

 eBook Gratuit

APPRENEZ vaadin

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

#vaadin

Table des matières

À propos.....	1
Chapitre 1: Démarrer avec le vaadin.....	2
Remarques.....	2
Versions.....	2
Exemples.....	2
Installation ou configuration.....	2
Créer un projet Vaadin avec Maven.....	2
Créer un projet Vaadin dans Eclipse.....	3
Vaadin Plugin pour Netbeans.....	4
Créer un projet.....	4
Explorer le projet.....	6
Premier programme - "Hello World".....	9
Chapitre 2: Des thèmes.....	10
Exemples.....	10
Valo.....	10
Renne.....	10
Chapitre 3: Page de connexion.....	11
Exemples.....	11
SimpleLoginView.....	11
SimpleLoginUI.....	13
SimpleLoginMainView.....	14
Chapitre 4: Utiliser des add-ons avec Vaadin.....	15
Exemples.....	15
Utilisation de modules complémentaires dans un projet Maven.....	15
Modules complémentaires dans Eclipse.....	16
Chapitre 5: Vaadin et Maven.....	17
Remarques.....	17
Exemples.....	17
Configuration de Vaadin Maven.....	17
Pom.....	17

Chapitre 6: Vaadin TouchKit	22
Examples.....	22
Installer.....	22
Crédits	23

À propos

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

Chapitre 1: Démarrer avec le vaadin

Remarques

Vaadin est un langage de script côté serveur, écrit en Java, qui générera la plupart du code côté client nécessaire à une application Web. Il utilise Google Web Toolkit pour générer les objets côté client et peut être étendu en créant l'extension Google Web Toolkit.

Versions

Version	Date de sortie
6.8.17	2016-03-21
7.6.8	2016-07-13

Exemples

Installation ou configuration

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

Créer un projet Vaadin avec Maven

Avec Maven, vous pouvez créer un projet Vaadin avec `vaadin-archetype-application` archetype. Vous pouvez également ajouter cet archétype dans IDE pour créer un projet Maven avec 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
```

Une fois que vous avez exécuté la commande ci-dessus, vous aurez la structure de projet suivante.

```
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
```

Le projet Maven créé par défaut peut être importé directement dans IDE. Pour exécuter l'application Maven, nous devons compiler les jeux de widgets par défaut de vaadin.

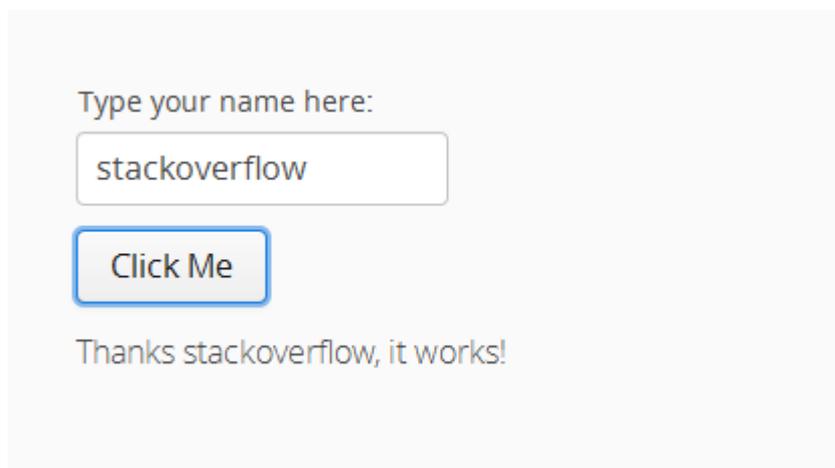
Notez que nous pouvons directement utiliser la commande suivante pour compiler l'application vaadin et compiler les widgets par défaut. Vous pouvez utiliser le plug-in maven jetty pour déployer l'application vaadin sur Jetty.

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

Cela déploiera l'application par défaut et commencera à l'exécuter sur le port 8080 par défaut. Vous pouvez accéder à l'application déployée à l' [adresse http://localhost:8080](http://localhost:8080) .

Il est prêt à fonctionner sans aucune modification. Par défaut, l'archétype Vaadin ajoute le thème par défaut, le widgetset xml et la classe `MyUI` , qui est un point d'entrée pour l'application vaadin.

Dans le navigateur, nous verrons le formulaire suivant.



Type your name here:

Thanks stackoverflow, it works!

Créer un projet Vaadin dans Eclipse

Le plug-in Vaadin pour eclipse permet de créer rapidement un projet vaadin avec le gestionnaire de dépendances Apache Ivy. La [documentation](#) de Vaadin explique comment créer un projet vaadin à l'aide du plugin Eclipse.

Pour installer le plug-in, rendez-vous sur eclipse marketplace et recherchez *vaadin* . La version actuelle du plug-in est 3.0.0 .

Après avoir installé le plug-in, vous aurez les fonctionnalités rapides suivantes,

- Créer un projet `Vaadin6` ou `Vaadin 7` (le gestionnaire de dépendance par défaut est Ivy)
- Compiler les widgets (*pour compiler les widgets côté client*)
- Compiler Thème (*Compiler Thème pour construire le CSS final*)
- Créer un widget (*pour créer votre widget personnalisé*)
- Créer un thème Vaadin

Ainsi, après avoir configuré le plug-in, créez simplement un nouveau projet Vaadin avec une configuration minimale. Vous pouvez également spécifier la version de vaadin lors de la création du projet.

- `File > New > Vaadin7 Project`
- Spécifiez la version de vaadin à utiliser dans le projet
- Spécifiez le runtime cible que vous souhaitez utiliser
- Terminer!

Il faudra du temps pour télécharger tous les fichiers requis pour le vaadin, une fois que Ivy aura résolu toutes les dépendances. Vous pouvez directement exécuter le projet sur le serveur et vous verrez un `Button` avec `Click Me` dans l'écran du navigateur. Veuillez noter que Vaadin7 est compatible avec Java 6 et versions ultérieures.

Vaadin Plugin pour Netbeans

Création d'un projet avec l'EDI NetBeans

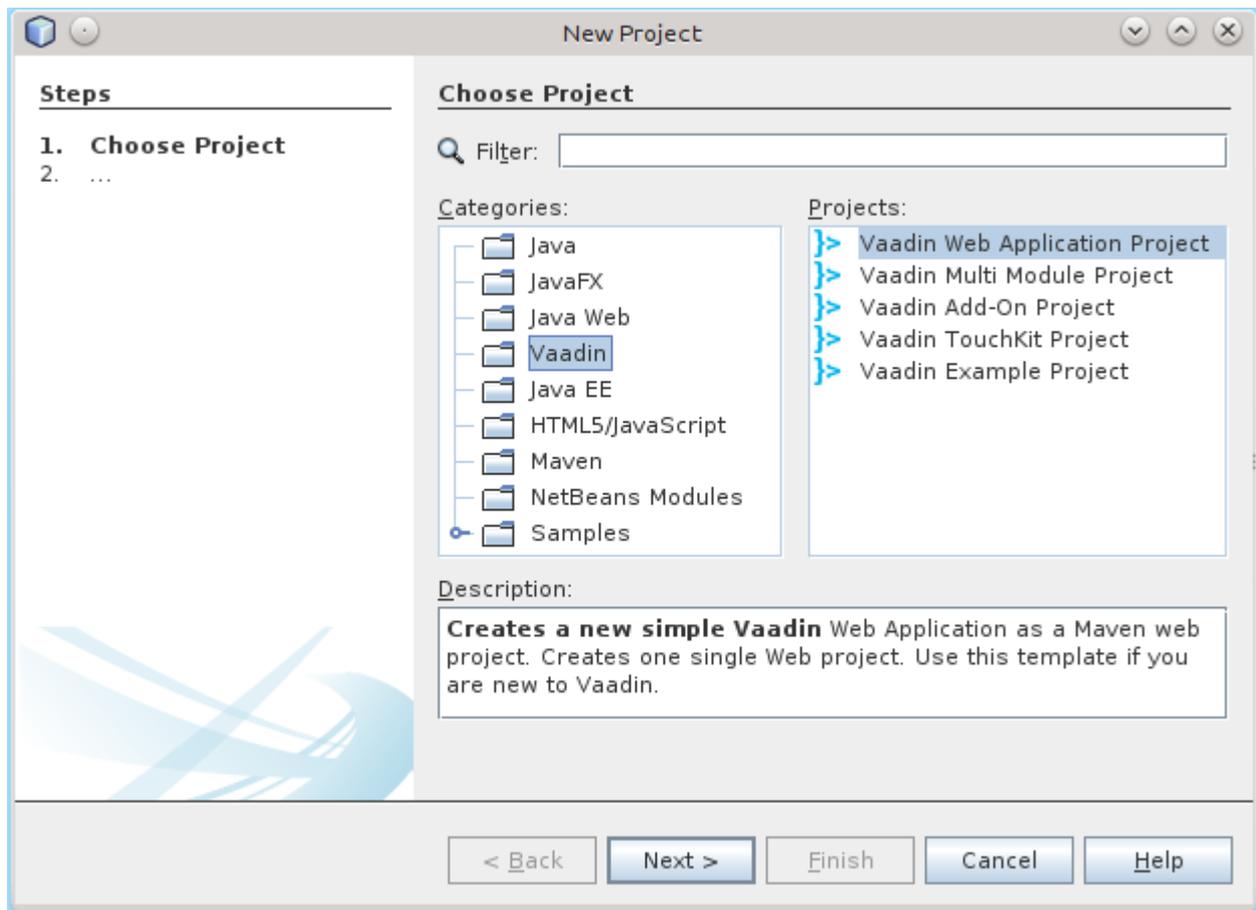
Nous vous présentons ci-après la création d'un projet Vaadin dans NetBeans et montrons comment l'exécuter.

L'installation de NetBeans et du plugin Vaadin est décrite dans Installation de l'EDI et du plug-in NetBeans.

Sans le plugin, vous pouvez facilement créer un projet Vaadin en tant que projet Maven en utilisant un archétype Vaadin. Vous pouvez également créer un projet Vaadin en tant que projet d'application Web classique, mais cela nécessite de nombreuses étapes manuelles pour installer toutes les bibliothèques Vaadin, créer la classe d'interface utilisateur, configurer le servlet, créer un thème, etc.

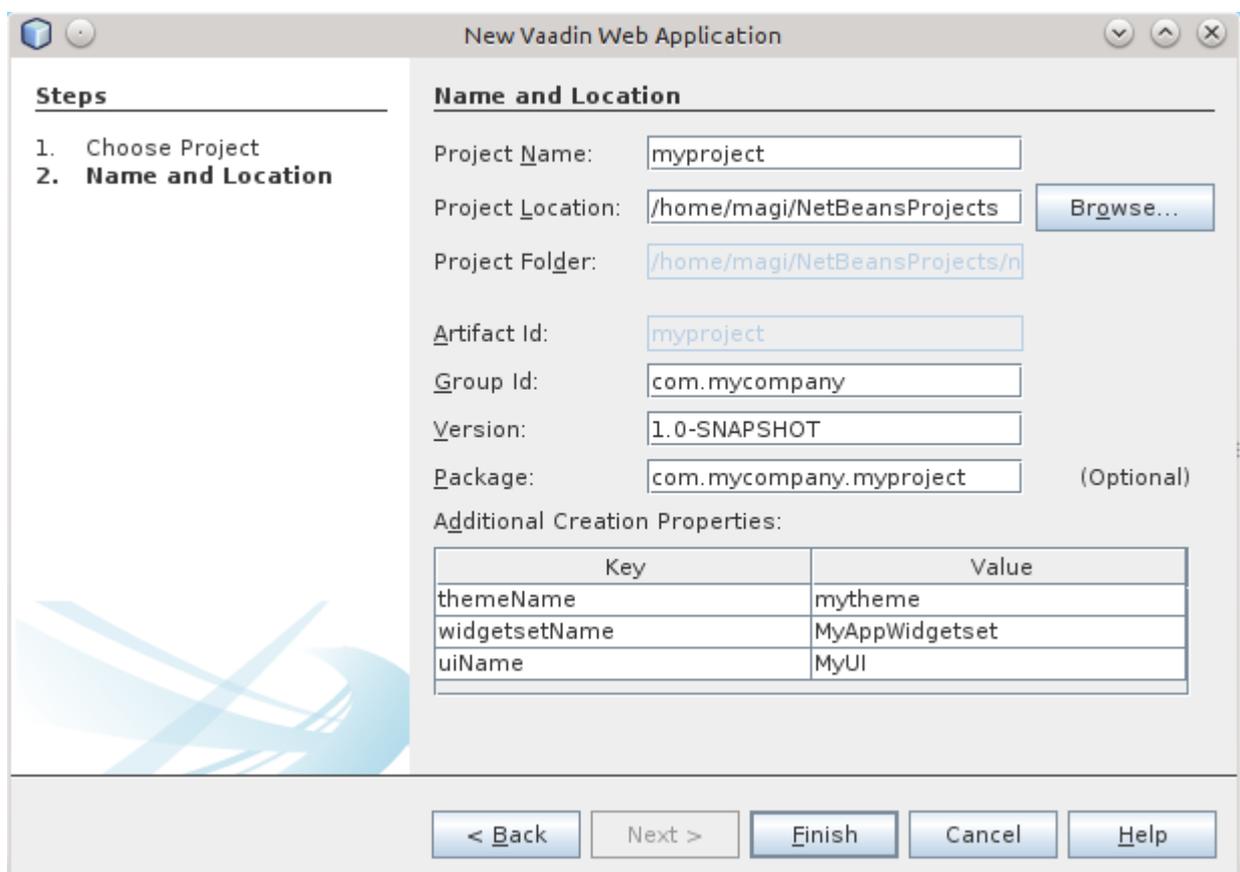
Créer un projet

1. Sélectionnez **Fichier > Nouveau projet ...** dans le menu principal ou appuyez sur `Ctrl + Maj + N`.
2. Dans la fenêtre Nouveau projet qui s'ouvre, sélectionnez la catégorie Vaadin et l'un des archétypes Vaadin à droite.



Les archétypes sont décrits plus en détail dans Présentation des archétypes Maven.

3. Dans l'étape **Nom et emplacement** , entrez les paramètres du projet.



nom du projet

Un nom de projet. Le nom doit être un identificateur valide qui ne peut contenir que des caractères alphanumériques, moins et un trait de soulignement. Il est ajouté à l'ID de groupe pour obtenir le nom du package Java pour les sources.

Emplacement du projet

Chemin d'accès au dossier dans lequel le projet doit être créé.

Identifiant de groupe

Un identifiant de groupe Maven pour votre projet. Il s'agit normalement du nom de domaine de votre organisation dans l'ordre inverse, tel que com.example. L'ID de groupe est également utilisé comme préfixe pour le package source Java, il doit donc s'agir d'un nom de package compatible avec Java.

Version

Version initiale de votre application. Le numéro doit respecter le format de numérotation de la version Maven.

Paquet

Le nom du package Java dans lequel placer les sources.

Propriétés de création supplémentaires

Les propriétés contrôlent différents noms. Ils sont spécifiques à l'archétype que vous avez choisi.

Cliquez sur Terminer.

La création du projet peut prendre un certain temps car Maven charge toutes les dépendances nécessaires.

Explorer le projet

L'assistant de projet a fait tout le travail pour vous: un squelette de classe d'interface utilisateur a été écrit dans le répertoire src. La hiérarchie de projet affichée dans l'explorateur de projet est affichée dans [Un nouveau projet Vaadin dans NetBeans](#) .

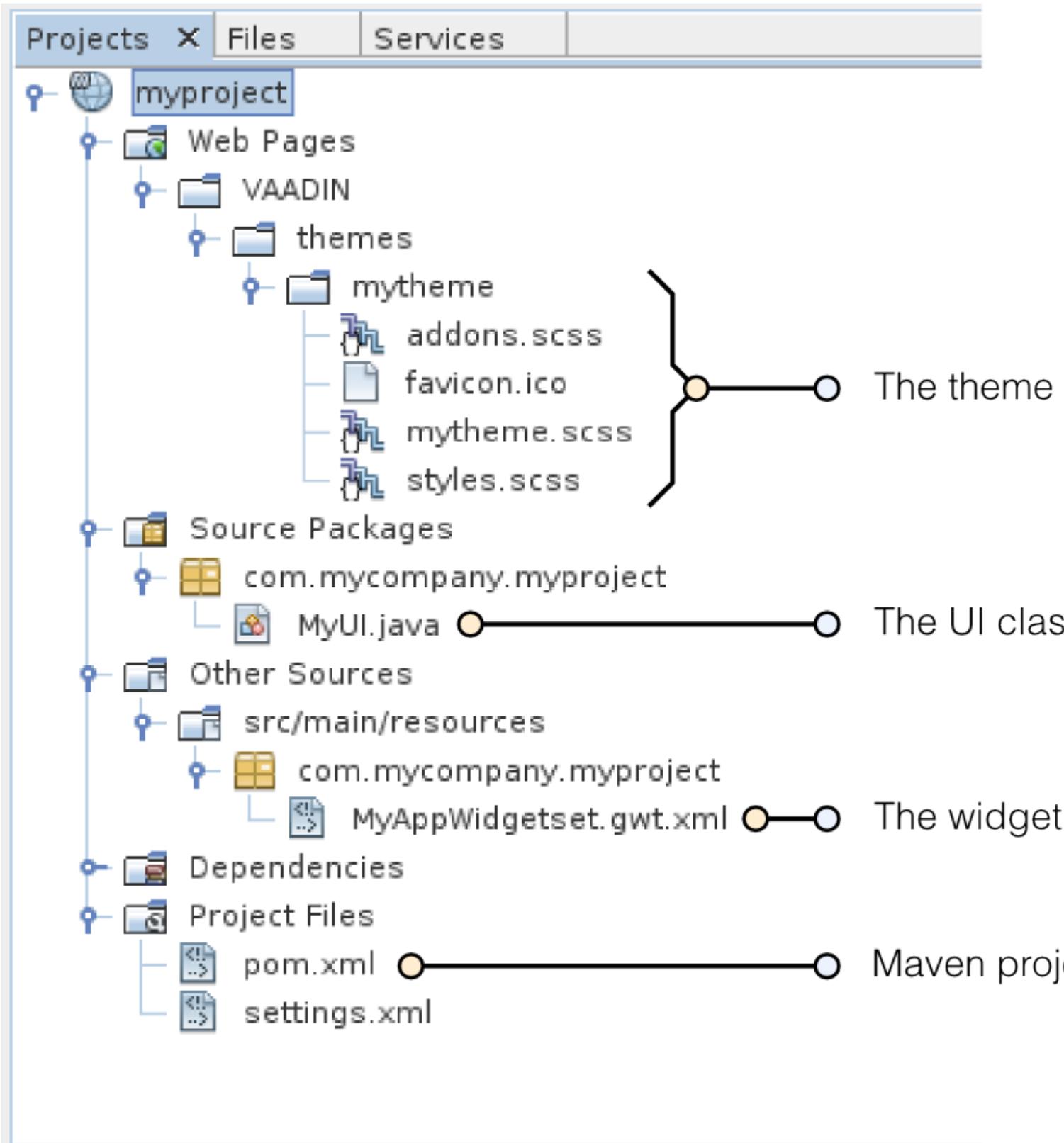


Figure 1. Un nouveau projet Vaadin dans NetBeans

mon thème

Le thème de l'interface utilisateur. Voir Thèmes pour plus d'informations sur les thèmes.

MyUI.java

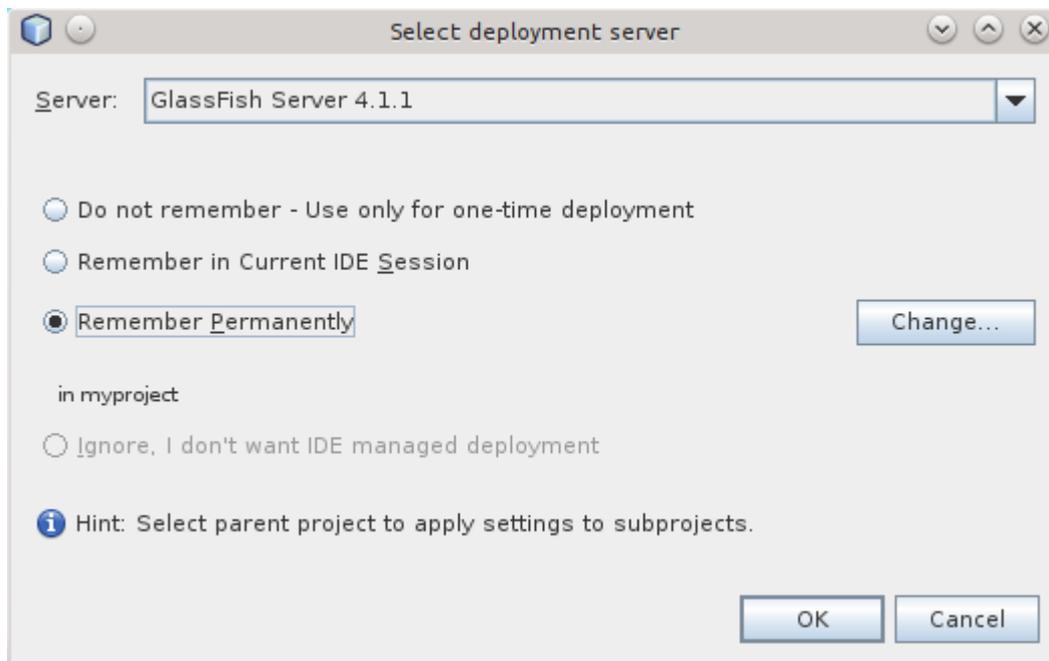
La classe d'interface utilisateur, qui est le principal point d'entrée de votre application. Voir

Applications côté serveur pour plus d'informations sur la structure de base des applications Vaadin.

Les bibliothèques Vaadin et autres dépendances sont gérées par Maven. Notez que les bibliothèques ne sont pas stockées dans le dossier du projet, même si elles figurent dans le dossier virtuel `Ressources Java > Bibliothèques > Dépendances Maven`. Exécution de l'application

Une fois créé, vous pouvez l'exécuter sur un serveur comme suit.

1. Dans l'onglet Projets, sélectionnez le projet et cliquez sur le bouton Exécuter le projet dans la barre d'outils (ou appuyez sur F6).
2. Dans la fenêtre Sélectionner le serveur de déploiement, sélectionnez un serveur dans la liste Serveur. Il devrait afficher GlassFish ou Apache Tomcat ou les deux, en fonction de ce que vous avez choisi dans l'installation de NetBeans.



De même, sélectionnez l'option Mémoriser en permanence si vous souhaitez également utiliser le même serveur lors du développement d'applications.

Cliquez sur OK.

L'ensemble de widgets sera compilé à ce stade, ce qui peut prendre un certain temps.

Si tout se passe bien, NetBeans démarre le serveur sur le port 8080 et, en fonction de la configuration de votre système, lance le navigateur par défaut pour afficher l'application Web. Sinon, vous pouvez l'ouvrir manuellement, par exemple, à l'adresse <http://localhost:8080/myproject>. Le nom du projet est utilisé par défaut comme chemin de contexte de l'application.

Désormais, lorsque vous éditez la classe d'interface utilisateur dans l'éditeur source et que vous l'enregistrez, NetBeans redéploie automatiquement l'application. Après quelques secondes, vous pouvez recharger l'application dans le navigateur.

Premier programme - "Hello World"

Copiez collez ce code et lancez votre programme:

```
@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);
    }
}
```

Une fois le lancement réussi, accédez à [localhost: 8080 / yourApplicationName](http://localhost:8080/yourApplicationName) ou [http: // localhost: 8080 /](http://localhost:8080/) pour voir votre application opérationnelle.

Lire Démarrer avec le vaadin en ligne: <https://riptutorial.com/fr/vaadin/topic/967/demarrer-avec-le-vaadin>

Chapitre 2: Des thèmes

Exemples

Valo

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

Renne

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

Lire Des thèmes en ligne: <https://riptutorial.com/fr/vaadin/topic/7220/des-themes>

Chapitre 3: Page de connexion

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);
    }
}

```

Lire Page de connexion en ligne: <https://riptutorial.com/fr/vaadin/topic/7912/page-de-connexion>

Chapitre 4: Utiliser des add-ons avec Vaadin

Exemples

Utilisation de modules complémentaires dans un projet Maven

Pour afficher les modules complémentaires Vaadin dans l'annuaire, vous devez être enregistré sur vaadin.com. Après la découverte initiale des détails de l'artefact, par exemple pour le téléchargement et l'utilisation, l'enregistrement n'est pas requis. En outre, l'utilisation de modules complémentaires dans un projet Maven n'est pas spécifique à l'IDE et les mêmes instructions s'appliquent.

À partir d'un projet Maven normal, commencez par éditer votre fichier pom.xml:

1. Ajouter le dépôt complémentaire Vaadin

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

2. Ajouter le plugin Vaadin Maven dans le build maven

```
<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. Ajouter le module complémentaire en tant que dépendance normale

```
<dependency>
```

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

4. Si l'add-on a du code côté client, vous devez mettre à jour le code XML et compiler le widget:

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

Utilisez le module complémentaire en code Java comme vous utiliseriez tout autre composant Vaadin.

Notez que si vous avez utilisé un archétype Vaadin Maven pour générer le projet, il vous suffit de passer par les étapes 3 et 4, car le fichier pom.xml généré contient les informations nécessaires.

Modules complémentaires dans Eclipse

Téléchargez le fichier .jar à partir des [add-ons vaadin](#) et placez-le dans le dossier lib de WEB-INF, puis cliquez avec le bouton droit sur le fichier .jar et cliquez sur Build Path -> Add To Build Path

Lire Utiliser des add-ons avec Vaadin en ligne: <https://riptutorial.com/fr/vaadin/topic/5155/utiliser-des-add-ons-avec-vaadin>

Chapitre 5: Vaadin et Maven

Remarques

Ce serait très utile pour la communauté Vaadin et Maven car il n'y a pas de documentation

Exemples

Configuration de Vaadin Maven

Maven commun

```
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
```

Maven Avancé

```
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
```

Une fois cette opération terminée, exécutez la commande suivante: `cd ~/mycompanyapp && mvn install -Dmaven.skip.tests=true`

Pom

- Référentiels

```
<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>  
  <releases>  
    <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>`

```

- Propriétés

```

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

```

- Les dépendances

```

<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>`

```

- Construire
- 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>
```

Lire Vaadin et Maven en ligne: <https://riptutorial.com/fr/vaadin/topic/7221/vaadin-et-maven>

Chapitre 6: Vaadin TouchKit

Examples

Installer

```
@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());
    }
}
```

Lire Vaadin TouchKit en ligne: <https://riptutorial.com/fr/vaadin/topic/7913/vaadin-touchkit>

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec le vaadin	coder-croc , Community , Draken , javydreamercsw , Morfic , Reborn , Will Pierlot
2	Des thèmes	Will Pierlot
3	Page de connexion	Will Pierlot
4	Utiliser des add-ons avec Vaadin	coder-croc , Draken , ripla , Will Pierlot
5	Vaadin et Maven	Reborn , Will Pierlot
6	Vaadin TouchKit	Reborn , Will Pierlot