peace of mind is better to use some kind of separation of concerns in the software architecture.

### 1.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.5
- Tomcat Server 8

### 1.2 Introduction

Vaadin Framework 7 is a server side web framework that uses Ajax to keep the UI synchronized with the server. Usually on each user interaction, Vaadin makes one or more Ajax calls against the server to "keep the server informed" whats going on with the UI. You can change this behavior but it's better to keep the server side nature of Vaadin. This design was done to secure your data, so it's better to keep it that way. Vaadin consists of three separate layers. In this tutorial we are going to see with a simple application how the Vaadin architecture works and where is every part of the architecture when you are coding.

### 1.3 Prerequisites

- JDK installed
  - Eclipse Mars installed and working
-

- Vaadin plug-in installed
- Tomcat 8 installed and running

## 1.4 Set up the project

In the file menu choose File → New → Other

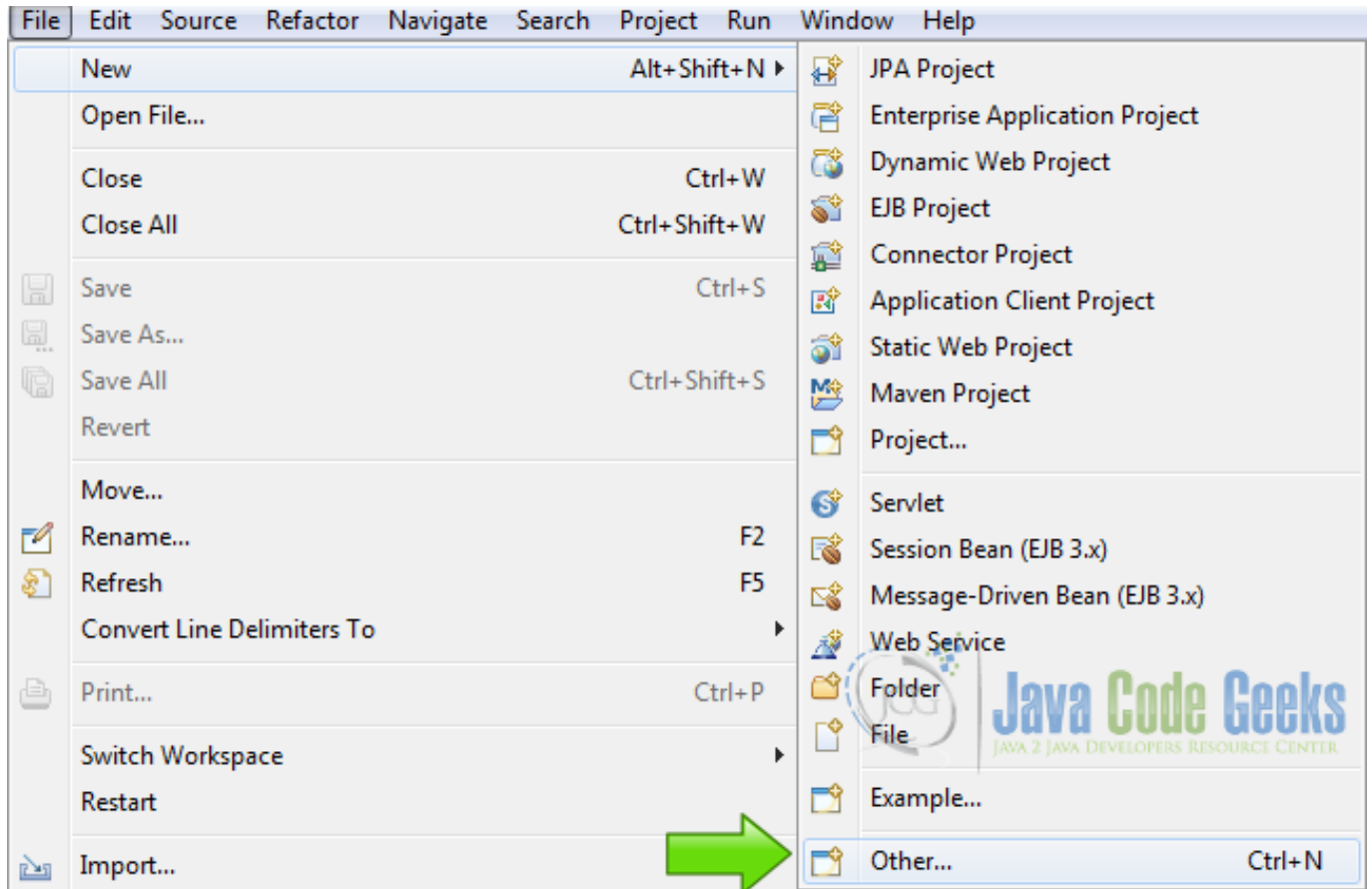


Figure 1.1: New Project

Now from the list choose Vaadin 7 project

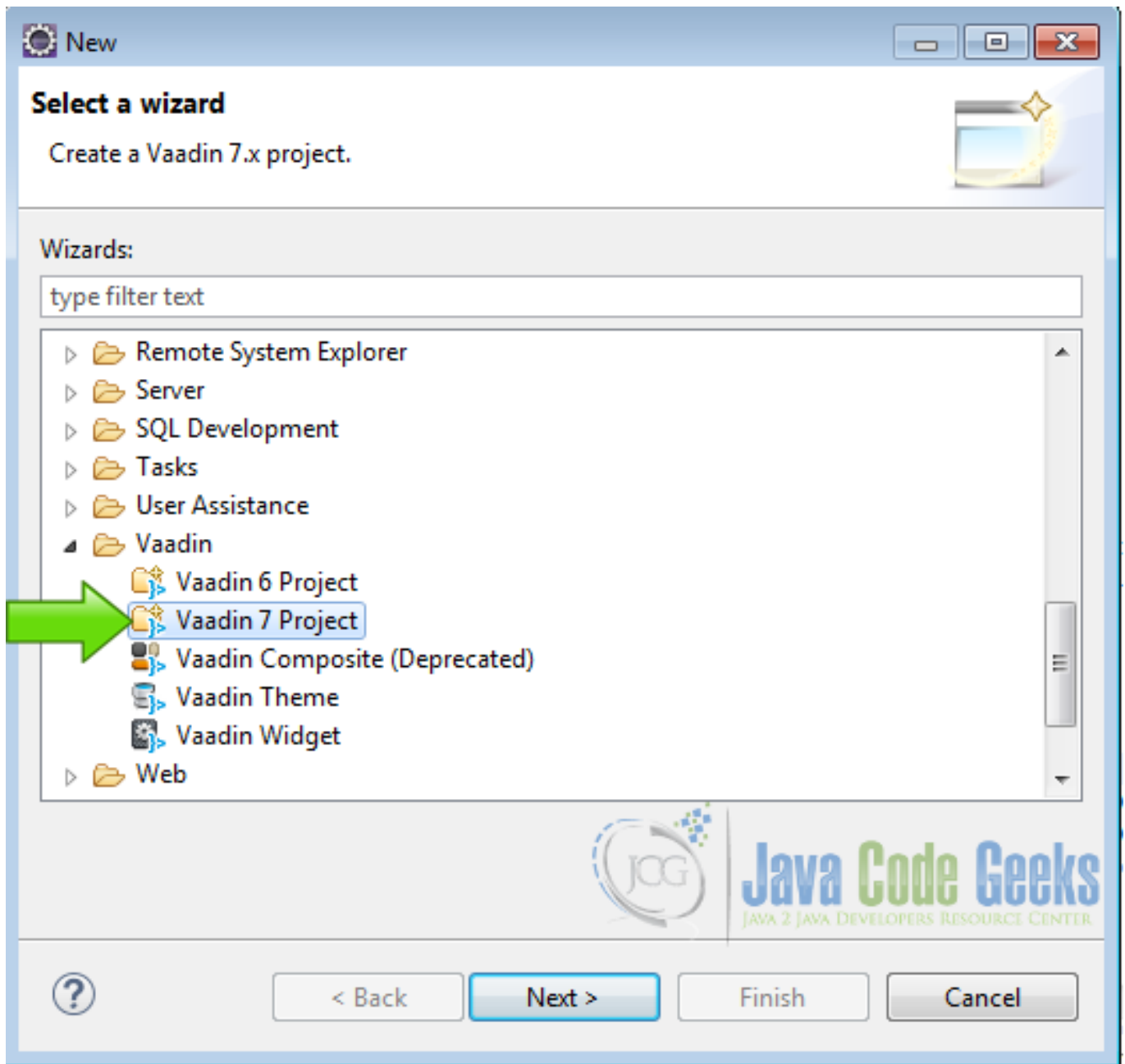


Figure 1.2: Vaadin Project

Hit next and name your project, then hit finish.

## 1.5 The Architecture

### 1.5.1 The layers

- Client widgets
- Web Server layer
- Persistence layer

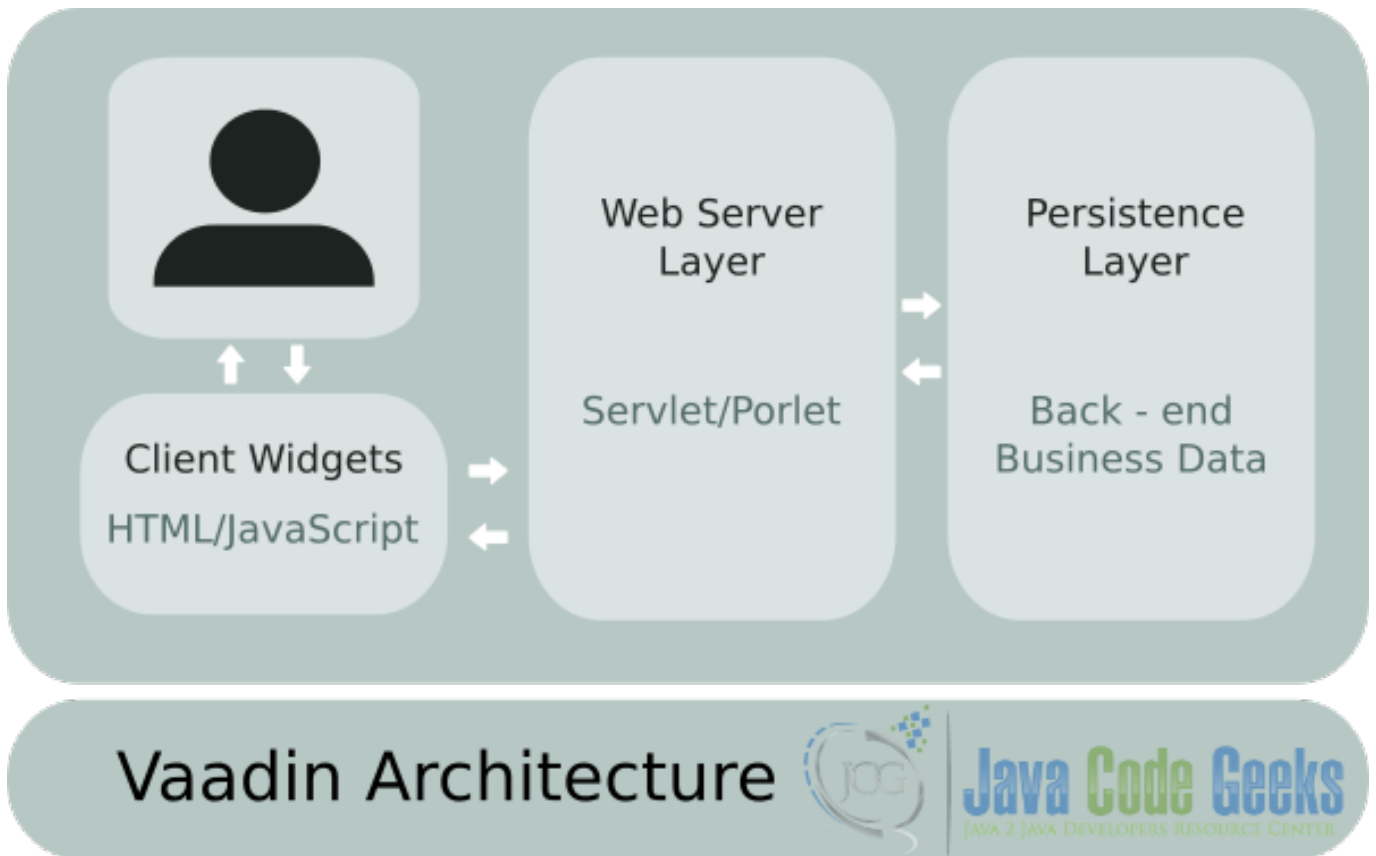


Figure 1.3: Vaadin Architecture

### 1.5.1.1 Client Widgets

The client widgets are the user interface elements that are shown in the browser. These widgets are a combination of HTML and JavaScript, the native execution environment for browsers. Vaadin makes extensive use of GWT - Google Widget Toolkit, while GWT is an open source set of tools that allows web developers to create complex JavaScript applications in Java. With Vaadin you can also create your widgets using HTML, JavaScript and CSS directly. The client widgets compose the user interface in a regular Vaadin application and usually each widget has a server counterpart that keeps the user interaction always server side, preventing data loss.

### 1.5.1.2 Web Server Layer

This layer is a Java servlet or a portlet that is in charge of intercept the requests of the widgets and send a response to update the UI. Moreover, it can make asynchronous calls to the widgets like server push calls to update the UI without user interaction.

**Java Servlet** A Java servlet is a Java program that runs within a Web server. Servlets usually use the Request-response message exchange pattern defined by the HyperText Transfer Protocol. Servlets are also capable of make asynchronous server push on any time required.

**Portlets** Portlets are pluggable user interface software components. A portlet UI is just like a regular Vaadin application, and is used like pieces of a web page with complete and concise functionality. Like a weather gadget that has a well known function, portlets are commonly used in enterprise portals like Liferay.

### 1.5.1.3 Persistence Layer

The persistence layer is in charge of the data in the application. This layer usually has an interface with a database or an in-memory datasets or a library to store data in the file-system or whatever persistence method you have. If your application needs

to store data this is the place where you put the code to manage all you need to save the data.

## 1.5.2 Our project explained

We created a Vaadin project to show the Vaadin layers.

### 1.5.2.1 Persistence

We are going to simulate persistence with an `ArrayList`, this is for the purpose of this tutorial. But you can plug here a database or write files on disk or connect to the cloud and save your data there.

Persistence

```
private ArrayList myArrayList;

public VaadinArchPersistence() {
    myArrayList = new ArrayList();
}

public void addItem(String item) {
    if(!item.isEmpty()) {
        myArrayList.add(item);
    }
}

public String getItems() {
    StringBuilder sb = new StringBuilder();
    Iterator myIterator = myArrayList.iterator();
    while(myIterator.hasNext()) {
        String element = myIterator.next();
        sb.append(element+" ");
    }
    return sb.toString();
}
```

We have here a private `ArrayList` called `myArrayList` to store our data, in this case a string.

A constructor `public VaadinArchPersistence()` to initialize our `ArrayList`, this only lasts until you refresh the page. On every page refresh all data is lost. A persistence container should save the data in a more durable media.

`public void addItem(String item)` is a method to add an item to our collection. This method also checks that the item is not empty. This is the place to validate your data, when you send the data to an external medium that data must be validated before. Also here you need to check the security to avoid database exploits, because a malicious user could find vulnerabilities in your application if you didn't put a filter between layers.

`public String getItems()` is a method to get all our items into a big string to show all items.

With that we can simulate a persistence fully working layer.

### 1.5.2.2 Web Server Layer

The Web Server layer is our servlet:

Vaadin Servlet

```
public class VaadinarchitectureUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinarchitectureUI.class ←
        , widgetset = "com.example.vaadinarchitecture.widgetset. ←
        VaadinarchitectureWidgetset")
```

```
public static class Servlet extends VaadinServlet {  
}
```

The Vaadin servlet extends `javax.servlet.http.HttpServlet` and implements `java.io.Serializable`, `javax.servlet.Servlet`, `javax.servlet.ServletConfig`, which is a subclass of the standard `HttpServlet`. Is in charge of processing the requests from the client. When a client requests a page in a Vaadin servlet, the first thing it does is to look the compiled version of that page and send it to the client. In a Vaadin application, exists multiple versions of the compiled client, one for each browser instructed to compile. That is a technique used by GWT to reduce the overhead. For example if you open the page with Firefox, the servlet only sends the compiled version of Firefox, and when you open the page with chrome you get a different version compiled and optimized for chrome. The GWT compiler uses the Google closure compiler to compile each version of the client widgets.

### Init

```
@Override  
protected void init(VaadinRequest request)
```

When you start a Vaadin application opening a web page the `VaadinServlet` calls the `Init` method. In the `Init` method we create the components. These components are server side with a client side counterpart. This example has a textbox to input text from the client to the persistence layer, a button to send the text in the textbox to the server, a button to retrieve all items added from the server and a label to show the items.

### The layout

```
final VerticalLayout layout = new VerticalLayout();  
layout.setMargin(true);  
setContent(layout);
```

Create the layout of the page.

Connect the layout to the persistence

```
VaadinArchPersistence vap = new VaadinArchPersistence();
```

Create an instance of the data-store.

### TextField

```
TextField tf = new TextField("Data");  
tf.setWidth("200px");
```

The `TextField` to input items.

### Label

```
Label lItems = new Label("");  
lItems.addStyleName("mylabelstyle");  
lItems.setWidth("200px");
```

The `Label` to show the items stored, we need to create `mylabelstyle` on the client CSS.

### Add Item

```
Button bAddItem = new Button("Add Item");  
bAddItem.setWidth("200px");  
bAddItem.addClickListener(new Button.ClickListener() {  
    public void buttonClick(ClickEvent event) {  
        vap.addItem(tf.getValue());  
        tf.clear();  
    }  
});
```



The Button to send the items to the server.

Show all

```
Button bShowItems = new Button("Show all items");
bShowItems.setWidth("200px");
bShowItems.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        lItems.setValue(vap.getItems());
    }
});
```

The Button to retrieve the items from the server.

Add components to the layout

```
layout.addComponent(tf);
layout.addComponent(bAddItem);
layout.addComponent(bShowItems);
layout.addComponent(lItems);
```

Add the items to the layout.

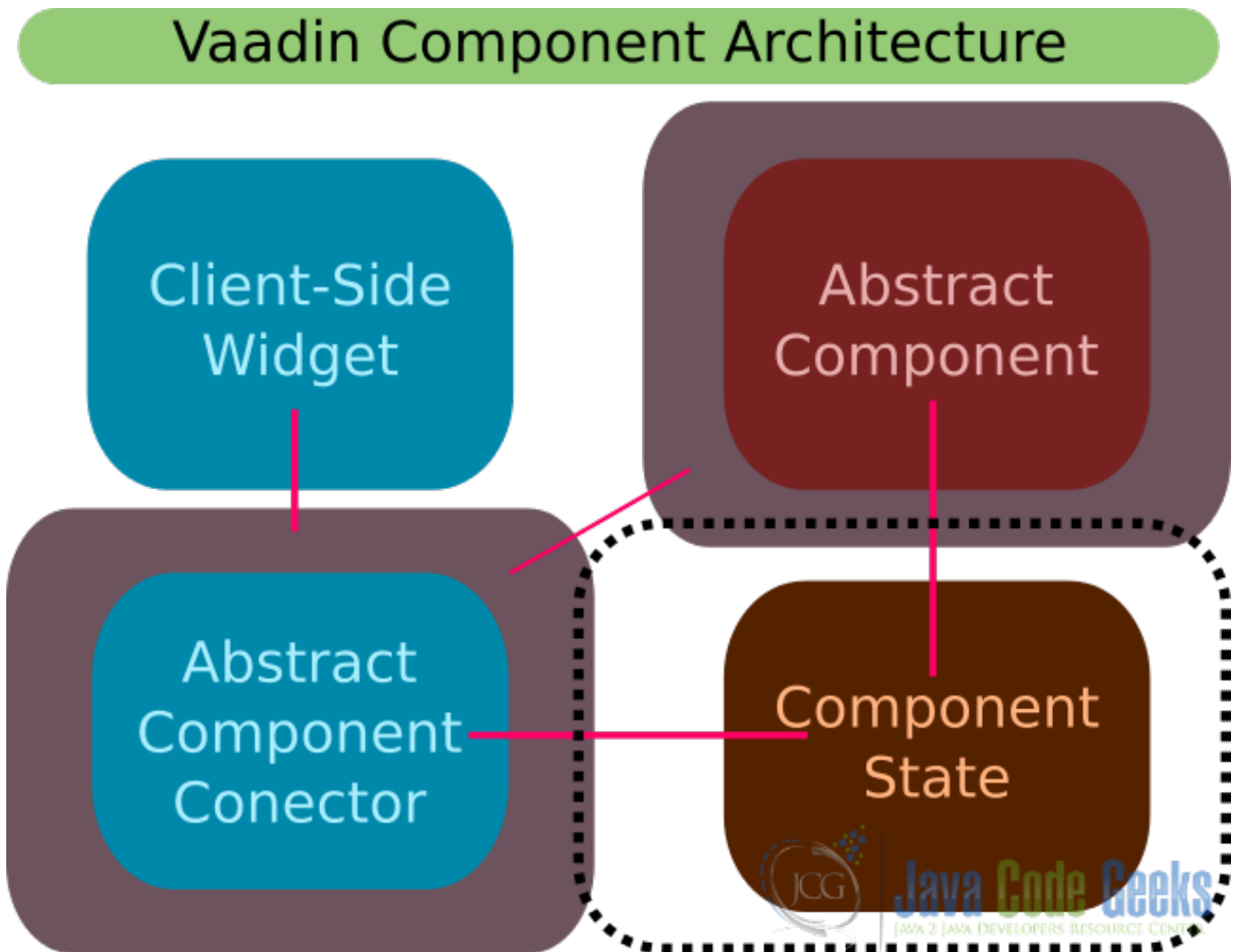


Figure 1.4: Vaadin Component Architecture

The Abstract component shares the state with the abstract component connector in order to keep the state synchronized between the client and the server part of the component/widget. GWT creates the client widget with the closure compiler. The widget calls the Abstract component connector. The abstract component connector then updates the state and make Ajax calls to the Abstract component that is server side.

### 1.5.2.3 Client Tier

The client uses the GWT compiler to convert the Java code into JavaScript and the JavaScript is also compiled with the Google closure compiler to optimize it. Now let's compile the widgetset. Click on the Vaadin toolbar menu and compile the widgetset:

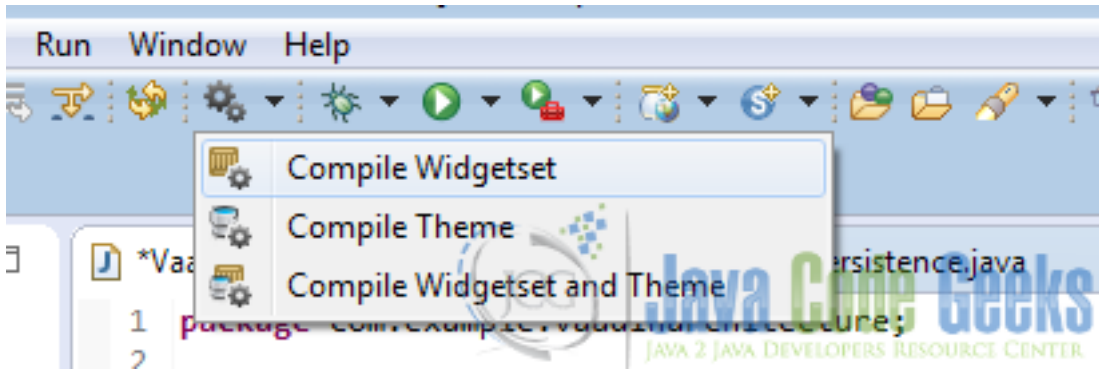


Figure 1.5: Compile widgetset

Open the folder WebContent → Widgetsets

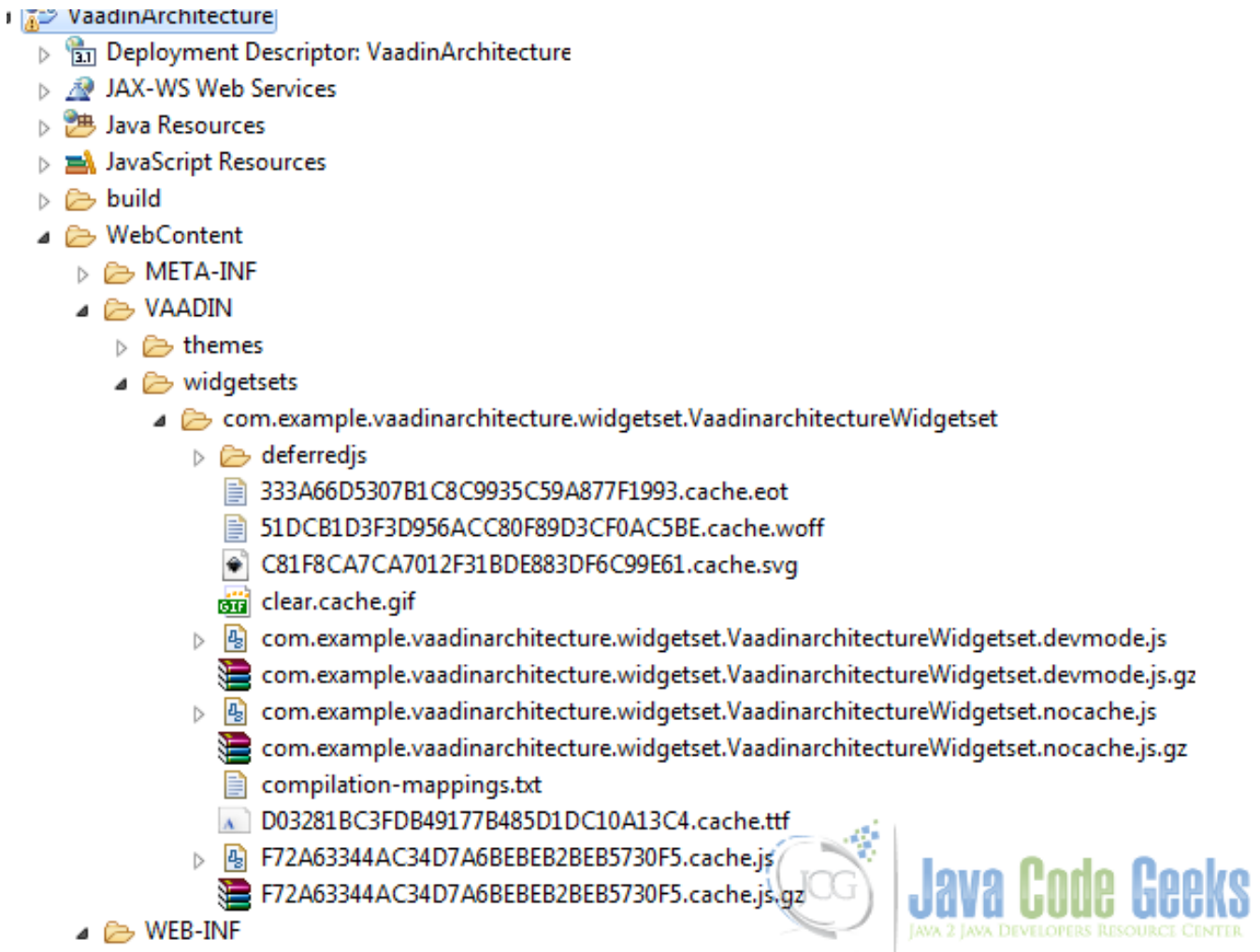


Figure 1.6: Widgetsets

In this folder you have the compiled widget-sets into JavaScript, you have a version for each browser supported and you also have "gz" compressed versions to send it instead when is supported. Vaadin take in charge of all these details for you. So you only need to know how to write the Java code and forget about these details until you need to write your own components.

## 1.6 Complete Source Code

VaadinarchitectureUI.java

```
package com.example.vaadinarchitecture;

import javax.servlet.annotation.WebServlet;

import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Label;
import com.vaadin.ui.TextField;
```

```
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadinarchitecture")
public class VaadinarchitectureUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinarchitectureUI.class ←
        , widgetset = "com.example.vaadinarchitecture.widgetset. ←
        VaadinarchitectureWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);
        VaadinArchPersistence vap = new VaadinArchPersistence();

        TextField tf = new TextField("Data");
        tf.setWidth("200px");

        Label lItems = new Label("");
        lItems.addStyleName("mylabelstyle");
        lItems.setWidth("200px");

        Button bAddItem = new Button("Add Item");
        bAddItem.setWidth("200px");
        bAddItem.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                vap.addItem(tf.getValue());
                tf.clear();
            }
        });

        Button bShowItems = new Button("Show all items");
        bShowItems.setWidth("200px");
        bShowItems.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                lItems.setValue(vap.getItems());
            }
        });

        layout.addComponent(tf);
        layout.addComponent(bAddItem);
        layout.addComponent(bShowItems);
        layout.addComponent(lItems);
    }
}
```

#### VaadinArchPersistence.java

```
package com.example.vaadinarchitecture;

import java.util.ArrayList;
import java.util.Iterator;

public class VaadinArchPersistence {
```

```
private ArrayList myArrayList;

public VaadinArchPersistence() {
    myArrayList = new ArrayList();
}

public void addItem(String item) {
    if(!item.isEmpty()) {
        myArrayList.add(item);
    }
}

public String getItems() {
    StringBuilder sb = new StringBuilder();
    Iterator myIterator = myArrayList.iterator();
    while(myIterator.hasNext()) {
        String element = myIterator.next();
        sb.append(element+" ");
    }
    return sb.toString();
}
}
```

vaadinarchitecture.scss

```
@import "../valo/valo.scss";

@mixin vaadinarchitecture {
    @include valo;

    // Insert your own theme rules here

    .v-label-mylabelstyle {
        color: white;
        text-align: left;
        background-color: black;
        border-color: white;
        font-weight: bold;
    }
}
```

## 1.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 1.8 Results

As you run the application you get a text box with two buttons



Figure 1.7: Run Application

Press the CONTROL+SHIFT+i key combination in the browser window to get to the console. Locate the Network tab and press the buttons of the application. As you can see, every time you press a button the client tier makes an Ajax call to the server.

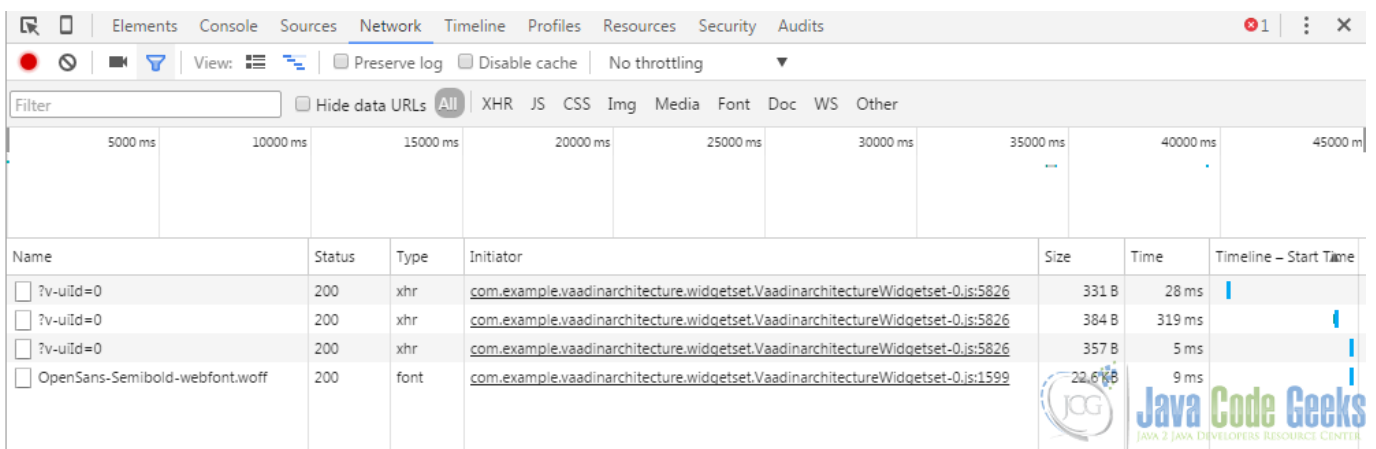


Figure 1.8: Ajax Calls

## 1.9 Download the Source Code

This was a tutorial of: Vaadin Architecture.

### Download

You can download the Eclipse project here: [VaadinArchitecture](#)

## Chapter 2

# Vaadin Example Application

An application is a computer program created to perform a group of useful tasks for an end user. Application could be a vast topic to cover, just because a simple "Hello World" program is an application. Most applications do much more than only print a hello message on the user interface.

The "Hello world" application is useful as an initial program when you are learning a new programming language to illustrate the basic syntax for constructing a working program and get it to run on a computer. The first "Hello World" application was created by Brian Kernigham in a internal memorandum of Bell Laboratories "Programming in C a tutorial".

### 2.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.6
- Tomcat Server 8

### 2.2 Introduction

In this example we are going to create a sample Vaadin application to show common widgets interacting with each other. We generate a character sheet for an RPG game using random values. If the values appeal to us, we could save them for later use. This application could be useful to you if you play tabletop RPG games but further than that, is an application to illustrate some common Vaadin widgets working together.

### 2.3 Prerequisites

- JDK installed
  - Eclipse Mars installed and working
  - Vaadin plug-in installed
  - Tomcat 8 installed and running
-

## 2.4 Set up the project

In the file menu choose File → New → Other

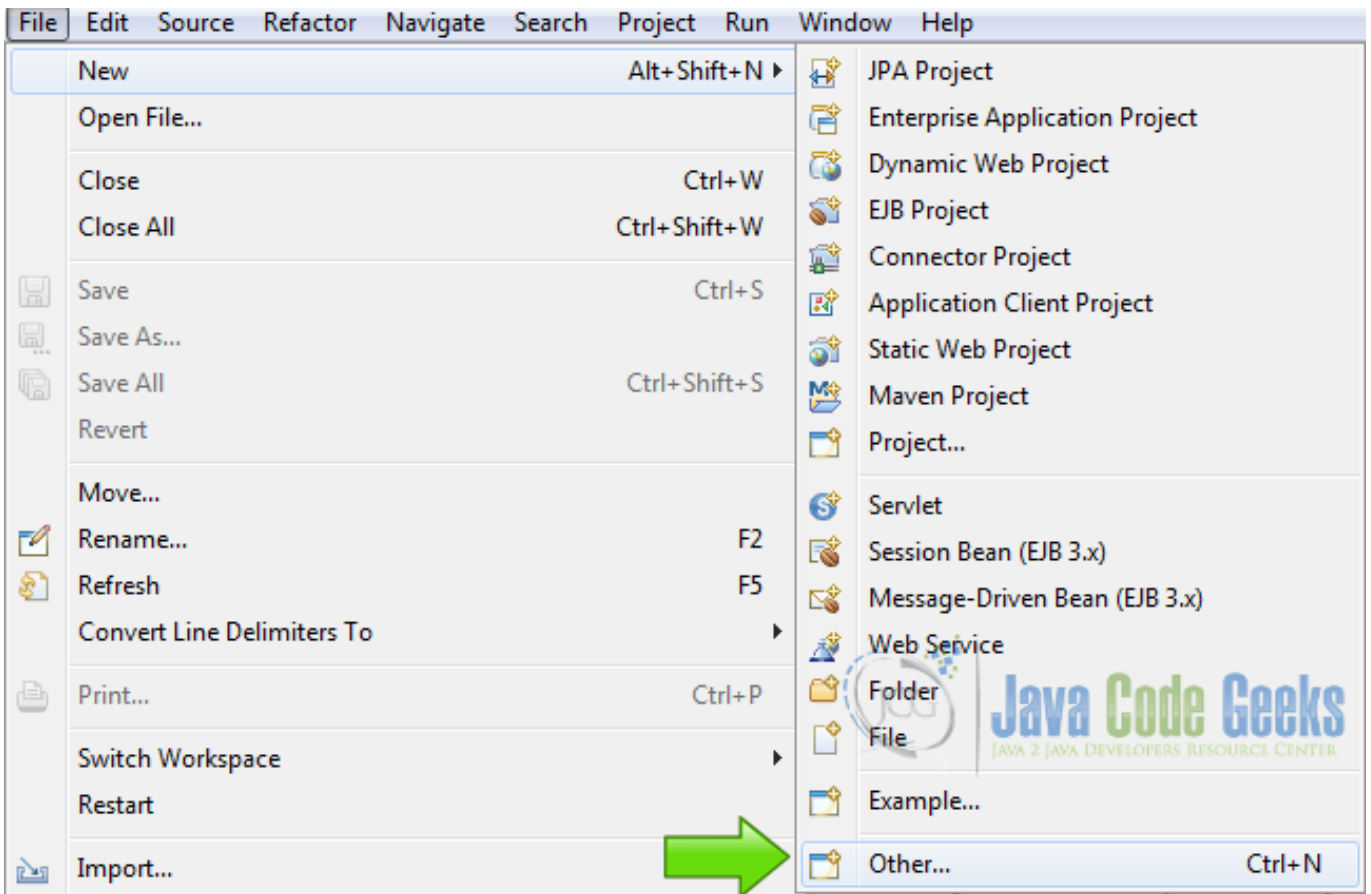


Figure 2.1: New Project

Now from the list choose Vaadin 7 project



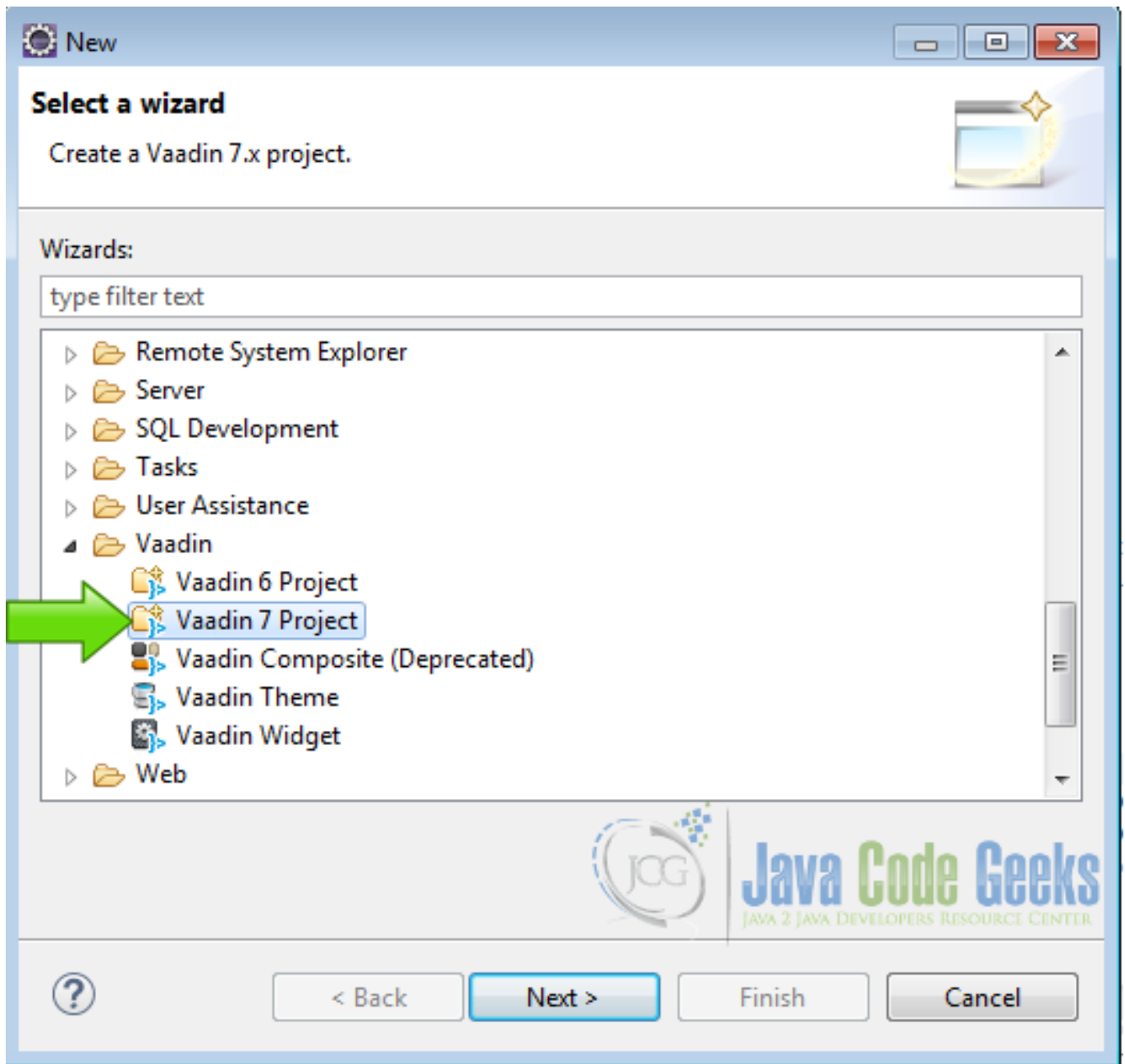


Figure 2.2: Vaadin Project

Hit next and name your project then hit finish.

## 2.5 Coding the example

Class variables

```
private Table attr;  
private Table savedAttr;  
private Item str;  
private Item con;  
private Item nte;
```

```
private Item agi;
private Item wis;
private Random rnd;
```

We declared two tables and five items to manage our data, also a random instance to generate random numbers.

### Layouts

```
final VerticalLayout layout = new VerticalLayout();
layout.setMargin(true);
setContent(layout);

HorizontalLayout topHorLay = new HorizontalLayout();
topHorLay.setSizeFull();
topHorLay.setSpacing(true);

VerticalLayout butLay = new VerticalLayout();
butLay.setSizeFull();
butLay.setSpacing(true);
Label butLabel = new Label("Is this OK?");
butLabel.setStyleName("h2");
```

We create the layouts, a main vertical layout, a horizontal layout for the first row of the main layout and a vertical layout to place the buttons of the application.

### CellStyleGenerator

```
CellStyleGenerator myCellColors = new CellStyleGenerator() {

    @Override
    public String getStyle(Table source, Object itemId, Object propertyId) {
        if(propertyId!=null){
            Item item = source.getItem(itemId);
            if(item.getItemProperty(propertyId).getValue().getClass()==Integer.class){
                Integer cellValue = (Integer)item.getItemProperty(
                    propertyId).getValue();
                if(cellValue > 15 && cellValue < 17){
                    return "green";
                }
                if(cellValue < 8){
                    return "red";
                }
            }
        }
        return null;
    }
};
```

This `CellStyleGenerator` changes the color of individual cells based on it's value. First we check that the cell is no empty with `if(propertyId!=null)`. Then we get the `Item` in the cell with `Item item =source.getItem(itemId);`.

We then Check the value of the `Item` and verify that is an integer with `if(item.getItemProperty(propertyId).getValue().getClass()==Integer.class)`.

If the value is an `Integer`, we cast it to a local variable with `Integer cellValue = (Integer)item.getItemProperty(propertyId).getValue();` and based on the value of the variable we set the colors of the cells in the table.

### Attributes Table

```
attr = new Table("Attributes");
attr.addContainerProperty("Attribute", String.class, null);
attr.addContainerProperty("Value", Integer.class, 0);
```

```
attr.setPageLength(attr.size());
attr.setSelectable(false);
attr.setImmediate(true);
attr.setFooterVisible(false);
attr.setCellStyleGenerator(myCellColors);
```

This table is going to have the generated values, one string column with the name of the attribute and an integer column holding the value of that attribute. We create the table with `attr =new Table("Attributes");`. `attr` is a class field. `attr.addContainerProperty("Attribute", String.class, null);` defines the first value data type as string.

`attr.addContainerProperty("Value", Integer.class, 0);` defines the second column data type as integer. With `attr.setPageLength(attr.size());` we constrain the size of the table on the screen to the size of the data contained on it.

`attr.setSelectable(false);`, the user cannot select the rows on the table.`attr.setImmediate(true);` set immediate tells the application to send any changes on the table to the server, when they occur. Hide the footer of the table `attr.setFooterVisible(false);`.

And tell the table to use our custom cell style generator `attr.setCellStyleGenerator(myCellColors);`.

### Helper initialization

```
rnd = new Random();
createCharacterTable();
```

`rnd =new Random();` initializes the random number generator. `createCharacterTable();`, calls a custom method to fill the table with the initial values. This is explained later on this article.

### Table of saved attributes

```
savedAttr = new Table("Saved Rolls");
savedAttr.setSizeFull();
savedAttr.addContainerProperty("Name", String.class, "");
savedAttr.addContainerProperty("Strength", Integer.class, 0);
savedAttr.addContainerProperty("Constitution", Integer.class, 0);
savedAttr.addContainerProperty("Intelligence", Integer.class, 0);
savedAttr.addContainerProperty("Agility", Integer.class, 0);
savedAttr.addContainerProperty("Wisdom", Integer.class, 0);
savedAttr.setCellStyleGenerator(myCellColors);
```

This table is going to have the saved data of our application.`savedAttr =new Table("Saved Rolls");` creates the table.`savedAttr.setSizeFull();` sets the size of the table to fit all the space available.

- `savedAttr.addContainerProperty("Name", String.class, "");`, adds a string column to hold the name of our character.
- `savedAttr.addContainerProperty("Strenght", Integer.class, 0);` adds an integer column to hold the strength attribute.
- `savedAttr.addContainerProperty("Constitution", Integer.class, 0);`, adds an integer column to hold the constitution attribute.
- `savedAttr.addContainerProperty("Intelligence", Integer.class, 0);`, adds an integer column to hold the intelligence attribute.
- `savedAttr.addContainerProperty("Agility", Integer.class, 0);` adds an integer column to hold the agility attribute.
- `savedAttr.addContainerProperty("Wisdom", Integer.class, 0);` adds the column of the wisdom attribute.
- `savedAttr.setCellStyleGenerator(myCellColors);` attaches the custom cell styles to the table.

## Button reroll

```

Button reroll = new Button("Reroll");
reroll.setWidth("200px");
reroll.setIcon(FontAwesome.DIAMOND);
reroll.addStyleName("friendly");
reroll.addClickListener(new ClickListener() {

    @SuppressWarnings("unchecked")
    @Override
    public void buttonClick(ClickEvent event) {
        str.getItemProperty("Value").setValue(getReroll());
        con.getItemProperty("Value").setValue(getReroll());
        nte.getItemProperty("Value").setValue(getReroll());
        agi.getItemProperty("Value").setValue(getReroll());
        wis.getItemProperty("Value").setValue(getReroll());
    }
});

```

With this button we randomize all attributes with new values. `Button reroll =new Button("Reroll");` creates the button. `reroll.setWidth("200px");` sets the width of the button to "200px". `reroll.setIcon(FontAwesome.DIAMOND);` assigns an icon to the button using fontawesome that is shipped with Vaadin out of the box.

`reroll.addStyleName("friendly");` adds the Vaadin friendly style to the button, this style is predefined in Vaadin. `reroll.addClickListener(new ClickListener())` adds the listener to the button, inside the listener we call `str.getItemProperty("Value").setValue(getReroll());` to get a new strength value using `getReroll()` method that is explained later.

The same procedure is done with all attributes in the table to get new random values.

## Name TextField

```

TextField name = new TextField("Character Name");
name.setRequired(true);
name.setWidth("200px");

```

With `TextField name =new TextField("Character Name");` we create the text field to input the name of the character. `name.setRequired(true);` sets the field required. `name.setWidth("200px");` sets the width of the field the same as the buttons.

## Button save

```

Button save = new Button("Save");
save.addStyleName("primary");
save.setIcon(FontAwesome.SAVE);
save.setWidth("200px");
save.addClickListener(new ClickListener()

```

Create a new button with `Button save =new Button("Save");`. `save.addStyleName("primary");` assigns style primary to the button. `save.setIcon(FontAwesome.SAVE);` sets an icon for font awesome.

`save.setWidth("200px");` sets the width to the button to 200 pixels. `save.addClickListener(new ClickListener())` adds a click listener to the button.

## Save click listener

```

public void buttonClick(ClickEvent event) {
    int istr = (int) str.getItemProperty("Value").getValue();
    int icon = (int) con.getItemProperty("Value").getValue();
    int inte = (int) nte.getItemProperty("Value").getValue();
    int iagi = (int) agi.getItemProperty("Value").getValue();
    int iwis = (int) wis.getItemProperty("Value").getValue();

```

```

        if((istr != 0) && (icon != 0) && (inte != 0) && (iagi != 0) && (iwis != 0)){
            if(name.isEmpty()){
                Notification.show("The name is required");
            }else{
                String cName = name.getValue();
                if(findDuplicates(cName)){
                    Notification.show("Name is duplicated");
                }else{
                    Object savedAttrId = savedAttr.addItem();
                    Item nSavAttr = savedAttr.getItem(savedAttrId);
                    nSavAttr.getItemProperty("Name").setValue(cName);
                    nSavAttr.getItemProperty("Strenght").setValue(istr);
                    nSavAttr.getItemProperty("Constitution").setValue(icon);
                    nSavAttr.getItemProperty("Intelligence").setValue(inte);
                    nSavAttr.getItemProperty("Agility").setValue(iagi);
                    nSavAttr.getItemProperty("Wisdom").setValue(iwis);
                    name.setValue("");
                    Notification.show("Character saved!");
                }
            }
        }else{
            Notification.show("You must generate a character first");
        }
    }
}

```

When the save button is clicked the click listener is called by the framework. `int istr = (int) str.getItemProperty("Value").getValue();` gets the value of strength and cast it to an integer. `int icon = (int) con.getItemProperty("Value").getValue();` gets the value of constitution and cast it to an integer. `int inte = (int) nte.getItemProperty("Value").getValue();` gets the value of intelligence and cast it to an integer.

`int iagi = (int) agi.getItemProperty("Value").getValue();` gets the value of agility and cast it to an integer. `int iwis = (int) wis.getItemProperty("Value").getValue();` gets the value of wisdom and cast it to an integer. `if((istr != 0) && (icon != 0) && (inte != 0) && (iagi != 0) && (iwis != 0))` checks the values has been generated.

Otherwise a notification is send to the user. `name.isEmpty()` checks the name is no empty. If the name is empty a notification is triggered. `String cName = name.getValue()` gets the name into a string variable. `findDuplicates(cName)` check if the name is already used.

When all conditions are meet then we proceed to save the character into the saved rolls table. `Object savedAttrId = savedAttr.addItem();` creates a new object from the saved attributes table. `Item nSavAttr = savedAttr.getItem(savedAttrId);` gets the item from the container.

`nSavAttr.getItemProperty("Name").setValue(cName);` sets the value of the name into the saved attribute container, also we set all other properties from the container with the values we already have.

`nSavAttr.getItemProperty("Strenght").setValue(istr);` sets the strength. `nSavAttr.getItemProperty("Constitution").setValue(icon);` sets the constitution.

`nSavAttr.getItemProperty("Intelligence").setValue(inte);` sets the intelligence. `nSavAttr.getItemProperty("Agility").setValue(iagi);` sets the agility. `nSavAttr.getItemProperty("Wisdom").setValue(iwis);`.

Now we set the name text field to an empty string and send a notification to the user indicating that the character has been saved.

#### Add the widgets to the layouts

```

butLay.addComponent(butLabel);
butLay.addComponent(reroll);
butLay.addComponent(save);
butLay.addComponent(name);

topHorLay.addComponent(attr);

```

```
topHorLay.addComponent(butLay);

layout.addComponent(topHorLay);
layout.addComponent(savedAttr);
```

We add the buttons to the vertical `butLay` layout and then we add the `attr` table and the `butLay` to the horizontal `topHorLay` layout. Finally we add the `topHorLay` layout and the `savedAttr` table to the main layout.

`createCharacterTable` method

```
private void createCharacterTable()
{
    Object strItemId = attr.addItem();
    str = attr.getItem(strItemId);
    str.getItemProperty("Attribute").setValue("STR");
    str.getItemProperty("Value").setValue(0);

    Object conItemId = attr.addItem();
    con = attr.getItem(conItemId);
    con.getItemProperty("Attribute").setValue("CON");
    con.getItemProperty("Value").setValue(0);

    Object nteItemId = attr.addItem();
    nte = attr.getItem(nteItemId);
    nte.getItemProperty("Attribute").setValue("INT");
    nte.getItemProperty("Value").setValue(0);

    Object agiItemId = attr.addItem();
    agi = attr.getItem(agiItemId);
    agi.getItemProperty("Attribute").setValue("AGI");
    agi.getItemProperty("Value").setValue(0);

    Object wisItemId = attr.addItem();
    wis = attr.getItem(wisItemId);
    wis.getItemProperty("Attribute").setValue("WIS");
    wis.getItemProperty("Value").setValue(0);
}
```

This method populate the `attr` table with two columns attribute and value and each row is an attribute used.

`getReroll` method

```
private int getReroll() {
    return rnd.nextInt(19)+1;
}
```

This method generates random integers to get the values of the attributes.

`findDuplicates` method

```
private boolean findDuplicates(String newName) {
    for(Iterator i = savedAttr.getItemIds().iterator(); i.hasNext();){

        int iid = (Integer) i.next();
        Item item = savedAttr.getItem(iid);
        String curName = (String) item.getItemProperty("Name").getValue();

        if(newName.toLowerCase().equals(curName.toLowerCase())){
            return true;
        }
    }
}
```

```
        return false;
    }
```

This method returns true if a name already exist in the saved attributes table.

## 2.6 The complete source code

VaadinappexampleUI.java

```
package com.example.vaadinappexample;

import javax.servlet.annotation.WebServlet;

import java.util.Iterator;
import java.util.Random;
import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.data.Item;
import com.vaadin.server.FontAwesome;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Label;
import com.vaadin.ui.Notification;
import com.vaadin.ui.Table;
import com.vaadin.ui.Table.CellStyleGenerator;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;

@SuppressWarnings("serial")
@Theme("vaadinappexample")
public class VaadinappexampleUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinappexampleUI.class, ←
        widgetset = "com.example.vaadinappexample.widgetset.VaadinappexampleWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    private Table attr;
    private Table savedAttr;
    private Item str;
    private Item con;
    private Item nte;
    private Item agi;
    private Item wis;
    private Random rnd;

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);

        HorizontalLayout topHorLay = new HorizontalLayout();
        topHorLay.setSizeFull();
    }
}
```

```

topHorLay.setSpacing(true);

VerticalLayout butLay = new VerticalLayout();
butLay.setSizeFull();
butLay.setSpacing(true);
Label butLabel = new Label("Is this OK?");
butLabel.setStyleName("h2");

CellStyleGenerator myCellColors = new CellStyleGenerator() {

    @Override
    public String getStyle(Table source, Object itemId, Object ←
        propertyId) {
        if(propertyId!=null){
            Item item = source.getItem(itemId);
            if(item.getItemProperty(propertyId).getValue(). ←
                getClass()==Integer.class){
                Integer cellValue = (Integer)item. ←
                    getItemProperty(propertyId).getValue();
                if(cellValue > 15 && cellValue < 17){
                    return "green";
                }
                if(cellValue < 8){
                    return "red";
                }
            }
        }
        return null;
    }
};

attr = new Table("Attributes");
attr.addContainerProperty("Attribute", String.class, null);
attr.addContainerProperty("Value", Integer.class, 0);
attr.setPageLength(attr.size());
attr.setSelectable(false);
attr.setImmediate(true);
attr.setFooterVisible(false);
attr.setCellStyleGenerator(myCellColors);

rnd = new Random();
createCharacterTable();

savedAttr = new Table("Saved Rolls");
savedAttr.setSizeFull();
savedAttr.addContainerProperty("Name", String.class, "");
savedAttr.addContainerProperty("Strength", Integer.class, 0);
savedAttr.addContainerProperty("Constitution", Integer.class, 0);
savedAttr.addContainerProperty("Intelligence", Integer.class, 0);
savedAttr.addContainerProperty("Agility", Integer.class, 0);
savedAttr.addContainerProperty("Wisdom", Integer.class, 0);
savedAttr.setCellStyleGenerator(myCellColors);

Button reroll = new Button("Reroll");
reroll.setWidth("200px");
reroll.setIcon(FontAwesome.DIAMOND);
reroll.addStyleName("friendly");
reroll.addClickListener(new ClickListener() {

    @SuppressWarnings("unchecked")
    @Override
    public void buttonClick(ClickEvent event) {

```



```

        str.getItemProperty("Value").setValue(getReroll());
        con.getItemProperty("Value").setValue(getReroll());
        nte.getItemProperty("Value").setValue(getReroll());
        agi.getItemProperty("Value").setValue(getReroll());
        wis.getItemProperty("Value").setValue(getReroll());

    }

});

TextField name = new TextField("Character Name");
name.setRequired(true);
name.setWidth("200px");

Button save = new Button("Save");
save.addStyleName("primary");
save.setIcon(FontAwesome.SAVE);
save.setWidth("200px");
save.addClickListener(new ClickListener() {

    @SuppressWarnings("unchecked")
    @Override
    public void buttonClick(ClickEvent event) {
        int istr = (int) str.getItemProperty("Value").getValue();
        int icon = (int) con.getItemProperty("Value").getValue();
        int inte = (int) nte.getItemProperty("Value").getValue();
        int iagi = (int) agi.getItemProperty("Value").getValue();
        int iwis = (int) wis.getItemProperty("Value").getValue();

        if((istr != 0) && (icon != 0) && (inte != 0) && (iagi != 0) && (iwis != 0)){
            if(name.isEmpty()){
                Notification.show("The name is required");
            }else{
                String cName = name.getValue();
                if(findDuplications(cName)){
                    Notification.show("Name is duplicated");
                }else{
                    Object savedAttrId = savedAttr.addItem();
                    Item nSavAttr = savedAttr.getItem(savedAttrId);
                    nSavAttr.getItemProperty("Name").setValue(cName);
                    nSavAttr.getItemProperty("Strength").setValue(istr);
                    nSavAttr.getItemProperty("Constitution").setValue(icon);
                    nSavAttr.getItemProperty("Intelligence").setValue(inte);
                    nSavAttr.getItemProperty("Agility").setValue(iagi);
                    nSavAttr.getItemProperty("Wisdom").setValue(iwis);
                    name.setValue("");
                    Notification.show("Character saved!");
                }
            }
        }else{
            Notification.show("You must generate a character first");
        }
    }
});

```

```
        }

        });

        butLay.addComponent (butLabel);
        butLay.addComponent (reroll);
        butLay.addComponent (save);
        butLay.addComponent (name);

        topHorLay.addComponent (attr);
        topHorLay.addComponent (butLay);

        layout.addComponent (topHorLay);
        layout.addComponent (savedAttr);
    }

    @SuppressWarnings ("unchecked")
    private void createCharacterTable ()
    {
        Object strItemId = attr.addItem ();
        str = attr.getItem (strItemId);
        str.getItemProperty ("Attribute").setValue ("STR");
        str.getItemProperty ("Value").setValue (0);

        Object conItemId = attr.addItem ();
        con = attr.getItem (conItemId);
        con.getItemProperty ("Attribute").setValue ("CON");
        con.getItemProperty ("Value").setValue (0);

        Object nteItemId = attr.addItem ();
        nte = attr.getItem (nteItemId);
        nte.getItemProperty ("Attribute").setValue ("INT");
        nte.getItemProperty ("Value").setValue (0);

        Object agiItemId = attr.addItem ();
        agi = attr.getItem (agiItemId);
        agi.getItemProperty ("Attribute").setValue ("AGI");
        agi.getItemProperty ("Value").setValue (0);

        Object wisItemId = attr.addItem ();
        wis = attr.getItem (wisItemId);
        wis.getItemProperty ("Attribute").setValue ("WIS");
        wis.getItemProperty ("Value").setValue (0);

    }

    private int getReroll () {
        return rnd.nextInt (19)+1;
    }

    private boolean findDuplicates (String newName) {
        for (Iterator i = savedAttr.getItemIds ().iterator (); i.hasNext ();) {

            int iid = (Integer) i.next ();
            Item item = savedAttr.getItem (iid);
            String curName = (String) item.getItemProperty ("Name").getValue ();

            if (newName.toLowerCase ().equals (curName.toLowerCase ())) {
                return true;
            }
        }
    }
}
```

```
        }  
        return false;  
    }  
}
```

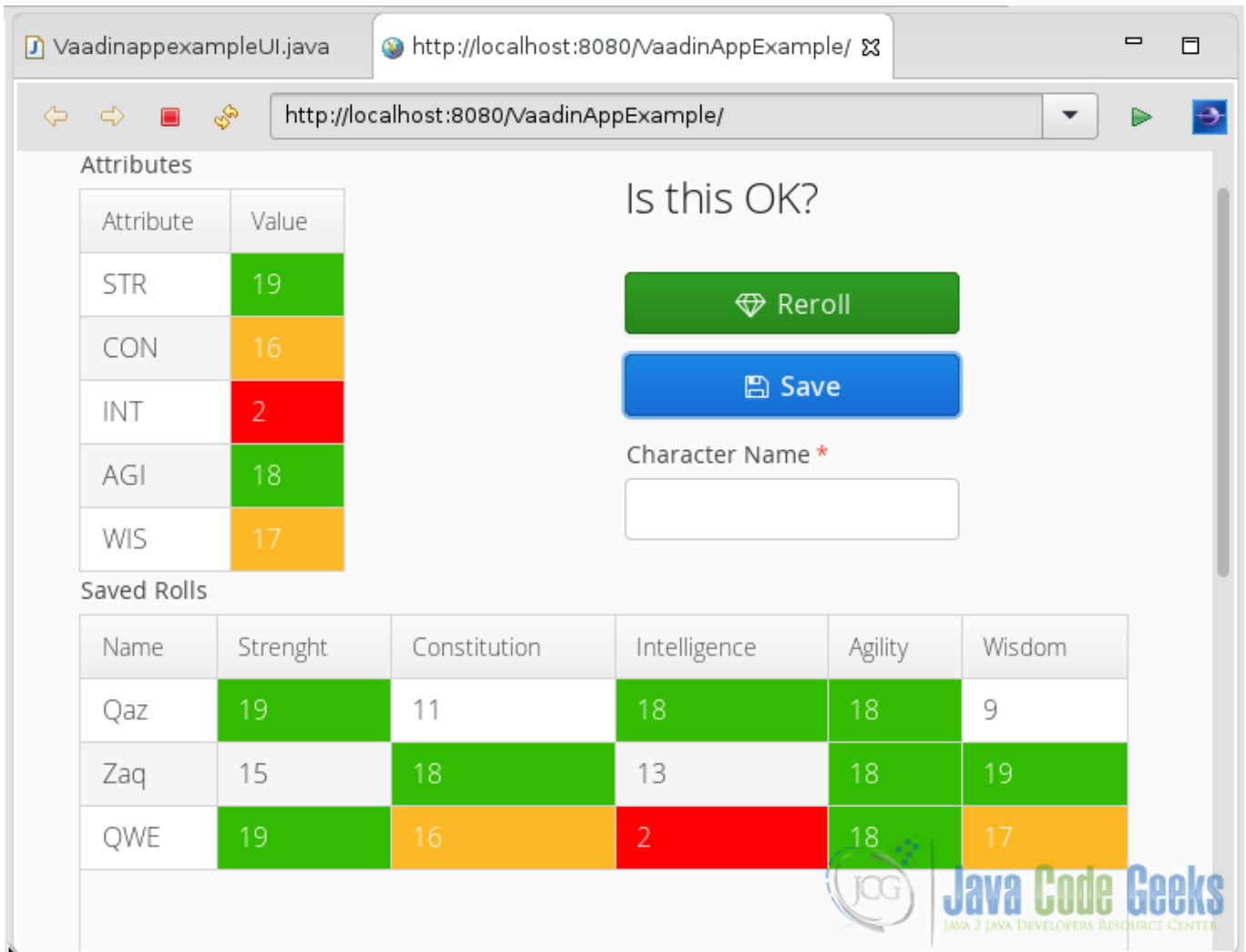
### vaadinappexample.scss

```
@import "../valo/valo.scss";  
  
@mixin vaadinappexample {  
    @include valo;  
  
    .v-table-cell-content-green {  
        background: #33BB00;  
        color: #FFFFFF;  
    }  
  
    .v-table-cell-content-orange {  
        background: #FCB724;  
        color: #FFFFFF;  
    }  
  
    .v-table-cell-content-red {  
        background: #FF0000;  
        color: #FFFFFF;  
    }  
}
```

## 2.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 2.8 Results



The screenshot shows a web browser window with the URL `http://localhost:8080/VaadinAppExample/`. The application displays a character's attributes and a confirmation dialog.

**Attributes**

Attribute	Value
STR	19
CON	16
INT	2
AGI	18
WIS	17

**Is this OK?**

Character Name \*

**Saved Rolls**

Name	Strength	Constitution	Intelligence	Agility	Wisdom
Qaz	19	11	18	18	9
Zaq	15	18	13	18	19
QWE	19	16	2	18	17

Java Code Geeks  
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

Figure 2.3: Vaadin Example Application

## 2.9 Download the Source Code

This was an example of: Vaadin Application.

### Download

You can download the Eclipse project here: [VaadinApplicationExample](#)

## Chapter 3

# Vaadin Best Practices

Best practices are procedures that are accepted or prescribed as being correct or most effective.

### 3.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.8
- Tomcat Server 8

### 3.2 Introduction

In this example we are going to illustrate best practices used to make Vaadin applications. We are going to make a Vaadin example to illustrate these practices.

### 3.3 Prerequisites

- JDK installed
- Eclipse Mars installed and working
- Vaadin plug-in installed
- Tomcat 8 installed and running

### 3.4 Set up the project

In the file menu choose File → New → Other

---

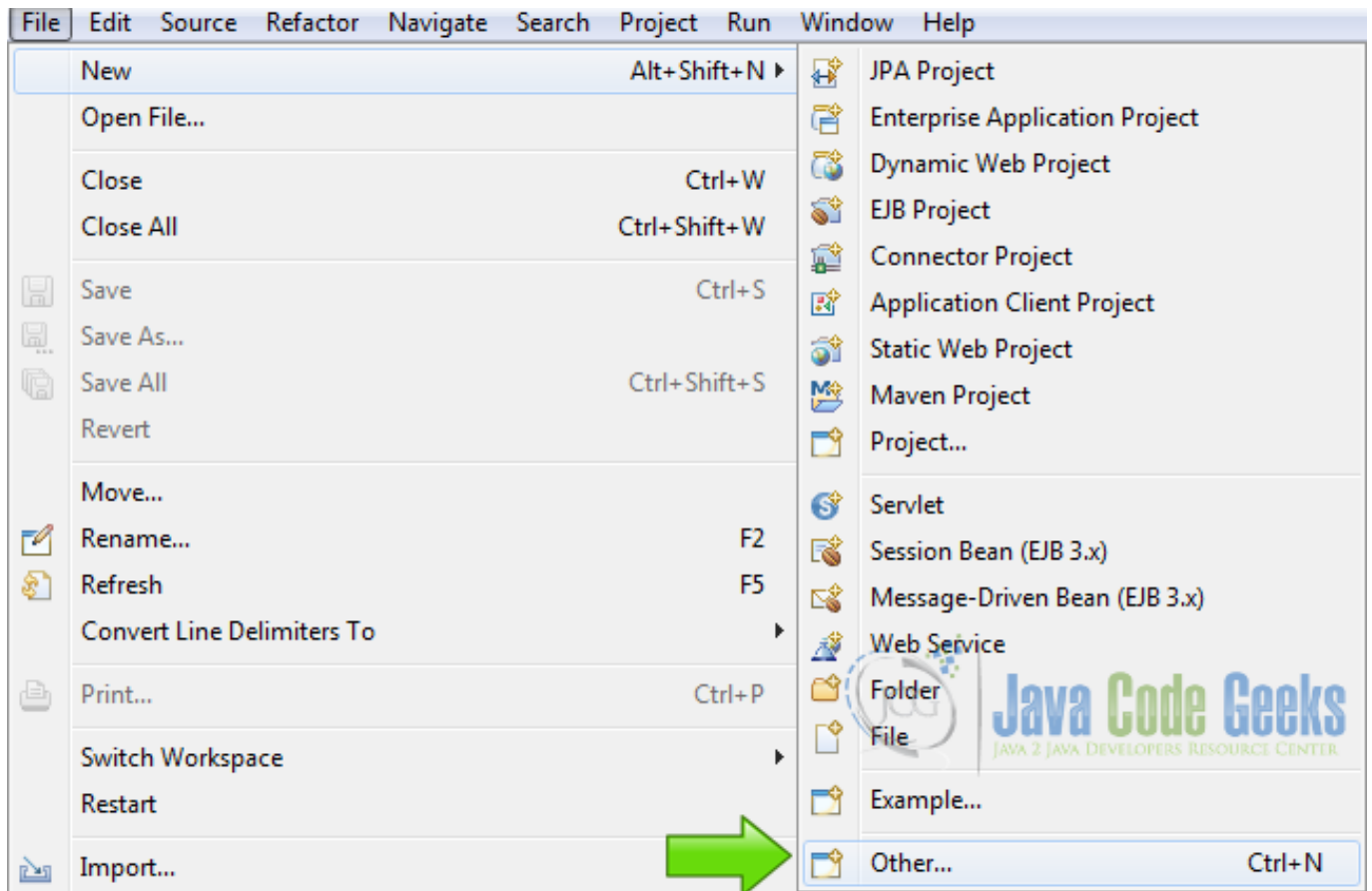


Figure 3.1: New Project

Now from the list choose Vaadin 7 project

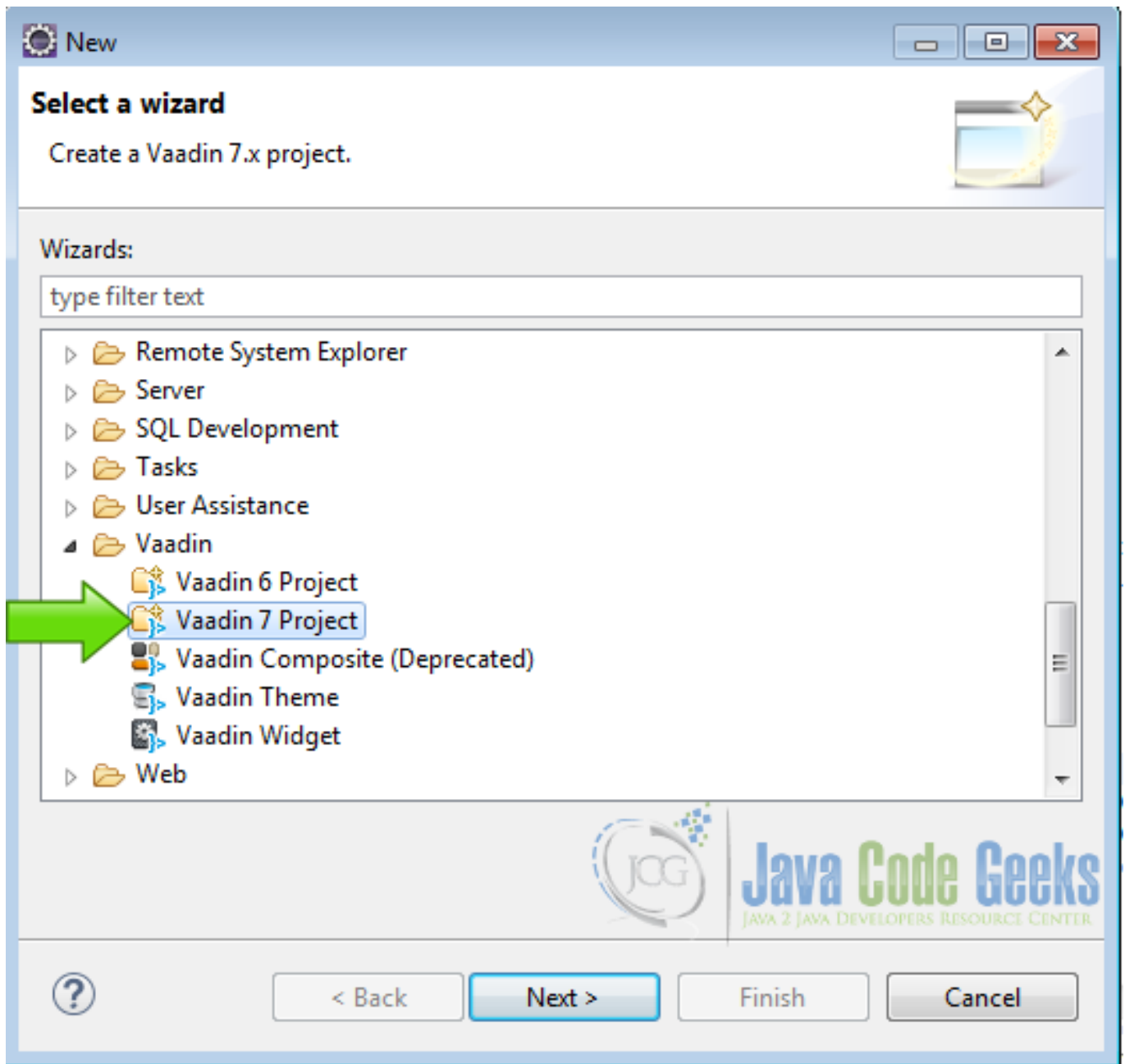


Figure 3.2: Vaadin Project

Click next and name your project then click finish.

## 3.5 The example

### 3.5.1 Make a design

The design is the blueprint of our program. It is better to invest some time making a good design and when the design is ready, start coding the application. In our case we have an application that has a menu and three views, each button on the menu changes the view. We have a welcome view that displays a welcome label. An input view to input some fields, and a view to show all the data.

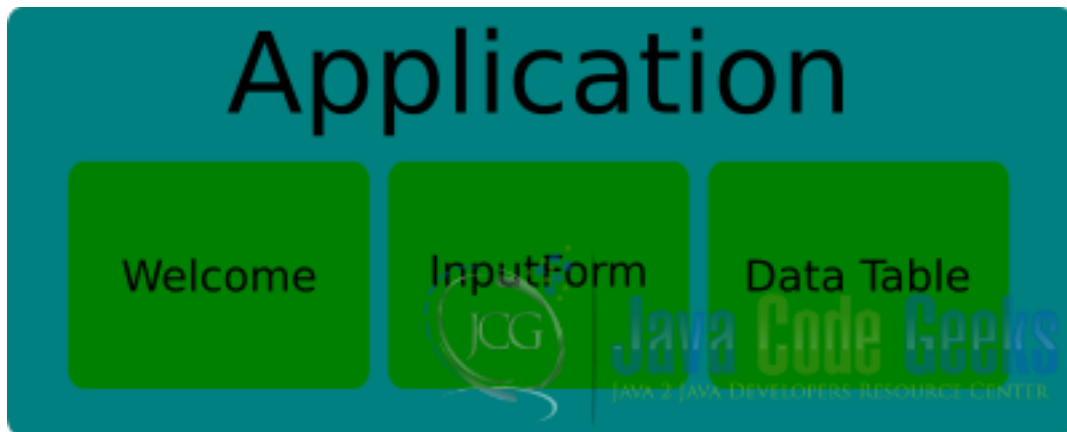


Figure 3.3: Design

### 3.5.2 Annotations

It is recommended to use annotations to define our servlet, because Vaadin uses annotations by default for convenience. Convenience over configuration is a design pattern used to avoid huge configuration files and promotes flexibility.

Annotations

```
@WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinbestpracticesUI. ←
        class, widgetset = "com.example.vaadinbestpractices.widgetset. ←
            VaadinbestpracticesWidgetset")
```

### 3.5.3 Navigator

Use a navigator to change the views in the application. The navigator was created for this task. We use the navigator in our init application method.

#### 3.5.3.1 Layout & content

In our init method, first we create the layout and the content panel to use with the navigator.

Layout & content

```
final VerticalLayout layout = new VerticalLayout();
layout.setMargin(true);
setContent(layout);
Panel contentPanel = new Panel("Main Panel");
contentPanel.setSizeUndefined();
```

`final VerticalLayout layout =new VerticalLayout();` Creates the layout. `layout.setMargin(true);` Sets the margin of the layout.

`setContent(layout);` Sets the layout as the main layout. `Panel contentPanel =new Panel("Main Panel");` Creates a panel yo use with the navigator.

`contentPanel.setSizeUndefined();` Sets the size of the panel.



### 3.5.3.2 Navigator views

We create the navigator and attach the views used in our application. In this case we have 3 views: welcome, input and data.

Navigators views

```
new Navigator(this, contentPanel);
getNavigator().addView(InputPage.NAME, InputPage.class);
getNavigator().addView(WelcomePage.NAME, WelcomePage.class);
getNavigator().addView(DataPage.NAME, DataPage.class);
```

`new Navigator(this, contentPanel);` Creates the navigator using the panel as a placeholder. `getNavigator().addView(InputPage.NAME, InputPage.class);` Adds the input view to the navigator.

`getNavigator().addView(WelcomePage.NAME, WelcomePage.class);` Adds the welcome view to the navigator. `getNavigator().addView(DataPage.NAME, DataPage.class);` Adds the data view to the navigator.

### 3.5.3.3 Menu listeners

We are going to navigate our application using a menu. Each time we click on a menu button the navigator changes the view. For each menu button we have a listener to change the view.

Menubar listeners

```
MenuBar.Command welcome = new Command() {

    @Override
    public void menuSelected(MenuItem selectedItem) {
        getNavigator().navigateTo(WelcomePage.NAME);
    }
};

MenuBar.Command input = new Command() {

    @Override
    public void menuSelected(MenuItem selectedItem) {
        getNavigator().navigateTo(InputPage.NAME);
    }
};

MenuBar.Command data = new Command() {

    @Override
    public void menuSelected(MenuItem selectedItem) {
        getNavigator().navigateTo(DataPage.NAME);
    }
};
```

`MenuBar.Command welcome =new Command()` Creates a new menu command welcome. `getNavigator().navigateTo(WelcomePage.NAME);` Navigates to the welcome page.

`MenuBar.Command input =new Command()` Creates a new menu command input. `getNavigator().navigateTo(InputPage.NAME);` Navigates to the input view.

`MenuBar.Command data =new Command()` Creates a new menu command data. `getNavigator().navigateTo(DataPage.NAME);` Navigates to the data view.

### 3.5.3.4 Menu

We create the menu and attach the buttons to it. When a button is attached to the menu we use the menu command listener created before.

Main menu

```
MenuBar mainMenu = new MenuBar();
mainMenu.addItem("Welcome", FontAwesome.ARROW_CIRCLE_LEFT, welcome);
mainMenu.addItem("Input", FontAwesome.WEIXIN, input);
mainMenu.addItem("Data", FontAwesome.LIST, data);
```

`MenuBar mainMenu =new MenuBar();` Creates a new menu bar. `mainMenu.addItem("Welcome", FontAwesome.ARROW_CIRCLE_LEFT, welcome);` Add the welcome button to the menu.

`mainMenu.addItem("Input", FontAwesome.WEIXIN, input);` Add the input button to the menu. `mainMenu.addItem("Data", FontAwesome.LIST, data);` Add the data button to the menu.

### 3.5.3.5 Navigator initial page

We redirect the navigator to the page we want to show when the application is started.

Navigator initial page

```
layout.addComponent(mainMenu);
layout.addComponent(contentPanel);
getNavigator().navigateTo(WelcomePage.NAME);
```

`layout.addComponent(mainMenu);` Adds the menu to the layout. `layout.addComponent(contentPanel);` Adds the content panel to the layout.

`getNavigator().navigateTo(WelcomePage.NAME);` Navigates to the welcome page when the application is loaded.

### 3.5.3.6 Welcome page

The welcome page is used as the initial page for the navigator.

Welcome page

```
public class WelcomePage extends VerticalLayout implements View {

    private static final long serialVersionUID = 1L;

    public static final String NAME = "welcomepage";

    public WelcomePage() {
        setMargin(true);
        setSpacing(true);
        Label welcome = new Label("Welcome");
        welcome.addStyleName("h1");
        addComponent(welcome);
    }
}
```

`public class WelcomePage extends VerticalLayout implements View` The welcome page used by the navigator must implements the view interface. `public static final String NAME = "welcomepage";` The id of the welcome page to use with the navigator.

`setMargin(true);` Sets the margin of the layout. `setSpacing(true);` Sets the spacing of the layout.

`Label welcome =new Label("Welcome");` Creates a label. `welcome.addStyleName("h1");` Adds a predefined style to the label.

`addComponent(welcome);` Adds the label to the layout.

### 3.5.4 Validate user input

The data entered by a user is prone to errors and mistakes and is sane to have a validation process in the input of the data. We have a view with three input fields to show the validation process. To validate our input fields we use the Vaadin validator.

#### 3.5.4.1 Input form

##### Input form

```
FormLayout inputForm = new FormLayout();
inputForm.setMargin(true);
inputForm.setSpacing(true);
inputPanel.setContent(inputForm);
```

`FormLayout inputForm =new FormLayout();` Creates the input form. `inputForm.setMargin(true);` Sets the margin of the input form.

`inputForm.setSpacing(true);` Sets the spacing of the input form. `inputPanel.setContent(inputForm);` Sets the input form as the content of the input panel.

#### 3.5.4.2 Name field validator

##### Name Field

```
TextField name = new TextField("Name");
name.setNullSettingAllowed(true);
name.setNullRepresentation("");
name.addValidator(new StringLengthValidator("Name must have 3-15 characters lenght", 3, 15, ←
    true));
name.setValidationVisible(true);
inputForm.addComponent(name);
```

`TextField name =new TextField("Name");` Creates a name text field. `name.setNullSettingAllowed(true);` Allows null in the text field.

`name.setNullRepresentation("");` Sets the null representation to a empty string. `name.addValidator(new StringLengthValidator("Name must have 3-15 characters lenght", 3, 15, true));` Adds the validator to the text field. The validator checks that the string entered into the text field has a length greater than 3 and less than 15. `name.setValidationVisible(true);` Sets the validator visible. `inputForm.addComponent(name);` Add the text field to the form.

#### 3.5.4.3 Surname field validator

##### Surname Field

```
TextField surname = new TextField("Surname");
surname.setNullSettingAllowed(true);
surname.setNullRepresentation("");
surname.addValidator(new StringLengthValidator("Surname must have 3-15 characters lenght", ←
    3, 15, true));
surname.setValidationVisible(true);
inputForm.addComponent(surname);
```

`TextField surname =new TextField("Surname");` Creates a text field to the surname. `surname.setNullSettingAllowed(true);` Alows null in the text field.

`surname.setNullRepresentation("");` Sets the null representation to a empty string. `surname.addValidator(new StringLengthValidator("Surname must have 3-15 characters lenght", 3, 15, true));` Adds the validator to the text field. The validator checks that the string entered into the text field has a length greater than 3 and less than 15. `surname.setValidationVisible(true);` Sets the validator visible. `inputForm.addComponent(surname);` Add the text field to the form.

### 3.5.4.4 Age field validator

#### Age field

```
TextField age = new TextField("Age");
age.setNullRepresentation("0");
age.addValidator(new IntegerRangeValidator("Age must be between 1 and 110", 1, 110));
inputForm.addComponent(age);
```

`TextField age =new TextField("Age");` Creates a text field for the age. `age.setNullRepresentation("0");` Sets the null representation to the "0" string.

`age.addValidator(new IntegerRangeValidator("Age must be between 1 and 110", 1, 110));` Adds the validator to the field. The value of the input must be an integer between 1 and 110. `inputForm.addComponent(age);` Adds the text fiel to the form.

### 3.5.4.5 Age field validator

#### Validation buttons

```
HorizontalLayout btLayout = new HorizontalLayout();
Button btSave = new Button("Save");
btLayout.addComponent(btSave);
Button btClear = new Button("Clear");
btLayout.addComponent(btClear);
inputForm.addComponent(btLayout);
```

`HorizontalLayout btLayout =new HorizontalLayout();` Creates a horizontal layout for the buttons. `Button btSave =new Button("Save");` Creates a button for save the form data.

`btLayout.addComponent(btSave);` Adds the button to the layout. `Button btClear =new Button("Clear");` Creates a new button to clear the fields.

`btLayout.addComponent(btClear);` Adds the clear button to the layout. `inputForm.addComponent(btLayout);` Adds the button layout to the form.

### 3.5.4.6 Validation process

#### Checks if the fields are empty

```
btSave.addClickListener(new ClickListener() {

    @Override
    public void buttonClick(ClickEvent event) {
        if(!name.isEmpty() && !surname.isEmpty() && !age.isEmpty()){

        }else{
            Notification.show("All fields must be filled");
        }
    }
});
```

Checks that all fields have a value otherwise it shows a notification.

#### Try to validate

```
Boolean save = true;
try{
    name.validate();
} catch(InvalidValueException e){
```

```

        save = false;
    }

    try{
        surname.validate();
    }catch(InvalidValueException e){
        save = false;
    }

    try{
        age.validate();
    }catch(InvalidValueException e){
        save = false;
    }

```

Tries to validate the fields. A boolean is used to keep the status of the validation process. If any validation fails we set save to false.

#### Save click listener

```

if (save) {
    VaadinbestpracticesUI.dataBean.addBean(
        new DataBean(name.getValue(), surname.getValue(), Integer.valueOf(↵
            age.getValue())));
    Notification.show("Data saved!");
    name.setValue("");
    surname.setValue("");
    age.setValue("0");
    btSave.setComponentError(null);
}

```

`if (save)` We check the boolean to save. `VaadinbestpracticesUI.dataBean.addBean(new DataBean(name.getValue(), surname.getValue(), Integer.valueOf(age.getValue())));` We create a bean with the new data. `Notification.show("Data saved!");` Notifies that the data is saved. `name.setValue("");` Clears the name field.

`surname.setValue("");` Clears the surname field. `age.setValue("0");` Clears the age field with the null value.

#### 3.5.4.7 Clear fields

```

btClear.addClickListener(new ClickListener() {

    @Override
    public void buttonClick(ClickEvent event) {
        name.clear();
        surname.clear();
        age.clear();
    }
});

```

`name.clear();` Clears the name. `surname.clear();` Clears the surname. `age.clear();` Clears the age.

### 3.5.5 Use containers in fields

The container allow us to bind our input fields to a data type and help in the validation process.

### 3.5.5.1 Property sets

```
PropertysetItem fProperties = new PropertysetItem();
fProperties.addItemProperty("nameValidator", new ObjectProperty(""));
fProperties.addItemProperty("surnameValidator", new ObjectProperty(""));
fProperties.addItemProperty("integerValidator", new ObjectProperty(0));
```

`PropertysetItem fProperties =new PropertysetItem();` Creates a property set. `fProperties.addItemProperty("nameValidator", new ObjectProperty(""));` Adds the name property.

`fProperties.addItemProperty("surnameValidator", new ObjectProperty(""));` Adds the surname property. `fProperties.addItemProperty("integerValidator", new ObjectProperty(0))` Adds the age property.

### 3.5.5.2 Field groups

```
FieldGroup fGroup = new FieldGroup(fProperties);
fGroup.bind(name, "nameValidator");
fGroup.bind(surname, "surnameValidator");
fGroup.bind(age, "integerValidator");
```

`FieldGroup fGroup =new FieldGroup(fProperties);` Creates a field group. `fGroup.bind(name, "nameValidator");` Binds the name text field to the name property.

`fGroup.bind(surname, "surnameValidator");` Binds the surname text field to the surname property. `fGroup.bind(age, "integerValidator");` Binds the age text field to the age property.

## 3.5.6 Separate the UI from the data

Separation of the data from the UI allow us to change the UI or the data store without affecting each other.

In the image the UI, the datasets and the database are in different layers. If you change any of these three pieces you only have to define the same interfaces to communicate each other. The change of one layer doesn't have to affect any other layer. If you want to change the database from MySQL to PostgreSQL for example, this change is transparent to the UI code.

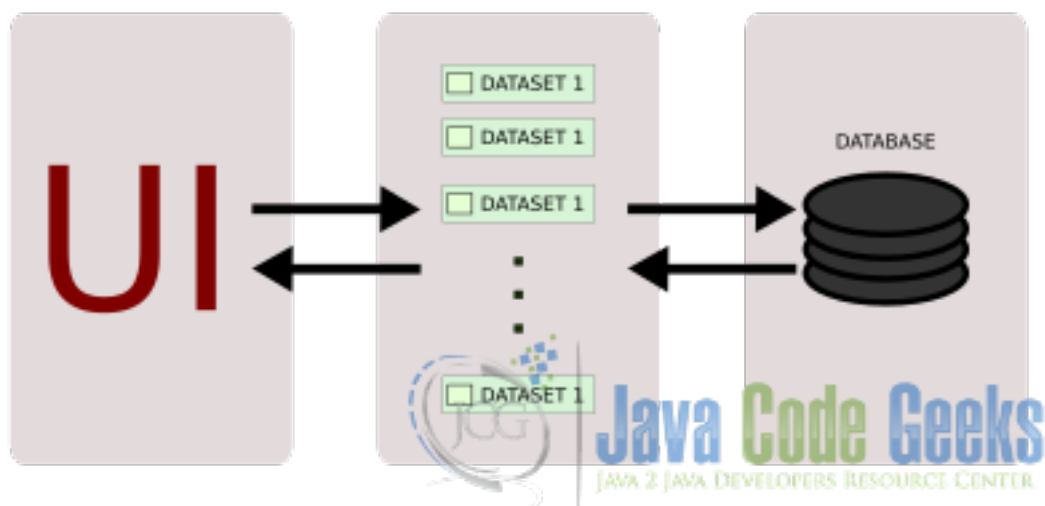


Figure 3.4: Design pattern

```
public class DataBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;
    private String surname;
    private Integer age;

    public DataBean(String pName, String pSurname, Integer pAge) {
        this.name = pName;
        this.surname = pSurname;
        this.age = pAge;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

}
```

This is a standard java class that extends serializable. This class has three fields to store the name, the surname and the age with its getters and setters.

#### Data View

```
public static final String NAME = "datapage";

public DataPage() {
    Table dataTable = new Table("Data Table", VaadinbestpracticesUI.dataBean);
    dataTable.setVisibleColumns(new Object[]{"name", "surname", "age"});
    dataTable.setHeight("200px");
    addComponent(dataTable);
}
```

`public static final String NAME = "datapage";` Creates the id of the data view. `Table dataTable = new Table("Data Table", VaadinbestpracticesUI.dataBean);` Creates a table to show all the records we have loaded. The table is using the container as a data source. `dataTable.setVisibleColumns(new Object[]{"name", "surname", "age"});` Sets the visible columns. `dataTable.setHeight("200px");` Sets the height of the table. `addComponent(dataTable);` Adds the table to the layout.

### 3.5.7 Deploy on https

If our application is going to be on a public domain is better to deploy it using http secure protocol. Https encrypts our connection protecting us for some kind of attacks that could compromise our data.

## 3.6 The complete source code

### 3.6.1 VaadinbestpracticesUI.java

VaadinbestpracticesUI.java

```
package com.example.vaadinbestpractices;

import javax.servlet.annotation.WebServlet;

import com.example.vaadinbestpractices.data.DataBean;
import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.data.util.BeanContainer;
import com.vaadin.navigator.Navigator;
import com.vaadin.server.FontAwesome;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.MenuBar;
import com.vaadin.ui.MenuBar.Command;
import com.vaadin.ui.MenuBar.MenuItem;
import com.vaadin.ui.Panel;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadinbestpractices")
public class VaadinbestpracticesUI extends UI {

    public static BeanContainer dataBean;

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinbestpracticesUI.class, widgetset = "com.example.vaadinbestpractices.widgetset.VaadinbestpracticesWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);
        Panel contentPanel = new Panel("Main Panel");
        contentPanel.setSizeUndefined();
        dataBean = new BeanContainer(DataBean.class);
        dataBean.setBeanIdProperty("name");

        new Navigator(this, contentPanel);
        getNavigator().addView(InputPage.NAME, InputPage.class);
        getNavigator().addView>WelcomePage.NAME, WelcomePage.class);
        getNavigator().addView>DataPage.NAME, DataPage.class);

        MenuBar.Command welcome = new Command() {
```



```
        @Override
        public void menuSelected(MenuItem selectedItem) {
            getNavigator().navigateTo(WelcomePage.NAME);
        }
    };

    MenuBar.Command input = new Command() {

        @Override
        public void menuSelected(MenuItem selectedItem) {
            getNavigator().navigateTo(InputPage.NAME);
        }
    };

    MenuBar.Command data = new Command() {

        @Override
        public void menuSelected(MenuItem selectedItem) {
            getNavigator().navigateTo(DataPage.NAME);
        }
    };

    MenuBar mainMenu = new MenuBar();
    mainMenu.addItem("Welcome", FontAwesome.ARROW_CIRCLE_LEFT, welcome);
    mainMenu.addItem("Input", FontAwesome.WEIXIN, input);
    mainMenu.addItem("Data", FontAwesome.LIST, data);

    layout.addComponent(mainMenu);
    layout.addComponent(contentPanel);
    getNavigator().navigateTo(WelcomePage.NAME);
}
}
```

### 3.6.2 WelcomePage.java

WelcomePage.java

```
package com.example.vaadinbestpractices;

import com.vaadin.navigator.View;
import com.vaadin.navigator.ViewChangeListener.ViewChangeEvent;
import com.vaadin.ui.Label;
import com.vaadin.ui.VerticalLayout;

public class WelcomePage extends VerticalLayout implements View {

    private static final long serialVersionUID = 1L;

    public static final String NAME = "welcomepage";

    public WelcomePage() {
        setMargin(true);
        setSpacing(true);
        Label welcome = new Label("Welcome");
        welcome.addStyleName("h1");
        addComponent(welcome);
    }
}
```

```
    @Override
    public void enter(ViewChangeEvent event) {

    }
}
```

### 3.6.3 InputPage.java

#### InputPage.java

```
package com.example.vaadinbestpractices;

import com.example.vaadinbestpractices.data.DataBean;
import com.google.appengine.api.memcache.InvalidValueException;
import com.vaadin.data.fieldgroup.FieldGroup;
import com.vaadin.data.util.ObjectProperty;
import com.vaadin.data.util.PropertysetItem;
import com.vaadin.data.validator.IntegerRangeValidator;
import com.vaadin.data.validator.StringLengthValidator;
import com.vaadin.navigator.View;
import com.vaadin.navigator.ViewChangeListener.ViewChangeEvent;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Notification;
import com.vaadin.ui.Panel;
import com.vaadin.ui.TextField;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
public class InputPage extends VerticalLayout implements View {

    public static final String NAME = "inputpage";

    public InputPage() {
        Panel inputPanel = new Panel("Input data");
        inputPanel.setSizeUndefined();
        addComponent(inputPanel);

        PropertysetItem fProperties = new PropertysetItem();
        fProperties.addItemProperty("nameValidator", new ObjectProperty(""));
        fProperties.addItemProperty("surnameValidator", new ObjectProperty(""));
        fProperties.addItemProperty("integerValidator", new ObjectProperty(0));

        FormLayout inputForm = new FormLayout();

        TextField name = new TextField("Name");
        name.setNullSettingAllowed(true);
        name.setNullRepresentation("");
        name.addValidator(new StringLengthValidator("Name must have 3-15 characters ←
            lenght", 3, 15, true));
        name.setValidationVisible(true);
        inputForm.addComponent(name);

        TextField surname = new TextField("Surname");
        surname.setNullSettingAllowed(true);
        surname.setNullRepresentation("");
    }
}
```

```
surname.addValidator(new StringLengthValidator("Surname must have 3-15 ↵
    characters lenght", 3, 15, true));
surname.setValidationVisible(true);
inputForm.addComponent(surname);

TextField age = new TextField("Age");
age.setNullRepresentation("0");
age.addValidator(new IntegerRangeValidator("Age must be between 1 and 110", ↵
    1, 110));
inputForm.addComponent(age);

HorizontalLayout btLayout = new HorizontalLayout();
Button btSave = new Button("Save");
btLayout.addComponent(btSave);
Button btClear = new Button("Clear");
btLayout.addComponent(btClear);
inputForm.addComponent(btLayout);

btSave.addClickListener(new ClickListener() {

    @Override
    public void buttonClick(ClickEvent event) {
        if(!name.isEmpty() && !surname.isEmpty() && !age.isEmpty()) ↵
        {
            Boolean save = true;
            try{
                name.validate();

            }catch(InvalidValueException e){
                save = false;
            }

            try{
                surname.validate();

            }catch(InvalidValueException e){
                save = false;
            }

            try{
                age.validate();

            }catch(InvalidValueException e){
                save = false;
            }

            if(save){
                VaadinbestpracticesUI.dataBean.addBean(
                    new DataBean(name.getValue() ↵
                        (), surname.getValue(), ↵
                            Integer.valueOf(age. ↵
                                getValue())));
                Notification.show("Data saved!");
                name.setValue("");
                surname.setValue("");
                age.setValue("0");
                btSave.setComponentError(null);
            }
        }else{
            Notification.show("All fields must be filled");
        }
    }
});
```

```
    });

    btClear.addClickListener(new ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {
            name.clear();
            surname.clear();
            age.clear();
        }
    });

    FieldGroup fGroup = new FieldGroup(fProperties);
    fGroup.bind(name, "nameValidator");
    fGroup.bind(surname, "surnameValidator");
    fGroup.bind(age, "integerValidator");

    inputForm.setMargin(true);
    inputForm.setSpacing(true);
    inputPanel.setContent(inputForm);
}

@Override
public void enter(ViewChangeEvent event) {

}

}
```

### 3.6.4 DataPage.java

DataPage.java

```
package com.example.vaadinbestpractices;

import com.vaadin.navigator.View;
import com.vaadin.navigator.ViewChangeListener.ViewChangeEvent;
import com.vaadin.ui.Table;
import com.vaadin.ui.VerticalLayout;

public class DataPage extends VerticalLayout implements View {

    private static final long serialVersionUID = 1L;

    public static final String NAME = "datapage";

    public DataPage() {
        Table dataTable = new Table("Data Table", VaadinbestpracticesUI.dataBean);
        dataTable.setVisibleColumns(new Object[]{"name", "surname", "age"});
        dataTable.setHeight("200px");
        addComponent(dataTable);
    }

    @Override
    public void enter(ViewChangeEvent event) {

    }

}
```

### 3.6.5 DataBean.java

DataBean.java

```
package com.example.vaadinbestpractices.data;

import java.io.Serializable;

public class DataBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;
    private String surname;
    private Integer age;

    public DataBean(String pName, String pSurname, Integer pAge) {
        this.name = pName;
        this.surname = pSurname;
        this.age = pAge;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
}
```

## 3.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and click finish.

## 3.8 Results

### 3.8.1 Welcome view

This is the start page. Every time you open the application page, this page is shown.

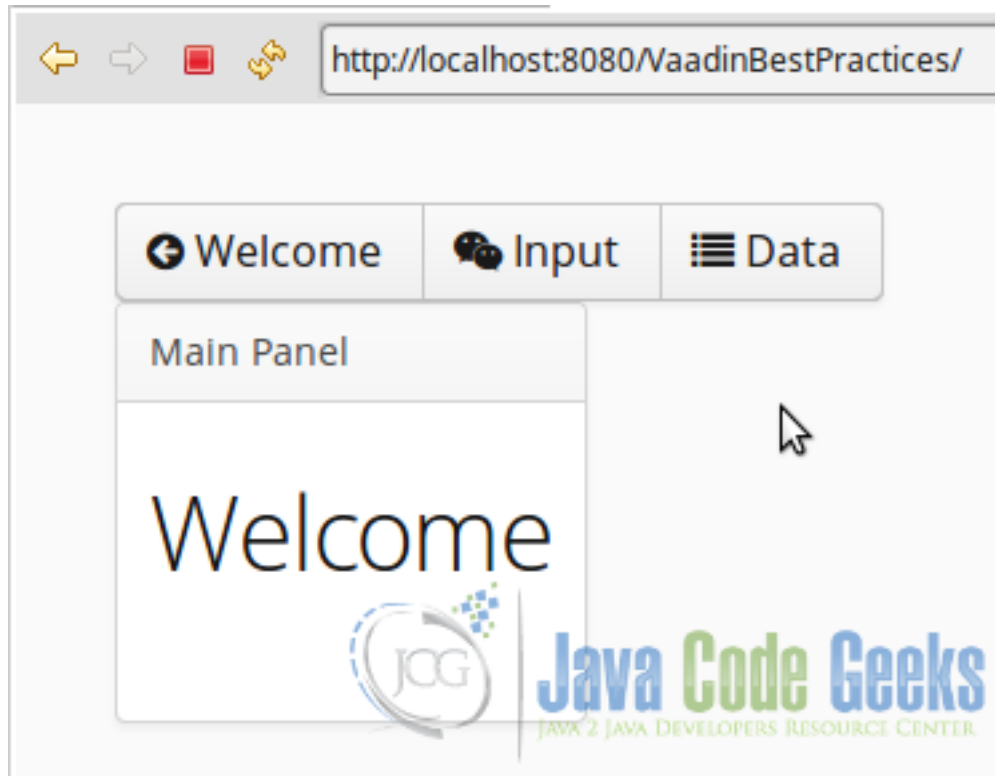


Figure 3.5: Welcome View

### 3.8.2 Input view

In this page we can add records to our example. Here we validate the fields and then store the data into a container.



Figure 3.6: Input View

### 3.8.3 Input view

In this view, we retrieve all the records from the container and shown them into a table.

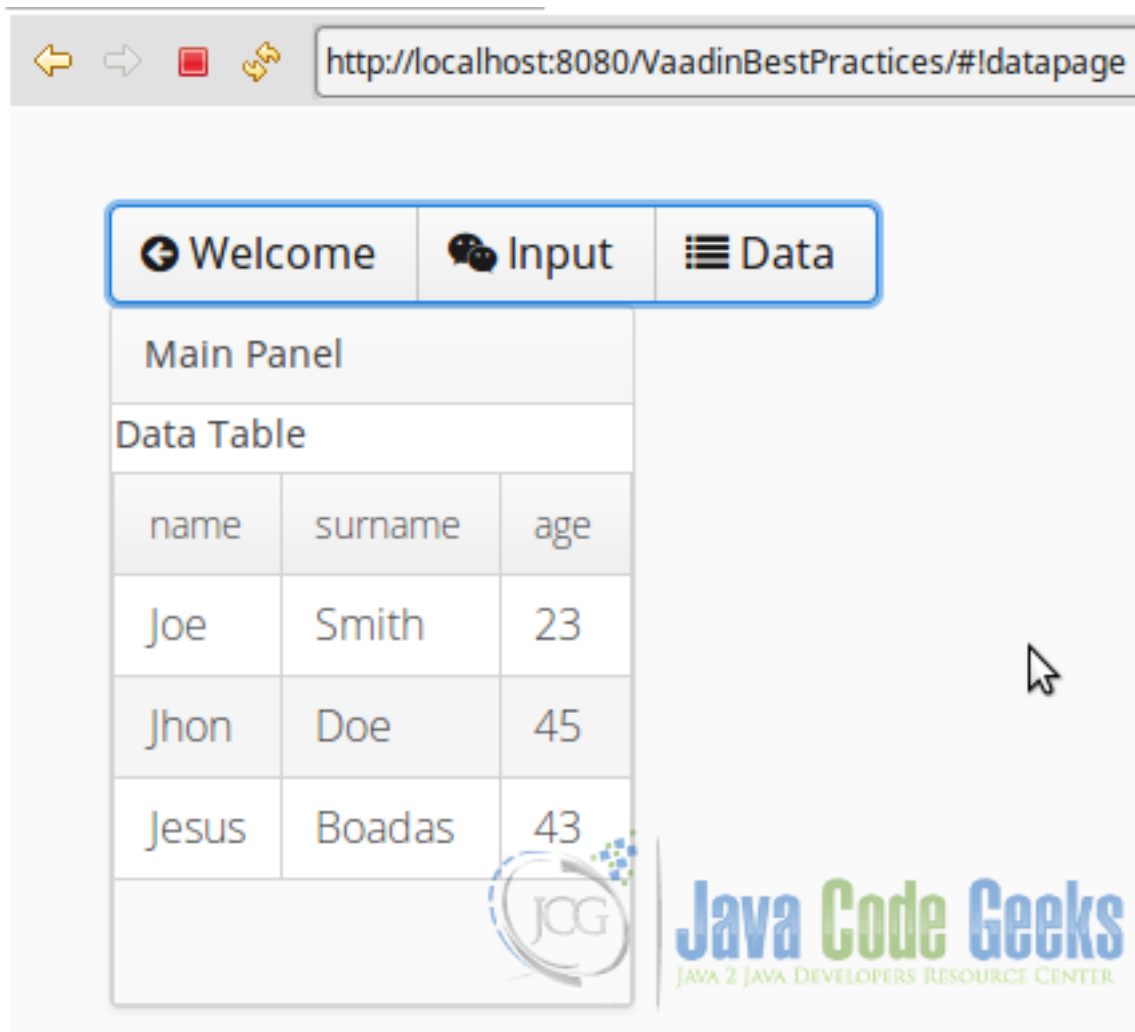


Figure 3.7: Data View

### 3.9 Download the Source Code

This was an example of: Vaadin Best Practices.

#### Download

You can download the Eclipse project here: [VaadinBestPractices](#)



## Chapter 4

# Vaadin Visual Designer Example

Modern rapid application development environments usually have a visual tool to make the UI. The visual tool allows to put the widgets on the application without use code.

### 4.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.6
- Tomcat Server 8

### 4.2 Introduction

In this example we are going to bind widgets in Vaadin using some common techniques. We are going to use some text boxes to bind labels using the same data. This can be used to bind any widget with either, widgets or a back end with data as a data source.

### 4.3 Prerequisites

- JDK installed
- Eclipse Mars installed and working
- Vaadin plug-in installed
- Tomcat 8 installed and running

### 4.4 Set up the project

#### 4.4.1 Get the Visual Designer

We are going to use the trial version of Vaadin Visual Designer. Go to the page [Vaadin Designer](#). Click on Start your free trial.

---

A promotional banner for Vaadin Designer. At the top center is the Vaadin logo, a white stylized 'D' with a curly brace inside, set within a white grid. Below the logo, the text 'Vaadin Designer' is written in a large, bold, white sans-serif font. Underneath that, the tagline 'From Idea to App in an Instant' is written in a smaller, bold, black sans-serif font. A paragraph of white text follows: 'We wanted to challenge the old myth that "Java developers can't create beautiful UIs". This in mind, we introduced Vaadin Designer. Now any Java developer can create great looking web UIs with great productivity.' At the bottom, there are two buttons: a dark grey button with white text 'Watch introduction' and a green button with white text 'Buy now'. Below these buttons is a green arrow pointing right, followed by the text 'or Start your free trial'. In the bottom right corner, there is a logo for 'Java Code Geeks' with the tagline 'Java & Java Developer's Resource Centre'.

Figure 4.1: 1 Get visual designer

Log in into your Vaadin account and now you can see the follow screen.

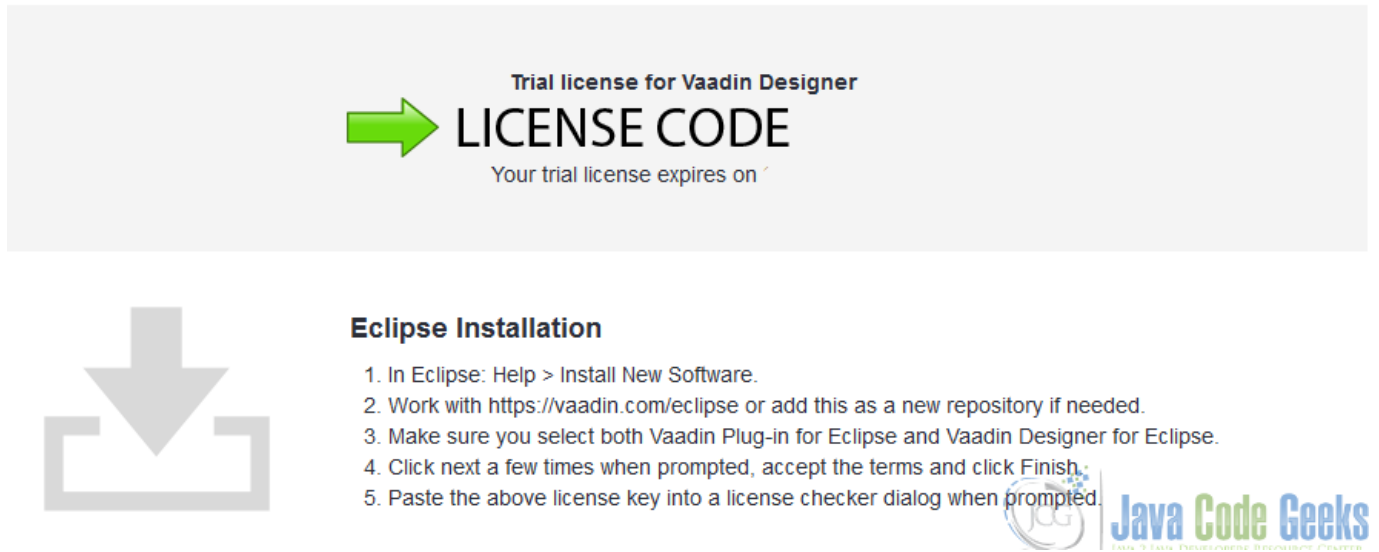


Figure 4.2: License Code

In this pop-up you can see your Vaadin license code and the installation instructions to get the Visual designer working. Start eclipse and follow the instructions to install the visual designer.

#### 4.4.2 Create the project

In eclipse file menu click on new → other.

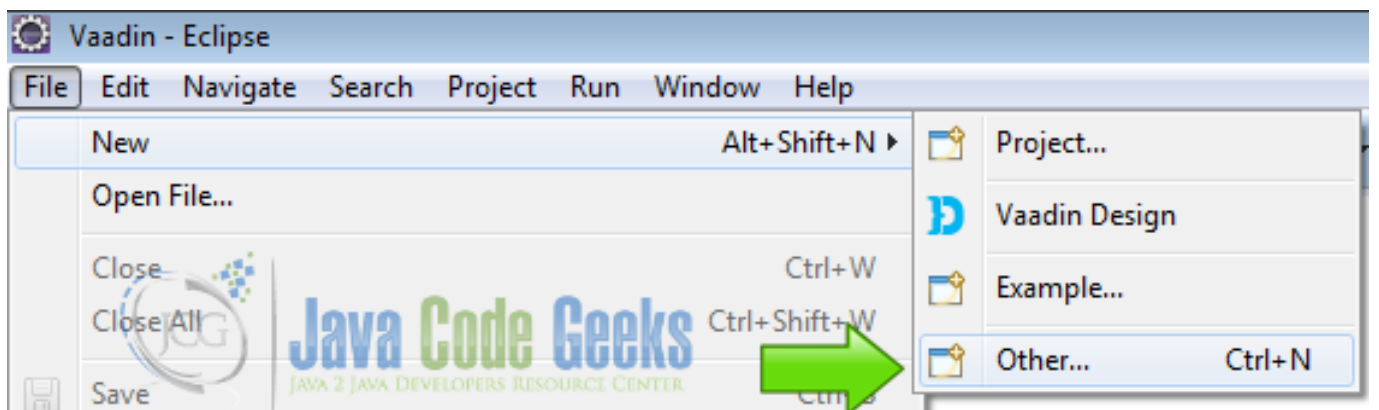


Figure 4.3: New Project

Choose Vaadin 7 project from the drop down list.

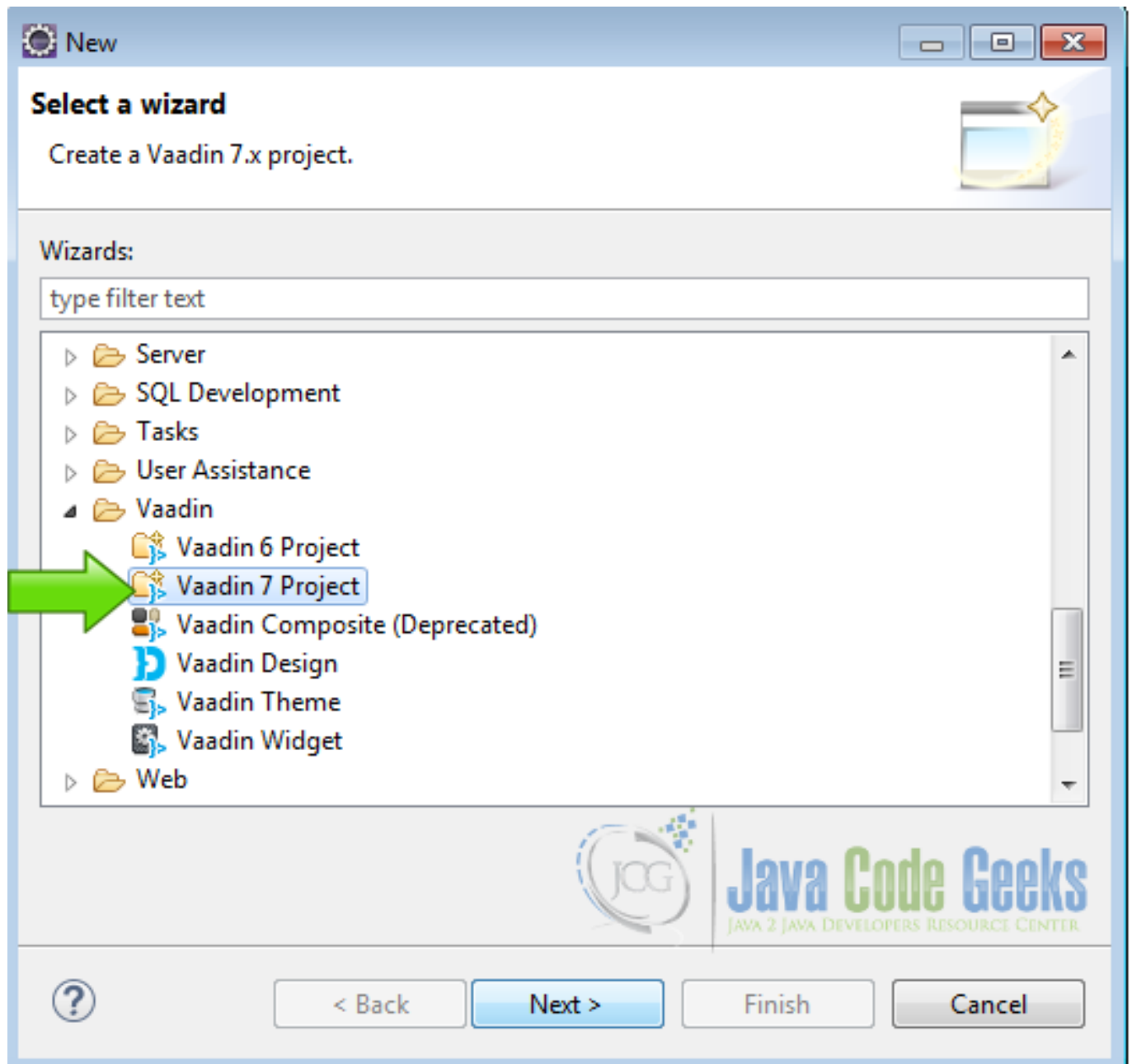


Figure 4.4: Vaadin 7 Project

Name the project.

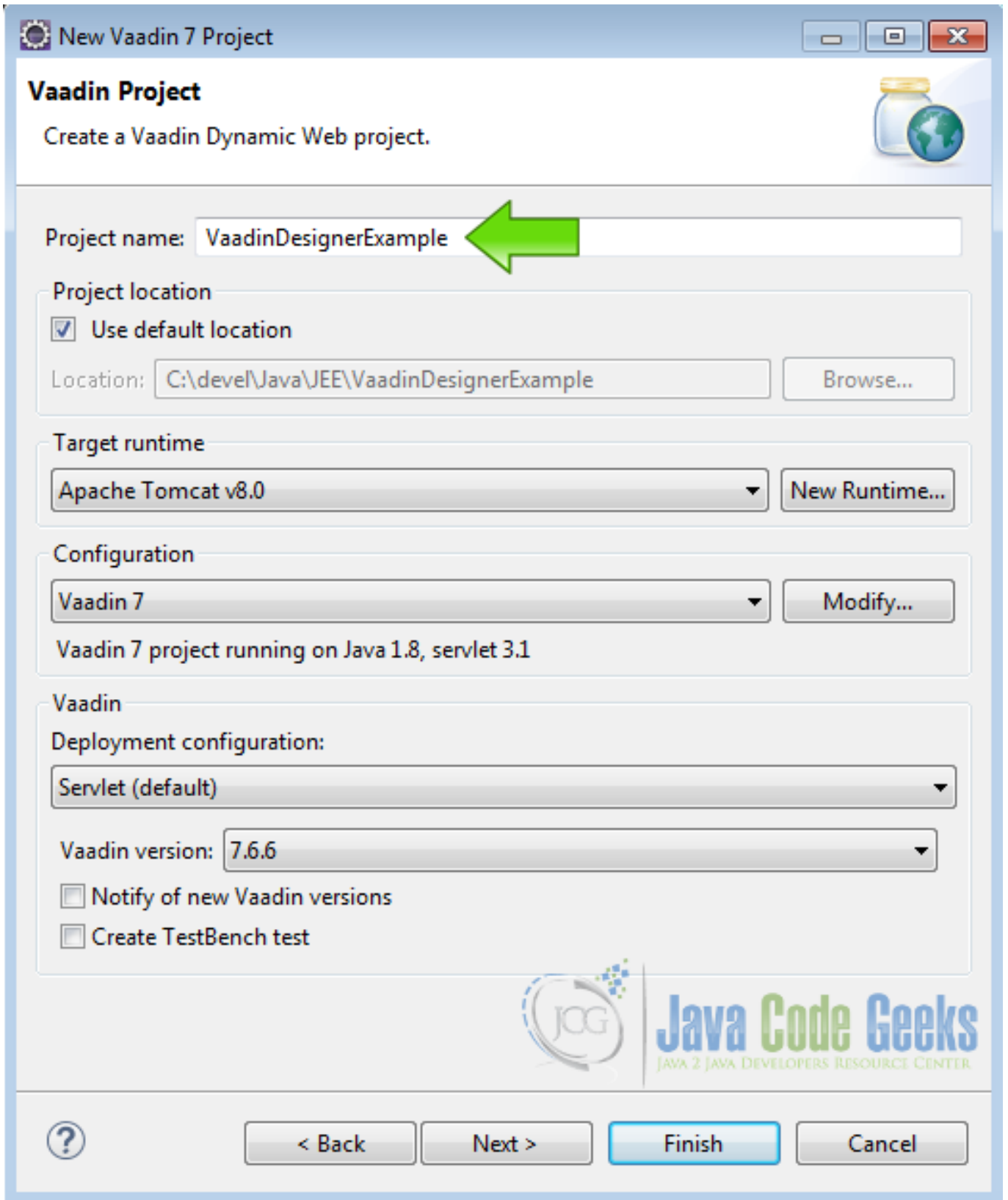


Figure 4.5: Name the project

Click next until the last screen and deselect the option "Create project template" and press finish.

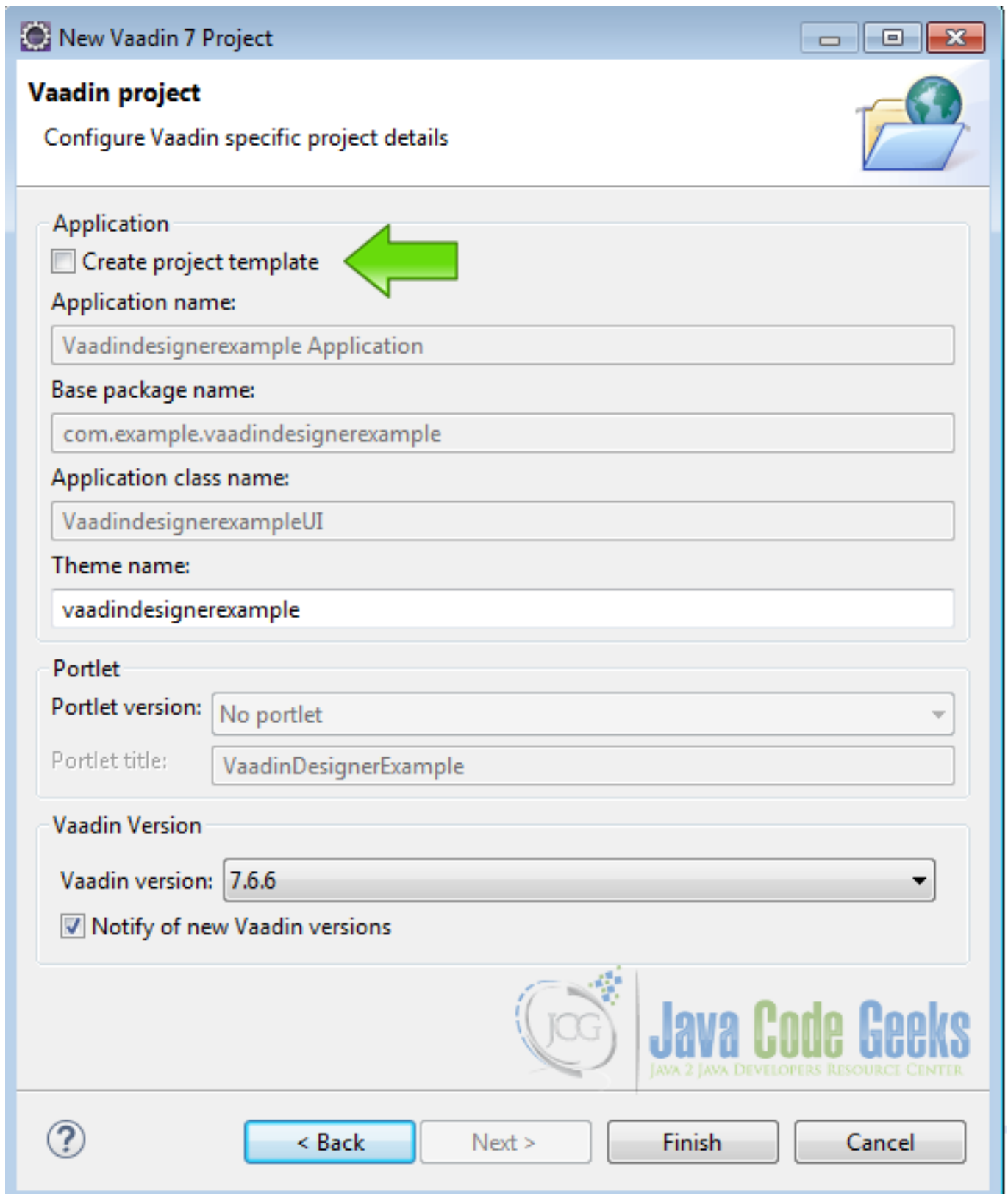


Figure 4.6: Project Template

Right click on the project folder and choose New → Other.

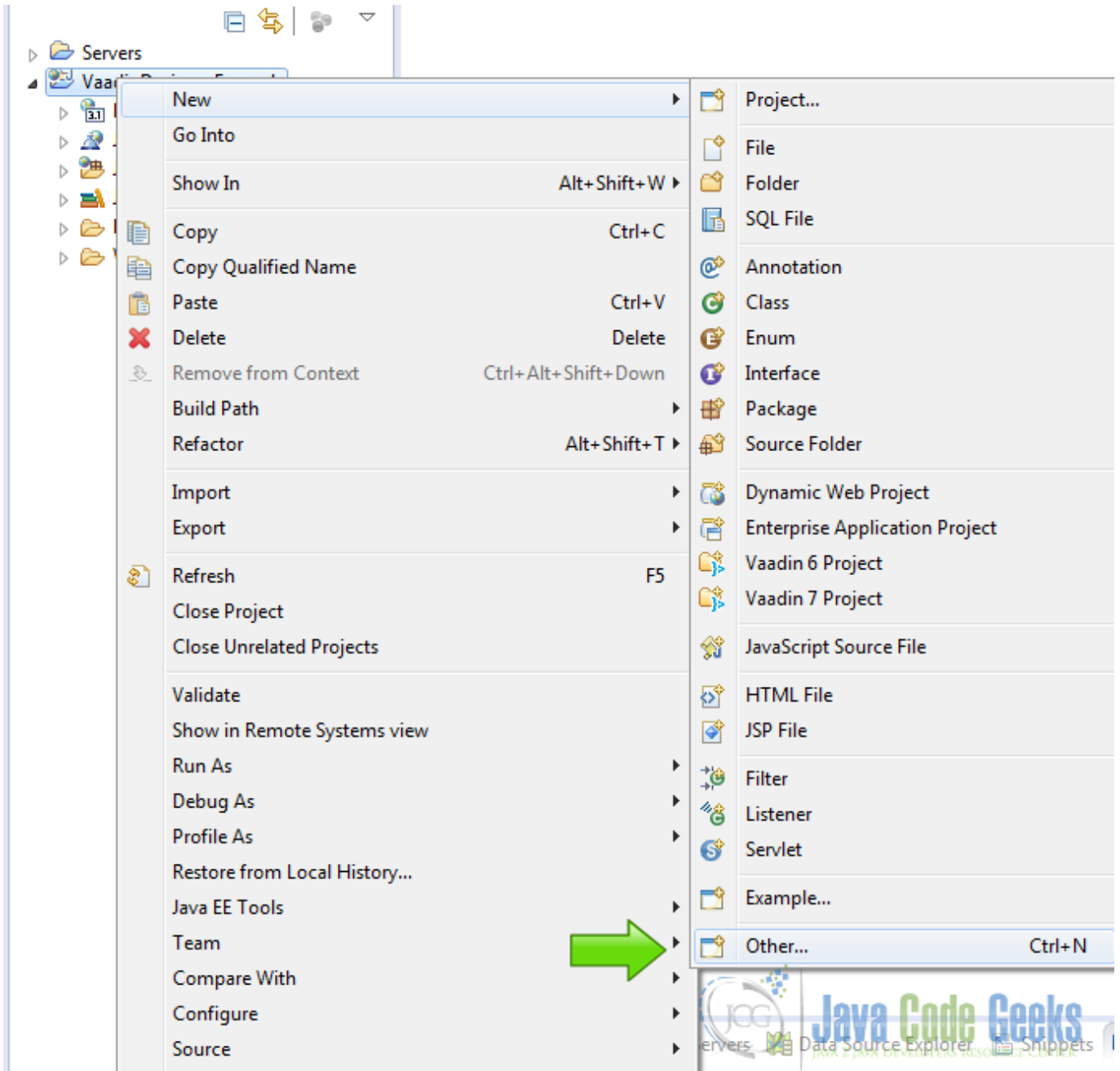


Figure 4.7: New Other

From the list choose Vaadin Design and hit next.

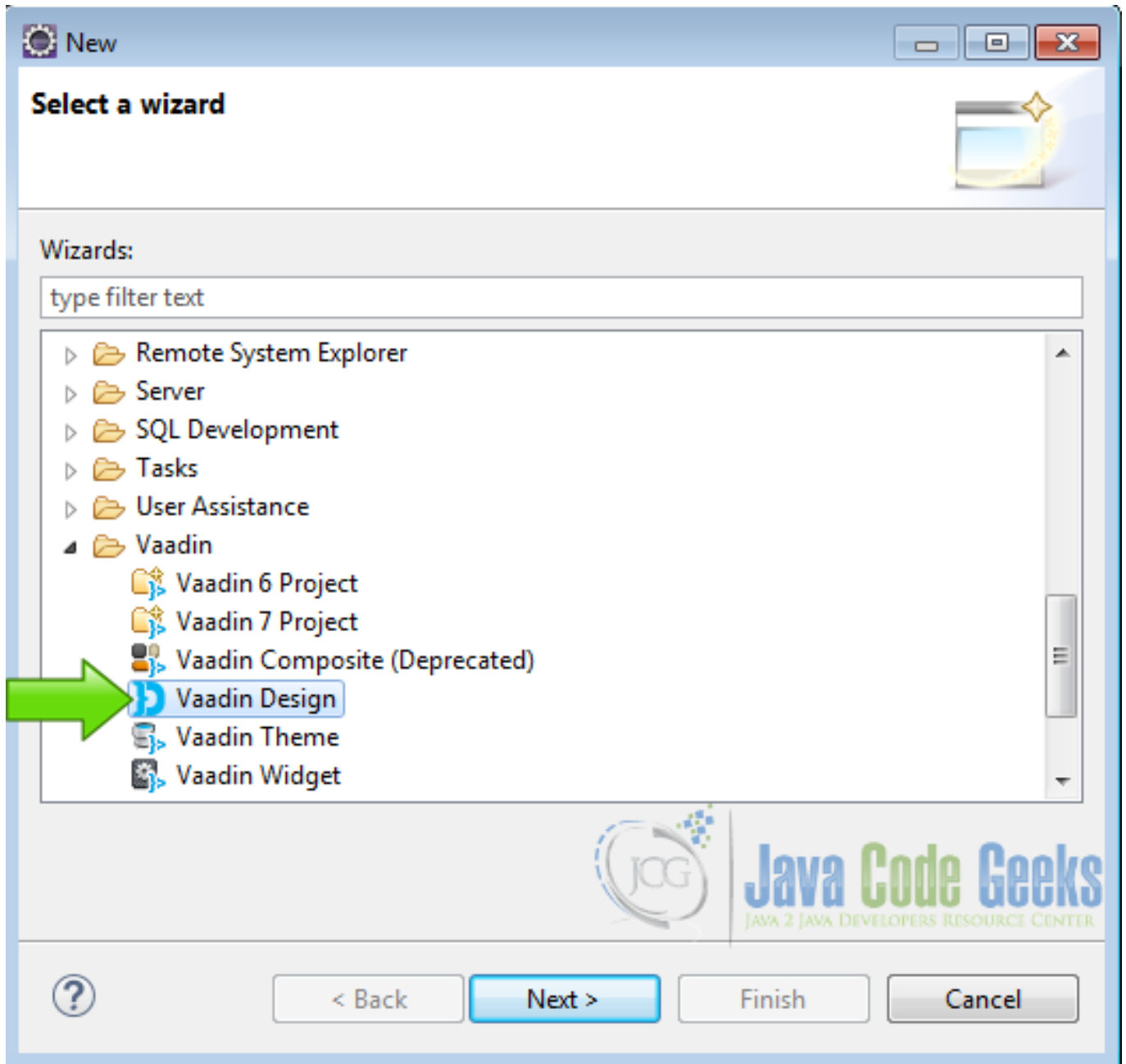


Figure 4.8: Vaadin Design Wizard

Fill the name and the package of the design. Also choose the main layout and hit finish.



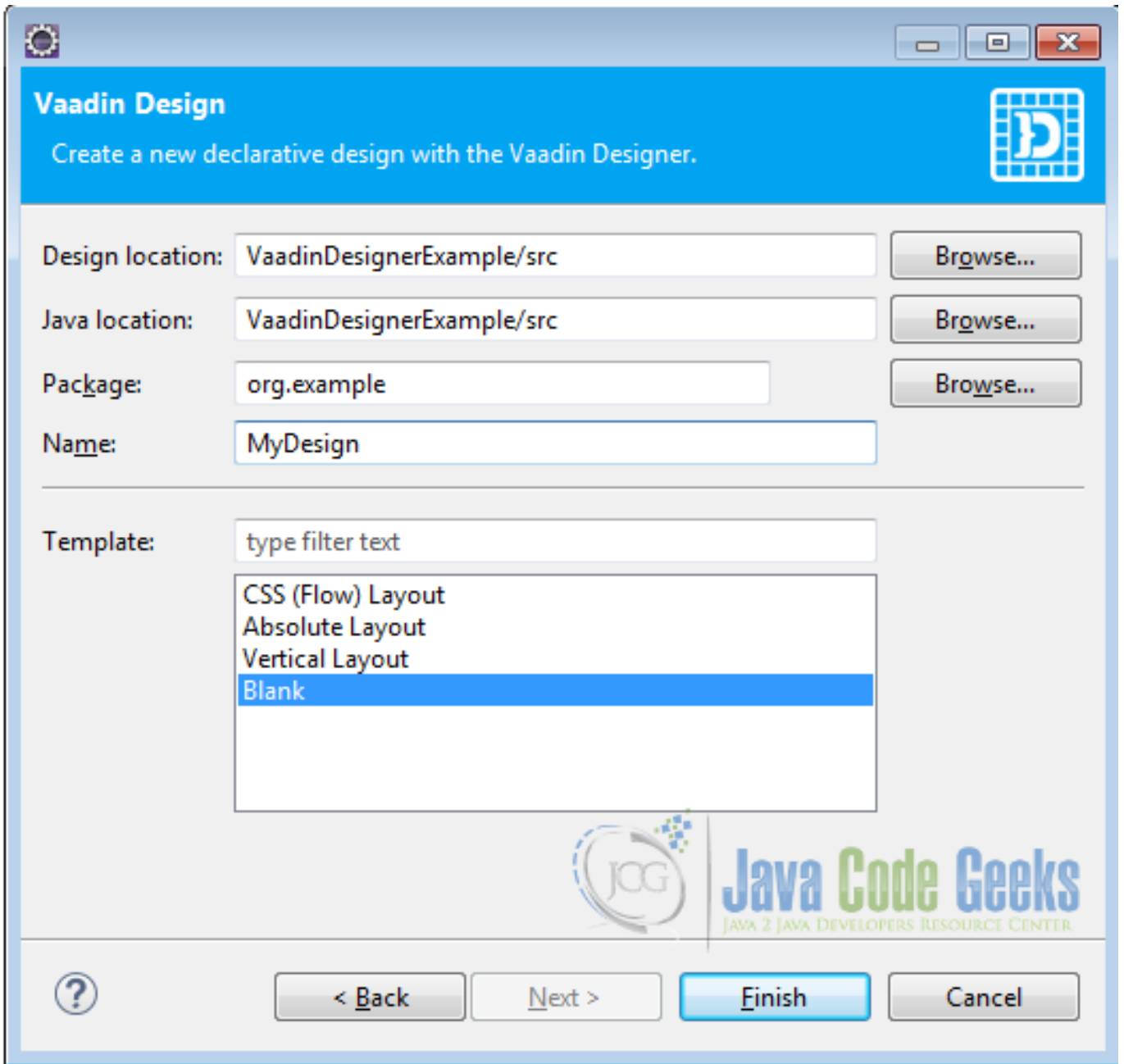


Figure 4.9: Vaadin Design details

Now Vaadin is gonna ask for the license code of the designer, write the code and click apply.

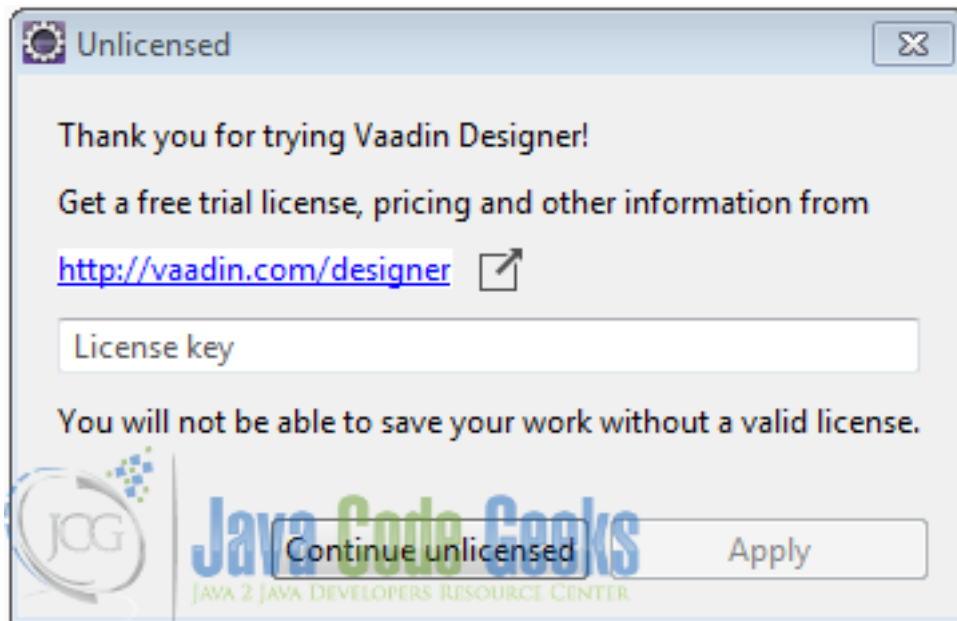


Figure 4.10: designer license

Now eclipse changes the perspective to show the Vaadin visual designer. On the center of the screen you can see your design and on the right side you can see 3 panels. The first panel has the widget palette, the second panel has the outline of the design and the third panel has the properties of the current selected widget.

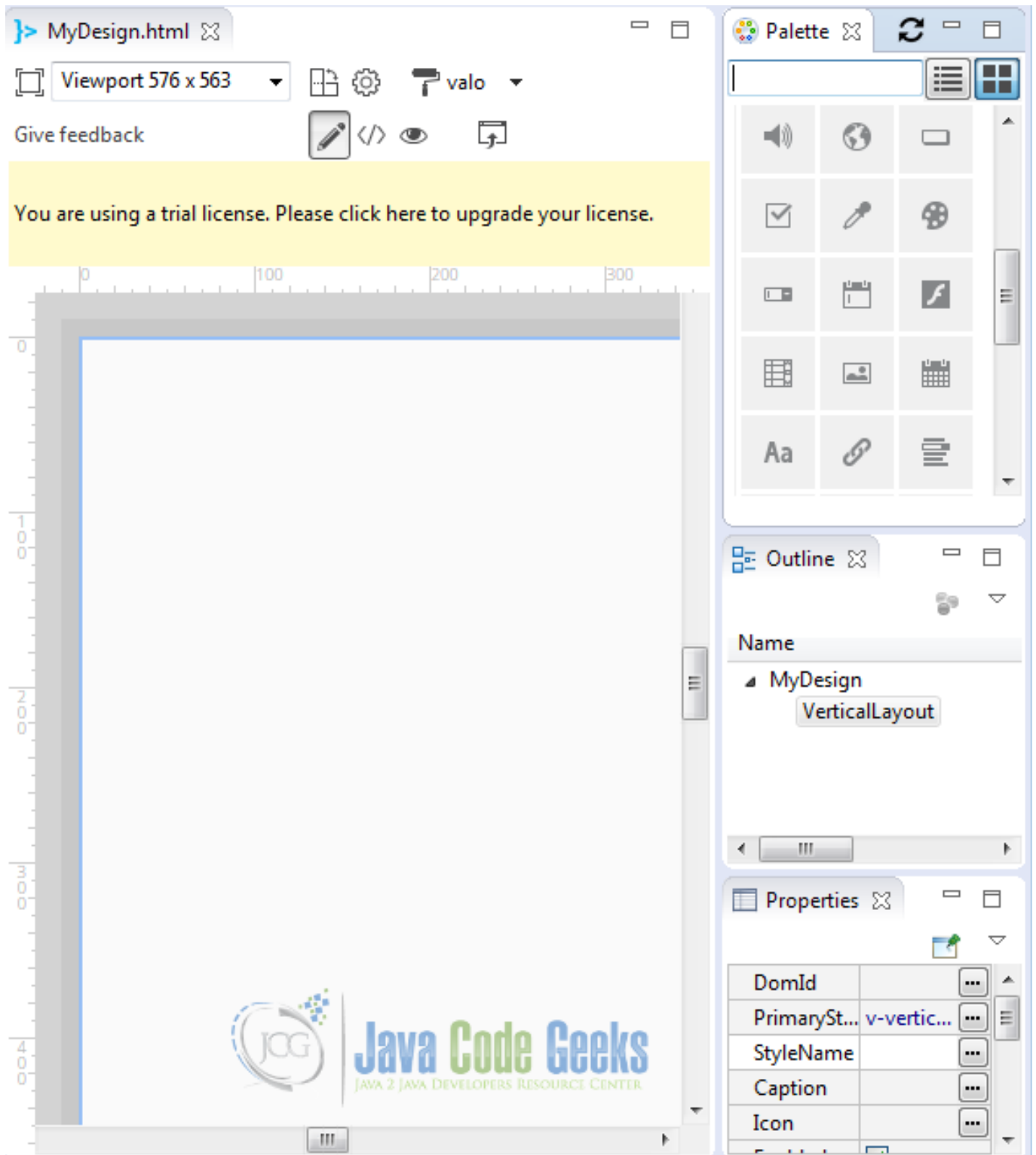


Figure 4.11: Designer perspective

## 4.5 Coding the example

From the widget palette, drag a button into the design view.

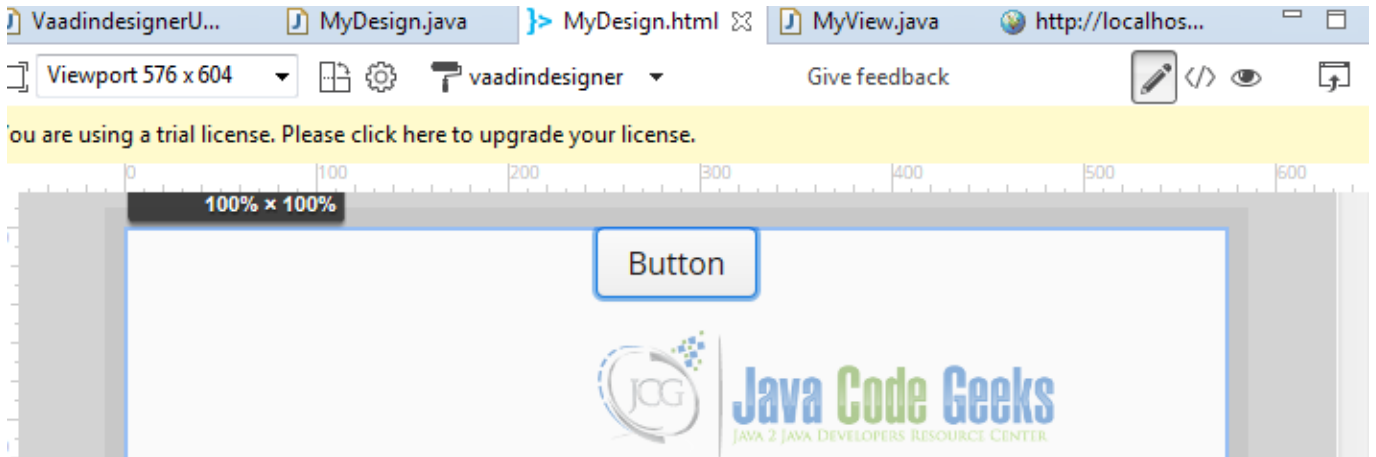


Figure 4.12: Button

Click on the button and you can see the controls to expand and position the widget on a predefined place of the screen. Center the button in the top center position of the screen, using the controls.



Figure 4.13: Center the button

Make sure the button is selected in the outline.

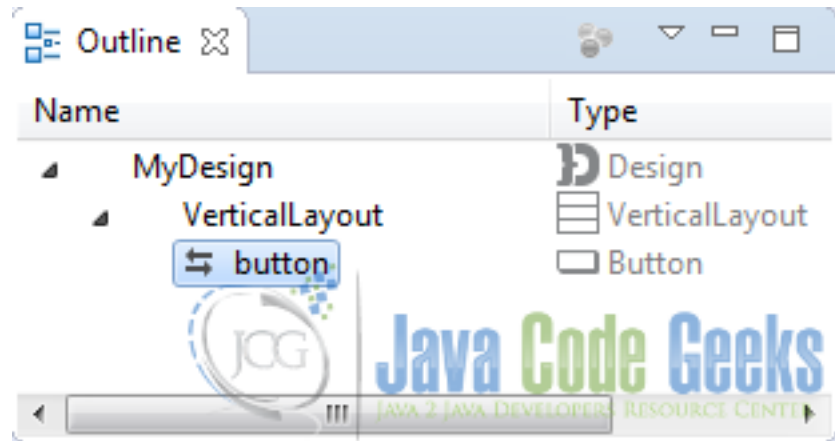


Figure 4.14: Select Button

Assign a name to the button in the properties panel to use it later.

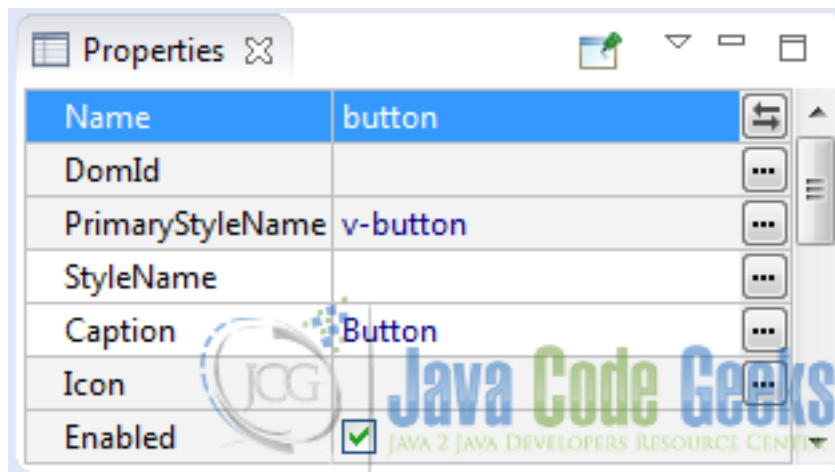


Figure 4.15: Button properties

Open the code created by the designer using the button in the toolbar.

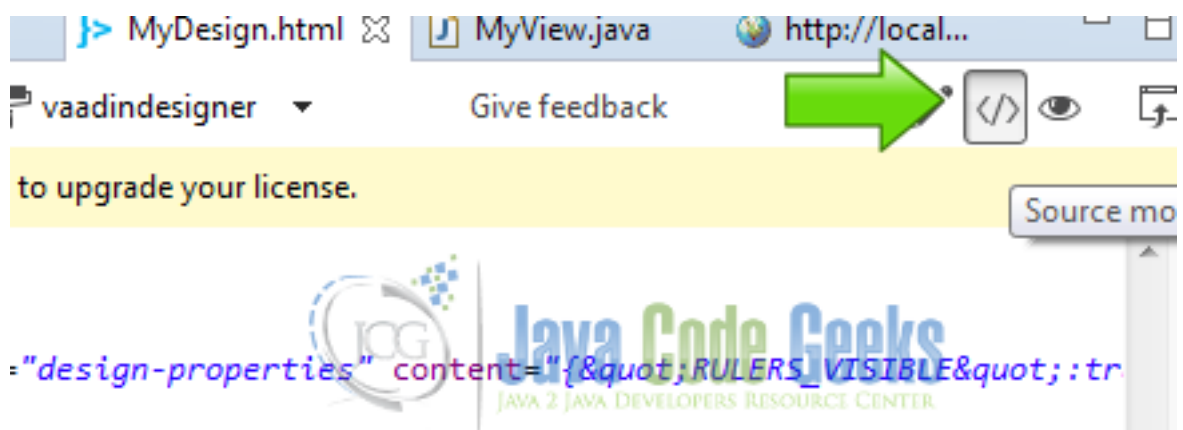


Figure 4.16: Designer Code

## Design code

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" name="design-properties" content="{\"RULERS_VISIBLE\":true,\" ←
      GUIDELINES_VISIBLE\":false,\"SNAP_TO_OBJECTS\":true,\"SNAP_TO_GRID\":true,\" ←
      SNAPPING_DISTANCE\":10,\"JAVA_SOURCES_ROOT\":\"src\",\"THEME\":\"vaadindesigner\"}">
  </head>
  <body>
    <vaadin-vertical-layout size-full>
      <vaadin-button plain-text _id="button" :center>
        Button
      </vaadin-button>
    </vaadin-vertical-layout>
  </body>
</html>
```

The code is a normal HTML5 document with special Vaadin tags for the layouts and the widgets. `vaadin-vertical-layout size-full` is the main vertical layout used when we created the design. `vaadin-button plain-text _id="button" :center` is our button.

The designer also created a class for every view created by the designer. Open the class `MyDesign.java`.

## MyDesign.java

```
package com.example.vaadin.designer;

import com.vaadin.annotations.AutoGenerated;
import com.vaadin.annotations.DesignRoot;
import com.vaadin.ui.Button;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.declarative.Design;

/**
 * !! DO NOT EDIT THIS FILE !!
 *
 * This class is generated by Vaadin Designer and will be overwritten.
 *
 * Please make a subclass with logic and additional interfaces as needed,
 * e.g class LoginView extends LoginDesign implements View { }
 */
@DesignRoot
@AutoGenerated
@SuppressWarnings("serial")
public class MyDesign extends VerticalLayout {
    protected Button button;

    public MyDesign() {
        Design.read(this);
    }
}
```

As you can see this class is only for Vaadin internal use. When you want to add code to the design you must create a subclass. `Design.read(this);` reads the declarative design for the given root component.

Create a subclass of my design to use it.

## MyDesign subclass

```
public class MyView extends MyDesign {

    private static final long serialVersionUID = 1L;
```

```

public MyView() {
    button.addClickListener(new ClickListener() {

        private static final long serialVersionUID = 1L;

        @Override
        public void buttonClick(ClickEvent event) {
            Notification.show("Button");
        }

    });
}
}

```

`public class MyView` extends `MyDesign` extends the design. `button.addClickListener(new ClickListener())` using the name of the button, adds a listener. `Notification.show("Button");` when the button is clicked a notification is shown.

Create the main class of our example.

#### Main Class

```

public class VaadindesignerUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadindesignerUI.class, ↵
        widgetset = "com.example.vaadindesigner.widgetset.VaadindesignerWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    @Override
    protected void init(VaadinRequest request) {
        MyView myview = new MyView();
        setContent(myview);
    }
}

```

In the `init` method of the main class: `MyView myview =new MyView();` instantiates the view, that is a subclass of the design. `setContent(myview);` sets the root of the project to the view.

## 4.6 The complete source code

### VaadindesignerUI.java

```

package com.example.vaadindesigner;

import javax.servlet.annotation.WebServlet;

import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.UI;

@SuppressWarnings("serial")
@Theme("vaadindesigner")
public class VaadindesignerUI extends UI {

```

```

@WebServlet(value = "/*", asyncSupported = true)
@VaadinServletConfiguration(productionMode = false, ui = VaadinDesignerUI.class, ←
    widgetset = "com.example.vaadindesigner.widgetset.VaadinDesignerWidgetset")
public static class Servlet extends VaadinServlet {
}

@Override
protected void init(VaadinRequest request) {
    MyView myview = new MyView();
    setContent(myview);
}
}

```

### MyDesign.java

```

package com.example.vaadindesigner;

import com.vaadin.annotations.AutoGenerated;
import com.vaadin.annotations.DesignRoot;
import com.vaadin.ui.Button;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.declarative.Design;

/**
 * !! DO NOT EDIT THIS FILE !!
 *
 * This class is generated by Vaadin Designer and will be overwritten.
 *
 * Please make a subclass with logic and additional interfaces as needed,
 * e.g class LoginView extends LoginDesign implements View { }
 */
@DesignRoot
@AutoGenerated
@SuppressWarnings("serial")
public class MyDesign extends VerticalLayout {
    protected Button button;

    public MyDesign() {
        Design.read(this);
    }
}

```

### MyView.java

```

package com.example.vaadindesigner;

import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;
import com.vaadin.ui.Notification;

public class MyView extends MyDesign {

    private static final long serialVersionUID = 1L;

    public MyView() {
        //Button myB = (Button) this.getComponent(0);
        button.addClickListener(new ClickListener() {

            private static final long serialVersionUID = 1L;

            @Override

```



```
        public void buttonClick(ClickEvent event) {
            Notification.show("Button");
        }
    });
}
}
```

### MyDesign.html

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8" name="design-properties" content="{\"RULERS_VISIBLE\":true,\" ←
GUIDELINES_VISIBLE\":false,\"SNAP_TO_OBJECTS\":true,\"SNAP_TO_GRID\":true,\" ←
SNAPPING_DISTANCE\":10,\"JAVA_SOURCES_ROOT\":\"src\",\"THEME\":\"vaadindesigner}\"}>
</head>
<body>
<vaadin-vertical-layout size-full>
<vaadin-button plain-text _id="button" :center>
    Button
</vaadin-button>
</vaadin-vertical-layout>
</body>
</html>
```

## 4.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 4.8 Results

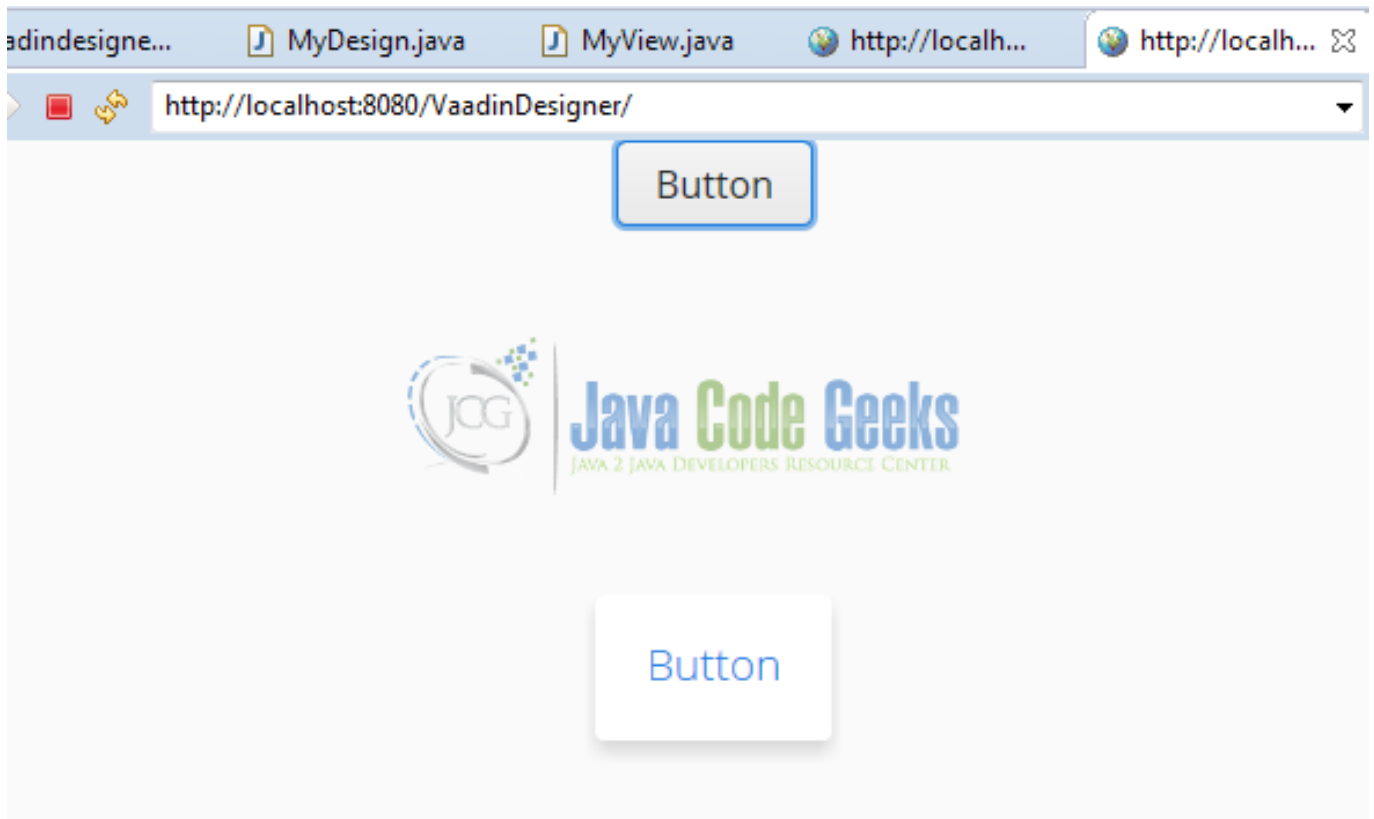


Figure 4.17: Result

As you can see, you have on the screen a button centered and when you click on it you get a Vaadin notification.

## 4.9 Download the Source Code

This was an example of: Vaadin Designer.

### Download

You can download the Eclipse project here: [VaadinDesigner](#)

## Chapter 5

# Vaadin Data Binding Example

Data binding is a technique that binds the provider of the data with the consumer. Whenever the data change in the provider or the consumer, the changes are reflected in the other side.

### 5.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.6
- Tomcat Server 8

### 5.2 Introduction

In this example we are going to bind widgets in Vaadin using some common techniques. We are going to use some text boxes to bind labels using the same data. This can be used to bind any widget with either, widgets or a back end with data as a data source.

### 5.3 Prerequisites

- JDK installed
- Eclipse Mars installed and working
- Vaadin plug-in installed
- Tomcat 8 installed and running

### 5.4 Set up the project

In the file menu choose File → New → Other

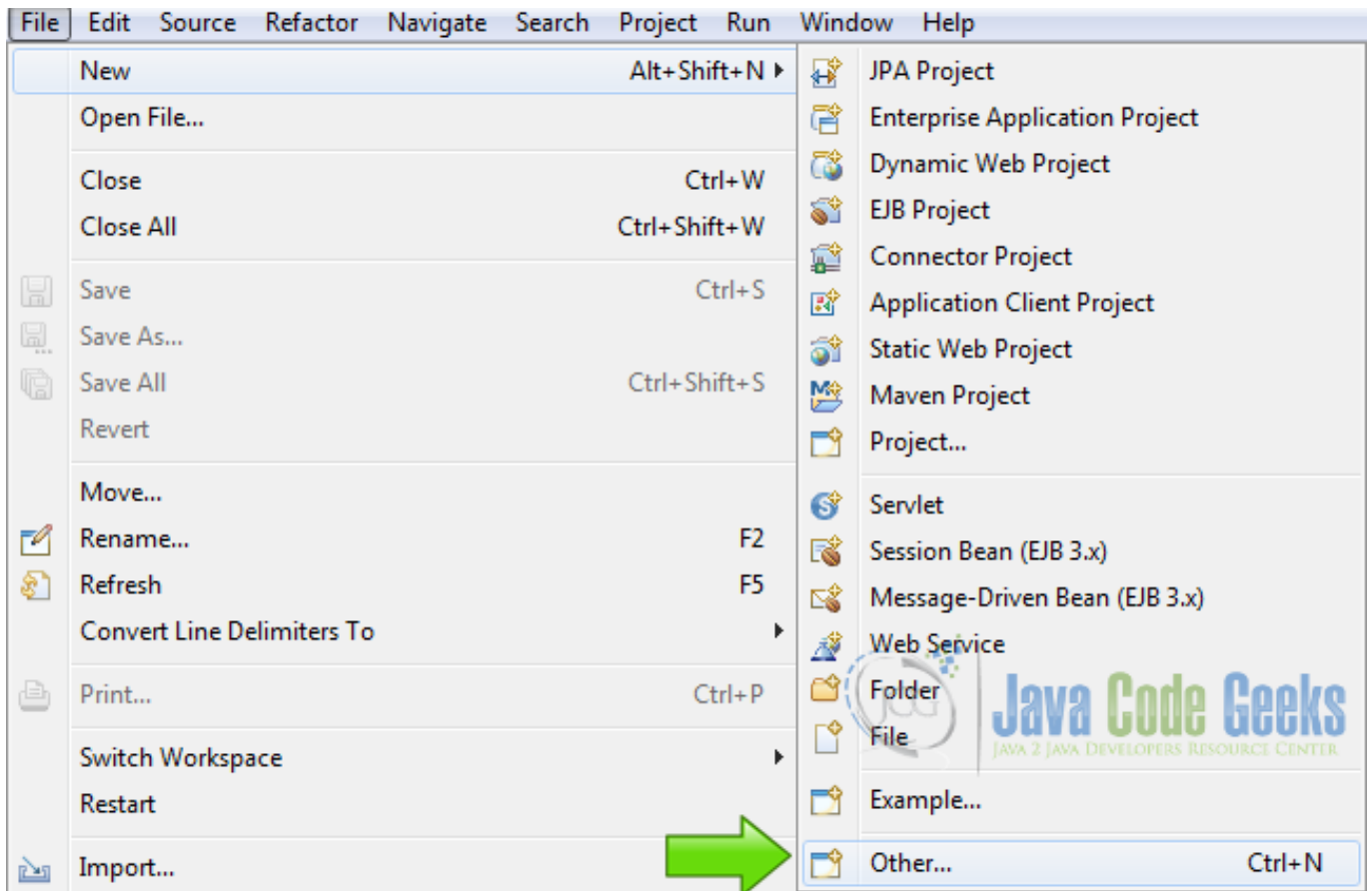


Figure 5.1: 1 New Project

Now from the list choose Vaadin 7 project

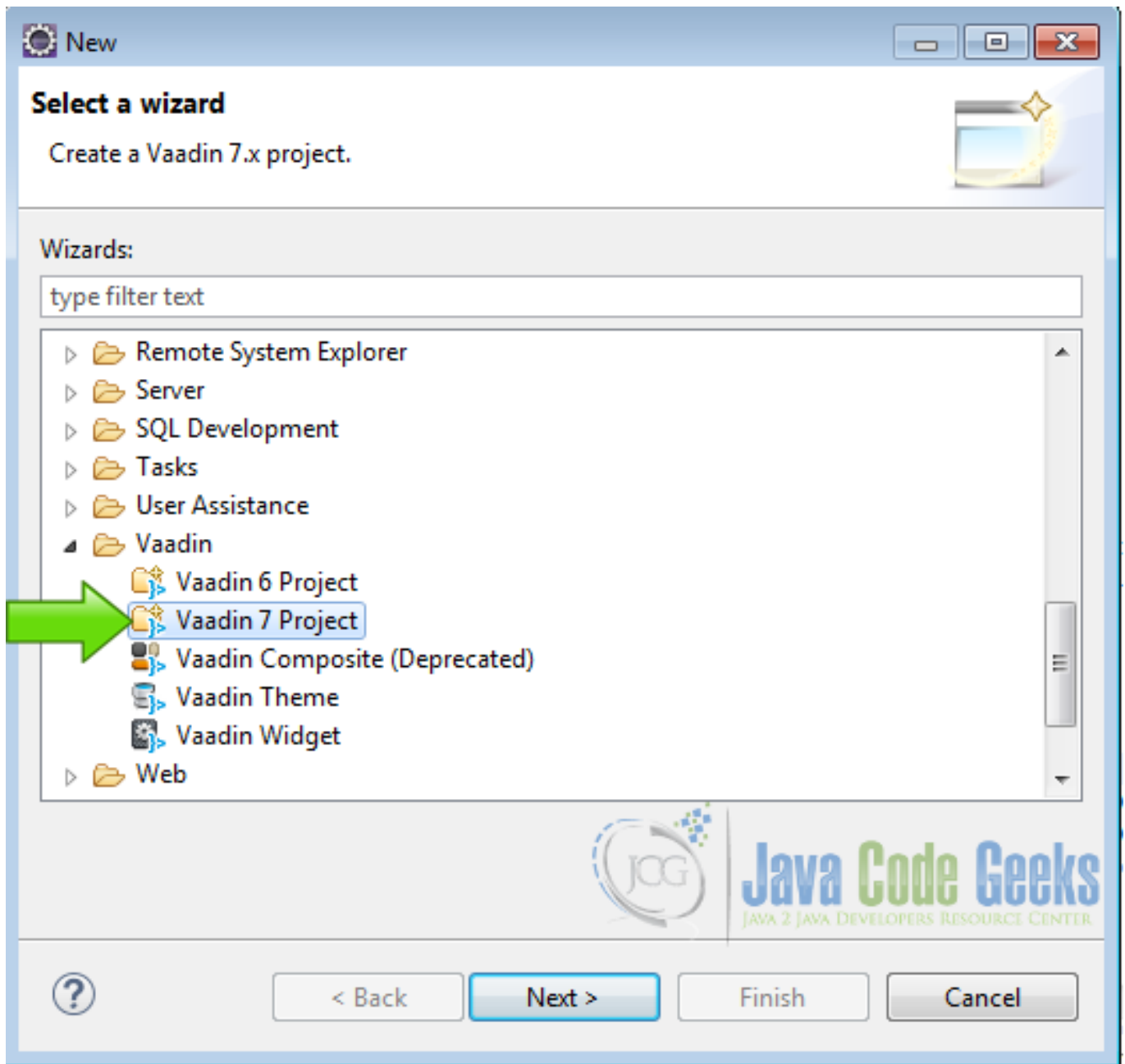


Figure 5.2: Vaadin Project

Hit next and name your project then hit finish.

## 5.5 Coding the example

Layouts

```
final VerticalLayout layout = new VerticalLayout();
layout.setSizeFull();
layout.setSpacing(true);
HorizontalLayout firstRowLayout = new HorizontalLayout();
firstRowLayout.setSizeFull();
```

```

firstRowlayout.setSpacing(true);
HorizontalLayout secondRowlayout = new HorizontalLayout();
secondRowlayout.setSizeFull();
secondRowlayout.setSpacing(true);
VerticalLayout stringLayout = new VerticalLayout();
VerticalLayout integerLayout = new VerticalLayout();
VerticalLayout doubleLayout = new VerticalLayout();
VerticalLayout beanLayout = new VerticalLayout();
VerticalLayout itemLayout = new VerticalLayout();
layout.setMargin(true);
setContent(layout);

```

`final VerticalLayout layout =new VerticalLayout();` creates the main vertical layout. `layout.setSizeFull();` sets the size of the main layout to occupy all the client screen. `layout.setSpacing(true);` for clarity we set spaces between the rows of the layout.

`HorizontalLayout firstRowlayout =new HorizontalLayout();` a first row of the vertical layout is horizontal layout. `firstRowlayout.setSizeFull();` sets the first row size to fill the screen. `firstRowlayout.setSpacing(true);` for better visualization of the example sets space between widgets.

`HorizontalLayout secondRowlayout =new HorizontalLayout();` second row of widgets. `secondRowlayout.setSizeFull();` second row layout size to full. `secondRowlayout.setSpacing(true);` second row space between widgets.

`VerticalLayout stringLayout =new VerticalLayout();` a layout for the string binding. `VerticalLayout integerLayout =new VerticalLayout();` a layout for the integer binding. `VerticalLayout doubleLayout =new VerticalLayout();` a layout for the double data type binding.

`VerticalLayout beanLayout =new VerticalLayout();` a layout for the bean binding. `VerticalLayout itemLayout =new VerticalLayout();` a layout for the item binding. `layout.setMargin(true);` sets the margin of the main layout.

`setContent(layout);` sets layout as main layout

### Object property

```

ObjectProperty propertyString = new ObjectProperty("string", String.class);
    ObjectProperty propertyInteger = new ObjectProperty(0, Integer.class);
    ObjectProperty propertyDouble = new ObjectProperty(0.0, Double.class);

```

Create some object properties to make the binding of the data types.

`ObjectProperty propertyString =new ObjectProperty("string", String.class);` creates object property of type string. `ObjectProperty propertyInteger =new ObjectProperty(0, Integer.class);` creates object property of type integer. `ObjectProperty propertyDouble =new ObjectProperty(0.0, Double.class);` creates object property of type double.

The `ObjectProperty` class creates a property data source using a data type, this binding allows to validate at the runtime the data binded to this property.

### Custom CSS style

```

.v-label-mylabelstyle {
    color: white;
    text-align: center;
    background-color: black;
    border-color: white;
    font-weight: bold;
}

```

This is a custom CSS style used in the binded labels.

`.v-label-mylabelstyle` sets the class name of the style. `color:white;` sets the foreground color to white. `text-align:center;` aligns the text to the center.

`background-color:black;` sets the background color of the label to black. `border-color:white;` sets the border color of the label to white. `font-weight:bold;` sets the font to bold.

### String TextField

```
TextField textFieldString = new TextField("String Text Field");
textFieldString.setWidth("200px");
textFieldString.setPropertyDataSource(propertyString);
textFieldString.setImmediate(true);
```

This text field is binded to the `propertyString`, all content of this text field is going to be a string.

`TextField textFieldString =new TextField("String Text Field");` creates the text field. `textFieldString.setWidth("200px");` sets width of the text field to 200 pixels.

`textFieldString.setPropertyDataSource(propertyString);` binds the text field to the property string. `textFieldString.setImmediate(true);` sets all the changes at the runtime sended immediate to the server.

### String label

```
Label labelString = new Label();
labelString.setWidth("200px");
labelString.addStyleName("mylabelstyle");
labelString.setPropertyDataSource(propertyString);
```

This label is binded to the property string and is gonna change every time the string text field changes.

`Label labelString =new Label();` creates the label. `labelString.setWidth("200px");` sets the label width to 200 pixels.

`labelString.addStyleName("mylabelstyle");` adds a custom style to the label. `labelString.setPropertyDataSource(propertyString);` binds the label to the property string using the property as a data source.

### Integer TextField

```
TextField textFieldInteger = new TextField("Integer Text Field");
textFieldInteger.setWidth("200px");
textFieldInteger.setPropertyDataSource(propertyInteger);
textFieldInteger.setImmediate(true);
```

This text field is binded to an integer data source.

`TextField textFieldInteger =new TextField("Integer Text Field");` creates the text field. `textFieldInteger.setWidth("200px");` sets the text field width to 200 pixels.

`textFieldInteger.setPropertyDataSource(propertyInteger);` binds the text field to the integer data source. `textFieldInteger.setImmediate(true);` sets the immediate mode.

### Integer Label

```
Label labelInteger = new Label(propertyInteger);
labelInteger.setWidth("200px");
labelInteger.addStyleName("mylabelstyle");
labelInteger.setPropertyDataSource(propertyInteger);
```

This label is binded to the integer data source.

`Label labelInteger =new Label(propertyInteger);` creates the label. `labelInteger.setWidth("200px");` sets the width of the label to 200 pixels.

`labelInteger.addStyleName("mylabelstyle");` adds a custom style to the label. `labelInteger.setPropertyDataSource(propertyInteger);` sets the integer data source to the label.

### Double TextField

```
TextField textFieldDouble = new TextField("Double Text Field");
textFieldDouble.setWidth("200px");
textFieldDouble.setPropertyDataSource(propertyDouble);
textFieldDouble.setImmediate(true);
```

Text field bounded to a double data source.

`TextField textFieldDouble = new TextField("Double Text Field");` creates the text field. `textFieldDouble.setWidth("200px");` sets the width of the text field to 200 pixels.

`textFieldDouble.setPropertyDataSource(propertyDouble);` sets the text field data source. `textFieldDouble.setImmediate(true);` sets the immediate mode.

### Double Label

```
Label labelDouble = new Label(propertyDouble);
labelDouble.setWidth("200px");
labelDouble.addStyleName("mylabelstyle");
labelDouble.setPropertyDataSource(propertyDouble);
```

This label is bounded to the double property data source.

`Label labelDouble = new Label(propertyDouble);` creates the label. `labelDouble.setWidth("200px");` sets the width of the label.

`labelDouble.addStyleName("mylabelstyle");` sets a custom style to the label. `labelDouble.setPropertyDataSource(propertyDouble);` sets the double property to the label.

### BindBean Class

```
public class BindBean {

    private String bindBeanString;

    public BindBean(String bindBeanString) {
        this.bindBeanString = bindBeanString;
    }

    public String getBindBeanString() {
        return bindBeanString;
    }

    public void setBindBeanString(String bindBeanString) {
        this.bindBeanString = bindBeanString;
    }

}
```

This class is a pojo java object that works as a data source for a bean field group.

`private String bindBeanString;` sets the string property. `public BindBean(String bindBeanString)` sets the constructor.

`public String getBindBeanString()` sets the getter. `public void setBindBeanString(String bindBeanString)` sets the setter.

### BindFieldGroup

```
BindBean bindBean = new BindBean("string-Bind-Bean");
BeanFieldGroup beanItem = new BeanFieldGroup(BindBean.class);
beanItem.setItemDataSource(bindBean);
TextField textFieldBean = (TextField) beanItem.buildAndBind("BindBeanString", "↔
    bindBeanString");
textFieldBean.setWidth("200px");
```



We create a bind field group that is binded to a pojo class and set it as a text field data source.

```
BindBean bindBean =new BindBean("string-Bind-Bean"); creates a new instance to the BindBean class. BeanFieldGroup beanItem =new BeanFieldGroup(BindBean.class); creates a BeanFieldGroup using the BindBean class as skeleton.
```

```
beanItem.setItemDataSource(bindBean); sets the data source of the beanItem BeanFieldGroup using the instance created before. TextField textFieldBean =(TextField) beanItem.buildAndBind("BindBeanString", "bindBeanString"); creates a text field with the beanItem, this is possible because the BeanFieldGroup only have one field.
```

```
textFieldBean.setWidth("200px"); sets the width of the text field.
```

#### labelBean

```
Label labelBean = new Label(textFieldBean);  
labelBean.setWidth("200px");  
labelBean.addStyleName("mylabelstyle");  
labelBean.setPropertyDataSource(textFieldBean);
```

This label is binded to the bean text field so every time the text field changes this label is gonna change too.

```
Label labelBean =new Label(textFieldBean); creates the label. labelBean.setWidth("200px"); sets the width to the label.
```

```
labelBean.addStyleName("mylabelstyle"); adds a custom style to the label. labelBean.setPropertyDataSource(textFieldBean); sets the data source of the label.
```

#### java

```
PropertysetItem item = new PropertysetItem();  
item.addItemProperty("ItemProperty", new ObjectProperty("item-property"));  
TextField textFieldItemProperty = new TextField("Item Property TextField");  
textFieldItemProperty.setWidth("200px");  
FieldGroup fieldGroup = new FieldGroup(item);  
fieldGroup.bind(textFieldItemProperty, "ItemProperty");
```

We bind a field group property to a text field as it data source.

```
PropertysetItem item =new PropertysetItem(); creates a new property set item. item.addItemProperty("ItemProperty", new ObjectProperty("item-property")); adds a new string property to the property set item.
```

```
TextField textFieldItemProperty =new TextField("Item Property TextField"); creates a text field. textFieldItemProperty.setWidth("200px"); sets the width of the text field.
```

```
FieldGroup fieldGroup =new FieldGroup(item); creates a field group. fieldGroup.bind(textFieldItemProperty, "ItemProperty"); binds the text field to the field group string property.
```

#### Item Property Label

```
Label labelItem = new Label(textFieldItemProperty);  
labelItem.setWidth("200px");  
labelItem.addStyleName("mylabelstyle");  
labelItem.setPropertyDataSource(textFieldItemProperty);
```

This label is binded to the item property text field.

```
Label labelItem =new Label(textFieldItemProperty); creates the label. labelItem.setWidth("200px"); sets the width of the label.
```

```
labelItem.addStyleName("mylabelstyle"); adds a custom style to the label. labelItem.setPropertyDataSource(textFieldItemProperty); binds the label to the text field.
```

Add the widgets to the layouts

```
stringLayout.addComponent(textFieldString);
stringLayout.addComponent(labelString);
integerLayout.addComponent(textFieldInteger);
integerLayout.addComponent(labelInteger);
doubleLayout.addComponent(textFieldDouble);
doubleLayout.addComponent(labelDouble);
beanLayout.addComponent(textFieldBean);
beanLayout.addComponent(labelBean);
itemLayout.addComponent(textFieldItemProperty);
itemLayout.addComponent(labelItem);

firstRowLayout.addComponent(stringLayout);
firstRowLayout.addComponent(integerLayout);
firstRowLayout.addComponent(doubleLayout);

secondRowLayout.addComponent(beanLayout);
secondRowLayout.addComponent(itemLayout);

layout.addComponent(firstRowLayout);
layout.addComponent(secondRowLayout);
```

Add all the widgets to the layouts.

## 5.6 The complete source code

VaadindatabindingUI.java

```
package com.example.vaadindatabinding;

import javax.servlet.annotation.WebServlet;

import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.data.fieldgroup.BeanFieldGroup;
import com.vaadin.data.fieldgroup.FieldGroup;
import com.vaadin.data.util.ObjectProperty;
import com.vaadin.data.util.PropertysetItem;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Label;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadindatabinding")
public class VaadindatabindingUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadindatabindingUI.class, widgetset = "com.example.vaadindatabinding.widgetset.VaadindatabindingWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
```

```
layout.setSizeFull();
layout.setSpacing(true);
HorizontalLayout firstRowLayout = new HorizontalLayout();
firstRowLayout.setSizeFull();
firstRowLayout.setSpacing(true);
HorizontalLayout secondRowLayout = new HorizontalLayout();
secondRowLayout.setSizeFull();
secondRowLayout.setSpacing(true);
VerticalLayout stringLayout = new VerticalLayout();
VerticalLayout integerLayout = new VerticalLayout();
VerticalLayout doubleLayout = new VerticalLayout();
VerticalLayout beanLayout = new VerticalLayout();
VerticalLayout itemLayout = new VerticalLayout();
layout.setMargin(true);
setContent(layout);

ObjectProperty propertyString = new ObjectProperty("string", String.class);
ObjectProperty propertyInteger = new ObjectProperty(0, Integer.class);
ObjectProperty propertyDouble = new ObjectProperty(0.0, Double.class);

TextField textFieldString = new TextField("String Text Field");
textFieldString.setWidth("200px");
textFieldString.setPropertyDataSource(propertyString);
textFieldString.setImmediate(true);

Label labelString = new Label();
labelString.setWidth("200px");
labelString.addStyleName("mylabelstyle");
labelString.setPropertyDataSource(propertyString);

TextField textFieldInteger = new TextField("Integer Text Field");
textFieldInteger.setWidth("200px");
textFieldInteger.setPropertyDataSource(propertyInteger);
textFieldInteger.setImmediate(true);

Label labelInteger = new Label(propertyInteger);
labelInteger.setWidth("200px");
labelInteger.addStyleName("mylabelstyle");
labelInteger.setPropertyDataSource(propertyInteger);

TextField textFieldDouble = new TextField("Double Text Field");
textFieldDouble.setWidth("200px");
textFieldDouble.setPropertyDataSource(propertyDouble);
textFieldDouble.setImmediate(true);

Label labelDouble = new Label(propertyDouble);
labelDouble.setWidth("200px");
labelDouble.addStyleName("mylabelstyle");
labelDouble.setPropertyDataSource(propertyDouble);

BindBean bindBean = new BindBean("string-Bind-Bean");
BeanFieldGroup beanItem = new BeanFieldGroup(BindBean.class);
beanItem.setItemDataSource(bindBean);
TextField textFieldBean = (TextField) beanItem.buildAndBind("BindBeanString ←", "bindBeanString");
textFieldBean.setWidth("200px");

Label labelBean = new Label(textFieldBean);
labelBean.setWidth("200px");
labelBean.addStyleName("mylabelstyle");
labelBean.setPropertyDataSource(textFieldBean);
```

```
PropertysetItem item = new PropertysetItem();
item.addItemProperty("ItemProperty", new ObjectProperty("item-property"));
TextField textFieldItemProperty = new TextField("Item Property TextField");
textFieldItemProperty.setWidth("200px");
FieldGroup fieldGroup = new FieldGroup(item);
fieldGroup.bind(textFieldItemProperty, "ItemProperty");

Label labelItem = new Label(textFieldItemProperty);
labelItem.setWidth("200px");
labelItem.addStyleName("mylabelstyle");
labelItem.setPropertyDataSource(textFieldItemProperty);

stringLayout.addComponent(textFieldString);
stringLayout.addComponent(labelString);
integerLayout.addComponent(textFieldInteger);
integerLayout.addComponent(labelInteger);
doubleLayout.addComponent(textFieldDouble);
doubleLayout.addComponent(labelDouble);
beanLayout.addComponent(textFieldBean);
beanLayout.addComponent(labelBean);
itemLayout.addComponent(textFieldItemProperty);
itemLayout.addComponent(labelItem);

firstRowLayout.addComponent(stringLayout);
firstRowLayout.addComponent(integerLayout);
firstRowLayout.addComponent(doubleLayout);

secondRowLayout.addComponent(beanLayout);
secondRowLayout.addComponent(itemLayout);

layout.addComponent(firstRowLayout);
layout.addComponent(secondRowLayout);

}

}
```

### BindBean.java

```
package com.example.vaadinbinding;

public class BindBean {

    private String bindBeanString;

    public BindBean(String bindBeanString){
        this.bindBeanString = bindBeanString;
    }

    public String getBindBeanString() {
        return bindBeanString;
    }

    public void setBindBeanString(String bindBeanString) {
        this.bindBeanString = bindBeanString;
    }

}
```

### vaaindatabinding.scss

```
@import "../valo/valo.scss";

@mixin vaadindatabinding {
  @include valo;

  .v-label-mylabelstyle {
    color: white;
    text-align: center;
    background-color: black;
    border-color: white;
    font-weight: bold;
  }
}
```

## 5.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 5.8 Results



Figure 5.3: String TextField

The string data source.

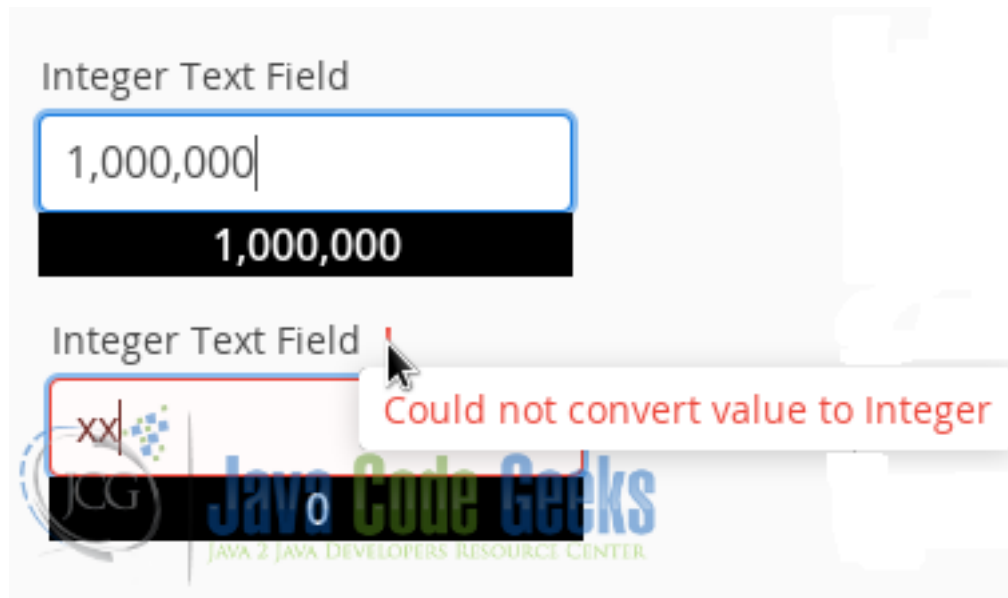


Figure 5.4: Integer TextField

The integer data source with validation check.

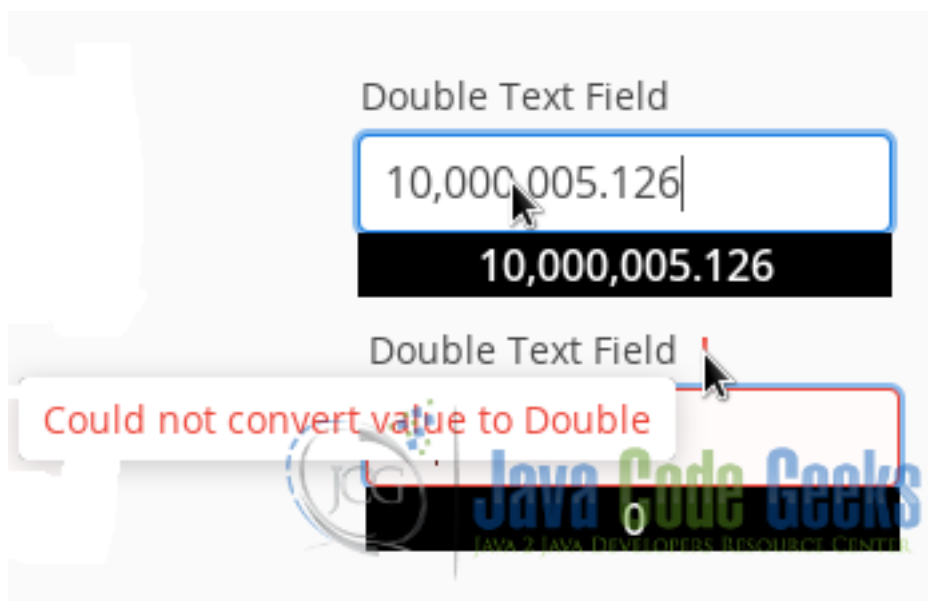


Figure 5.5: Double TextField

The double data source with validation check.



Figure 5.6: BindBean

The bean data source.



Figure 5.7: Item Property

The item property data source.

## 5.9 Download the Source Code

This was an example of: Vaadin data binding.

### Download

You can download the Eclipse project here: [VaadinDataBinding](#)

## Chapter 6

# Vaadin Rest Example

REST stands for Representational State Transfer. Created by Roy Fielding in the year 2000, is a communication protocol where everything is a resource. REST principal characteristics are: client-server, stateless, cache-able, layered and uniform interface to access resources. Resources are accessed using a stateless protocol like HTTP. REST allows text, XML, JSON and other resources. PUT, GET, POST and DELETE methods are implemented in the REST architecture. Java defines REST using the JAX-RS Specification (JSR) 311. JAX-RS is implemented using Jersey.

### 6.1 Introduction

In this example we are going to create a server using Jersey with text, XML and JSON resources and access these resources from Vaadin, to show it on the user interface. First we are going to create the server and run it. Next we are going to create the Vaadin client and consume the resources on the server.

### 6.2 Prerequisites

- JDK installed
- Eclipse Mars installed and working
- Vaadin 7.6.5 plug-in installed
- Tomcat 8 installed and running

### 6.3 Set up the server project

#### 6.3.1 Download Jersey

Download latest Jersey library from [Download Jersey](#), uncompress the zip file and save it for later.

---



## Jersey - RESTful Web Services in Java.

# Jersey

## RESTful Web Services in Java.

### Jersey 2.x

Jersey 2.22.2, that implements [JAX-RS 2.0 API](#) API is the most recent release of Jersey [Release Notes](#).

For the convenience of non-maven developers the following links are provided:

- [Jersey JAX-RS 2.0 RI bundle](#)  contains the JAX-RS 2.0 API jar, all the core Je
- [Jersey 2.22.2 Examples bundle](#) provides convenient access to the Jersey 2 exampl

All the Jersey 2 release binaries, including the source & apidocs jars, are available for dc [maven repository](#) as well as from the [java.net maven repository](#).

Chances are you are using Apache Maven as a build & dependency management tool & 2.22.2 by generating the skeleton application from one of the Jersey 2 maven archetyp server container, use

Figure 6.1: Download Jersey

### 6.3.2 Create the server project

Fire up eclipse and choose from the menu `File->New->Other`. From the list choose `Dynamic Web Project`.

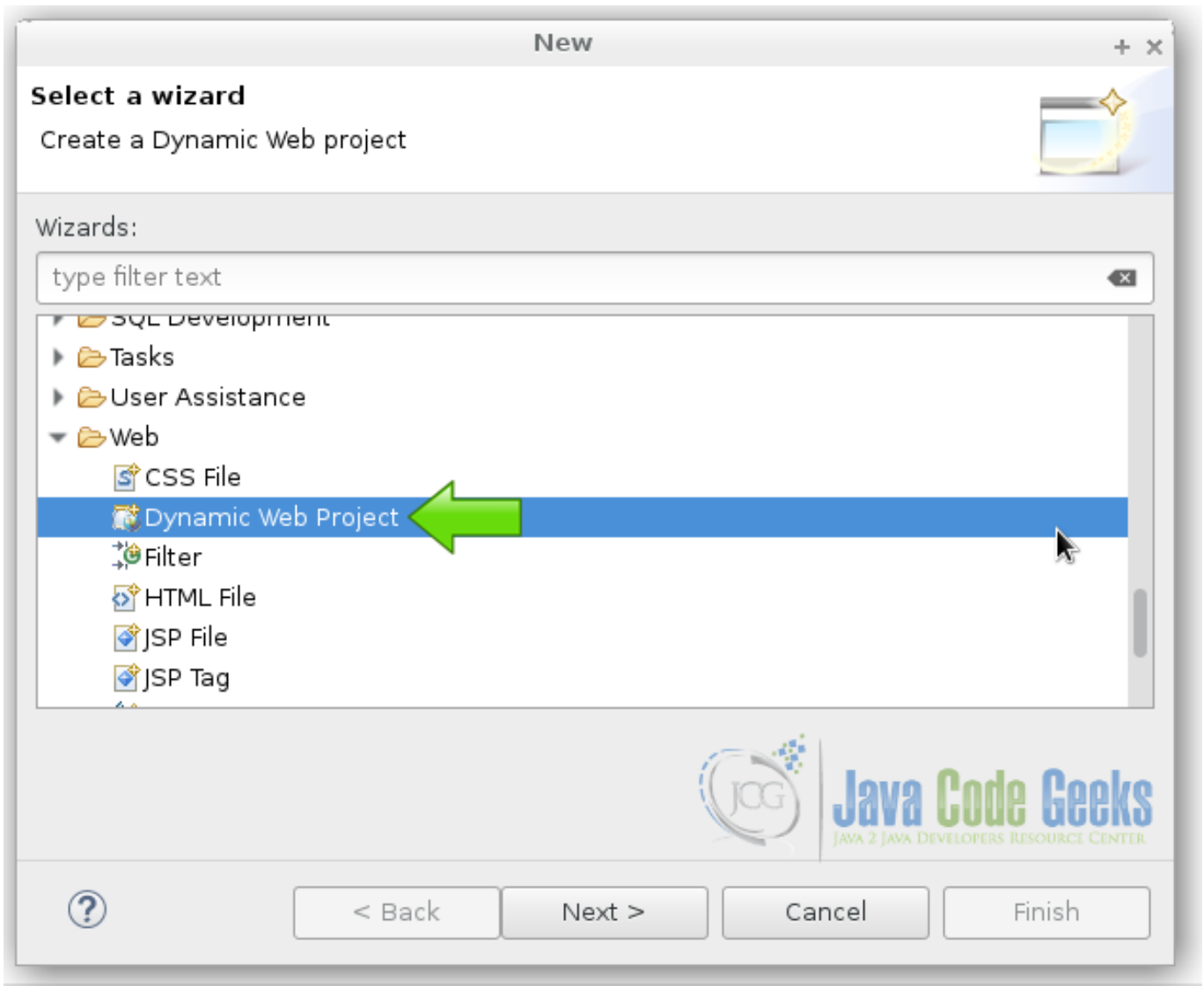


Figure 6.2: Create dynamic web project

### 6.3.3 Copy Jersey Files

Copy all the Jersey jar files from the downloaded zip into the folder `WebContent->WEB-INF->lib`.

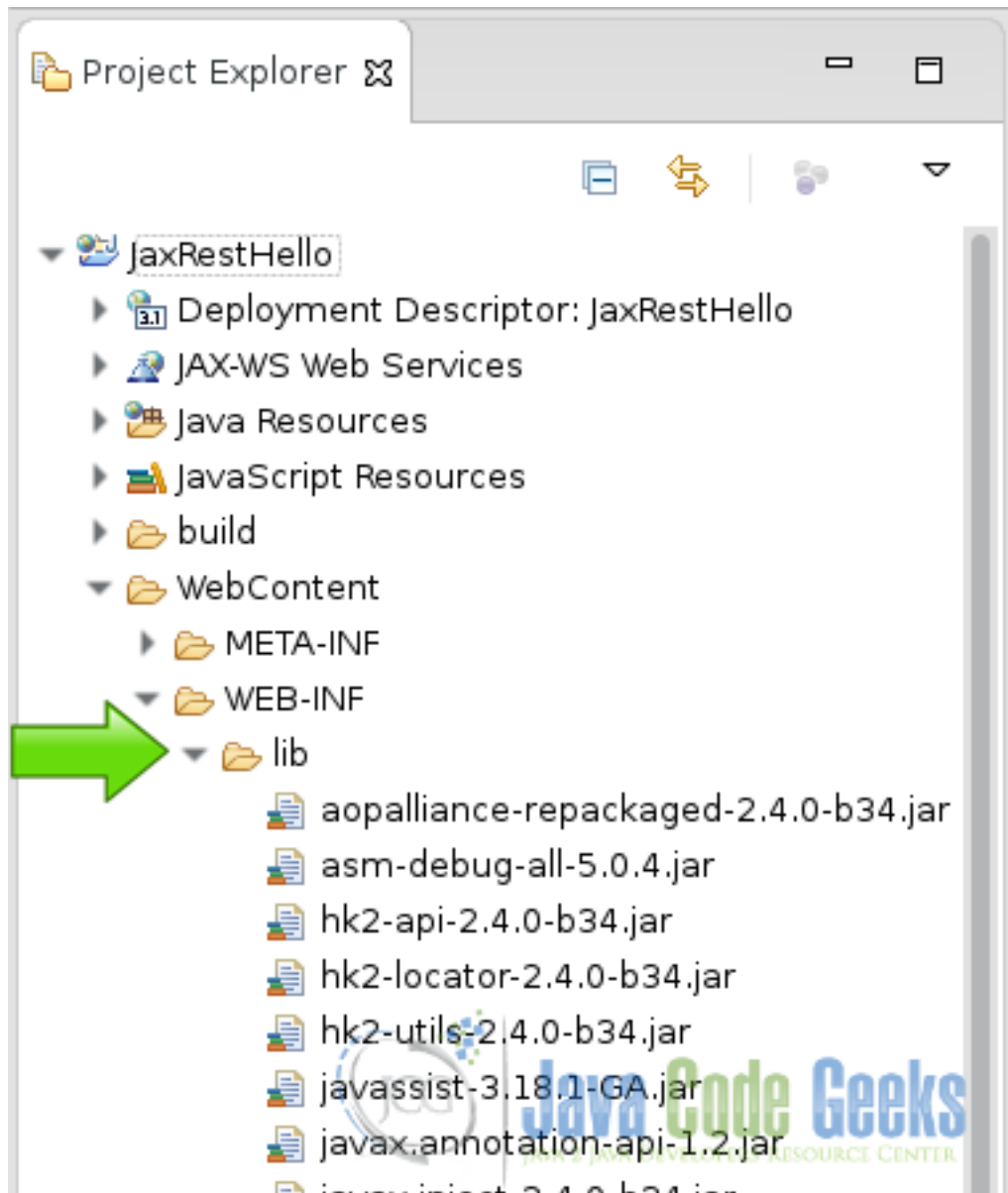


Figure 6.3: Copy jars into lib folder

## 6.4 Coding the server

Create a new pojo java class.

Server class

```
@Path("/restserver")
public class RestServer {

}
```

This class use the annotation `@Path` of the specification JAX-RS. It's a plain old java object because inherits no one. The JAX-RS specification of Jersey uses annotations to create the REST services.

Text response

```
@GET
@Produces(MediaType.TEXT_PLAIN)
public String serverMessageText() {
    return "This is a message from the server";
}
```

When you make an html request with a GET with plain text as the type of the data, this content is served. The annotation @GET, serves an html GET request and the annotation @Produces, specify the type of data of the response according to the request. The server first analyzes the request and then based on the data type requested it sends the response. Request/Response is the standard html communication protocol. REST uses the standard request/response communication protocol.

#### XML response

```
@GET
@Produces(MediaType.TEXT_XML)
public String serverMessageXML() {
    return " This is a message from the server ";
}
```

When you make an html request with a GET with XML as the type of the data, this content is served.

#### JSON response

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public String serverMessageJSON() {
    return "[{'message' : 'server json message', 'content' : 'message content'}]";
}
```

When you make an html request with a GET with JSON as the type of the data, this content is served.

## 6.5 The server web.xml

In the server we are going to use the web.xml configuration to expose our REST services and define our paths.

#### Servlet

```
<servlet>
  <servlet-name>Rest Server</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>com.example</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

This defines the servlet and map the appropriate jersey class to our package. To make the Jersey library recognizes where are the content to be served.

#### Servlet mapping

```
<servlet-mapping>
  <servlet-name>Rest Server</servlet-name>
  <url-pattern>/server/*</url-pattern>
</servlet-mapping>
```

The servlet mapping, maps the content of our class to the URI used in the browser.

## 6.6 Create the client project

In the file menu choose File → New → Other

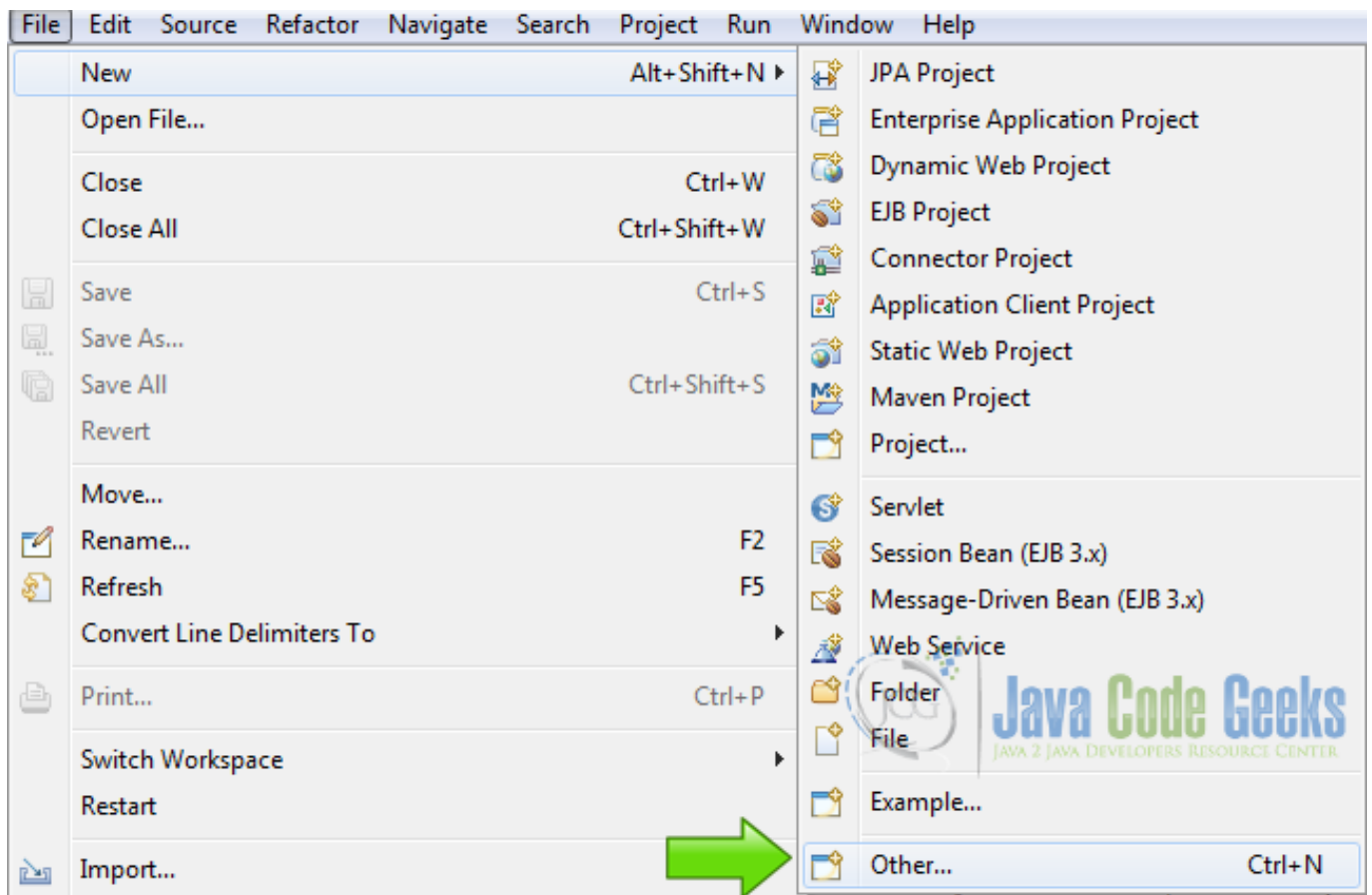


Figure 6.4: New Project

Now from the list choose Vaadin 7 project

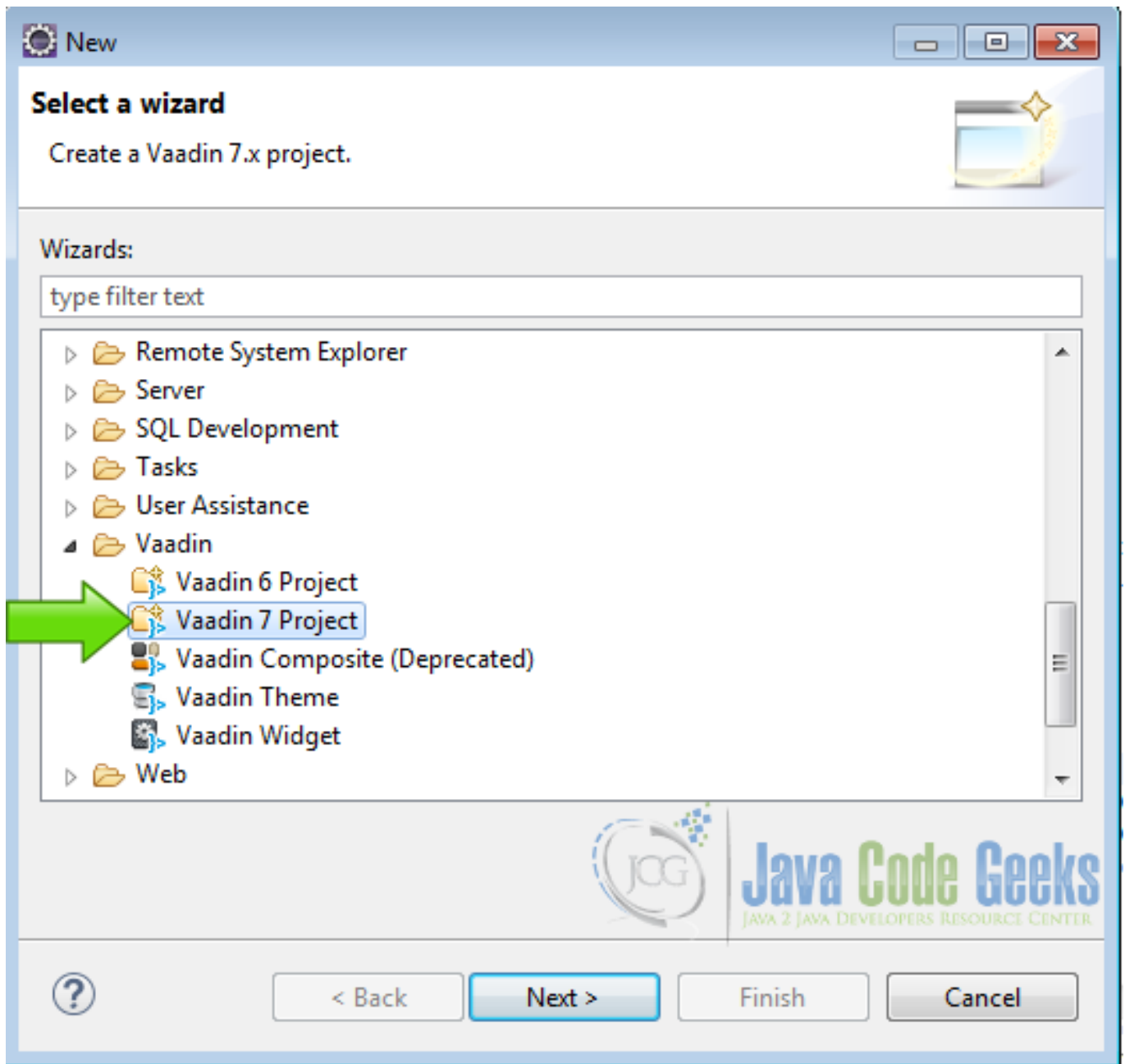


Figure 6.5: Vaadin Project

Press next and name your project then press finish.

Copy all the Jersey jar files from the downloaded zip into the folder `WebContent->WEB-INF->lib`. As we did with the server.

## 6.7 Coding the client project

### 6.7.1 Client Class to access the server

We are going to create a supplementary client class to create the server REST services. Create a POJO class:

## POJO client

```
public class RestClient {  
  
}
```

In this class we are going to create all the client methods we need to access the data on the server.

### fields

```
ClientConfig config;  
Client client;  
WebTarget webtarget;
```

Create the fields of the the Jersey REST client.

### Constructor

```
public RestClient() {  
    config = new ClientConfig();  
    client = ClientBuilder.newClient(config);  
    webtarget = client.target("https://localhost:8080/JaxRestHello");  
}
```

In the constructor of the Jersey client, we have our base URI to access the REST services.

### getResponse

```
public String getResponse() {  
    String response = webtarget.path("server").  
        path("restserver").  
        request().  
        accept(MediaType.TEXT_PLAIN).  
        get(Response.class).toString();  
  
    return response;  
}
```

This is a method to get the response from the server. The response could tell us the state of the server and the availability of the services.

### Get text answer

```
public String getAnswerText() {  
    String answer =  
        webtarget.path("server").  
        path("restserver").  
        request().  
        accept(MediaType.TEXT_PLAIN).  
        get(String.class);  
  
    return answer;  
}
```

This is a method to get a plain text answer. In the field `accept`, we specify the type of the response. With REST, the data between the client and the server is just plain text but here we could know the kind of structure these plain text has. In this example it could be just plain text, XML or JSON.

### Get XML answer

```
public String getAnswerXML() {  
    String answer =  
        webtarget.path("server").  
        path("restserver").  
        request().  
        accept(MediaType.TEXT_XML).  
}
```

```
        get (String.class);  
    }  
    return answer;  
}
```

This method gets an XML answer from the server, note the fields `path`, these fields are append to the base URI created in the constructor of the class.

#### Get JSON

```
public String getAnswerJSON() {  
    String answer =  
        webtarget.path("server").  
        path("restserver").  
        request().  
        accept(MediaType.APPLICATION_JSON).  
        get (String.class);  
    return answer;  
}
```

JSON is just plain text with some predefined structure.

## 6.7.2 Vaadin UI

In the `init` class of our Vaadin application we create an instance of the client class to access the REST services from our web application. Remember the REST server could be anywhere in the Internet. It could be on the same server as the client but we can have multiples REST servers all around the world and a client accessing all of them.

#### Main layout

```
final VerticalLayout layout = new VerticalLayout();  
layout.setMargin(true);  
setContent(layout);
```

We create the main layout of our application.

#### Rest client Instance

```
RestClient restClient = new RestClient();
```

We create an instance of our REST client, to use it from the Vaadin user interface.

#### Feedback label

```
Label resultLabel = new Label();  
resultLabel.setWidth("100%");  
resultLabel.setHeight("200px");  
resultLabel.addStyleName("h1");  
resultLabel.addStyleName("mylabelstyle");
```

We create a label to put the feedback from our server, and make some adjustments and styles to the label just for this example purpose.

#### Layout for buttons

```
HorizontalLayout hLayot = new HorizontalLayout();
```

We are going to create some buttons to get different answers from the server, and we organize it on an horizontal layout.

#### Get Response



```
Button buttonGetResponse = new Button("Get Response");
buttonGetResponse.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        resultLabel.setValue(restClient.getResponse());
    }
});
```

This is a button to get the server response. With this response we can check the status of the server.

#### Text Answer

```
Button buttonGetText = new Button("Get Text Answer");
buttonGetText.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        resultLabel.setValue(restClient.getAnswerText());
    }
});
```

This is a button to get the text answer from the REST server.

#### XML Answer

```
Button buttonGetXml = new Button("Get XML Answer");
buttonGetXml.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        resultLabel.setValue(restClient.getAnswerXML());
    }
});
```

With this button we get an XML response from the server. We are using the client class created before to call these methods.

#### JSON Answer

```
Button buttonGetJson = new Button("Get JSON Answer");
buttonGetJson.addClickListener(new Button.ClickListener() {
    public void buttonClick(ClickEvent event) {
        resultLabel.setValue(restClient.getAnswerJSON());
    }
});
```

A JSON answer from the server, JSON has been very popular lately and you could transfer it with REST.

#### Add elements to the layout

```
hLayout.addComponent(buttonGetResponse);
hLayout.addComponent(buttonGetText);
hLayout.addComponent(buttonGetXml);
hLayout.addComponent(buttonGetJson);
layout.addComponent(resultLabel);
layout.addComponent(hLayout);
```

Here we add all elements to the layout.

## 6.8 The complete source code

### 6.8.1 The server source code

RestServer.java

```
package com.example;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/restserver")
public class RestServer {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String serverMessageText() {
        return "This is a message from the server";
    }

    @GET
    @Produces(MediaType.TEXT_XML)
    public String serverMessageXML() {
        return " This is a message from the server ";
    }

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public String serverMessageJSON() {
        return "[{'message' : 'server json message', 'content' : 'message content ←
        '}]";
    }
}
```

#### web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xmlns="https://xmlns.jcp. ←
    org/xml/ns/javaee" xsi:schemaLocation="https://xmlns.jcp.org/xml/ns/javaee https://xmlns ←
    .jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>JaxRestHello</display-name>
  <servlet>
    <servlet-name>Rest Server</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>com.example</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Rest Server</servlet-name>
    <url-pattern>/server/*</url-pattern>
  </servlet-mapping>
</web-app>
```

## 6.8.2 The client Source Code

### RestClient.java

```
package com.example.vaadin_rest_example;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
```

```
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import org.glassfish.jersey.client.ClientConfig;

public class RestClient {
    ClientConfig config;
    Client client;
    WebTarget webtarget;

    public RestClient() {
        config = new ClientConfig();
        client = ClientBuilder.newClient(config);
        webtarget = client.target("https://localhost:8080/JaxRestHello");
    }

    public String getResponse() {
        String response = webtarget.path("server").
            path("restserver").
            request().
            accept(MediaType.TEXT_PLAIN).
            get(Response.class).toString();

        return response;
    }

    public String getAnswerText() {
        String answer =
            webtarget.path("server").
            path("restserver").
            request().
            accept(MediaType.TEXT_PLAIN).
            get(String.class);

        return answer;
    }

    public String getAnswerXML() {
        String answer =
            webtarget.path("server").
            path("restserver").
            request().
            accept(MediaType.TEXT_XML).
            get(String.class);

        return answer;
    }

    public String getAnswerJSON() {
        String answer =
            webtarget.path("server").
            path("restserver").
            request().
            accept(MediaType.APPLICATION_JSON).
            get(String.class);

        return answer;
    }
}
```

#### Vaadin\_rest\_exampleUI.java

```
package com.example.vaadin_rest_example;

import javax.servlet.annotation.WebServlet;
```

```
import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Label;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadin_rest_example")
public class Vaadin_rest_exampleUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = Vaadin_rest_exampleUI. ←
        class, widgetset = "com.example.vaadin_rest_example.widgetset. ←
            Vaadin_rest_exampleWidgetset")
    public static class Servlet extends VaadinServlet {
    }

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);

        RestClient restClient = new RestClient();

        Label resultLabel = new Label();
        resultLabel.setWidth("100%");
        resultLabel.setHeight("200px");
        resultLabel.addStyleName("h1");
        resultLabel.addStyleName("mylabelstyle");
        HorizontalLayout hLayout = new HorizontalLayout();

        Button buttonGetResponse = new Button("Get Response");
        buttonGetResponse.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                resultLabel.setValue(restClient.getResponse());
            }
        });

        Button buttonGetText = new Button("Get Text Answer");
        buttonGetText.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                resultLabel.setValue(restClient.getAnswerText());
            }
        });

        Button buttonGetXml = new Button("Get XML Answer");
        buttonGetXml.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                resultLabel.setValue(restClient.getAnswerXML());
            }
        });

        Button buttonGetJson = new Button("Get JSON Answer");
        buttonGetJson.addClickListener(new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
```

```
                resultLabel.setValue(restClient.getAnswerJSON());
            }
        });

        hLayout.addComponent(buttonGetResponse);
        hLayout.addComponent(buttonGetText);
        hLayout.addComponent(buttonGetXml);
        hLayout.addComponent(buttonGetJson);
        layout.addComponent(resultLabel);
        layout.addComponent(hLayout);
    }
}
```

#### vaadin\_rest\_example.scss

```
@import "../valo/valo.scss";

@mixin vaadin_rest_example {
    @include valo;

    .v-label-mylabelstyle {
        color: white;
        text-align: center;
        background-color: black;
        border-color: white;
        font-weight: bold;
    }
}
```

## 6.9 Running the example

First right click on the server project folder and choose Run as → Run on server choose Tomcat 8 server and press finish. When the server is running right click on the Vaadin project folder and choose Run as → Run on server choose Tomcat 8 server and press finish.

## 6.10 Results

### 6.10.1 Get the response

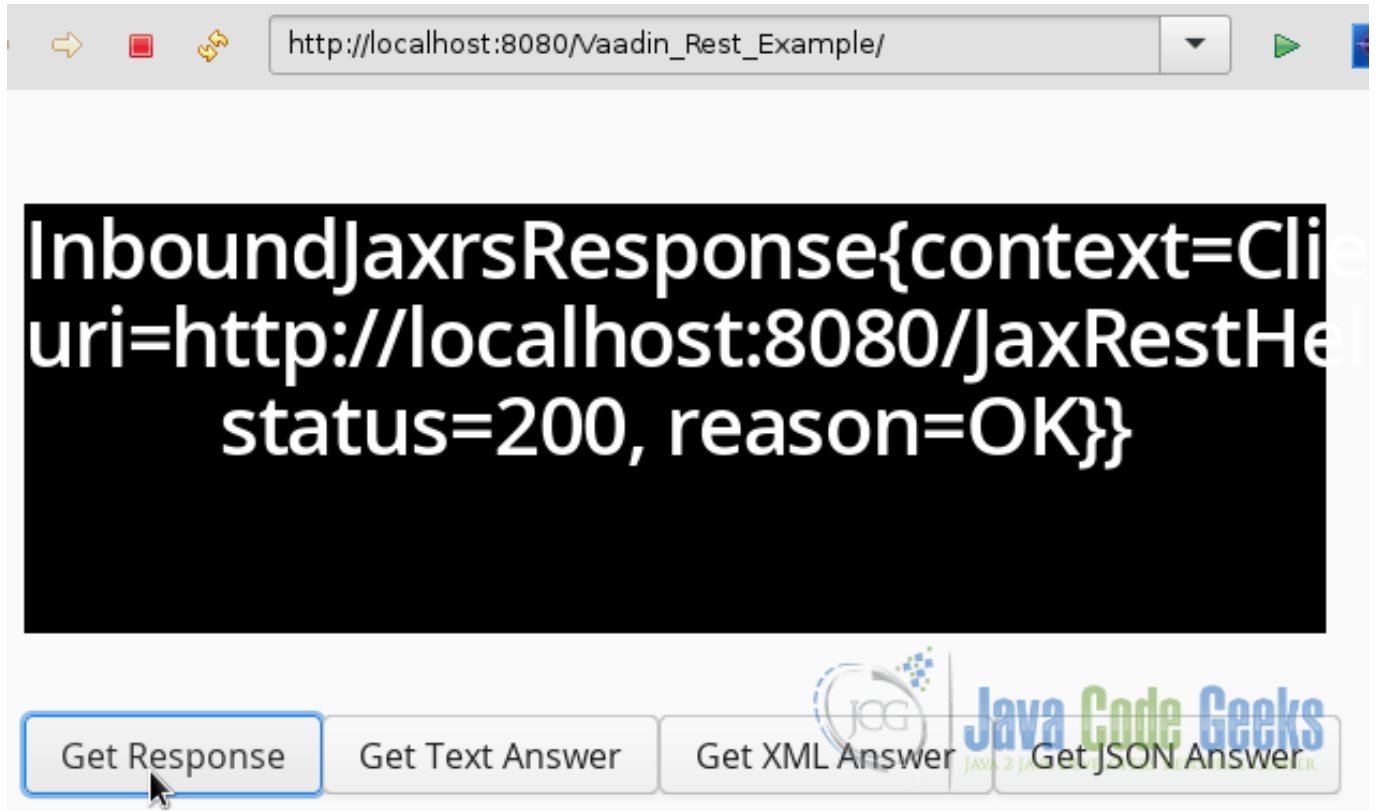


Figure 6.6: Get Response

Get the status of the server

## 6.10.2 Get the plain text

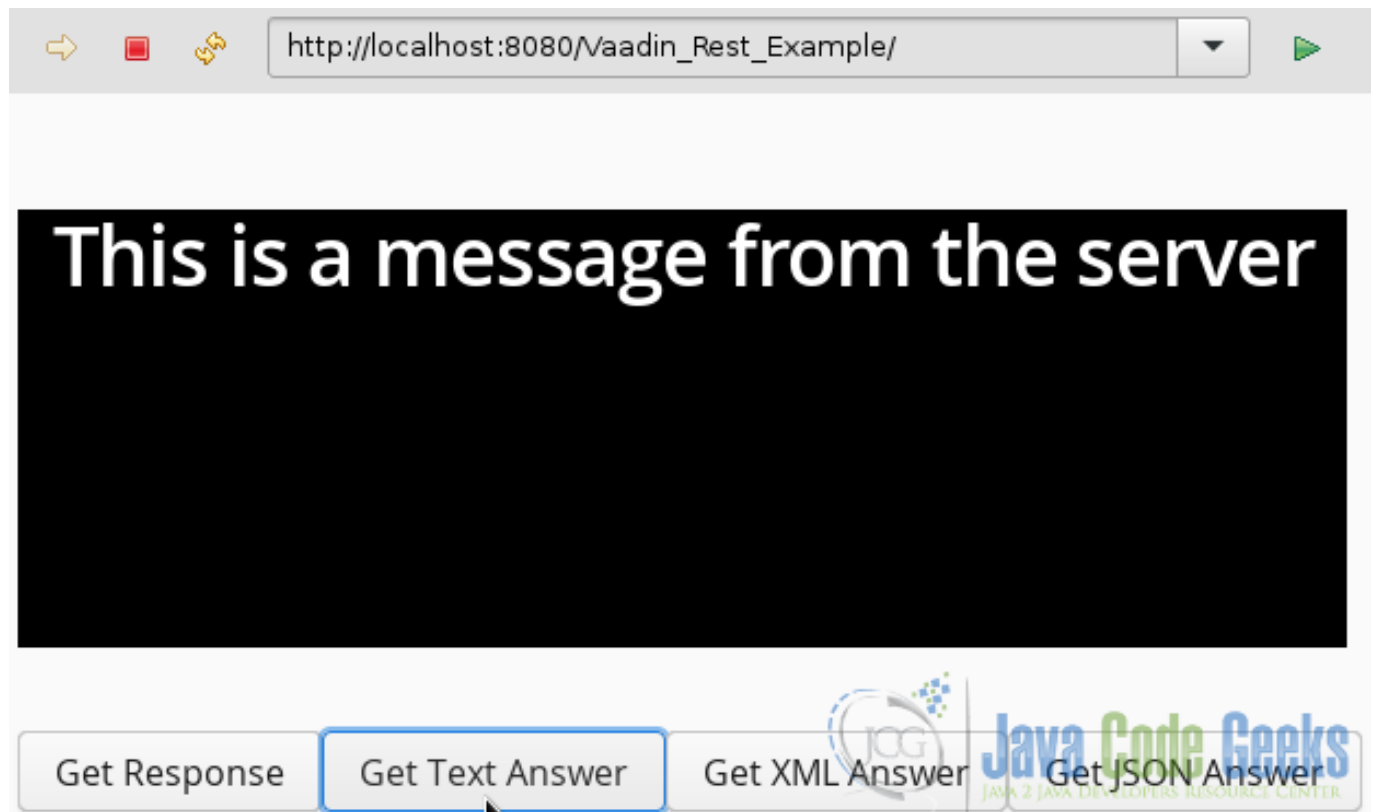


Figure 6.7: Get Text

Get the plain text answer from the server

### 6.10.3 Get the XML

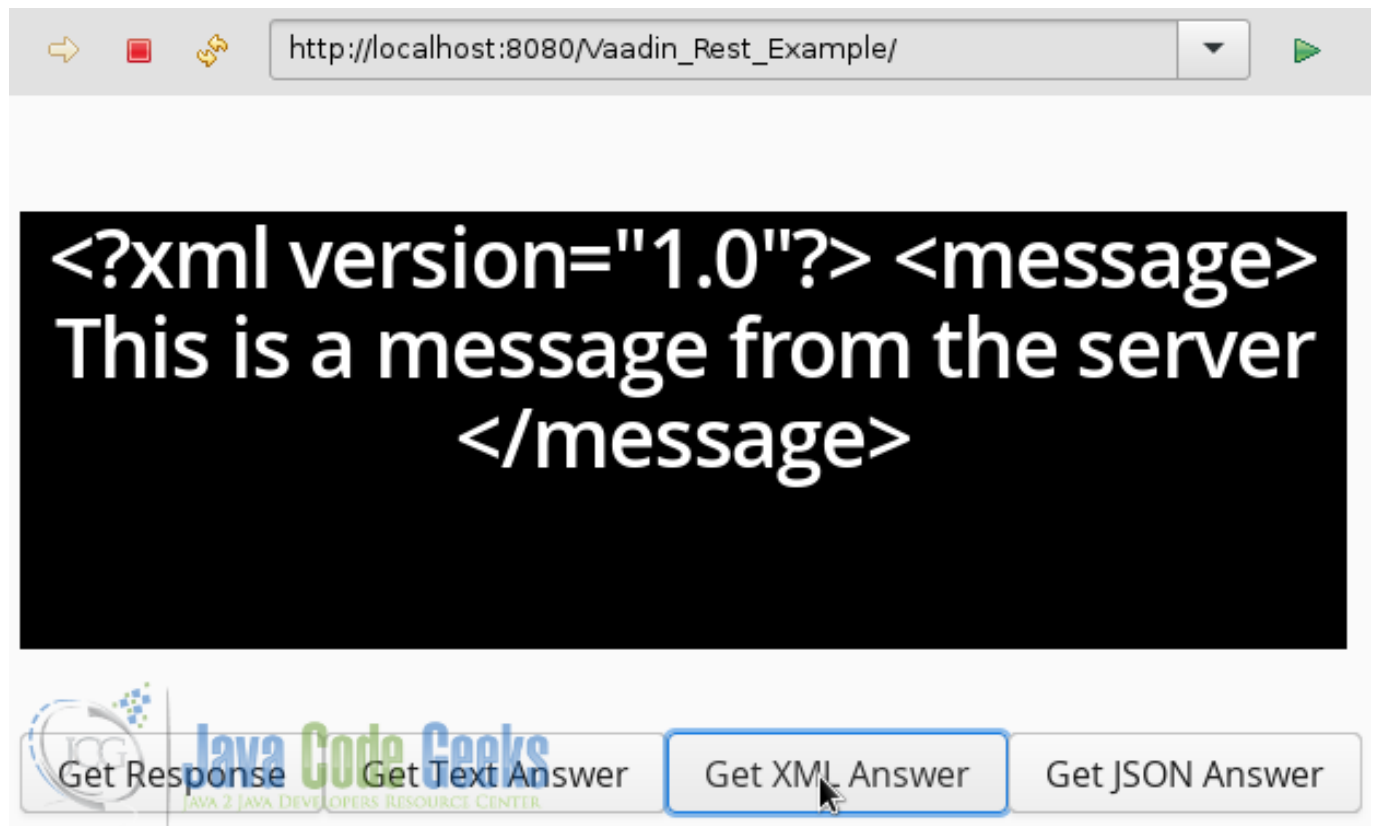


Figure 6.8: Get XML

Get the XML answer from the server



### 6.10.4 Get the JSON



Figure 6.9: Get JSON

Get the JSON answer from the server

## 6.11 Download the Source Code

This was an example of: Vaadin REST.

### Download

You can download the Eclipse project here: [VaadinRestExample](#)

## Chapter 7

# Vaadin Container Example

The relational model is the organization of data into collections of two-dimensional tables called relations, the relational data model was developed for databases but you can use this model to group sets of items and define relations between these sets, a container of sets of items and its relations.

### 7.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.4
- Tomcat Server 8

### 7.2 Introduction

Vaadin Container is a set of Items, that have a relational model, you can use it to easily wrap a database and associate it with the Vaadin controls that implements the `Container` interface. It's use is not restricted to a database, you can use it to model your own sets of items and it's relations inside your application, and get advantage of the strong structure that the Vaadin container provides. A container must follow certain rules to use it:

- Each Item have the same number of Properties.
- Each Item have an ID property.
- All Properties in different items must have the same data type.
- The Item ID of a container id unique and non-null.

In this example I am going to show how to use a Vaadin Container.

### 7.3 Prerequisites

- JDK installed
  - Eclipse Mars installed and working
  - Vaadin 7.6.4 plugin installed
  - Tomcat 8 installed and running
-

## 7.4 Set up the project

In the file menu choose File → New → Other.

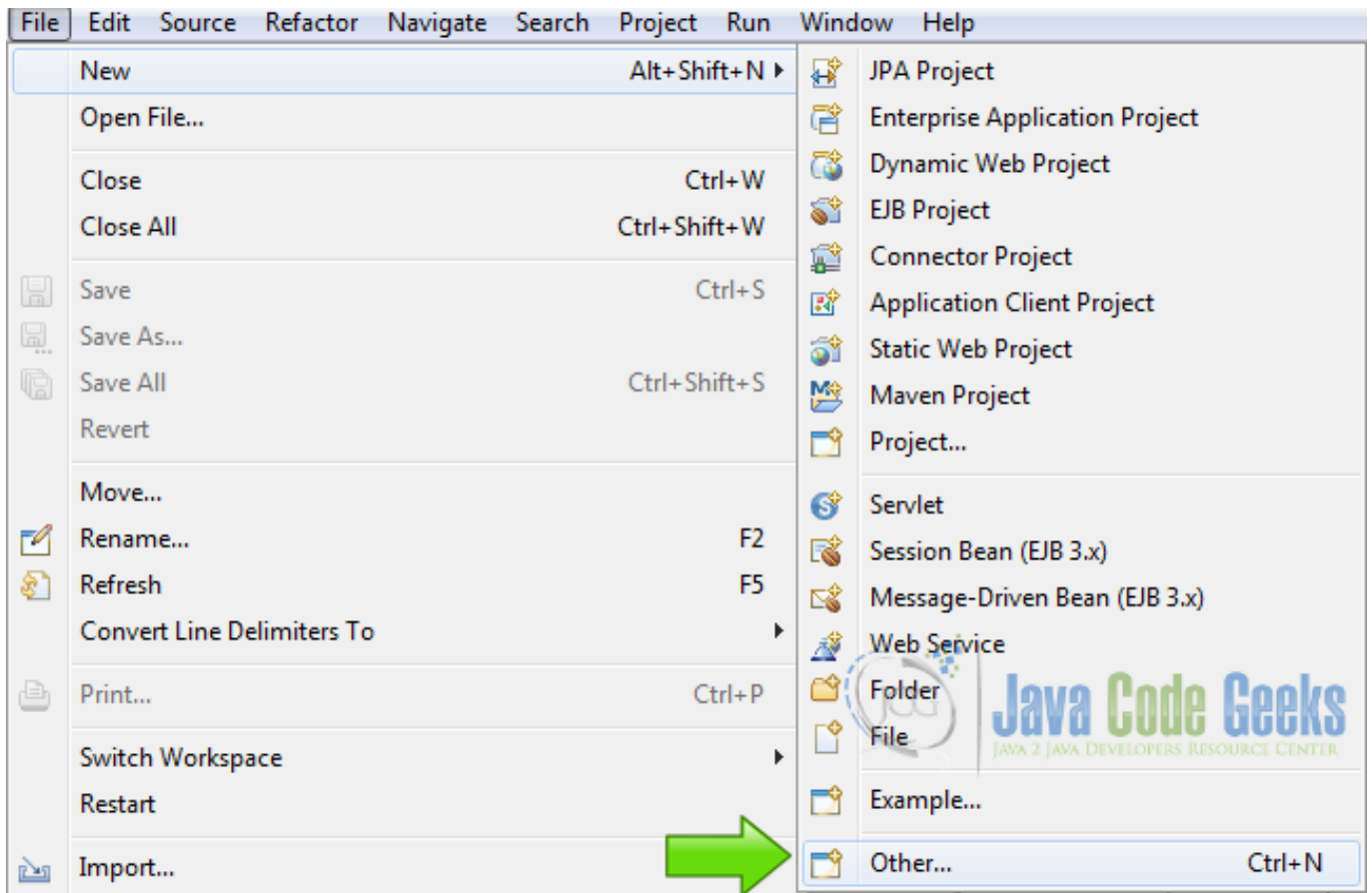


Figure 7.1: New Project

Now from the list choose Vaadin 7 project.

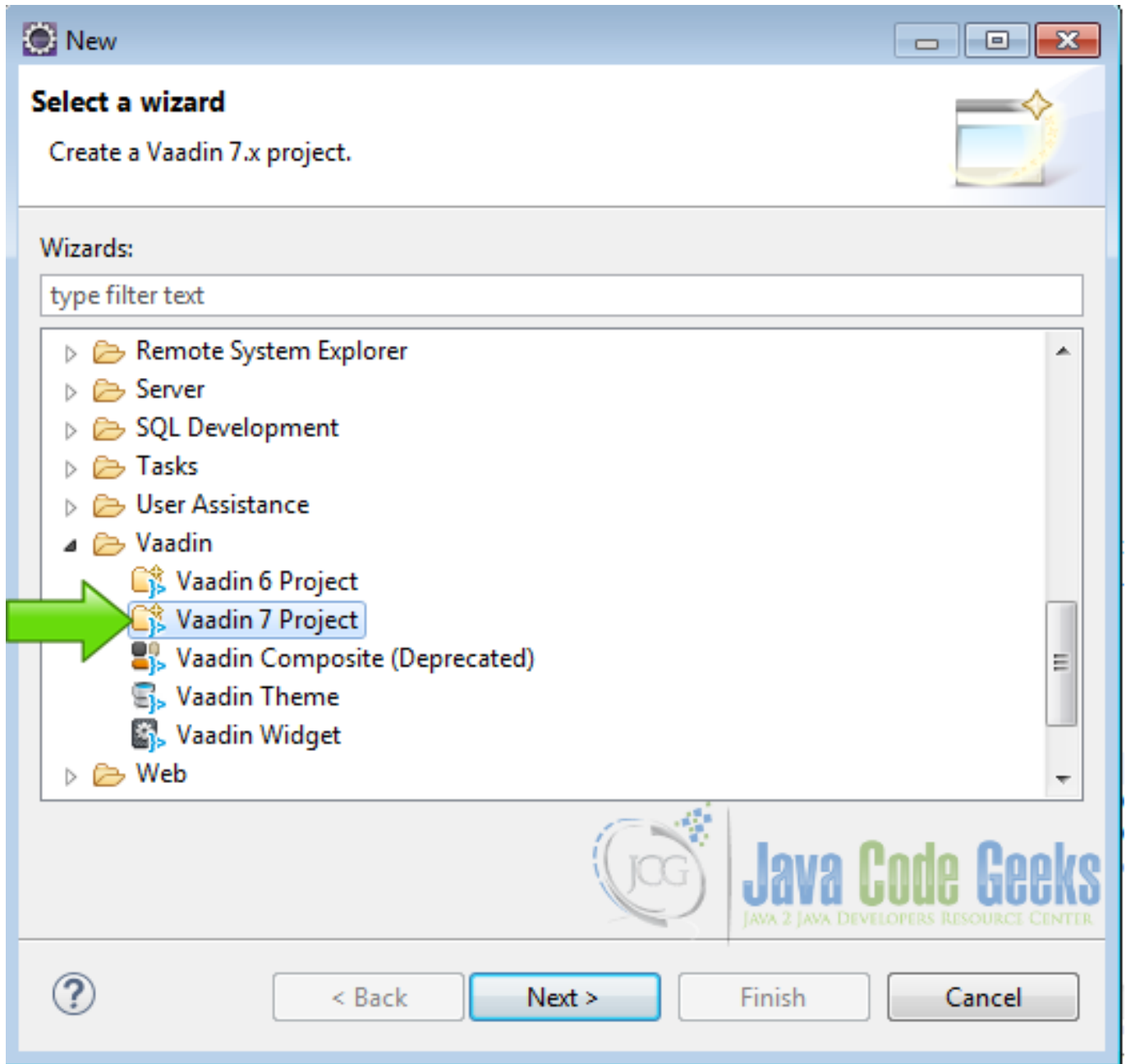


Figure 7.2: Vaadin Project

Hit next and name your project then hit finish.

## 7.5 Coding the example

### 7.5.1 MyBean BeanContainer

The BeanContainer is a container for JavaBean objects. It is a simple Java class that implements Serializable and has all the fields that we want in our container.

MyBean.java

```
package com.example.vaadincontainer;

import java.io.Serializable;

public class MyBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;
    private String surname;
    private MySubBean osUsed;

    public MyBean (String pname, String psurname, MySubBean pSubBean){

        this.name = pname;
        this.surname = psurname;
        this.osUsed = pSubBean;

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public MySubBean getOsUsed() {
        return osUsed;
    }

    public void setOsUsed(MySubBean osUsed) {
        this.osUsed = osUsed;
    }

}
```

MyBean has two string fields called name and surname and another bean which is called MySubBean with a parent-child relationship. This is a simple pojo Java class with the field declarations, a single constructor `public MyBean (String pname, String psurname, MySubBean pSubBean)` and standard getters and setters for each property field. Also we have here MySubBean fields that are the fields that define the nested relationship, having as a member another bean.

## 7.5.2 MySubBean BeanContainer

MySubBean.java

```
package com.example.vaadincontainer;

import java.io.Serializable;
```

```
public class MySubBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String os;

    public MySubBean(String pOs) {
        this.os = pOs;
    }

    public String getOs() {
        return os;
    }

    public void setOs(String os) {
        this.os = os;
    }

}
```

This is the child bean and has only one string field, to the purpose of this tutorial a nested bean is working. As before this is a simple pojo Java class that has the field declarations, a constructor, getters and setters.

## 7.5.3 The UI

### 7.5.3.1 Random name generator

Random OS

```
private MySubBean randomOS()
{
    Random osr = new Random();
    int osn = osr.nextInt(5);
    String oss = "";

    switch (osn) {
    case 0:
        oss = "Linux";
        break;
    case 1:
        oss = "OSX";
        break;
    case 2:
        oss = "Android";
        break;
    case 3:
        oss = "Unix";
        break;
    case 4:
        oss = "Windows 10";
        break;
    default:
        oss = "Linux";
        break;
    }

    return new MySubBean(oss);
}
```

I created a function that generates random OS names. This function is going to be used to create our beans and to generate random names. First I create a random generator with `Random osr =new Random();` then i generate a random integer between 0 and 4 with `int osn =osr.nextInt(5);` and choose an OS name with a switch statement.

### 7.5.3.2 The layout

The layout

```
final VerticalLayout layout = new VerticalLayout();
layout.setMargin(true);
setContent(layout);
```

A vertical layout as a main content layout.

### 7.5.3.3 First BeanContainer

First BeanContainer

```
BeanContainer beans = new BeanContainer(MyBean.class);
beans.addNestedContainerProperty("osUsed.os");
beans.setBeanIdProperty("name");
beans.addBean(new MyBean("John", "Doe", randomOS()));
beans.addBean(new MyBean("Matt", "Foe", randomOS()));
```

First I create a `BeanContainer` using `MyBean.class` as the blue print. `BeanContainer` is an in-memory container for JavaBeans and the properties of the container are determined automatically by introspecting the used JavaBean class. `beans.addNestedContainerProperty("osUsed.os");` declares the nested properties of the container. `beans.setBeanIdProperty("name");` tells the container what is the ID of the container and with `beans.addBean` we add two sample items to the container with the name, surname and a using the random os generator name.

### 7.5.3.4 Second BeanContainer

Second BeanContainer

```
BeanContainer filteredBeans = new BeanContainer(MyBean.class);
filteredBeans.addNestedContainerProperty("osUsed.os");
filteredBeans.setBeanIdProperty("name");
filteredBeans.addBean(new MyBean("John", "Doe", randomOS()));
filteredBeans.addBean(new MyBean("Matt", "Foe", randomOS()));
Filter filter = new SimpleStringFilter("surname", "Doe", true, false);
filteredBeans.addContainerFilter(filter);
```

This bean container is the same as the first container. I added a filter to show a filtered bean container. `Filter filter =new SimpleStringFilter("surname", "Doe", true, false);` adds a string filter that only shows the items with surname "doe", the case is indifferent in this particular filter. `filteredBeans.addContainerFilter(filter);` associates the filter with the container.

### 7.5.3.5 Employee Table

Unfiltered Table

```
table = new Table("Employees", beans);
table.setColumnHeader("osUsed.os", "OS Used");
table.setHeight("200px");
table.setVisibleColumns(new Object[]{"name", "surname", "osUsed.os"});
```

When I create the table "table", the bean container `beans` is passed as a parameter. With this parameter we tell to the table to associate the data on the bean as a data source for the table. `table.setColumnHeader("osUsed.os", "OS Used");` changes the name of the header that is the name of the variable to a humanized name. `table.setVisibleColumns(new Object[]{"name", "surname", "osUsed.os"});`, tells the table what columns are visible.

### 7.5.3.6 Doe's Table

#### Filtered Table

```
Table famTable = new Table("Surname Doe", filteredBeans);
famTable.setColumnHeader("osUsed.os", "OS Used");
famTable.setHeight("200px");
famTable.setVisibleColumns(new Object[]{"name", "surname", "osUsed.os"});
```

This famTable only shows items with surname "doe", all other items are filtered, thanks to the filter added to the filtered Beans bean container.

### 7.5.3.7 Dynamic input

#### Input

```
TextField tname = new TextField("Name");
TextField tsurname = new TextField("Surname");
Button badd = new Button("Add");
```

I created two text fields and a button to add items using the UI.

### 7.5.3.8 Click listener

#### Button click listener

```
badd.addClickListener(new Button.ClickListener() {

    @Override
    public void buttonClick(ClickEvent event) {

        if(!tname.isEmpty() && !tsurname.isEmpty()){
            String ctName = tname.getValue();
            String ctSurname = tsurname.getValue();
            String result = findDuplicates(ctName, ctSurname);
            if(result.equals("name")){
                Notification.show("Name is duplicate - not adding", ←
                    Notification.Type.ERROR_MESSAGE);
            }else
                if(result.equals("surname")){
                    Notification.show("Surname is duplicate");
                }
            beans.addBean(new MyBean(ctName, ctSurname, randomOS()));
            filteredBeans.addBean(new MyBean(ctName, ctSurname, randomOS()));
        }else{
            Notification.show("Missing Data ...", Notification.Type. ←
                WARNING_MESSAGE);
        }

    }

});
```

When you click the button, `if(!tname.isEmpty() && !tsurname.isEmpty())` checks if the input is empty. If not empty, we get both values and find if a duplicate exists with `String result = findDuplicates(ctName, ctSurname);`. If a duplicate exists in the name or in the surname a notification is send to the user. In any case the item is added to the bean, which itself avoids to add items with duplicate ID.



### 7.5.3.9 Find Duplicates

#### findDuplicates

```
private String findDuplicates(String pName, String pSurname){

    for (Iterator i = table.getItemIds().iterator(); i.hasNext();) {
        String iid = (String) i.next();
        Item item = table.getItem(iid);
        String currName = (String) item.getItemProperty("name").getValue();
        String currSurname = (String) item.getItemProperty("surname").getValue();

        if(pName.equals(currName)){
            return "name";
        }

        if(pSurname.equals(currSurname)){
            return "surname";
        }
    }

    return "none";
}
```

To find duplicates we need to iterate over the container, the `table.getItemIds()` method of `Container` returns a `Collection` of items so you can iterate all the entire collection of items. `String iid = (String) i.next();` gets the next item ID. `Item item = table.getItem(iid);` gets the item using the ID. `String currName = (String) item.getItemProperty("name").getValue();`, gets the value of the current item "name" property. `String currSurname = (String) item.getItemProperty("surname").getValue();` gets the current surname.

## 7.6 The complete source code

#### VaadincontainerUI.java

```
package com.example.vaadincontainer;

import java.util.Iterator;
import java.util.Random;

import javax.servlet.annotation.WebServlet;

import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.data.Container.Filter;
import com.vaadin.data.Item;
import com.vaadin.data.util.BeanContainer;
import com.vaadin.data.util.filter.SimpleStringFilter;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.Notification;
import com.vaadin.ui.Table;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@SuppressWarnings("serial")
@Theme("vaadincontainer")
```

```
public class VaadincontainerUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadincontainerUI.class)
    public static class Servlet extends VaadinServlet {
    }

    private Table table;

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);

        BeanContainer beans = new BeanContainer(MyBean.class);
        beans.addNestedContainerProperty("osUsed.os");
        beans.setBeanIdProperty("name");
        beans.addBean(new MyBean("John", "Doe", randomOS()));
        beans.addBean(new MyBean("Matt", "Foe", randomOS()));

        BeanContainer filteredBeans = new BeanContainer(MyBean.class);
        filteredBeans.addNestedContainerProperty("osUsed.os");
        filteredBeans.setBeanIdProperty("name");
        filteredBeans.addBean(new MyBean("John", "Doe", randomOS()));
        filteredBeans.addBean(new MyBean("Matt", "Foe", randomOS()));
        Filter filter = new SimpleStringFilter("surname", "Doe", true, false);
        filteredBeans.addContainerFilter(filter);

        table = new Table("Employees", beans);
        table.setColumnHeader("osUsed.os", "OS Used");
        table.setHeight("200px");
        table.setVisibleColumns(new Object[]{"name", "surname", "osUsed.os"});

        Table famTable = new Table("Surname Doe", filteredBeans);
        famTable.setColumnHeader("osUsed.os", "OS Used");
        famTable.setHeight("200px");
        famTable.setVisibleColumns(new Object[]{"name", "surname", "osUsed.os"});

        TextField tname = new TextField("Name");
        TextField tsurname = new TextField("Surname");
        Button badd = new Button("Add");
        badd.addClickListener(new Button.ClickListener() {

            @Override
            public void buttonClick(ClickEvent event) {

                if(!tname.isEmpty() && !tsurname.isEmpty()){
                    String ctName = tname.getValue();
                    String ctSurname = tsurname.getValue();
                    String result = findDuplicates(ctName, ctSurname);
                    if(result.equals("name")){
                        Notification.show("Name is duplicate - not ←
adding", Notification.Type.ERROR_MESSAGE ←
);
                    }else
                        if(result.equals("surname")){
                            Notification.show("Surname is duplicate");
                        }
                    beans.addBean(new MyBean(ctName, ctSurname, ←
randomOS()));
                    filteredBeans.addBean(new MyBean(ctName, ctSurname, ←
```

```
        randomOS());
    }else{
        Notification.show("Missing Data ...", Notification. ←
            Type.WARNING_MESSAGE);
    }
}

});

HorizontalLayout hl = new HorizontalLayout();
hl.addComponent(tname);
hl.addComponent(tsurname);

layout.addComponent(hl);
layout.addComponent(badd);
HorizontalLayout hlTab = new HorizontalLayout();
hlTab.addComponent(table);
hlTab.addComponent(famTable);
layout.addComponent(hlTab);
}

private String findDuplicates(String pName, String pSurname){

    for (Iterator i = table.getItemIds().iterator(); i.hasNext();) {
        String iid = (String) i.next();
        Item item = table.getItem(iid);
        String currName = (String) item.getItemProperty("name").getValue();
        String currSurname = (String) item.getItemProperty("surname"). ←
            getValue();

        if(pName.equals(currName)){
            return "name";
        }

        if(pSurname.equals(currSurname)){
            return "surname";
        }
    }

    return "none";
}

private MySubBean randomOS(){
    Random osr = new Random();
    int osn = osr.nextInt(5);
    String oss = "";

    switch (osn) {
    case 0:
        oss = "Linux";
        break;
    case 1:
        oss = "OSX";
        break;
    case 2:
        oss = "Android";
        break;
    case 3:
        oss = "Unix";
        break;
    case 4:
        oss = "Windows 10";
```

```
                break;
            default:
                oss = "Linux";
                break;
        }

        return new MySubBean(oss);
    }
}
```

### MyBean.java

```
package com.example.vaadincontainer;

import java.io.Serializable;

public class MyBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String name;
    private String surname;
    private MySubBean osUsed;

    public MyBean (String pname, String psurname, MySubBean pSubBean){

        this.name = pname;
        this.surname = psurname;
        this.osUsed = pSubBean;

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public MySubBean getOsUsed() {
        return osUsed;
    }

    public void setOsUsed(MySubBean osUsed) {
        this.osUsed = osUsed;
    }

}
```

### MySubBean.java

```
package com.example.vaadincontainer;

import java.io.Serializable;

public class MySubBean implements Serializable {

    private static final long serialVersionUID = 1L;

    private String os;

    public MySubBean(String pOs){
        this.os = pOs;
    }

    public String getOs() {
        return os;
    }

    public void setOs(String os) {
        this.os = os;
    }

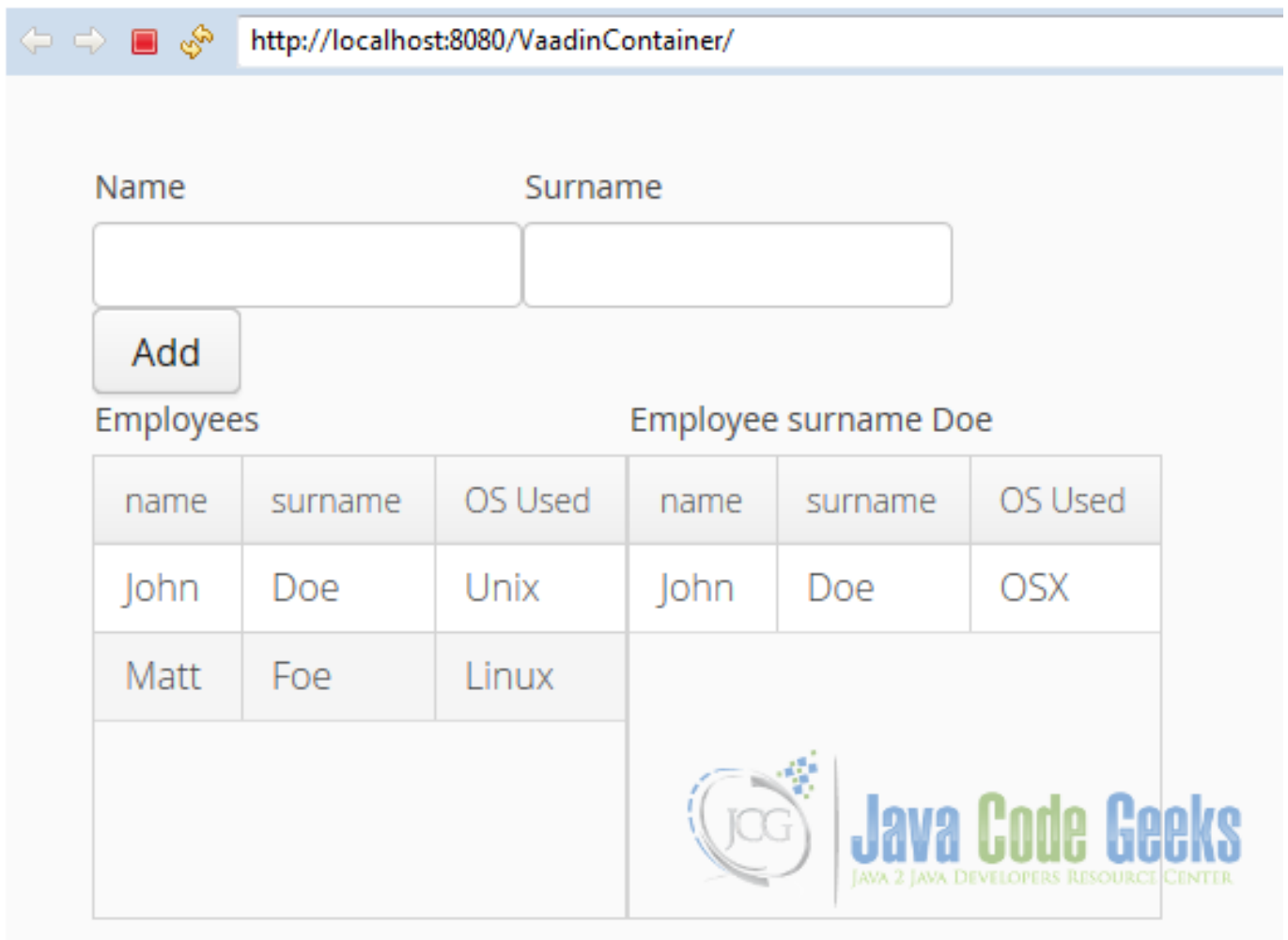
}
```

## 7.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 7.8 Results

When you first run the example you only get the hardcoded sample data:



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/VaadinContainer/`. The application interface includes two input fields labeled "Name" and "Surname", an "Add" button, and a table of employee data. The table has two sections: "Employees" and "Employee surname Doe". The "Employees" section contains three rows of data, and the "Employee surname Doe" section contains one row of data. A watermark for "Java Code Geeks" is visible in the bottom right corner of the table area.

name	surname	OS Used	name	surname	OS Used
John	Doe	Unix	John	Doe	OSX
Matt	Foe	Linux			

Figure 7.3: Sample Data

Trying to add empty data:

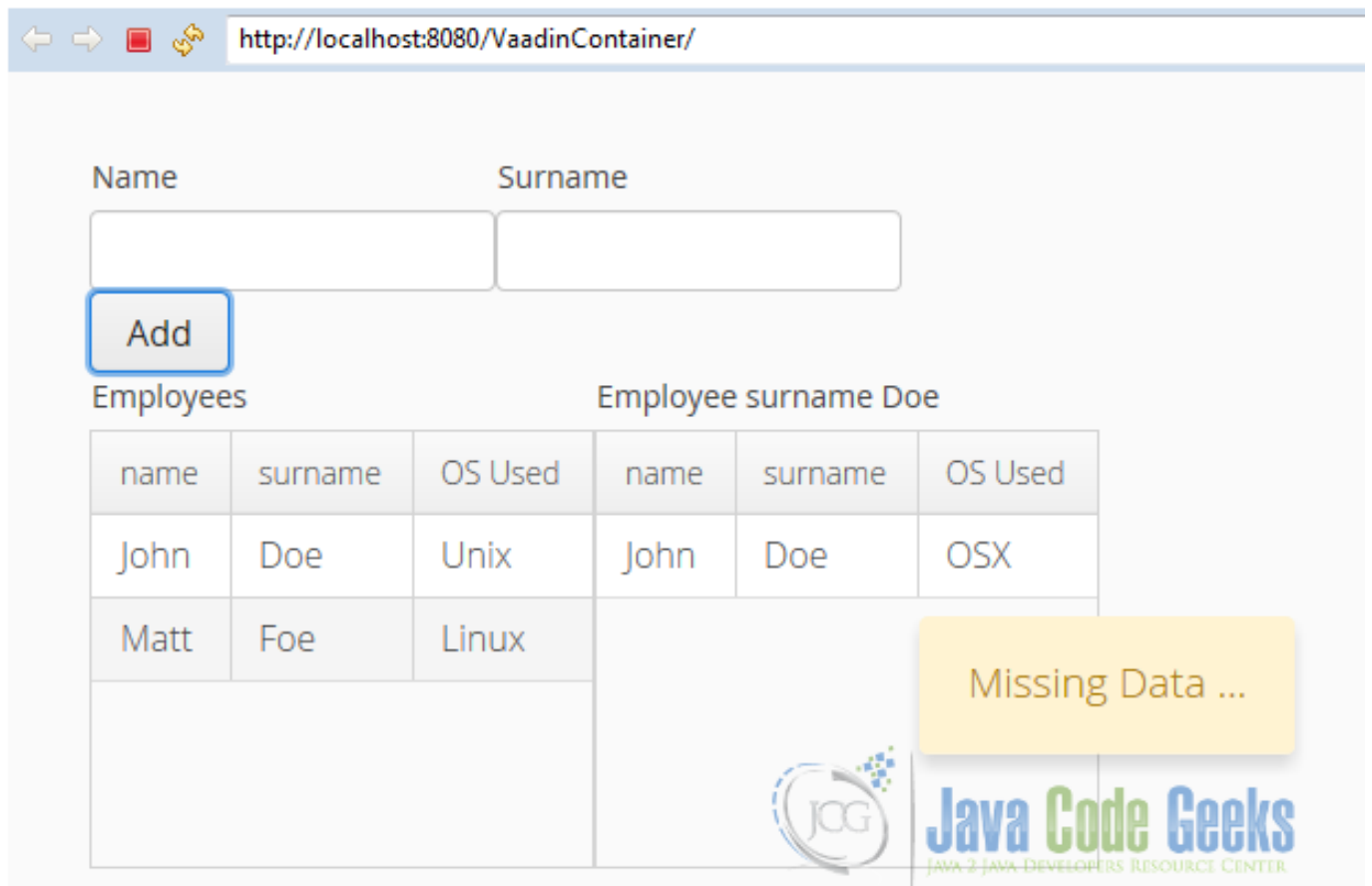


Figure 7.4: No Data

When you add a duplicate surname a Notification is showed:

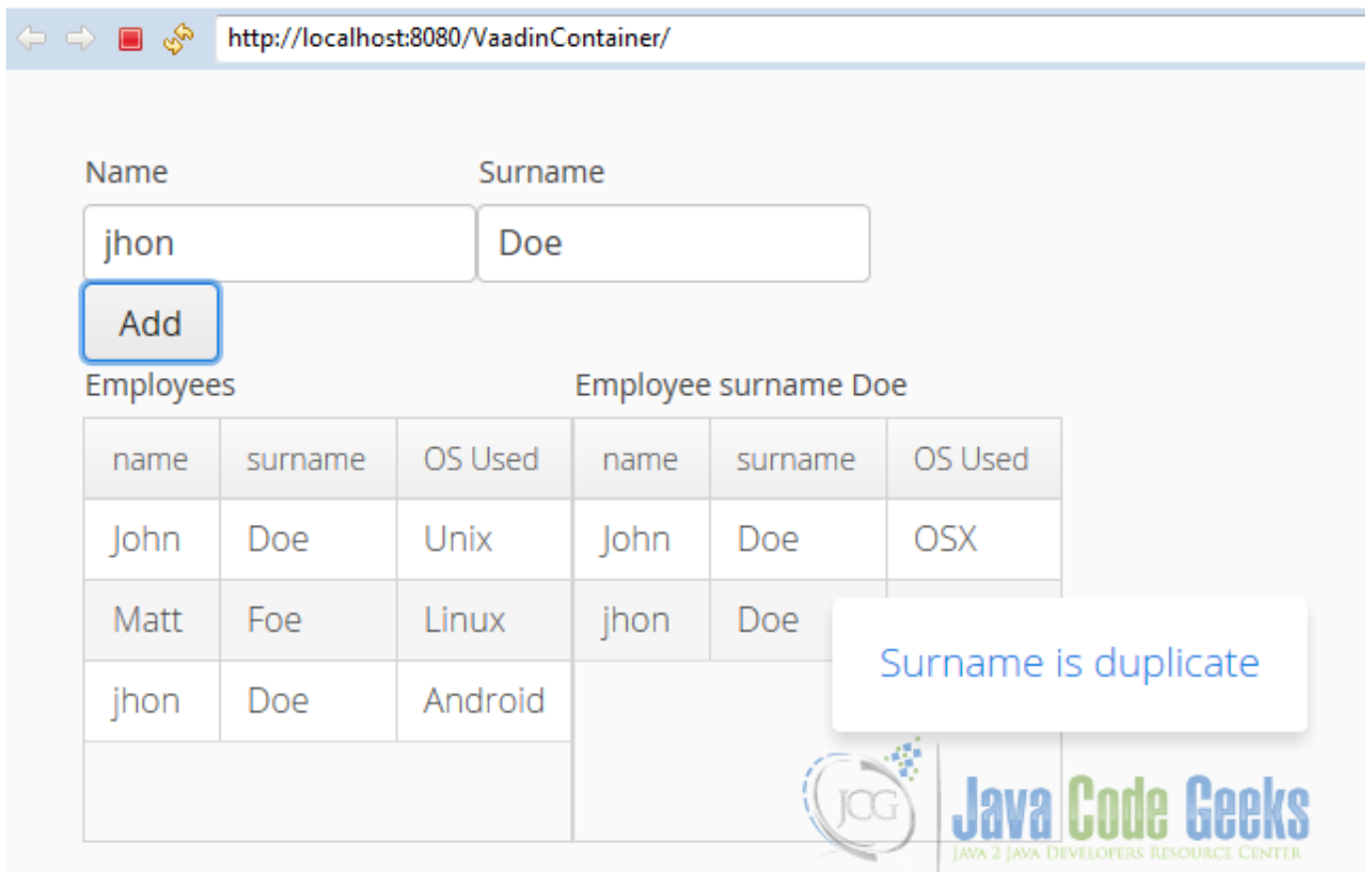


Figure 7.5: Duplicate Surname

When you try to add a duplicate name, it's not added because name is the primary ID and the container ensures unicity of the primary ID:



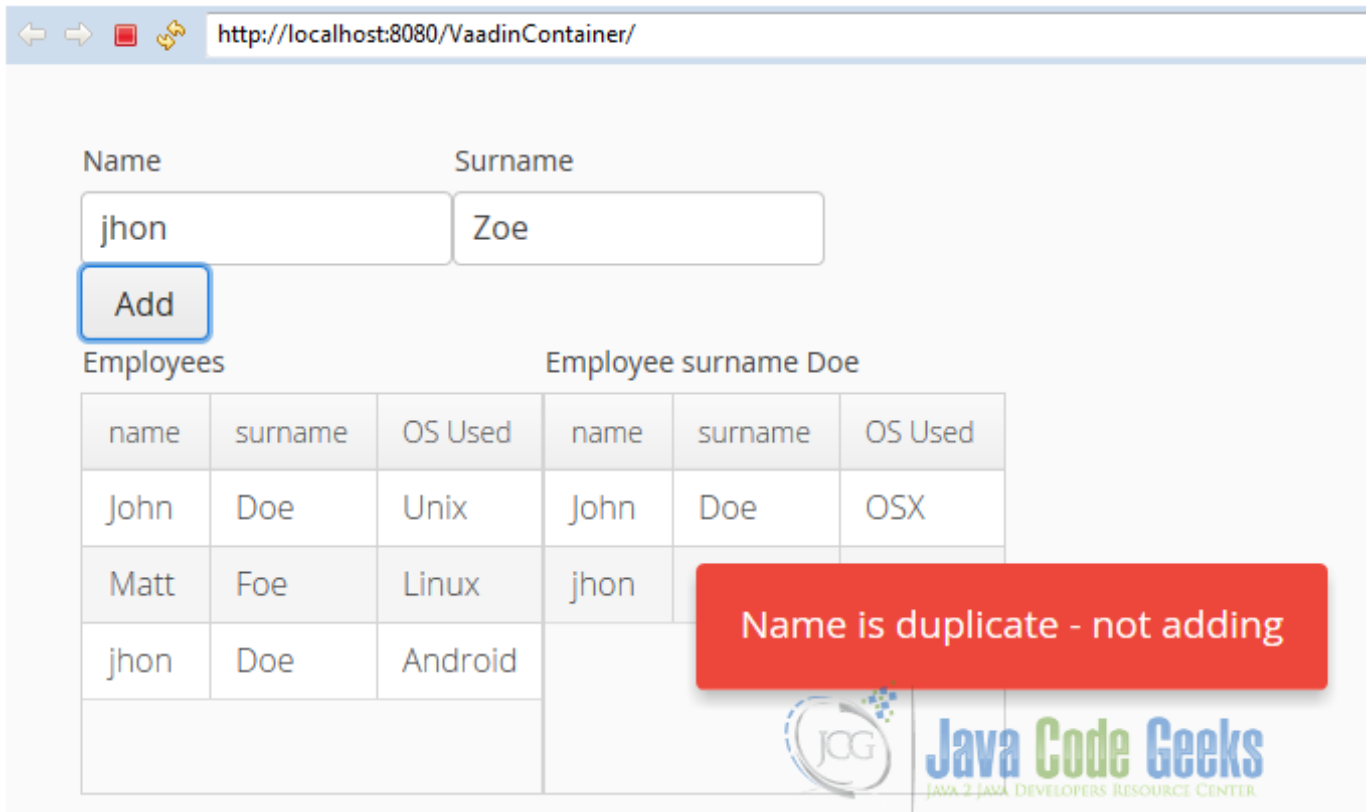


Figure 7.6: Duplicate Name

## 7.9 Download the Source Code

This was an example of: Vaadin Container.

### Download

You can download the Eclipse project here: [VaadinContainer](#)

## Chapter 8

# Vaadin Custom Component Example

Suppose you need a reusable component in your web application, a component widget that you need to add in multiple places into your application, instead of coding the component functionality every time, Vaadin offers in handy the possibility of creating a reusable widget that you can use every time you need it, saving your time and giving you a more elegant solution than coding the whole thing every time.

### 8.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.2

### 8.2 Introduction

In this example we are going to create a Vaadin widget that you can adapt to your needs. With the Vaadin plugin you have an automated way to make the widget and add it to your project ready to customize and use. Let's get started!

### 8.3 Prerequisites

- JDK installed
- Eclipse Mars installed and working
- Vaadin Plugin 7.6.2

### 8.4 Create the project

Fire up Eclipse and go to the menu File → New → Other:

---

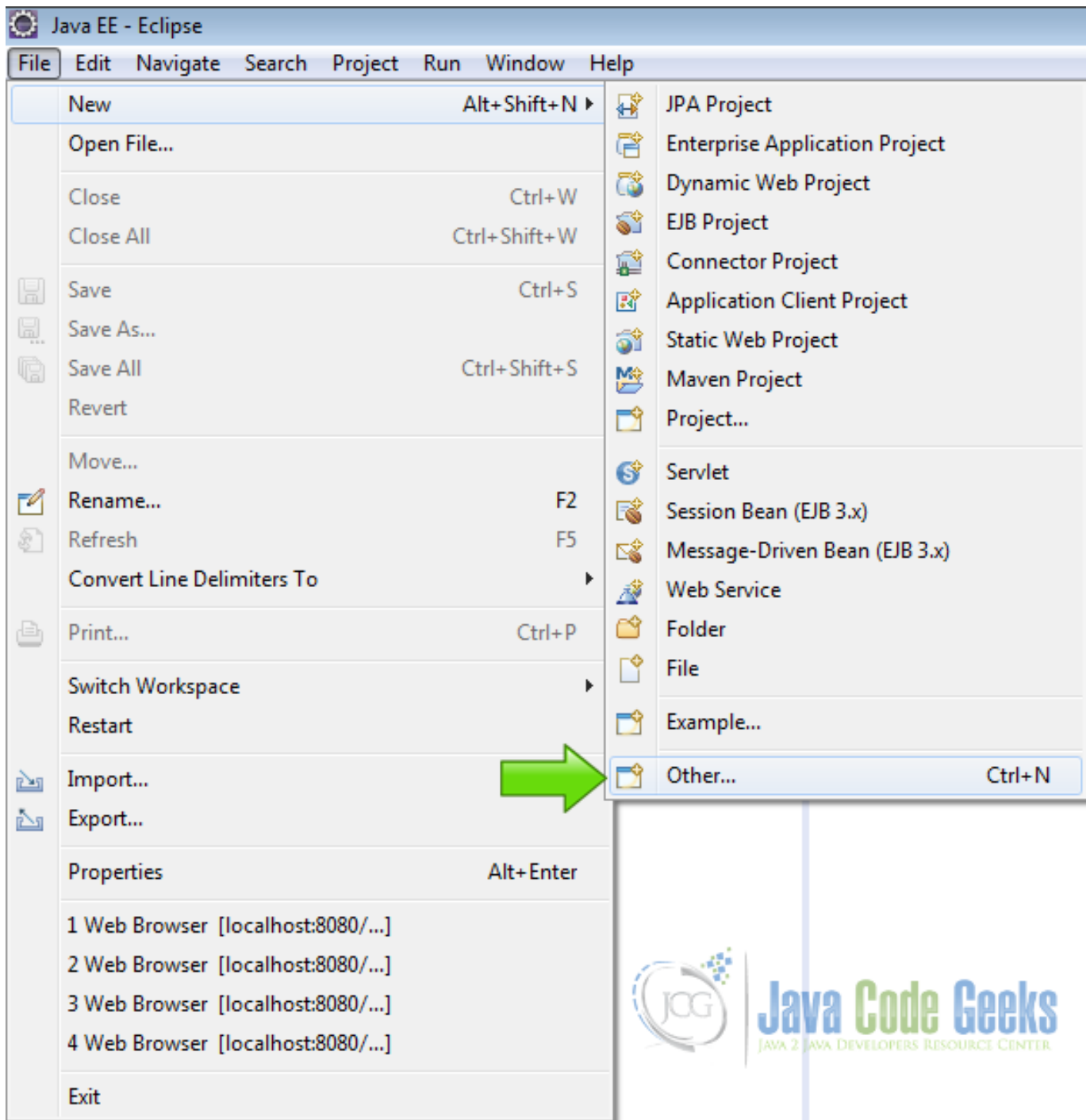


Figure 8.1: Create Project

Choose from the list Vaadin 7 Project and hit next.

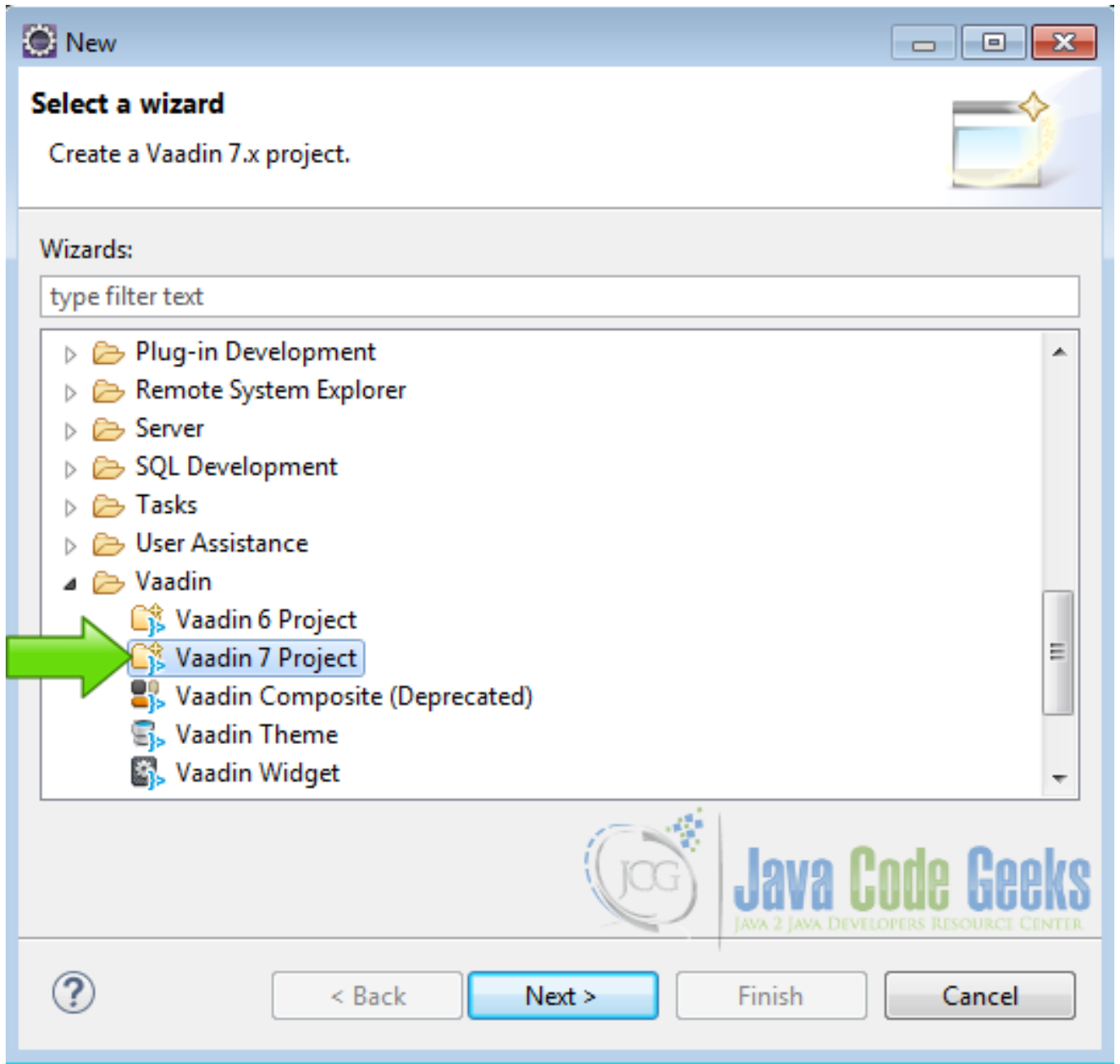


Figure 8.2: Vaadin project

Name your project and hit finish:

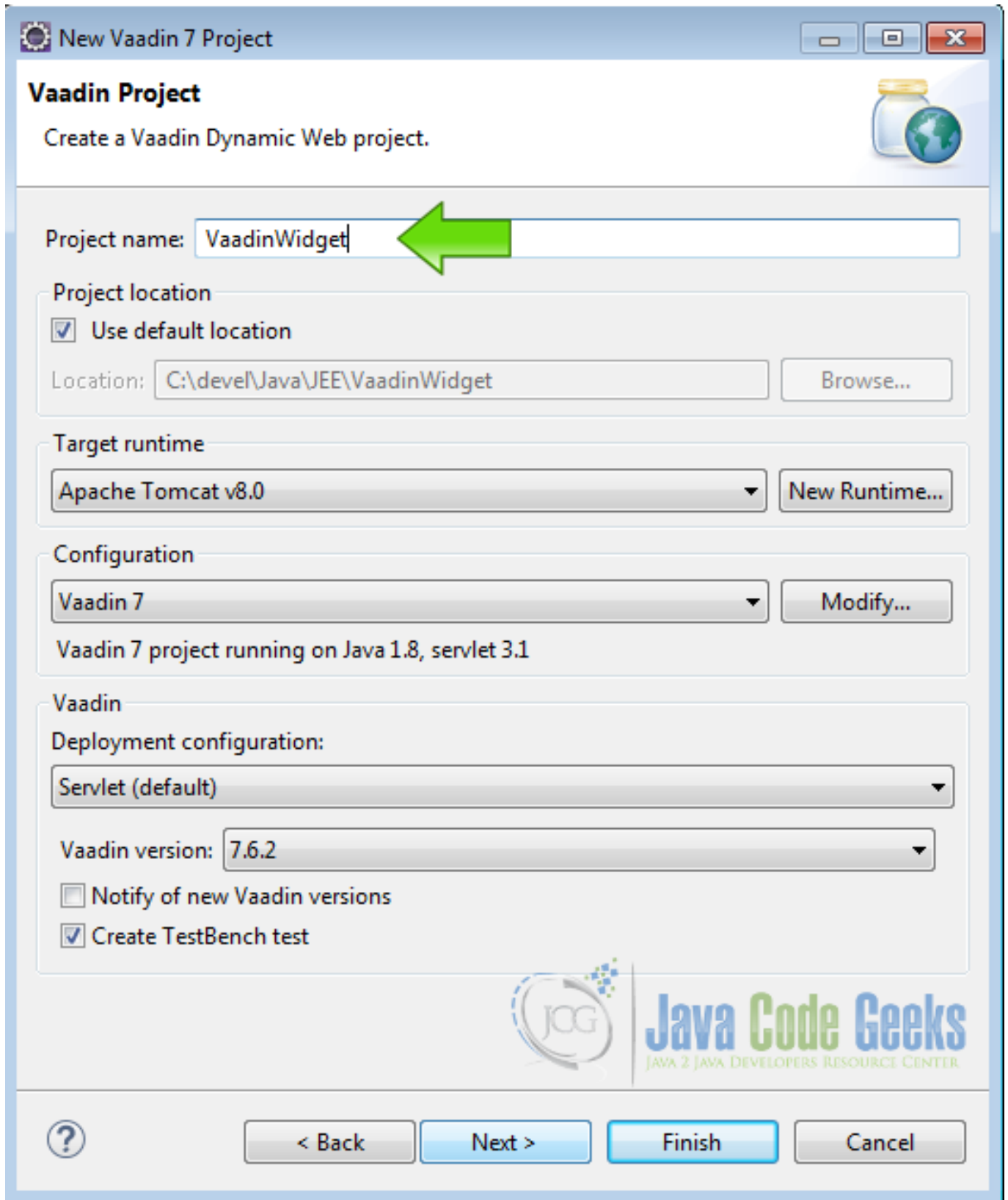


Figure 8.3: Name the project

Now you have a fresh Vaadin project ready to run. Let's add the custom component to the project.

## 8.5 Create the Vaadin custom component

Right click in the project folder inside eclipse and choose New → Other:

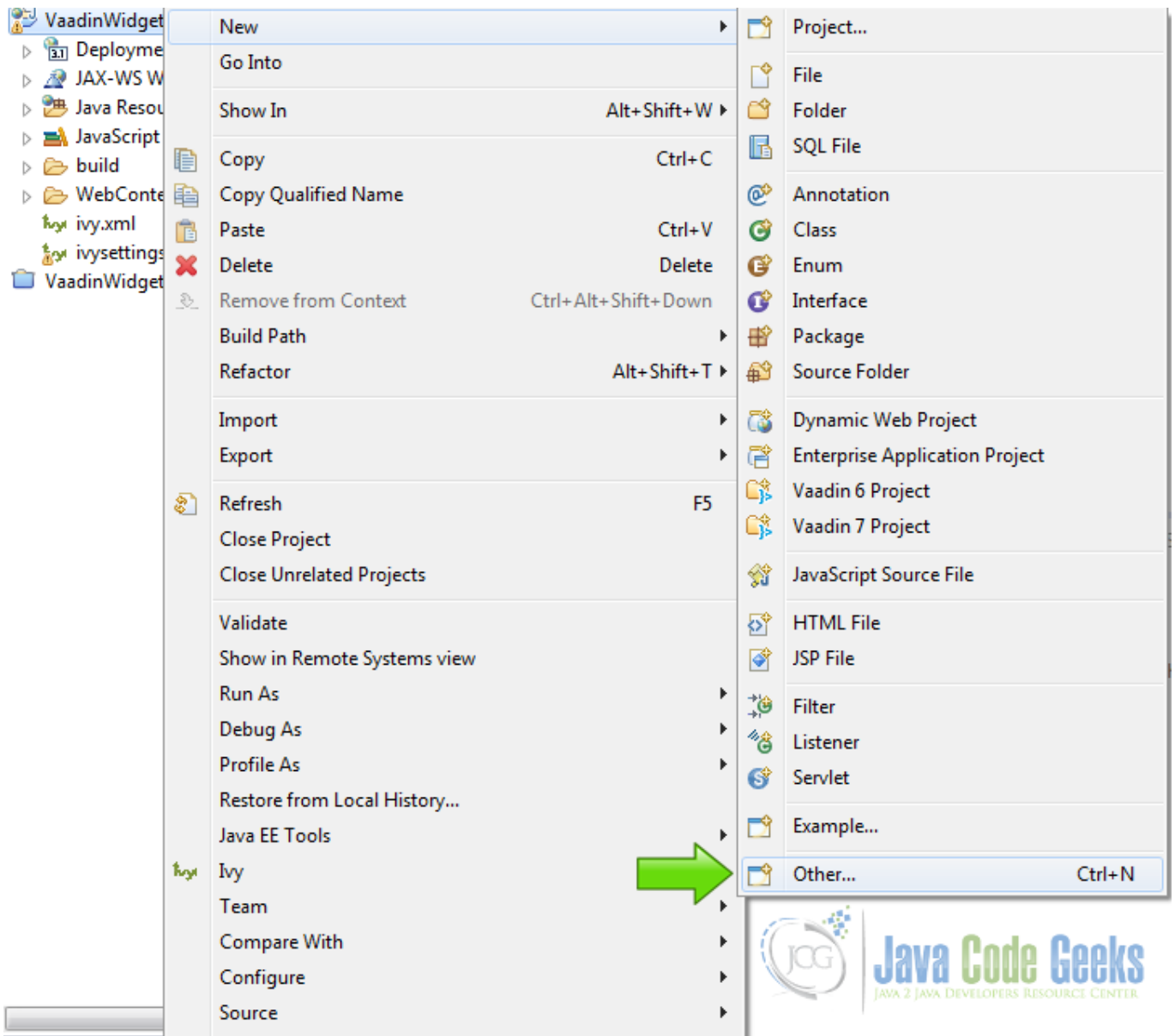


Figure 8.4: Add widget

In the following pop-up window, choose from the list Vaadin Widget and hit next.

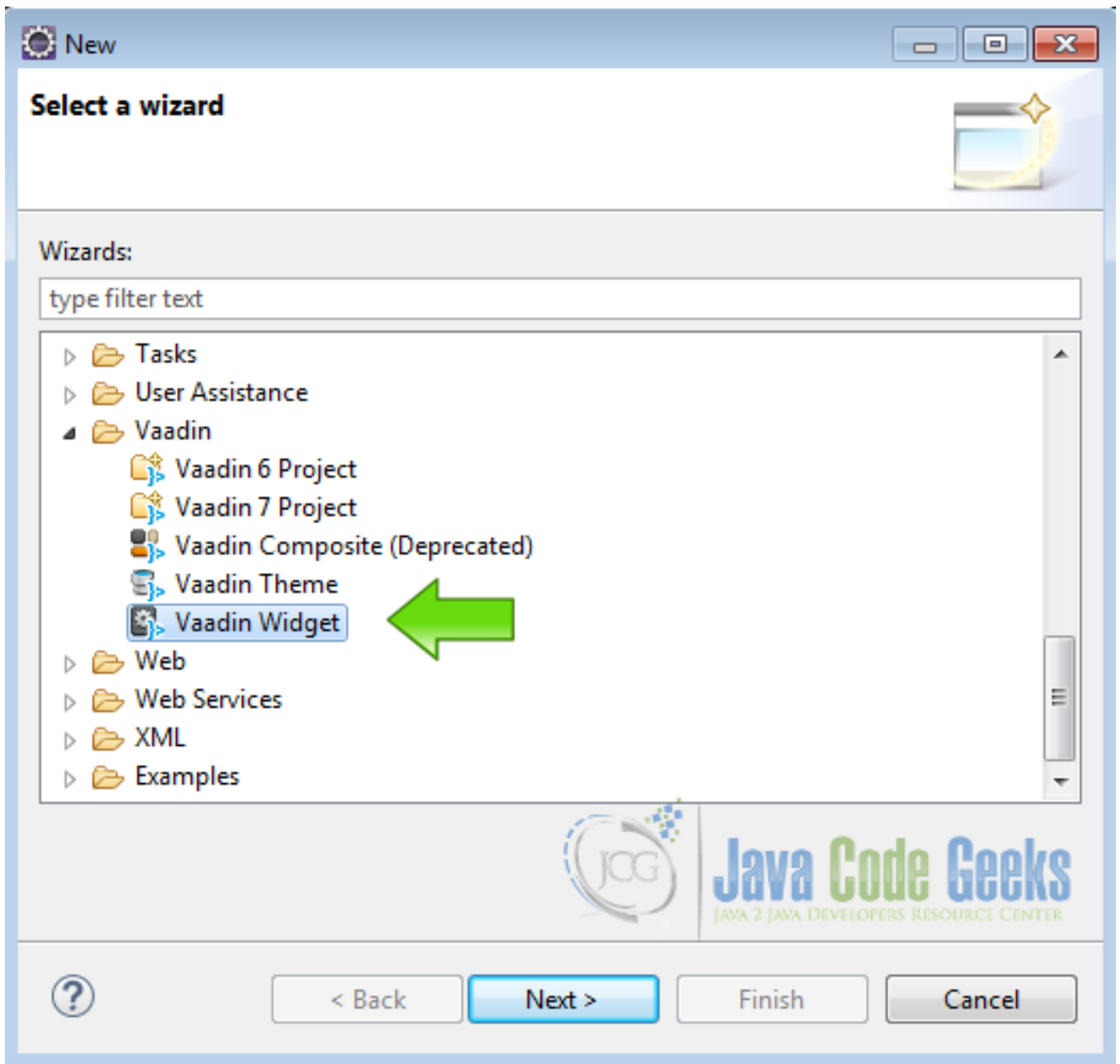


Figure 8.5: Create widget

Now pick a name for your component and hit finish.

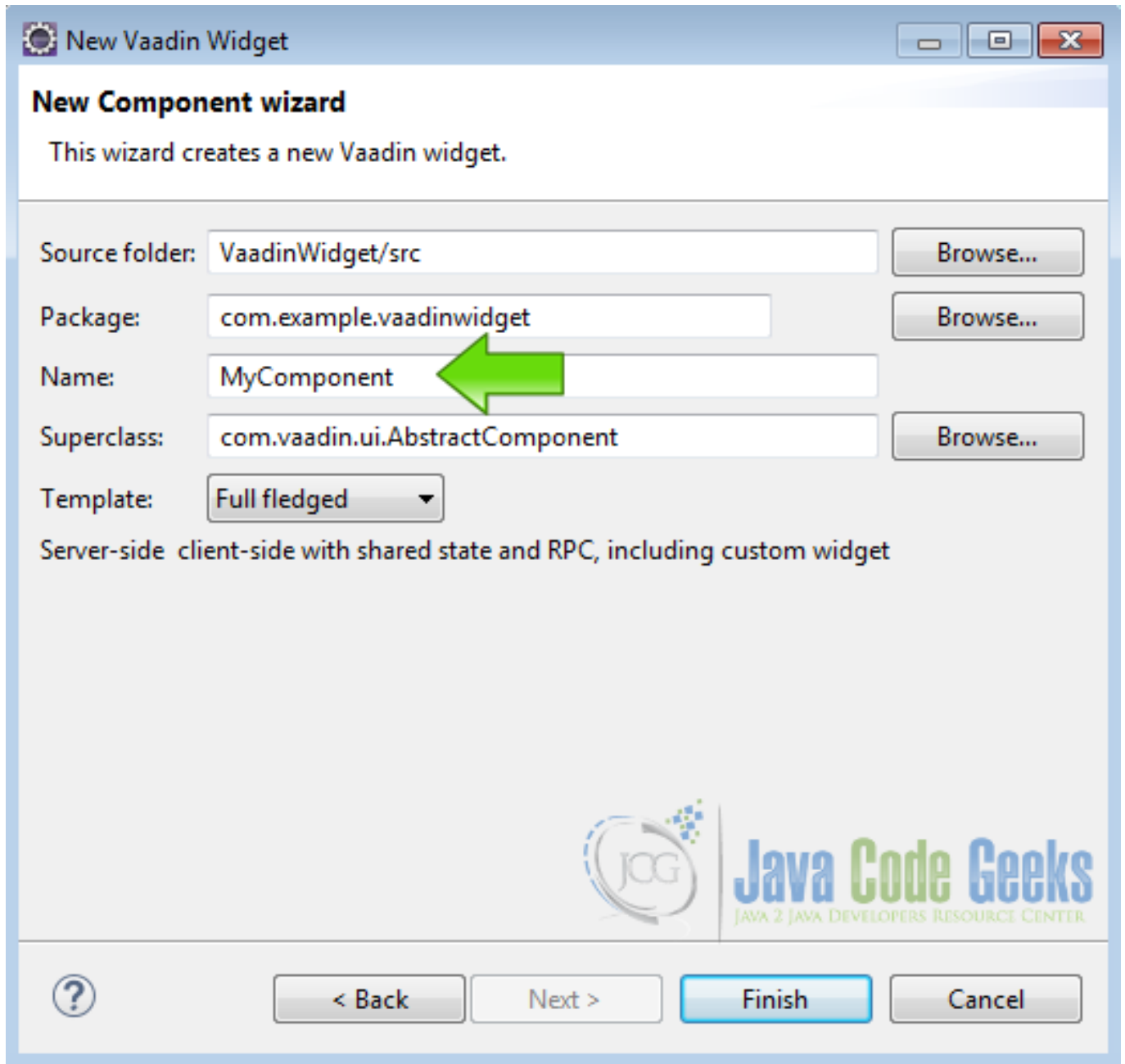


Figure 8.6: Name the widget

You are done creating the custom widget.

## 8.6 Review your project

Take a look at your project tree and note that Vaadin have created a bunch of files for you saving you to type a lot of boilerplate code, also note that Vaadin organize your project in packages with your main application in one package `com.example.vaadinwidget` and the custom component in other package `com.example.vaadinwidget.client.mycomponent`, abstracting the inner details of the widget.



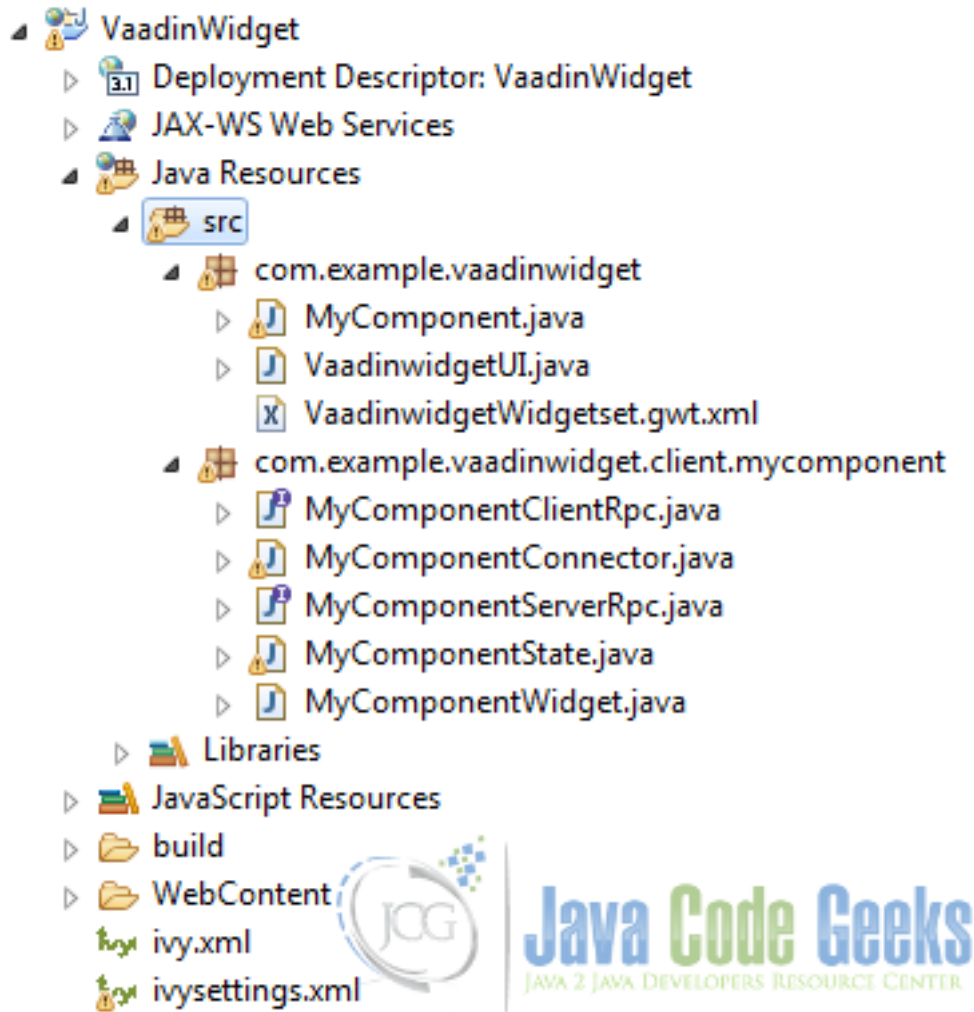


Figure 8.7: project tree

All the code of the widget are in the project for you to customize if your widget is complex enough or if you need to.

Vaadin have created a client side text widget when you click on it, it changes the text of the widget and if you click 5 times, make the application fire an alert with a message.

Look at the code of the file that manage the functionality of the component, it extends `com.vaadin.ui.AbstractComponent`

`MyComponent.java`

```
package com.example.vaadinwidget;

import com.example.vaadinwidget.client.mycomponent.MyComponentClientRpc;
import com.example.vaadinwidget.client.mycomponent.MyComponentServerRpc;
import com.vaadin.shared.MouseEventDetails;
import com.example.vaadinwidget.client.mycomponent.MyComponentState;

public class MyComponent extends com.vaadin.ui.AbstractComponent {

    private MyComponentServerRpc rpc = new MyComponentServerRpc() {
        private int clickCount = 0;

        public void clicked(MouseEventDetails mouseDetails) {
            getRpcProxy(MyComponentClientRpc.class).alert(
```

```
        "Ok, that's enough!");
        // update shared state
        getState().text = "You have clicked " + clickCount + " times";
    }
};

public MyComponent() {
    registerRpc(rpc);
}

@Override
public MyComponentState getState() {
    return (MyComponentState) super.getState();
}
}
```

we have `clickCount` that counts the mouse clicks and also we have a function `clicked` that capture the mouse click and when `clickCount` is 5 it fire an alert also modifies the widget text with `getState().text = "You have clicked " + clickCount + " times";`

## 8.7 Compile the widgetset

From the Vaadin button in the toolbar choose compile widgetset.

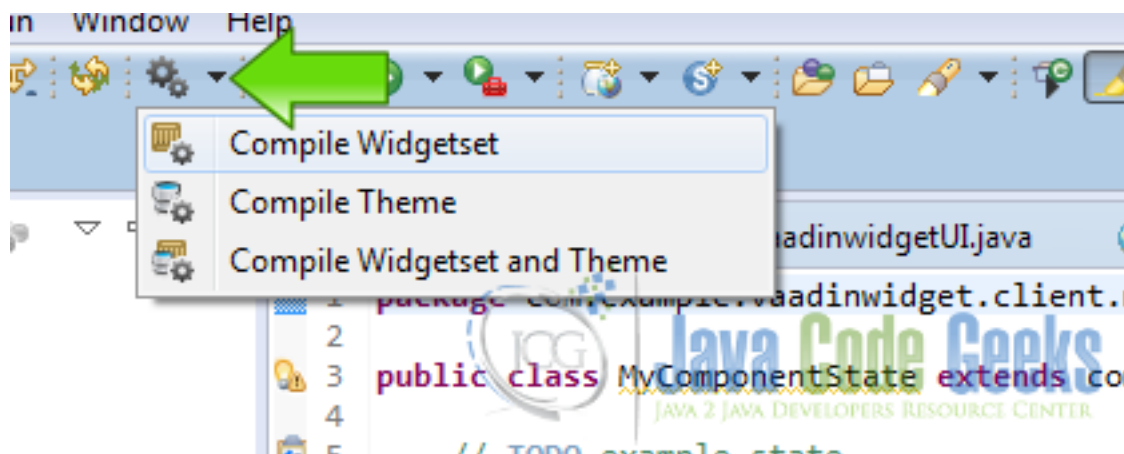


Figure 8.8: compile widgetset

and after a while you widgetset is compiled using the GWT compiler shipped with the Vaadin plugin.

## 8.8 Run The project

Right click in the project folder and choose Run As → Run on server.

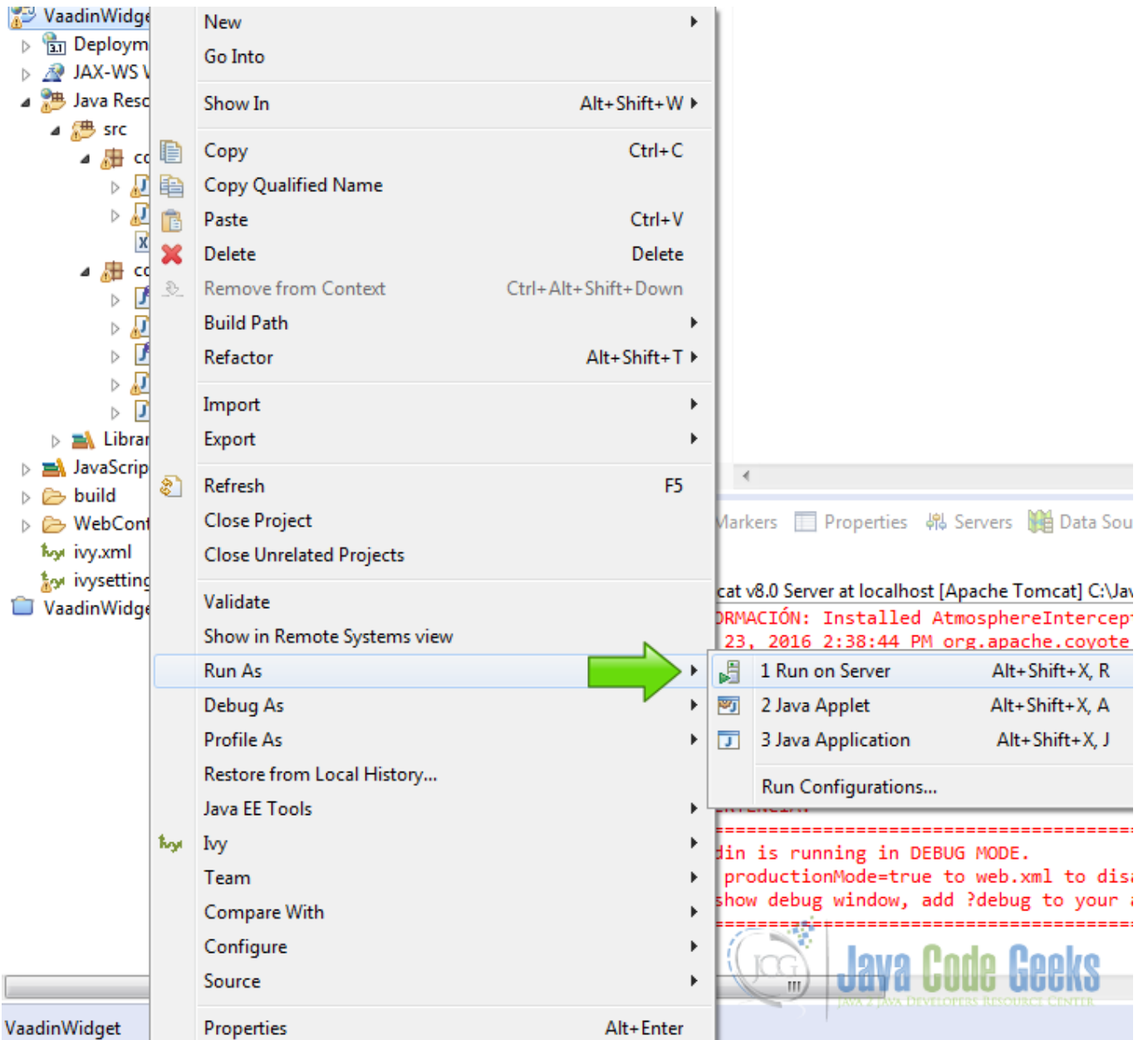


Figure 8.9: Run Project

Then choose your favorite server and hit finish.

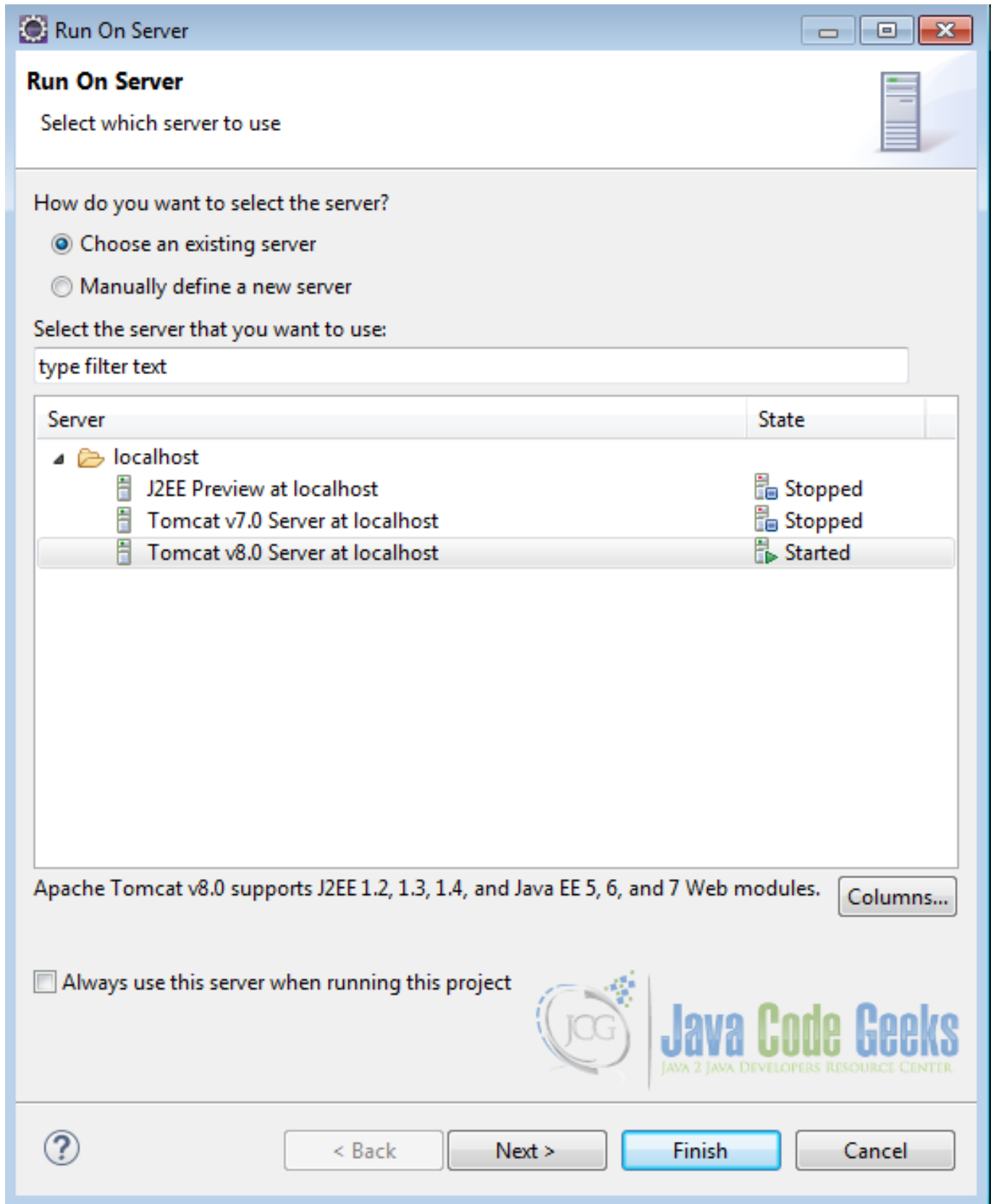


Figure 8.10: Choose server

## 8.9 Application output

You should have your custom component running in your application:

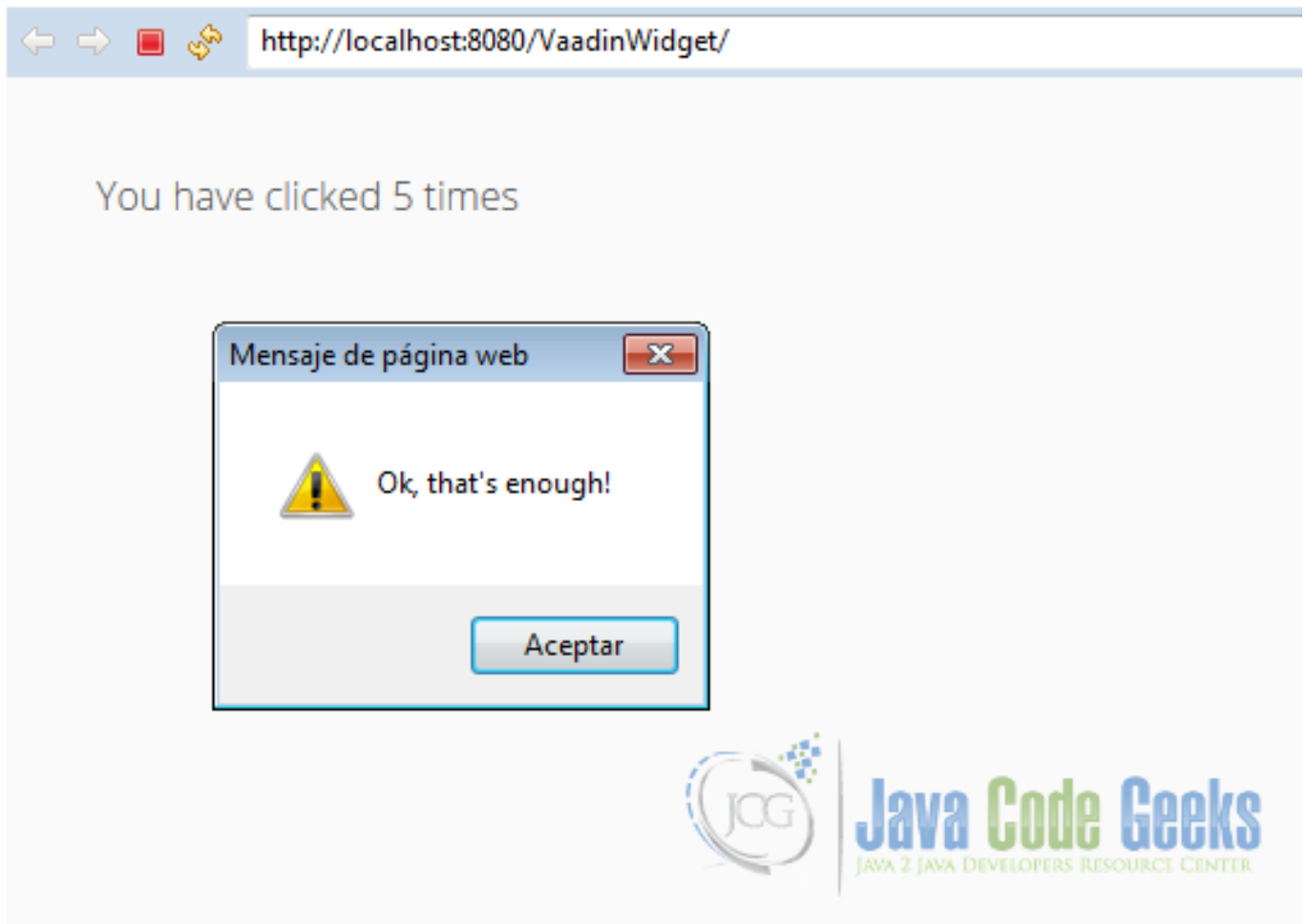


Figure 8.11: Application output

## 8.10 Get the source code

This was an example about Vaadin Custom Component.

### Download

You can download the Eclipse project here: [VaadinWidget](#)

## Chapter 9

# Vaadin Server Push Example

Server push is a technology when the server pushes data to the client without the client asking for that data just like the old intranet client/server architecture when the clients get updated by the server. This kind of communication was near to impossible in the web few years before, first the bandwidth, when the Internet begins was not enough for server push, several attempts before the HTML5 websocket like webcasting and comet have not been successful enough and is with HTML5 that server push become widely used.

### 9.1 The tools

- Java JDK 8
- Latest Eclipse Mars
- Vaadin 7.6.4
- Tomcat Server 8

### 9.2 Introduction

Vaadin make very easy to use server push, you only need to use the `@Push` annotation and Vaadin is using server push. In this example I am going to show you how to handle the server push to send content to the client. I am using here Labels as a containers to the data pushed from the server but you can use any container you want. In the first server push, the server is pushing the time every second and in the second one the server is pushing every 10 seconds.

### 9.3 Prerequisites

- JDK installed
  - Eclipse Mars installed and working
  - Vaadin 7.6.4 plugin installed
  - Tomcat 8 installed and running
-

## 9.4 Set up the project

In the file menu choose File → New → Other:

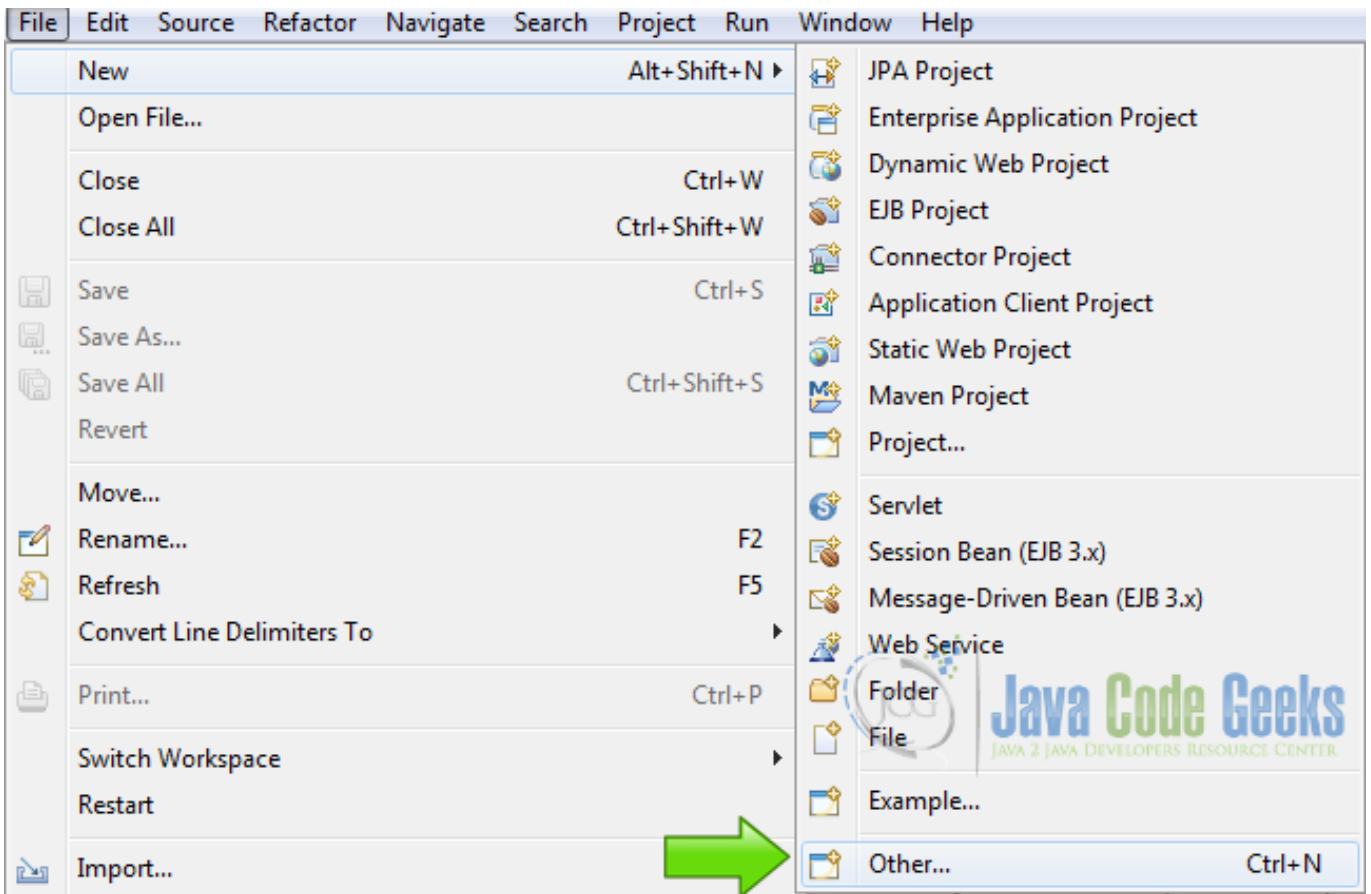


Figure 9.1: New Project

Now from the list choose Vaadin 7 project:

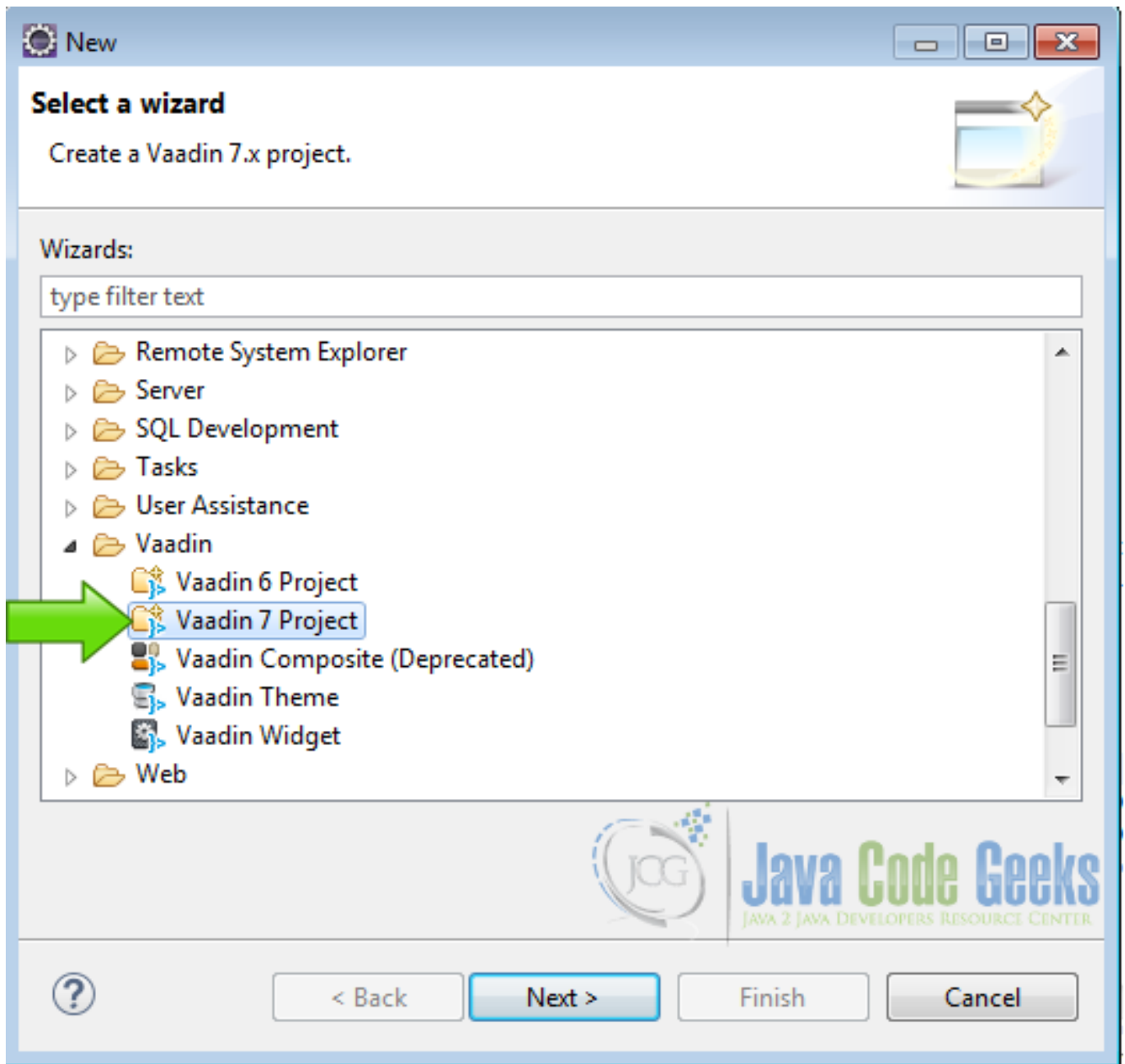


Figure 9.2: Vaadin Project

Hit next and name your project then hit finish.

## 9.5 Coding the example

### 9.5.1 The quote generator class

I created a class that generates some random quotes from an array.

Quote array

```
private String[] quotes = new String[20];
```



Declared the array.

Generate quotes

```
public String getQuote() {
    Random r = new Random();
    return quotes[r.nextInt(20)];
}
```

Generate a random quote from the array.

## 9.5.2 Initial preparations

@Push annotation

```
@Push
@SuppressWarnings("serial")
@Theme("vaadinseverpushbeta")
public class VaadinseverpushbetaUI extends UI{
```

To use server push just add the annotation @Push to the UI class annotations. Adding @Push to a UI class configures the UI for automatic push.

Class Variables

```
private QuoteGenerator qg;
private VerticalLayout layout;
private Label theTime;
```

The private QuoteGenerator qg; is used for the custom POJO Class created before to generate quotes. private VerticalLayout layout; the layout of our UI class and private Label theTime; to store the datetime pushed from the server.

## 9.5.3 First Thread

I am using Threads to make the server pushes.

First Thread

```
class MyFirstThread extends Thread {

    @Override
    public void run() {
        try {
            while (true) {
                Thread.sleep(1000);

                access(new Runnable() {
                    @Override
                    public void run() {
                        theTime.setValue("Its now : " + Instant.now ←
                            ());
                    }
                });
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

This thread update the private Label theTime; every second as long as the webpage is running, the time is coming from the server using server push. Thread.sleep(1000);, sleep the thread for 1000 milliseconds = 1 second. access(new Runnable()) locks the UI and provide exclusive access to the current runnable, All the UI operations must be inside an access block because the thread needs a lock to modify the UI, if you try to modify the UI outside an access block an exception is raised. theTime.setValue("Its now : " + Instant.now()); modify the content of the Label using server push.

## 9.5.4 Second Thread

### Second Thread

```
class MySecondThread2 extends Thread {
    int count = 0;

    @Override
    public void run() {
        try {
            while (count < 4) {
                Thread.sleep(10000);

                access(new Runnable() {
                    @Override
                    public void run() {
                        layout.addComponent(new Label(qg.getQuote() ←
                        ));
                        count++;
                    }
                });

            }

            access(new Runnable() {
                @Override
                public void run() {
                    layout.addComponent(new Label("&quot;No more ←
                    messages for you !&quot;));
                }
            });
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

This thread run four times every 10 seconds and every time it runs, adds a label with a random quote from the server using server push.int count =0; its a counter to limit the number of times this thread runs.

while (count < 4) {}, the while checks the counter.

Thread.sleep(10000); the thread sleep for 10 seconds, remember the parameter of the sleep is in milliseconds

access(new Runnable() { using the access block to lock the UI

layout.addComponent(new Label(qg.getQuote())); adds a new Label with a random quote every time it runs

count++; Update the counter. After the thread have added 4 labels it exits from the while and add a last label to show to the user that it finish

layout.addComponent(new Label("No more messages for you !")); with no more messages.

## 9.5.5 The init method

### Init Method

```

@Override
protected void init(VaadinRequest request) {
    qq = new QuoteGenerator();
    layout = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);
    theTime = new Label();
    theTime.setValue("Its now : " + Instant.now());
    layout.addComponent(theTime);
    new MyFirstThread().start();
    new MySecondThread2().start();
}

```

In the `init` method first create the random quote generator instance `qq =new QuoteGenerator()`; . Then create the layout `layout =new VerticalLayout()`; . Create the label to hold the time `theTime =new Label()`; . Start the first thread `new MyFirstThread().start()`; and start the second thread `new MySecondThread2().start()`; .

## 9.6 The complete source code

QuoteGenerator.java

```

package com.example.vaadinserverpush;

import java.util.Random;

public class QuoteGenerator {
    private String[] quotes = new String[20];

    public QuoteGenerator () {
        quotes[0] = "A friend asks only for your time not your money.";
        quotes[1] = "Your high-minded principles spell success.";
        quotes[2] = "Enjoy the good luck a companion brings you.";
        quotes[3] = "Hidden in a valley beside an open stream- This will be the ←
            type of place where you will find your dream.";
        quotes[4] = "What ever you're goal is in life, embrace it visualize it, and ←
            for it will be yours.";
        quotes[5] = "You will become great if you believe in yourself.";
        quotes[6] = "Never give up. You're not a failure if you don't give up.";
        quotes[7] = "It is now, and in this world, that we must live.";
        quotes[8] = "Adversity is the parent of virtue.";
        quotes[9] = "A stranger, is a friend you have not spoken to yet.";
        quotes[10] = "A new voyage will fill your life with untold memories.";
        quotes[11] = "Its amazing how much good you can do if you dont care who ←
            gets the credit.";
        quotes[12] = "Stop wishing. Start doing.";
        quotes[13] = "Your fortune is as sweet as a cookie.";
        quotes[14] = "Don't pursue happiness - create it.";
        quotes[15] = "Everything happens for a reason.";
        quotes[16] = "Rivers need springs.";
        quotes[17] = "All progress occurs because people dare to be different.";
        quotes[18] = "It is not necessary to show others you have change; the ←
            change will be obvious.";
        quotes[19] = "Next full moon brings an enchanting evening.";
    }

    public String getQuote() {
        Random r = new Random();
        return quotes[r.nextInt(20)];
    }
}

```

```
}
```

### VaadinserverpushUI.java

```
package com.example.vaadinserverpush;

import java.time.Instant;

import javax.servlet.annotation.WebServlet;

import com.vaadin.annotations.Push;
import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Label;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;

@Push
@SuppressWarnings("serial")
@Theme("vaadinserverpush")
public class VaadinserverpushUI extends UI {

    @WebServlet(value = "/*", asyncSupported = true)
    @VaadinServletConfiguration(productionMode = false, ui = VaadinserverpushUI.class)
    public static class Servlet extends VaadinServlet {
    }

    private QuoteGenerator qg;
    private VerticalLayout layout;
    private Label theTime;

    @Override
    protected void init(VaadinRequest request) {
        qg = new QuoteGenerator();
        layout = new VerticalLayout();
        layout.setMargin(true);
        setContent(layout);
        theTime = new Label();
        theTime.setValue("Its now : " + Instant.now());
        layout.addComponent(theTime);
        new MyFirstThread().start();
        new MySecondThread2().start();
    }

    class MyFirstThread extends Thread {

        @Override
        public void run() {
            try {
                while (true) {
                    Thread.sleep(1000);

                    access(new Runnable() {
                        @Override
                        public void run() {
                            theTime.setValue("Its now : " + Instant.now());
                        }
                    });
                }
            }
        }
    }
}
```

```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

class MySecondThread2 extends Thread {
    int count = 0;

    @Override
    public void run() {
        try {
            while (count < 4) {
                Thread.sleep(10000);

                access(new Runnable() {
                    @Override
                    public void run() {
                        layout.addComponent(new Label(qg.getQuote()));
                        count++;
                    }
                });
            }

            access(new Runnable() {
                @Override
                public void run() {
                    layout.addComponent(new Label("&quot;No more messages for you !& <-
                    quot;));
                }
            });
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
}
```

## 9.7 Running the example

Right click on the project folder and choose Run as → Run on server choose Tomcat 8 server and hit finish.

## 9.8 Results

Open your application in a browser and fire Up the developer tools CTRL+SHIFT+i shortcut in most browsers, you should see something like the following image, go to the console tab:

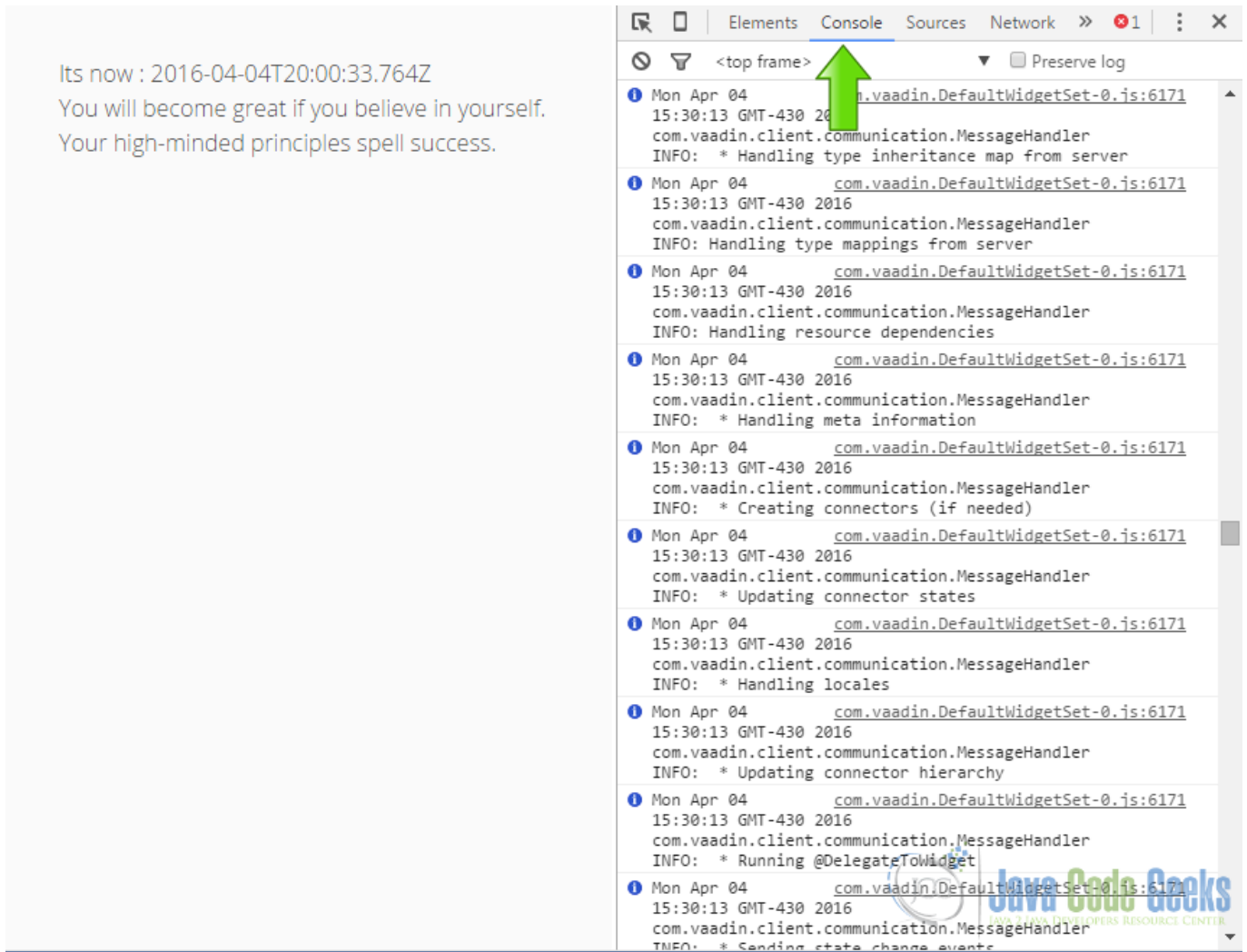


Figure 9.3: Browser developer tools

Let's run the application and examine the console output to see what Vaadin is doing under the hood.

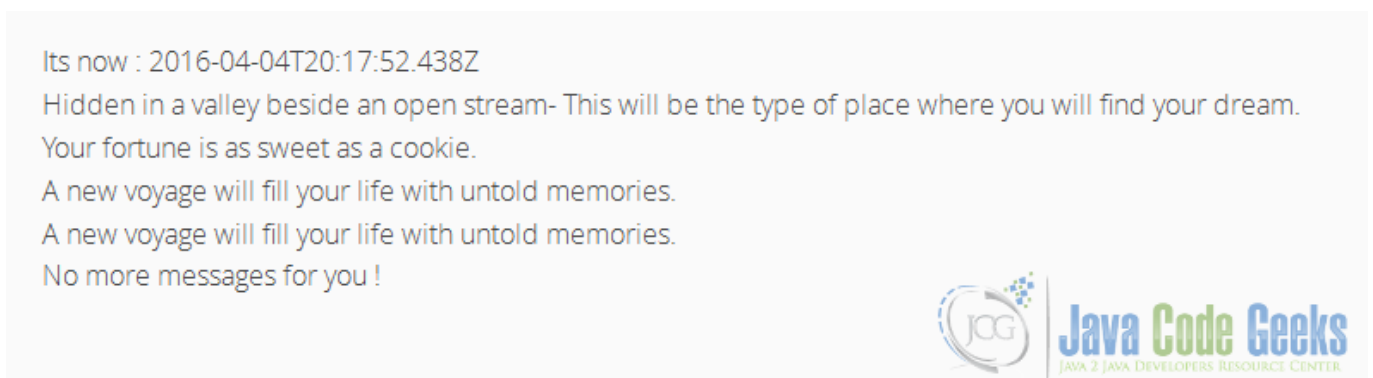


Figure 9.4: Running the example

After running the application for a while you get a lot of messages. Let's filter these messages to get only the server push messages.

First you get a update time message every second:

#### Time update

```
Mon Apr 04 15:47:05 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 20, "clientId": 0, "changes" : ←
  [], "state":{"34":{"text":"Its now : 2016-04-04T20:17:05.428Z"}}, "types":{"34":"1"}, " ←
  hierarchy":{}}, "rpc" : [], "meta" : {"async":true}, "resources" : {}, "timings":[115, ←
  15]] VaadinServerPush:1:5094
Mon Apr 04 15:47:05 GMT-430 2016 com.vaadin.client.communication.MessageHandler
Mon Apr 04 15:47:06 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 21, "clientId": 0, "changes" : ←
  [], "state":{"34":{"text":"Its now : 2016-04-04T20:17:06.429Z"}}, "types":{"34":"1"}, " ←
  hierarchy":{}}, "rpc" : [], "meta" : {"async":true}, "resources" : {}, "timings":[115, ←
  15]] VaadinServerPush:1:5094
Mon Apr 04 15:47:06 GMT-430 2016 com.vaadin.client.communication.MessageHandler
```

And every 10 seconds you should get a random message:

#### Random Message

```
Mon Apr 04 15:47:06 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 22, "clientId": 0, "changes" : ←
  [], "state":{"33":{"childData":{"34":{"alignmentBitmask":5,"expandRatio":0},"35":{" ←
  alignmentBitmask":5,"expandRatio":0},"36":{"alignmentBitmask":5,"expandRatio":0}}}, "36" ←
  :{"text":"Your fortune is as sweet as a cookie.", "width":"100.0%"}}, "types":{"33":"2", " ←
  36":"1"}, "hierarchy":{"33":["34","35","36"]}, "rpc" : [], "meta" : {"async":true}, " ←
  resources" : {}, "timings":[115, 15]] VaadinServerPush:1:5094
Mon Apr 04 15:47:06 GMT-430 2016 com.vaadin.client.communication.MessageHandler
Mon Apr 04 15:47:16 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 33, "clientId": 0, "changes" : ←
  [], "state":{"33":{"childData":{"34":{"alignmentBitmask":5,"expandRatio":0},"35":{" ←
  alignmentBitmask":5,"expandRatio":0},"36":{"alignmentBitmask":5,"expandRatio":0},"37":{" ←
  alignmentBitmask":5,"expandRatio":0}}}, "37":{"text":"A new voyage will fill your life ←
  with untold memories.", "width":"100.0%"}}, "types":{"33":"2","37":"1"}, "hierarchy":{"33 ←
  "":["34","35","36","37"]}, "rpc" : [], "meta" : {"async":true}, "resources" : {}, " ←
  timings":[115, 15]] VaadinServerPush:1:5094
Mon Apr 04 15:47:16 GMT-430 2016 com.vaadin.client.communication.MessageHandler
Mon Apr 04 15:47:26 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 44, "clientId": 0, "changes" : ←
  [], "state":{"33":{"childData":{"34":{"alignmentBitmask":5,"expandRatio":0},"35":{" ←
  alignmentBitmask":5,"expandRatio":0},"36":{"alignmentBitmask":5,"expandRatio":0},"37":{" ←
  alignmentBitmask":5,"expandRatio":0},"38":{"alignmentBitmask":5,"expandRatio":0}}}, "38" ←
  :{"text":"A new voyage will fill your life with untold memories.", "width":"100.0%"}}, " ←
  types":{"33":"2","38":"1"}, "hierarchy":{"33":["34","35","36","37","38"]}, "rpc" : [], " ←
  meta" : {"async":true}, "resources" : {}, "timings":[115, 15]] VaadinServerPush:1:5094
Mon Apr 04 15:47:26 GMT-430 2016 com.vaadin.client.communication.MessageHandler
Mon Apr 04 15:47:26 GMT-430 2016 com.vaadin.client.communication.AtmospherePushConnection
INFO: Received push (websocket) message: for(;;);[{"syncId": 45, "clientId": 0, "changes" : ←
  [], "state":{"33":{"childData":{"34":{"alignmentBitmask":5,"expandRatio":0},"35":{" ←
  alignmentBitmask":5,"expandRatio":0},"36":{"alignmentBitmask":5,"expandRatio":0},"37":{" ←
  alignmentBitmask":5,"expandRatio":0},"38":{"alignmentBitmask":5,"expandRatio":0},"39":{" ←
  alignmentBitmask":5,"expandRatio":0}}}, "39":{"text":"No more messages for you !", "width" ←
  : "100.0%"}}, "types":{"33":"2","39":"1"}, "hierarchy":{"33":["34","35","36","37","38", " ←
  39"]}, "rpc" : [], "meta" : {"async":true}, "resources" : {}, "timings":[115, 15]] ←
  VaadinServerPush:1:5094
Mon Apr 04 15:47:26 GMT-430 2016 com.vaadin.client.communication.MessageHandler
```

Let's see this in detail. For every message you get:

`com.vaadin.client.communication.AtmospherePushConnection` Vaadin uses the Atmosphere framework to manage server push, Atmosphere is a framework that that include a mix of WebSocket, Comet and RESTful behavior.

Also you get: `INFO:Received push (websocket) message:` that is telling you that the webapplication is receiving server push messages.

Now we are sure that Vaadin is using server push to update our page.

## 9.9 Download the Source Code

This was an example of Vaadin Server Push.

### Download

You can download the Eclipse project here: [VaadinServerPush](#)

---