Arhitectura unui server de Web

- Orice aplicatie Web contine cateva componente fixe:
  - Un program prin intermediul caruia se lanseaza cererile de catre utilizatori;
  - O baza de date care contine datele intr-o forma gata prelucrata;
  - Un program care stie sa prezinte rezultatele in forma asteptata de cel care incearca sa le acceseze;
- Tehnologii implicate:
  - HTML – reprezentare standard a formatarii textelor;
  - XML – reprezentare standard a datelor;
  - HTTP – protocol de nivel aplicatie – pentru transfer de informatie in format text;
  - Tehnologii bazate pe Java (servleti, JSP …)
- In functie de complexitatea aplicatiei, unele componente pot fi simplificate la extrem (de ex. daca sunt numai pagini statice, baza de date va contine numai aceste pagini) sau arhitectura poate fi foarte complexa (ca in cazul unor site-uri precum Amazon, Yahoo, eBay);
- Accesul la Internet se face cel mai usor prin intermediul unui browser (Firefox, IE, Chrome, Safari …); acesta discuta cu un server de la care obtine informatia si o prezinta; browserul poate executa si el aplicatii (Applet, JavaScript);
- Serverul de Web e un program care se executa pe masina (sistemul) server si care asteapta cereri pe care le trateaza pe masura ce apar; serverul poate sa construiasca o pagina in format HTML sau pur si simplu sa calculeze ceva, rezultatul transmitandu-l catre navigator.
- Cel mai cunoscut server de Web este Apache.

Arhitectura unui server de Web

- Solutia optima pentru aplicatia client-server este pe trei nivele:
  - Primul nivel: cel al prezentarii, este constituit dintr-o interfata grafica;
  - Al doilea nivel: codul caruia i se adreseaza cererile prin intermediul nivelului de prezentare;
  - Al treilea nivel: nivelul datelor, acestea putand fi orice baza de date, documente XML, un serviciu de nume;
- In contextul arhitecturii Web, cele 3 nivele sunt:
  - Primul nivel: navigatorul si serverul care trebuie sa asambleze datele intr-o forma afisabila;
  - Nivelul aplicatiei: program sau script care implementeaza logica aplicatiei (business logic);
  - Nivelul datelor: sursa de date, cel mai adesea o baza de date, nu neaparat relationala;

Construirea si executia cererilor conform protocolului HTTP

- Pentru a face o cerere la server, browserul trebuie sa emita o cerere in format HTTP
- Protocolul HTTP este un protocol de nivel aplicatie implementat peste TCP/IP
- HTTP este un protocol fara stari si bazat pe cereri si raspunsuri
- Clientul initiaza intotdeauna cererea; clientul stabileste conexiunea; serverul hotaraste cand sa inchida conexiunea
- Orice cerere sau raspuns HTTP consta din 3 parti:
  - Linia de cerere sau raspuns;
  - Un titlu (header);
  - Corpul (body);

Construirea si executia cererilor conform protocolului HTTP

- O cerere HTTP consta din:
  - Un nume de comanda;
  - Un URL (pentru o pagina sau o linie de comanda pe care utilizatorul doreste sa o execute);
  - Datele care se transmit;
  - Versiunea de protocol HTTP utilizata;

```
GET          /index.html          HTTP/1.0
```

- Initial protocolul HTTP a fost construit pentru a transmite informatii statice intre client si server; GET si POST aveau rol de a aduce, respectiv transmite o pagina.
- In prezent, atat GET cat si POST sunt utilizate si pentru a solicita executia unor programe pe server.
- In cazul GET:
  - Dupa URL se adauga informatiile care trebuie transmise la server;
  - In mod implicit orice cerere este transmisa ca GET;
  - Cererea poate fi memorata (bookmark) si refolosita;
  - Daca in cache-ul local exista raspunsul pentru aceeasi cerere, atunci nu se mai executa script-ul si se preia raspunsul direct din cache;
- In cazul POST:
  - Textul care trebuie trimis se include in corpul (body) cererii (request-ului); dimensiunea nu mai este astfel limitata de dimensiunea maxima a URL-ului;
  - Cererea se va transmite intotdeauna la server (nu se foloseste cache);

Construirea si executia cererilor conform protocolului HTTP

- Daca resursa referita e o pagina HTML, executia serviciului inseamna doar transmiterea paginii in corpul raspunsului;
- Daca serviciul presupune executia unui program, tipul acestuia va fi determinat de extensia folosita;
- Ca raspuns la o cerere HTTP serverul raspunde cu starea raspunsului si cu alte informatii;
- Raspunsul are un titlu (header) si un corp (body); in header sunt informatii ca Date, Content-type, Expires; Corpul unui raspuns poate fi simplu text sau un text in html sau xml sau un fisier binar, imagine, sunet, video. In HTTP indicarea tipului se face folosind MIME in forma text/htmp, image/gif;
- Cel mai cunoscut server de Web este Apache; un astfel de server trebuie sa indeplineasca mai multe conditii:
  - Existenta unei interfete de programare API;
  - Tratarea fazei de executie; suport pentru serviciile de retea si alte elemente necesare in faza de executie;
  - Posibilitatea de a instala noi aplicatii pe server si facilitati pentru configurarea de aplicatii;

Servleti

- Servlet: aplicatie Java care se executa pe server (un Applet este o aplicatie Java care se executa pe client);
- Executia servletului se produce ca raspuns la primirea de catre server a unei cereri de tip HTTP a clientului;
- Pentru a izola programatorii de detalii neplacute precum legatura cu reteaua, acceptarea cererilor, producerea unor raspunsuri corect formatate, se utilizeaza un container sau o "servlet engine". Etapele tratarii unei cereri HTTP sunt:
  - Analiza cererii pentru a stabili daca se refera la o pagina statica sau o cerere care trebuie trimisa la containerul de servleti;
  - Daca este vorba de o cerere ce trebuie tratata de container, serverul o sa faca un apel corespunzator;
  - Daca containerul a fost invocat, acesta va determina tipul de aplicatie ce trebuie invocata. Containerul trebuie sa stie si sa faca legatura intre diferitele resurse instalate cu o aplicatie Web si numele prin care sunt invocate;
  - Daca containerul determina ca cererea trebuie facuta de un anume servlet fie il va instantia, fie il va folosi pe unul deja existent si ii va trimite cererea conform API-ului corespunzator;
  - Transmiterea cererii se face sub forma unor obiecte care incapsuleaza cererea si respectiv raspunsul.

Servleti

- In continuare consideram ca se utilizeaza containerul TomCat, asociat serverului de Web Apache. O aplicatie Web este formata din 4 componente:
  - Un director public; - contine aplicatia si resursele utilizate de aceasta;
  - Un fisier WEB-INF/web.XML – contine descrierea aplicatiei;
  - Un subdirector WEB-INF/class sau WEB-INF/classes – contine clasele compilate ale servletului;
  - Un subdirector WEB-INF/lib – contine bibilioteci, daca sunt folosite si biblioteci;

Exemplu: servlet care asambleaza un raspuns de tip text cu un mesaj de salut

HelloWorldHttpServlet.java

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


public class HelloWorldHttpServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public void service (HttpServletRequest req, HttpServletResponse res) throws IOException {
        res.setContentType("text/plain");
        ServletOutputStream os = res.getOutputStream();
        os.write("Hello, world !".getBytes());
        os.flush();
      }
      public String getServletInfo() {
            return "Hello World servlet";
      }
}
```

Arhitectura unui server de Web

web.xml

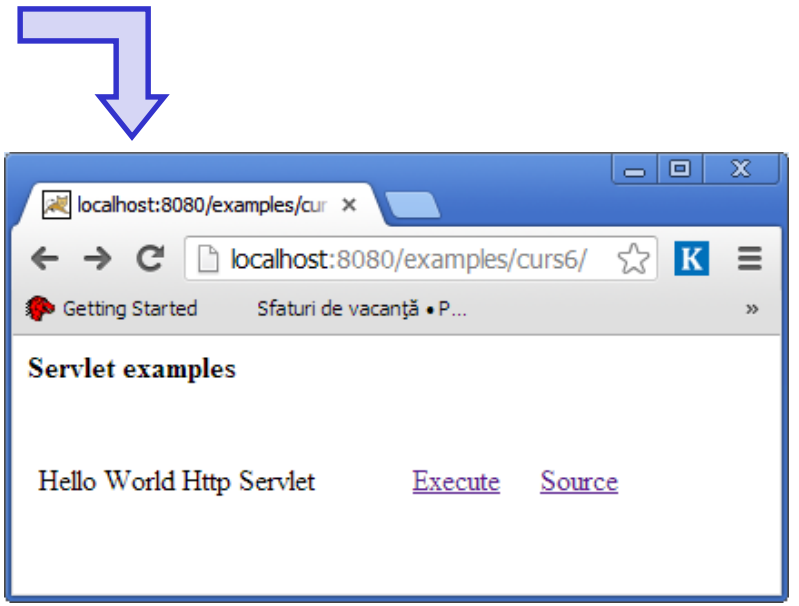
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<web-app>
  <servlet>
    <servlet-name>HelloWorldHttpServlet</servlet-name>
    <servlet-class>HelloWorldHttpServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloWorldHttpServlet</servlet-name>
    <url-pattern>/curs6/servletapplications/HelloWorldHttpServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

index.html

```
<html>
<table BORDER=0 CELLSPACING=5 WIDTH="85%" >
<tr VALIGN=TOP>
<p><strong>Servlet examples</strong></p>
<br>
<td>Hello World Http Servlet</td>
<td><a href="servletapplications/HelloWorldHttpServlet">Execute</a></td>
<td><a href="helloworldhttpservlet.html">Source</a></td>
</tr>
</table>

</body>
</html>
```

Apelul HelloWorldHttpServlet



Servleti

Exemplu: servlet care incrementeaza un contor de fiecare data cand primeste un request, si paseaza in raspuns valoarea curenta a contorului; cum poate fi accesat de mai multi clienti (browsere), portiunea critica se sincronizeaza

SimpleCounterServlet.java

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

public class SimpleCounterServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    int count = 0;
    public  void doGet(HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException {
        int localCounter;
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();
        synchronized(this) {
            localCounter = count++;
        }
    out.println("Servlet was used for " + localCounter + " times.");
    }
}
```

Servleti

Exemplu: servlet care citeste un nume trimis ca parametru in request si asambleaza in raspuns o pagina formatata html si continand numele primit ca parametru

HelloNameHttpServlet.java

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

public class HelloNameHttpServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        public void service (HttpServletRequest req, HttpServletResponse res) throws IOException {
            String name = req.getParameter("name");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.print("<html><head></head><body>");
            out.print("<p><strong>This program will print a greeting specially for you</strong></p>");
            out.print("<p>Hello,<b> " + name + " </b> !</p>");

            out.print("</body></html>");
            out.close();
        }
        public String getServletInfo() {
            return "Hello <<Name>> servlet";
        }
}
```

Servleti

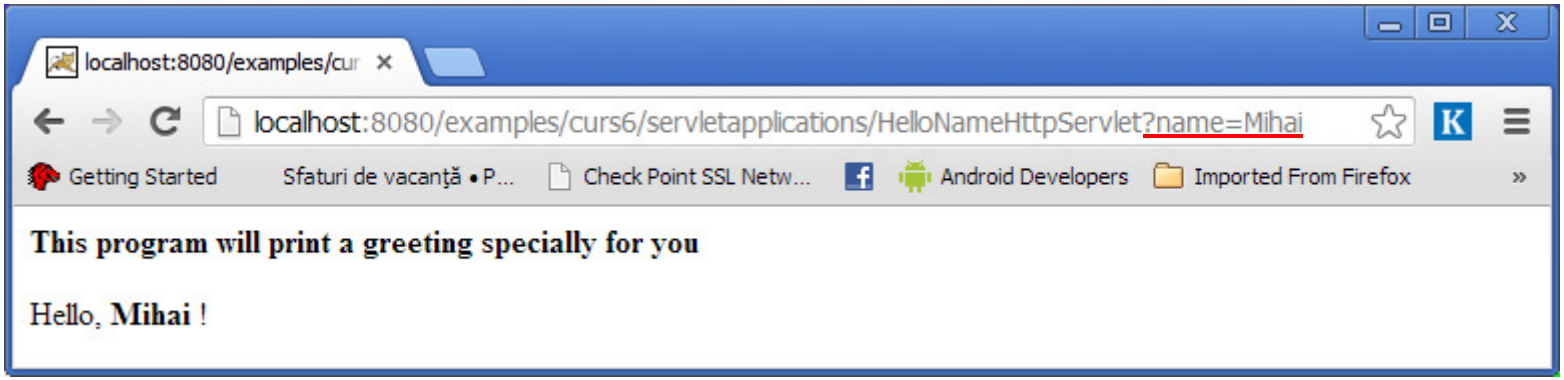
```
<tr>
<td>Simple Helo Name Servlet</td>

<td><form action="servletapplications/HelloNameHttpServlet" method="GET"
<p> Insert your name <input text="text" size="10" name="name">
<input type="submit" VALUE="Send"></td>
<td><a href="hellonamehttpservlet.html">Source</a></td>
</tr>
```

Simple Helo Name Servlet

Insert your name

Mihai Send Source



Servleti: sesiuni

Sesiuni

•Protocolul HTTP este stateless (protocol fara stari); se realizeaza prin cereri si raspunsuri care sunt tranzactii izolate. Pentru simpla navigare pe internet aceasta comportare este potrivita. Daca este insa vorba de o aplicatie, de multe ori este nevoie sa se faca o corelatie intre diferitele cereri, de exemplu sa se stie care sunt cererile care vin de la acelasi client. Solutiile utilizate sunt:

-Utilizarea de campuri ascunse;
-Rescrierea URL-urilor pentru a contine parametri suplimentari;
-Utilizarea de cookie-uri;
-Utilizarea de instrumente de trasare a sesiunilor;

- Rescrierea URL-urilor presupune ca toate paginile sa fie generate dinamic;
- Utilizarea de campuri ascunse in formulare continute in textele HTML pe care serverul le poate trimite ca rezultat permite transmiterea de informatii;

```
<INPUT TYPE="HIDDEN" NAME="name" VALUE="value">
```

- Un cookie este un fisier care contine perechi de tip cheie=valoare. Un astfel de fisier cookie este creat de catre server care il transmite ca instructiuni in partea de header a raspunsului HTTP.

```
Set-Cookie: Nume=valoare; Comment=Comentariu; Domain=domeniu;
Max-Age=SECUNDE; Path=cale; secure; Version=cifra
```

- Interfata Servlet contine si clasa Cookie. Un servlet poate sa creeze un obiect de tip Cookie, sa ii fixeze numele, valoarea, data de expirare.

Servleti: sesiuni

```
Cookie co = new Cookie("nume", "5");      // camp si valoare
co.setMaxAge(2*24*60*60);                 // doua zile
co.setDomain("www.elth.pub.ro");
co.setPath("/");

Res.addCookie(co);
```

- Pentru server exista clasa HttpSession. Obiectele instantiate din aceasta clasa vor memora informatiile despre client. Ex: intr-o aplicatie de tip magazin virtual se memoreaza informatiile despre client.

```
HttpSession sesiune = req.getSession(true); // se creeaza un obiect nou
Integer numar = new Integer(5);
Sesiune.setAttribute("numar", numar);    // se adauga info
```

- Secventa care preia o informatie din sesiune este:

```
HttpSession sesiune = req.getSession(true);
Integer numar = (Integer) sesiune.getAttribute("numar");
```

Servleti: sesiuni

Exemplu: servlet care publica un formular (pentru a fi completat din browser de un cumparator); fiecarui cumparator ii va corespunde o sesiune , el putand, in mai multe postari succesive, sa isi completeze lista de cumparaturi.

Item.java

```
public class Item {
        private String name;
        private int price;
        private int quantity;

        public Item(String name, int price, int quantity){
            this.name = name;
            this.price = price;
            this.quantity = quantity;
        }

        String getName() {
            return name;
        }
        int getPrice() {
            return price;
        }
        int getQuantity() {
            return quantity;
        }
    }
```

Clasa Item pastreaza informatiile despre un articol cumparat: nume, pret, cantitate; pentru fiecare cumparator, cosul sau de cumparaturi se salveaza in colectia Shopping

Shopping.java

```
import java.io.Serializable;
import java.util.Iterator;
import java.util.Vector;

public class Shopping implements Serializable {

    private static final long serialVersionUID = 1L;
    private String name;
    private Vector<Item> selectedItems;

    public Shopping() {
        selectedItems = new Vector<Item>();
    }
    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }
    public Iterator<Item> getSelected() {
        return this.selectedItems.iterator();
    }
    public boolean addSelected(String name, int price, int quantity){
        return  selectedItems.add(new Item(name, price, quantity));
    }
    public String toString(){
        return "Shopping for " + name;
    }
}
```

Servleti: sesiuni

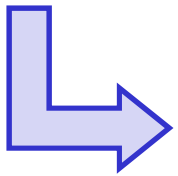
ShoppingServlet.java

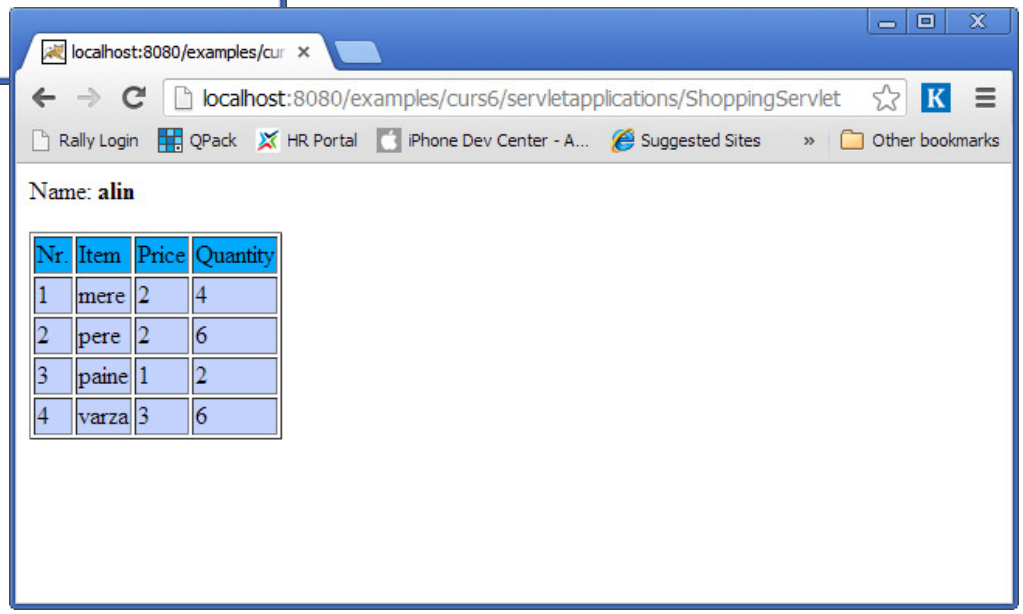
```
import javax.servlet.http.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Iterator;

public class ShoppingServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<html><body><center><h1>Welcome !</h1>");
        out.println("<form action = 'ShoppingServlet' method='POST'>");
        out.println("<p> Insert your name: <input type='text' size='10' name='name'></p>");
        out.println("<p> Insert an item: <input type='text' size='10' name='item'></p>");
        out.println("<p> Insert a price: <input type='text' size='10' name='price'></p>");
        out.println("<p> Insert a quantity: <input type='text' size='10' name='quantity'></p>");
        out.println("<input type='submit' value='Done'>");
        out.println("</form></center></body></html>");
    }
```

ShoppingServlet foloseste raspunsul (response) din doGet pentru a publica un formular pe pagina curenta, in care clientul (folosind browser-ul) poate completa datele cumparaturilor sale. Cand cumparatorul apasa pe butonul “Done”, se depune formularul folosind o cerere POST care va ingloba si datele introduse de client (in body-ul sau)

Servleti: sesiuni





Servleti: sesiuni

```
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws IOException {
        String name =  (String) req.getParameter("name");

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        Shopping shoppingBasket;
        HttpSession session = req.getSession(true);
        if(session.getAttribute("basket") == null) {
            shoppingBasket = new Shopping();
            session.setAttribute("basket", shoppingBasket);
            ((Shopping)session.getAttribute("basket")).setName(name);
        }
        else {
            shoppingBasket = (Shopping)session.getAttribute("basket");
        }

        String price = (String) req.getParameter("price");
        String quantity = (String) req.getParameter("quantity");
        String item = (String) req.getParameter("item");
        int nPrice = Integer.parseInt(price);
        int nQuantity = Integer.parseInt(quantity);

        shoppingBasket.setName(name);
        shoppingBasket.addSelected(item, nPrice, nQuantity);

        out.println("<html><body>");
        out.println("<p>Name: <strong>" + shoppingBasket.getName() + "</strong></p>");

        out.println("<table border='1'>");
        out.println("<tr bgColor='#00aaff'><td>Nr.</td><td>Item</td><td>Price</td><td>Quantity</td></tr>");
        Iterator<Item> itr = shoppingBasket.getSelected();
        int count = 0;
        while( itr.hasNext() ) {
            count++;
            Item value = (Item)itr.next();
            out.println("<tr bgColor='#c3d3fe'><td>"
                    + count + "</td><td>" + value.getName() + "</td><td>" + value.getPrice() +
                    "</td><td>" + value.getQuantity() + "</td></tr>");
        }
        out.println("</table>");
        out.println("</body></html>");
    }
}
```