



FREE eBook

LEARNING vaadin

Free unaffiliated eBook created from
Stack Overflow contributors.

#vaadin

Table of Contents

About.....	1
Chapter 1: Getting started with vaadin.....	2
Remarks.....	2
Versions.....	2
Examples.....	2
Installation or Setup.....	2
Create Vaadin project with Maven.....	2
Create Vaadin Project in Eclipse.....	3
Vaadin Plugin for Netbeans.....	4
Creating a Project.....	4
Exploring the Project.....	6
First program - "Hello World".....	8
Chapter 2: Login Page.....	10
Examples.....	10
SimpleLoginView.....	10
SimpleLoginUI.....	12
SimpleLoginMainView.....	13
Chapter 3: Themes.....	14
Examples.....	14
Valo.....	14
Reindeer.....	14
Chapter 4: Using add-ons with Vaadin.....	15
Examples.....	15
Using add-ons in a Maven project.....	15
Add-ons In Eclipse.....	16
Chapter 5: Vaadin and Maven.....	17
Remarks.....	17
Examples.....	17
Vaadin Maven Setup.....	17
Pom.....	17

Chapter 6: Vaadin TouchKit	22
Examples.....	22
Setup.....	22
Credits	23

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [vaadin](#)

It is an unofficial and free vaadin ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official vaadin.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with vaadin

Remarks

Vaadin is a server side scripting language, written in Java, which will generate most of the client side code needed for a web application. It uses Google Web Toolkit to generate the client side objects and can be extended by creating by extending the Google Web Toolkit.

Versions

Version	Release Date
6.8.17	2016-03-21
7.6.8	2016-07-13

Examples

Installation or Setup

<https://vaadin.com/framework/get-started>

Create Vaadin project with Maven

With Maven you can create Vaadin project with `vaadin-archetype-application` archetype. You can also add that archetype in IDE to create maven project with IDE.

```
mvn archetype:generate
-DarchetypeGroupId=com.vaadin
-DarchetypeArtifactId=vaadin-archetype-application
-DarchetypeVersion=7.6.8
-DgroupId=myvaadin.project
-DartifactId=DemoVaadinProject
-Dversion=0.1
-Dpackaging=war
```

Once you execute above command you will have following project structure.

```
DemoVaadinProject
|-src
  |-main
    |-java
      |   |-myvaadin
      |       |-project
      |           |-MyUI.java
    |-resource
      |   |-myvaadin
```

```
|           |-project
|           |-MyAppWidgetset.gwt.xml
|-webapps
    |- VAADIN
        |-theme
            |- mytheme.scss
            |- addons.scss
            |- styles.scss
            |- favicon.ico
```

The created default maven project can be imported in IDE directly. To run the maven application we must compile the default widget sets of vaadin.

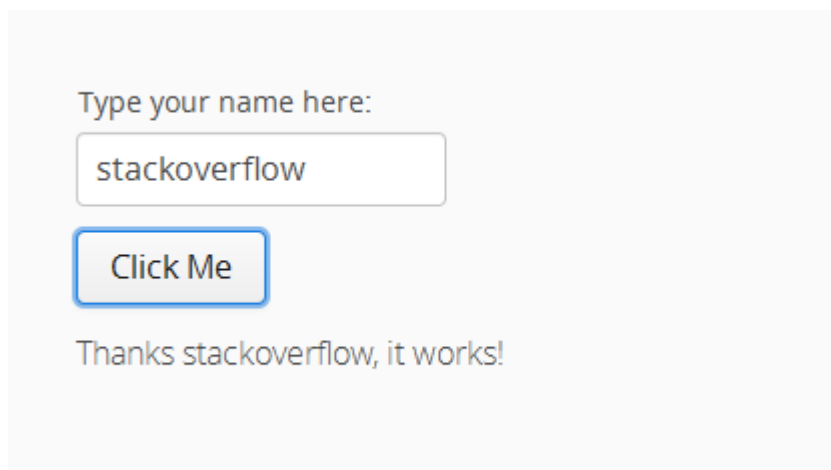
Note that we can directly use following maven command to package vaadin application and it will compile the widgetsets by default. You can use maven jetty plugin to deploy the vaadin application on Jetty.

```
cd path/to/DemoVaadinProject
mvn package jetty:run
```

This will deploy default application and start running it on default port 8080. You can access the deployed application at <http://localhost:8080>.

It is ready to run without any changes. By default Vaadin archetype adds default theme, widgetset xml and `MyUI` class which is an entry point for vaadin application.

In browser we will see following form.



Type your name here:

Thanks stackoverflow, it works!

Create Vaadin Project in Eclipse

Vaadin plug-in for eclipse provides a quick way to build vaadin project with Apache Ivy dependency manager. Vaadin's [documentation](#) explains how to create vaadin project with the help of Eclipse plugin.

To install the plug-in just go to eclipse marketplace and search *vaadin*. Current version of the plug-in is 3.0.0.

After installing the plug-in you will have following quick features,

- Create `Vaadin6` or `Vaadin 7` project (*Default dependency manager is Ivy*)
- Compile Widgets (*To compile client side widgets*)
- Compile Theme (*To compile Theme to build final CSS*)
- Create Widget (*To build your custom widget*)
- Create Vaadin Theme

So, after setting up the plug-in, just create new Vaadin project with minimal configuration. You can also specify version of vaadin while creating the project.

- `File > New > Vaadin7 Project`
- Specify version of vaadin to be used in project
- Specify Target Runtime you want to use
- Finish!

It will take time to download all the jars required for vaadin,once Ivy resolve all the dependencies. You can directly run the project on the server and you will see one `Button` with `Click Me` in the browser screen. Kindly note that Vaadin7 is compatible with Java 6 and newer.

Vaadin Plugin for Netbeans

Creating a Project with the NetBeans IDE

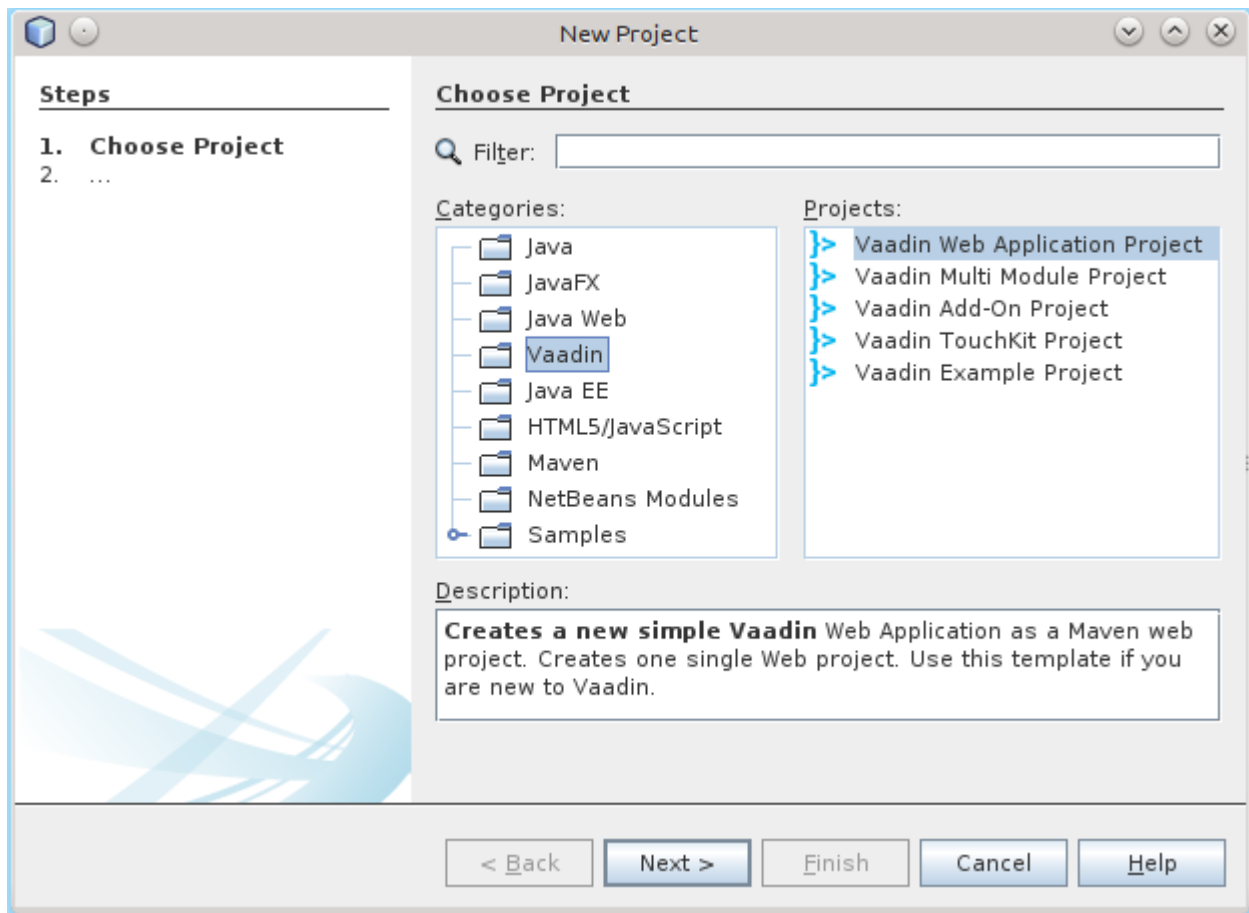
In the following, we walk you through the creation of a Vaadin project in NetBeans and show how to run it.

Installation of NetBeans and the Vaadin plugin is covered in [Installing the NetBeans IDE and Plugin](#).

Without the plugin, you can most easily create a Vaadin project as a Maven project using a Vaadin archetype. You can also create a Vaadin project as a regular web application project, but it requires many manual steps to install all the Vaadin libraries, create the UI class, configure the servlet, create theme, and so on.

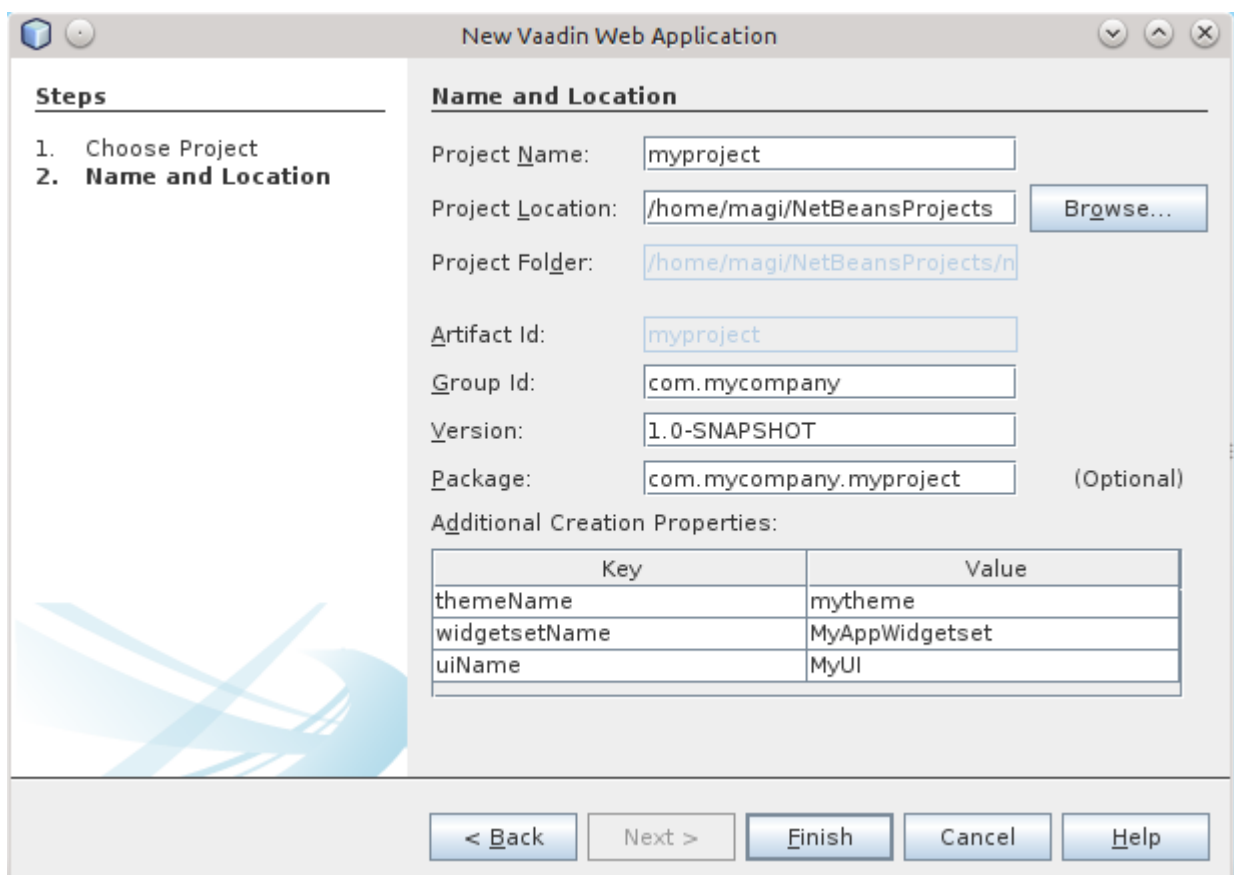
Creating a Project

1. Select **File > New Project...** from the main menu or press `Ctrl+Shift+N`.
2. In the New Project window that opens, select the Vaadin category and one of the Vaadin archetypes from the right.



The archetypes are described in more detail in Overview of Maven Archetypes.

3. In the **Name and Location** step, enter the project parameters.



Project Name

A project name. The name must be a valid identifier that may only contains alphanumerics, minus, and underscore. It is appended to the group ID to obtain the Java package name for the sources.

Project Location

Path to the folder where the project is to be created.

Group Id

A Maven group ID for your project. It is normally your organization domain name in reverse order, such as com.example. The group ID is also used as a prefix for the Java source package, so it should be Java-compatible package name.

Version

Initial version of your application. The number must obey the Maven version numbering format.

Package

The Java package name to put sources in.

Additional Creation Properties

The properties control various names. They are specific to the archetype you chose.

Click Finish.

Creating the project can take a while as Maven loads all the needed dependencies.

Exploring the Project

The project wizard has done all the work for you: a UI class skeleton has been written to the src directory. The project hierarchy shown in the Project Explorer is shown in [A new Vaadin project in NetBeans](#).

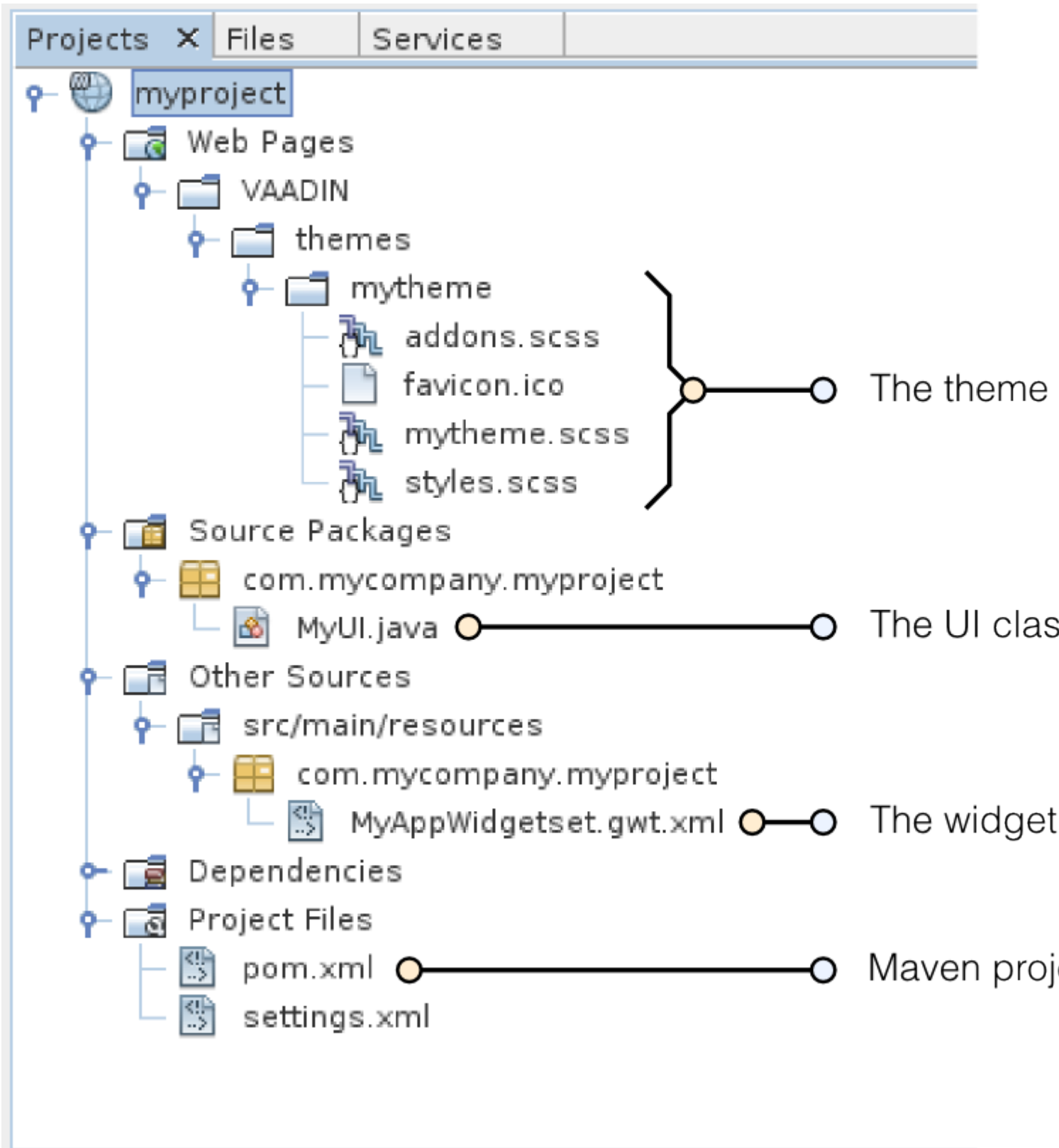


Figure 1. A new Vaadin project in NetBeans

mytheme

The theme of the UI. See Themes for information about themes.

MyUI.java

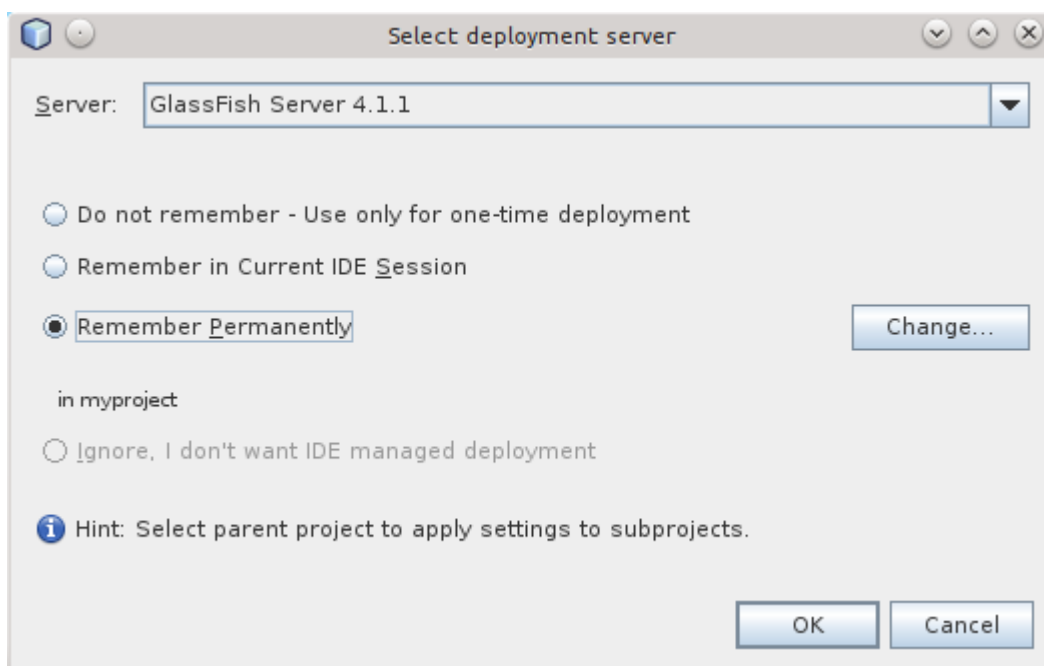
The UI class, which is the main entry-point of your application. See Server-Side Applications for

information about the basic structure of Vaadin applications.

The Vaadin libraries and other dependencies are managed by Maven. Notice that the libraries are not stored under the project folder, even though they are listed in the Java Resources > Libraries > Maven Dependencies virtual folder. Running the Application

Once created, you can run it in a server as follows.

1. In Projects tab, select the project and click in the Run Project button in the tool bar (or press F6).
2. In the Select deployment server window, select a server from the Server list. It should show either GlassFish or Apache Tomcat or both, depending on what you chose in NetBeans installation.



Also, select Remember Permanently if you want to use the same server also in future while developing applications.

Click OK.

The widget set will be compiled at this point, which may take a while.

If all goes well, NetBeans starts the server in port 8080 and, depending on your system configuration, launches the default browser to display the web application. If not, you can open it manually, for example, at <http://localhost:8080/myproject>. The project name is used by default as the context path of the application.

Now when you edit the UI class in the source editor and save it, NetBeans will automatically redeploy the application. After it has finished after a few seconds, you can reload the application in the browser.

First program - "Hello World"

Copy paste this code and launch your program :

```
@Theme(ValoTheme.THEME_NAME) //[optional] adds Vaadin built in theming
public class SampleUI extends UI {

    @Override
    protected void init(VaadinRequest request) {
        final VerticalLayout rootLayout = new VerticalLayout();
        Label label = new Label("Hello World!");
        rootLayout.addComponent(label);
        setContent(rootLayout);
    }
}
```

After launching was successful, please navigate to localhost:8080/yourApplicationName or <http://localhost:8080/> to see your app is up and running.

Read **Getting started with vaadin online**: <https://riptutorial.com/vaadin/topic/967/getting-started-with-vaadin>

Chapter 2: Login Page

Examples

SimpleLoginView

```
public class SimpleLoginView extends CustomComponent implements View,
Button.ClickListener {

    public static final String NAME = "login";

    private final TextField user;

    private final PasswordField password;

    private final Button loginButton;

    public SimpleLoginView() {
        setSizeFull();

        // Create the user input field
        user = new TextField("User:");
        user.setWidth("300px");
        user.setRequired(true);
        user.setInputPrompt("Your username (eg. joe@email.com)");
        user.addValidator(new EmailValidator(
            "Username must be an email address"));
        user.setInvalidAllowed(false);

        // Create the password input field
        password = new PasswordField("Password:");
        password.setWidth("300px");
        password.addValidator(new PasswordValidator());
        password.setRequired(true);
        password.setValue("");
        password.setNullRepresentation("");

        // Create login button
        loginButton = new Button("Login", this);

        // Add both to a panel
        VerticalLayout fields = new VerticalLayout(user, password, loginButton);
        fields.setCaption("Please login to access the application. (test@test.com/passw0rd)");
        fields.setSpacing(true);
        fields.setMargin(new MarginInfo(true, true, true, false));
        fields.setSizeUndefined();

        // The view root layout
        VerticalLayout viewLayout = new VerticalLayout(fields);
        viewLayout.setSizeFull();
        viewLayout.setComponentAlignment(fields, Alignment.MIDDLE_CENTER);
        viewLayout.setStyleName(Reindeer.LAYOUT_BLUE);
        setCompositionRoot(viewLayout);
    }

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

```

    // focus the username field when user arrives to the login view
    user.focus();
}

// Validator for validating the passwords
private static final class PasswordValidator extends
    AbstractValidator<String> {

    public PasswordValidator() {
        super("The password provided is not valid");
    }

    @Override
    protected boolean isValidValue(String value) {
        //
        // Password must be at least 8 characters long and contain at least
        // one number
        //
        if (value != null
            && (value.length() < 8 || !value.matches(".*\\d.*"))) {
            return false;
        }
        return true;
    }

    @Override
    public Class<String> getType() {
        return String.class;
    }
}

@Override
public void buttonClick(ClickEvent event) {

    //
    // Validate the fields using the navigator. By using validors for the
    // fields we reduce the amount of queries we have to use to the database
    // for wrongly entered passwords
    //
    if (!user.isValid() || !password.isValid()) {
        return;
    }

    String username = user.getValue();
    String password = this.password.getValue();

    //
    // Validate username and password with database here. For examples sake
    // I use a dummy username and password.
    //
    boolean isValid = username.equals("test@test.com")
        && password.equals("passw0rd");

    if (isValid) {

        // Store the current user in the service session
        getSession().setAttribute("user", username);

        // Navigate to main view
        getUI().getNavigator().navigateTo(SimpleLoginMainView.NAME); //
    }
}

```

```

    } else {

        // Wrong password clear the password field and refocuses it
        this.password.setValue(null);
        this.password.focus();

    }
}
}

```

SimpleLoginUI

```

public class SimpleLoginUI extends UI {

    @Override
    protected void init(VaadinRequest request) {

        //
        // Create a new instance of the navigator. The navigator will attach
        // itself automatically to this view.
        //
        new Navigator(this, this);

        //
        // The initial log view where the user can login to the application
        //
        getNavigator().addView(SimpleLoginView.NAME, SimpleLoginView.class);

        //
        // Add the main view of the application
        //
        getNavigator().addView(SimpleLoginMainView.NAME,
            SimpleLoginMainView.class);

        //
        // We use a view change handler to ensure the user is always redirected
        // to the login view if the user is not logged in.
        //
        getNavigator().addViewChangeListener(new ViewChangeListener() {

            @Override
            public boolean beforeViewChange(ViewChangeEvent event) {

                // Check if a user has logged in
                boolean isLoggedIn = getSession().getAttribute("user") != null;
                boolean isLoginView = event.getNewView() instanceof SimpleLoginView;

                if (!isLoggedIn && !isLoginView) {
                    // Redirect to login view always if a user has not yet
                    // logged in
                    getNavigator().navigateTo(SimpleLoginView.NAME);
                    return false;
                } else if (isLoggedIn && isLoginView) {
                    // If someone tries to access to login view while logged in,
                    // then cancel
                    return false;
                }
            }
        });
    }
}

```

```

        return true;
    }

    @Override
    public void afterViewChange(ViewChangeEvent event) {

    }
});
}
}

```

SimpleLoginMainView

```

public class SimpleLoginMainView extends CustomComponent implements View {

    public static final String NAME = "";

    Label text = new Label();

    Button logout = new Button("Logout", new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {

            // "Logout" the user
            getSession().setAttribute("user", null);

            // Refresh this view, should redirect to login view
            getUI().getNavigator().navigateTo(NAME);
        }
    });

    public SimpleLoginMainView() {
        setCompositionRoot(new CssLayout(text, logout));
    }

    @Override
    public void enter(ViewChangeEvent event) {
        // Get the user name from the session
        String username = String.valueOf(getSession().getAttribute("user"));

        // And show the username
        text.setValue("Hello " + username);
    }
}

```

Read Login Page online: <https://riptutorial.com/vaadin/topic/7912/login-page>

Chapter 3: Themes

Examples

Valo

```
@theme("valo")
```

Reindeer

```
@theme("reindeer")
```

Read Themes online: <https://riptutorial.com/vaadin/topic/7220/themes>

Chapter 4: Using add-ons with Vaadin

Examples

Using add-ons in a Maven project

To view and browse Vaadin add-ons in the Directory, you must be registered to vaadin.com. After the initial discovery of artifact details, e.g. for download and usage, registration is not required. Also, the usage of add-ons in a Maven project is not IDE-specific and the same instructions apply.

From a normal Maven project, start by editing your pom.xml :

1. Add the Vaadin add-on repository

```
<repositories>
  <repository>
    <id>vaadin-addons</id>
    <url>http://maven.vaadin.com/vaadin-addons</url>
  </repository>
  ...
</repositories>
```

2. Add the Vaadin Maven plugin in the maven build

```
<plugin>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-maven-plugin</artifactId>
  <version>7.6.8</version>
  <configuration>
    <extraJvmArgs>-Xmx512M -Xss1024k</extraJvmArgs>
    <webappDirectory>${basedir}/target/classes/VAADIN/widgetsets</webappDirectory>
    <draftCompile>false</draftCompile>
    <compileReport>false</compileReport>
    <style>OBF</style>
    <strict>true</strict>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>update-theme</goal>
        <goal>update-widgetset</goal>
        <goal>compile</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

3. Add the add-on as a normal dependency

```
<dependency>
  <groupId>org.vaadin</groupId>
  <artifactId>viritin</artifactId>
  <version>1.54</version>
</dependency>
```

```
</dependency>
```

4. If the add-on has client-side code, you need to update the widgetset XML and compile the widgetset:

```
mvn vaadin:update-widgetset vaadin:compile
```

Use the add-on in Java code as you would use any other Vaadin component.

Note that if you used a Vaadin Maven archetype to generate the project, you only need to go through steps 3 and 4, as the generated pom.xml contains the necessary information.

Add-ons In Eclipse

Download the .jar file from [vaadin add-ons](#) And put it in the lib folder of WEB-INF then right click on the .jar file and click on to Build Path --> Add To Build Path

Read [Using add-ons with Vaadin online](#): <https://riptutorial.com/vaadin/topic/5155/using-add-ons-with-vaadin>

Chapter 5: Vaadin and Maven

Remarks

This would be very useful to the Vaadin and Maven community because there is no documentation

Examples

Vaadin Maven Setup

Common Maven

```
mvn -B archetype:generate -DarchetypeGroupId=com.vaadin -DarchetypeArtifactId=vaadin-archetype-application -DarchetypeVersion=7.7.3 -DgroupId=org.test -DartifactId=vaadin-app -Dversion=1.0-SNAPSHOT
```

Advanced Maven

```
mvn archetype:generate \  
-DgroupId=com.mycompany.mycompanyapp \  
-DartifactId=mycompanyapp \  
-Dversion=1.0 \  
-DpackageName=com.mycompany.mycompanyapp \  
-DarchetypeGroupId=com.vaadin \  
-DarchetypeArtifactId=vaadin-archetype-application \  
-DthemeName=mytheme \  
-DuiName=MyCompanyAppUI \  
-DwidgetsetName=MyCompanyAppAppWidgetSet \  
-DarchetypeVersion=LATEST \  
-DinteractiveMode=false
```

After this is done, run following: `cd ~/mycompanyapp && mvn install -Dmaven.skip.tests=true`

Pom

- Repositories

```
<repository>  
  <id>vaadin-addons</id>  
  <url>http://maven.vaadin.com/vaadin-addons</url>  
</repository>  
<repository>  
  <id>vaadin-snapshots</id>  
  <name>Vaadin snapshot repository</name>  
  <url>http://oss.sonatype.org/content/repositories/vaadin-snapshots</url>  
  <snapshots>  
    <enabled>true</enabled>  
  </snapshots>  
</repository>
```

```

        <enabled>false</enabled>
    </releases>
</repository>
<repository>
    <id>vaadin-releases</id>
    <name>Vaadin releases</name>
    <url>https://oss.sonatype.org/content/repositories/vaadin-releases/</url>
</repository>`

```

- **Properties**

```

<properties>
    <vaadin.version>6.8-SNAPSHOT</vaadin.version>
    <gwt.version>2.3.0</gwt.version>
</properties>

```

- **Dependencies**

```

<dependency>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin-testbench</artifactId>
    <version>3.0.4</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>com.vaadin.addon</groupId>
    <artifactId>vaadin-touchkit-agpl</artifactId>
    <version>2.1.3</version>
    <type>jar</type>
</dependency>
<dependency>
    <groupId>org.vaadin.vol</groupId>
    <artifactId>openlayers-wrapper</artifactId>
    <version>1.2.0</version>
</dependency>
<dependency>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin</artifactId>
    <version>${vaadin.version}</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.google.gwt</groupId>
    <artifactId>gwt-user</artifactId>
    <version>${gwt.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>com.google.gwt</groupId>
    <artifactId>gwt-dev</artifactId>
    <version>${gwt.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>

```

```

    <!-- jsoup HTML parser library @ http://jsoup.org/ -->
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.6.3</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
</dependency>
<dependency>
    <groupId>org.vaadin.addons</groupId>
    <artifactId>formbinder</artifactId>
    <version>2.0.0</version>
</dependency>
<dependency>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-servlets</artifactId>
    <version>8.1.7.v20120910</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>LATEST</version>
    <scope>test</scope>
</dependency>`

```

- Build
- Plugins

```

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
        <source>1.7</source>
        <target>1.7</target>
    </configuration>
</plugin>
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>gwt-maven-plugin</artifactId>
    <version>2.3.0-1</version>
    <configuration>
        <extraJvmArgs>-Xmx512M -Xss1024k</extraJvmArgs>
        <!-- <runTarget>mobilemail</runTarget> -->
        <!-- We are doing "inplace" but into subdir VAADIN/widgetsets. This
             way compatible with Vaadin eclipse plugin. -->
        <webappDirectory>${basedir}/src/main/webapp/VAADIN/widgetsets
        </webappDirectory>
        <hostedWebapp>${basedir}/src/main/webapp/VAADIN/widgetsets
        </hostedWebapp>
        <noServer>true</noServer>
        <!-- Remove draftCompile when project is ready -->
        <draftCompile>false</draftCompile>
        <compileReport>false</compileReport>
        <style>OBF</style>
        <runTarget>http://localhost:8080</runTarget>
    </configuration>
    <executions>

```

```

        <execution>
            <goals>
                <goal>resources</goal>
                <goal>compile</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<!-- As we are doing "inplace" GWT compilatio, ensure the widgetset -->
<!-- directory is cleaned properly -->
<plugin>
    <artifactId>maven-clean-plugin</artifactId>
    <version>2.4.1</version>
    <configuration>
        <filesets>
            <fileset>
                <directory>src/main/webapp/VAADIN/widgetsets</directory>
            </fileset>
        </filesets>
    </configuration>
</plugin>

<plugin>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin-maven-plugin</artifactId>
    <version>1.0.2</version>
    <executions>
        <execution>
            <configuration>
                <!-- if you don't specify any modules, the plugin will find them -->
                <!-- <modules>
<module>com.vaadin.demo.mobilemail.gwt.ColorPickerWidgetSet</module>
                </modules> -->
            </configuration>
            <goals>
                <goal>update-widgetset</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<plugin>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>8.1.6.v20120903</version>
    <configuration>
        <systemProperties>
            <systemProperty>
                <name>jetty.port</name>
                <value>5678</value>
            </systemProperty>
        </systemProperties>
    </configuration>
    <executions>
        <!-- start and stop jetty (running our app) when running integration
        tests -->
        <execution>
            <id>start-jetty</id>
            <phase>pre-integration-test</phase>
            <goals>
                <goal>run-exploded</goal>
            </goals>

```

```
        <configuration>
            <scanIntervalSeconds>0</scanIntervalSeconds>
            <daemon>>true</daemon>
            <stopKey>STOP</stopKey>
            <stopPort>8866</stopPort>
        </configuration>
    </execution>
    <execution>
        <id>stop-jetty</id>
        <phase>post-integration-test</phase>
        <goals>
            <goal>stop</goal>
        </goals>
        <configuration>
            <stopPort>8866</stopPort>
            <stopKey>STOP</stopKey>
        </configuration>
    </execution>
</executions>
</plugin>
```

Read Vaadin and Maven online: <https://riptutorial.com/vaadin/topic/7221/vaadin-and-maven>

Chapter 6: Vaadin TouchKit

Examples

Setup

```
@Theme("mobiletheme")
@Widgetset("com.example.myapp.MyAppWidgetSet")
@Title("My Mobile App")
public class SimplePhoneUI extends UI {

    @Override
    protected void init(VaadinRequest request) {

        // Define a view

        class MyView extends NavigationView {

            public MyView() {
                super("Planet Details");
                CssLayout content = new CssLayout();
                setContent(content);
                VerticalComponentGroup group = new VerticalComponentGroup();
                content.addComponent(group);
                group.addComponent(new TextField("Planet"));
                group.addComponent(new NumberField("Found"));
                group.addComponent(new Switch("Probed"));
                setRightComponent(new Button("OK"));
            }
        }

        // Use it as the content root
        setContent(new MyView());
    }
}
```

Read Vaadin TouchKit online: <https://riptutorial.com/vaadin/topic/7913/vaadin-touchkit>

Credits

S. No	Chapters	Contributors
1	Getting started with vaadin	coder-croc , Community , Draken , javydreamercsw , Morfic , Reborn , Will Pierlot
2	Login Page	Will Pierlot
3	Themes	Will Pierlot
4	Using add-ons with Vaadin	coder-croc , Draken , ripla , Will Pierlot
5	Vaadin and Maven	Reborn , Will Pierlot
6	Vaadin TouchKit	Reborn , Will Pierlot