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

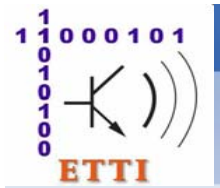
- 2.1. Elemente de baza. Tipuri de date referinta. Clase de biblioteca
- 2.2. Clase pentru fluxuri de intrare-iesire (IO)

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)



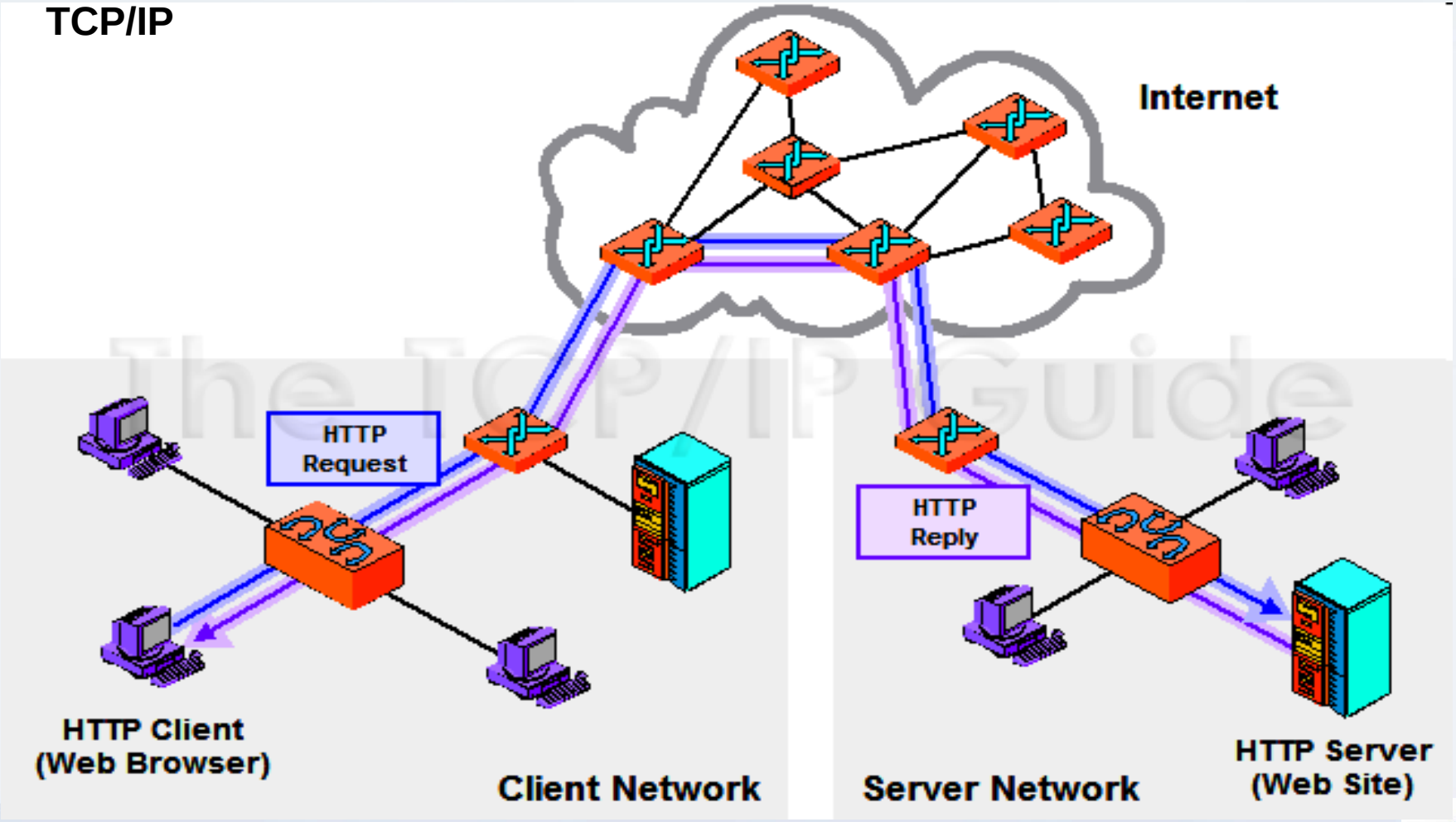
4. Tehnologii Java de programare a aplicatiilor Web (EE) Java

4.4. Tehnologii server. Tehnologia Java Servlet

4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

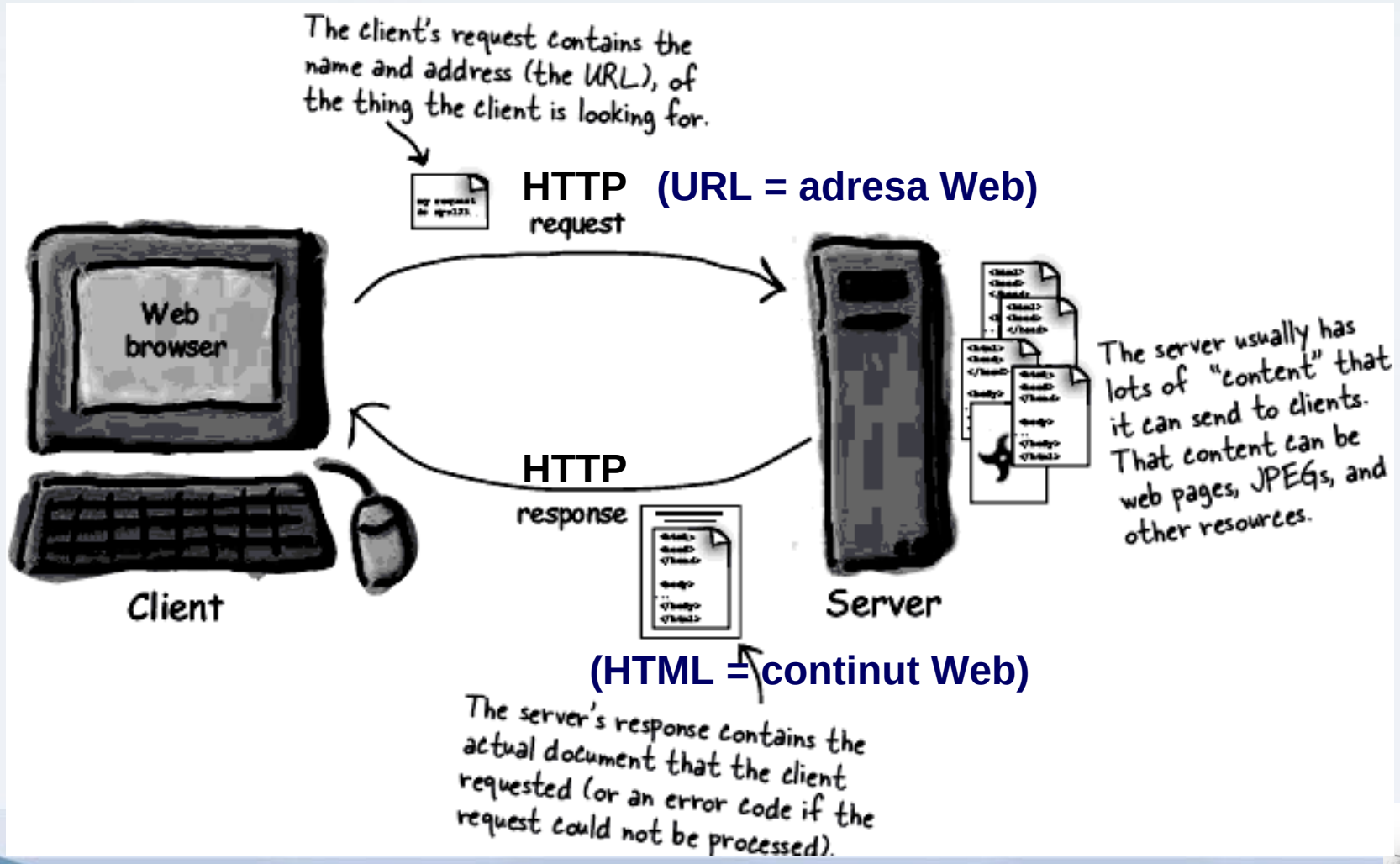
Comunicatiile client-server HTTP (*HyperText Transport Protocol*) peste TCP/IP



4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Comunicatiile client-server HTTP (*HyperText Transport Protocol*)

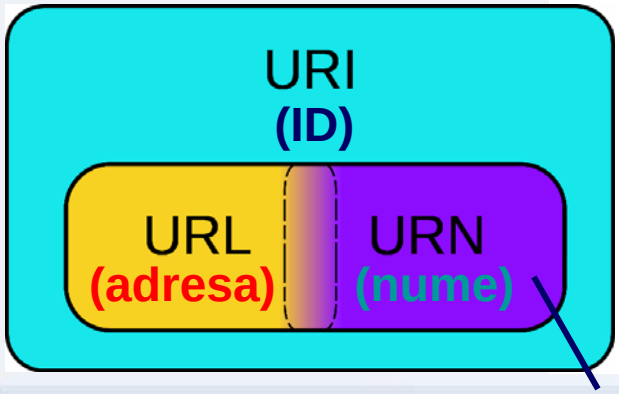


4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Comunicatiile client-server HTTP (*HyperText Transport Protocol*)

URL (*Uniform Resource **Locator***) este un **subset al URI** (*Uniform Resource **Id***) care **identifica o resursa**, si ofera un **mijloc de a o localiza** descriind **mecanismul de a o accesa** si **locatia**



(**ISBN,** de exemplu)

Protocol: Tells the server which communications protocol (in this case HTTP) will be used.

Port: This part of the URL is optional. A single server supports many ports. A server application is identified by a port. If you don't specify a port in your URL, then port 80 is the default, and as luck would have it, that's the default port for web servers.

Resource: The name of the content being requested. This could be an HTML page, a servlet, an image, PDF, music, video, or anything else the server feels like serving. If this optional part of the URL is left out, most web servers will look for index.html by default.

`http://www.wickedlysmart.com:80/beeradvice/select/beer1.html`

Server: The unique name of the physical server you're looking for. This name maps to a unique IP address. IP addresses are numeric and take the form "xxx.yyy.zzz.aaa". You can specify an IP address here instead of a server name, but a server name is a lot easier to remember.

Path: The path to the location, on the server, of the resource being requested. Because most of the early servers on the web ran Unix, Unix syntax is still used to describe the directory hierarchies on the web server.

4.4. Tehnologii server. Java Servlet

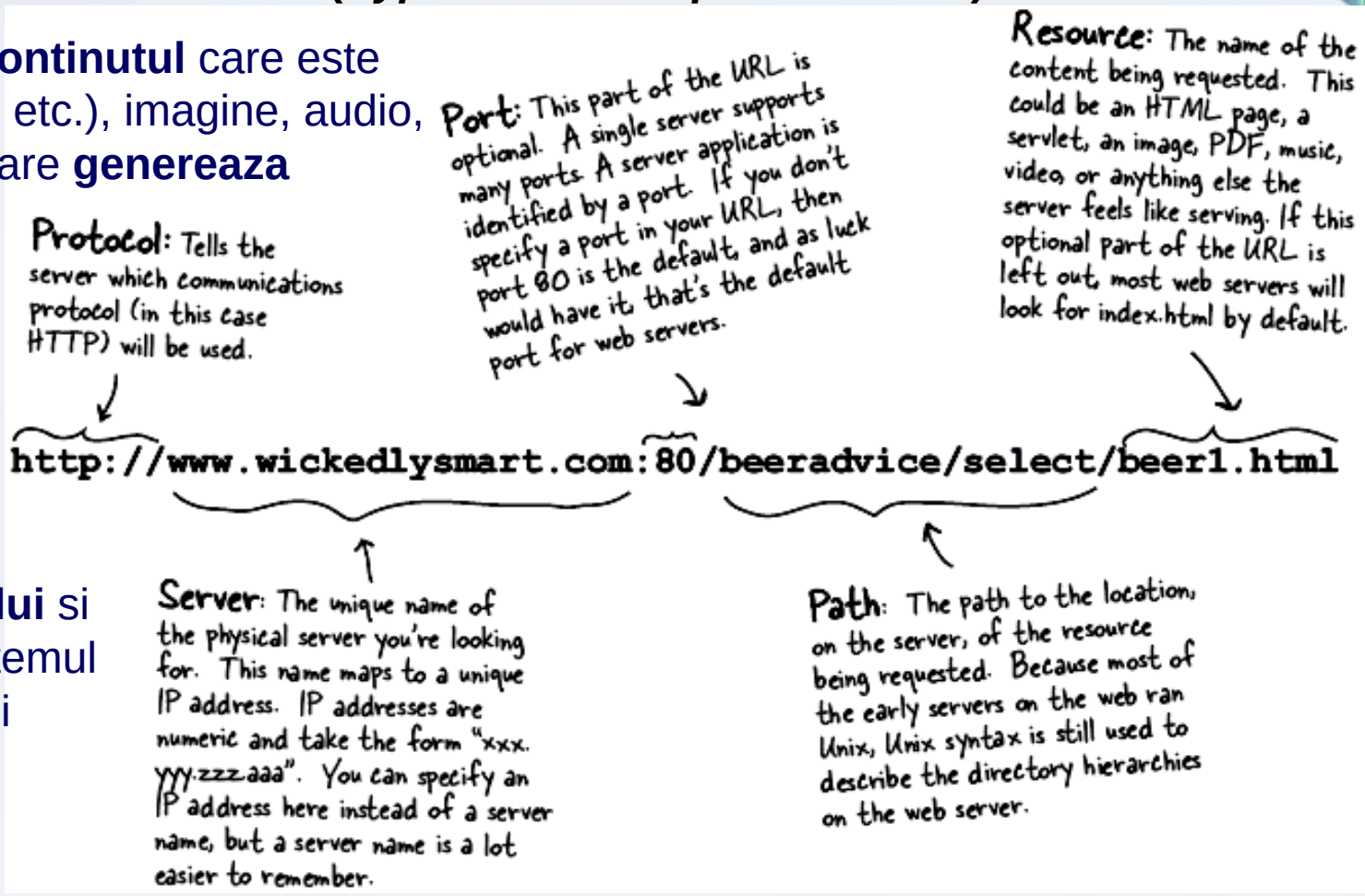
Modelul de comunicatie client-server (tranzactional)

Comunicatiile client-server HTTP (*HyperText Transport Protocol*)

Resursa inseamna **continutul** care este **pagina HTML** (WML, etc.), imagine, audio, video sau **program** care **genereaza** continutul propriu-zis

Protocolul si **portul** asociat specifica **mecanismul de acces**

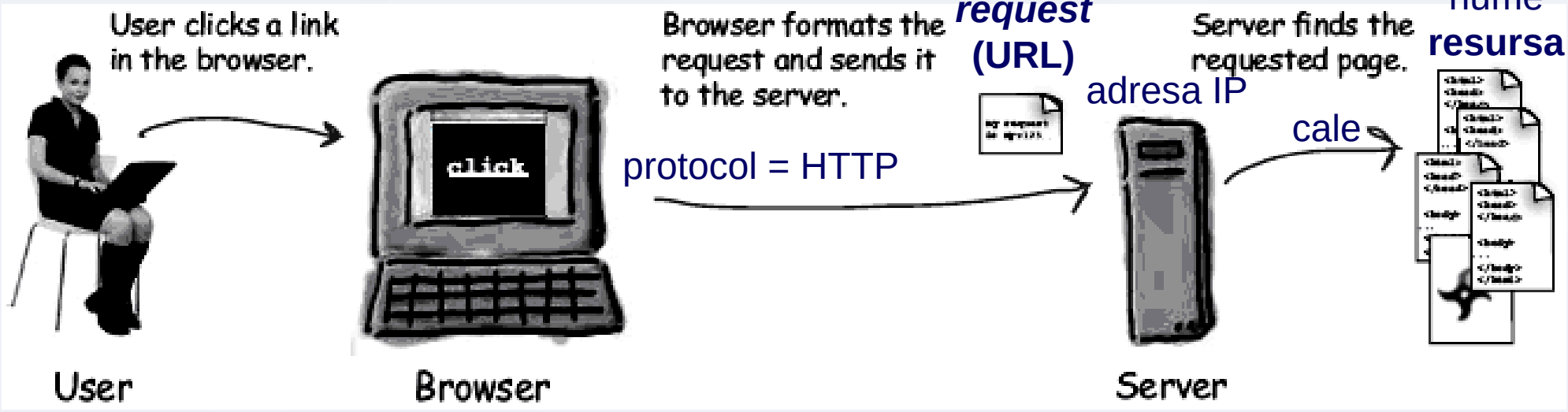
Adresa IP a serverului si **calea resursei** in sistemul de fisiere al serverului specifica **locatia**



4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Cererea HTTP contine “adresa Web a resursei” (URL)



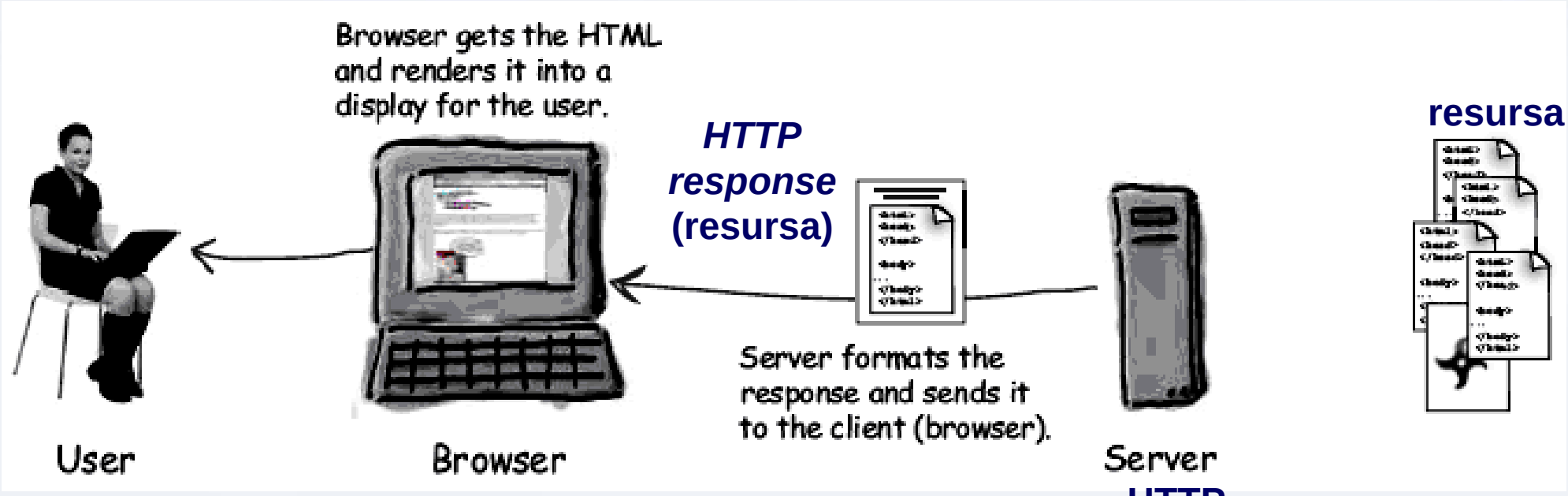
Agent client
= Client HTTP
+ Interpretor resursa

URL = protocol **+** adresa IP **+** cale **+** nume **resursa**

4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Raspunsul HTTP contine “resursa” (continut HTML, etc.)



Agent client

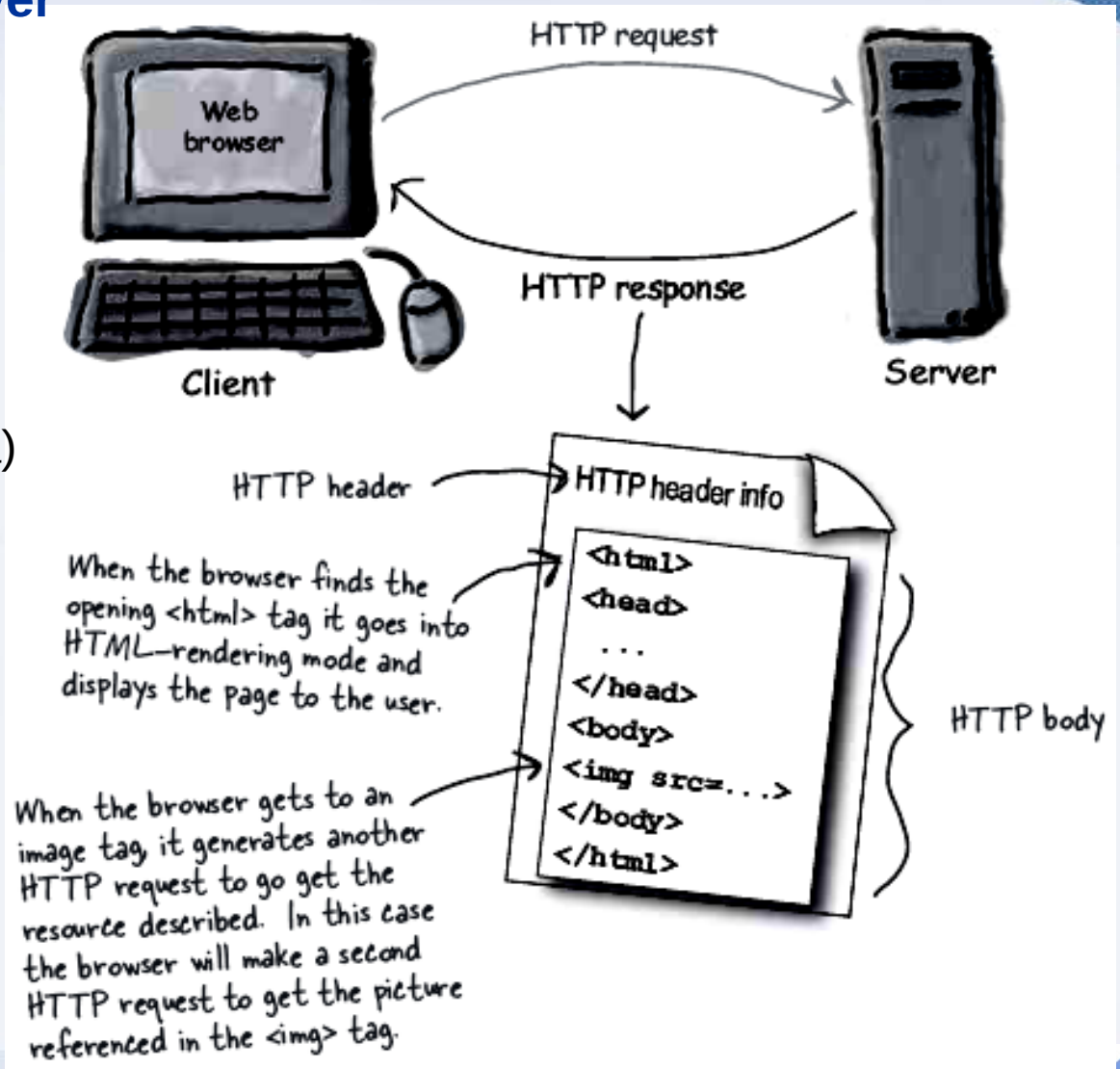
= Client HTTP

+ Interpretor resursa

Modelul de client-server

Prelucrarea raspunsului HTTP poate conduce la

generarea altei cereri HTTP pentru a **obtine resurse** care **completeaza resursa primita** (fiind “citate” in ea)



4.4. Tehnologii server. Java Servlet

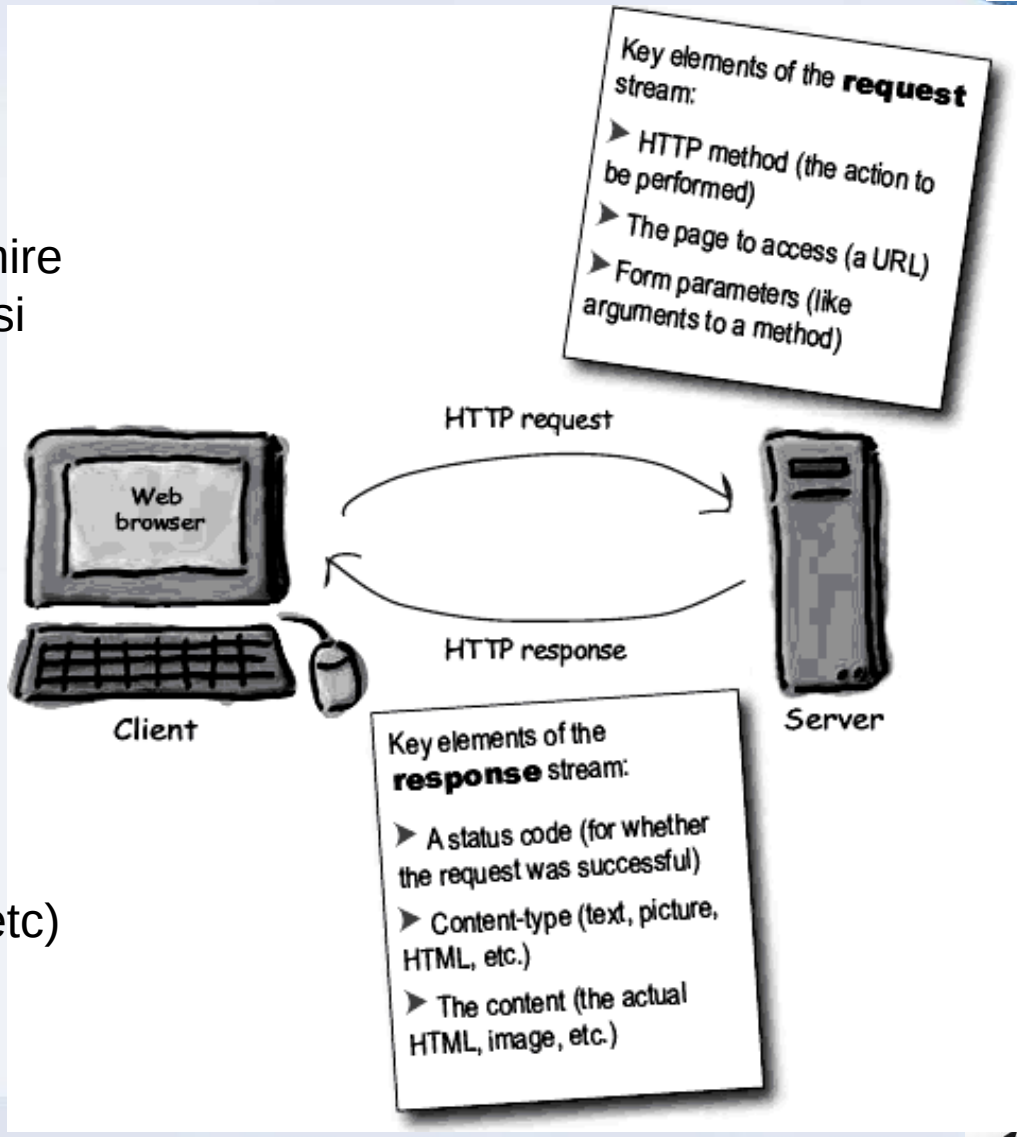
Modelul client-server

Elemente ale cererii HTTP

- **URL** (adresa resursei)
- **metoda** (**GET** – doar primire continut, **POST** – trimitere si primire continut)
- **parametrii formularului** (in cazul resurselor dinamice = programe generatoare de continut)

Elemente raspuns HTTP

- **cod stare** (200 = succes)
- **tip continut** (text,HTML,etc)
- **continutul propriu-zis** (HTML, imagine, etc.)



4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Elemente de marcare din limbajul HTML (*HyperText Markup Language*)

Tag	Description
`<!-- -->`	where you put your *comments*
`<a>`	*anchor* - usually for putting in a hyperlink
`<align>`	*align* the contents left, right, centered, or justified
`<body>`	define the boundaries of the document's *body*
` `	a *line break*
`<center>`	*center* the contents
`<form>`	define a *form* (which usually provides input fields)
`<h1>`	the first level *heading*
`<head>`	define the boundaries of the document's *header*
`<html>`	define the boundaries of the HTML *document*
`<input type>`	defines an *input widget* to a form

(Technically, the and <align> tags have been deprecated in HTML 4.0, but we're using them in some of our examples because it's simpler to read than the alternative, and you're not here to learn HTML anyway.)

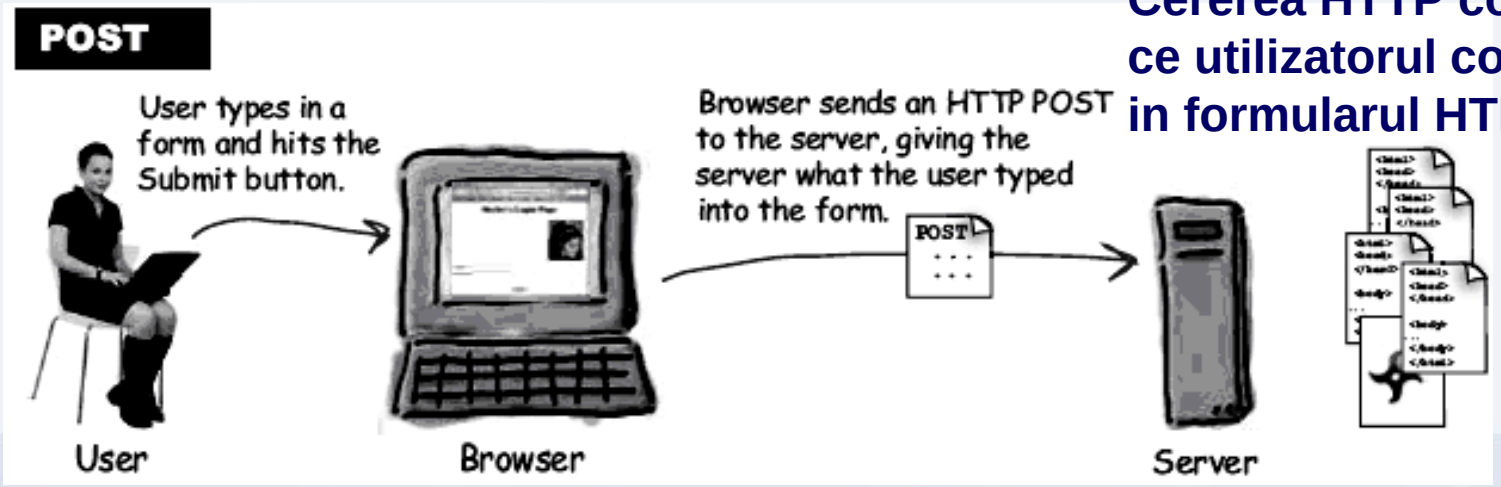
Elemente utilizate in formulare HTML

4.4. Tehnologii server. Java Servlet

Modelul de comunicatie client-server (tranzactional)

Exemple de **cereri HTTP** bazata pe **metodele GET** si **POST**

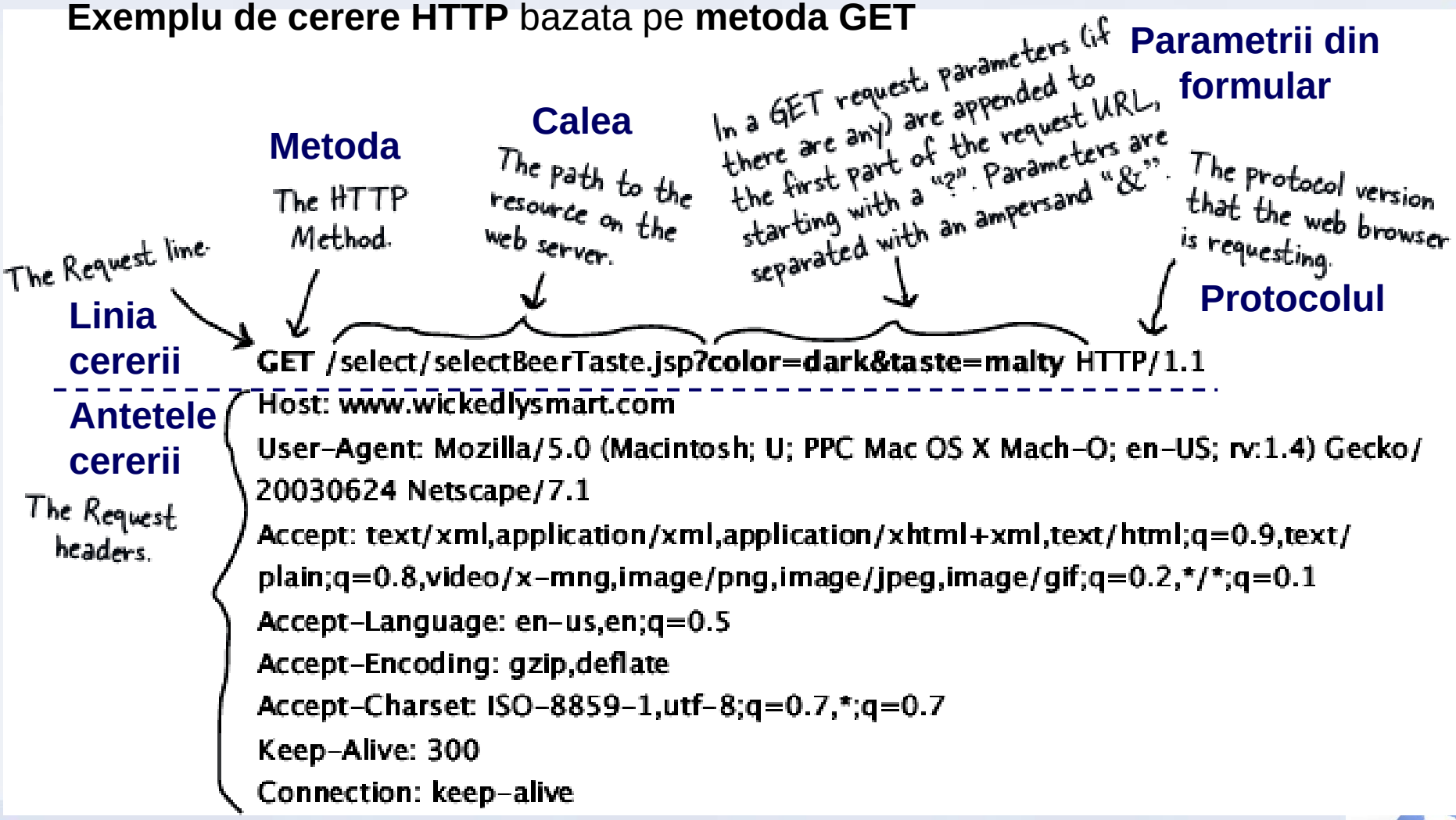
Cererea HTTP specifica doar resursa ceruta



Cererea HTTP contine ceea ce utilizatorul completeaza in formularul HTML

Modelul de comunicatie client-server (tranzactional)

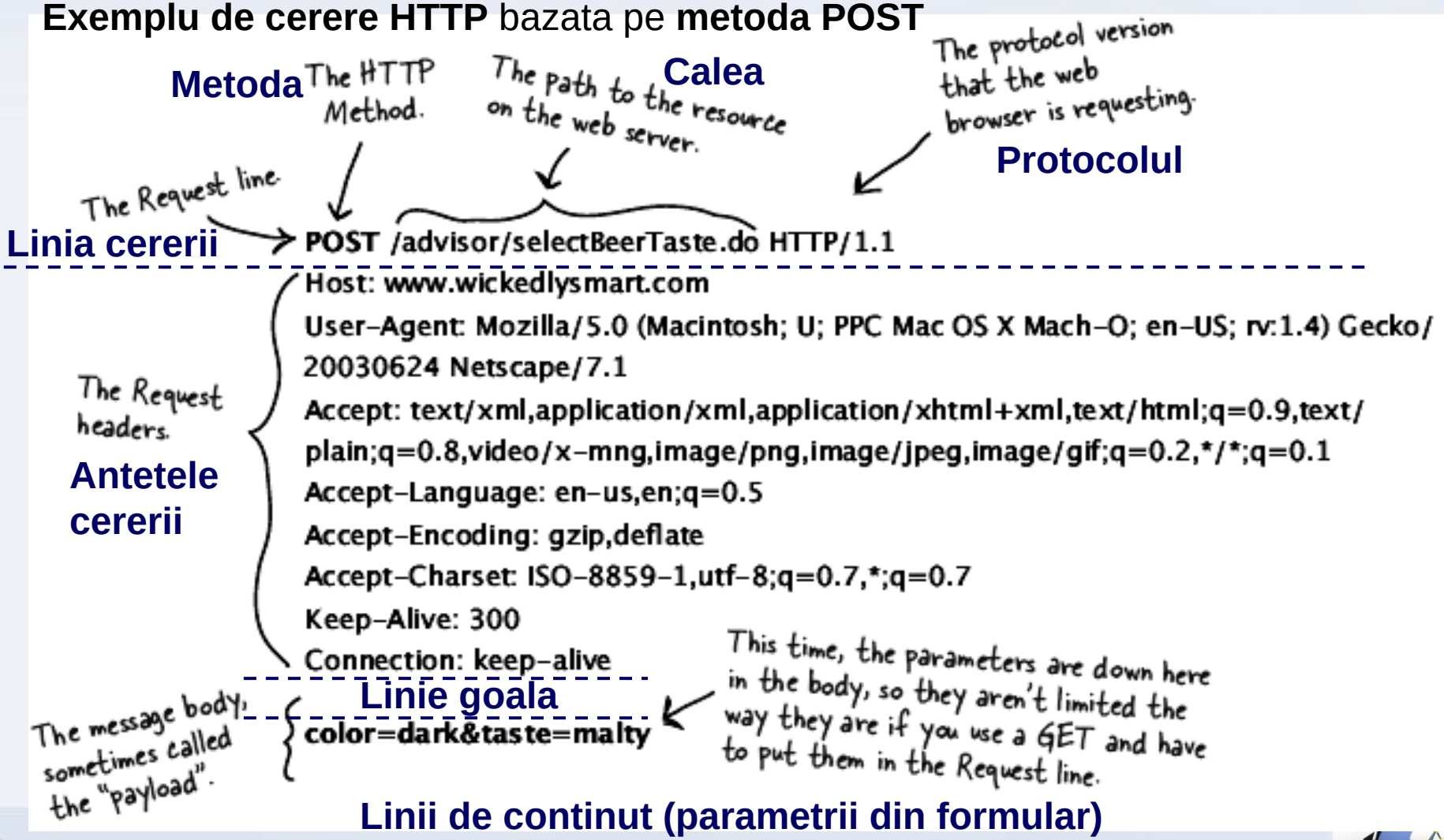
Exemplu de cerere HTTP bazata pe **metoda GET**



4.4. Tehnologii server. Java Servlet

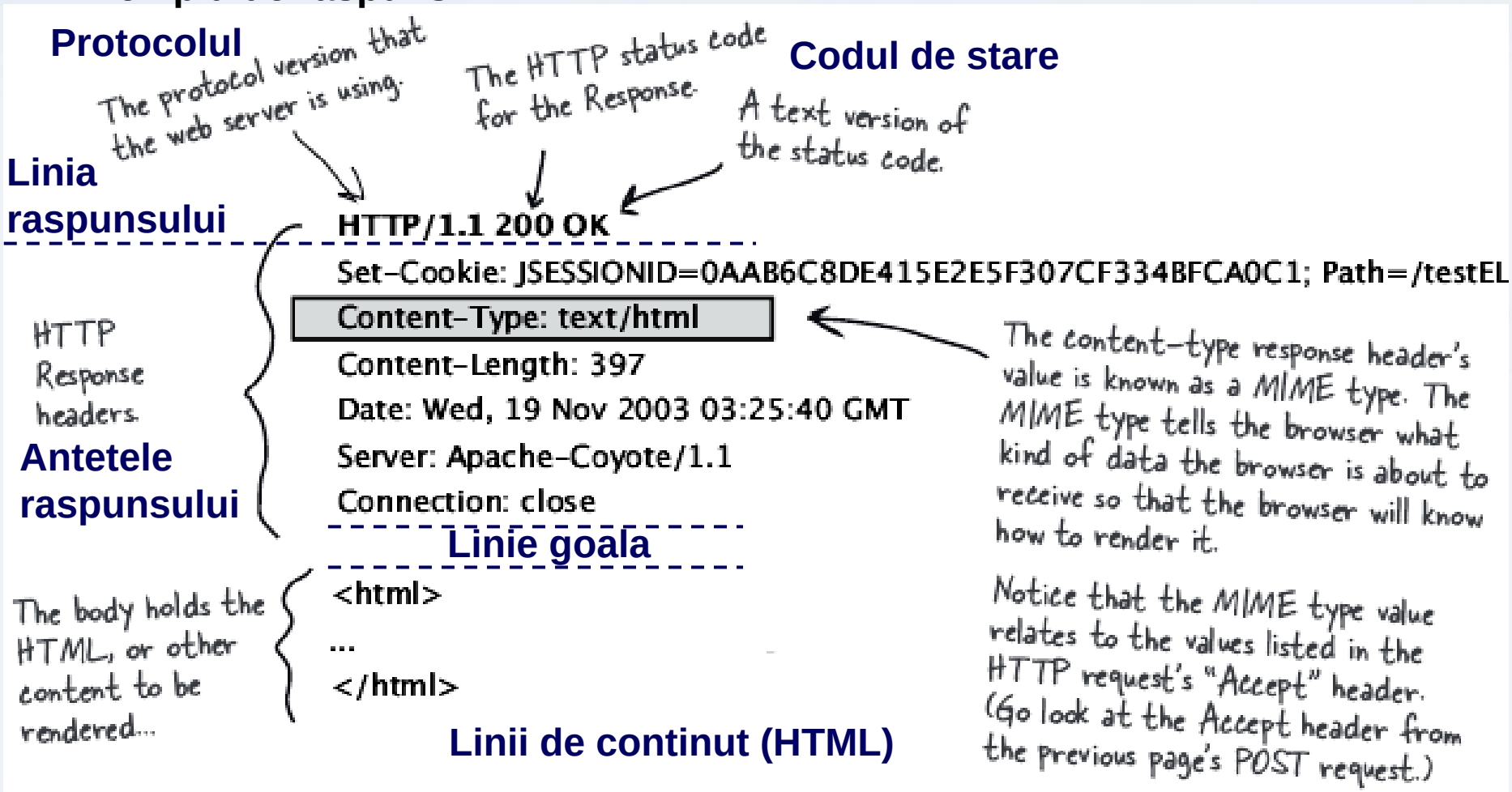
Modelul de comunicatie client-server (tranzactional)

Exemplu de cerere HTTP bazata pe **metoda POST**



Modelul de comunicatie client-server (tranzactional)

Exemplu de raspuns HTTP



Modelul de comunicatie client-server (tranzactional)

Continut static *versus* continut dinamic

Pagina HTML

When instead of this:

```
<html>
<body>
The current time is
always 4:20 PM
on the server
</body>
</html>
```

You want this:

```
<html>
<body>
The current time is
[insertTimeOnServer]
on the server
</body>
</html>
```

Continut generat dinamic (pagina **JSP**, executie **servlet**, etc.)

Continutul static este **limitat in indeplinirea cererilor**

Continutul dinamic poate **indeplini cererile** care **implica schimbare**

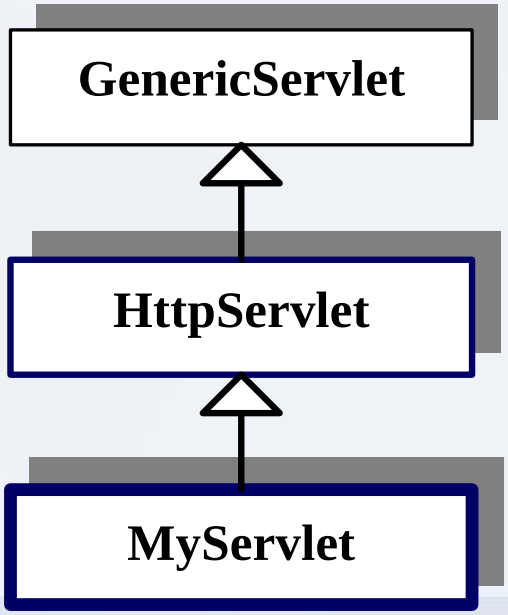
Tehnologia Java Servlet

Un *servlet* este

- un **obiect al unei clase Java** ce **extinde**, prin **crearea** unui **continut dinamic**, **functionalitatea unui server** care lucreaza dupa modelul cerere-raspuns

Un *servlet* Web

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



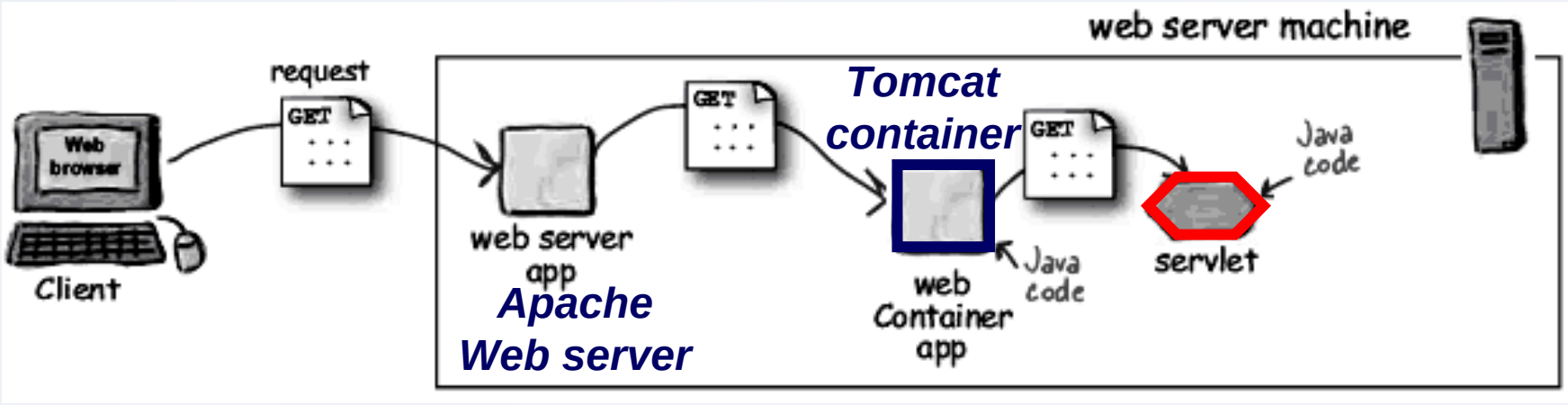
Functionalitate generica
adaugata server-ului

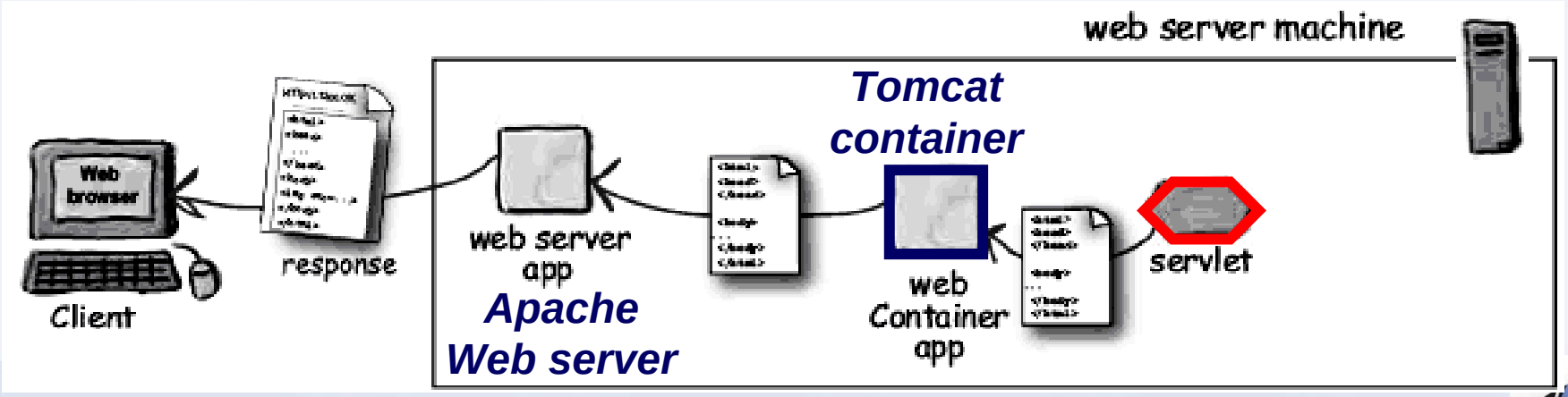
Functionalitate asociata
protocolului **HTTP**

Functionalitate specifica
aplicatiei = ***servletul***

Tehnologia Java Servlet

Servletul* Web** este o **componenta** care **se executa intr-un container Web** (numit si ***Web engine), tot asa cum *appleturile* sunt executate intr-un *browser* Web





Tehnologia Java Servlet

Containerul Web

- **identifica servletul** pe baza URL-ului

- executa **operatiile** care tin de **etapele de viata ale *servletului*** in momentele in care acestea **sunt necesare** (initializarea, incarcarea, etc.)

  - apelurile metodelor **init()**, **destroy()**, **service()**

- creeaza **obiecte** care **incapsuleaza cererea si raspunsul HTTP**

  - **HttpServletRequest** si **HttpServletResponse**

- le paseaza catre metoda **service()**

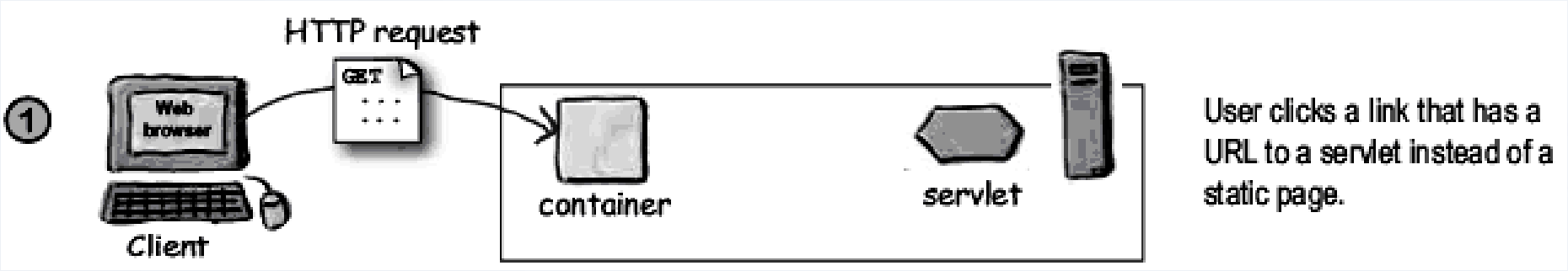
- gestioneaza variabilele CGI

- realizeaza multe alte servicii la momentul potrivit

Tehnologia Java Servlet

Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**

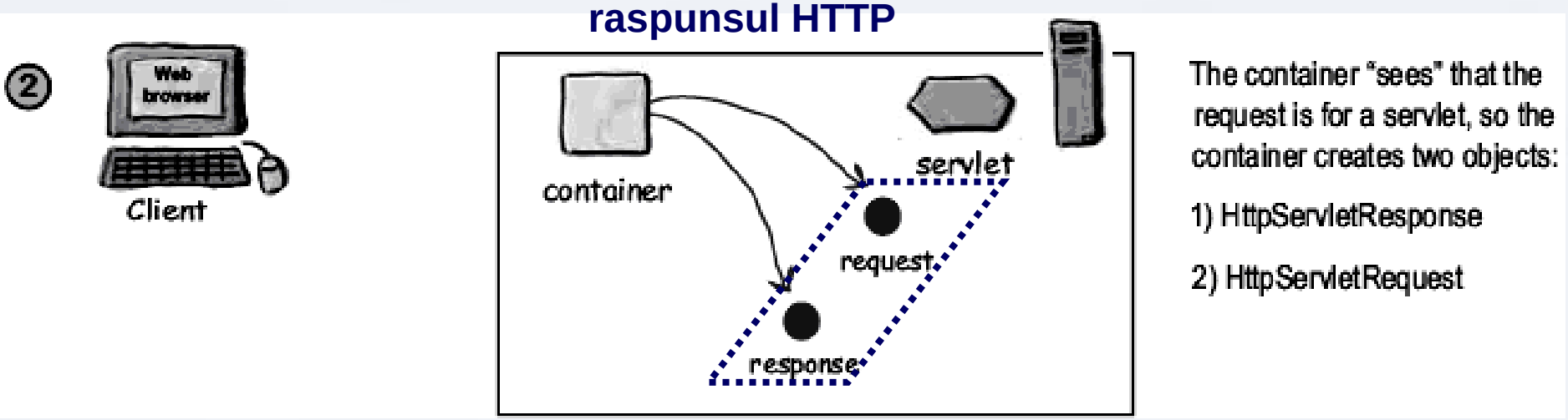
Primirea cererii de tip GET



Tehnologia Java Servlet

Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**

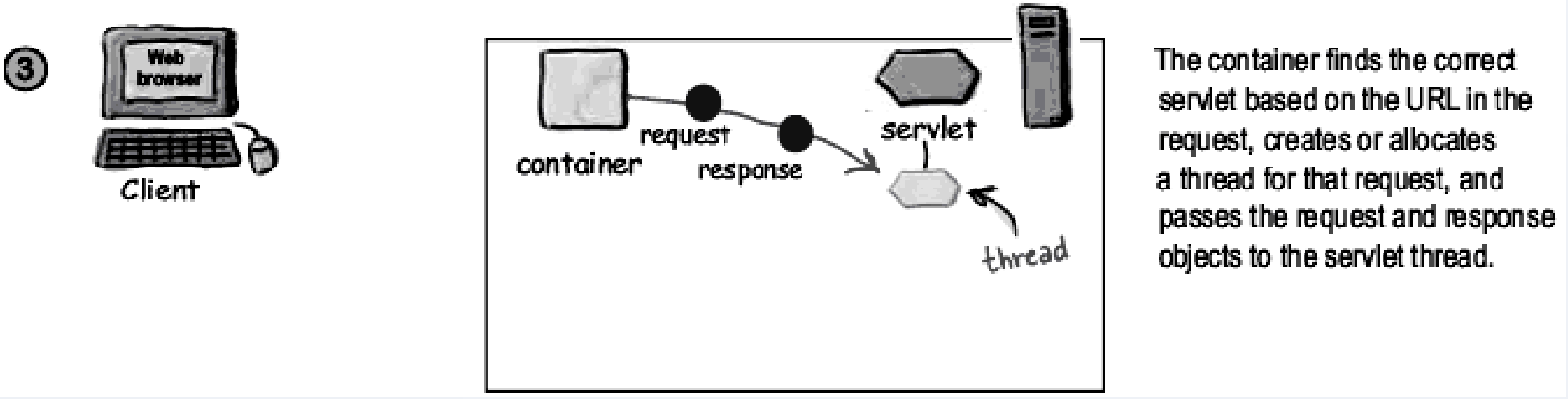
Crearea obiectelor care **incapsuleaza cererea si raspunsul HTTP**



Tehnologia Java Servlet

Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**

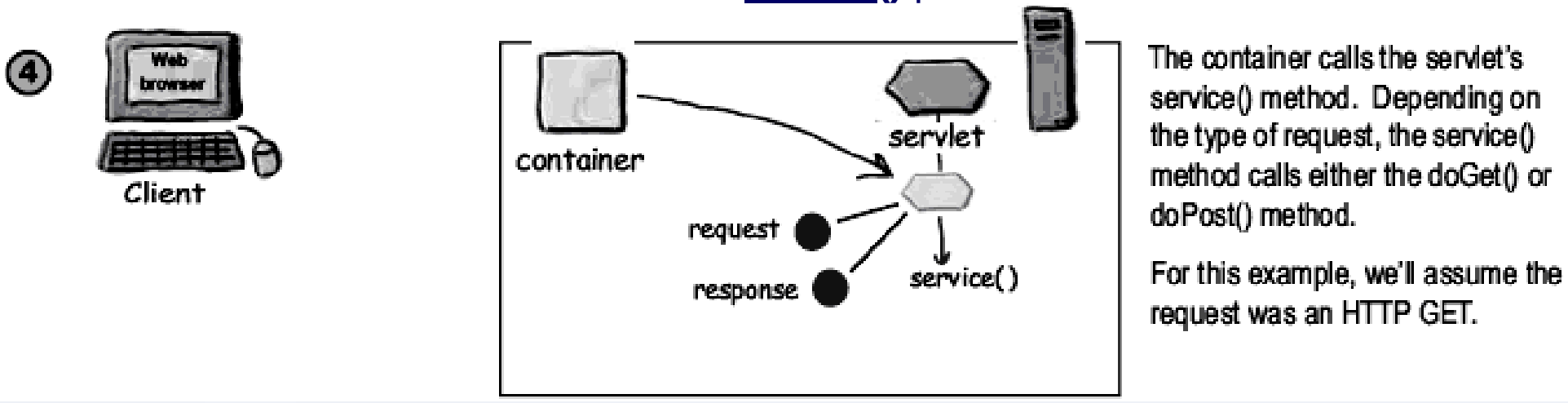
Identificarea servletului pe baza URL-ului,
crearea / alocarea unui **fir servlet**,
pasarea obiectelor cerere si **raspuns**



Tehnologia Java Servlet

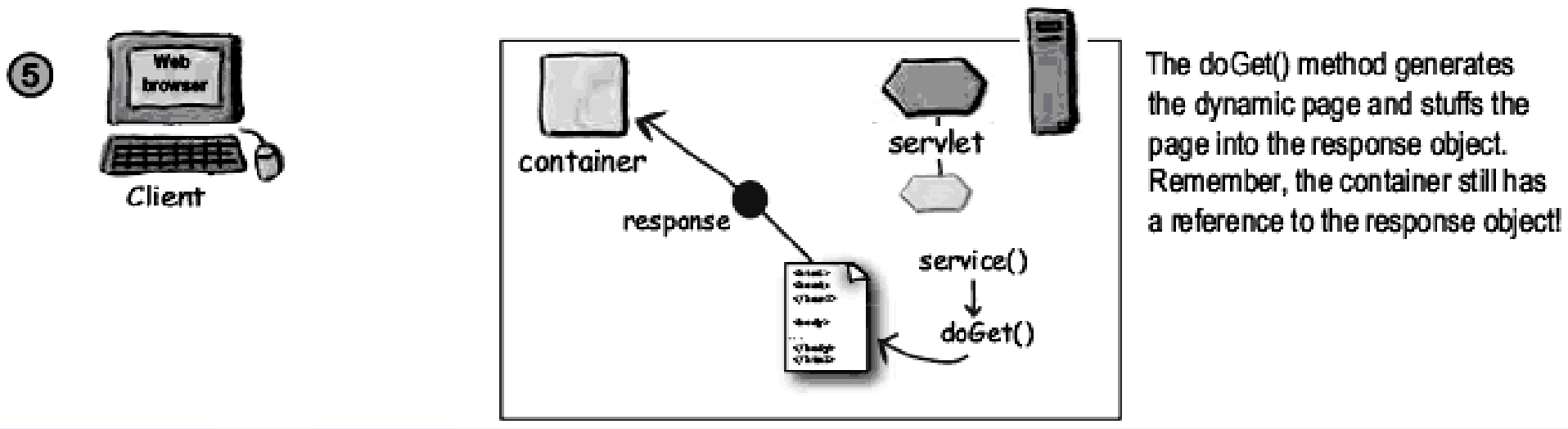
Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**

Invocarea metodei service() care **apeleaza doGet()** pentru **cereri GET** si **doPost()** pentru **cereri POST**



Tehnologia Java Servlet

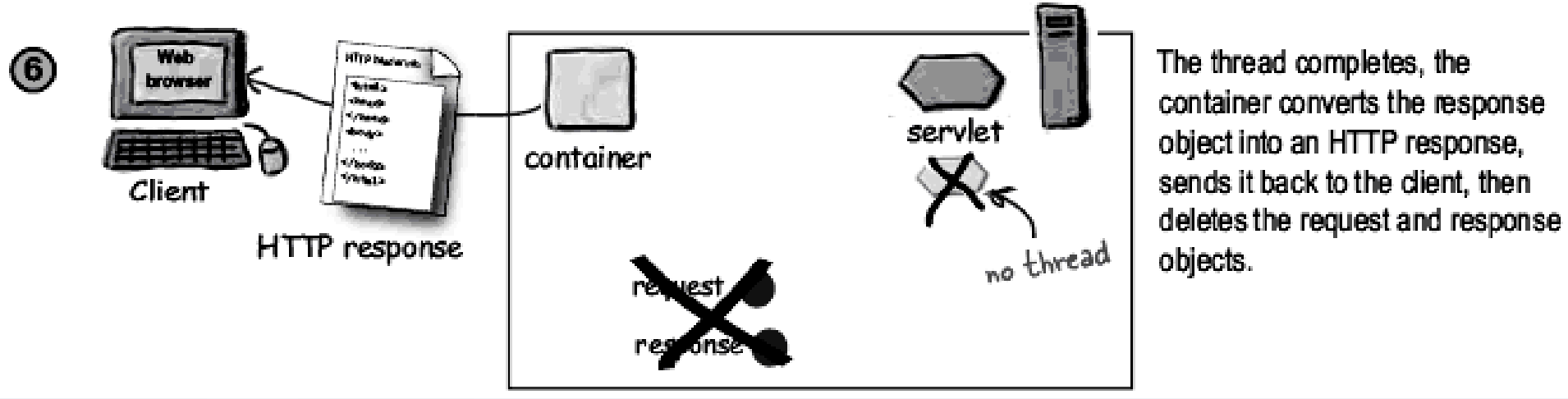
Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**



Metoda doGet() genereaza pagini dinamice in obiectul raspuns

Tehnologia Java Servlet

Pasii pe care ii face containerul Web pentru **tratarea unei cereri HTTP** adresate unui **servlet**

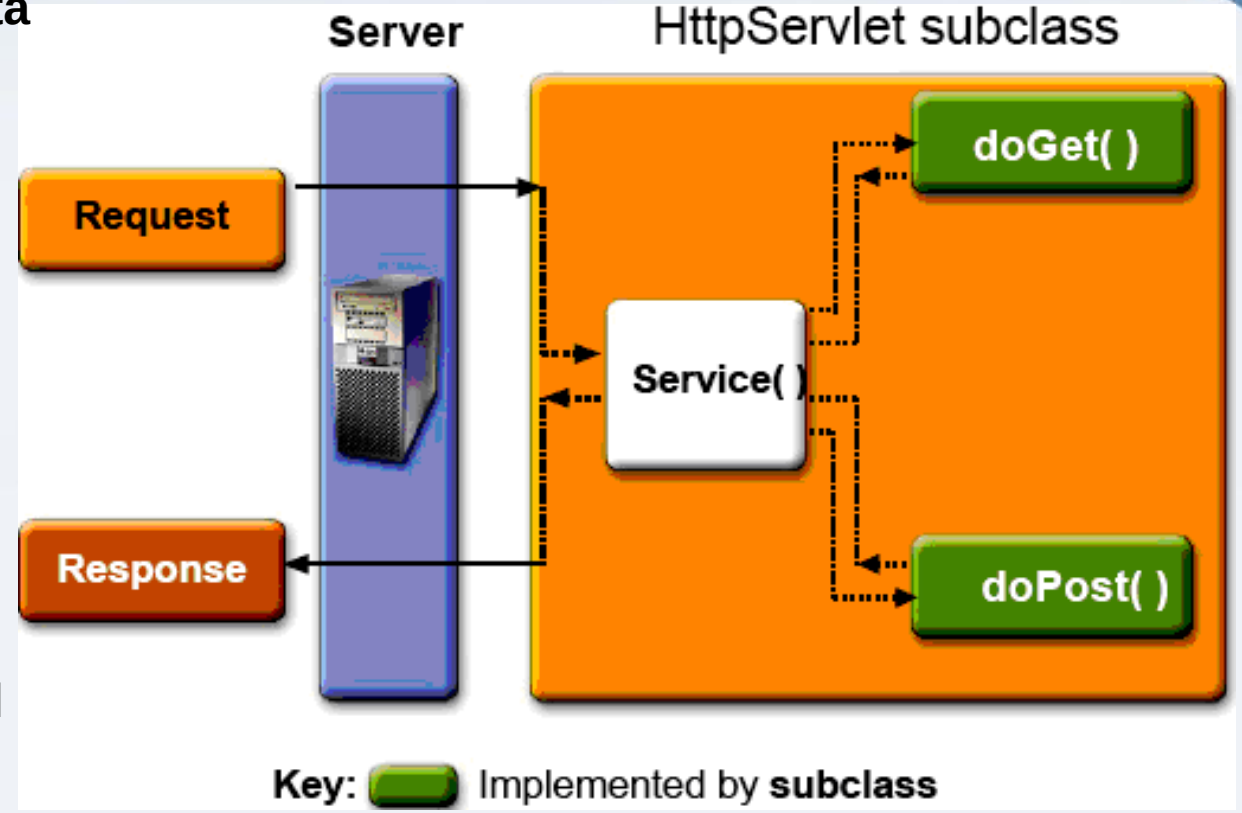


**Inchiderea firului servletului,
conversia obiectului raspuns in raspuns HTTP si
trimiterea lui catre client, distrugerea obiectelor**

Tehnologia Java Servlet

Metoda service() mostenita de la clasa **HttpServlet**

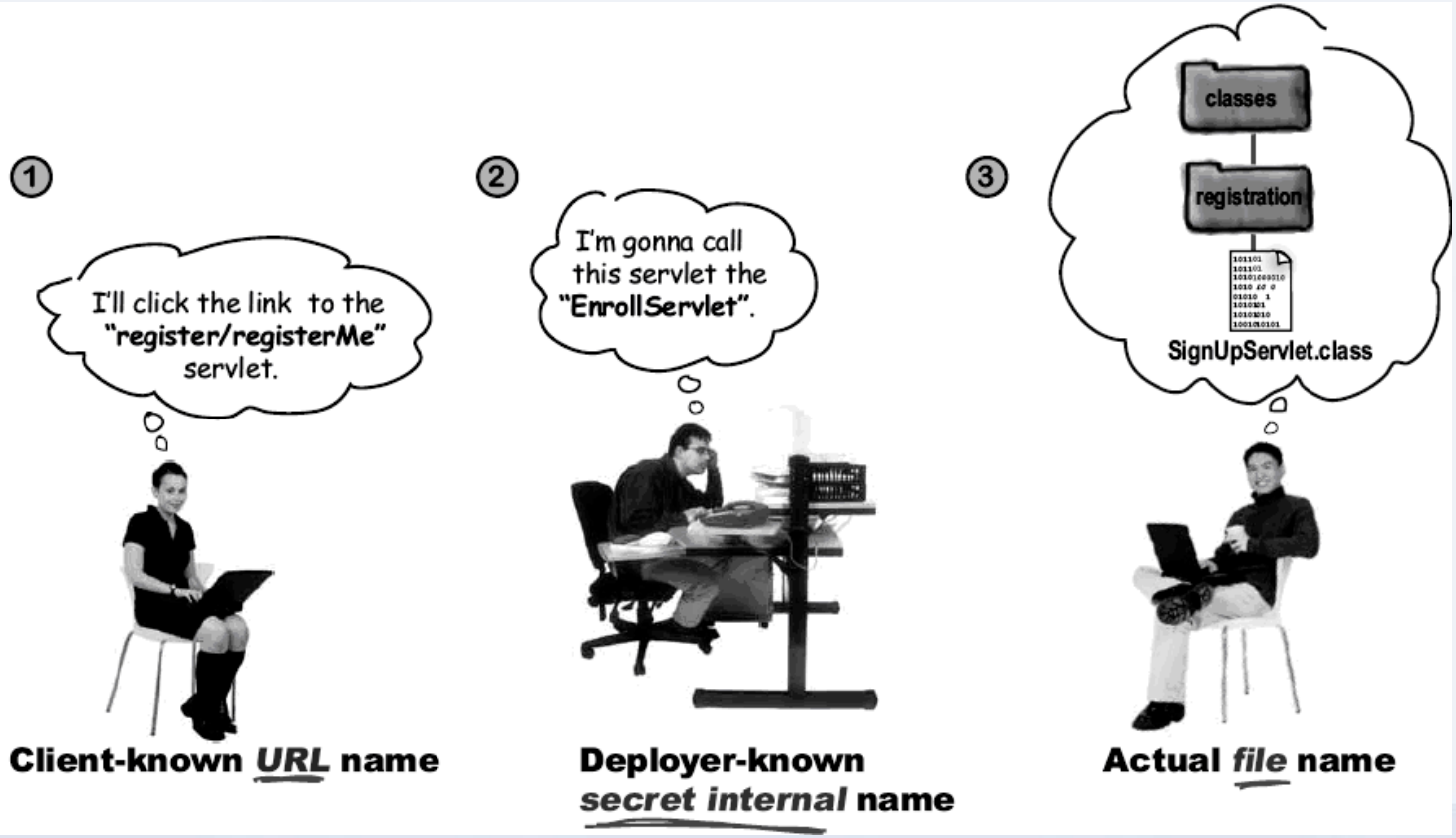
- are o **implementare generica** care
- se **recomanda sa fie pastrata** deoarece
- ea **identifica** tipul de **metoda** a cererii HTTP si **apeleaza metoda potrivita**
  - **doPost()** in cazul metodei **POST**
  - **doGet()** in cazul metodei **GET**, etc.



Tehnologia Java Servlet

Pentru a putea accesa servletul, **clientul** trebuie sa furnizeze o **adresa URL**

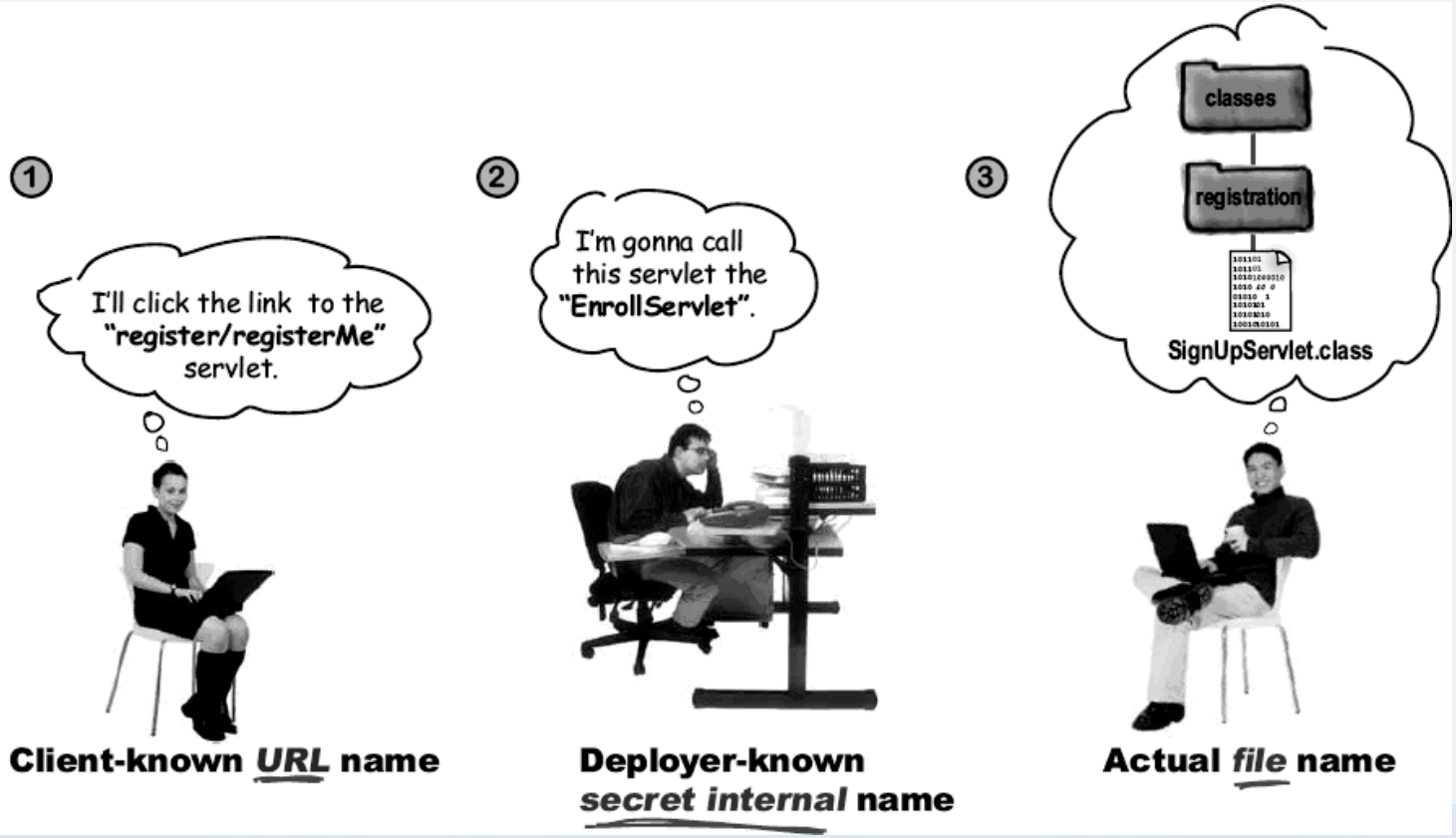
- care **difera de adresa la care se afla** cu adevarat **fisierul** cu **codul sursa al servlet-ului**



Tehnologia Java Servlet

Adresa URL (1)

- este **asociata prin** intermediul unui **nume intern (2)** dat de programator
- cu **calea completa necesara identificarii fisierului sursa (3)**



Tehnologia Java Servlet

Asocierea URL - nume intern - calea completa fisier sursa se realizeaza prin

- **codul XML** scris intr-un fisier **(web.xml)** denumit ***deployment descriptor*** (descriptor de amplasare/*deployment* - DD)
  - elementul **<servlet> asociind**
    - **nume intern** dat de programator
    - **calea completa necesara identificarii fisierului sursa**
  - elementul **<servlet-mapping> asociind**
    - **nume intern** dat de programator
    - **adresa URL**

1. **<servlet>**
   maps internal name to fully-qualified class name
2. **<servlet-mapping>**
   maps internal name to public URL name

Tehnologia Java Servlet

Exemplu de continut al unui fisier **web.xml** care specifica

- un *servlet* cu numele **ClasaServlet** aflat in directorul **numepachet** (cu cod sursa in fisierul **ClasaServlet.java** si codul compilat in **ClasaServlet.class**)

- asocierea *servletului* **ClasaServlet** aflat in directorul **numepachet** cu aliasul **numeintern** (prin intermediul elementului XML **<servlet-name>**)

- asocierea aliasului **numeintern** cu formatul utilizat de client pentru URL **/ServletAccesServiciu** (prin intermediul elementului XML **<servlet-mapping>**)

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

Tehnologia Java Servlet

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

- **primirea cererilor HTTP** de la client (sub forma de obiecte **HttpServletRequest**)
  - si **eventual utilizarea parametrilor** obtinuti **din formularul care a generat cererea**
- **executarea sarcinilor aplicatiei** (denumite ***business logic***) fie **direct**, fie prin **delegarea catre o alta componenta**
  - componenta **Web** – **servlet** sau **pagina JSP**
  - componenta ***business*** **locala** (**JavaBeans**) sau **distribuita** (***Enterprise JavaBeans*** **- EJB**),
- **generarea dinamica a continutului** (sub forma de obiecte **HttpServletResponse**)
  - si **trimiterea lui in raspunsul catre client** prin intermediul **raspunsurilor HTTP**

Tehnologia Java Servlet

Servlet-urile pot deschide catre raspunsul HTTP

- **fluxuri de caractere**

```
PrintWriter writer = response.getWriter();

writer.println("some text and HTML");
```

- sau **fluxuri de octeti** prin care **genereaza continutul raspunsului** catre client

```
ServletOutputStream out = response.getOutputStream();

out.write(aByteArray);
```

Tehnologia Java Servlet

Posibil sablon (*template*) al *servleturilor* Java

```
 1  import java.io.*;
 2  import javax.servlet.*;
 3  import javax.servlet.http.*;
 4  public class ClasaServlet extends HttpServlet {
 5    protected void doGet(HttpServletRequest request, HttpServletResponse response)
 6              throws ServletException, IOException {
 7      processRequest(request, response);
 8    }
 9    protected void doPost(HttpServletRequest request, HttpServletResponse response)
10              throws ServletException, IOException {
11      processRequest(request, response);
12    }
13    protected void processRequest(HttpServletRequest request,
14                HttpServletResponse response) throws ServletException, IOException {
15      // Stabilirea tipului de continut
16      response.setContentType("text/html");
17
18      // Citire din "request" antete HTTP primite (ex. cookies) si date formular HTML
19
20      // Generare in "response" linie si antete raspuns HTTP (tip continut, cookies)
21      PrintWriter out = response.getWriter();
22
23      // Utilizare "out" pentru a trimite continut HTML catre browser
24    }
25  }
```

Tehnologia Java Servlet

Exemplificare – clasa care ofera serviciul *business* utilizat de servlet

```
 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  }
```

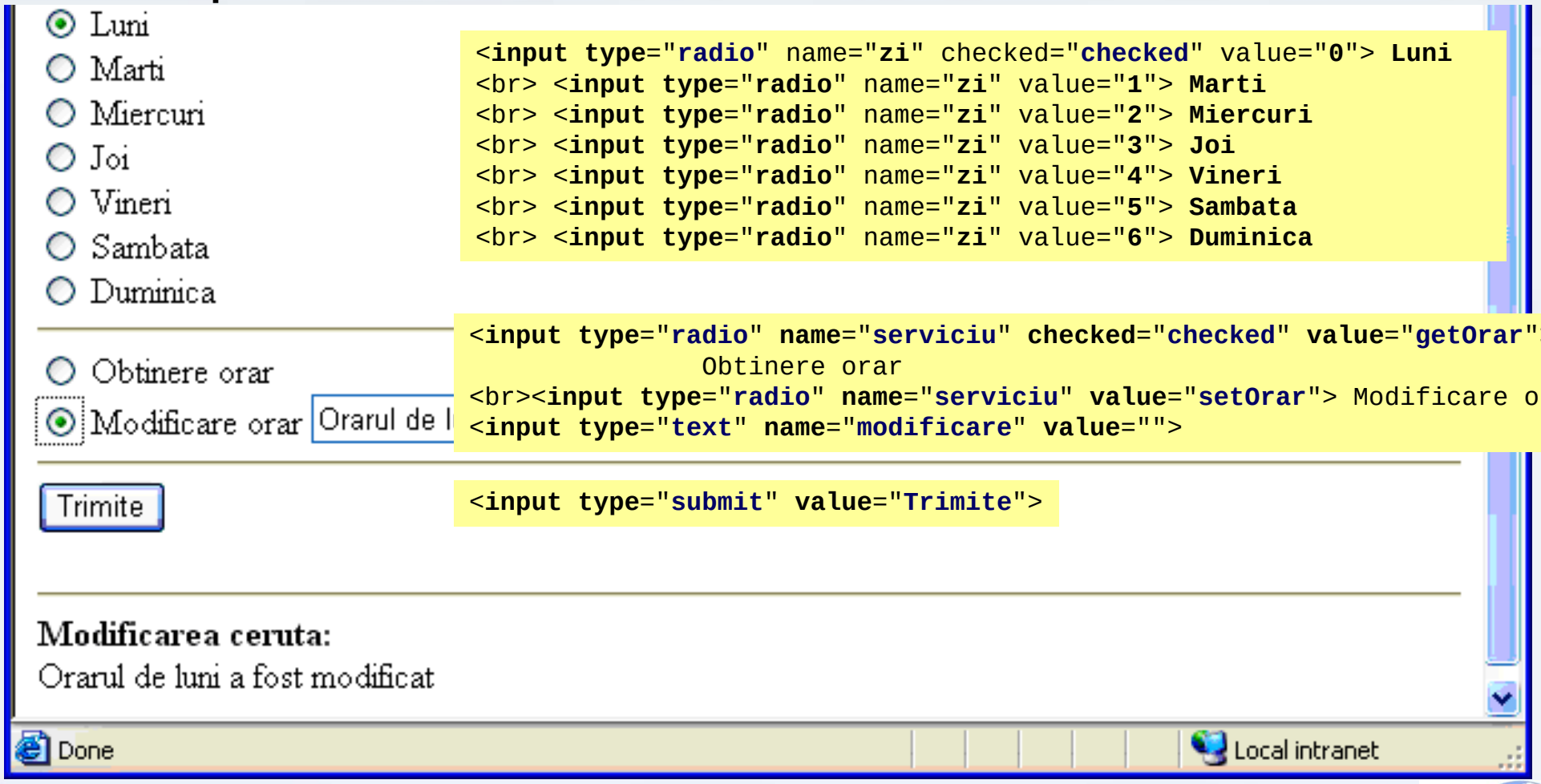
Tehnologia Java Servlet

Exemplificare – formularul initial din care este accesat *servletul*

```
 1  <html>
 2      <head>
 3          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 4          <title>Acces orar</title>
 5      </head>
 6      <body>
 7        <h1>Acces orar (forma initiala)</h1>
 8           <hr><form name="input" action="/AplicatieOrar1/AccesInitial" method="get">
 9           <input type="radio" name="zi" checked="checked" value="0"> Luni
10           <br> <input type="radio" name="zi" value="1"> Marti
11           <br> <input type="radio" name="zi" value="2"> Miercuri
12           <br> <input type="radio" name="zi" value="3"> Joi
13           <br> <input type="radio" name="zi" value="4"> Vineri
14           <br> <input type="radio" name="zi" value="5"> Sambata
15           <br> <input type="radio" name="zi" value="6"> Duminica
16           <hr>
17           <input type="radio" name="serviciu" checked="checked" value="getOrar">
18                 Obtinere orar
19           <br> <input type="radio" name="serviciu" value="setOrar"> Modificare orar
20           <input type="text" name="modificare" value="">
21           <input type="submit" value="Trimite">
22        </form>
23        <hr>
24      </body>
25  </html>
```

Tehnologia Java Servlet

Exemplificare – formularul initial din care este accesat *servletul*




```
<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
```

```
<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">
```

Tehnologia Java Servlet

Exemplificare – servlet care utilizeaza clasa anterioara (1)

```
 1 import java.io.*;
 2 import java.net.*;
 3 import javax.servlet.*;
 4 import javax.servlet.http.*;
 5 public class ServletOrarInitial extends HttpServlet {
 6
 7   // Metoda utilitara catre care doGet() si doPost() deleaga executia
 8   protected void processRequest(HttpServletRequest request,
 9           HttpServletResponse response) throws ServletException, IOException {
10
11     // Stabilirea tipului de continut
12     response.setContentType("text/html;charset=UTF-8");
13
14     PrintWriter out = response.getWriter();
15
16     // Generarea formularului pentru accesul recursiv la servicii
17     out.println("<html>");
18     out.println("<head>");
19     out.println("<title>Acces orar</title>");
20     out.println("</head>");
21     out.println("<body>");
```

Tehnologia Java Servlet

Exemplificare – servlet care utilizeaza clasa anterioara (2)

```
22      out.println("<h1>Acces orar (forma initiala) - generat de servlet</h1>");
23
24      out.println("<hr><form name=\"input\" action=\"AccesInitial\" method=\"get\">");
25
26      out.println("<input type=\"radio\" name=\"zi\" checked=\"checked\""
27                  + " value=\"0\"> Luni");
28      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"1\"> Marti");
29      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"2\"> Miercuri");
30      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"3\"> Joi");
31      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"4\"> Vineri");
32      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"5\"> Sambata");
33      out.println("<br> <input type=\"radio\" name=\"zi\" value=\"6\"> Duminica");
34      out.println("<hr>");
35
36      out.println("<input type=\"radio\" name=\"serviciu\" checked=\"checked\""
37                  + " value=\"getOrar\"> Obtinere orar");
38      out.println("<br><input type=\"radio\" name=\"serviciu\" value=\"setOrar\">"
39                  + " Modificare orar");
40
41      out.println("<input type=\"text\" name=\"modificare\" value=\"\">");
42
43      out.println("<hr><input type=\"submit\" value=\"Trimite\">");
44      out.println("</form><hr>");
```

Tehnologia Java Servlet

Exemplificare – servlet care utilizeaza clasa anterioara (3)

```
45      Orar orar = new Orar();
46
47      // Obtinerea parametrilor introdusi de utilizator in formular
48      int zi = Integer.parseInt(request.getParameter("zi"));
49
50      // Daca serviciul cerut e obtinere orar
51      if (request.getParameter("serviciu").equals("getOrar")) {
52         out.println("<b>Orarul cerut:</b> <br>" + orar.getOrar(zi));
53      }
54
55      // Daca serviciul 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
62      out.println("</body>");
63      out.println("</html>");
64      out.close();
65   }
```

Tehnologia Java Servlet

Exemplificare – servlet care utilizeaza clasa anterioara (4)

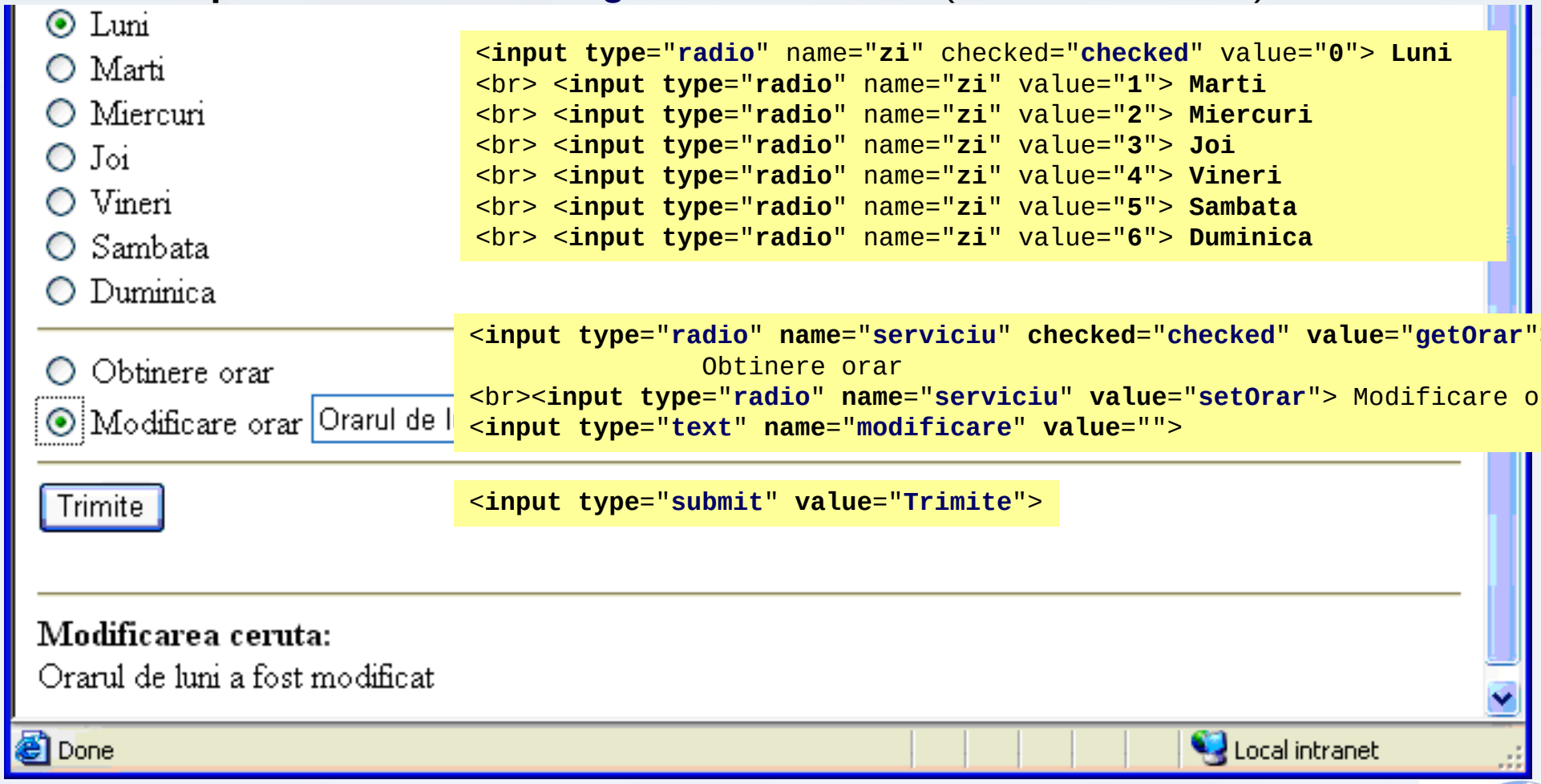
```
66    // Metoda corespunzatoare cererilor HTTP de tip GET
67    protected void doGet(HttpServletRequest request,
68            HttpServletResponse response) throws ServletException, IOException {
69      processRequest(request, response);
70    }
71
72    // Metoda corespunzatoare cererilor HTTP de tip POST
73    protected void doPost(HttpServletRequest request,
74            HttpServletResponse response) throws ServletException, IOException {
75      processRequest(request, response);
76    }
77  }
```

Formularul generat de servlet contine urmatoarele **sectiuni**

- **7 butoane radio** denumite "**zi**" (dintre care **primul selectat** - ***checked***)
- **2 butoane radio** denumite "**serviciu**" (dintre care **primul selectat**)
- o **intrare text**
- un **buton** de tip "**submit**" cu eticheta "**Trimite**"
- un **text generat dinamic** in **functie de serviciul cerut**

Tehnologia Java Servlet

Exemplificare – formularul generat de *servlet* (similar celui initial)




```
<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
```

```
<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">
```

Tehnologia Java Servlet

Exemplificare – continutul fisierului web.xml in acest caz

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
 3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4           xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
 5           http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 6
 7      <servlet>
 8          <servlet-name>servletinitial</servlet-name>
 9          <servlet-class>ServletOrarInitial</servlet-class>
10      </servlet>
11
12      <servlet-mapping>
13          <servlet-name>servletinitial</servlet-name>
14          <url-pattern>/AccesInitial</url-pattern>
15      </servlet-mapping>
16
17      <welcome-file-list>
18        <welcome-file>
19              index.jsp
20          </welcome-file>
21      </welcome-file-list>
22  </web-app>
```

Tehnologia Java Servlet

Protocolul HTTP nu are stari (este ***stateless***) asa incat

- **serverul HTTP nu retine informatii privind cererile anterioare**
- asa incat se poate spune ca **nu are "memorie"**

In plus, pentru ca **servlet-urile** sa fie **accesate eficient** de catre **mai multi clienti in acelasi timp**

- **containerul de *servleturi*** formeaza un asa-numit ***thread pool*** **(bazin) cu fire de executie ale servletului** din care **alege unul oarecare** pentru **fiecare client**

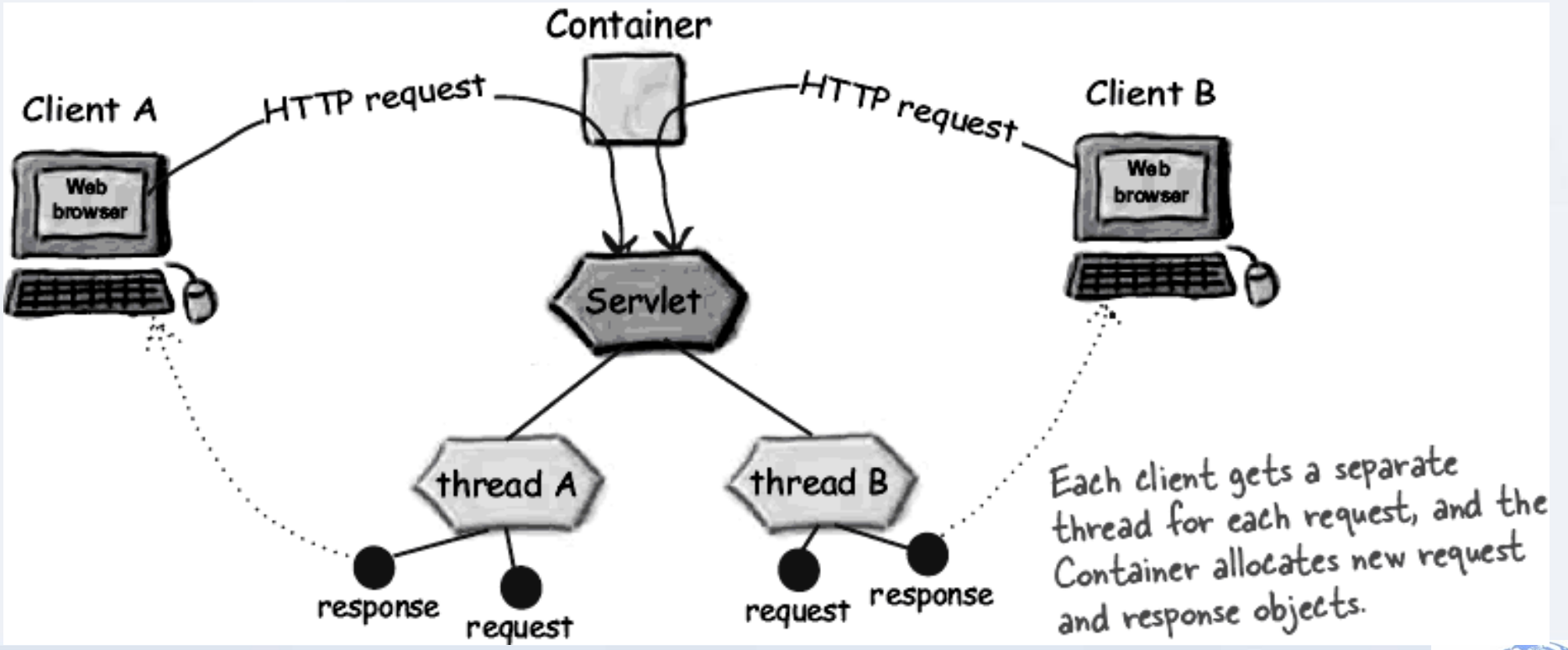
De aceea **declararea obiectului** de tip **Orar** ca **variabila instanta**

- **nu este** o solutie ***thread safe***

Tehnologia Java Servlet

Pentru ca **servlet-urile** sa fie **accesate eficient** de catre **mai multi clienti in acelasi timp**

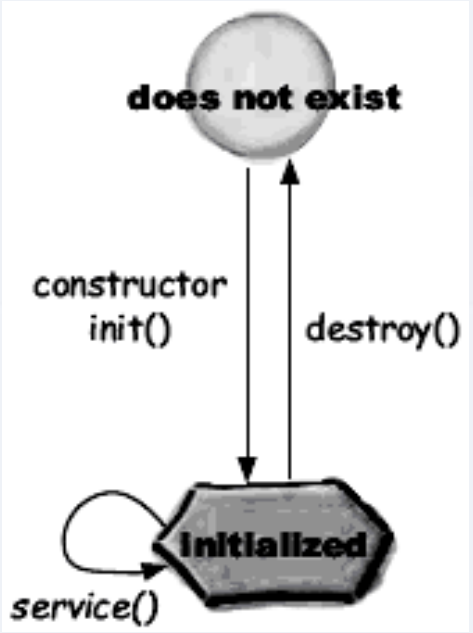
- **containerul de *servleturi*** formeaza un ***thread pool*** **(bazin) cu fire de executie ale servletului** din care **alege unul oarecare** pentru **fiecare client**



Tehnologia Java Servlet

Etapele de viata ale *servleturilor* sunt **gestionate de container**

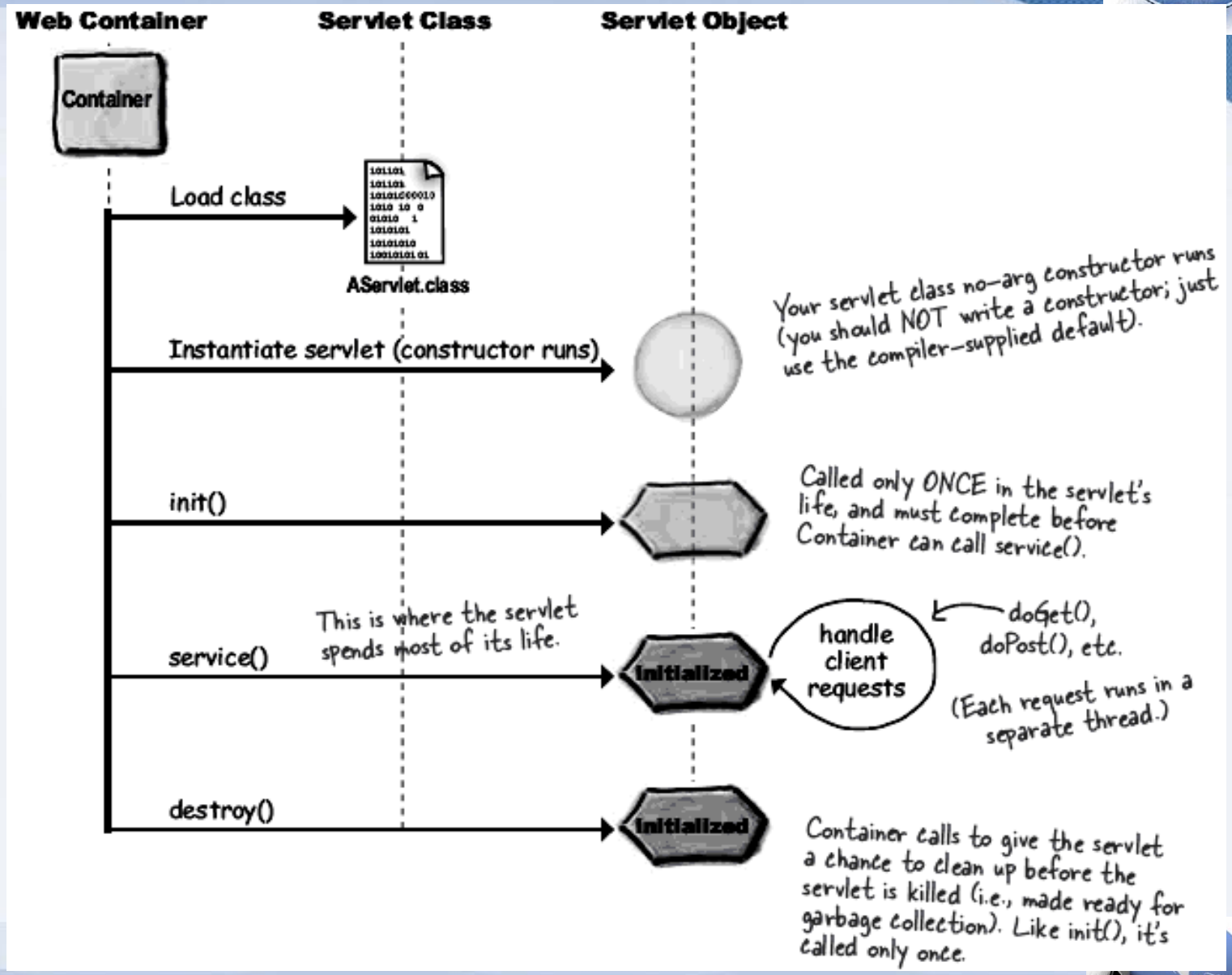
- incep cu **crearea *servletului***
  - prin **apelul constructorului implicit** (fara parametri si fara cod) **urmat de** apelul metodei **init()**
- ceea ce **conduce servletul** in starea **initializat**
  - in care **accepta si trateaza** apelurile **service()** (ca **fire de executie separate** pentru **fiecare client**)
- stare din care **iese prin apelul** metodei **destroy()** **de catre container**



4.4. Tehnologii server. Java Servlet

Tehnologia Java Servlet

Etapele de viata ale *servleturilor* sunt **gestionate de container**



Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

- solutia Java EE la lipsa "memoriei" din serverele HTTP (Web)

Un obiect "sesiune" din clasa HttpSession

- **gestionat de containerul** de servleturi

- permite **pastrarea referintelor catre obiecte** ale **aplicatiei** (sub forma de "**atribute**" ale **obiectului** din clasa **HttpSession**)

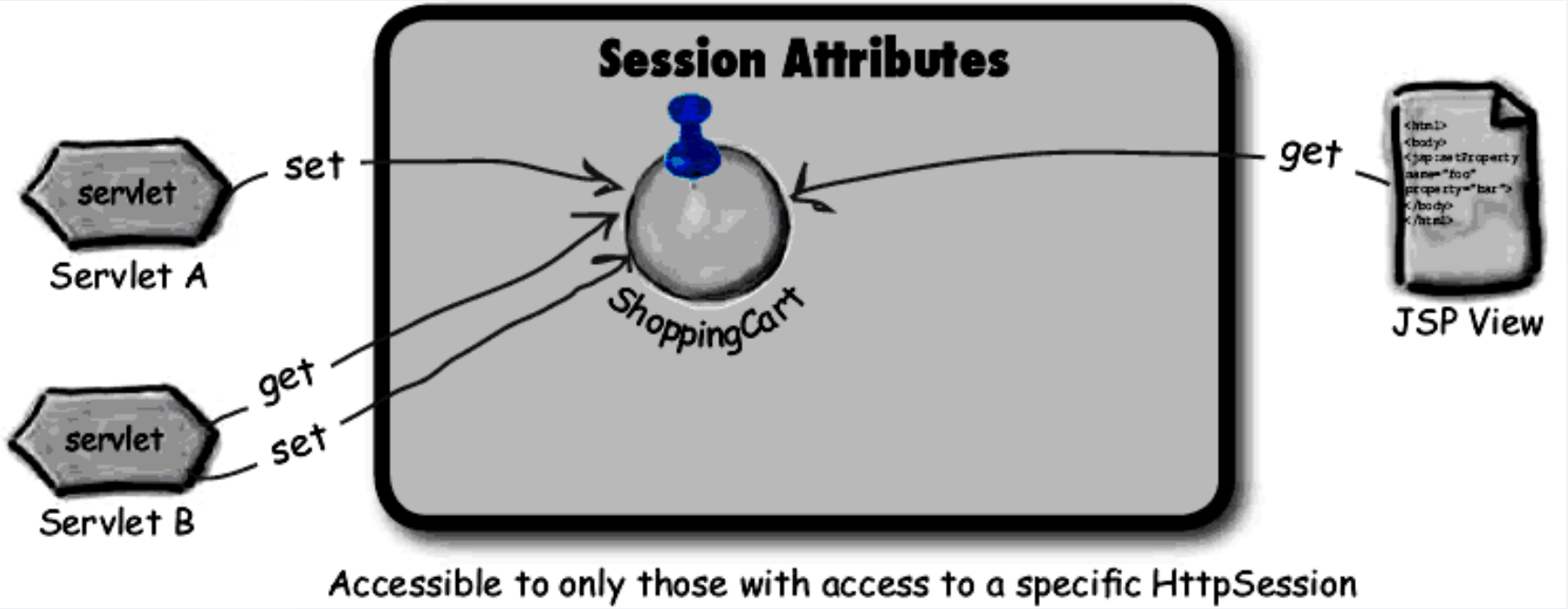
  - **prin intermediul** metodei **setAttribute()**

- si **regasirea acestora**

  - **prin intermediul** metodei **getAttribute()**

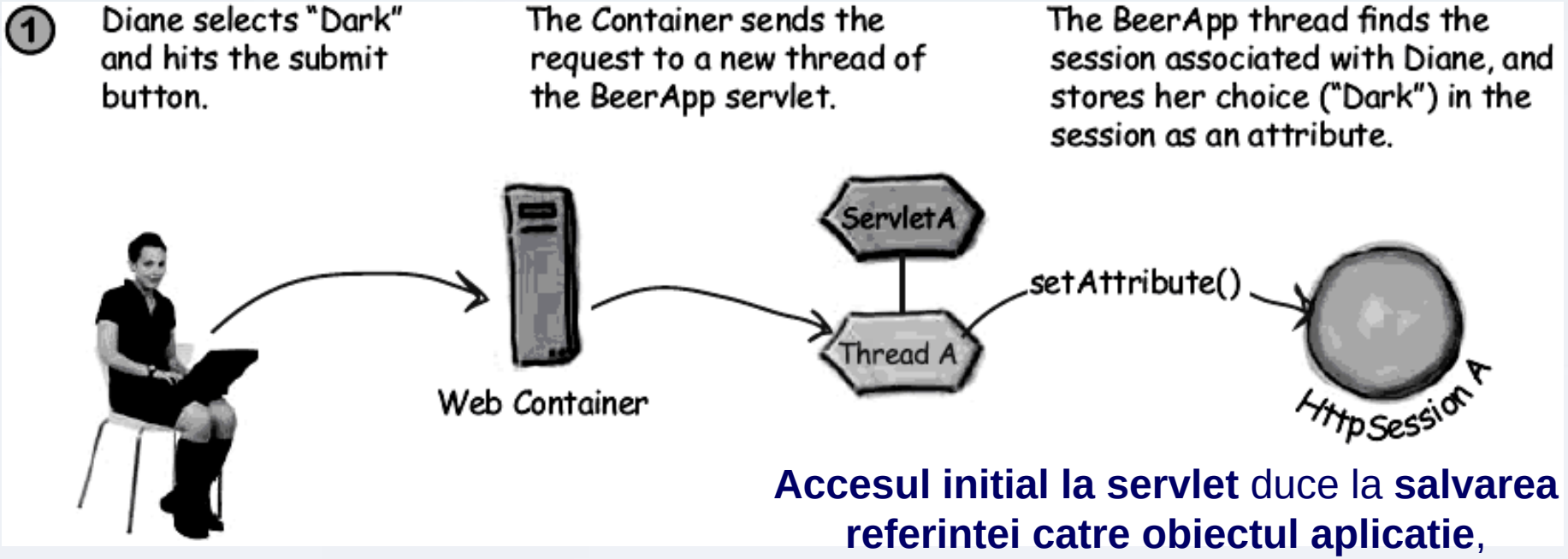
Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi



Tehnologia Java Servlet

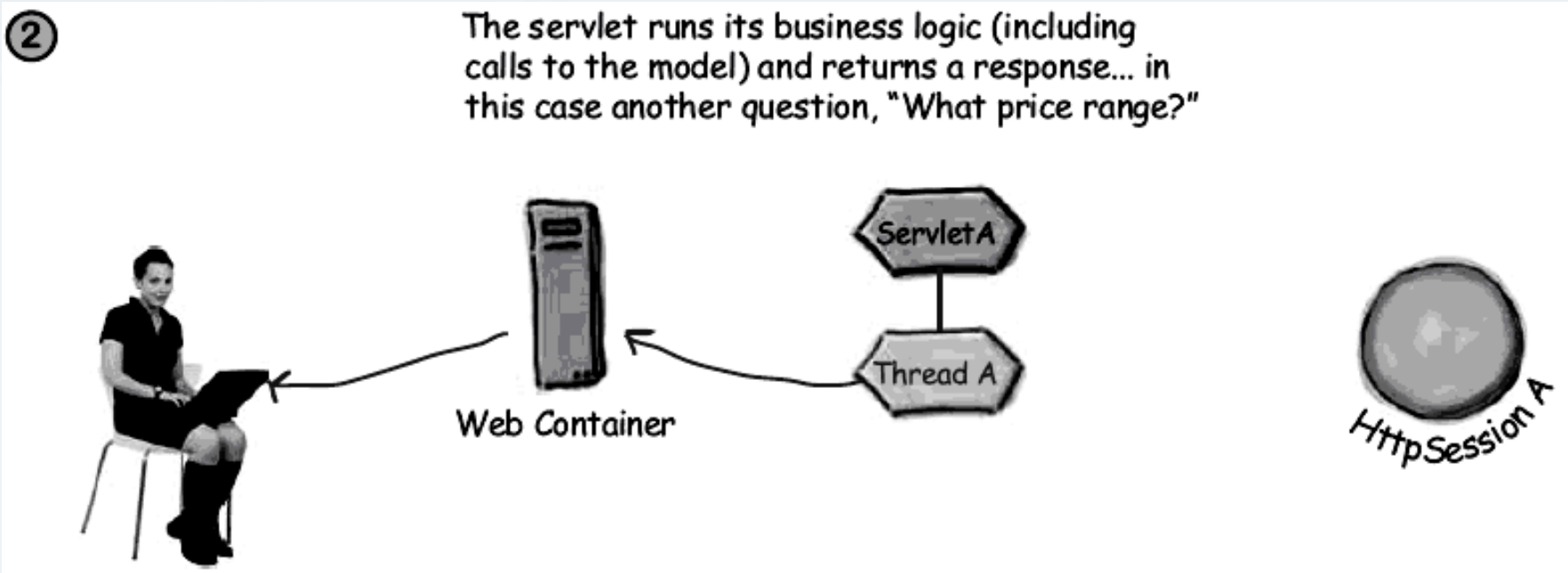
Session tracking* – adaugarea "memoriei" in *servleturi



Accesul initial la servlet duce la **salvarea referintei catre obiectul aplicatie**, ca **"atribut"** al obiectului **"sesiune"**, **prin** apelul **setAttribute()**

Tehnologia Java Servlet

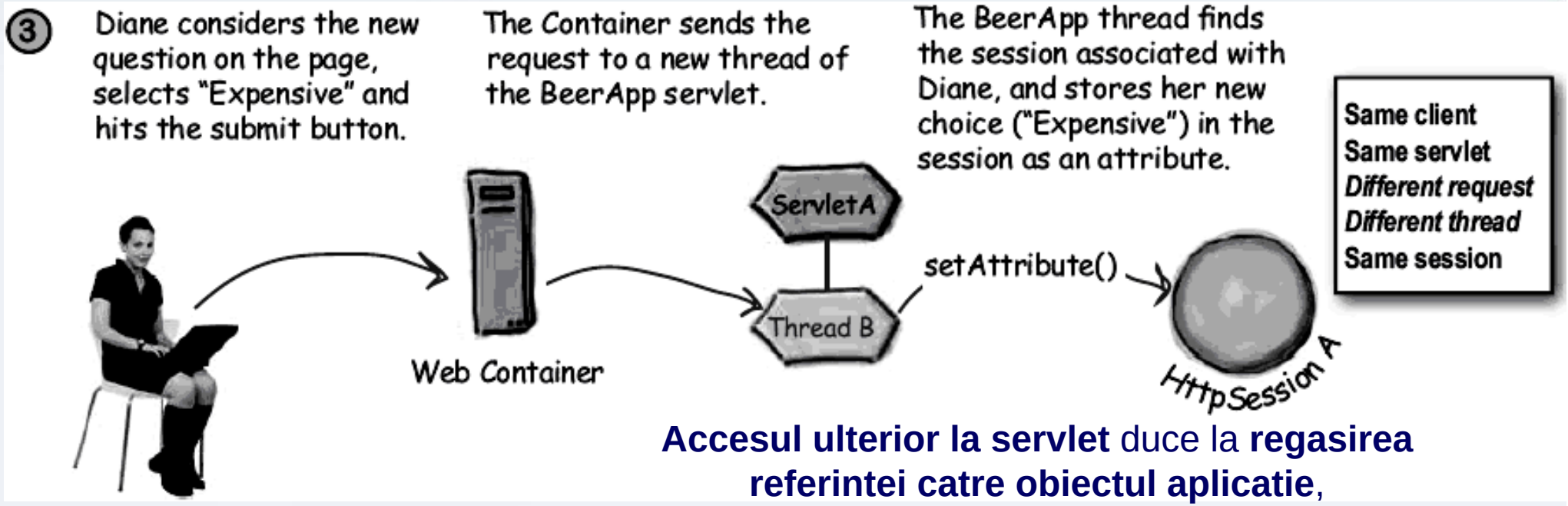
Session tracking* – adaugarea "memoriei" in *servleturi



Odata cu raspunsul, containerul trebuie sa adauge informatie de identificare a sesiunii (**cookie**, etc.) pe care **clientul o va retrimite**

Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

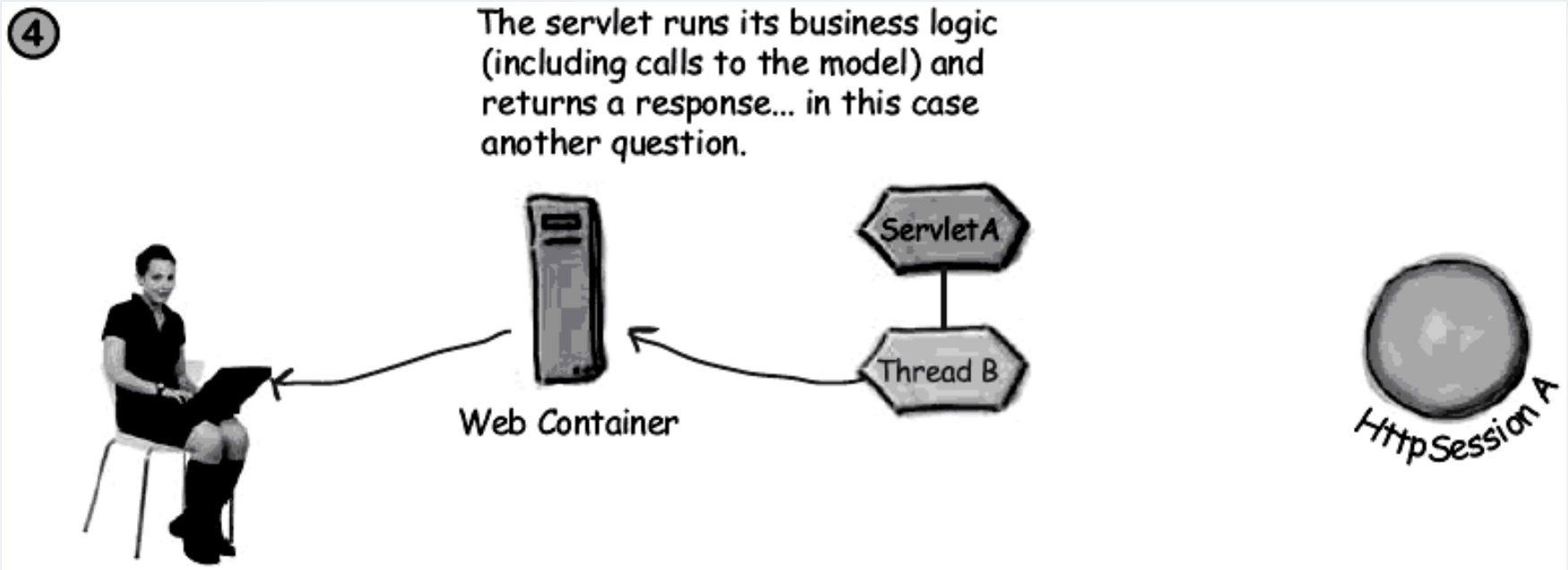


Accesul ulterior la servlet duce la **regasirea referintei catre obiectul aplicatie**, ca **"atribut"** al obiectului **"sesiune"**, **prin** apelul **getAttribute()**

Odata cu cererea, containerul primeste informatia de identificare a sesiunii (**cookie**, etc.) **retrimisa de client**

Tehnologia Java Servlet

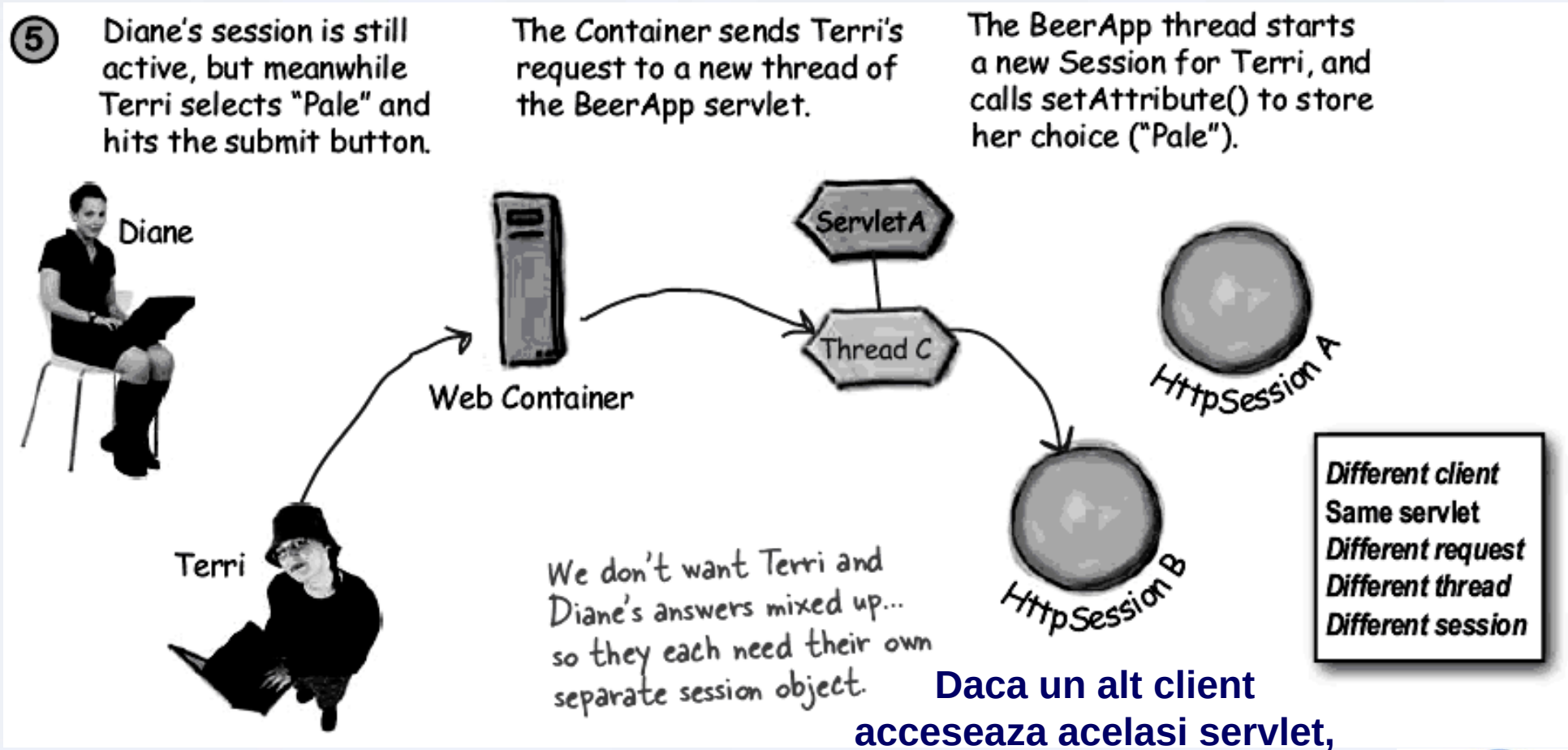
Session tracking* – adaugarea "memoriei" in *servleturi



Odata cu raspunsul, containerul adauga din nou informatia de identificare a sesiunii pe care **clientul o va retrimite**

Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

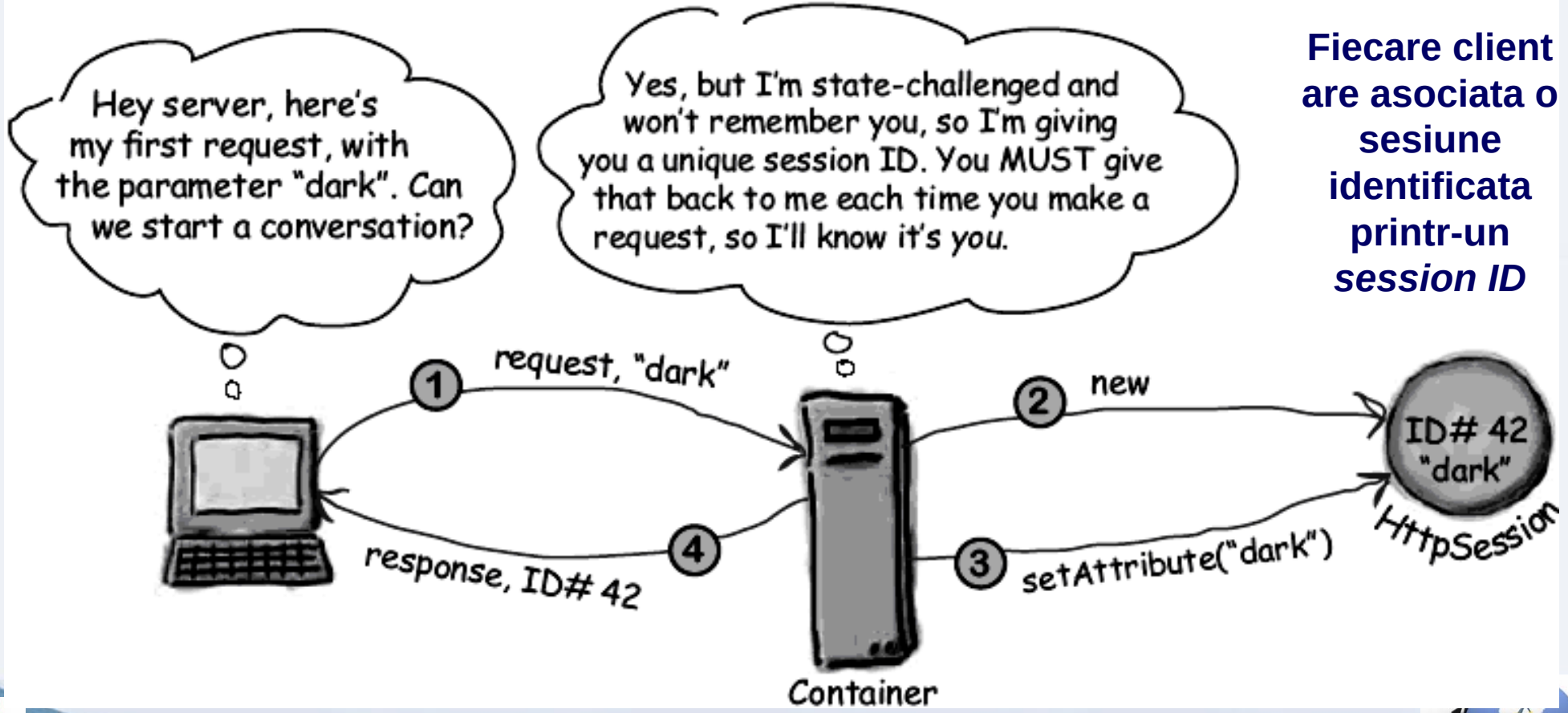


Daca un alt client acceseaza acelasi servlet, el primeste o alta sesiune

Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

Accesul initial la servlet duce la
atribuirea unui identificator al **"sesiunii"**

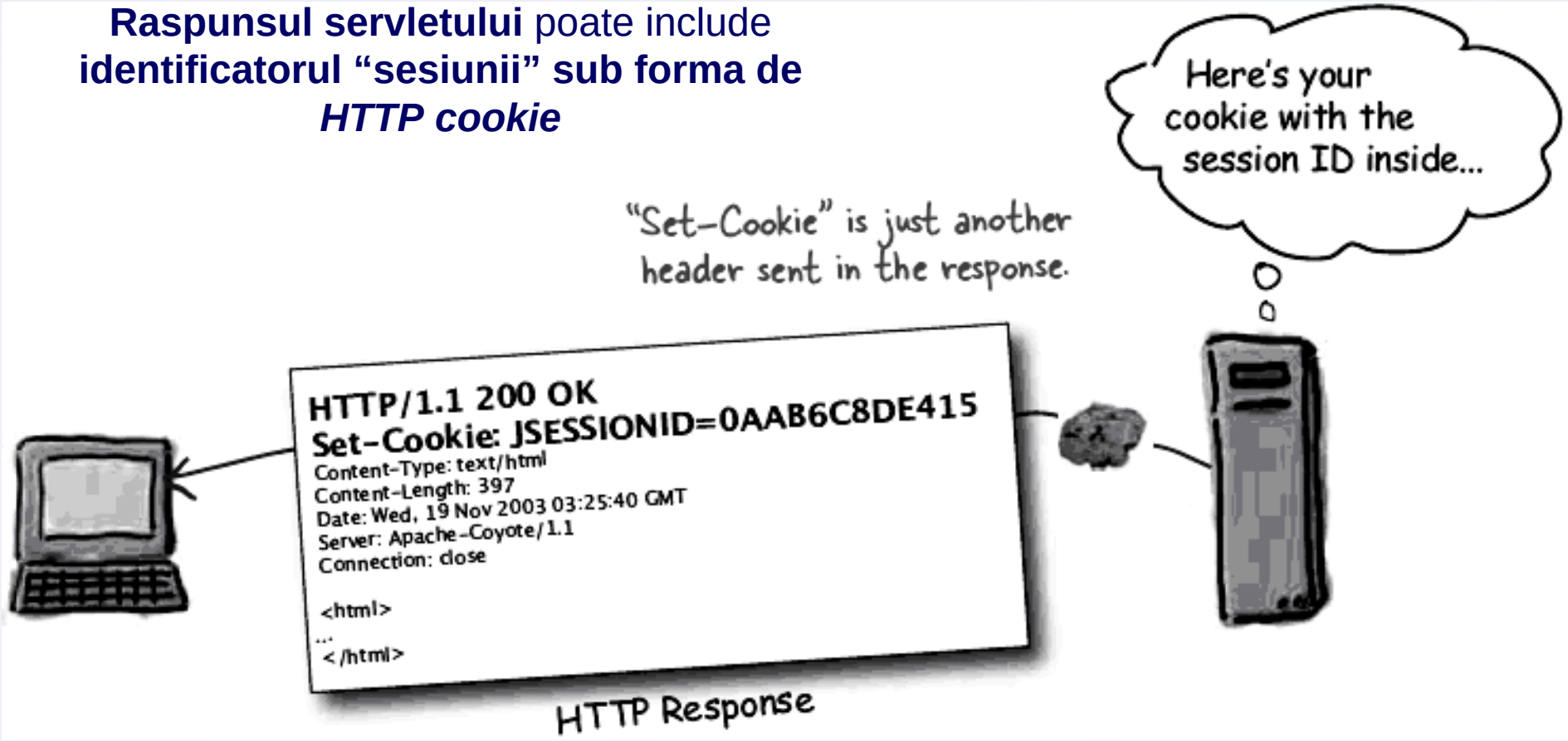


Fiecare client are asociata o sesiune identificata printr-un *session ID*

Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

Raspunsul servletului poate include **identificatorul "sesiunii" sub forma de *HTTP cookie***

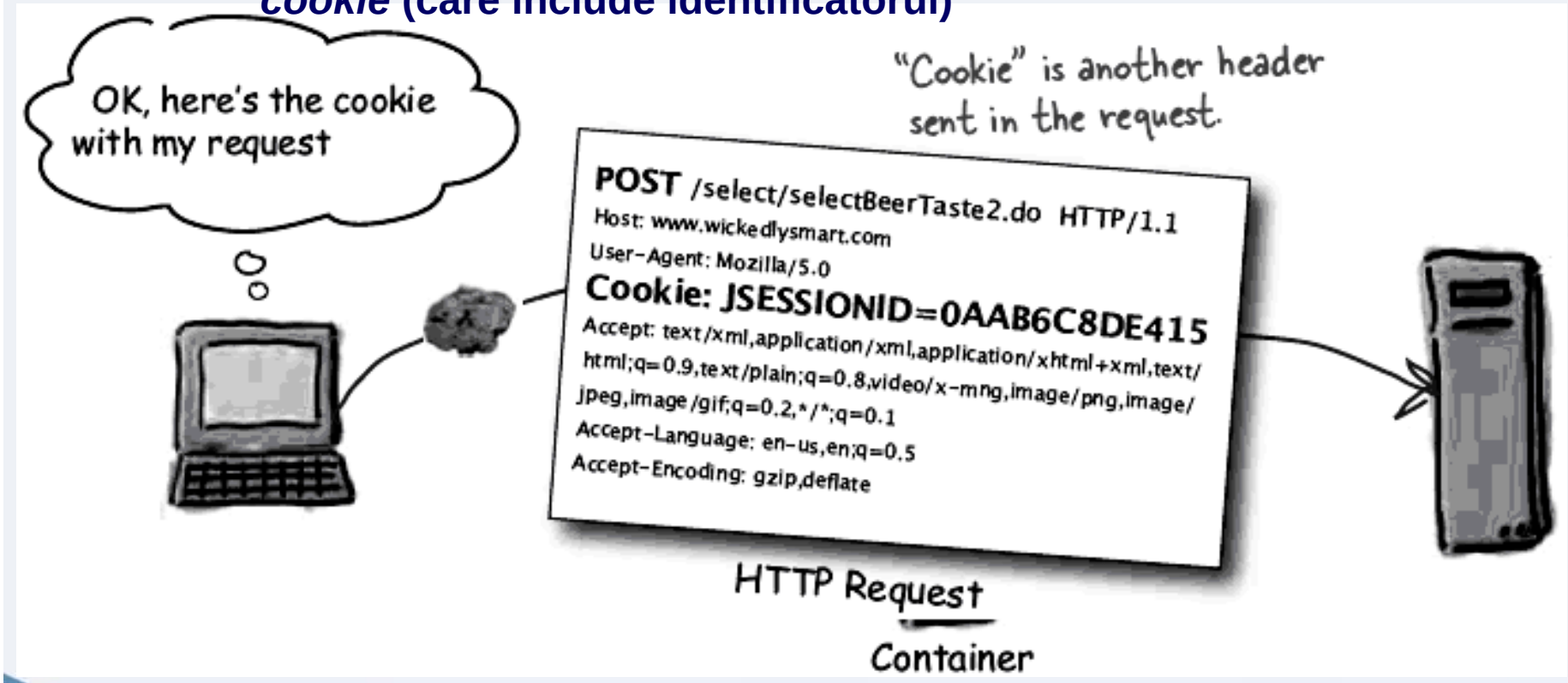


4.4. Tehnologii server. Java Servlet

Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

Clientul HTTP (broserul) este obligat sa retrimita odata cu cererile si *HTTP cookie* (care include identificatorul)

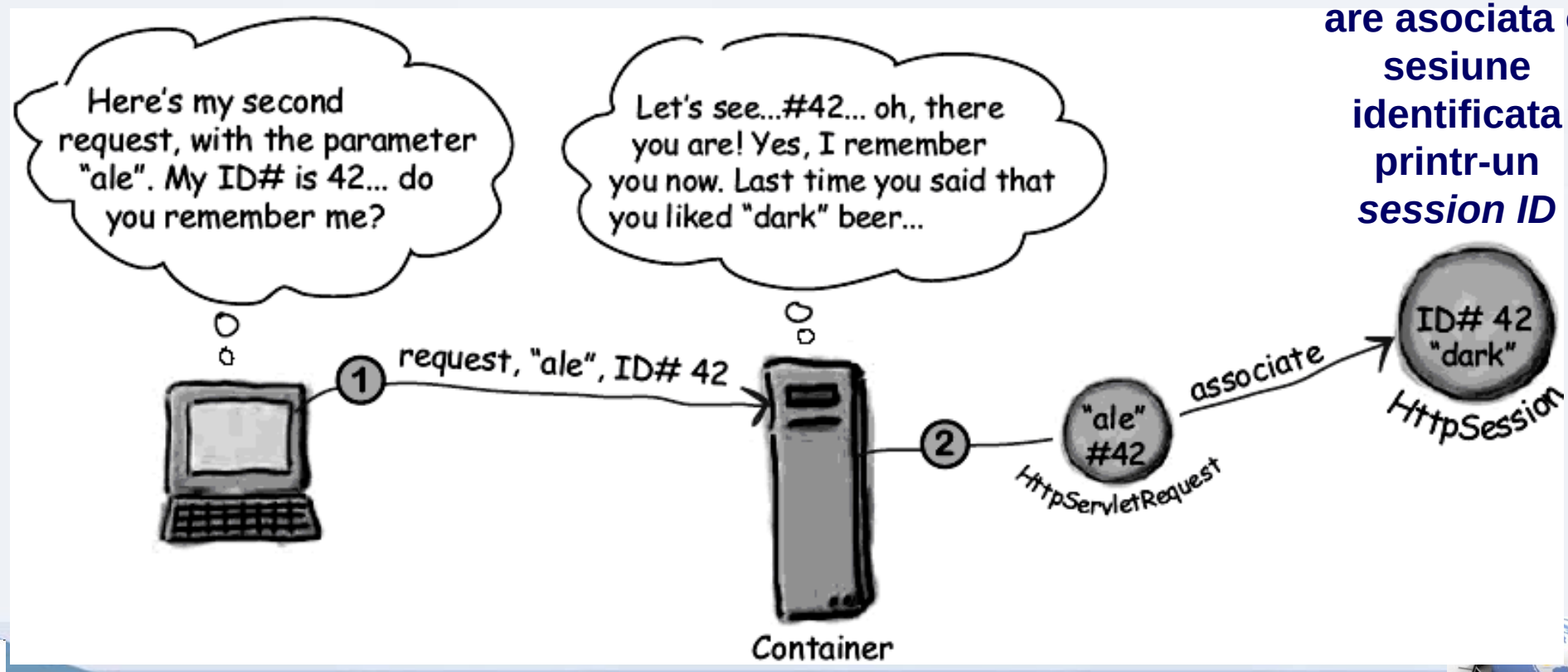


Tehnologia Java Servlet

Session tracking* – adaugarea "memoriei" in *servleturi

Accesul ulterior la servlet duce la **regasirea "sesiunii" pe baza identificatorului**

Fiecare client are asociata o sesiune identificata printr-un *session ID*



Tehnologia Java Servlet

Session tracking – adaugarea "memoriei" in *servleturi*

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws IOException, ServletException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("test session attributes<br>");


    HttpSession session = request.getSession();


    if (session.isNew()) {
       out.println("This is a new session.");
    } else {
       out.println("Welcome back!");
    }
}
```

getSession() returns a session no matter what... but you can't tell if it's a new session unless you ask the session.

isNew() returns true if the client has not yet responded with this session ID.

Varianta in care sesiunea este creata daca nu exista deja

Tehnologia Java Servlet

Session tracking – adaugarea "memoriei" in *servleturi*

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws IOException, ServletException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("test sessions<br>");

    HttpSession session = request.getSession(false);

    if (session==null) {
       out.println("no session was available");
       out.println("making one...");
       session = request.getSession();
    } else {
       out.println("there was a session!");
    }
}
```

Passing "false" means the method returns a pre-existing session, or null if there was no session associated with this client.

Now we can test for whether there was already a session (the no-arg getSession() would NEVER return null).

Here we KNOW we're making a new session.

Varianta in care se foloseste doar o sesiune preexistenta

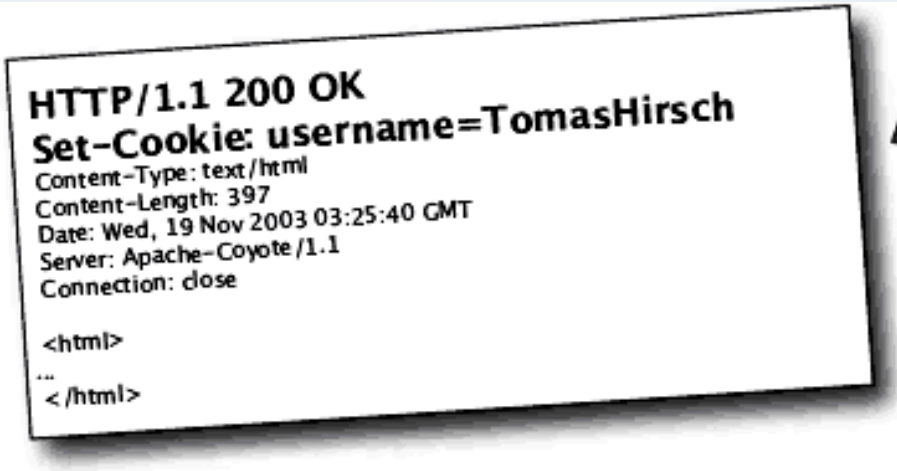
Tehnologia Java Servlet

Exemplificare – adaugarea "memoriei" in *servleturi*

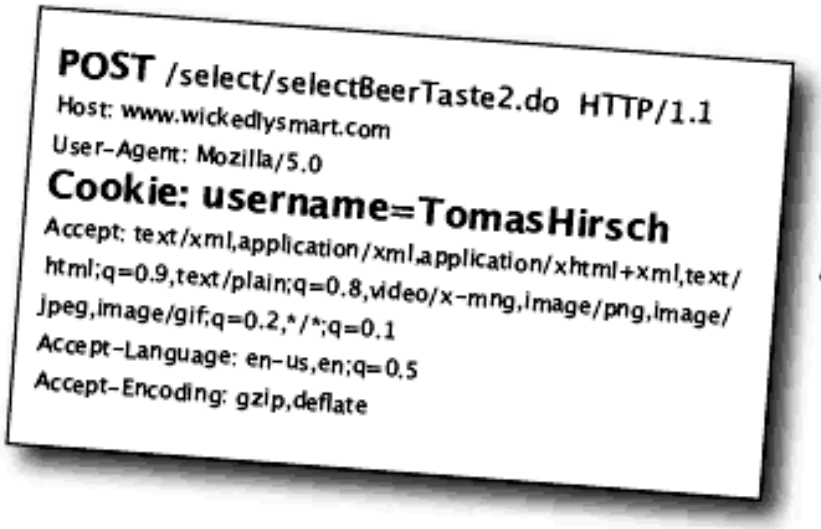
```
45      // Incercare de obtinere sesiune curenta – noua la cererea initiala
46      // respectiv cea cu IDul trimis de client, la cereri ulterioare
47      HttpSession ses = request.getSession();
48
49      // Incercare de obtinere atribut
50      // Reuseste DOAR la cereri ulterioare
51      Orar orar = (Orar) ses.getAttribute("orar");
52
53      // Daca nu exista orarul ca atribut al sesiunii – DOAR initial
54      if (orar == null) {
55
56         // Crearea obiectului care va fi "salvat"
57         // Ca atribut al sesiunii
58         orar = new Orar();
59
60         // Salvarea obiectului creat, ca atribut al sesiunii
61         ses.setAttribute("orar", orar);
62      }
```

Tehnologia Java Servlet

HTTP Cookies - Mesaje HTTP care contin antete *cookies*



Raspuns HTTP prin care **serverul trimite *cookieul*** pentru a fi **salvat de client**



Cerere HTTP prin care **clientul retrimite *cookieul* catre server**

Tehnologia Java Servlet

HTTP Cookies - Clase si interfete pentru lucrul cu *cookies*

Tehnologia Java Servlet

HTTP Cookies - Servlet care seteaza un *cookie*

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class CookieTest extends HttpServlet {
   public void doPost(HttpServletRequest request, HttpServletResponse response)
                                          throws IOException, ServletException {
      response.setContentType("text/html");

      String name = request.getParameter("username");

      Cookie cookie = new Cookie("username", name);

      cookie.setMaxAge(30*60);

      response.addCookie(cookie);


      RequestDispatcher view = request.getRequestDispatcher("cookieresult.jsp");
      view.forward(request, response);
   }
}
```

Get the user's name submitted in the form.

Make a new cookie so store the user's name.

Keep it alive on the client for 30 minutes.

Add the cookie as a "Set-Cookie" response header.

Let a JSP make the response page.

Tehnologia Java Servlet

HTTP Cookies - Servlet care citeste un *cookie*

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class CheckCookie extends HttpServlet {
   public void doGet(HttpServletRequest request, HttpServletResponse response)
                                          throws IOException, ServletException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      Cookie[] cookies = request.getCookies();

      for (int i = 0; i < cookies.length; i++) {
         Cookie cookie = cookies[i];
         if (cookie.getName().equals("username")) {
            String userName = cookie.getValue();
            out.println("Hello " + userName);
            break;
         }
      }
   }
}
```

Get the cookies from the request.

Loop through the cookie array looking for a cookie named "username". If there is one, get the value and print it.

Tehnologia Java Servlet

Session tracking – adaugarea "memoriei" in *servleturi*


```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                    throws IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();

    out.println("<html><body>");
    out.println("<a href=\"" + response.encodeURL("/BeerTest.do") + "\">click me</a>");
    out.println("</body></html>");
}
```

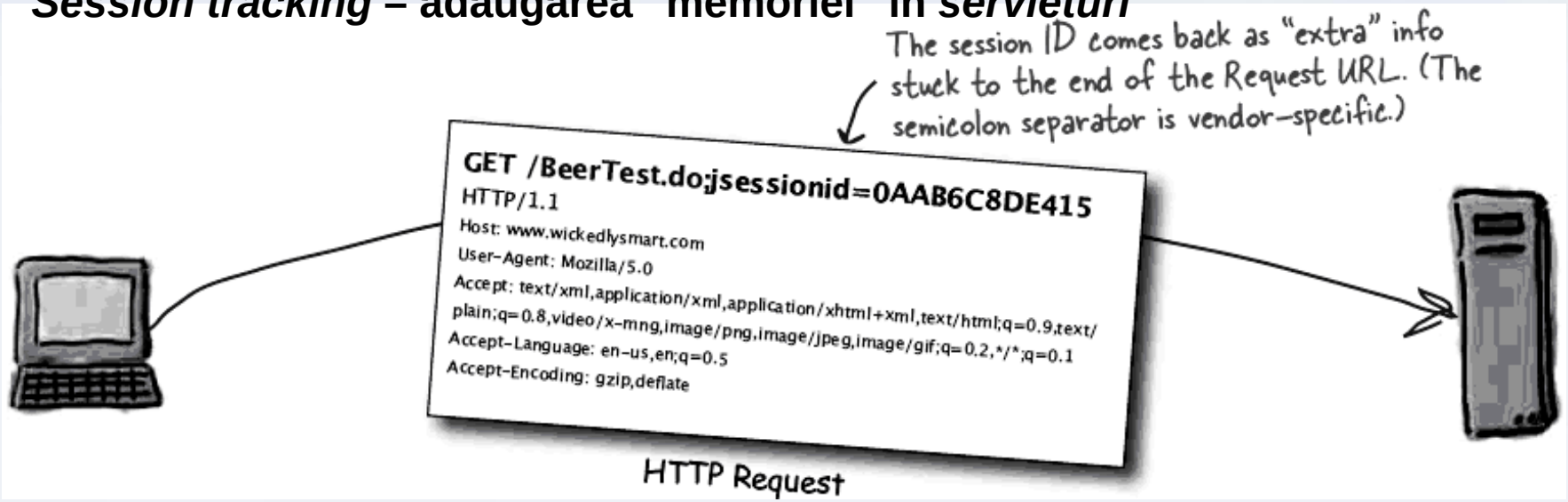
get a session

Add the extra session ID info to this URL.

Pentru cazul in care sunt **dezactivate *cookieurile*** se poate folosi **rescrierea URL-ului**

Tehnologia Java Servlet

Session tracking – adaugarea "memoriei" in *servleturi*




```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                    throws IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();

    out.println("<html><body>");
    out.println("<a href=\"" + response.encodeURL("/BeerTest.do") + "\">click me</a>");
    out.println("</body></html>");
}
```

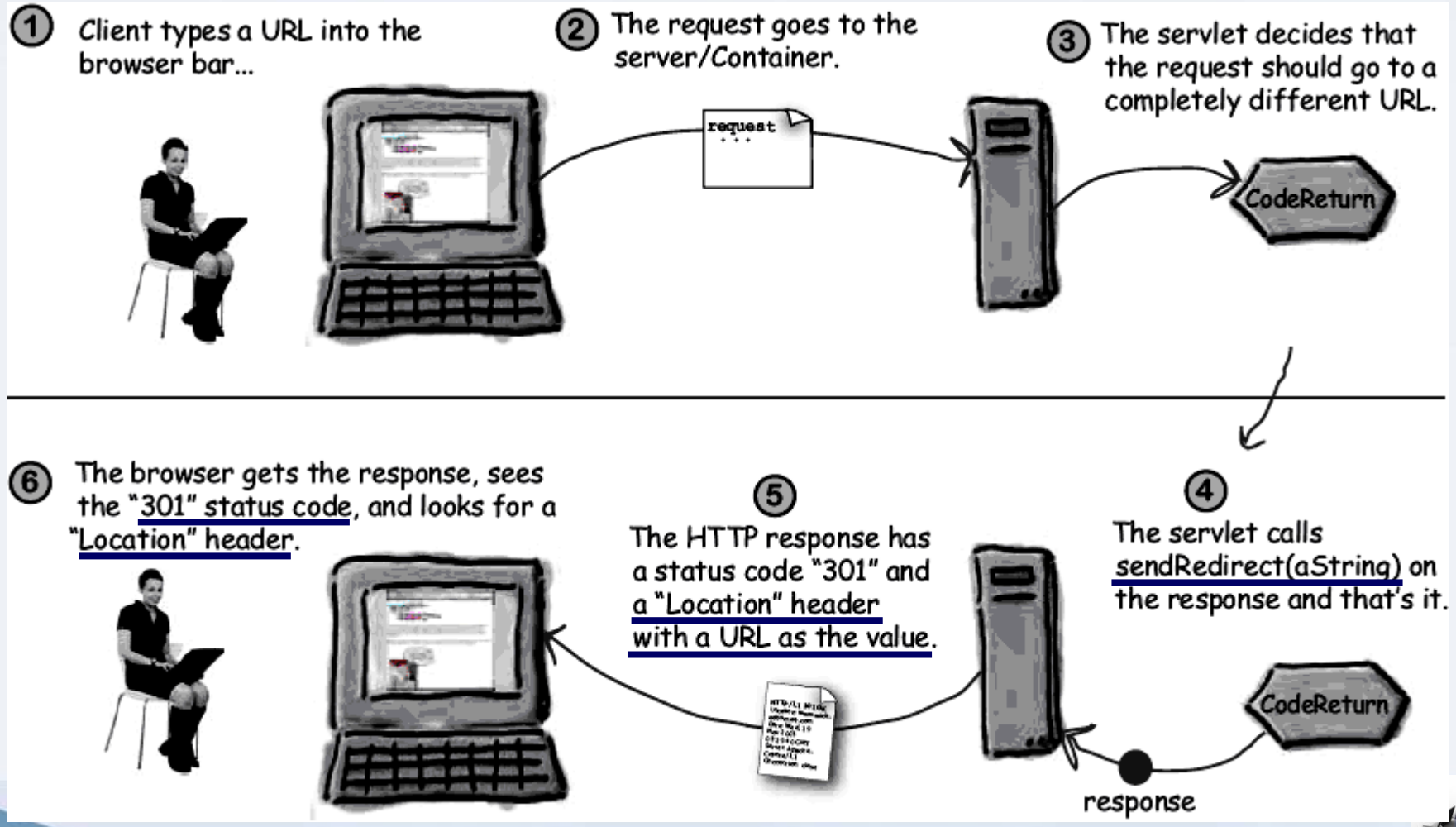
get a session

Add the extra session ID info to this URL.

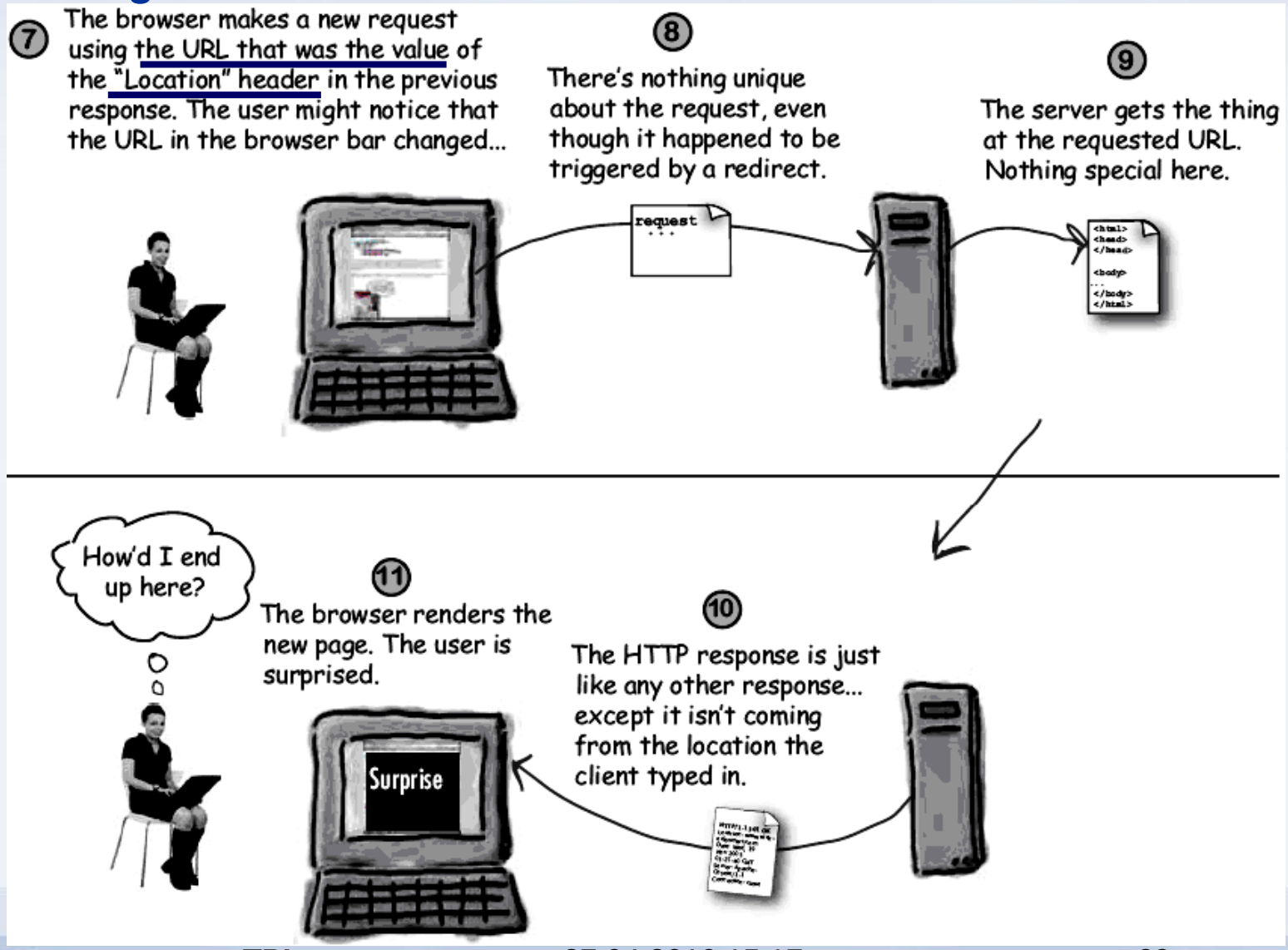
Pentru cazul in care sunt **dezactivate *cookieurile*** se poate folosi **rescrierea URL-ului**

Tehnologia Java Servlet

***Servleturile* pot redirecta *browserul* catre alt URL folosind sendRedirect()**

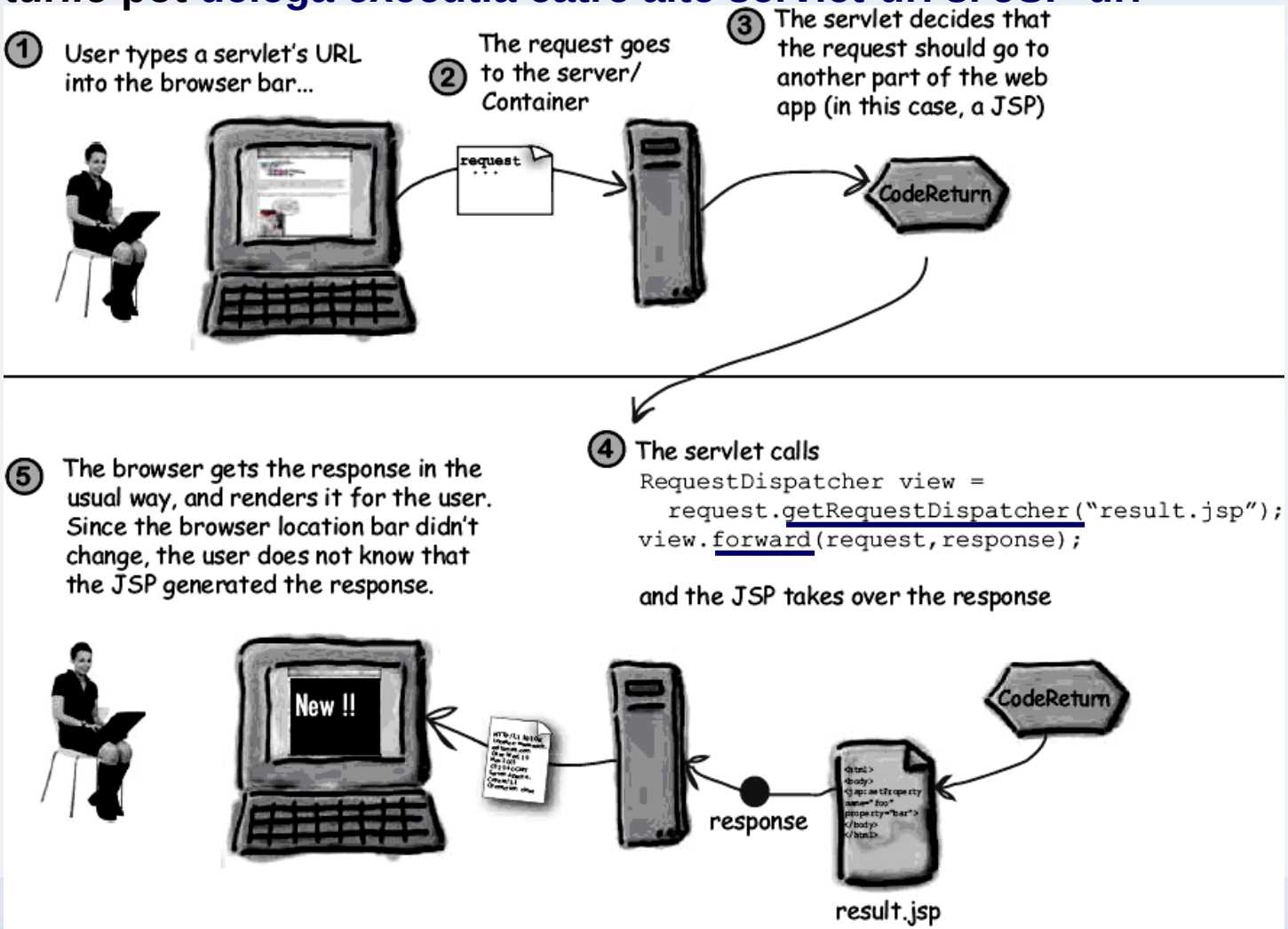


Tehnologia Java Servlet



Tehnologia Java Servlet

***Servleturile* pot delega executia catre alte servlet-uri si JSP-uri**



Tehnologia Java Servlet

Pentru a-si transmite informatii (inclusiv parametri de initializare)

- ***servleturile*** si **paginile JSP** care **deleaga executia**
  - pot crea **atribute ale cererii**
    - in cazul **paginilor JSP** un **obiect implicit** numit "**request**"
    - in cazul **servleturilor** obiectul **request** de tip **HTTPServletRequest**

  carora le dau ca valori **informatiile de transmis**

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");
```

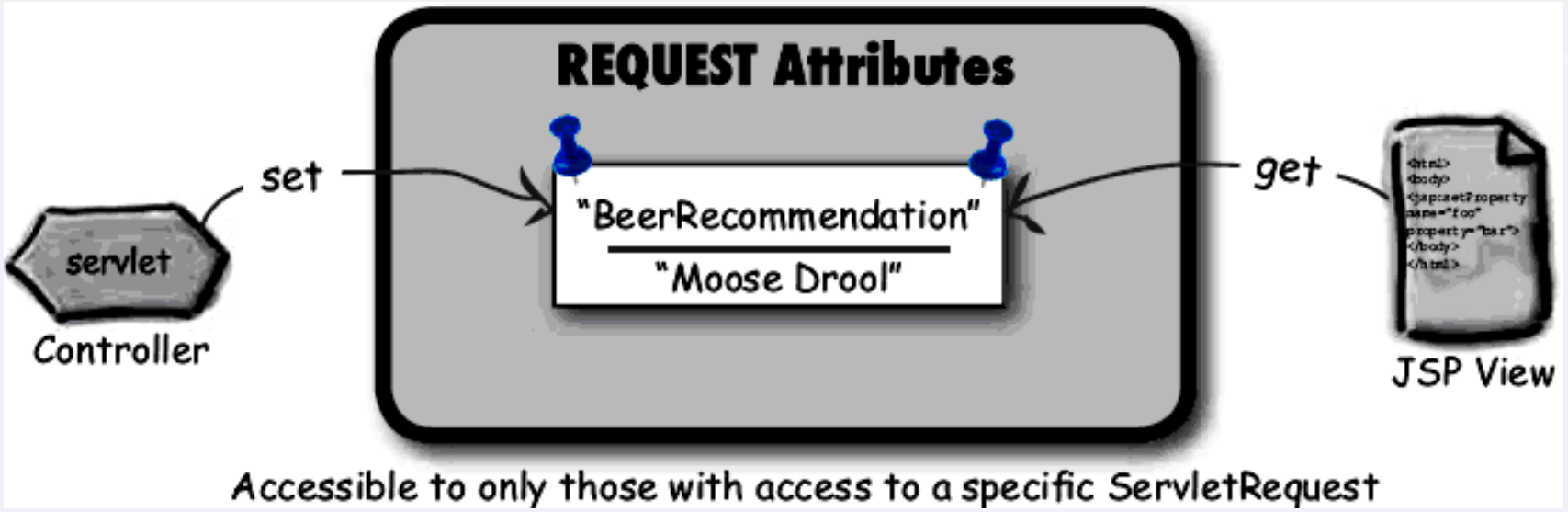
Tehnologia Java Servlet

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 Servlet

Pentru comunicatia intre toate componentele Web ale unei aplicatii Web formata din mai multe **servleturi** si **pagini JSP**

- pot fi de asemenea definite atribute, numite **atribute context**