Aplicaţii Web cu Servleţi şi pagini JSP

Se considerã cã existã un folder cu distributia apache-tomcat; vom numi acest director "**tomcat7**" (în documentația Apache este valoarea variabilei de mediu CATALINA_HOME). Această distribuție poate fi obținută gratuit de la adresa http://tomcat.apache.org/download-70.cgi sau http://tomcat.apache.org/download-80.cgi sau http://tomcat.apache.org/download-90.cgi.
Exemplele inițiale din versiunea 7 funcționează și pe versiunile ulterioare.

1. Utilizarea unui servlet simplu

Creaţi un director (folder) pentru aplicatie cu numele "hello1".
În acest director creați un fișier sursă numit "HelloServlet.java" cu următorul conținut:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println ("<html>");
      out.println ("<body>");
      out.println ("<h1>Hello World!</h1>");
      out.println ("</body>");
      out.println ("</html>");
  }
}
```

Compilați fișierul "HelloServlet.java" cu biblioteca "servlet-api.jar" din "tomcat7\lib". Intr-un IDE trebuie adaugata biblioteca respectiva la cãile unde se cautã bibliotecile de clase. In linie de comandã se compileazã astfel (**tomcat7** se inlocuieste cu calea corespunzatoare instalarii):

```
javac -cp c:\tomcat7\lib\servlet-api.jar HelloServlet.java
```

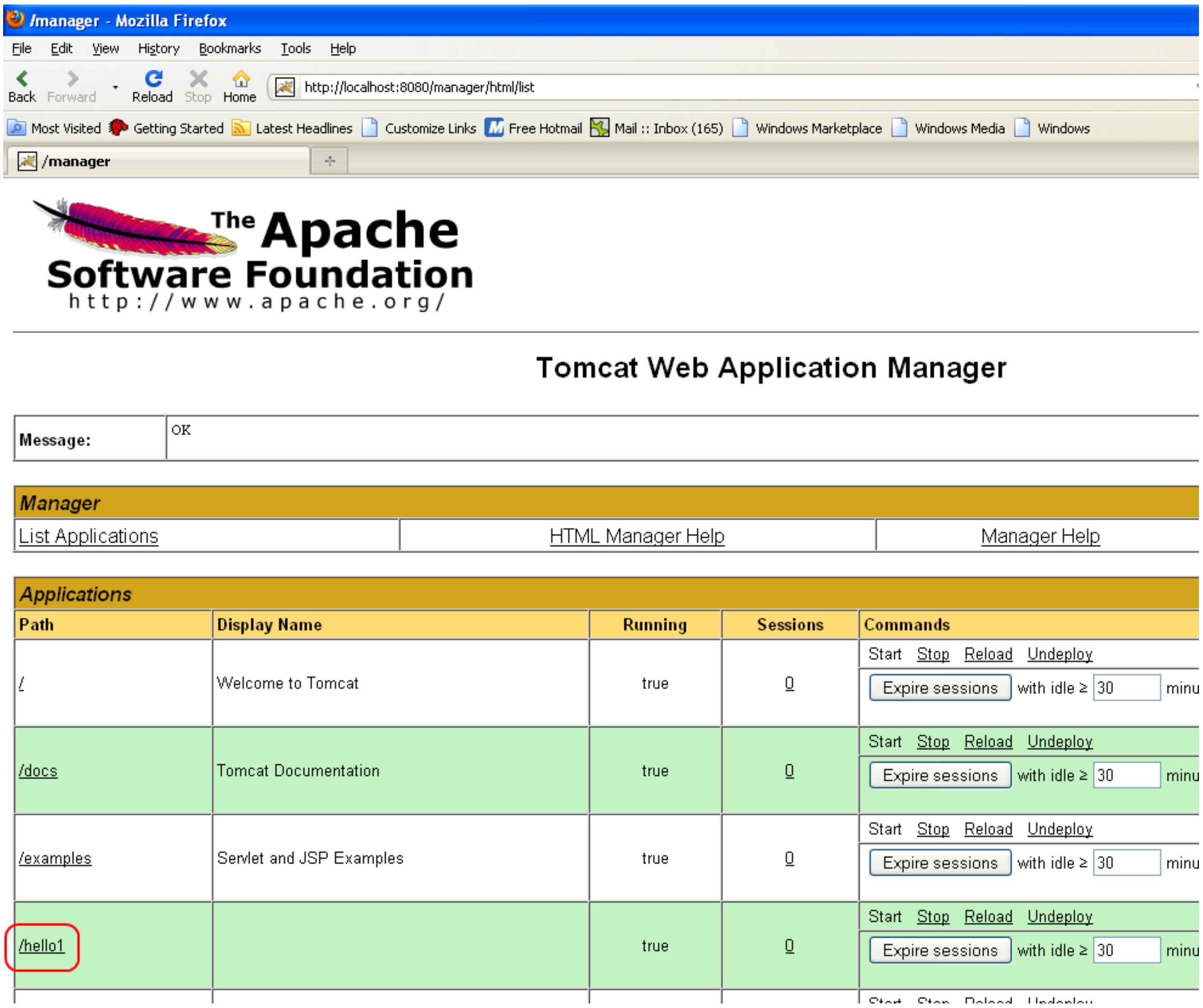
Creaţi subdirectorul "WEB-INF" în directorul "hello1".
Creaţi subdirectorul hello1/WEB-INF/classes si copiati (sau mutati) în el fișierul "HelloServlet.class".
În subdirectorul hello1/WEB-INF Creați fisierul "web.xml" cu urmatorul continut:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

Copiati directorul "hello1" in directorul tomcat7\webapps ("deployment").
Intr-un browser Web introduceti urmatorul URL:

```
http://localhost:8080/hello1/hello
```

Aplicatia este vizibila si in consola de administrare Tomcat:


/manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop Home
http://localhost:8080/manager/html/list
Most Visited Getting Started Latest Headlines Customize Links Free Hotmail Mail :: Inbox (165) Windows Marketplace Windows Media Windows
/manager
The Apache Software Foundation http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minu
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minu
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minu
/hello1		true	0	Start Stop Reload Undeploy; Expire sessions with idle ≥ 30 minu

Modificati textul afisat in "Salut lume!" (in clasa servlet) si refaceti toate operatiile necesare pentru ca sa se afiseze acest mesaj la client (cu acelasi URL).

2. Extragere parametri dintr-o cerere

Creați un director (folder) pentru aplicație cu numele "hello2".
In acest director Creați un fișier sursa numit "ParamServlet.java" cu următorul conținut:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ParamServlet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws
  ServletException, IOException {
    String first,last;
    first=request.getParameter("first");
    last=request.getParameter("last");
    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    if (first != null || last != null) {
      out.println ("<html>");
      out.println ("<body>");
      out.print ("<hr><h1>");
      out.print ("Hello " + first + " " + last + "!");
      out.println ("</h1><hr>");
      out.println ("</body>");
      out.println ("</html>");
    } else
        out.println("No Parameters, Please enter some");
  }
}
```

Compilati fisierul "ParamServlet.java" cu biblioteca "servlet-api.jar" din "**tomcat7**\lib".
Creați subdirectorul "WEB-INF" in directorul "hello2".
Creați subdirectorul hello2/WEB-INF/classes si copiati (sau mutati) în el fisierul "ParamServlet.class".
In subdirectorul hello2/WEB-INF copiati fisierul "web.xml" din directorul "hello1" si modificati numele de servlet si de clasa in "ParamServlet" (in loc de "HelloServlet").
Copiati directorul "hello2" in directorul tomcat7\webapps.
Intr-un browser Web introduceti urmatorul URL:

```
http://localhost:8080/hello2/hello?first=good&last=student
```

3. Servlet cu afisare formular si preluare parametri din formular

Creați un director (folder) pentru aplicatie cu numele "hello3".
In acest director Creați un fisier sursa numit "FormServlet.java" cu urmatorul continut:

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FormServlet extends HttpServlet {
    String first,last;
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException  {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Request Parameters Example</title>");
        out.println("</head>");
        out.println("<body>");
        out.print("<form method=\"get\">");
        out.println("<p>");
        out.println("First Name:");
        out.println("<input type=text size=20 name=firstname>");
        out.println("<br>");
        out.println("Last Name:");
        out.println("<input type=text size=20 name=lastname>");
        out.println("<br>");
        out.println("<input type=submit value=\"Submit\">");
        out.println("</form>");
        first=request.getParameter("firstname");
        last=request.getParameter("lastname");
```

```
        if (first == null && last == null) {
          out.println("No parameters, please enter some!");
        } else {
            if (first.length() == 0 && last.length() == 0) {
              out.println("Empty parameters, please enter values!");
            } else {
                out.print("<h1>");
                out.print("Hello " + first + " " + last + "!");
                out.println("</h1>");
              }
           }
        out.println("</body>");
        out.println("</html>");
    }
}
```

Compilati fisierul “FormServlet.java” cu biblioteca “servlet-api.jar” din “**tomcat7**\lib”.
Creați subdirectorul “WEB-INF” in directorul “hello3”.
Creați subdirectorul hello3/WEB-INF/classes si copiati în el fisierul “FormServlet.class”.
In subdirectorul hello3/WEB-INF copiati fisierul “web.xml” din directorul “hello1” si modificati numele de servlet si de clasa in “FormServlet” (in loc de “HelloServlet”).
Copiati directorul “hello3” in directorul tomcat7\webapps.
Intr-un browser Web introduceti urmatorul URL:

```
http://localhost:8080/hello3/hello
```

4. Pagina JSP pentru o simplă afișare la client

Creați un folder cu numele “hello4”.
Creați un fisier numit “index.jsp” cu urmatorul continut:

```
<%@ page info="A hello world example" %>
<html>
<head><title>Hello, World</title></head>
<body bgcolor="#ffffff">
  <table border="2" cellpadding="0" cellspacing="0">
    <tr valign="middle">
      <td width="150" bgcolor="lightgreen"> &nbsp; </td>
      <td width="250" height="200" align="right"> <h1>Hello, World!</h1> </td>
    </tr>
  </table>
</body>
</html>
```

Copiati directorul “hello4” in subdirectorul “tomcat7/webapps”.
Intr-un browser Web introduceti urmatorul URL:

```
http://localhost:8080/hello4
```

Modificați textul afișat in “Salut lume !” si copiati fisierul “index.jsp” in tomcat7/webapps/hello4.
Reafișați pagina din browser (“page refresh”) si observati efectul.
Modificați numele paginii JSP din “index.jsp” in “hello.jsp” si lansați din nou aplicația:

```
http://localhost:8080/hello4/hello.jsp
```

5. Pagini JSP care folosesc clase JavaBeans

Creați un folder cu numele “hello5”.
Creați in “hello5” un fisier numit “index.jsp” cu urmatorul continut:

```
<%@ page import="beans.NameHandler" %>
<jsp:useBean id="mybean" scope="page" class="beans.NameHandler" />
<jsp:setProperty name="mybean" property="*" />
<html>
<head><title>Hello, Bean User!</title></head>
<table border="0" width="700">
  <tr>
    <td width="150"> &nbsp; </td>
    <td width="550">
      <h2>Your name please ?</h2>
    </td>
  </tr>
  <tr>
    <td width="150" &nbsp; </td>
    <td width="550">
      <form method="get">
        <input type="text" name="username" size="25">
        <br>
        <input type="submit" value="Submit">
      </form>
    </td>
  </tr>
</table>
<%
    String name= request.getParameter("username");
    if ( name==null || name.trim().length()>0 ) {
%>

<%@ include file="response.jsp" %>

<%
    } else {
%>
<h2>No name, try again!</h2>
<% } %>
</body>
</html>
```

Creați in “hello5” un fisier “response.jsp” cu urmatorul continut:

```
<table border="0" width="700">
  <tr>
    <td width="150"> &nbsp; </td>
    <td width="550">
      <h1>Hello, <jsp:getProperty name="mybean" property="username" />!</h1>
    </td>
  </tr>
</table>
```

Creați in “hello5” un fisier “NameHandler.java” cu următorul conținut:

```
package beans;
public class NameHandler {
    private String username;
    public NameHandler() { username = null;  }
    public void setUsername( String name ) { username = name; }
    public String getUsername() { return username; }
}
```

Compilati fisierul java folosind un IDE sau prin comanda:

```
javac –d . NameHandler.java
```

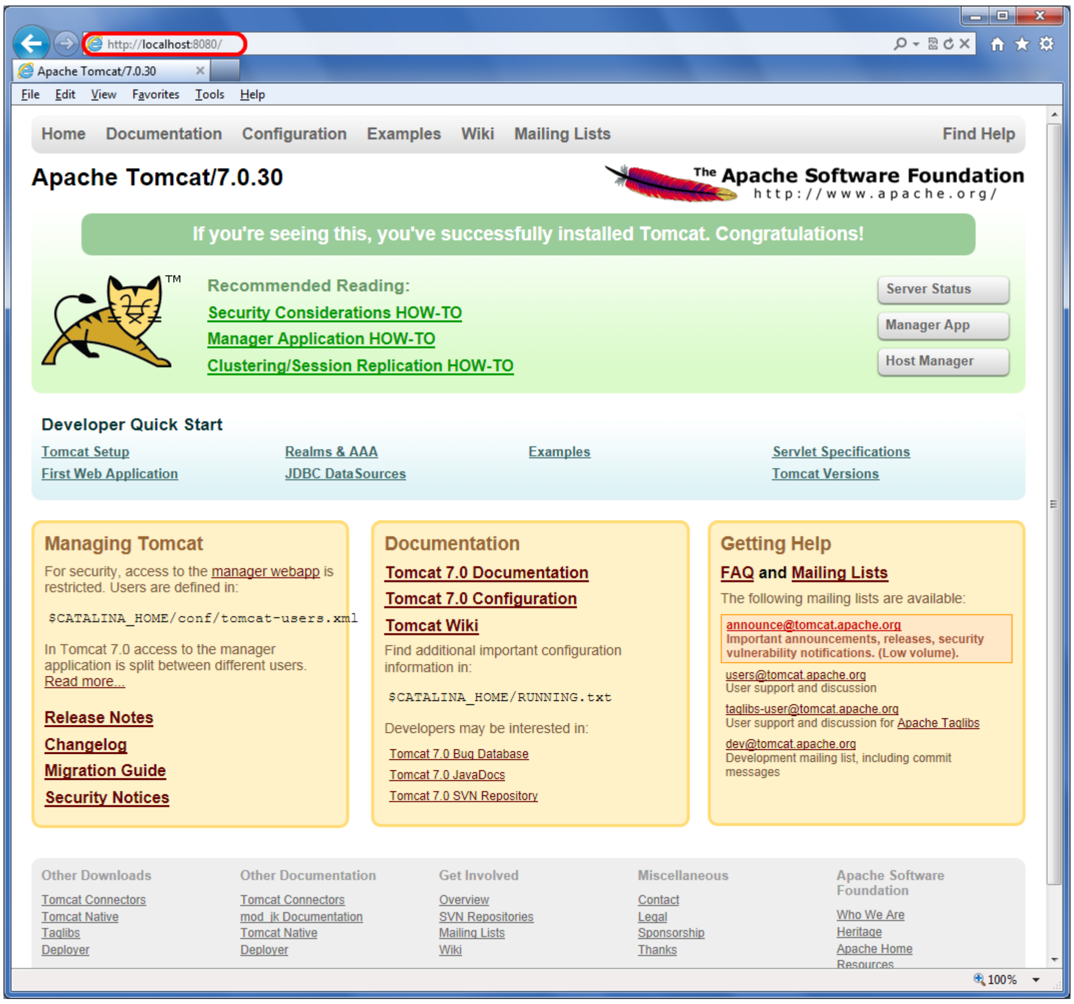
(Se va crea un subdirector “beans” cu un fisier “NameHandler.class”).
Mutati directorul “beans” in hello5/WEB-INF/classes.
Copiati directorul “hello5” in subdirectorul “tomcat7/webapps”.
Intr-un browser Web introduceti urmatorul URL:

```
http://localhost:8080/hello5
```

6. Exemple

Cu serverul Tomcat pornit, introduceti intr-un browser urmatorul URI:

```
http://localhost:8080/examples
```


http://localhost:8080/
Apache Tomcat/7.0.30
File Edit View Favorites Tools Help
Home Documentation Configuration Examples Wiki Mailing Lists Find Help
The Apache Software Foundation http://www.apache.org/
Server Status
Manager App
Host Manager
100%

Apache Tomcat/7.0.30

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

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 7.0 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation
Tomcat 7.0 Documentation
Tomcat 7.0 Configuration
Tomcat Wiki
Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:
Tomcat 7.0 Bug Database
Tomcat 7.0 JavaDocs
Tomcat 7.0 SVN Repository

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

Other Downloads	Other Documentation	Get Involved	Miscellaneous	Apache Software Foundation
Tomcat Connectors	Tomcat Connectors	Overview	Contact	Who We Are
Tomcat Native	mod_jk Documentation	SVN Repositories	Legal	Heritage
Taglibs	Tomcat Native	Mailing Lists	Sponsorship	Apache Home
Deployer	Deployer	Wiki	Thanks	Resources

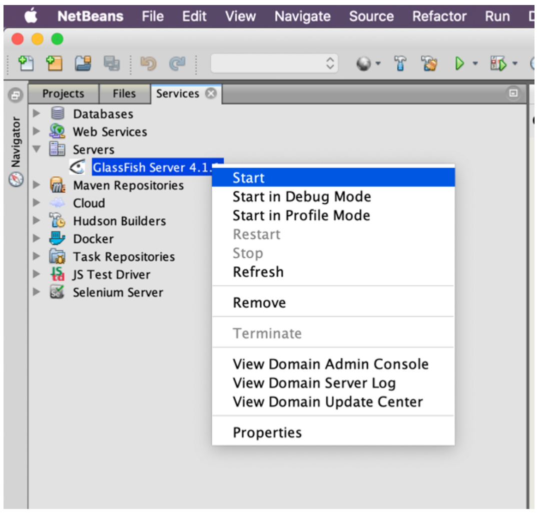
Executati exemplele si examinati sursele acestor exemple (servleti si pagini JSP).
Folosind ca model operatiile prezentate anterior Creați o aplicatie Web cu un servlet din exemple (de exemplu “sessions”) si apoi o aplicatie cu o pagina JSP din exemple (de exemplu “Calendar” din JSP 1.2 Examples).

NetBeans şi GlassFish

Se considerã cã mediul de dezvoltare NetBeans .

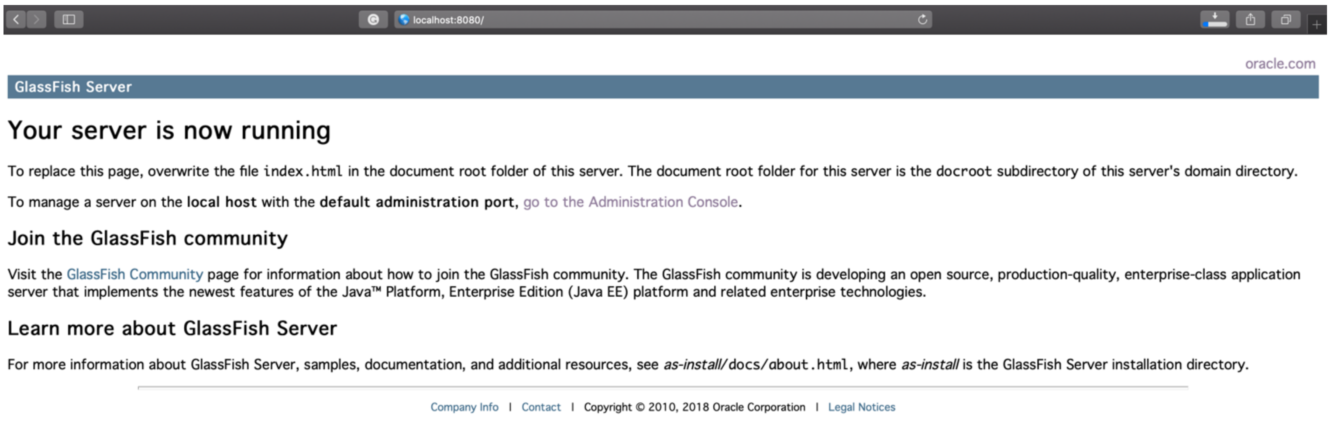
1. Utilizarea serverului GlassFish incorporat

Pentru simplificarea procesului de dezvoltare si testare, NetBeans contine un server de aplicatii propriu, care poate fi folosit pentru deploymentul aplicatiilor in curs de dezvoltare.
Porniti serverul GlassFish:



Dupa ce serverul a pornit, accesati URL-ul implicit:

```
http://localhost:8080/
```


localhost:8080/

oracle.com

GlassFish Server

Your server is now running

To replace this page, overwrite the file `index.html` in the document root folder of this server. The document root folder for this server is the `docroot` subdirectory of this server's domain directory.

To manage a server on the **local host** with the **default administration port**, go to the Administration Console.

Join the GlassFish community

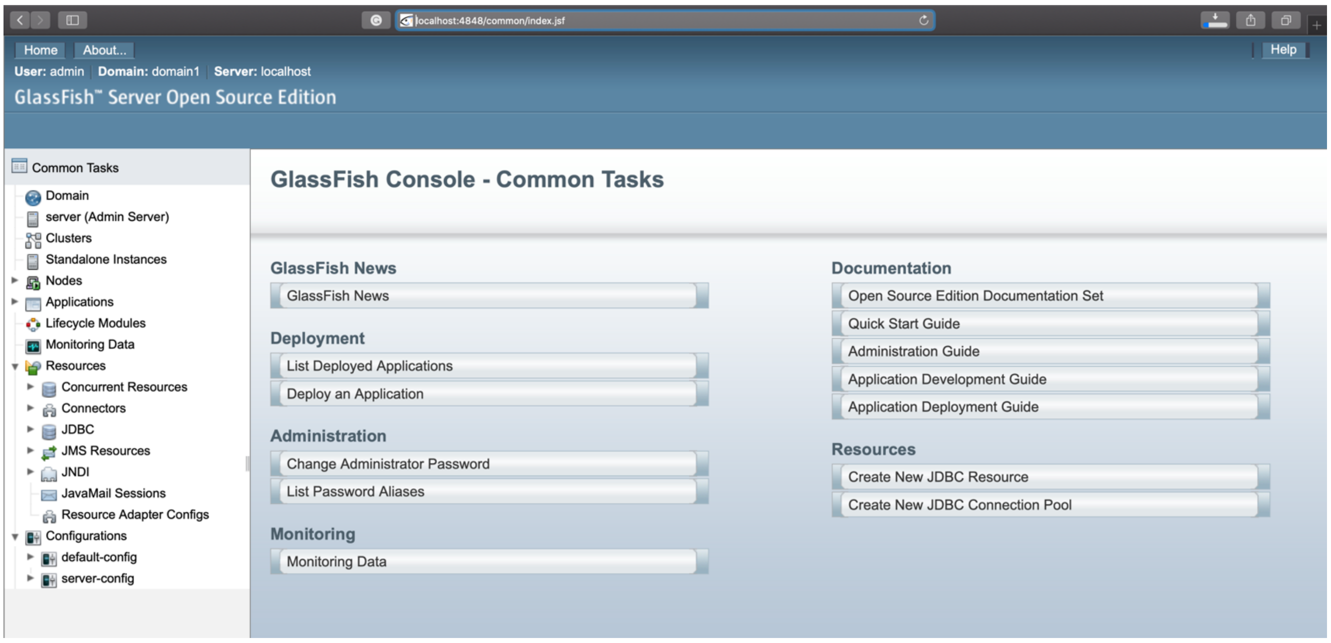
Visit the GlassFish Community page for information about how to join the GlassFish community. The GlassFish community is developing an open source, production-quality, enterprise-class application server that implements the newest features of the Java™ Platform, Enterprise Edition (Java EE) platform and related enterprise technologies.

Learn more about GlassFish Server

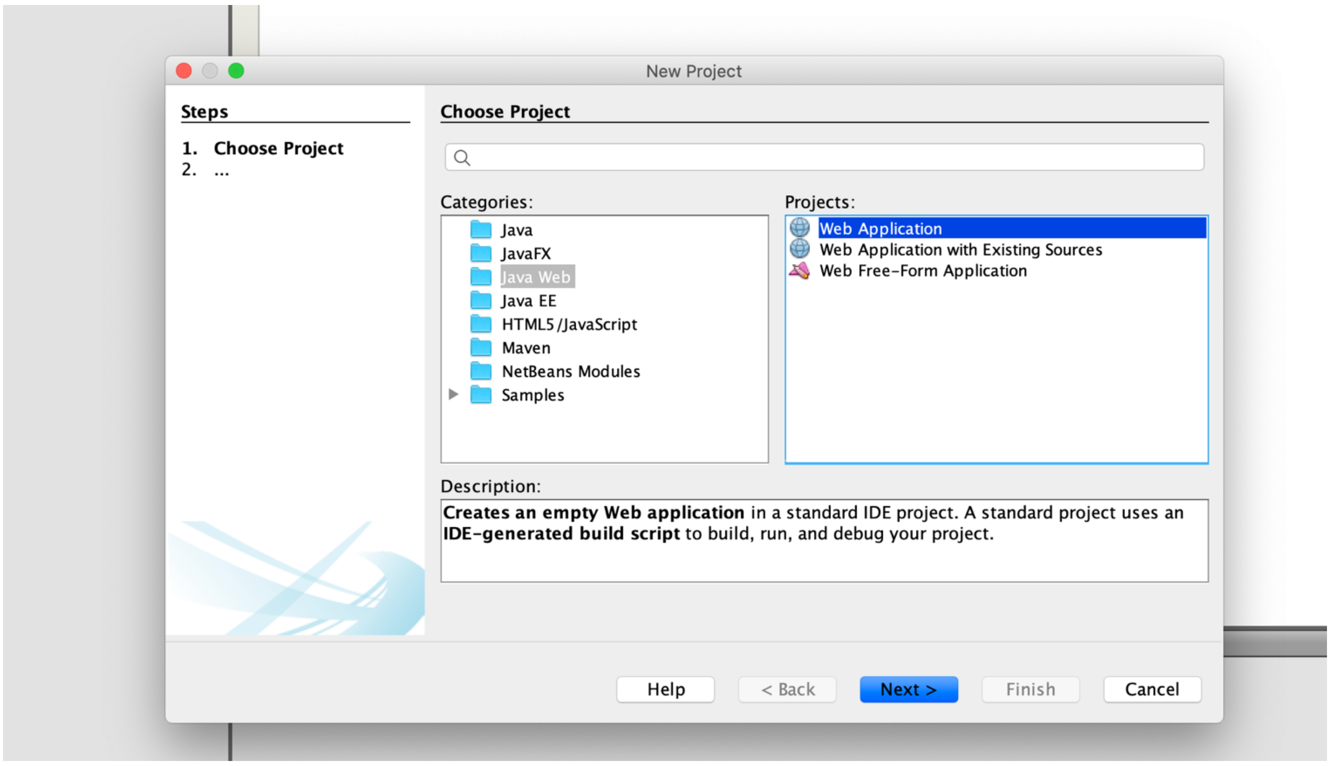
For more information about GlassFish Server, samples, documentation, and additional resources, see *as-install*/`docs/about.html`, where *as-install* is the GlassFish Server installation directory.

Company Info | Contact | Copyright © 2010, 2018 Oracle Corporation | Legal Notices

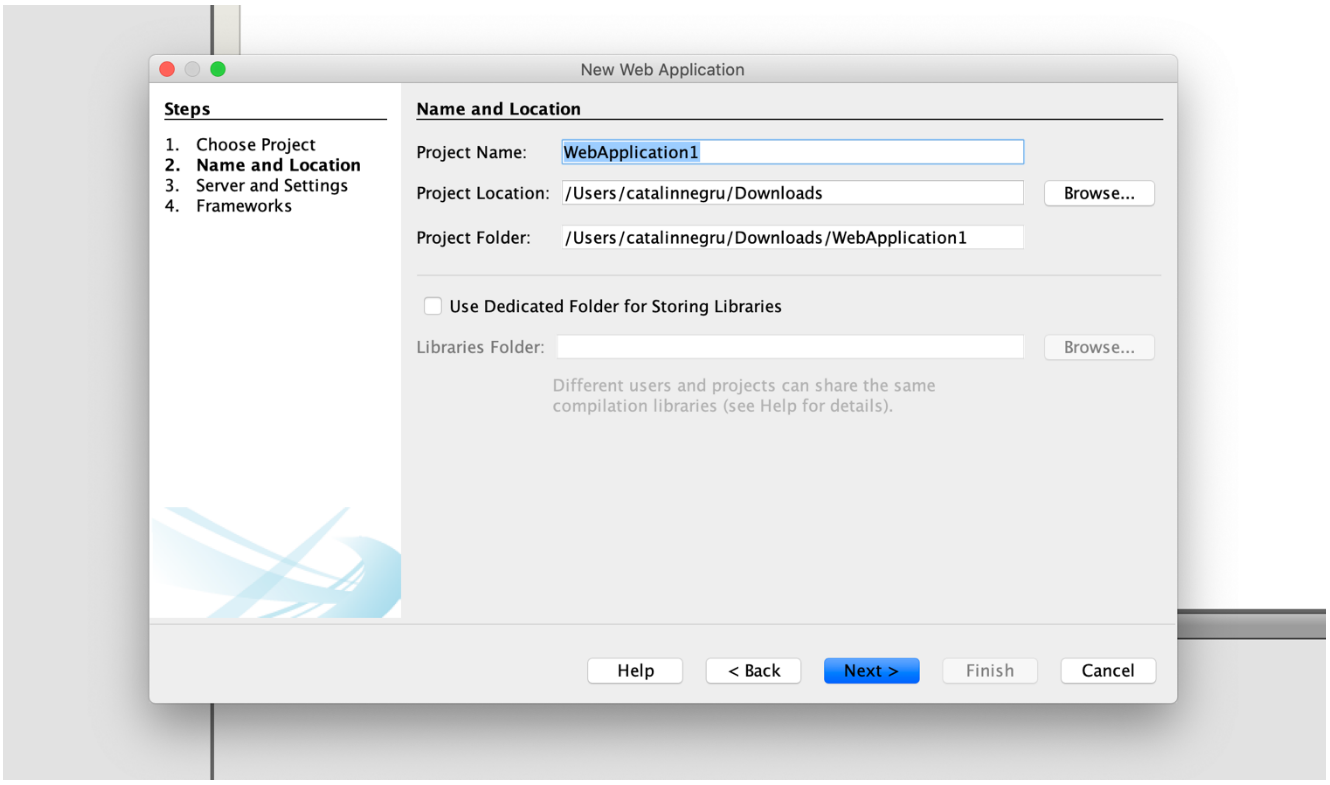
Accesati consola de administrare:



Creați o aplicatie Web simpla, folosind optiunile implicite (nu e obligatoriu ca serverul sa fie pornit):



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.



Project Name: WebApplication1

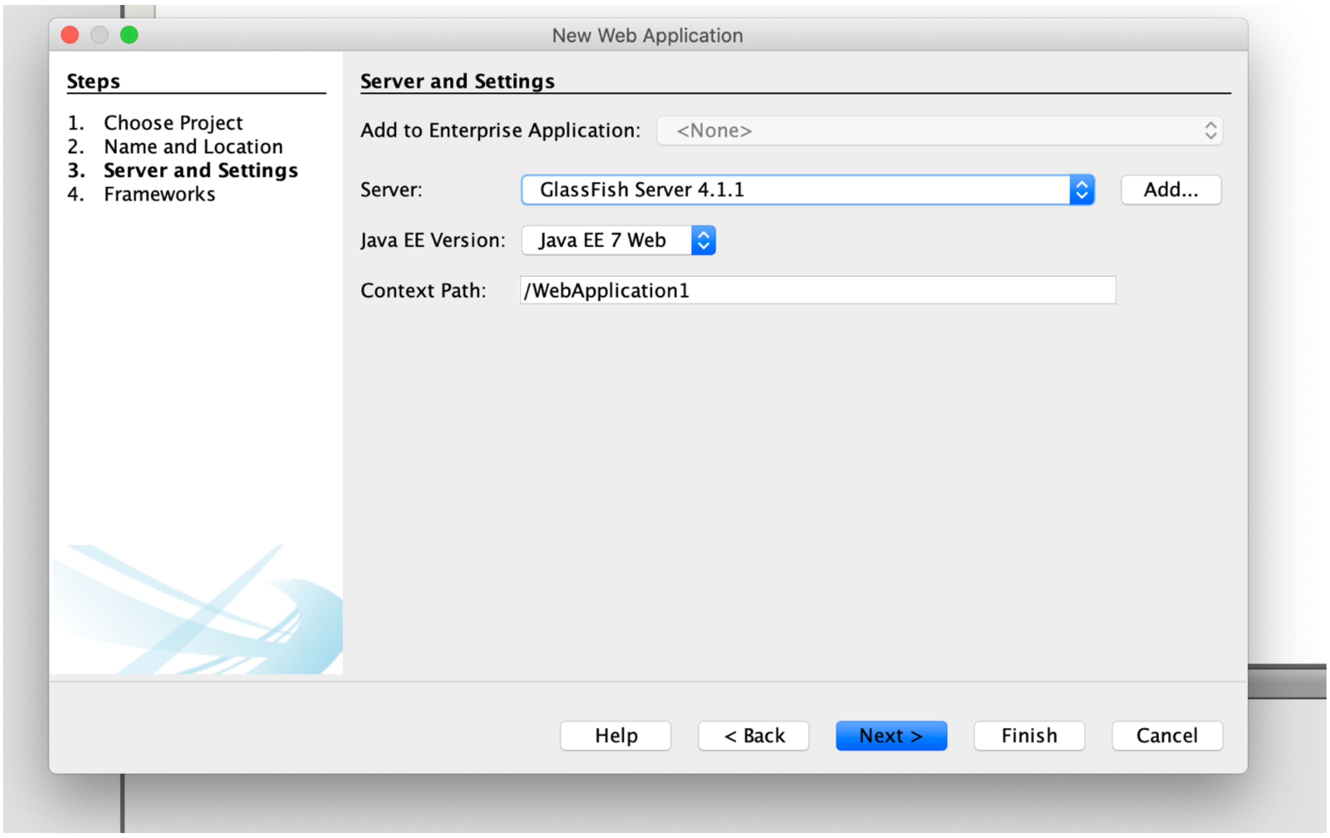
Project Location: /Users/catalinnegru/Downloads

Project Folder: /Users/catalinnegru/Downloads/WebApplication1

Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

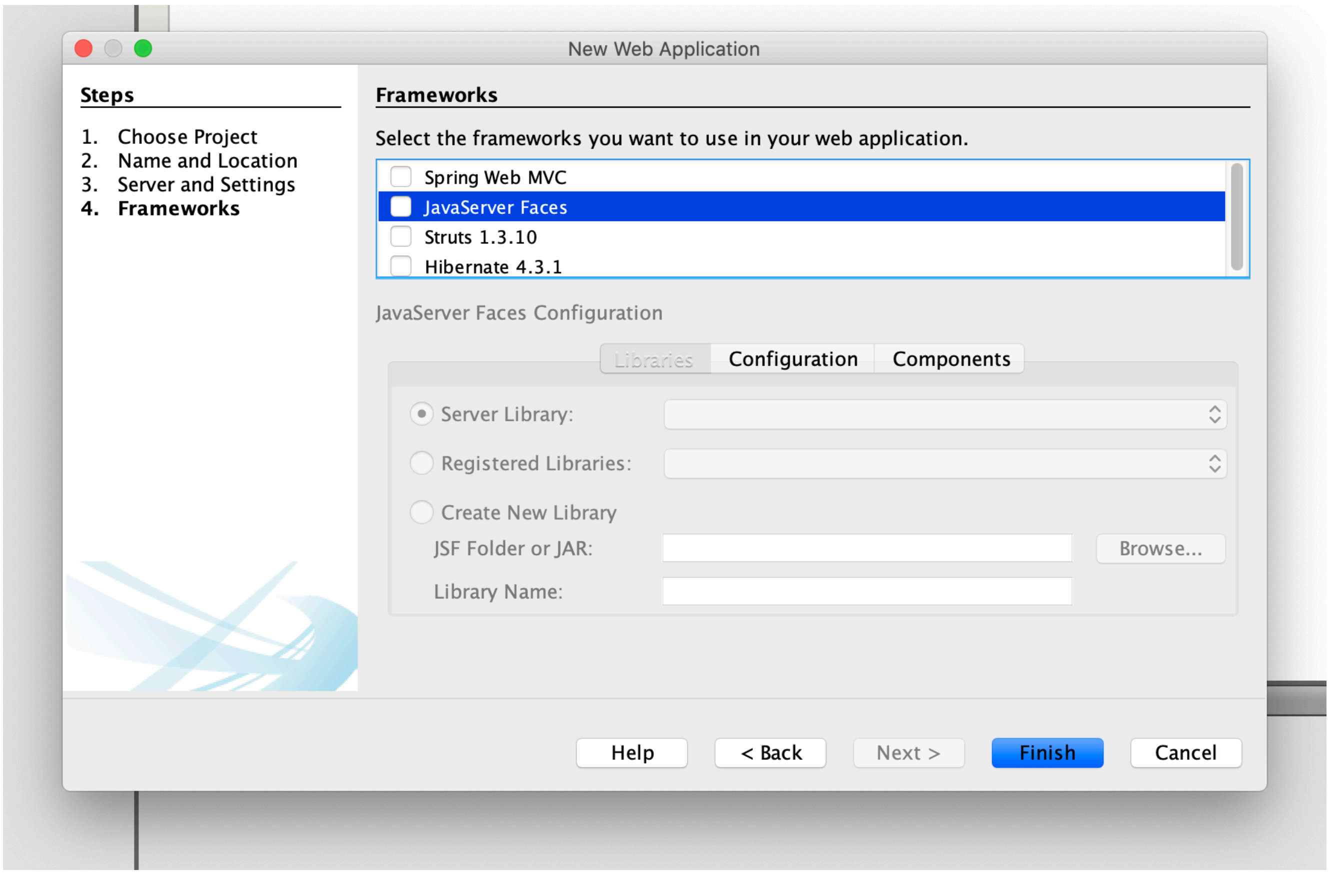


Add to Enterprise Application: <None>

Server: GlassFish Server 4.1.1

Java EE Version: Java EE 7 Web

Context Path: /WebApplication1



Frameworks

Select the frameworks you want to use in your web application.

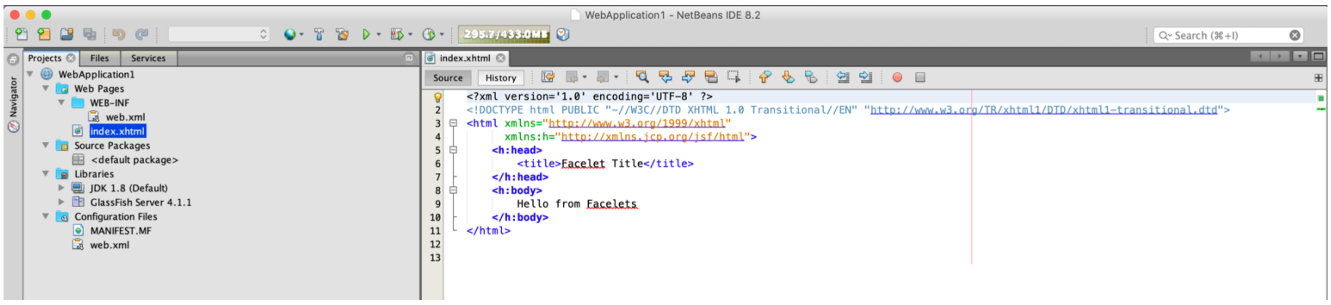
- [ ] Spring Web MVC
- [ ] JavaServer Faces
- [ ] Struts 1.3.10
- [ ] Hibernate 4.3.1

JavaServer Faces Configuration

Libraries | Configuration | Components

- [● Server Library:]
- Registered Libraries:
- Create New Library
  - JSF Folder or JAR: Browse...
  - Library Name:

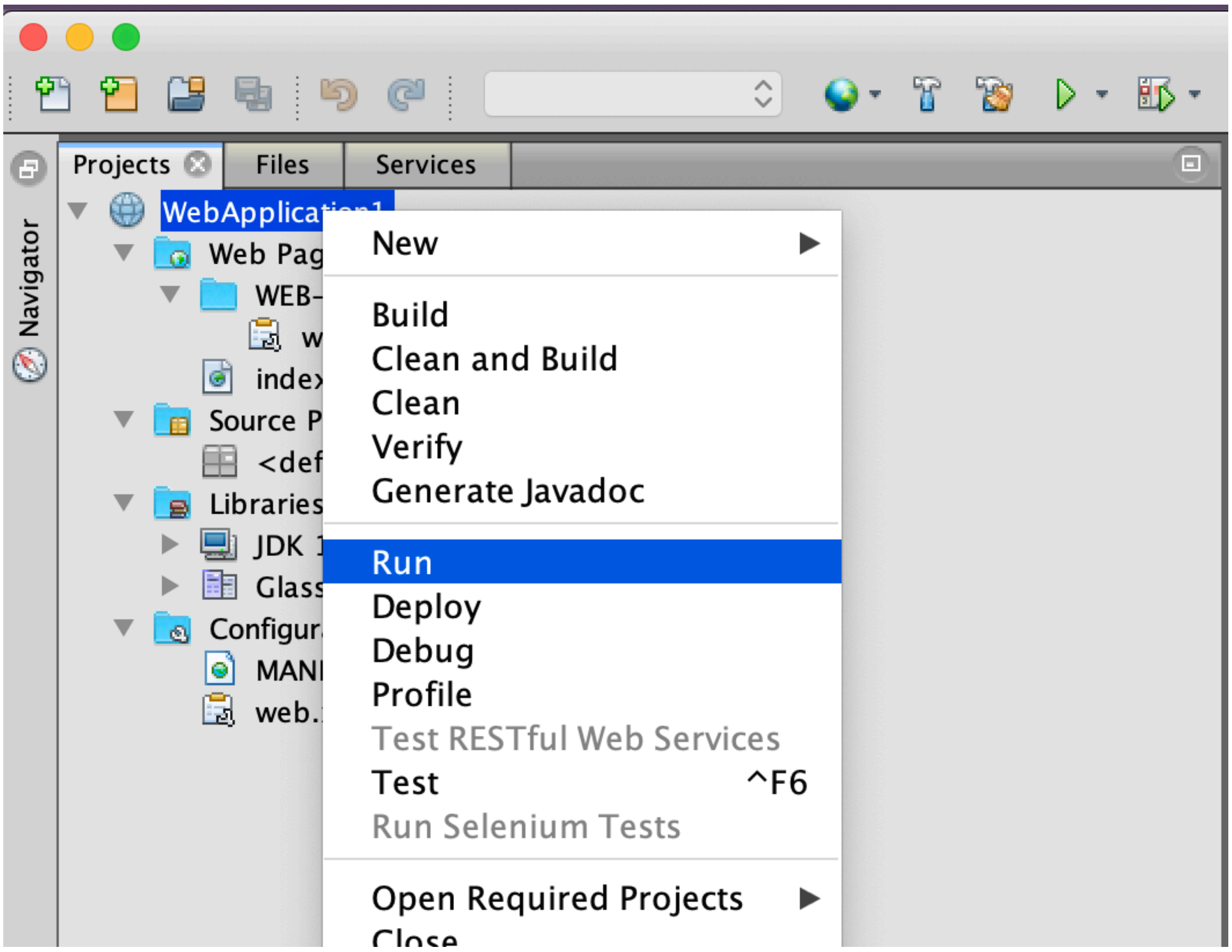
Observati structura proiectului rezultat:



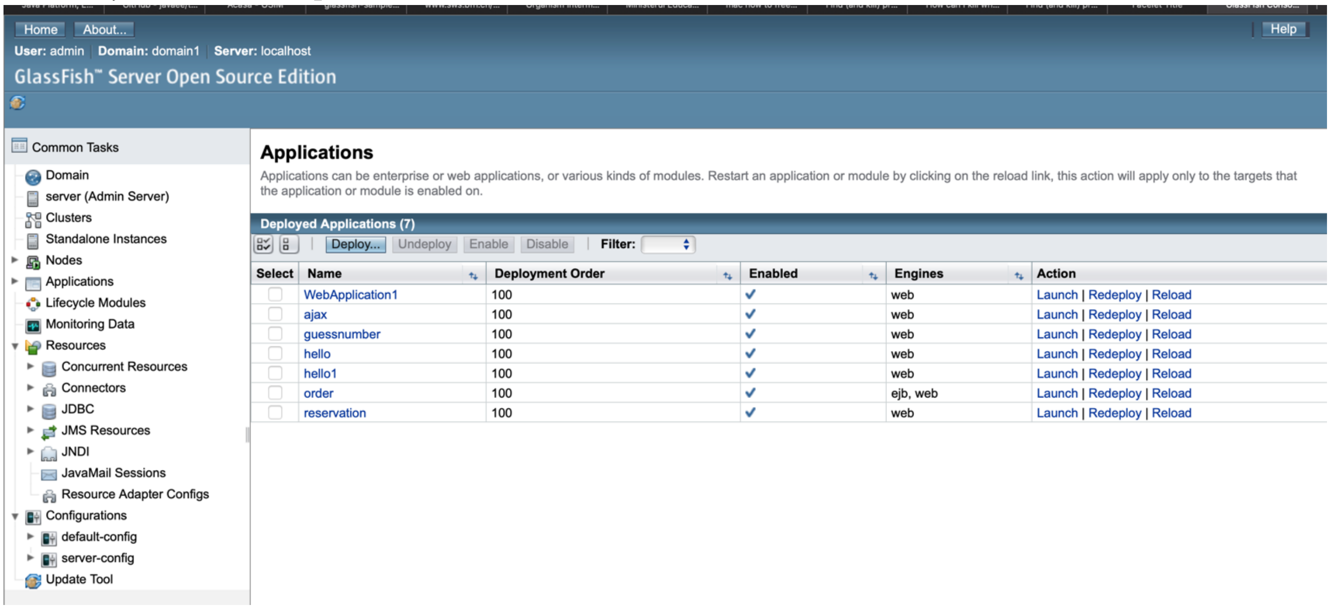
```
WebApplication1
  Web Pages
    WEB-INF
      web.xml
    index.xhtml
  Source Packages
    <default package>
  Libraries
    JDK 1.8 (Default)
    GlassFish Server 4.1.1
  Configuration Files
    MANIFEST.MF
    web.xml
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        Hello from Facelets
    </h:body>
</html>
```

Rulati aplicatia creata:



Observați modificările apărute in consola de administrare:



Applications

Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (7)

Select	Name	Deployment Order	Enabled	Engines	Action
☐	WebApplication1	100	✓	web	Launch | Redeploy | Reload
☐	ajax	100	✓	web	Launch | Redeploy | Reload
☐	guessnumber	100	✓	web	Launch | Redeploy | Reload
☐	hello	100	✓	web	Launch | Redeploy | Reload
☐	hello1	100	✓	web	Launch | Redeploy | Reload
☐	order	100	✓	ejb, web	Launch | Redeploy | Reload
☐	reservation	100	✓	web	Launch | Redeploy | Reload

2. Servleți și pagini JSP

Reluati exemplele din sectiunea Aplicații Web cu Servleţi şi pagini JSP, folosind NetBeans şi GlassFish.