2009 - **2010**

Tehnologii de Programare in Internet (TPI / RST)

Titulari curs: **Mihnea Magheti**, Eduard-Cristian Popovici

Suport curs: *http://discipline.elcom.pub.ro/tpi/*

Moodle: *http://electronica07.curs.ncit.pub.ro/course/category.php?id=3*

Structura cursului

Continut curs TPI

1. Introducere in tehnologiile Internet

2. Introducere in tehnologiile desktop (SE) Java

- 2.1. Elemente de baza. Tipuri de date referinta. Clase de biblioteca
- 2.2. Clase pentru fluxuri de intrare-iesire (IO)

3. Programarea la nivel socket in Java

- 3.1. Introducere in Protocolul Internet (IP) si stiva de protocoale IP
- 3.2. Socketuri flux (TCP) Java si programe multifilare (threads)
- 3.3. Socketuri datagrama (UDP) Java

4. Tehnologii Java de programare a aplicatiilor Web (EE) Java

- 4.1. Tehnologii client. Miniaplicatii Java (applet-uri)
- 4.2. Clase pentru interfete grafice cu utilizatorul (AWT, Swing)
- 4.3. Platforma Java EE. Arhitectura si tehnologiile implicate
- 4.4. Tehnologii server. Tehnologia Java Servlet
- 4.5. Tehnologia Java ServerPages (JSP)
- 4.6. Accesul la baze de date prin tehnologii Java (JDBC, Hibernate)
- 4.7. Tehnologii avansate (frameworks, componente EJB, Servicii Web)

Structura cursului

4. Tehnologii Java de programare a aplicatiilor Web

4.5. Tehnologia Java ServerPages (JSP)

Tehnologia Java ServerPages (JSP)

Servlet-urile au unele **dezavantaje**

- **trebuie scrise String-uri complexe** (care includ **caractere *escape***, de exemplu **\"** in locul fiecarei ghilimele) **pentru fiecare linie de cod HTML care urmeaza sa ii fie trimisa clientului**

- sunt **necesare cunostinte de Java pentru a scrie intreg codul** unui servlet

Pentru ca **puterea servlet-urilor** sa fie pusa **la dispozitia dezvoltatorilor Web fara a-i obliga sa invete Java** a aparut **specificatia Java ServerPages (JSP)**

- care **combina puterea si extensibilitatea limbajului Java** cu **simplitatea si usurinta de folosire a scripturilor** pe baza de etichete

O pagina JSP este un **document text** ce **contine 2 tipuri de text**

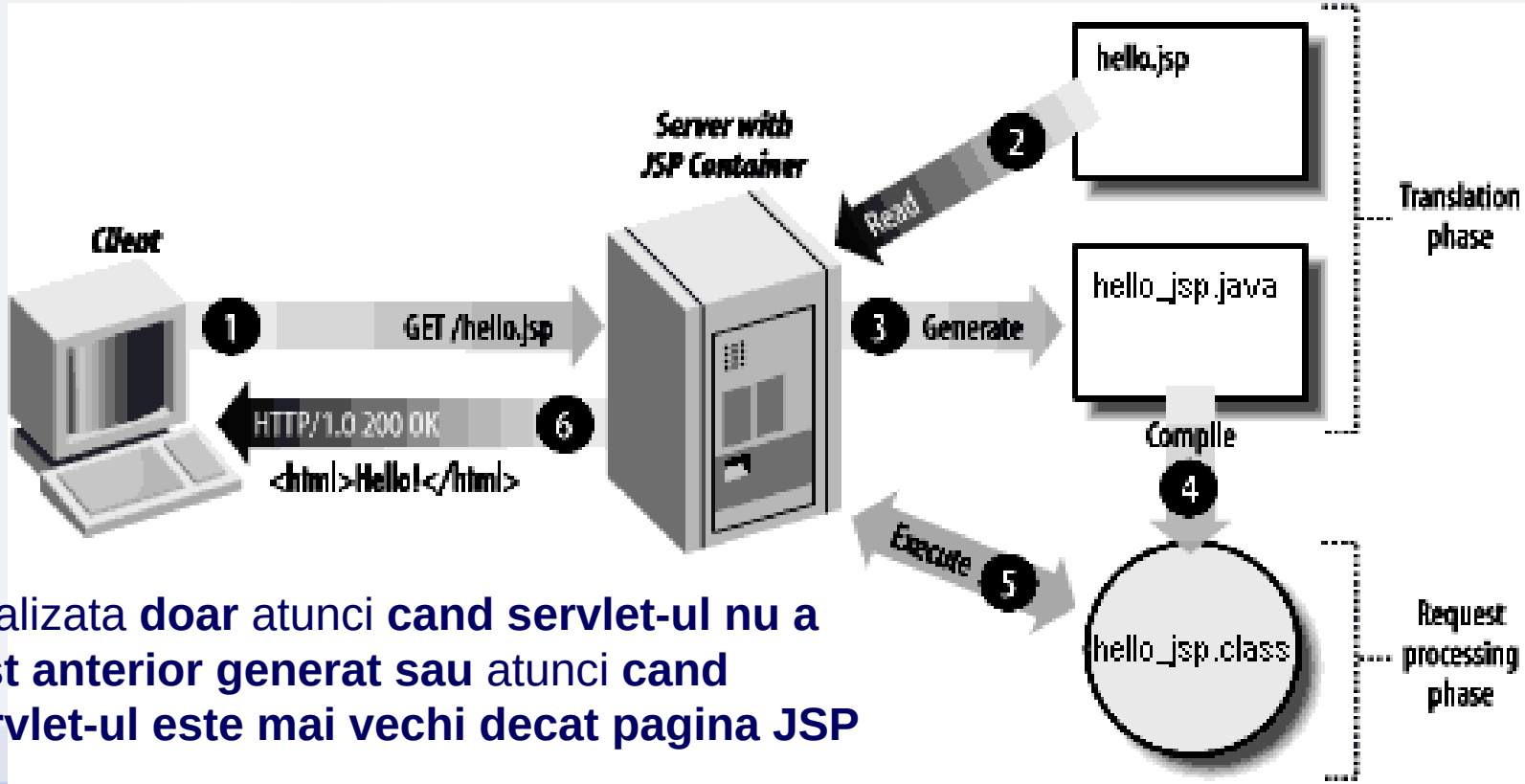
- **static**, ce poate fi **exprimat in orice tip de format bazat pe text** (**HTML**, **WML**, **XML**, etc.), si

- **continutul JSP propriu-zis**, altfel spus **dinamic**

Tehnologia Java ServerPages (JSP)

Pentru a se ajunge de la **continutul unei pagini JSP** la **continut generat dinamic** se parcurg **trei etape**

- **translatia (2,3)** - in care **pagina JSP** este **transformata** de catre **containerul de JSP-uri** intr-un **servlet** (de ex. **hello.jsp** in **hello_jsp.java**)

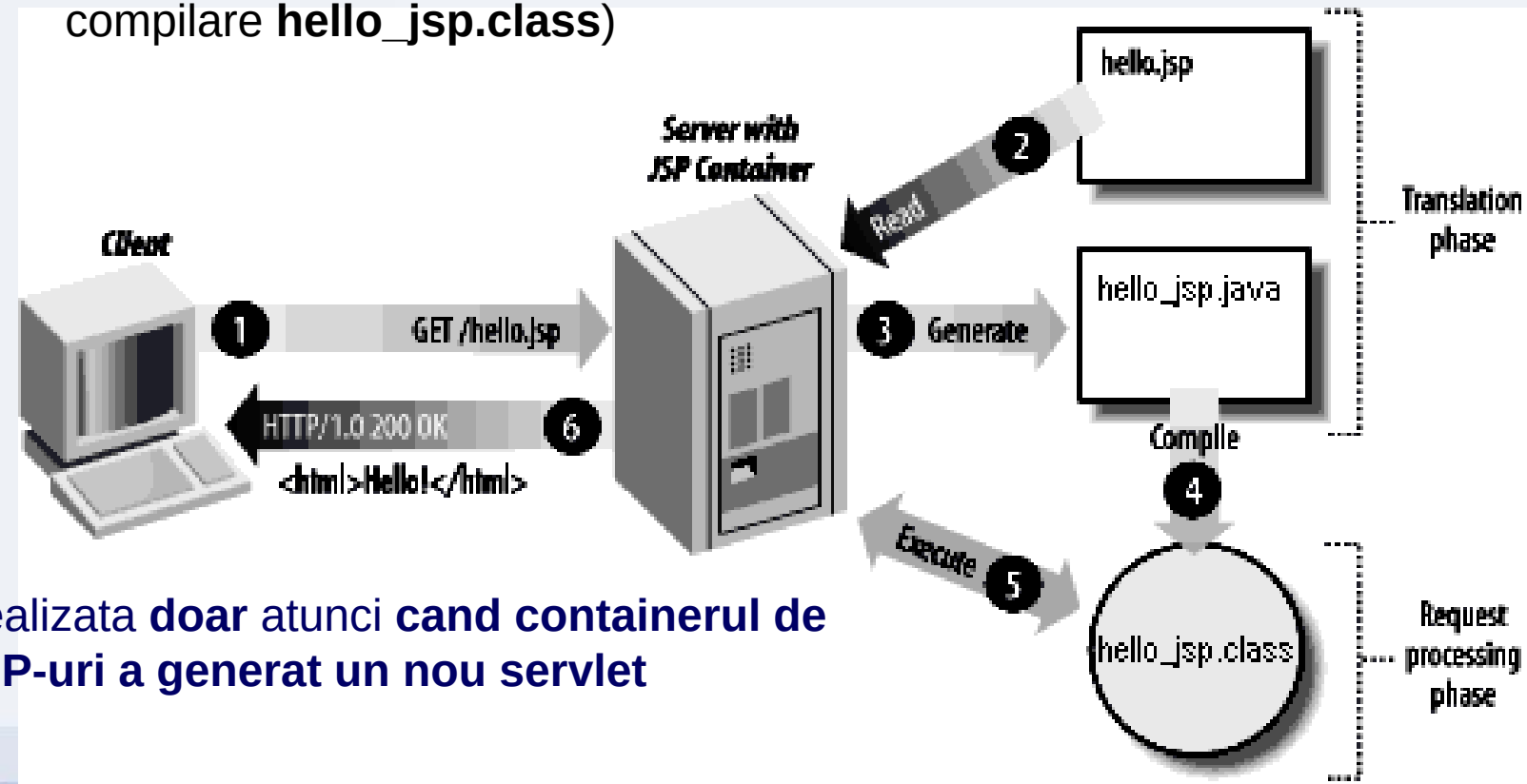


Realizata **doar** atunci **cand servlet-ul nu a fost anterior generat sau** atunci **cand servlet-ul este mai vechi decat pagina JSP**

Tehnologia Java ServerPages (JSP)

Pentru a se ajunge de la **continutul unei pagini JSP** la **continut generat dinamic** se parcurg **trei etape**

- **compilarea (4)** - in care **servletul** obtinut la primul pas este **compilat** de catre **containerul de JSP-uri** (de ex. din **hello_jsp.java** se obtine prin compilare **hello_jsp.class**)

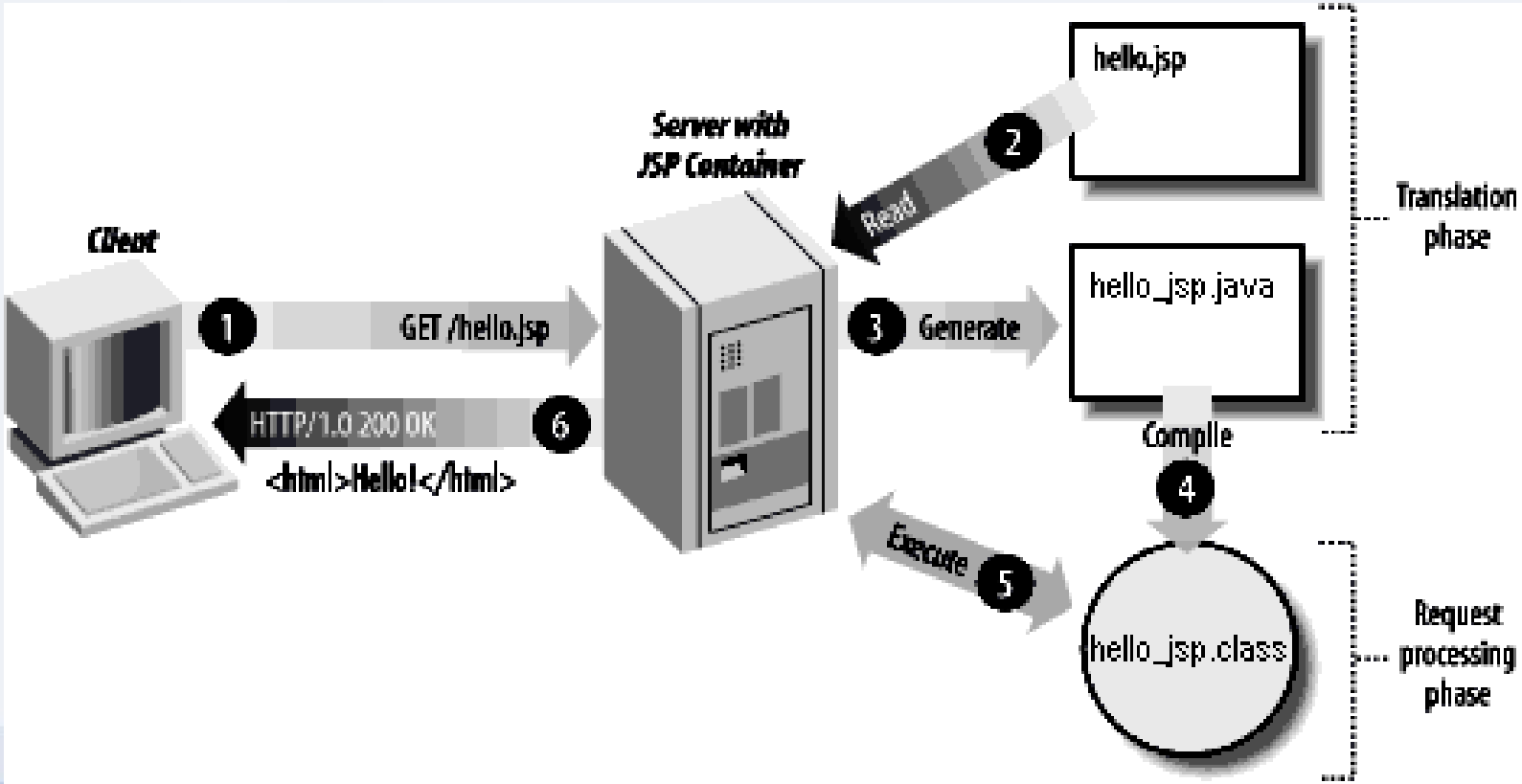


Realizata **doar** atunci **cand containerul de JSP-uri a generat un nou servlet**

Tehnologia Java ServerPages (JSP)

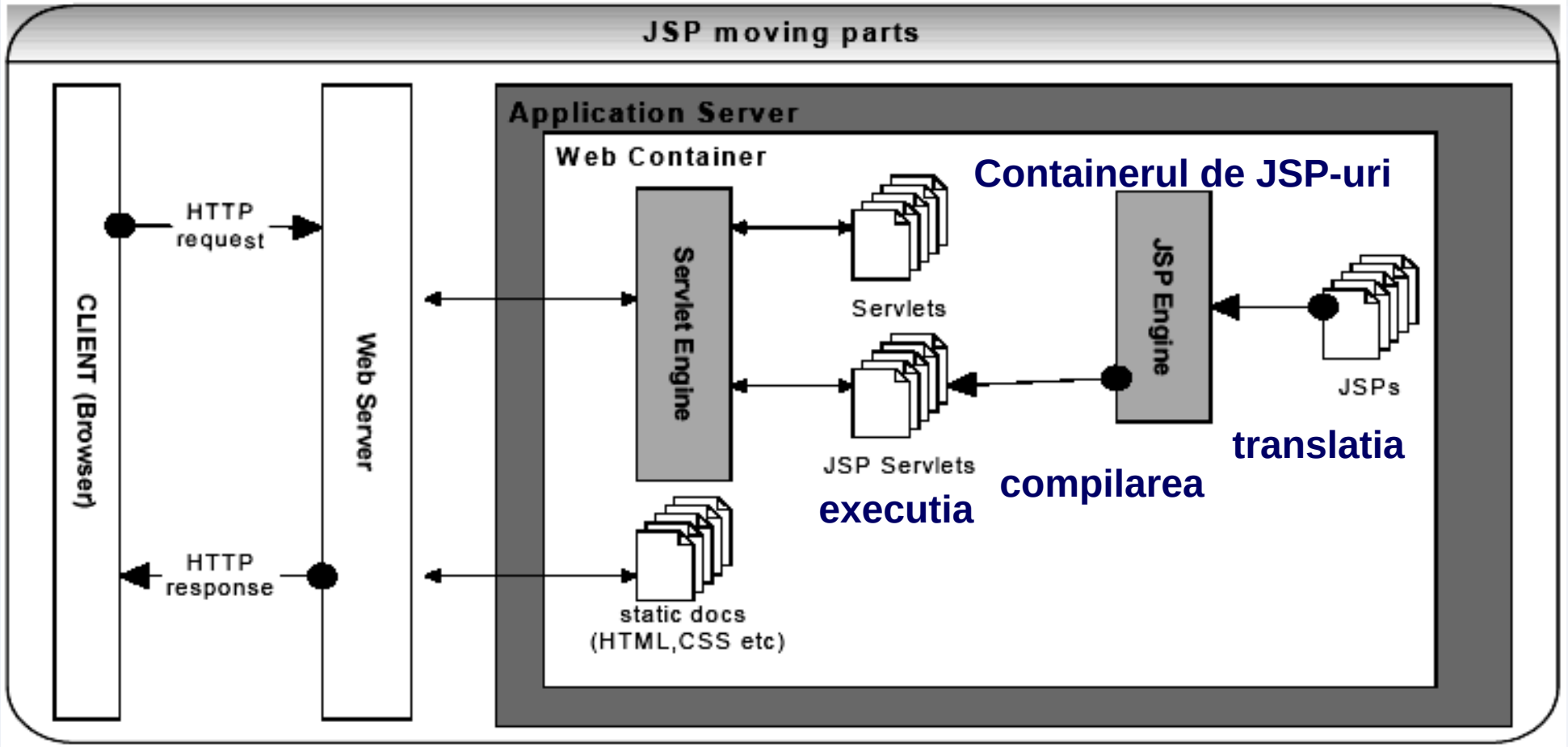
Pentru a se ajunge de la **continutul unei pagini JSP** la **continut generat dinamic** se parcurg **trei etape**

- **executia (5)** - in care **cererile catre pagina JSP** sunt **directionate catre servlet** (care **va genera raspunsul**)



Tehnologia Java ServerPages (JSP)

Etapele generarii continutului HTML din continutul paginii JSP



Tehnologia Java ServerPages (JSP)

Comentariile din paginile JSP sunt de **doua feluri**

- comentarii **JSP propriu-zise**, care **nu ajung in paginile generate dinamic**

```
<%-- Comentariu JSP propriu-zis --%>
```

- comentarii **SGML** (**HTML**, WML, XML), care **ajung in paginile generate dinamic**

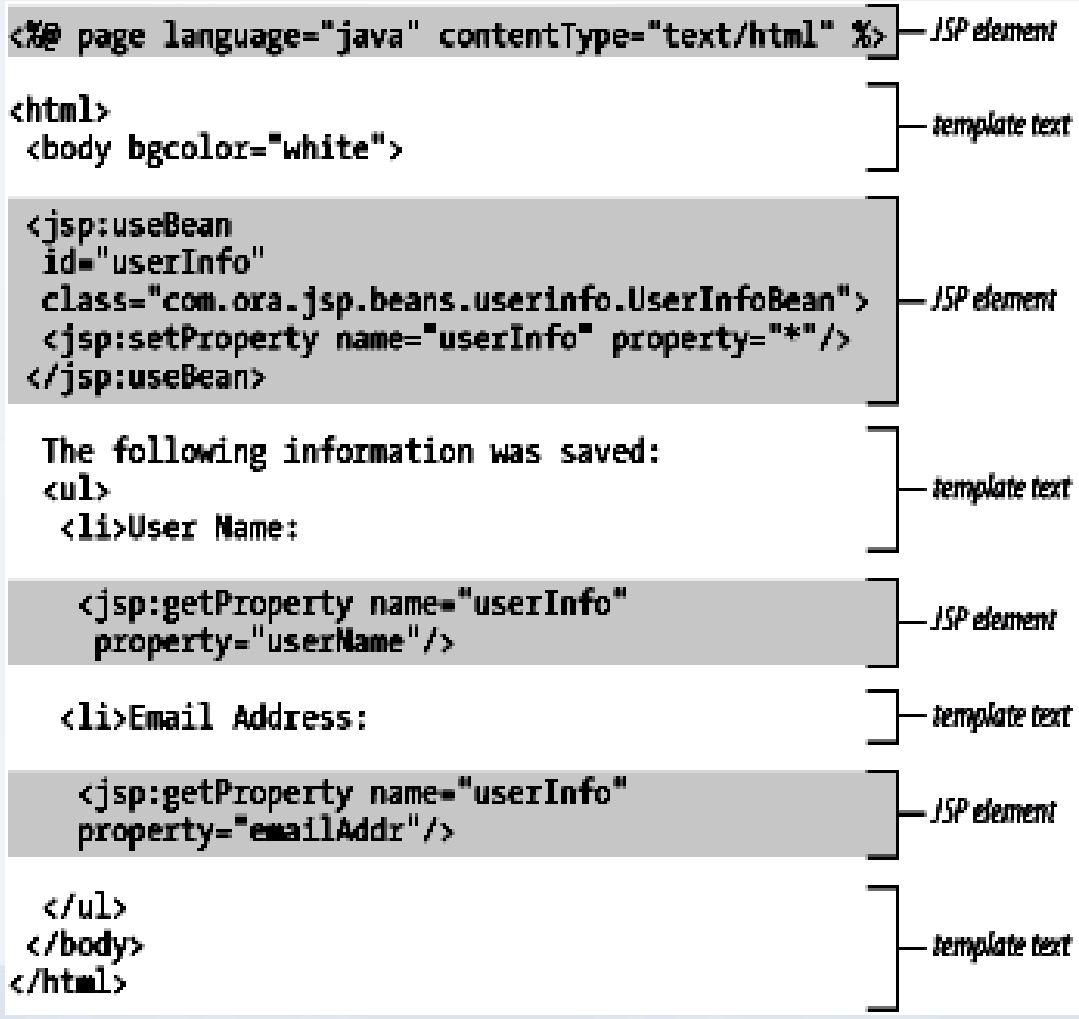
```
<!-- Comentariu HTML -->
```

Elementele constitutive ale **sintaxei JSP** sunt

- **continutul static** sau **tiparul** (***template text*** – HTML, WML, XML) si
- **elementele JSP**
  - **(I) directivele**
  - **(II) elementele de *scripting*** si
  - **(III) actiunile**

Tehnologia Java ServerPages (JSP)

Exemplu de sintaxa JSP



Tehnologia Java ServerPages (JSP)

(I) Directivele

- sunt **elemente JSP** care **furnizeaza informatii globale pentru** faza de **translatie**

- se adreseaza **containerului**

- de exemplu, directiva **page** specifica **atribute ale paginii generate**, cum ar fi

  - **bibliotecile importate**

```
<%-- import clasa biblioteca --%>
<%@ page import="java.util.Date" %>
```

  - **tipul de continut generat**

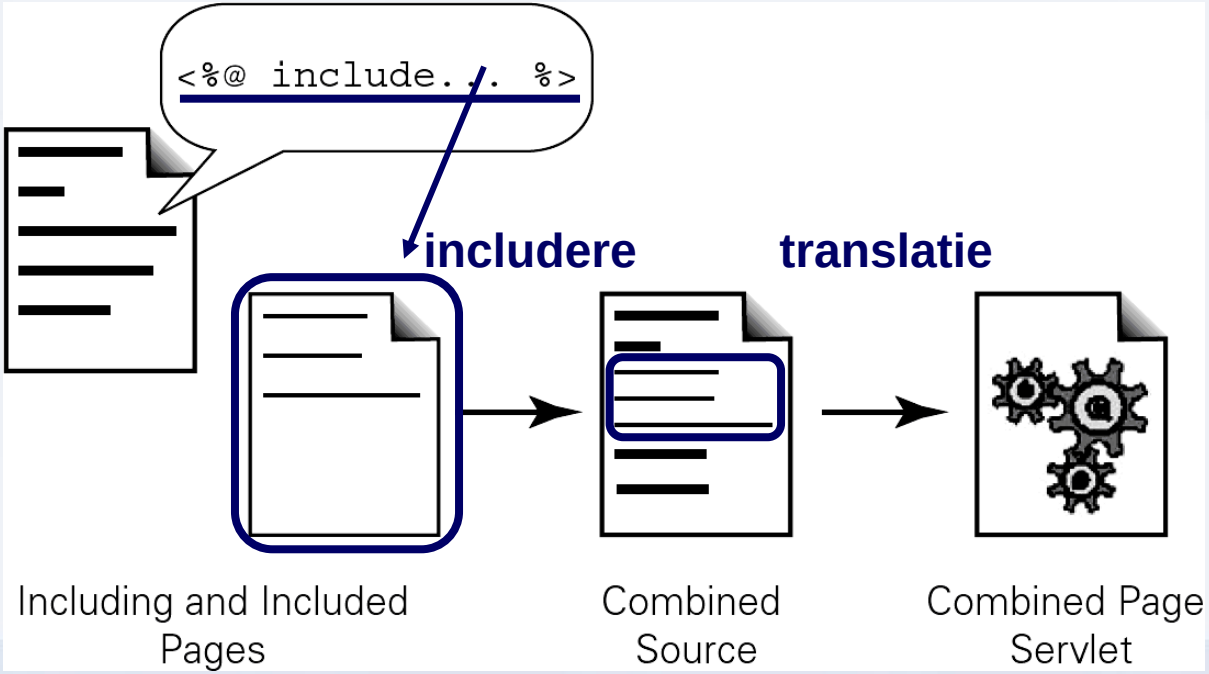
```
<%-- tip de continut generat --%>
<%@ page contentType="text/html" %>
```

Tehnologia Java ServerPages (JSP)

Directiva include

- specifica **includerea statica a continutului unui fisier in continutul celui curent in faza de translatie** (anterioara compilarii)

```
<%-- includere statica (in timpul translatiei) --%>
<%@ include file="altJSP.jsp" %>
```

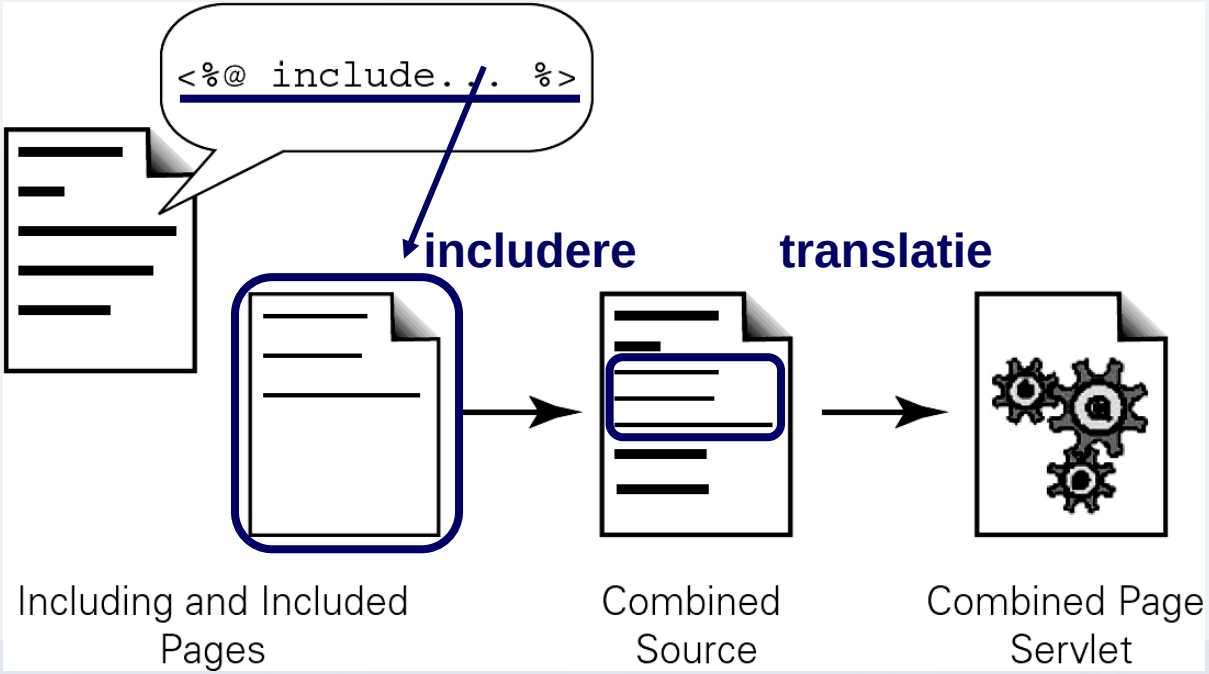


Tehnologia Java ServerPages (JSP)

Directiva include

- **elementele JSP, inclusiv directivele**, pot fi **specificate** si folosind **sintaxa** de tip **XML**

```
<%-- includere statica – format XML --%>
<jsp:directive.include file="altJSP.jsp" %>
```



Tehnologia Java ServerPages (JSP)

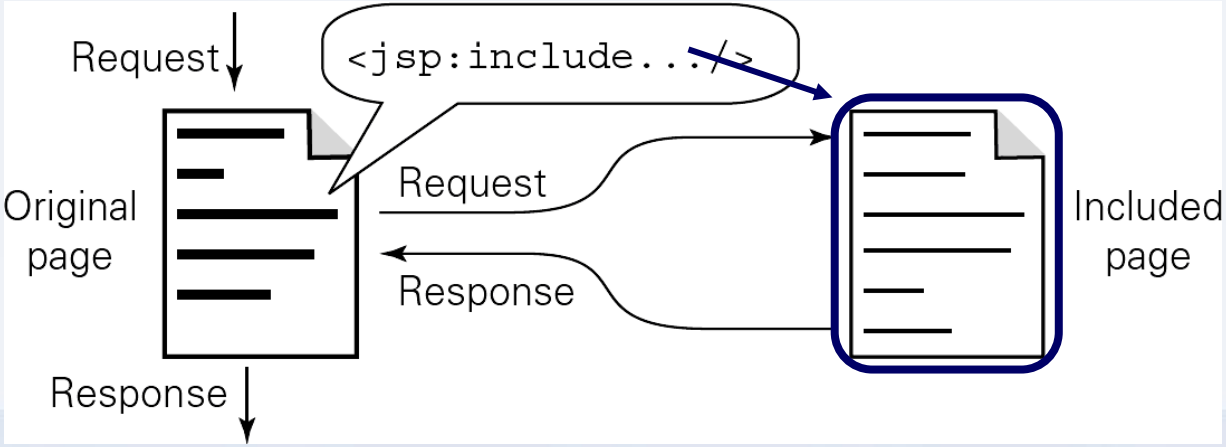
(II) Actiunile

- sunt **elemente JSP** care **furnizeaza informatii pentru** faza de **executie**

- se adreseaza **containerului**

- de exemplu, actiunea **include** specifica **includerea dinamica a continutului unui fisier in continutul celui curent in faza** de **executie** (dupa compilare)

```
<%-- includere dinamica (in timpul executiei) --%>
<jsp:include page="altJSP.jsp" %>
```

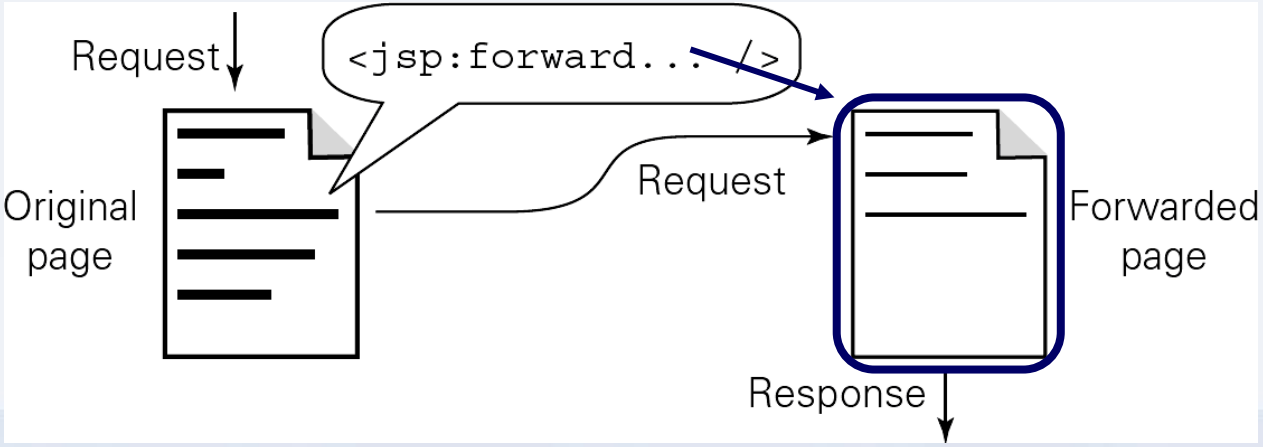


Tehnologia Java ServerPages (JSP)

Actiunea forward

- **JSP-urile** pot **delega executia catre** alte **servlet-uri** si **JSP-uri** folosind o sintaxa de genul

```
<%-- delegare executie --%>
<jsp:forward page="localURL" />
```



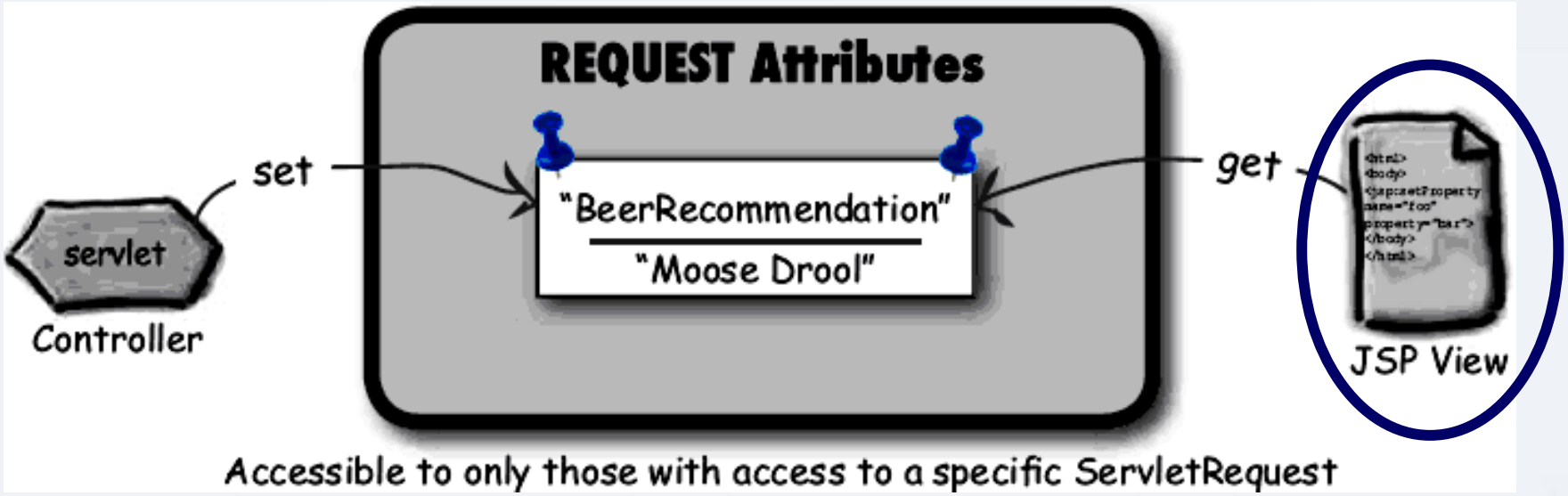
Tehnologia Java ServerPages (JSP)

Sintaxa pentru **atasarea informatiilor** obiectului **cerere**

```
request.setAttribute("raspuns", "Comanda a fost trimisa");
```

Sintaxa pentru **obtinerea informatiilor** de la obiectul **cerere**

```
String raspuns = request.getAttribute("raspuns");
```



Tehnologia Java ServerPages (JSP)

(III) Elementele de *scripting*

- sunt **elemente JSP** cu ajutorul carora **se include cod Java in pagina**

- urmand ca acest cod sa ajunga **nemodificat in codul servletului** obtinut prin **translatie**

- exista **3 categorii** de astfel de elemente

  - **(III.1) declaratiile**

  - **(III.2) expresiile**

  - **(III.3) *scriptlet-urile***

Tehnologia Java ServerPages (JSP)

(III.1) Declaratiile Java

- **introduc metode** (situatie rar intalnita) si
- **campuri** (variabile instanta - care **nu sunt *thread safe***) **ale servlet-ului**
- **sintaxa** include caracterul **!** care poate fi interpretat ca **“atentie”**

```
<%-- declaratie variabila instanta a servlet-ului --%>
<%! private String s; %>

<%-- declaratie metoda a servlet-ului --%>
<%! public String getS() {return s;} %>
```

Se observa ca **liniile de cod se termina cu ;**

Tehnologia Java ServerPages (JSP)

(III.2) Expresiile Java

- ajung sa fie **evaluate in codul servlet-ului** obtinut prin translatie

- spre deosebire de declaratii si *scriptlet-uri*, **sintaxa nu include caracterul ;**

```
<%= 2*a*b %>
```

(III.3) Un *scriptlet*

- este o **secventa de instructiuni Java** care ajung sa fie **incluse nemodificate in codul servlet-ului** obtinut prin translatie

- **variabilele declarate in interiorul lui** sunt **locale** (si deci ***thread safe***)

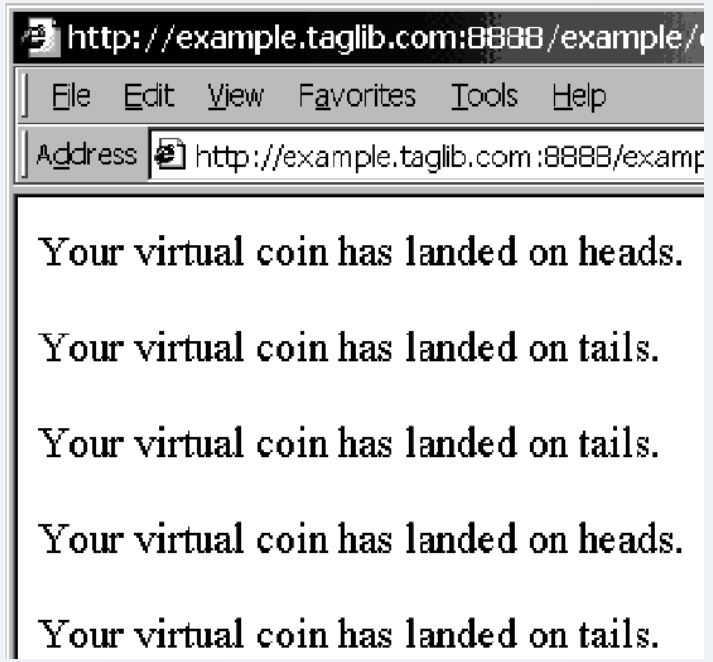
```
<%  String user = null;        // variabila locala
    username = request.getParameter("user");
%>
```

obiect implicit

Tehnologia Java ServerPages (JSP)

Exemplu de ***scriptlet*** care **utilizeaza decizii** si **bucle**

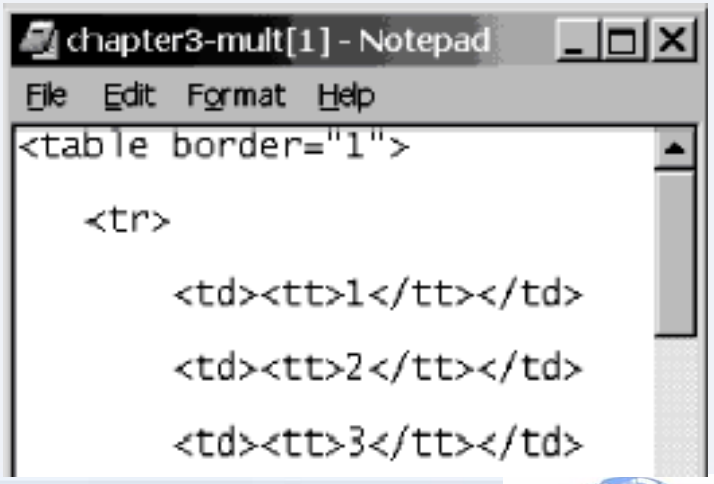
```
<html>
   <body>
      <% for (int i = 0; i < 5; i++) { %>
      <p> Your virtual coin has landed on

      <% if (Math.random() < 0.5) { %>
      heads.

      <% } else { %>
      tails.

      <% } %>
      </p>

      <% } %>

   </body>
</html>
```



Tehnologia Java ServerPages (JSP)

Exemplu de ***scriptlet*** care **genereaza dinamic un tablou**

```
<table border="1">
<% for (int row = 1; row < 11; row++) { %>
   <tr>
   <% for (int column = 1; column < 11; column++) { %>
      <td><tt><%= row * column %></tt></td>
   <% } %>
   </tr>
<% } %>
</table>
```



Tehnologia Java ServerPages (JSP)

Alternativa non-Java la utilizarea *scriptlet-urilor*

- ***tag-urile* JSP standard** (implementate ca clase Java)
  - au **rolul** de **a inlocui cat mai mult** din **codul Java cu elemente asemanatoare** celor **HTML / XML**

Using scriptlets

```
<html>
   <head>
     <title>simple  example<title>
   </head>
   <body>
        <%
           for(int i=0; i<5; i++) {
        %>
              <%= i %> <br/>

         <% } %>
   </body>
</html>
```

The above JSP code is hard to read and maintain.

Using JSTL tags

```
<%@ taglib prefix="c"
                uri="http//java.sun.com/jstl/core">

<html>
  <head><title>simple example<title></head>
  <body>
    <c:forEach var="i" begin="1" end="5" step="1">
       <c:out value="${i}"> <br/>
    </c:forEach>
   </body>
</html>
```

The above JSP code consists entirely of HTML & JSTL tags (in bold).

Tehnologia Java ServerPages (JSP)

Principalele biblioteci de *tag-uri* standard

Description	Tag Prefix (recommended)	Example

Core Tag Library – looping, condition evaluation, basic input, output etc.	c	`<c:out value="${hello}" />`
		`<c:if test="${param.name='Peter'}"> …`
		`<c:forEach items="${addresses}" var="address"> …`

Formatting/Internationalization Tag Library – parse data such as number, date, currency etc	fmt	`<fmt:formatNumber value="${now.time}" />`

XML Tag Library – tags to access XML elements.	x	`<x:forEach select="$doc/books/book" var="n">`
		`  <x:out select="$n/title" />`
		`</x:forEach>`

Database Tag Library – tags to	sql	`<sql:query var="emps" sql="SELECT * FROM Employee">`

Tehnologia Java ServerPages (JSP)

Principalul avantaj al JSP-urilor consta in **introducerea *template-urilor* de continut static**

- posibil de creat in **formate non-HTML**: **WML**, **XML**

care **pot fi realizate de dezvoltatori specializati in proiectarea interfetelor Web**

Exemplu WML

directiva

actiune

***scripting* (expresie)**

```
<%@ page contentType="text/vnd.wap.wml;charset=UTF-8"
         import="java.text.*, java.util.*" %>
<?xml version="1.0"?>
<%
   SimpleDateFormat df = new SimpleDateFormat("hh:mm a");
%>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
         "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card id="time" title="Time">
      <p>It's <%= df.format(new Date()) %>.</p>
      <p>(Do you know where your laptop is?)</p>
   </card>
</wml>
```

Tehnologia Java ServerPages (JSP)

Deoarece

- **partea de prelucrare a informatiei** necesara generarii de continut dinamic este **mai greu de scris in JSP**
- si **este preferabil sa fie separata pentru a fi scrisa de programatori Java**

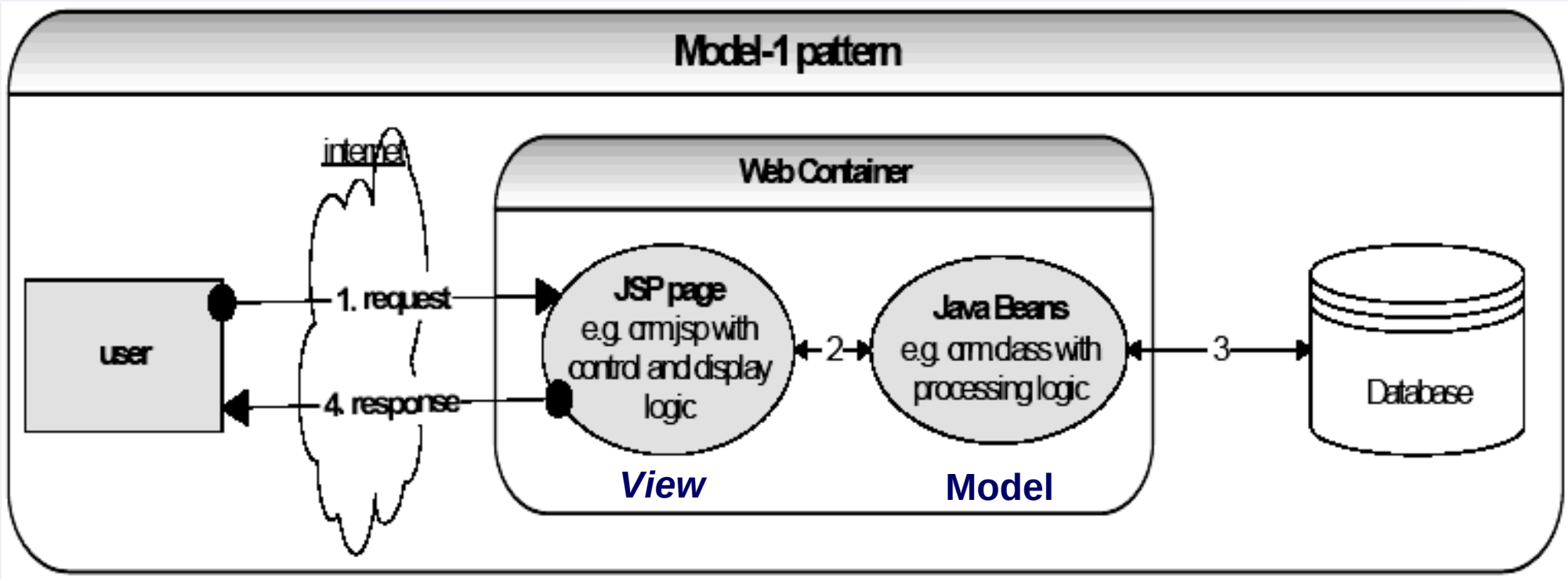
s-a trecut rapid

- **de la** lucrul **exclusiv** cu **pagini JSP** (**arhitectura** numita **"model-0"**)
- **la delegarea sarcinilor** de **stocare** si **prelucrare** catre **coduri Java** care pot fi
  - **clase Java** clasice (**POJO – *Plain Old Java Objects***) sau
  - **componente JavaBeans**

Arhitectura care a rezultat poarta numele de **"model-1"**

Tehnologia Java ServerPages (JSP)

Arhitectura "model-1" – pagina JSP ("*view*") si clasa "model"



Arhitectura MVC (*Model-View-Controller*)

Tehnologia Java ServerPages (JSP)

Pentru aplicatii Web mari sunt **necesare**

- o **mai buna gestiune a sistemului**
- o **mai buna separare a responsabilitatilor** de **dezvoltare** si
- o **mai buna specializare a tehnologiilor** utilizate

Acest avans a fost obtinut prin

- **introducerea arhitecturii** care poarta numele de **"model-2"** sau **MVC (*model-view-controler*)**
- in care se folosesc **3 categorii de componente** realizate cu **tehnologii diferite**
  - **controlerul** (in general ***servleturi***)
  - **modelul** (in general **componente JavaBeans)**
  - ***view-ul*** (**prezentarea**, in general **pagini JSP**)

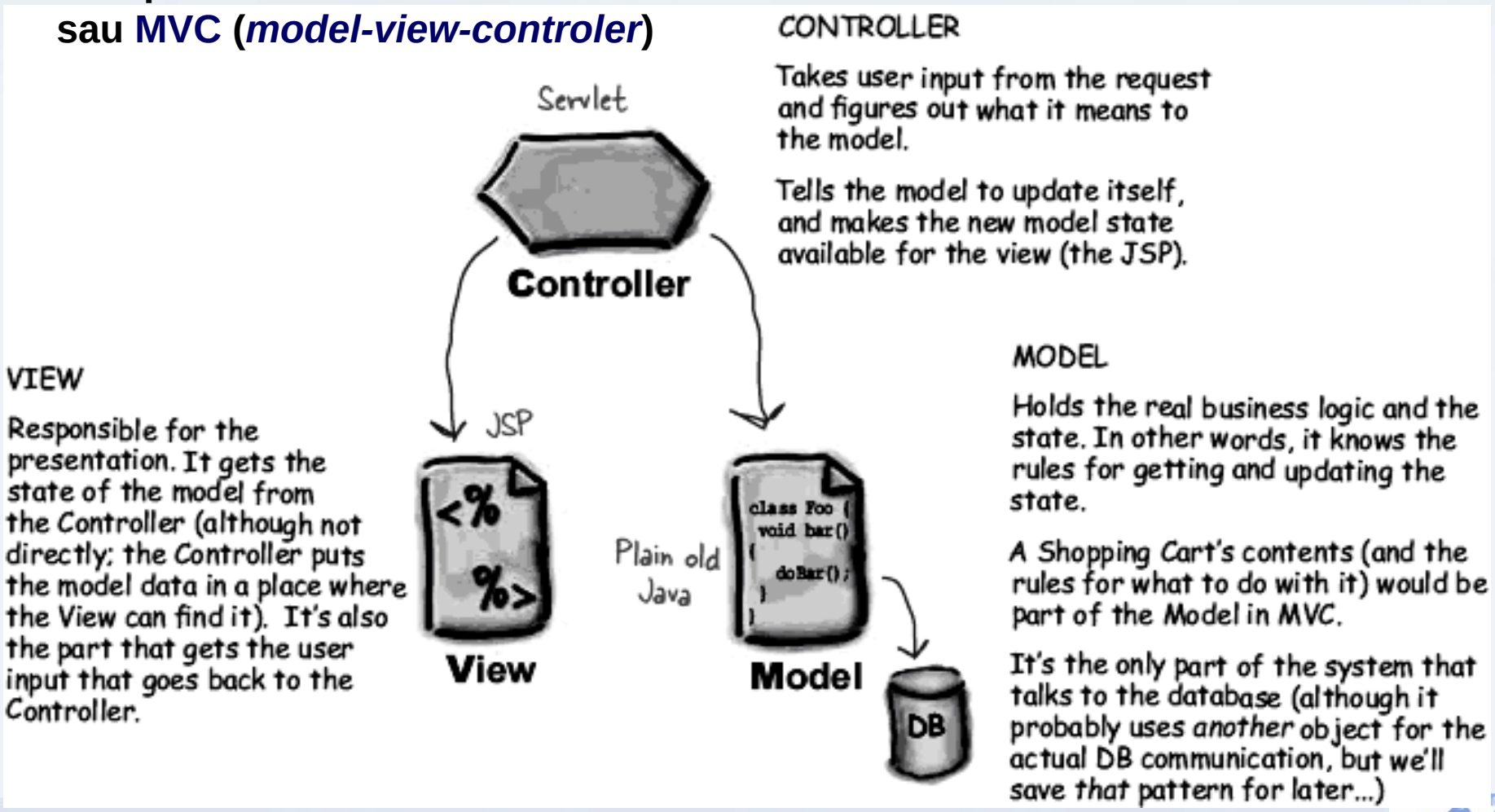
Tehnologia Java ServerPages (JSP)

Componentele arhitecturii "model-2" sau MVC (*model-view-controler*)

- **controlerul**
  - realizat in general cu un **servlet**
  - **primeste cererile**, **apeleaza la model** pentru a realiza **actualizarea datelor de stare** si **prelucrarile necesare** aplicatiei, si **in final deleaga prezentarea** catre componente specializate (***view***)
- **modelul**
  - realizat in general sub forma de **componente JavaBeans**
  - se ocupa de **pastrarea datelor (starii) si** de **prelucrarile necesare**
- ***view-ul* (prezentarea)**
  - este format din **componente de prezentare** (in general **pagini JSP**)
  - care sunt **folosite pentru a genera continutul raspunsului**

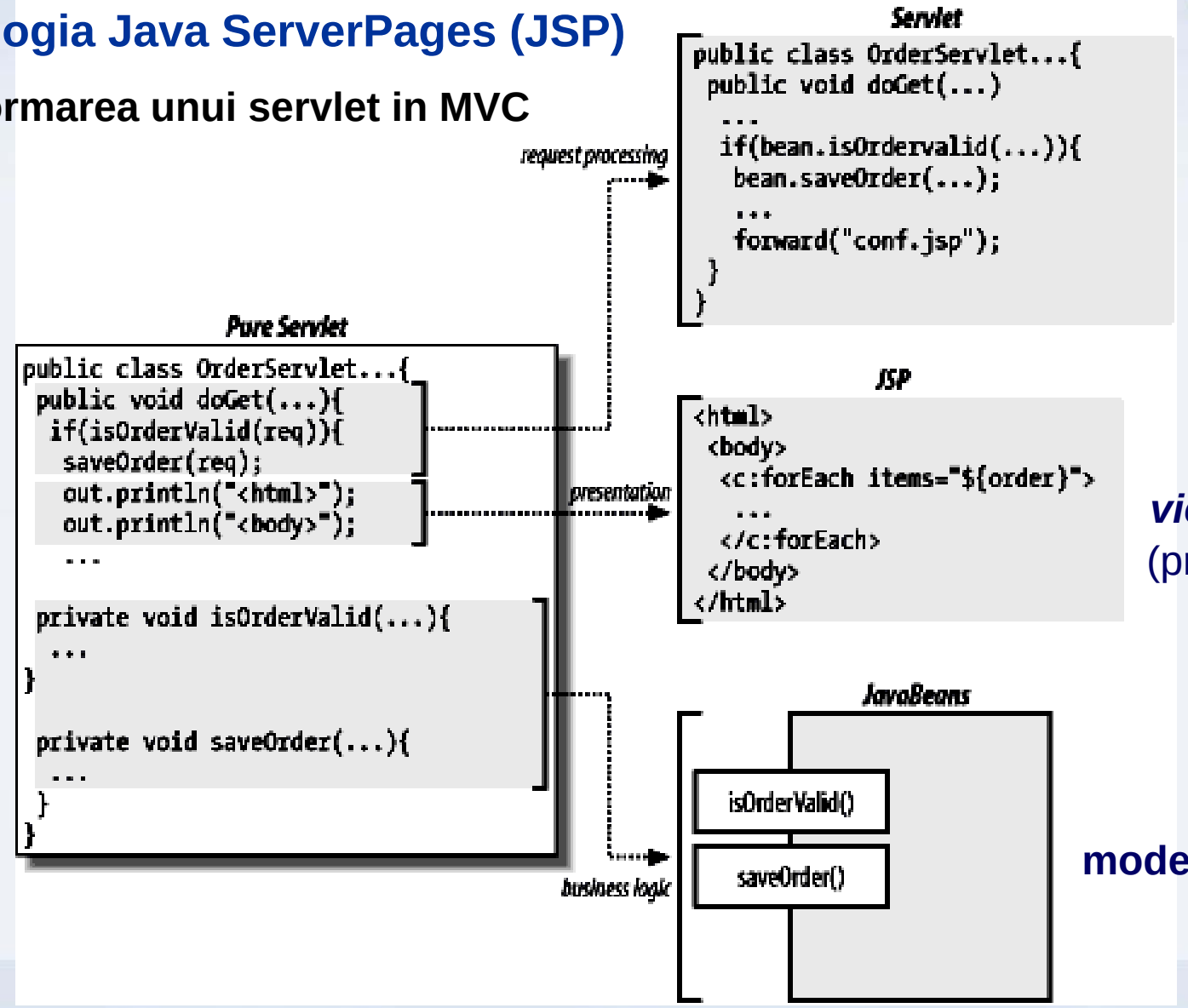
Tehnologia Java ServerPages (JSP)

Componentele arhitecturii "model-2" sau MVC (*model-view-controler*)



Tehnologia Java ServerPages (JSP)

Transformarea unui servlet in MVC



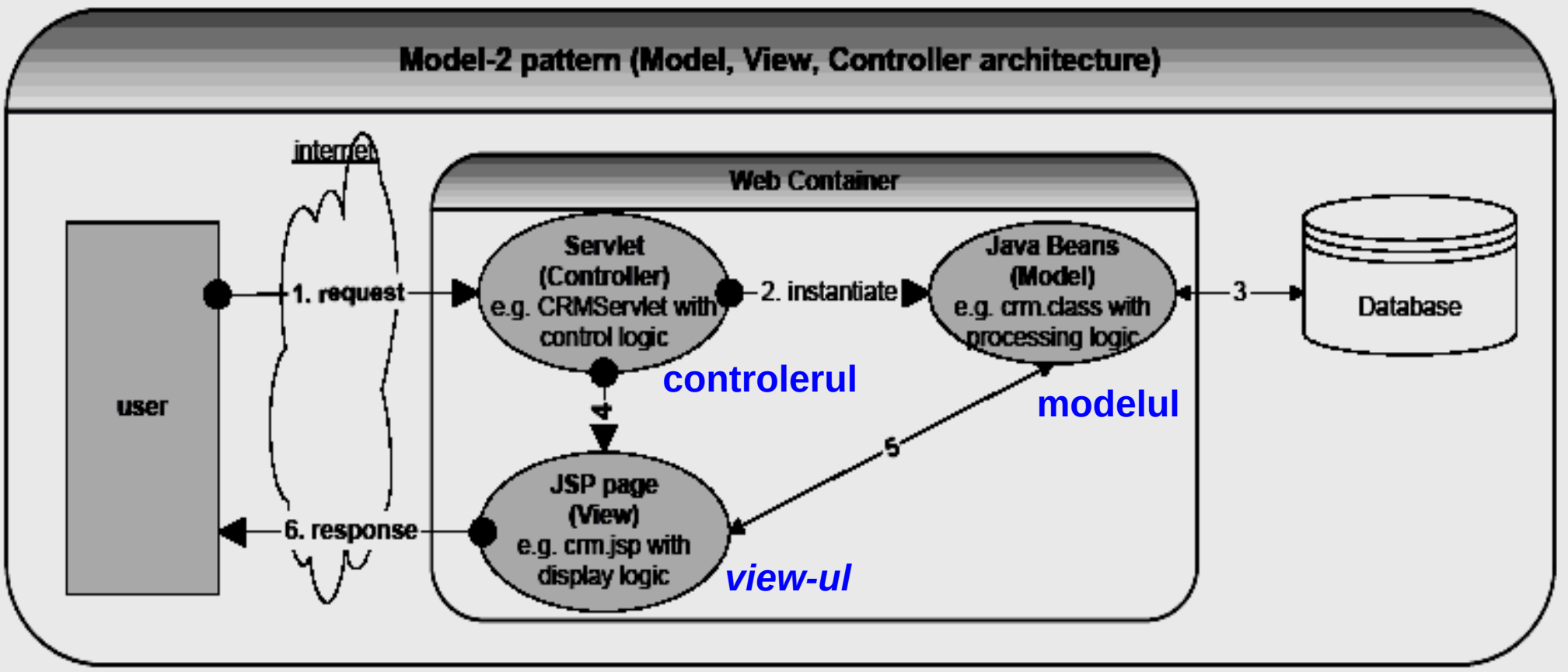
controlerul

view-ul (prezentarea)

modelul

Tehnologia Java ServerPages (JSP)

Dezvoltarea unei astfel de aplicatii poate fi realizata modular



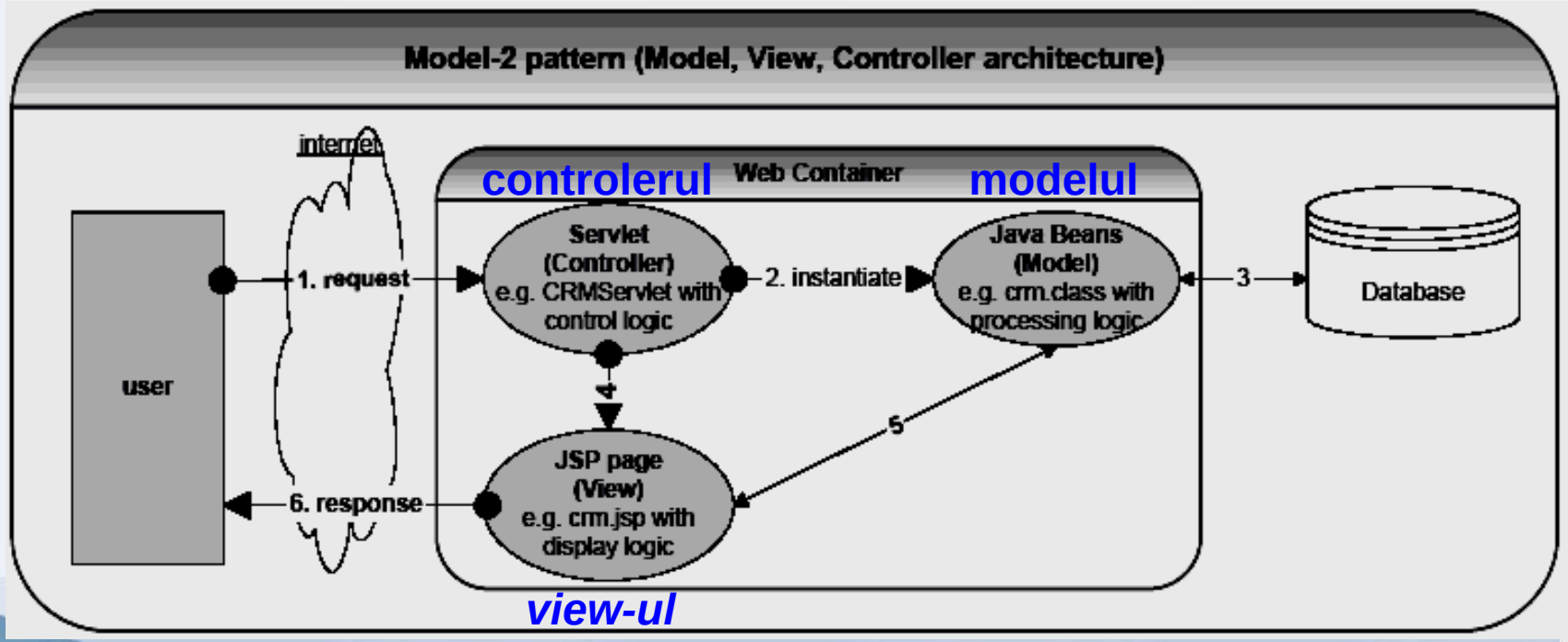
Tehnologia Java ServerPages (JSP)

Dezvoltarea unei astfel de aplicatii poate fi realizata modular de catre

- **specialisti** in ***servlet-uri*** – dezvoltatorii **controlerului**

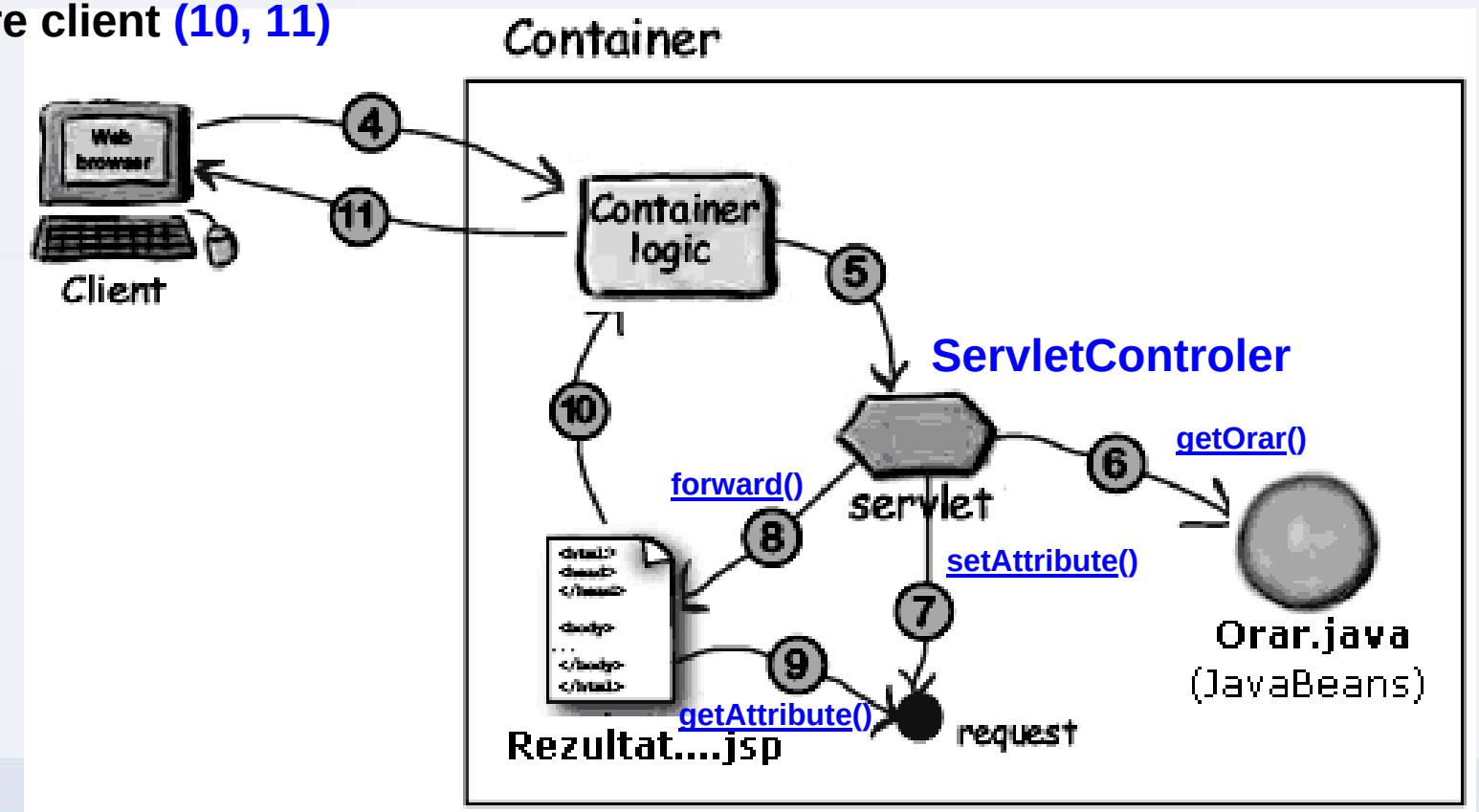
- **specialisti** in **proiectare Web** si **scripting JSP** (inclusiv utilizare si dezvoltare de **biblioteci de tag-uri**) – dezvoltatorii ***view-ului*** (prezentarii)

- **specialisti** in **proiectare si programare OO** (orientata spre obiecte) – dezvoltatorii **modelului**



Tehnologia Java ServerPages (JSP)

Exemplu de MVC - **un servlet** cu rol de **controler** va **primi cererea HTTP (4, 5)**, va **accesa informatiile din Orar (6)** si le **va pasa (7) prin intermediul unui atribut nou al cererii catre o pagina JSP (9) selectata in functie de valoarea unui parametru din formularul cererii (8), iar pagina va genera raspunsul catre client (10, 11)**



Tehnologia Java ServerPages (JSP)

Exemplu de MVC – controlerul

```
package controler;

import model.Orar;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletControler extends HttpServlet {

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

      // Transformarea obiectului orar in atribut al sesiunii curente pentru
      // salvarea starii lui
      HttpSession ses = request.getSession();
      Orar orar = (Orar) ses.getAttribute("orar");
      if (orar == null) { // Daca nu exista orarul salvat ca atribut al sesiunii
         orar = new Orar();
         ses.setAttribute("orar", orar);
      }

      // Obtinerea parametrilor introdusi de utilizator in formular
      int zi = Integer.parseInt(request.getParameter("zi"));
```

Controlerul primeste cererea HTTP (4, 5)

Parametru din formularul cererii folosit in selectia *view-ului* (6)

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – controlerul

Selectie pagina JSP (8)

```
RequestDispatcher view;

// Daca serviciul cerut e obtinere orar
if (request.getParameter("serviciu").equals("getOrar")) {
   view = request.getRequestDispatcher("RezultatObtinereOrar.jsp");
}

// Daca serviciul cerut e modificare orar
else if (request.getParameter("serviciu").equals("setOrar")) {
   String modificare = request.getParameter("modificare");
   orar.setOrar(zi, modificare);
   view = request.getRequestDispatcher("RezultatModificareOrar.jsp");
}

// Daca serviciul cerut nu e recunoscut
else {
   view = request.getRequestDispatcher("ServiciuNeimplementat.jsp");
}

request.setAttribute("raspuns", orar.getOrar(zi));

view.forward(request, response);
}
```

Accesarea informatiilor din model - Orar (6)

Pasarea informatiilor catre pagina JSP selectata (7) prin intermediul unui atribut nou al cererii

Delegarea prezentarii catre ***view***

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – *view-ul 1*

RezultatObtinereOrar.jsp

```
<%@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>
        <%@ include file="PaginaHTMLAcces.html" %>

        <b>Orarul cerut:</b>
        <br> <%=request.getAttribute("raspuns")%>
    </body>
</html>
```

Pagina genereaza raspunsul catre client (10, 11)

Pasarea informatiilor catre pagina JSP (9)

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – *view-ul 2*

RezultatModificareOrar.jsp

```
<%@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>
        <%@ include file="PaginaHTMLAcces.html" %>

        <b>Modificarea ceruta:</b>
        <br> <%=request.getAttribute("raspuns")%>
    </body>
</html>
```

Pagina genereaza raspunsul catre client (10, 11)

Pasarea informatiilor catre pagina JSP (9)

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – *view-ul 3*

ServiciuNeimplementat.jsp

```
<%@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>
        <%@ include file="PaginaHTMLAcces.html" %>

        <b>Serviciul cerut nu este implementat</b>
        <br>
    </body>
</html>
```

Pagina genereaza raspunsul catre client (10, 11)

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – *view-ul*

PaginaHTMLAcces.html

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<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="PaginaJSP.jsp">Pagina JSP initiala (Model 1)</a>
         <hr>
    </body>
</html>
```

Pagina HTML inclusa

Tehnologia Java ServerPages (JSP)

Exemplu de MVC – clasa model care ofera serviciul *business*

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

Tehnologia Java ServerPages (JSP)

Versiunea “model-1” – doar pagina JSP si clasa model

```
<%@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 JSP acces orar (Model 1)</h1>
        <hr><form name="input" action="PaginaJSP.jsp" 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>
```

Tehnologia Java ServerPages (JSP)

Versiunea “model-1” – doar pagina JSP si clasa model

```
<jsp:useBean scope="session" id="orar" class="model.Orar" />

<%
try {
    int zi = Integer.parseInt(request.getParameter("zi"));

    // Daca serviciul cerut e obtinere orar
    if (request.getParameter("serviciu").equals("getOrar")) {
        out.println("<b>Orarul cerut:</b> <br>" + orar.getOrar(zi));
    }

    // Daca serviciul cerut e modificare orar
    else if (request.getParameter("serviciu").equals("setOrar")) {
        String modificare = request.getParameter("modificare");
        orar.setOrar(zi, modificare);
        out.println("<b>Modificarea ceruta:</b> <br>" + orar.getOrar(zi));
    }
} catch (NumberFormatException ex) {}
%>

</body>
</html>
```

Obtinerea accesului la model - Orar

Parametru din formularul cererii

Accesarea informatiilor din model

Pagina genereaza raspunsul catre client