SwRTc ISw
POO – an II

Catedra de Telecomunicatii

ETTI

12/05/2008

Limbaje de Programare pentru Aplicatii Internet (LPAI)

Laborator 5

Programarea Web utilizand tehnologia Java Servlet

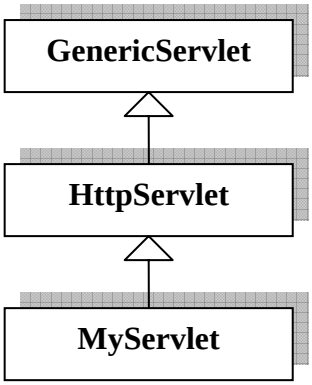
5.1. Descrierea laboratorului

In aceasta lucrare de laborator vor fi acoperite urmatoarele probleme:
- Tehnologia Java Servlet, Crearea aplicatiilor Web bazate pe servlet-uri utilizand NetBeans
- **Precizari privind temele pentru acasa si colocviul de laborator** si teme suplimentare
- Anexa

5.2. Tehnologia Java Servlet

5.2.1. Introducere in servlet-uri Java

Un **servlet** este un obiect al unei clase Java ce extinde functionalitatea unui server care lucreaza dupa modelul de acces cerere-raspuns (cum este cel utilizat de protocolul HTTP, pe care se bazeaza aplicatiile Web) prin crearea unui continut dinamic.



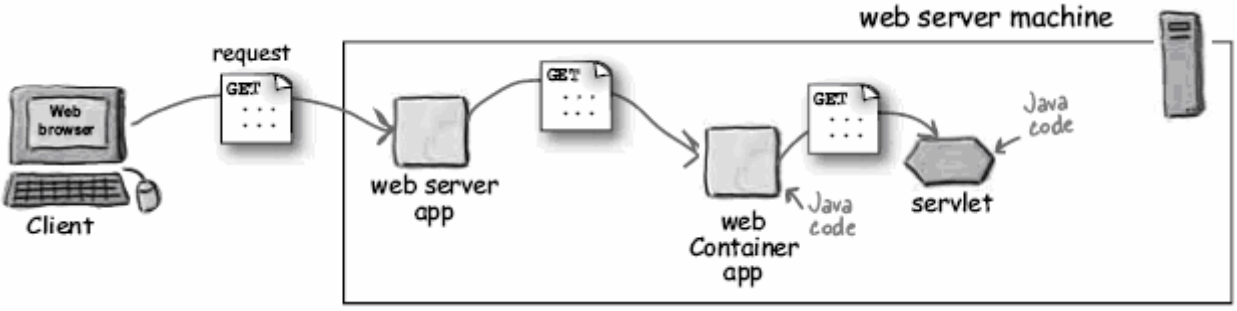
Functionalitate generica adaugata server-ului

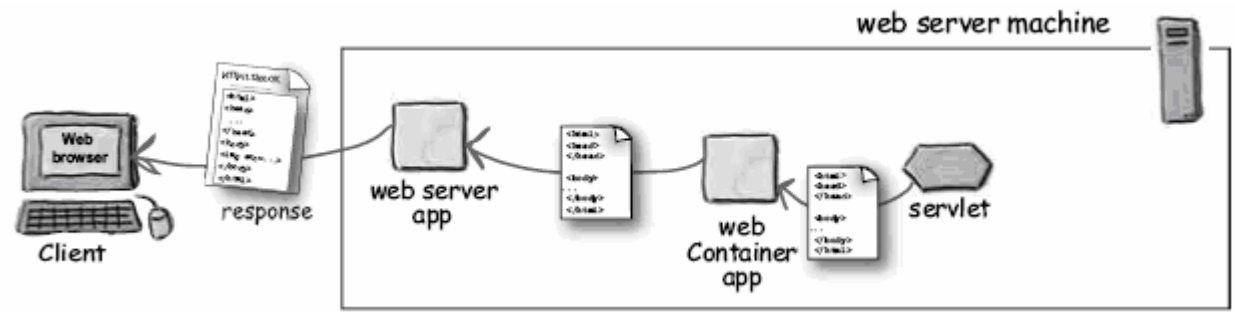
Functionalitate asociata protocolului HTTP

Functionalitate specifica aplicatiei

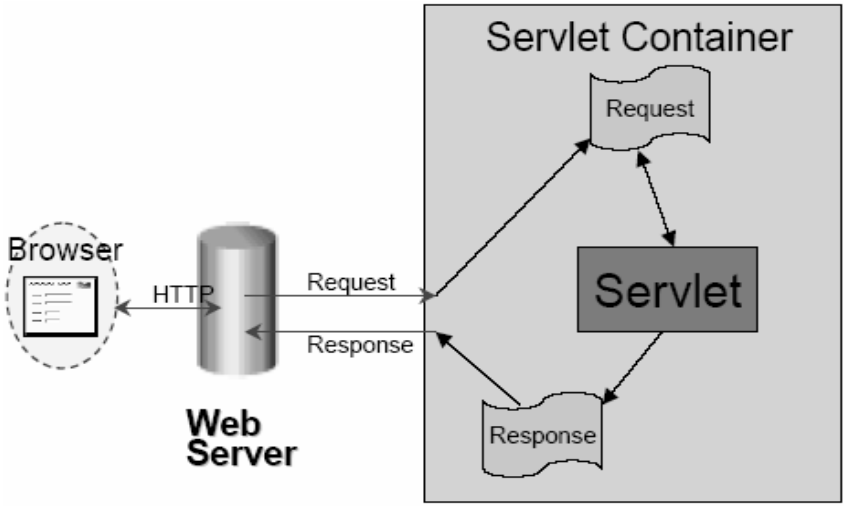
Un **servlet Web** (care adauga functionalitate unui server **HTTP**) **trebuie sa extinda** (prin mostenire) clasa **`HttpServlet`** din pachetul `javax.servlet.http`.

Servlet-urile Web sunt **componente** care **se executa intr-un container Web** (*Web container* sau *Web engine*), tot asa cum applet-urile sunt executate intr-un browser Web.

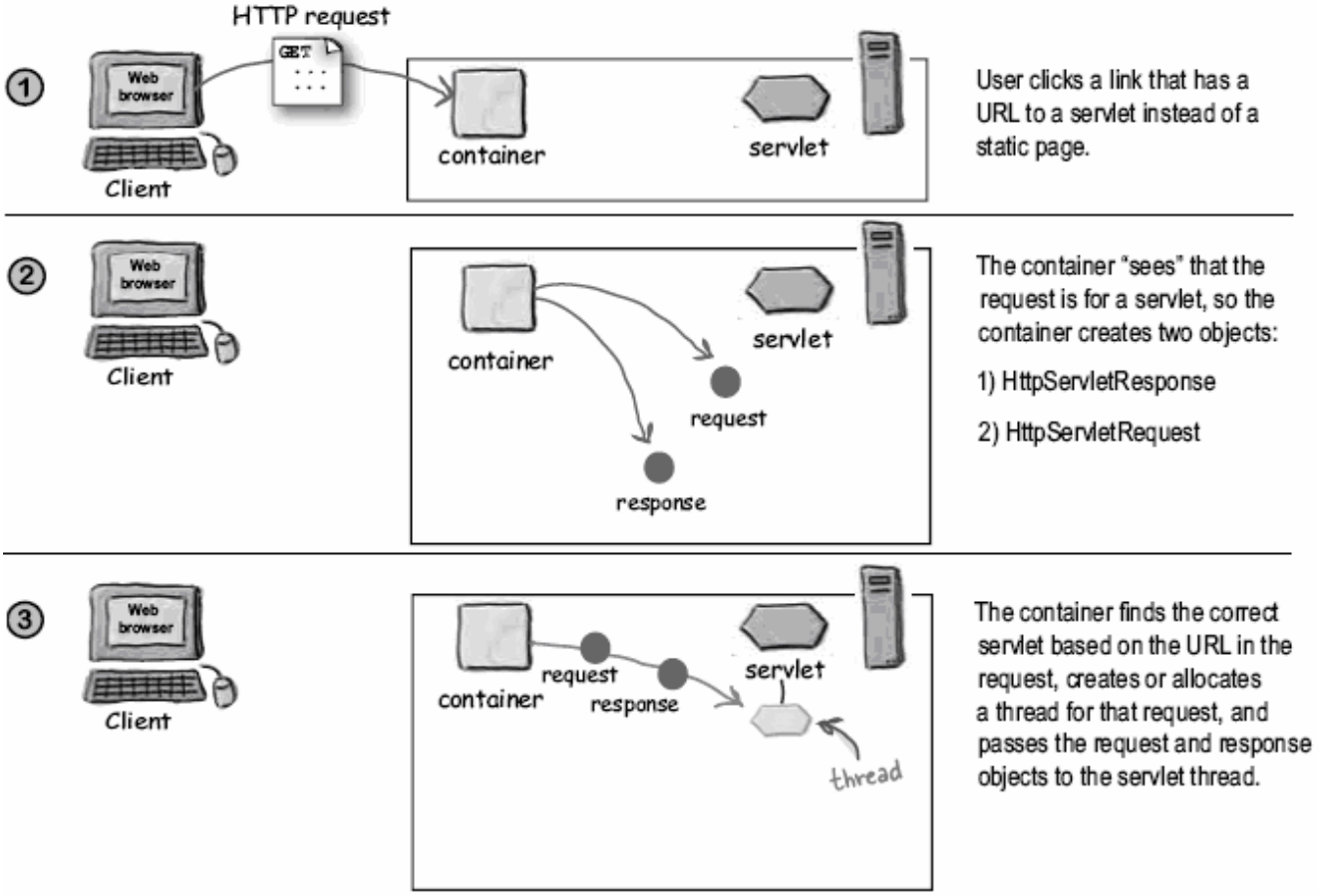


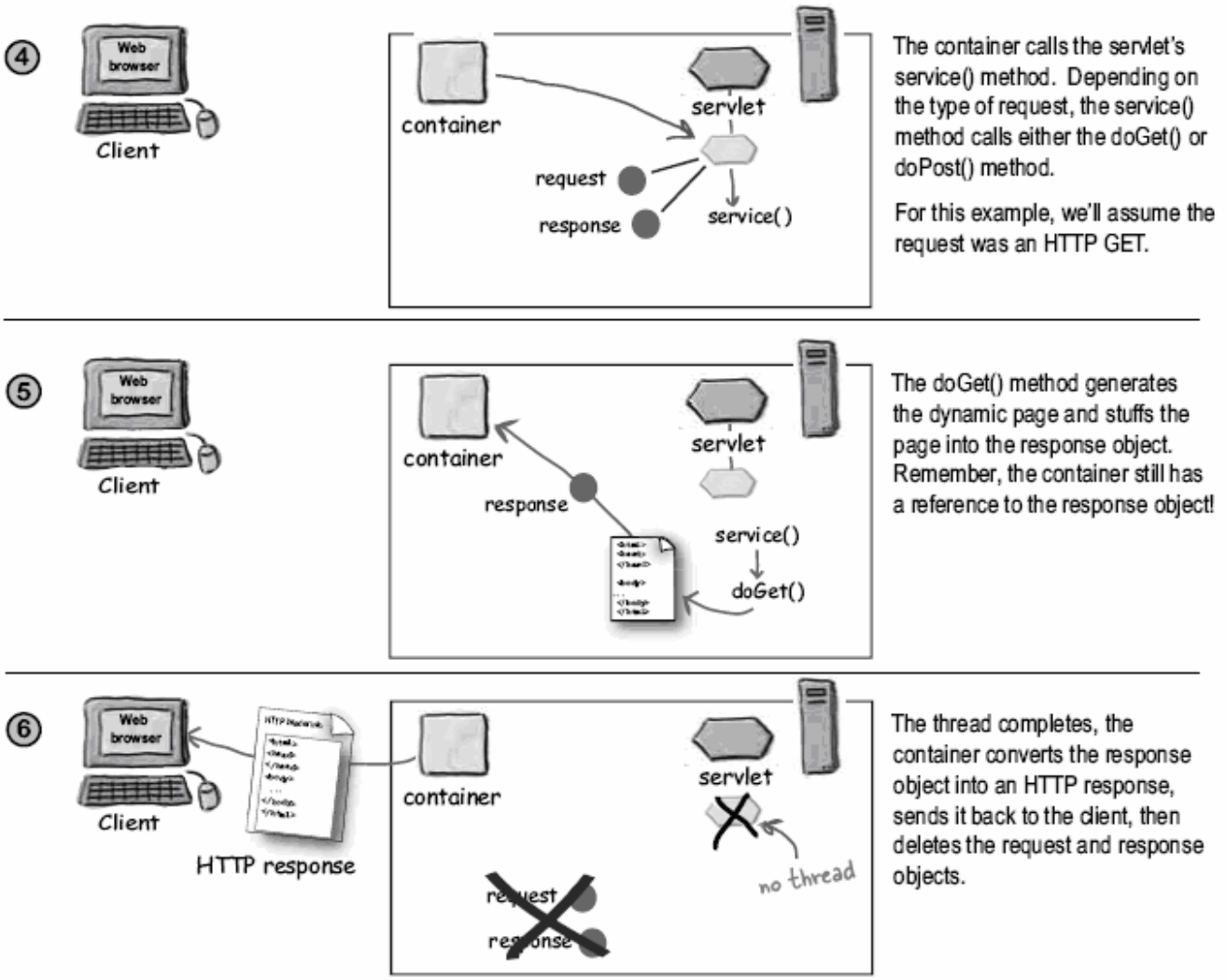


Astfel, operatiile care tin de ciclul de viata al servlet-ului (apelul metodelor **`init()`**, **`destroy()`**, **`service()`**) sunt realizate de catre container in momentele in care acestea sunt necesare (initializare, incarcare, etc.).

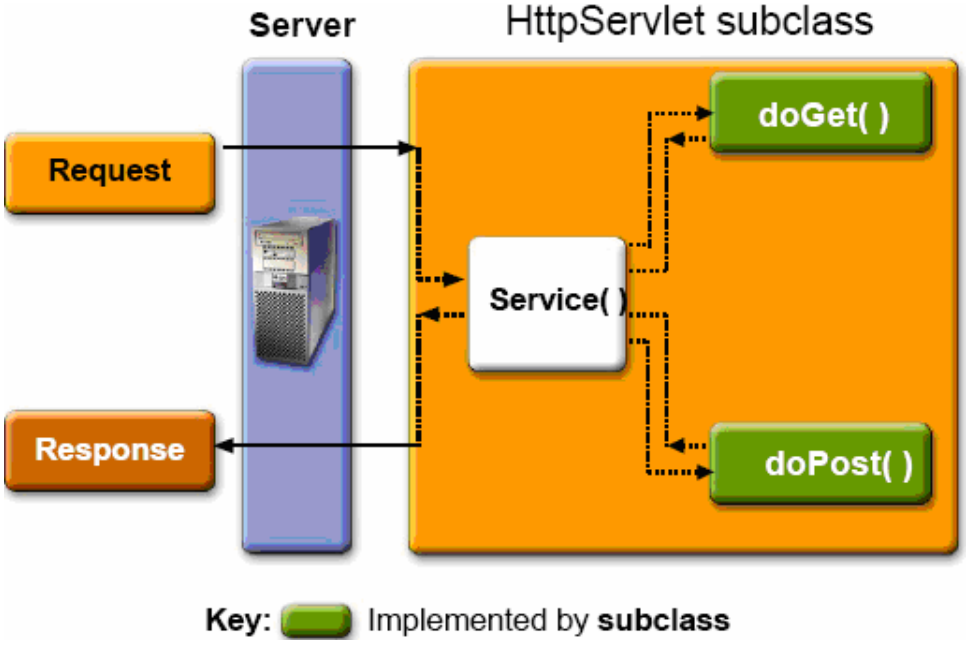


De asemenea, crearea obiectelor care incapsuleaza cererea si raspunsul HTTP, pasarea acestora metodei **`service()`**, gestionarea variabilelor CGI precum si multe alte servicii sunt realizate de catre container la momentul potrivit.

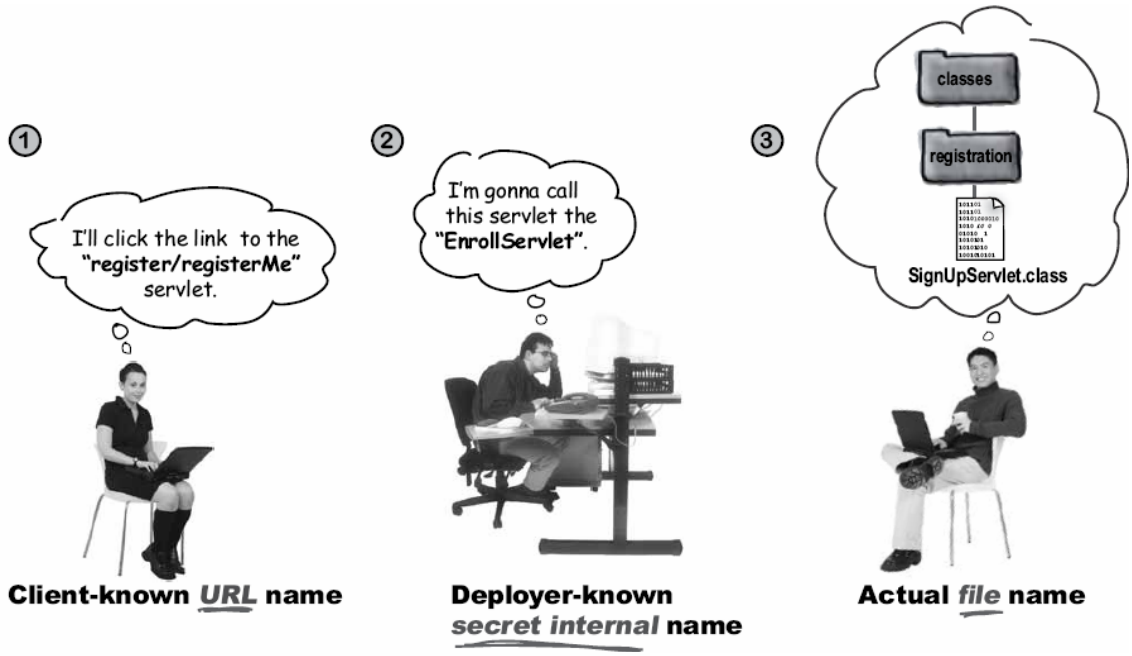




Metoda **`service()`** mostenita de la clasa **`HttpServlet`** are o implementare generica dar care se recomanda sa fie pastrata, deoarece ea identifica tipul de metoda a cererii HTTP si apeleaza metoda potrivita (**`doPost()`** in cazul metodei POST, **`doGet()`** in cazul metodei GET, etc.).



Pentru a putea fi accesat servlet-ul, clientului trebuie sa i se furnizeze o **adresa URL** care **in general difera de adresa la care se afla cu adevarat fisierul cu codul sursa al servlet-ului**.



① **<servlet>**
maps internal name to fully-qualified class name

② **<servlet-mapping>**
maps internal name to public URL name

Adresa URL (1) este **asociata prin** intermediul unui **alias** (2) dat de programator **cu calea completa necesara identificarii fisierului sursa** (3) prin codul XML scris intr-un fisier (**web.xml**) denumit *deployment descriptor* (descriptor de desfasurare/instalare - DD).

De exemplu, urmatorul continut al unui fisier **web.xml** specifica:

- **existenta unui servlet** cu numele **`ClasaServlet`** (al carui cod sursa se afla in **`ClasaServlet.java`** iar codul compilat in **`ClasaServlet.class`**) cu ajutorul tag-urilor XML **`<servlet>`** si **`<servlet-class>`**,
- asocierea servlet-ului **`ClasaServlet`** cu **aliasul** **`numeintern`** (prin intermediul tag-ului **`<servlet-name>`**),
- asocierea **aliasului** **`numeintern`** cu **formatul utilizat de client pentru URL** **`/ServletAccesServiciu`** (prin intermediul tag-urilor **`<servlet-mapping>`** si **`<url-pattern>`**),

```
<web-app>
    <servlet>
        <servlet-name>numeintern</servlet-name>
        <servlet-class>ClasaServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>numeintern</servlet-name>
        <url-pattern>/ServletAccesServiciu</url-pattern>
    </servlet-mapping>
</web-app>
```

5.2.2. Formatul general al unui servlet Java

Rolurile pe care componentele Web (servlet-urile dar si paginile JSP) le pot juca sunt:

1) **primirea cererilor** HTTP **de la client** (sub forma de obiecte HttpServletRequest) si eventual **utilizarea parametrilor obtinuti din formularul care a generat cererea**,

2) **executarea sarcinilor aplicatiei** (denumite ***business logic***) fie **direct** fie prin **delegarea catre o alta componenta**:

- alte componente **Web** – servlet-uri sau pagini JSP,
- componente **business** locale (JavaBeans) sau distribuite (Enterprise JavaBeans),

3) **generarea dinamica a continutului** si **trimiterea lui in raspunsul catre client prin intermediul raspunsurilor HTTP** (sub forma de obiecte HttpServletResponse).

Un posibil template al servlet-urilor Java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ClasaServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    processRequest(request, response);
  }
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    processRequest(request, response);
  }
  protected void processRequest(HttpServletRequest request,
          HttpServletResponse response) throws ServletException, IOException {
    // Stabilirea tipului de continut
    response.setContentType("text/html");

    // Utilizare "request" pentru a citi antetele HTTP primite (de ex. cookies)
    // si datele formularului HTML (pe care utilizatorul le-a introdus si trimis)

    // Utilizare "response" pentru a specifica linia si antetele raspunsului HTTP
    // (tipul de continut, cookies).

    PrintWriter out = response.getWriter();
    // Utilizare "out" pentru a trimite continut HTML catre browser
  }
}
```

Serverul Web de la Apache realizat in cadrul proiectului Jakarta, numit **Tomcat**, reprezinta o implementare de referinta oficiala a specificatiilor *servlet* si JSP. El **include un container pentru servlet-uri si a JSP-uri**. Ultima versiune de Tomcat poate fi obtinuta de la adresa Web: http ://jakarta.apache.org/site/binindex.cgi. Mai multe detalii despre lucrul direct cu containerele Jackarta Tomcat (structura de directoare, modul de desfasurare, etc.) pot fi obtinute la adresa http://discipline.elcom.pub.ro/swrtc/2006_Laborator/2006_SwRTc_Lab_3_Main_v01.htm.

5.3. Crearea aplicatiilor Web bazate pe servlet-uri utilizand NetBeans

5.3.1. Forma initiala a unei aplicatii Web de acces la serviciile oferite de clasa Orar

Pentru exemplificare, vom folosi din clasa Orar, accesata la distanta prin intermediul servlet-urilor:

```
 1  public class Orar {
 2
 3      private String[] orar; // camp ascuns (starea obiectului)
 4
 5      public Orar() {
 6          orar = new String[7]; // alocarea dinamica a spatiului pentru tablou
 7          // popularea tabloului cu valori
 8          orar[0] = "Luni nu sunt ore de LPAI.";
 9          orar[1] = "Marti sunt laboratoare de LPAI.";
10          orar[2] = "Miercuri nu sunt ore de LPAI.";
11          orar[3] = "Joi sunt proiecte de LPAI.";
12          orar[4] = "Vineri este curs de LPAI.";
13          orar[5] = "Sambata nu sunt ore de LPAI.";
14          orar[6] = "Duminica nu sunt ore de LPAI.";
15      }
16      public String getOrar(int zi) { // metoda accesor - getter
17          return orar[zi];            // returneaza referinta la tablou
18      }
19      public void setOrar(int zi, String text) { // metoda accesor - setter
20          orar[zi] = text;                       // inlocuieste un element
21      }
22  }
```

In laborator:

1. **Se deschide NetBeans IDE 5.5** si **se sterg** (**Delete Project**) toate **proiectele anterior deschise**.
2. **Se creaza un nou proiect** (**File > New Project** sau **Ctrl-Shift-N**).
3. In **Categories** se selecteaza **Web**, in **Projects** se selecteaza **Web Application**, si se apasa **Next**.
4. La **Project Name** se scrie numele ales pentru noul proiect: **AplicatieOrar1**
5. La **Server** se selecteaza: **Bundled Tomcat (5.5.17)**. Se apasa **Next**.
6. La **Frameworks nu se selecteaza nici un framework**. Se apasa **Finish**.

In laborator:

1. Se adauga un **New File/Folder** cu numele **Orar**, selectand **Java Classes** si **Java Class**.
2. **Se inlocuieste codul generat automat cu codul de mai sus.**

Vom incepe cu un servlet simplu care permite accesul la obiecte Orar:

```
 1  import java.io.*;
 2  import java.net.*;
 3  import javax.servlet.*;
 4  import javax.servlet.http.*;
 5
 6  public class ServletOrarInitial extends HttpServlet {
 7
 8     protected void processRequest(HttpServletRequest request,
 9           HttpServletResponse response) throws ServletException, IOException {
10        response.setContentType("text/html;charset=UTF-8");
11        PrintWriter out = response.getWriter();
12
13        // Generarea formularului pentru accesul recursiv la servicii
14        out.println("<html>");
15        out.println("<head>");
16        out.println("<title>Acces orar</title>");
17        out.println("</head>");
18        out.println("<body>");
19        out.println("<h1>Acces orar (forma initiala) - generat de servlet</h1>");
20        out.println("<hr><form name=\"input\" action=\"AccesInitial\""
21                  + " method=\"get\">");
22        out.println("<input type=\"radio\" name=\"zi\" checked=\"checked\""
23                  + " value=\"0\"> Luni");
24        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"1\"> Marti");
25        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"2\"> Miercuri");
26        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"3\"> Joi");
27        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"4\"> Vineri");
28        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"5\"> Sambata");
29        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"6\"> Duminica");
30        out.println("<hr>");
31        out.println("<input type=\"radio\" name=\"serviciu\" checked=\"checked\""
32                  + " value=\"getOrar\"> Obtinere orar");
33        out.println("<br><input type=\"radio\" name=\"serviciu\" value=\"setOrar\">"
34                  + " Modificare orar");
35        out.println("<input type=\"text\" name=\"modificare\" value=\"\">");
36        out.println("<hr><input type=\"submit\" value=\"Trimite\">");
37        out.println("</form><hr>");
38
39        Orar orar = new Orar();
40
41        // Obtinerea parametrilor introdusi de utilizator in formular
42        int zi = Integer.parseInt(request.getParameter("zi"));
43
44        // Daca serviciu cerut e obtinere orar
45        if (request.getParameter("serviciu").equals("getOrar")) {
46           out.println("<b>Orarul cerut:</b> <br>" + orar.getOrar(zi));
47        }
48        // Daca serviciu cerut e modificare orar
49        else if (request.getParameter("serviciu").equals("setOrar")) {
50           String modificare = request.getParameter("modificare");
51           orar.setOrar(zi, modificare);
52           out.println("<b>Modificarea ceruta:</b> <br>" + orar.getOrar(zi));
53        }
54        out.println("</body>");
```

```
55          out.println("</html>");
56          out.close();
57      }
58      protected void doGet(HttpServletRequest request, HttpServletResponse response)
59              throws ServletException, IOException {
60          processRequest(request, response);
61      }
62      protected void doPost(HttpServletRequest request, HttpServletResponse response)
63              throws ServletException, IOException {
64          processRequest(request, response);
65      }
66  }
```

In laborator:

1. Se adauga un **New File/Folder** selectand **Web** si **Servlet**.
2. La **Name and Location** se precizeaza numele clasei **ServletOrarInitial**.
3. La **Configure Servlet Deployment** se precizeaza numele intern (**Servlet Name**) **servletinitial** si **URL Pattern** **/AccesInitial**. Se apasa **Finish**.
4. **Se inlocuieste codul generat automat pentru ServletOrarInitial cu codul de mai sus.**

Se observa (in directorul WEB-INF) continutul generat automat al fisierului web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" 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">
    <servlet>
        <servlet-name>servletinitial</servlet-name>
        <servlet-class>ServletOrarInitial</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>servletinitial</servlet-name>
        <url-pattern>/AccesInitial</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
         <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>
</web-app>
```

In laborator:

1. **Se inlocuieste codul generat automat al fisierului index,jsp cu codul de mai jos.**

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
      <h1>Pagina Index</h1>
         <hr>
         <a href="PaginaAccesInitiala.html">Pagina acces la orar (initiala)</a>
         <hr>
    </body>
</html>
```

Acum va fi adaugata pagina HTML care contine formularul pentru accesul la servlet:

In laborator:

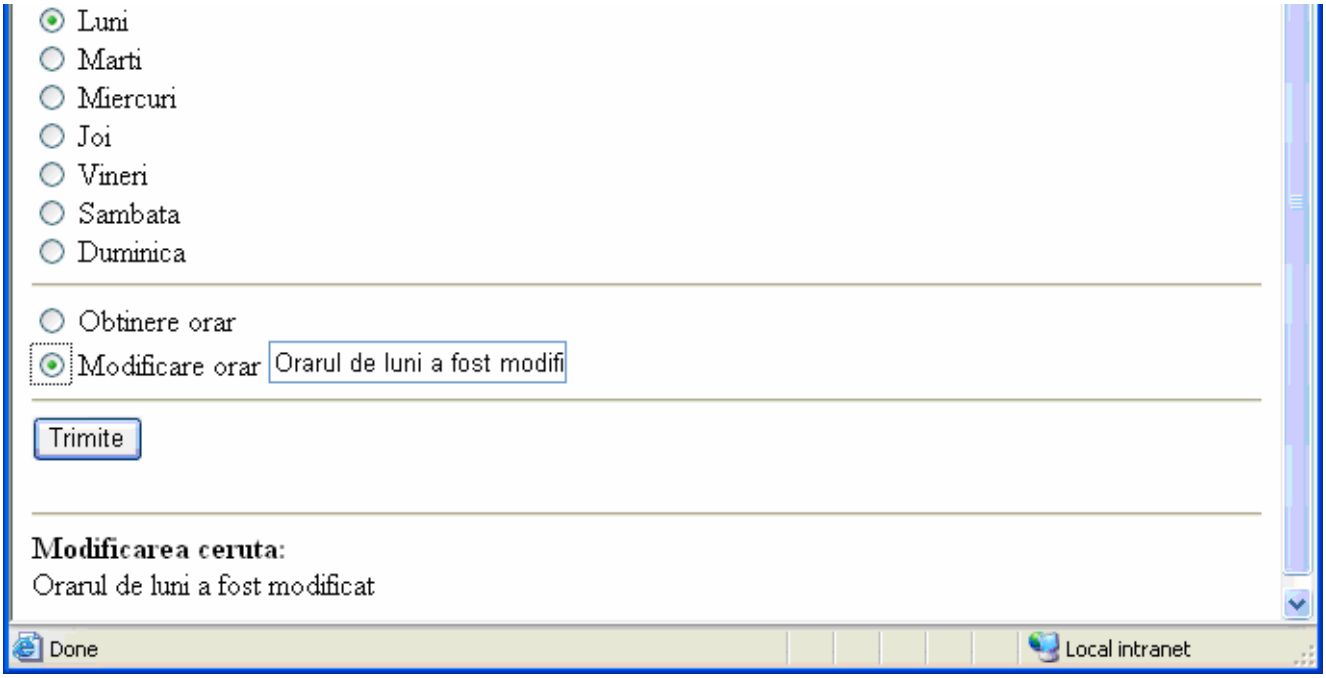
1. Se adauga un **New File/Folder** selectand **Web** si **HTML**.
2. La **Name and Location** se precizeaza numele **PaginaAccesInitiala**. Se apasa **Finish**.
3. **Se inlocuieste codul generat automat pentru cu codul de mai jos.**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Acces orar</title>
    </head>
    <body>
      <h1>Acces orar (forma initiala)</h1>
         <hr><form name="input" action="AccesInitial" method="get">
         <input type="radio" name="zi" checked="checked" value="0"> Luni
         <br> <input type="radio" name="zi" value="1"> Marti
         <br> <input type="radio" name="zi" value="2"> Miercuri
         <br> <input type="radio" name="zi" value="3"> Joi
         <br> <input type="radio" name="zi" value="4"> Vineri
         <br> <input type="radio" name="zi" value="5"> Sambata
         <br> <input type="radio" name="zi" value="6"> Duminica
         <hr>
         <input type="radio" name="serviciu" checked="checked" value="getOrar">
               Obtinere orar
         <br> <input type="radio" name="serviciu" value="setOrar"> Modificare orar
         <input type="text" name="modificare" value="">
         <input type="submit" value="Trimite">
      </form>
      <hr>
    </body>
</html>
```

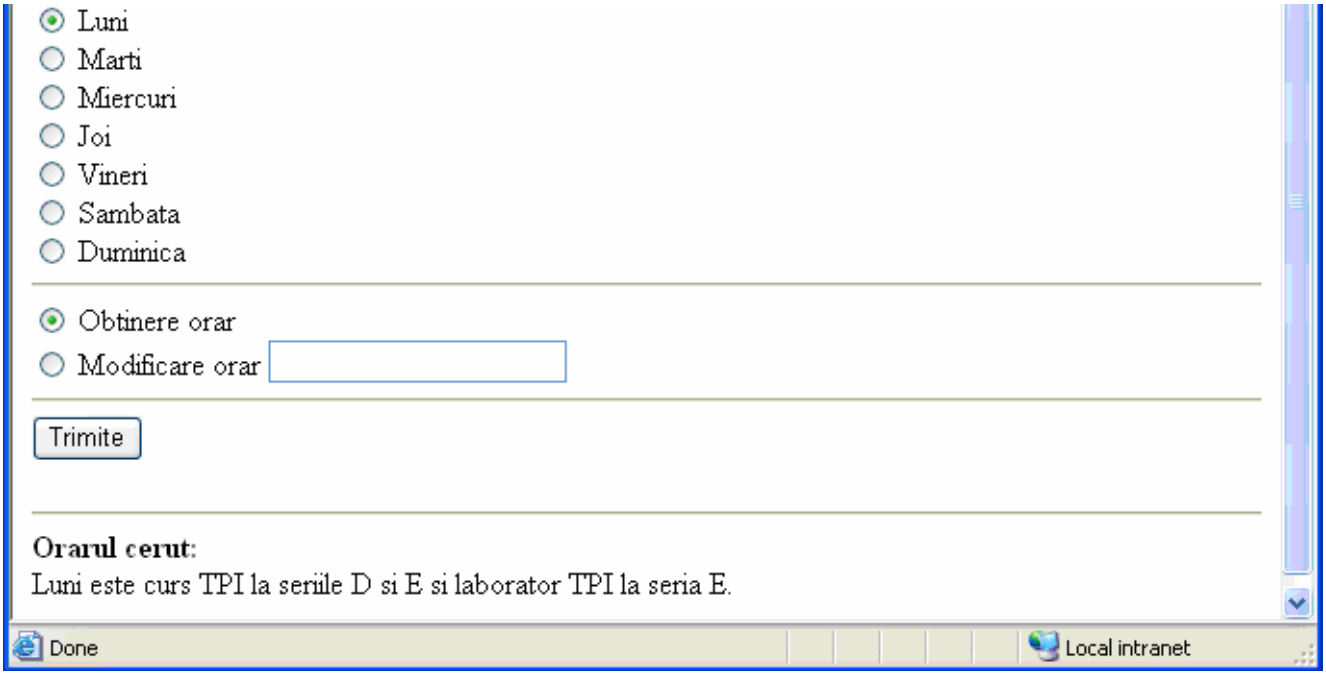
In laborator:

1. Se selecteaza **Build Project** pe **nodul proiectului AplicatieOrar1**.
2. Se selecteaza **Run Project** pe **nodul proiectului AplicatieOrar1**.
3. **Se utilizeaza pagina in diverse moduri, testand serviciile obtinere si modificare.**

Utilizand serviciul de modificare orar se face apel la metoda setOrar():



Dar daca se acceseaza din nou serviciul de obtinere modificarea nu se regaseste. Pentru ca modificarea sa fie retinuta trebuie modificat servlet-ul.



Protocolul HTTP nu are stari (este *stateless*) asa incat serverul HTTP nu retine informatii privind cererile anterioare. In plus, **pentru ca servlet-urile sa fie accesate eficient de catre mai multi clienti in acelasi timp containerul de servlet-uri formeaza un asa-numit *thread pool* cu instante ale servlet-ului din care alege unul oarecare pentru fiecare client.** De aceea **declararea obiectului de tip Orar ca variabila instanta nu este o solutie.**

Obiectele din clasa `HttpSession` gestionate de **containerul de servlet-uri** permit **pastrarea referintelor catre obiecte ale aplicatiei**, numite **atribute**, **si regasirea acestora**, prin intermediul metodelor **`setAttribute()`** si **`getAttribute()`**.

5.3.2. Aplicatia Web modificata pentru a folosi *session tracking*

Vom modifica servletul pentru a crea si utiliza o sesiune.

```
 1  import java.io.*;
 2  import java.net.*;
 3  import javax.servlet.*;
 4  import javax.servlet.http.*;
 5
 6  public class ServletOrarFinal extends HttpServlet {
 7
 8     protected void processRequest(HttpServletRequest request,
 9           HttpServletResponse response) throws ServletException, IOException {
10        response.setContentType("text/html;charset=UTF-8");
11        PrintWriter out = response.getWriter();
12
13        // Generarea formularului pentru accesul recursiv la servicii
14        out.println("<html>");
15        out.println("<head>");
16        out.println("<title>Acces orar</title>");
17        out.println("</head>");
18        out.println("<body>");
19        out.println("<h1>Acces orar (forma finala) - generat de servlet</h1>");
20        out.println("<hr><form name=\"input\" action=\"AccesFinal\""
21                  + " method=\"get\">");
22        out.println("<input type=\"radio\" name=\"zi\" checked=\"checked\""
23                  + " value=\"0\"> Luni");
24        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"1\"> Marti");
25        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"2\"> Miercuri");
26        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"3\"> Joi");
27        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"4\"> Vineri");
28        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"5\"> Sambata");
29        out.println("<br> <input type=\"radio\" name=\"zi\" value=\"6\"> Duminica");
30        out.println("<hr>");
```

```
31          out.println("<input type=\"radio\" name=\"serviciu\" checked=\"checked\""
32                  + " value=\"getOrar\"> Obtinere orar");
33          out.println("<br><input type=\"radio\" name=\"serviciu\" value=\"setOrar\">"
34                  + " Modificare orar");
35          out.println("<input type=\"text\" name=\"modificare\" value=\"\">");
36          out.println("<hr><input type=\"submit\" value=\"Trimite\">");
37          out.println("</form><hr>");
38
39          // Transformarea obiectului orar in atribut al sesiunii curente pentru
40          // salvarea starii lui
41          HttpSession ses = request.getSession();
42          Orar orar = (Orar) ses.getAttribute("orar");
43          if (orar == null) { // Daca nu exista orarul salvat ca atribut al sesiunii
44             orar = new Orar();
45             ses.setAttribute("orar", orar);
46          }
47
48          // Obtinerea parametrilor introdusi de utilizator in formular
49          int zi = Integer.parseInt(request.getParameter("zi"));
50
51          // Daca serviciu cerut e obtinere orar
52          if (request.getParameter("serviciu").equals("getOrar")) {
53             out.println("<b>Orarul cerut:</b> <br>" + orar.getOrar(zi));
54          }
55          // Daca serviciu cerut e modificare orar
56          else if (request.getParameter("serviciu").equals("setOrar")) {
57             String modificare = request.getParameter("modificare");
58             orar.setOrar(zi, modificare);
59             out.println("<b>Modificarea ceruta:</b> <br>" + orar.getOrar(zi));
60          }
61          out.println("</body>");
62          out.println("</html>");
63          out.close();
64       }
65       protected void doGet(HttpServletRequest request, HttpServletResponse response)
66               throws ServletException, IOException {
67          processRequest(request, response);
68       }
69       protected void doPost(HttpServletRequest request, HttpServletResponse response)
70               throws ServletException, IOException {
71          processRequest(request, response);
72       }
73    }
```

In laborator:

1. Se adauga un **New File/Folder** selectand **Web** si **Servlet**.
2. La **Name and Location** se precizeaza numele clasei `ServletOrarFinal`.
3. La **Configure Servlet Deployment** se precizeaza numele intern (**Servlet Name**) **servletfinal** si **URL Pattern /AccesFinal**. Se apasa **Finish**.
4. **Se inlocuieste codul generat automat pentru `ServletOrarFinal` cu codul de mai sus.**
5. **Se inlocuieste codul generat automat al fisierului index.jsp cu codul de mai jos.**

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
      <h1>Pagina Index</h1>
         <hr>
         <a href="PaginaAccesInitiala.html">Pagina acces la orar (initiala)</a>
         <hr>
         <a href="PaginaAccesFinala.html">Pagina acces la orar (finala)</a>
         <hr>
    </body>
</html>
```

Acum va fi adaugata pagina HTML care contine formularul pentru accesul la servlet:

In laborator:

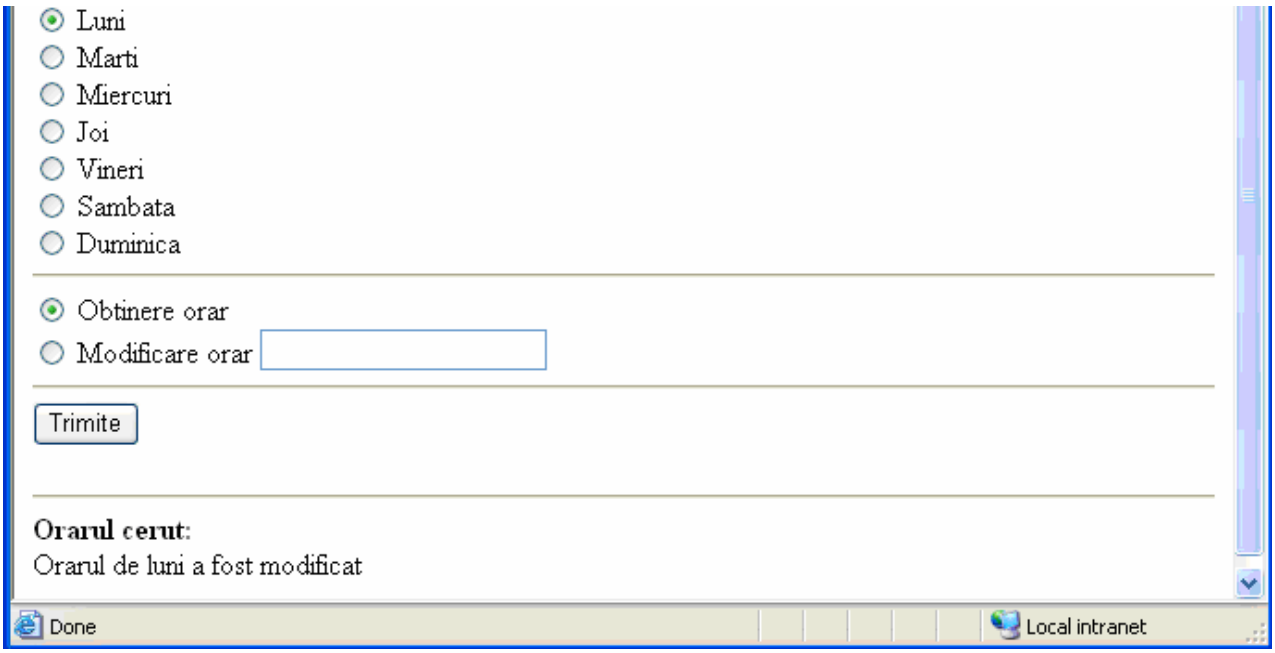
1. Se adauga un **New File/Folder** selectand **Web** si **HTML**.
2. La **Name and Location** se precizeaza numele **PaginaAccesFinala**. Se apasa **Finish**.
3. **Se inlocuieste codul generat automat pentru cu codul de mai jos.**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Acces orar</title>
    </head>
    <body>
      <h1>Acces orar (forma finala)</h1>
         <hr><form name="input" action="AccesFinal" method="get">
         <input type="radio" name="zi" checked="checked" value="0"> Luni
         <br> <input type="radio" name="zi" value="1"> Marti
         <br> <input type="radio" name="zi" value="2"> Miercuri
         <br> <input type="radio" name="zi" value="3"> Joi
         <br> <input type="radio" name="zi" value="4"> Vineri
         <br> <input type="radio" name="zi" value="5"> Sambata
         <br> <input type="radio" name="zi" value="6"> Duminica
         <hr><input type="radio" name="serviciu" checked="checked" value="getOrar">
               Obtinere orar
         <br> <input type="radio" name="serviciu" value="setOrar"> Modificare orar
         <input type="text" name="modificare" value="">
         <input type="submit" value="Trimite">
      </form><hr>
    </body>
</html>
```

In laborator:

1. Se selecteaza **Build Project** pe **nodul proiectului AplicatieOrar1**.
2. Se selecteaza **Run Project** pe **nodul proiectului AplicatieOrar1**.
3. **Se utilizeaza pagina in diverse moduri, testand serviciile obtinere si modificare.**

De aceasta data, daca se acceseaza serviciul de obtinere dupa cel de modificare, aceasta se regaseste.



5.4. Precizari privind temele pentru acasa si colocviul de laborator

Colocviul de laborator va fi sustinut la ultimul lucrare (in saptamanile 13-14), pe baza temelor pentru acasa de la laboratoarele 1-4 (care au fost realizate pe grupuri de 2-3 studenti) si a unor intrebari adresate individual din continutul temelor. De aceea este necesara existenta temelor pe hartie (scrise de mana sau listing).

Anexa

1. Resurse suplimentare privind servlet-urile (si NetBeans IDE 5.5)

De la www.**javapassion**.com (**Java EE Programming (with Passion!)**):

LAB-4002: Servlet Basics (arhiva 4002_servletbasics.zip)

LAB-4004: Session Tracking (arhiva 4004_sessiontracking.zip)

LAB-4005: Servlet Advanced (Filtering, Event Handling) (arhiva 4005_servletadv.zip)

LAB-4016 Building Bookstore Sample Apps (arhiva 4016_jspbookstore.zip)

2. Resurse suplimentare privind aplicatiile Web si NetBeans IDE 5.5

De la www.**javapassion**.com:

LAB-4001: Web Application Structure (arhiva 4001_webappstructure.zip)

LAB-4011: NetBeans Quick Start Guide for Web Apps II (arhiva 4011_netbeanswebapp2.zip)

De la NetBeans IDE 5.5.1 Documentation (http://www.netbeans.org/kb/55/index.html)

NetBeans IDE 5.5 Quick Start Guide

NetBeans IDE 5.5 Tutorial for Web Applications