

Installing Apache Tomcat with NetBeans



Sommario

- Apache Tomcat
 - Componenti di Apache Tomcat
- Web Application
 - Componenti di un Web Application
- Configurazione ambiente con Netbeans 8.x
 - Server Apache
 - Database
- Esempio di Registrazione e Login
- Tomcat Web Application Manager

Prima di iniziare...

- Scaricare Java SDK

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

- Scaricare Netbeans

<https://netbeans.org/downloads/>

- Scaricare la versione "MySQL Server" (32-64 bit)

<https://dev.mysql.com/downloads/mysql/>

Scaricare e installare Java SDK

- Scaricare Java SDK dal seguente link:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Java SE Development Kit 8u111

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Thank you for accepting the [Oracle Binary Code License Agreement for Java SE](#); you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.78 MB	jdk-8u111-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.73 MB	jdk-8u111-linux-arm64-vfp-hflt.tar.gz
Linux x86	160.35 MB	jdk-8u111-linux-i586.rpm
Linux x86	175.04 MB	jdk-8u111-linux-i586.tar.gz
Linux x64	158.35 MB	jdk-8u111-linux-x64.rpm
Linux x64	173.04 MB	jdk-8u111-linux-x64.tar.gz
Mac OS X	227.39 MB	jdk-8u111-macosx-x64.dmg
Solaris SPARC 64-bit	131.92 MB	jdk-8u111-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	93.02 MB	jdk-8u111-solaris-sparcv9.tar.gz
Solaris x64	140.38 MB	jdk-8u111-solaris-x64.tar.Z
Solaris x64	96.82 MB	jdk-8u111-solaris-x64.tar.gz
Windows x86	189.22 MB	jdk-8u111-windows-i586.exe
Windows x64	194.64 MB	jdk-8u111-windows-x64.exe

Scaricare Netbeans

- Scaricare Netbeans:

<https://netbeans.org/downloads/>

Subscribe to newsletters: Monthly Weekly
 NetBeans can contact me at this address

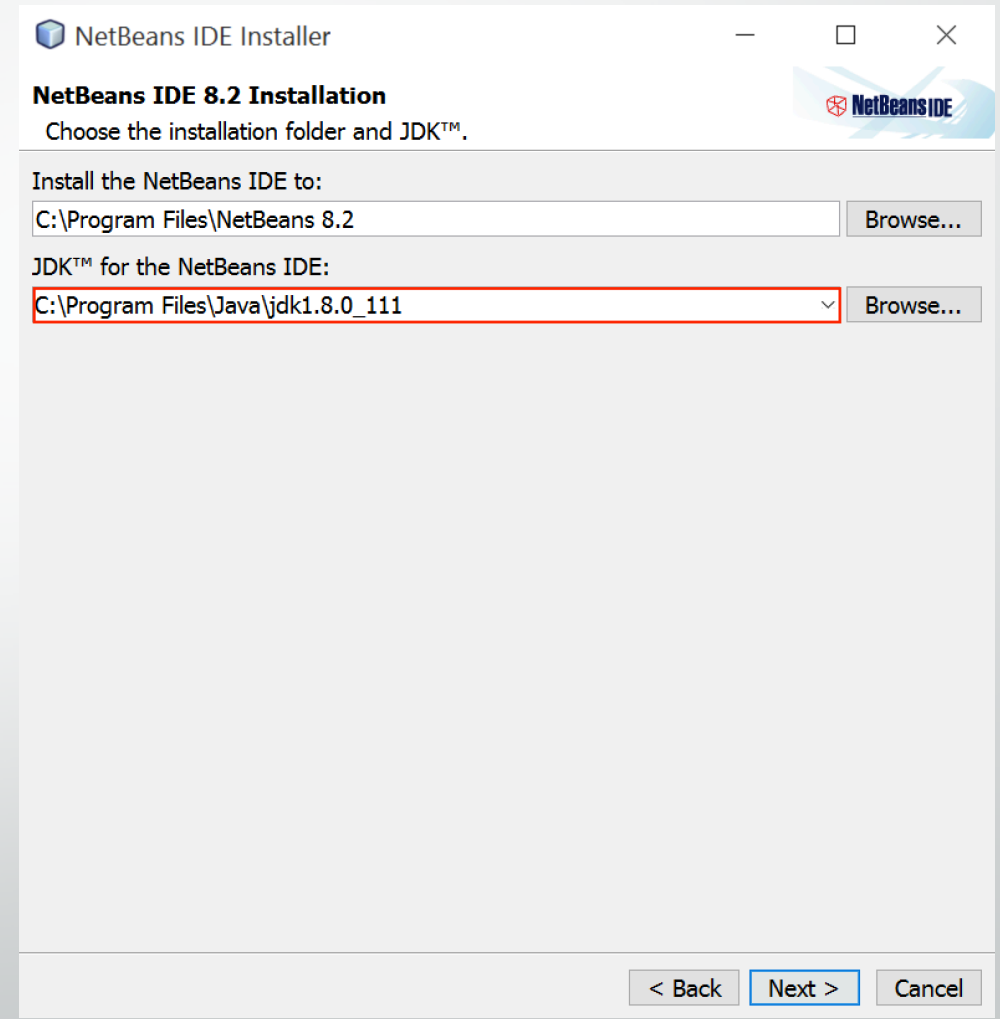
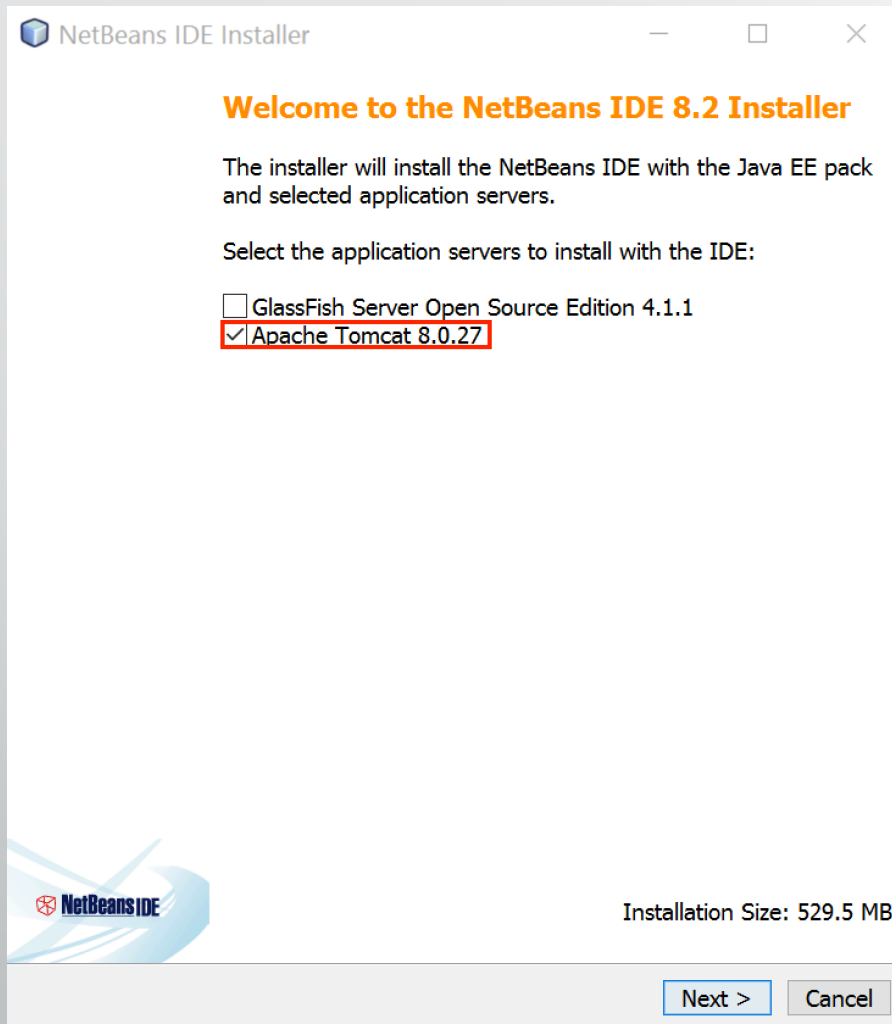
Language:

Note: Greyed out technologies are not supported

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++
NetBeans Platform SDK	•	•			
Java SE	•	•			
Java FX	•	•			
Java EE		•			
Java ME					
HTML5/JavaScript		•	•	•	
PHP			•	•	
C/C++					•
Groovy					
Java Card™ 3 Connected					
Bundled servers					
GlassFish Server Open Source Edition 4.1.1		•			
Apache Tomcat 8.0.27		•			

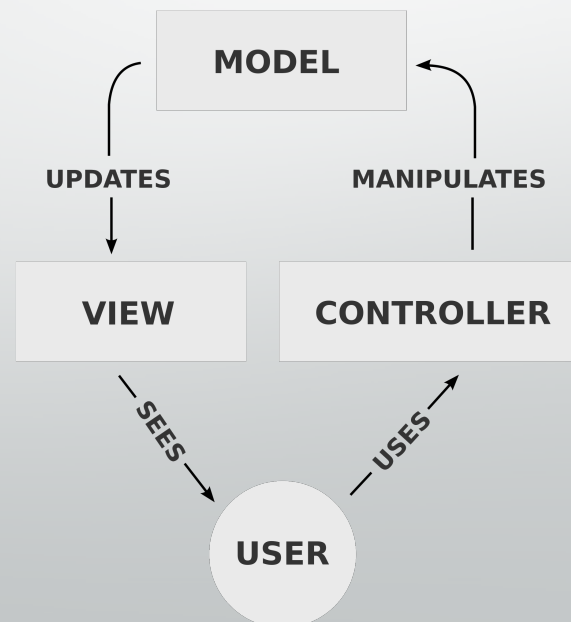
Installare Netbeans



Model View Controller (MVC)

MVC e' un utile pattern architetturale per la separazione dei compiti di un software.

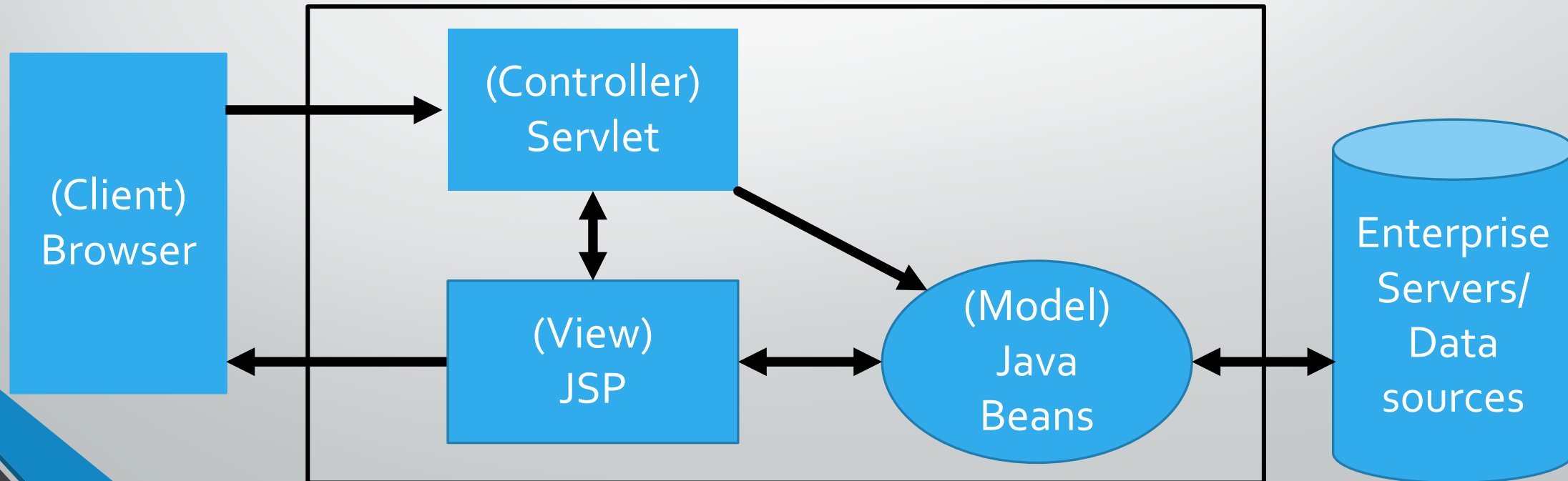
- **Model:** *Plain Old JavaScript Object* (POJO) che trasporta i dati.
- **View:** livello nel quale i dati sono rappresentati in formato visuale.
- **Controller:** componente responsabile per la comunicazione tra *model* e *view*.



Cos'è Apache Tomcat ?

- Apache Tomcat è un *Application Server* basato su Java Servlet
- Tomcat implementa diverse specifiche:
 - Java Servlet
 - JavaServer Pages (JSP)
 - ect

Architettura MVC



Componenti di Tomcat

1. Catalina

- Contenitore di *Servlet Java* di Tomcat
- Fornisce l'effettiva implementazione di Tomcat, delle specifiche della Servlet.

2. Coyote

- Componente "connettore HTTP" di Tomcat
- Supporta il protocollo HTTP per il Web Server
- Ascolta le connessioni in entrata su una specifica porta TCP, inoltra la richiesta al motore Tomcat e processa la richiesta e manda indietro la risposta al client.

3. Jasper

- Analizza e converte le porzioni di codice all'interno delle pagine JSP in Servlet e le passa a Catalina per processarle
- Quando lanciato, cerca eventuali cambiamenti avvenuti ai file JSP e, se necessario li ricompila.

Componenti di un Web Application (WA)

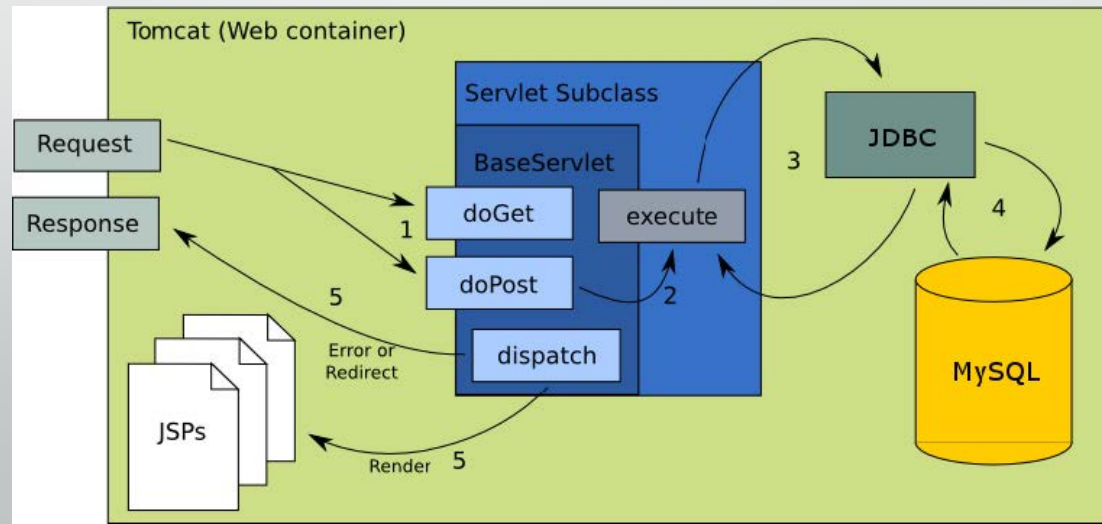
- Un *Web component* e' un oggetto che estende le funzionalita' di un Web Server aggiugendone di nuove:
 - **Servlet**
 - **Java Server Pages (JSP)**
 - ecc
- I *Web component* sono in grado di accedere ai servizi offerti da un Web container come:
 - Smistamento delle richieste
 - Sicurezza
 - Concorrenza
 - Gestione della memoria

Java Servlet (Controller)

- E' una classe scritta in linguaggio Java che estende le capacità di un server.
- Le funzionalità principali di una *Servlet* sono:
 - Elaborazione o memorizzazione di dati provenienti da form HTML;
 - Generazione di contenuti dinamici (pagine Web) a seconda dei parametri della richiesta inviata;
 - Gestione delle informazioni di stato che non esistono nel protocollo HTTP stateless.
- Comunicano attraverso un qualsiasi protocollo *Client-Server*, ma sono principalmente utilizzate con il protocollo *HTTP*.

Java Server Page (View)

- JSP e' una tecnologia di programmazione Web in Java per lo sviluppo di WA.
- Forniscono contenuti dinamici in formato HTML o XML.
- Si basa su un insieme di tag speciali con cui possono essere invocate funzioni predefinite o codice Java (e.g., `<% %>`).
- Puo' essere vista come una rappresentazione ad alto livello di una Servlet.



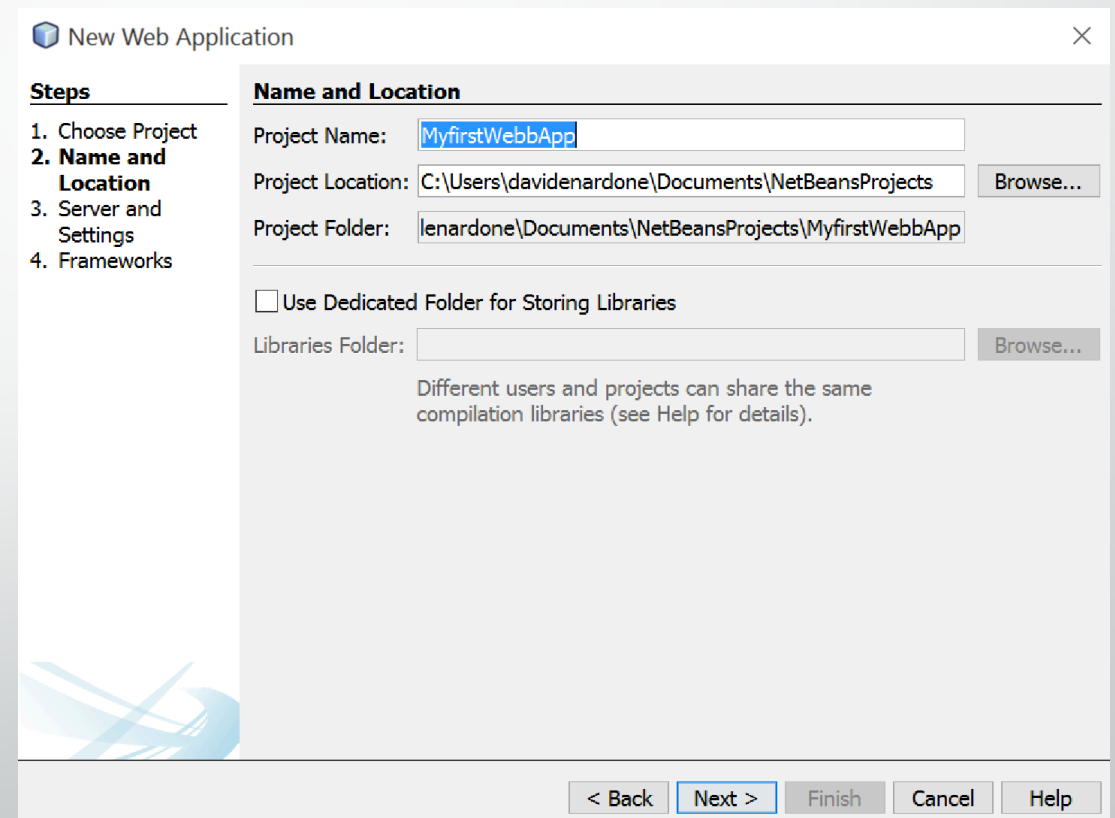
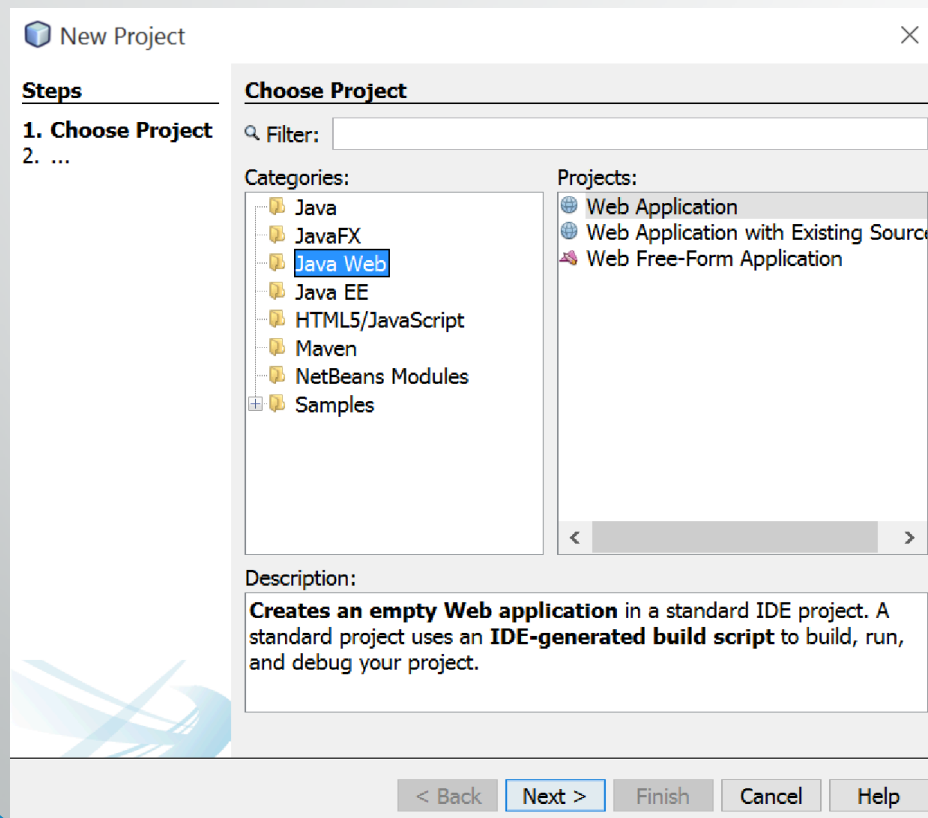
JavaBeans (Model)

- Classe che incapsula molti oggetti in un singolo oggetto (bean).
- Classe Java appositamente costruita e programmata in accordo alle specifiche delle API di JavaBeans.
- Oggetti serializzabili che permettono l'accesso a diverse proprietà attraverso i cosiddetti metodi *getter* e *setter*.
- Il modello rappresenta un oggetto POJO per il trasporto dei dati.

Vantaggi	Svantaggi
Controllo delle proprietà, eventi e metodi dei beans esposti ad altre applicazioni.	E' soggetto ad essere istanziato con un <i>stato invalido</i> avendo un costruttore nullo.
Registra eventi da altri oggetti e puo' generare eventi da poter inviare ad altri oggetti.	Sono oggetti intrinsecamente <i>mutabili</i> mancando così del vantaggio offerto dagli oggetti immutabili.
Impostazioni di configurazione di un bean possono essere memorizzati in modo persistente.	Il possedere molti metodi getter e setter puo' portare una quantità immensa di <i>boilerplate code</i> .

Configurazione dell'ambiente

- Una volta lanciato Netbeans...
- **Creare un nuovo progetto: *Java Web* -> *Web Application***



Configurazione dell'ambiente

- Se non presente di *default*, aggiungere il *Server Apache Tomcat 8.x*
- e modificare le *credenziali utente* di Tomcat

New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

Server: Apache Tomcat 8.0.27.0 **Add...**

Java EE Version: Java EE 7 Web

Context Path: /MyfirstWebbApp

< Back Next > Finish Cancel Help

Add Server Instance

Steps

1. Choose Server
2. **Installation and Login Details**

Installation and Login Details

Specify the Server Location (Catalina Home) and login details

Server Location: e Software Foundation\Apache Tomcat 8.0.27 **Browse ...**

Use Private Configuration Folder (Catalina Base)

Catalina Base: Browse ...

Enter the credentials of an existing user in the manager or manager-script role

Username: davide

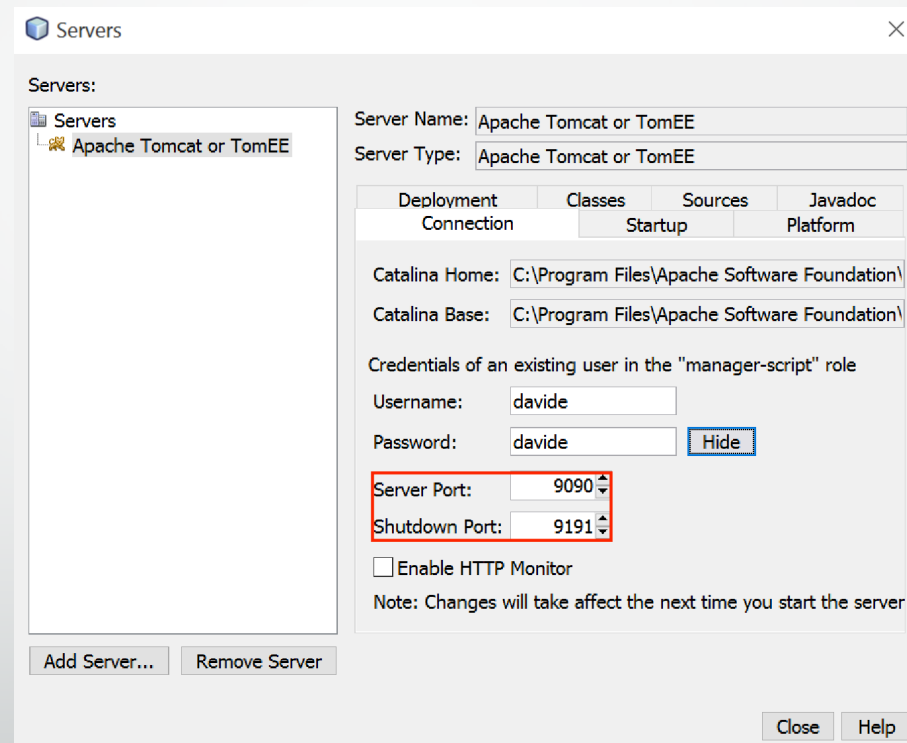
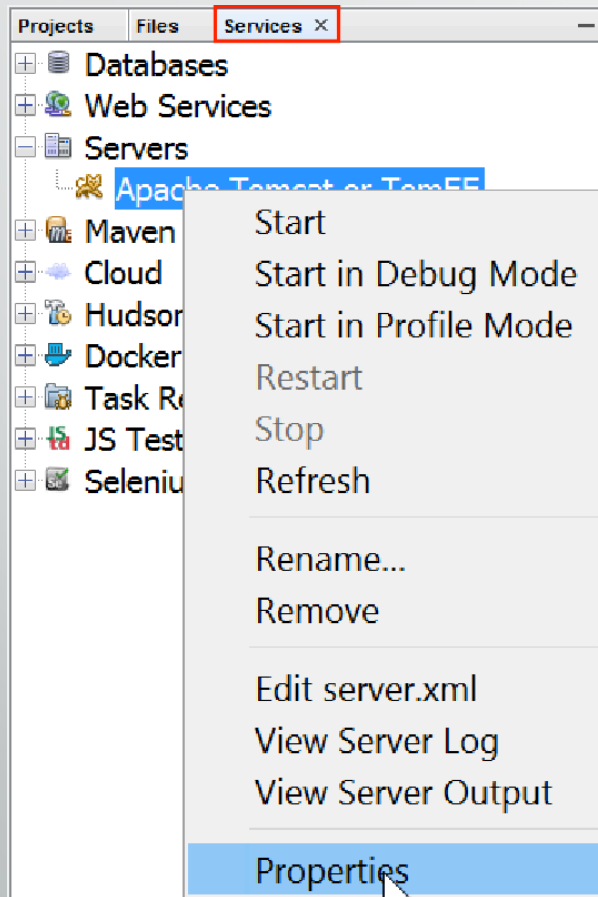
Password:

Create user if it does not exist

< Back Next > Finish Cancel Help

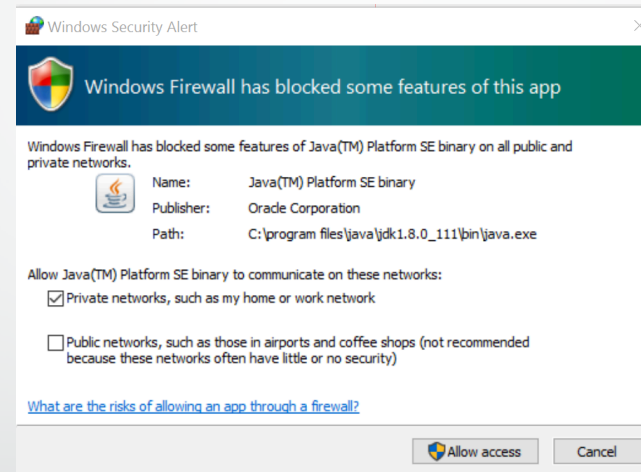
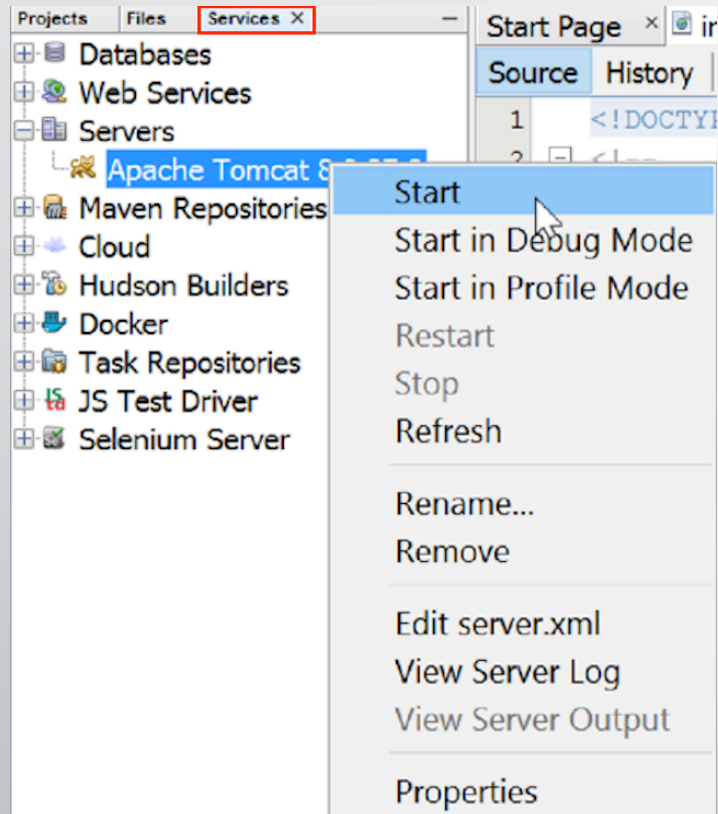
Configurare il Server Apache Tomcat

- Selezionare la voce *Services* -> Cliccare su *Properties*



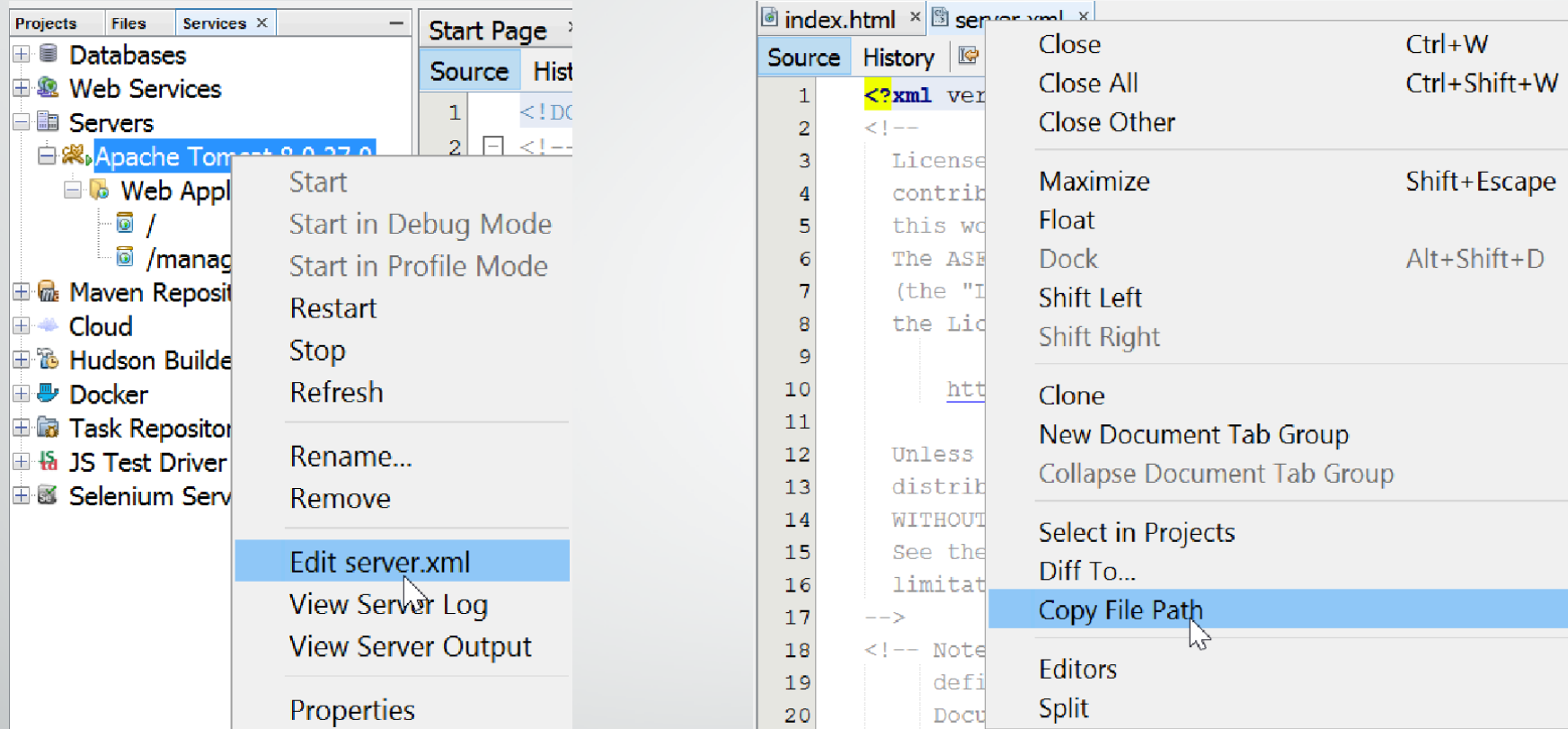
Lanciare il Server Apache Tomcat

- Selezionare la voce *Services* -> Cliccare su *Start*



- Permettere l'accesso in rete di *Java* e *Netbeans*.

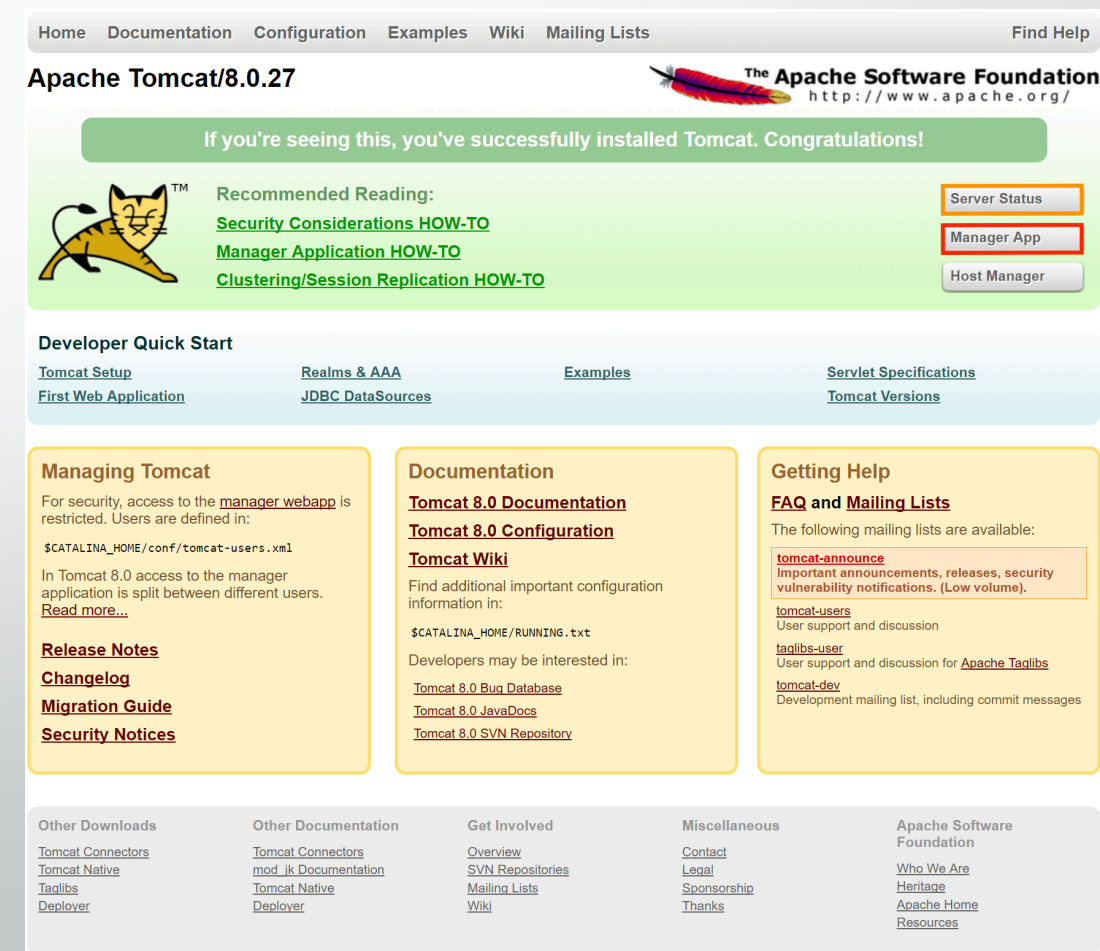
Configurare i ruoli d'accesso a Tomcat




- Aprire il file *tomcat-users.xml* al path precedentemente copiato
- Inserire la seguente riga in basso al file:
 - `<user username="..." password="..." roles="manager-script,admin-gui,manager-gui" />`

Accedere alla GUI di Apache Tomcat


- Aprire un browser e accedere al server locale di Tomcat all'indirizzo: *localhost:<server port>* (e.g., localhost:9090)
- **Server Status:** GUI per il monitoraggio delle risorse occupate dalla/e WA.
- **Manager App:** GUI per la visualizzazione dei dettagli di una sessione e/o per la rimozione di attributi/variabili di sessione.



Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/8.0.27  <http://www.apache.org/>

If you're seeing this, you've successfully installed Tomcat. Congratulations!

 **Recommended Reading:**
[Security Considerations HOW-TO](#)
[Manager Application HOW-TO](#)
[Clustering/Session Replication HOW-TO](#)

[Server Status](#)
[Manager App](#)
[Host Manager](#)

Developer Quick Start
[Tomcat Setup](#) [Realms & AAA](#) [Examples](#) [Servlet Specifications](#)
[First Web Application](#) [JDBC DataSources](#) [Tomcat Versions](#)

Managing Tomcat
For security, access to the [manager webapp](#) is restricted. Users are defined in:
\$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 8.0 access to the manager application is split between different users.
[Read more...](#)
[Release Notes](#)
[Changelog](#)
[Migration Guide](#)
[Security Notices](#)

Documentation
[Tomcat 8.0 Documentation](#)
[Tomcat 8.0 Configuration](#)
[Tomcat Wiki](#)
Find additional important configuration information in:
\$CATALINA_HOME/RUNNING.txt
Developers may be interested in:
[Tomcat 8.0 Bug Database](#)
[Tomcat 8.0 JavaDocs](#)
[Tomcat 8.0 SVN Repository](#)

Getting Help
[FAQ and Mailing Lists](#)
The following mailing lists are available:
[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).
[tomcat-users](#)
User support and discussion
[taglibs-user](#)
User support and discussion for [Apache Taglibs](#)
[tomcat-dev](#)
Development mailing list, including commit messages

Other Downloads
[Tomcat Connectors](#)
[Tomcat Native](#)
[Taglibs](#)
[Deployer](#)

Other Documentation
[Tomcat Connectors](#)
[mod_jk Documentation](#)
[Tomcat Native](#)
[Deployer](#)

Get Involved
[Overview](#)
[SVN Repositories](#)
[Mailing Lists](#)
[Wiki](#)

Miscellaneous
[Contact](#)
[Legal](#)
[Sponsorship](#)
[Thanks](#)

Apache Software Foundation
[Who We Are](#)
[Heritage](#)
[Apache Home](#)
[Resources](#)

Installare MySQL Server

1. Lanciare l'installer;
2. Scegliere come tipologia di installazione quale 'custom';
3. Scegliere come prodotti:
 1. MySQL Server 5.x
 2. Connector/J 5.x
4. Impostare la password per l'utente 'root';
5. Aggiungere (eventualmente) un nuovo utente con relativa password;
6. Applicare/eseguire le configurazioni impostate.

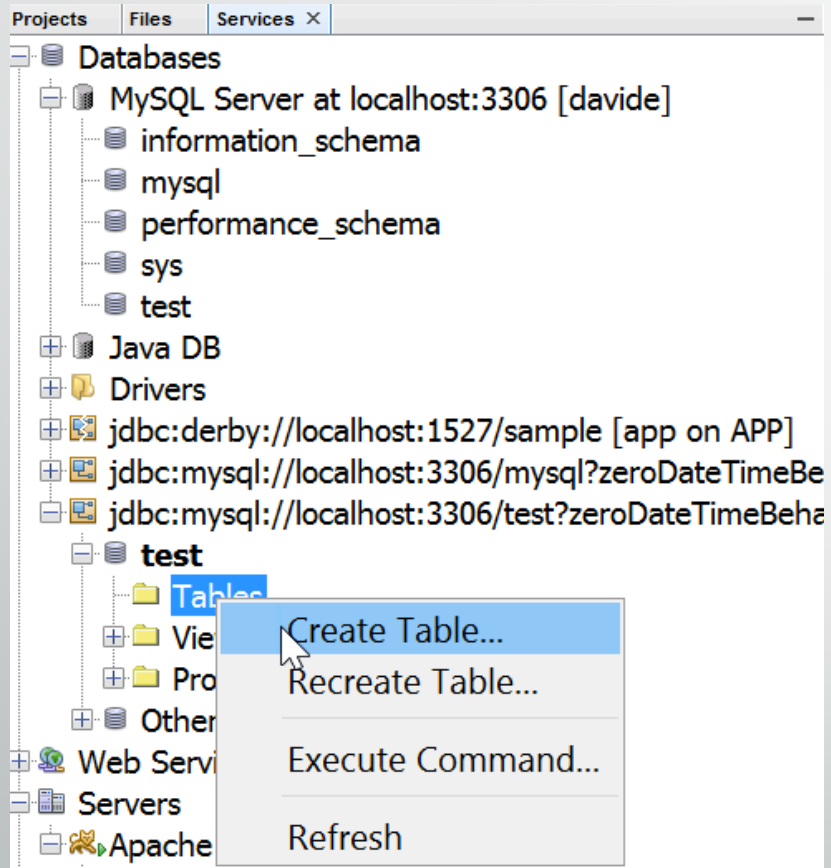
Nota: In alternativa allo stesso modo di *MySQL Server* e' possibile utilizzare la versione di *JavaDB* pre-installata in Netbeans.

Configurazione Database Netbeans

- Una volta scaricato e installato MySQL Server, effettuare i seguenti steps:
 1. *Services -> Register MySQL Server;*
 2. Inserire il nome dell'utente e password amministratore impostati nella fase d'installazione;
 3. Cliccare *Admin Properties* ed inserire il path dell'eseguibile di MySQL (mysqld.exe) alla voce *Path to start command*;
 4. Connettersi al database, creare un database di test e concedere all'utente tutti i permessi al db appena creato.
 5. Infine connettersi al database.

Configurazione Database Netbeans (cont.)

- Creare una nuova tabella con gli attributi sotto riportati



Attributi

Name <VARCHAR>

Surname <VARCHAR>

City <VARCHAR>

University <VARCHAR>

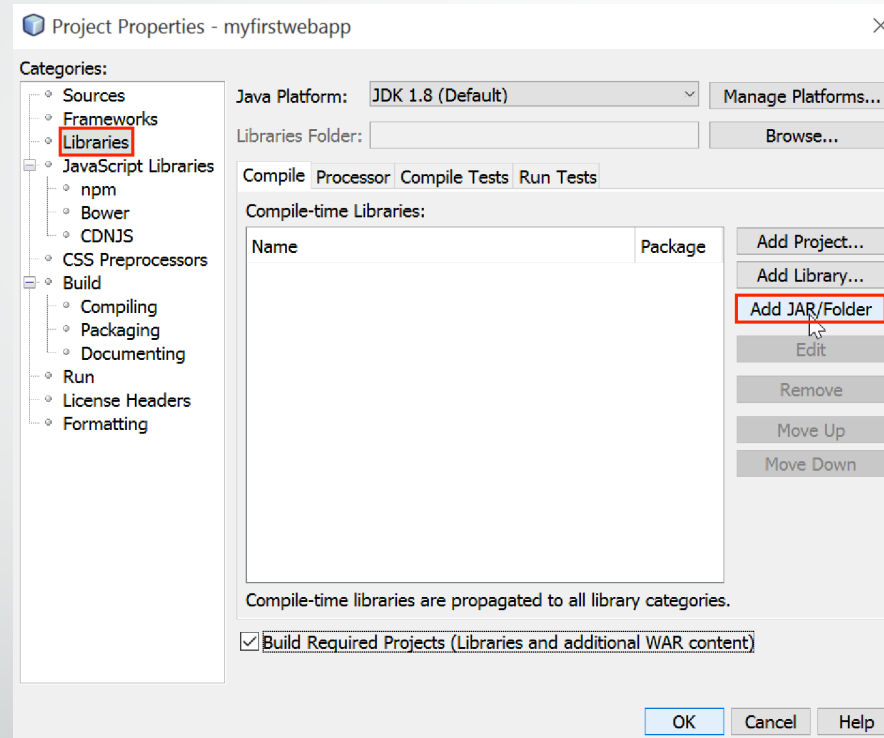
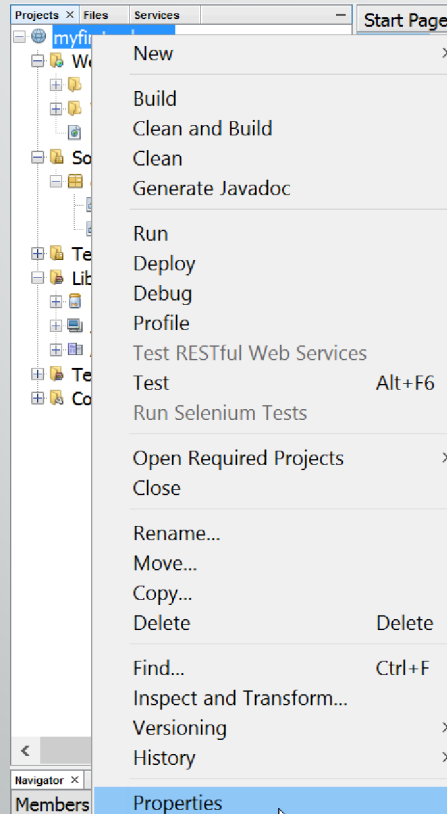
Phone <VARCHAR>

Mail <VARCHAR>

Password <VARCHAR>

Configurazione Database Netbeans (cont.)

- Aggiungere i driver JDBC al progetto



- I driver JDBC Mysql sono al path:

C:\Program Files\NetBeans 8.2\ide\modules\ext\mysql-connector-java-5.1.23-bin.jar

Esempio di Registrazione e Login

- In questo semplice esempio vedremo come gestire:
 1. Registrazione utente
 2. Login utente
 3. Visualizzazione utenti
- Sara' creata:
 - Una classe Database per effettuare semplici operazioni di *connessione* al db, *inserimento* e *reperimento* dati da esso.
 - Diverse pagine JSP per la gestione dell'interfaccia grafica della WA.
 - Infine alcune *Servlet* per l'elaborazione dei dati ricevuti dal front-end.

Singleton Class Database (1)

1. Creare un proprio *package* (e.g., *classes*) nella cartella *Source Packages*
2. Creare la Classe *Database* nel package appena creato

```
package classes;  
import java.sql.*;  
import java.util.logging.*;
```

```
public class Database {  
    //static reference to itself  
    private static Database dbInstance;  
    public static final String URL = "jdbc:mysql://localhost/test";  
    public static final String USER = "<tuoi_utente>";  
    public static final String PASSWORD = "<tua_password>";  
    public static final String DRIVER_CLASS = "com.mysql.jdbc.Driver";  
    private static Connection con ;  
    private static Statement stmt;  
  
    private Database() { /* private constructor */ }
```

Singleton Class Database (2)

```
public static Database getInstance(){
    if(dbsInstance == null) {
        dbsInstance = new Database();
    }
    return dbsInstance;
}

public Connection getConnection(){
    if(con==null) {
        try {
            Class.forName(DRIVER_CLASS);
            con = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException | ClassNotFoundException ex) {
            Logger.getLogger(Database.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return con;
}
}
```

JSP per SignUp

```
<%  
    session = request.getSession();  
    String flash_message = (String) session.getAttribute("flash_message");  
%>
```

```
</head>
```

```
<div class="container">
```

```
  <div class="row">
```

```
    <% if (flash_message=="1") {%>  
      <div class="alert alert-success text-center">  
        <b>Registrazione effettuata con successo!</b>  
        <button type="button" class="close" data-dismiss="alert"></button>  
      </div>  
    <%}%>
```

```
<form id="register-form" action="SignUp" method="post" role="form" style="display: none;">
```

```
  <div class="form-group">
```

```
    <input type="text" name="name" id="name" tabindex="1" class="form-control" placeholder="Name" value="">
```

```
  </div>
```

```
  ...
```

```
  <div class="form-group">
```

```
    <div class="row">
```

```
      <div class="col-sm-6 col-sm-offset-3">
```

```
        <input style="background-color: #029f5b" type="submit" name="register-submit" id="register-submit" tabindex="4"  
        class="form-control btn btn-register" value="Register Now">
```

```
      </div>
```

```
    </div>
```

```
  </div>
```

```
  ...
```

```
</form>
```

Servlet Class SignUp

```
public class SignUp extends HttpServlet {

    private HttpSession session;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try {
            session = request.getSession();
            String name = request.getParameter("name");
            ...
            String password = request.getParameter("password");
            ArrayList<String> credential = new ArrayList<String>();
            credential.add(name);
            ...
            credential.add(password);

            Connection con = Database.getInstance().getConnection("test");
            Database.getInstance().insertRow("users_db", credential);

            session.setAttribute("flash_message", "1");
            request.getRequestDispatcher("/first_page.jsp").forward(request, response);

        } catch (SQLException ex) {
            Logger.getLogger(SignUp.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

JSP per Login

```
<%  
    session = request.getSession();  
    String flash_message = (String) session.getAttribute("flash_message");  
%>
```

```
</head>
```

```
<div class="container">
```

```
  <div class="row">
```

```
    <% if (flash_message=="o") {%>  
      <div class="alert alert-success text-center">  
        <b>Nome utente o password errata!</b>  
        <button type="button" class="close" data-dismiss="alert"></button>  
      </div>  
    <%}%>
```

```
<form id="register-form" action="CheckUser" method="post" role="form" style="display:block;">
```

```
  <div class="form-group">
```

```
    <input type="text" name="mail" id="name" tabindex="1" class="form-control" placeholder="mail" value="">
```

```
  </div>
```

```
  <div class="form-group">
```

```
    <input type="password" name="password" id="password" tabindex="1" class="form-control" placeholder="Password" value="">
```

```
  </div>
```

```
  <div class="form-group">
```

```
    <div class="row">
```

```
      <div class="col-sm-6 col-sm-offset-3">
```

```
        <input style="background-color: #029f5b" type="submit" name="login-submit" id="login-submit" tabindex="4" class="form-control btn btn-login" value="Log In">
```

```
      </div>
```

```
    </div>
```

Servlet Class Check User

```
public class Check User extends HttpServlet {

    private HttpSession session;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        try {

            session = request.getSession();
            String mail = request.getParameter("mail");
            String password = request.getParameter("password");
            Database.getInstance().getConnection("test");

            if (Database.getInstance().checkUser(mail, password) == 1) {
                Map<String, String> info_user = Database.getInstance().getInfo("users", "mail", mail);
                session.setAttribute("info_user", info_user);
                getServletContext().getRequestDispatcher("/profile.jsp").forward(request, response);
            }
            else{
                session.setAttribute("flash_message", "o");
                getServletContext().getRequestDispatcher("/first_page.jsp").forward(request, response);
            }
        } catch (SQLException ex) {
            Logger.getLogger(CheckUser.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

JSP per Profile

```
<% session = request.getSession();  
    HashMap<String, String> info_user = (HashMap<String, String>) session.getAttribute("info_user");  
%>
```

```
</head>
```

```
<div class="container">
```

```
...
```

```
<div class="row">
```

```
<div class="col-sm-6 col-md-4">
```

```

```

```
</div>
```

```
<div class="col-sm-6 col-md-8">
```

```
<h4><% out.print(info_user.get("name") + " " + info_user.get("surname")); %></h4>
```

```
<small><cite title="San Francisco, USA"><%out.print(info_user.get("city"));%> ITA <i class="glyphicon glyphicon-map-marker"></i></cite></small>
```

```
<p>
```

```
<i class="glyphicon glyphicon-phone-alt"></i><%out.print(info_user.get("phone"));%>
```

```
<br/>
```

```
<i class="glyphicon glyphicon-home"></i><%out.print(info_user.get("university"));%>
```

```
<br/>
```

```
<i class="glyphicon glyphicon-envelope"></i><%out.print(info_user.get("mail"));%>
```

```
<br/>
```

```
<!-- Split button -->
```

```
<div class="btn-group">
```

```
<button type="button" class="btn btn-primary">View all users</button>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
...
```

```
</div>
```

JSP per user_table_page

```
<%  
    ArrayList<Map<String, String>> data = Database.getInstance().getAllRows("users");  
%>  
</head>
```

```
<div id="wrapper">  
    <h1>User Table</h1>  
    <table id="keywords" cellspacing="0" cellpadding="0">  
        <thead>  
            <tr>  
                <th><span>Name</span></th>  
                <th><span>Surname</span></th>  
                <th><span>City</span></th>  
                <th><span>University</span></th>  
                <th><span>Phone</span></th>  
                <th><span>Mail</span></th>  
            </tr>  
        </thead>  
        <tbody>
```

```
<%  
    for (int i = 0; i < data.size(); i++) {  
        out.print("<tr>");  
        out.print("<td class=lalign>" + data.get(i).get("name") + "</td>");  
        out.print("<td>" + data.get(i).get("surname") + "</td>");  
        out.print("<td>" + data.get(i).get("city") + "</td>");  
        out.print("<td>" + data.get(i).get("university") + "</td>");  
        out.print("<td>" + data.get(i).get("phone") + "</td>");  
        out.print("<td>" + data.get(i).get("mail") + "</td>");  
        out.print("</tr>");  
    }  
%>
```

```
</tbody>
```

```
</table>
```

```
</div>
```


Esercizio

- Implementare un classe Servlet per la gestione del LogOut di un utente
 - Rimozione della sessione
 - Redirezione alla pagina di login

WEB-INF: file di configurazione

web.xml: descrittore di distribuzione della WA.

- Descrive le servlet e gli altri componenti che compongono la WA.
- Utilizzato per l'impostazione dei parametri d'inizializzazione e per i vincoli di sicurezza che si vuole fare rispettare alla WA.

Note: *nel caso in cui tale file non sia disponibile nella cartella WEB-INF al path: **web/WEB-INF** del progetto, e' possibile reperirlo dal materiale messo a disposizione sulla piattaforma modificandolo appropriatamente.*

classes: cartella contenente i file delle classi Java richieste dalla WA, che include entrambe le classi Servlet e non-Servlet.

lib: cartella dei file JAR, contenenti i file delle classi JAVA richieste per la WA.

- Librerie di classi di terze parti e/o driver JDBC.

WEB-INF: file di configurazione (cont.)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd">
  <display-name>MyFirstWebApp</display-name>
  <welcome-file-list>
    <welcome-file>first_page.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>SignUp</servlet-name>
    <servlet-class>servlets.SignUp</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SignUp</servlet-name>
    <url-pattern>/SignUp</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>10</session-timeout>
  </session-config>
</web-app>
```

Tomcat Web Application Manager

- **Path:** URL della WA
- **Sessions:** numero di sessioni della WA create
- **Commands:** sezione per la gestione della WA

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	3	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 30 minutes
<u>/myfirstwebapp</u>	None specified	MyFirstWebApp	true	1	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> Expire sessions with idle ≥ 10 minutes

Tomcat Web Application Manager

- **WAR file to deploy:** sezione per l'upload della WA

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

Select WAR file to upload Nessun file selezionato

Diagnostics

Check to see if a web application has caused a memory leak on stop, reload or undeploy

This diagnostic check will trigger a full garbage collection. Use it with extreme caution on production systems.

SSL connector configuration diagnostics

List the configured ciphers for each connector

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/8.0.27	1.8.0_111-b14	Oracle Corporation	Windows 10	10.0	amd64	DESKTOP-GCI99R8	192.168.123.128

Riferimenti

- **Apache Tomcat, JavaBeans e JDBC**

1. https://www.ntu.edu.sg/home/ehchua/programming/howto/Tomcat_HowTo.html
(Windows and Unix)
2. <https://wolfpaulus.com/journal/mac/tomcat8/> (OS X)
3. <https://www.mulesoft.com/tcat/tomcat-linux> (Linux)
4. https://www.tutorialspoint.com/jsp/jsp_java_beans.htm (JavaBeans)
5. <http://www.java-samples.com/showtutorial.php?tutorialid=552> (JavaBeans)
6. <http://theopentutorials.com/tutorials/java/jdbc/jdbc-examples-introduction/> (JDBC example)
7. <https://netbeans.org/kb/docs/ide/java-db.html> (Java DB Derby)

Contatti:

davide.nardone@studenti.uniparthenope.it

“Laboratorio di Architettura e Sistemi” IV piano lato NORD, stanza 432.