



EBook Gratuito

APPENDIMENTO vaadin

Free unaffiliated eBook created from
Stack Overflow contributors.

#vaadin

Sommario

Di.....	1
Capitolo 1: Iniziare con vaadin.....	2
Osservazioni.....	2
Versioni.....	2
Examples.....	2
Installazione o configurazione.....	2
Crea un progetto Vaadin con Maven.....	2
Crea un progetto Vaadin in Eclipse.....	3
Vaadin Plugin per Netbeans.....	4
Creare un progetto.....	4
Esplorando il progetto.....	6
Primo programma - "Hello World".....	9
Capitolo 2: Pagina di login.....	10
Examples.....	10
SimpleLoginView.....	10
SimpleLoginUI.....	12
SimpleLoginMainView.....	13
Capitolo 3: Temi.....	14
Examples.....	14
Valo.....	14
Renna.....	14
Capitolo 4: Utilizzo di componenti aggiuntivi con Vaadin.....	15
Examples.....	15
Utilizzo di componenti aggiuntivi in un progetto Maven.....	15
Componenti aggiuntivi in Eclipse.....	16
Capitolo 5: Vaadin e Maven.....	17
Osservazioni.....	17
Examples.....	17
Installazione di Vaadin Maven.....	17
Pom.....	17

Capitolo 6: Vaadin TouchKit	22
Examples	22
Impostare	22
Titoli di coda	23

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

Capitolo 1: Iniziare con vaadin

Osservazioni

Vaadin è un linguaggio di scripting lato server, scritto in Java, che genererà la maggior parte del codice lato client necessario per un'applicazione web. Utilizza Google Web Toolkit per generare gli oggetti lato client e può essere esteso creando estendendo Google Web Toolkit.

Versioni

Versione	Data di rilascio
6.8.17	2016/03/21
7.6.8	2016/07/13

Examples

Installazione o configurazione

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

Crea un progetto Vaadin con Maven

Con Maven puoi creare un progetto `vaadin-archetype-application` archetipo di `vaadin-archetype-application`. Puoi anche aggiungere quell'archetipo in IDE per creare un progetto maven con 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
```

Una volta eseguito il comando sopra, avrai la seguente struttura del progetto.

```
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
```

Il progetto maven predefinito creato può essere importato direttamente in IDE. Per eseguire l'applicazione Maven, dobbiamo compilare i set di widget predefiniti di vaadin.

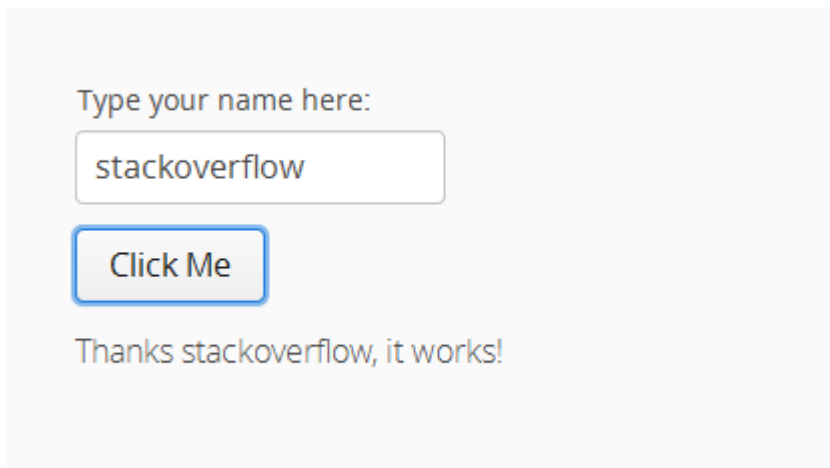
Nota che possiamo usare direttamente il seguente comando di maven per impacchettare l'applicazione di vaadin e che compilerà i widgetset per impostazione predefinita. È possibile utilizzare il plugin maven jetty per distribuire l'applicazione vaadin su Jetty.

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

Questo distribuirà l'applicazione predefinita e inizierà a eseguirla sulla porta predefinita 8080 . È possibile accedere all'applicazione distribuita all'indirizzo [http://localhost: 8080](http://localhost:8080) .

È pronto per essere eseguito senza modifiche. Per impostazione predefinita, l'archetipo Vaadin aggiunge il tema predefinito, widgetset xml e classe `MyUI` , che è un punto di ingresso per l'applicazione vaadin.

Nel browser vedremo il seguente modulo.



Crea un progetto Vaadin in Eclipse

Il plug-in Vaadin per eclipse fornisce un modo rapido per creare il progetto vaadin con il manager delle dipendenze di Apache Ivy. La [documentazione](#) di Vaadin spiega come creare un progetto vaadin con l'aiuto del plugin Eclipse.

Per installare il plug-in basta andare su eclipse marketplace e cercare *vaadin* . La versione corrente del plug-in è 3.0.0 .

Dopo aver installato il plug-in, avrai le seguenti funzioni rapide,

- Crea un progetto `Vaadin6` o `Vaadin 7` (*Il gestore delle dipendenze predefinito è Ivy*)
- Compilare Widgetsets (*per compilare i widget lato client*)
- Compilare il tema (*per compilare il tema per creare il CSS finale*)
- Crea widget (*per costruire il tuo widget personalizzato*)
- Crea un tema Vaadin

Quindi, dopo aver configurato il plug-in, basta creare un nuovo progetto Vaadin con una configurazione minima. È inoltre possibile specificare la versione di vaadin durante la creazione del progetto.

- `File > New > Vaadin7 Project`
- Specifica la versione di vaadin da utilizzare nel progetto
- Specifica Target Runtime che desideri utilizzare
- Finire!

Ci vorrà del tempo per scaricare tutti i barattoli necessari per vaadin, una volta che Ivy risolve tutte le dipendenze. Puoi eseguire direttamente il progetto sul server e vedrai un `Button` con `Click Me` nella schermata del browser. Si noti che Vaadin7 è compatibile con Java 6 e versioni successive.

Vaadin Plugin per Netbeans

Creazione di un progetto con l'IDE NetBeans

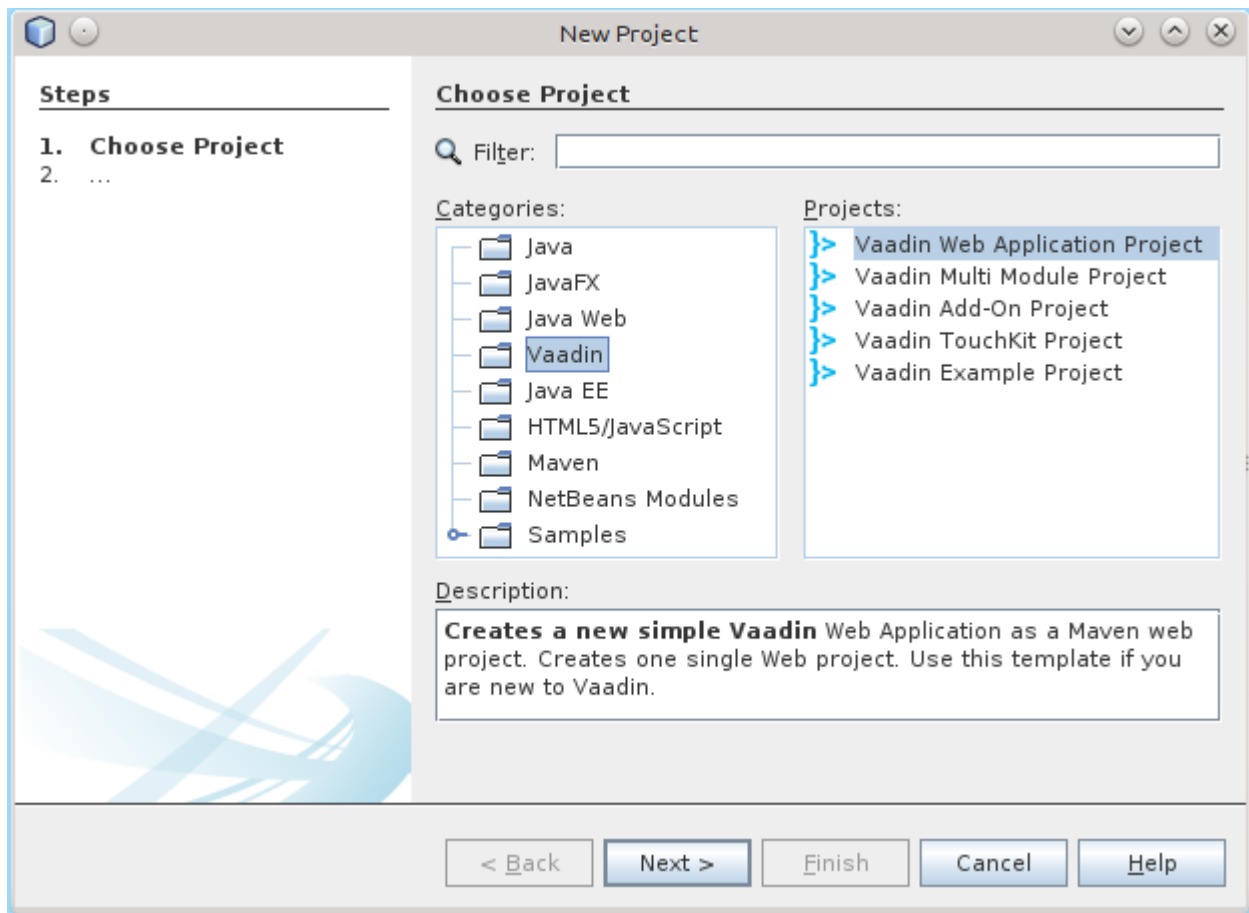
Di seguito, ti guideremo attraverso la creazione di un progetto Vaadin in NetBeans e mostreremo come eseguirlo.

L'installazione di NetBeans e del plug-in Vaadin è descritta in [Installazione dell'IDE e del plug-in NetBeans](#).

Senza il plugin, puoi facilmente creare un progetto Vaadin come progetto Maven usando un archetipo di Vaadin. È anche possibile creare un progetto Vaadin come un normale progetto di applicazione Web, ma richiede molti passaggi manuali per installare tutte le librerie Vaadin, creare la classe UI, configurare il servlet, creare il tema e così via.

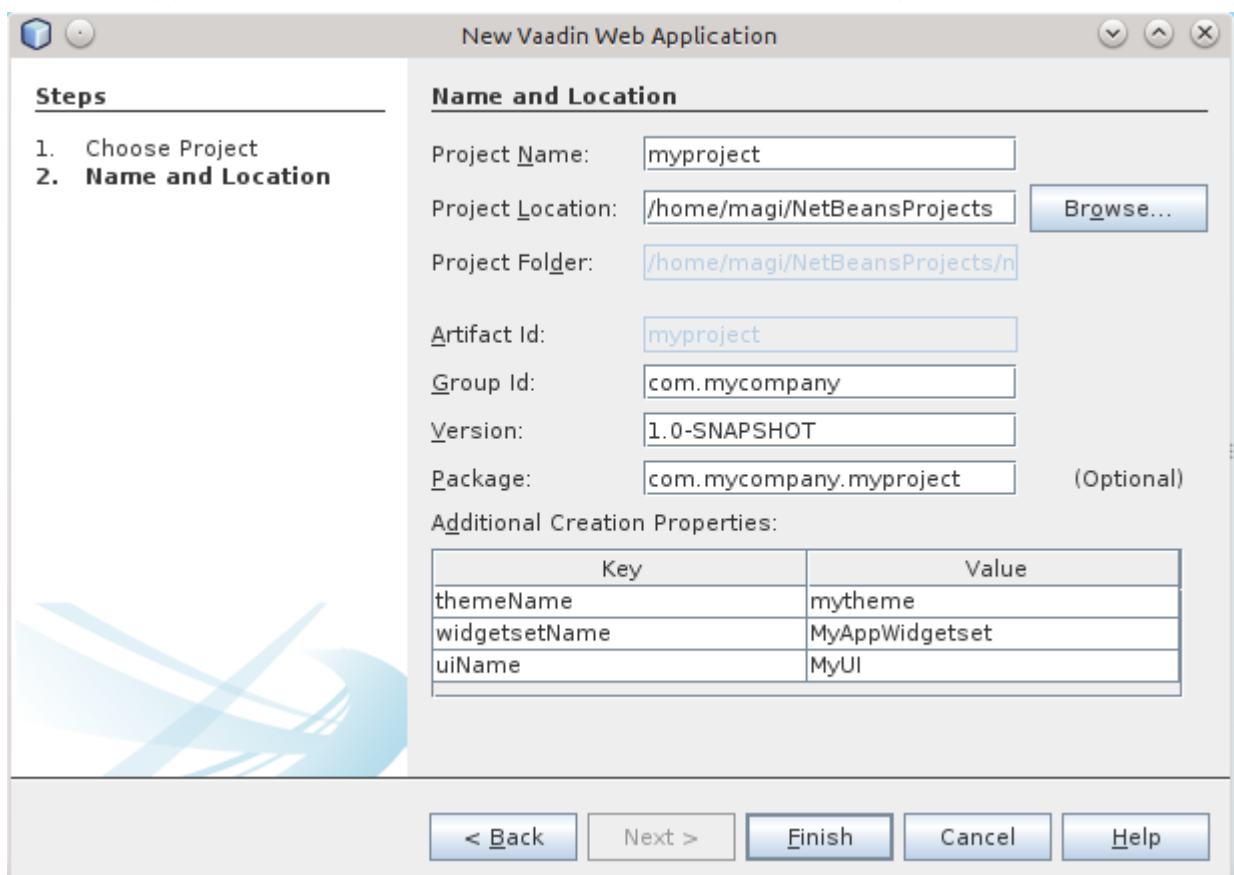
Creare un progetto

1. Selezionare **File** , **Nuovo progetto** ... dal menu principale oppure premere `Ctrl + Maiusc + N`.
2. Nella finestra Nuovo progetto che si apre, seleziona la categoria Vaadin e uno degli archetipi Vaadin dalla destra.



Gli archetipi sono descritti in modo più dettagliato in [Panoramica degli archetipi di Maven](#).

3. Nel passaggio **Nome e Posizione** , inserire i parametri del progetto.



Nome del progetto

Un nome di progetto. Il nome deve essere un identificatore valido che può contenere solo caratteri alfanumerici, meno e caratteri di sottolineatura. Viene aggiunto all'ID di gruppo per ottenere il nome del pacchetto Java per le origini.

Sede del progetto

Percorso della cartella in cui deve essere creato il progetto.

ID gruppo

Un ID di gruppo Maven per il tuo progetto. Normalmente è il nome del dominio dell'organizzazione in ordine inverso, come ad esempio com.example. L'ID gruppo viene anche usato come prefisso per il pacchetto sorgente Java, quindi dovrebbe essere il nome del pacchetto compatibile con Java.

Versione

Versione iniziale della tua applicazione. Il numero deve rispettare il formato di numerazione delle versioni di Maven.

Pacchetto

Il nome del pacchetto Java per inserire le fonti.

Ulteriori proprietà di creazione

Le proprietà controllano vari nomi. Sono specifici per l'archetipo che hai scelto.

Fai clic su Fine.

La creazione del progetto può richiedere del tempo poiché Maven carica tutte le dipendenze necessarie.

Esplorando il progetto

Il wizard del progetto ha fatto tutto il lavoro per te: uno scheletro della classe UI è stato scritto nella directory src. La gerarchia di progetto mostrata in Esplora progetti viene mostrata in [Un nuovo progetto Vaadin in NetBeans](#) .

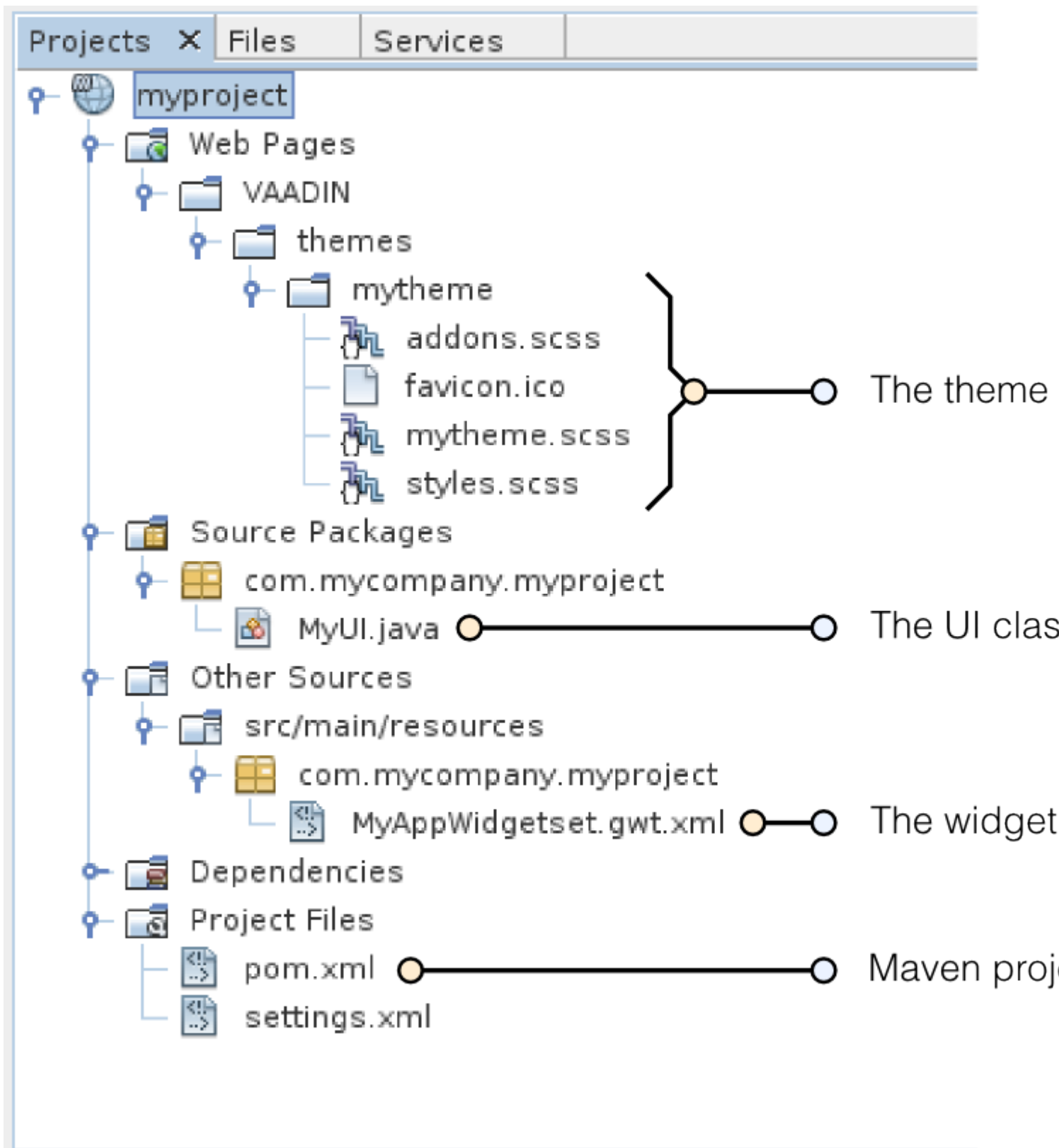


Figura 1. Un nuovo progetto Vaadin in NetBeans

MyTheme

Il tema dell'interfaccia utente. Vedi Temi per informazioni sui temi.

MyUI.java

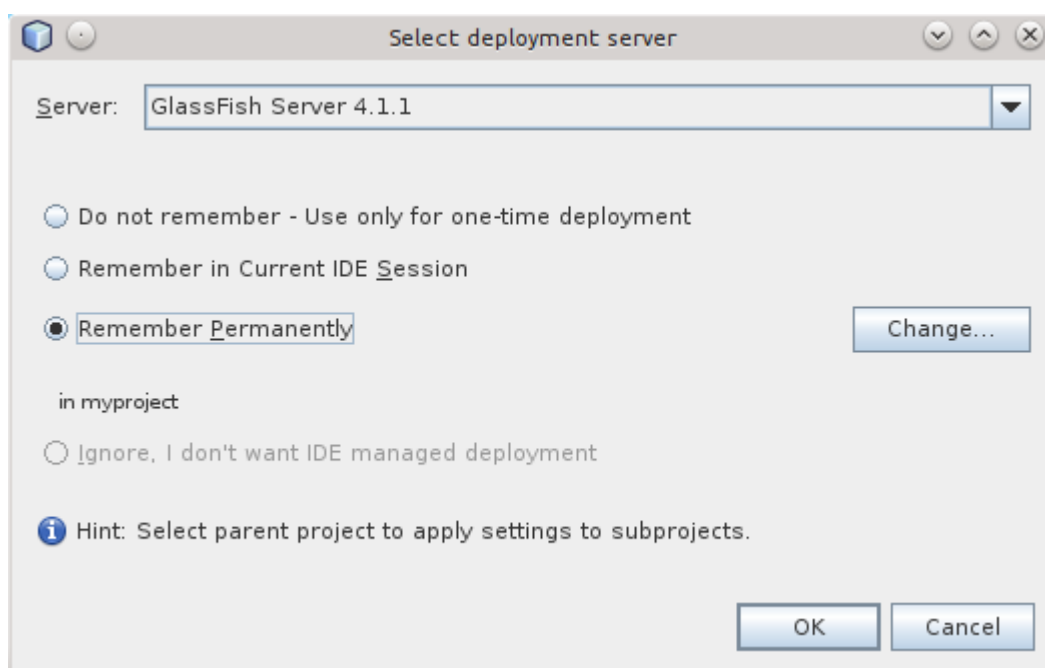
La classe UI, che è il punto di accesso principale della tua applicazione. Vedi Applicazioni lato

server per informazioni sulla struttura di base delle applicazioni Vaadin.

Le librerie di Vaadin e altre dipendenze sono gestite da Maven. Si noti che le librerie non sono archiviate nella cartella del progetto, anche se sono elencate nelle risorse Java ► Librerie ► Cartella virtuale Dipendenze Maven. Esecuzione dell'applicazione

Una volta creato, è possibile eseguirlo su un server come segue.

1. Nella scheda Progetti, selezionare il progetto e fare clic sul pulsante Esegui progetto nella barra degli strumenti (o premere F6).
2. Nella finestra Seleziona server di distribuzione, selezionare un server dall'elenco Server. Dovrebbe mostrare GlassFish o Apache Tomcat o entrambi, a seconda di cosa hai scelto nell'installazione di NetBeans.



Inoltre, selezionare Ricorda permanentemente se si desidera utilizzare lo stesso server anche in futuro durante lo sviluppo di applicazioni.

Clicca OK.

Il set di widget verrà compilato a questo punto, che potrebbe richiedere un po' di tempo.

Se tutto va bene, NetBeans avvia il server nella porta 8080 e, in base alla configurazione del sistema, avvia il browser predefinito per visualizzare l'applicazione web. In caso contrario, è possibile aprirlo manualmente, ad esempio, all'indirizzo <http://localhost:8080/myproject>. Il nome del progetto viene utilizzato per impostazione predefinita come percorso di contesto dell'applicazione.

Ora quando modifichi la classe UI nell'editor di origine e la salvi, NetBeans ridistribuirà automaticamente l'applicazione. Dopo che è terminato dopo alcuni secondi, puoi ricaricare l'applicazione nel browser.

Primo programma - "Hello World"

Copia incolla questo codice e avvia il tuo programma:

```
@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);
    }
}
```

Dopo aver effettuato il successo, passare a [localhost: 8080 / yourApplicationName](http://localhost:8080/yourApplicationName) o [http: // localhost: 8080 /](http://localhost:8080/) per vedere se la tua app è attiva e funzionante.

Leggi Iniziare con vaadin online: <https://riptutorial.com/it/vaadin/topic/967/iniziare-con-vaadin>

Capitolo 2: Pagina di login

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 validators 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);
    }
}

```

Leggi Pagina di login online: <https://riptutorial.com/it/vaadin/topic/7912/pagina-di-login>

Capitolo 3: Temi

Examples

Valo

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

Renna

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

Leggi Temi online: <https://riptutorial.com/it/vaadin/topic/7220/temi>

Capitolo 4: Utilizzo di componenti aggiuntivi con Vaadin

Examples

Utilizzo di componenti aggiuntivi in un progetto Maven

Per visualizzare i componenti aggiuntivi di Vaadin nella Directory, è necessario essere registrati su vaadin.com. Dopo la scoperta iniziale dei dettagli degli artefatti, ad esempio per il download e l'utilizzo, la registrazione non è richiesta. Inoltre, l'utilizzo di componenti aggiuntivi in un progetto Maven non è specifico per IDE e si applicano le stesse istruzioni.

Da un normale progetto Maven, inizia modificando il tuo pom.xml:

1. Aggiungi il repository aggiuntivo Vaadin

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

2. Aggiungi il plugin Maven di Vaadin nella build di 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. Aggiungi il componente aggiuntivo come una dipendenza normale

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

4. Se il componente aggiuntivo dispone di codice lato client, è necessario aggiornare il XML widgetset e compilare il widget widgets:

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

Utilizzare il componente aggiuntivo nel codice Java come si farebbe con qualsiasi altro componente Vaadin.

Tieni presente che se hai utilizzato un archetipo di Vaadin Maven per generare il progetto, devi solo eseguire i passaggi 3 e 4, poiché il pom.xml generato contiene le informazioni necessarie.

Componenti aggiuntivi in Eclipse

Scarica il file .jar dai [componenti aggiuntivi di vaadin](#) e inseriscilo nella cartella lib di WEB-INF, quindi fai clic con il pulsante destro del mouse sul file .jar e fai clic su Crea percorso -> Aggiungi a percorso di creazione

Leggi Utilizzo di componenti aggiuntivi con Vaadin online:

<https://riptutorial.com/it/vaadin/topic/5155/utilizzo-di-componenti-aggiuntivi-con-vaadin>

Capitolo 5: Vaadin e Maven

Osservazioni

Questo sarebbe molto utile alla comunità di Vaadin e Maven perché non c'è documentazione

Examples

Installazione di Vaadin Maven

Maven comune

```
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
```

Al termine, eseguire: `cd ~/mycompanyapp && mvn install -Dmaven.skip.tests=true`

Pom

- repository

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

```

- Proprietà

```

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

```

- dipendenze

```

<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>`

```

- Costruire
- plugin

```

<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>
```

Leggi Vaadin e Maven online: <https://riptutorial.com/it/vaadin/topic/7221/vaadin-e-maven>

Capitolo 6: Vaadin TouchKit

Examples

Impostare

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

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

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con vaadin	coder-croc , Community , Draken , javydreamercsw , Morfic , Reborn , Will Pierlot
2	Pagina di login	Will Pierlot
3	Temi	Will Pierlot
4	Utilizzo di componenti aggiuntivi con Vaadin	coder-croc , Draken , ripla , Will Pierlot
5	Vaadin e Maven	Reborn , Will Pierlot
6	Vaadin TouchKit	Reborn , Will Pierlot