# SwRTc – Laborator 3

# *Servlet-uri* Java

## *3.1. Descrierea laboratorului*

**In aceasta lucrare de laborator vor fi acoperite (partial) urmatoarele probleme:**

**- arhitectura aplicatiilor Web bazate pe *servlet-uri* Java si dezvoltarea lor**

**- descrierea si ilustrarea interactiunii dintre elementele aplicatiilor Web bazate pe *servlet-uri*** (**clientul** Web - *browser-ul*, **serverul** Web, **containerul** de *servlet-uri*, ***servlet-urile*** Java, etc.)

**- obtinerea, instalarea si configurarea serverelor Web cu suport pentru *servlet-uri* Java**

**- instalarea, compilarea si invocarea *servlet-urilor* Java**

**- exemple de servlet-uri comentate (sursele principale pentru tema de casa: `Polinom1.html`, `PolinomServlet1.java`, `Polinom1.java`)**

**Linkuri utile (locale):**

**- jakarta-tomcat-4.1.30.zip**

**- jakarta-tomcat-5.0.25.zip**

**- jakarta-tomcat-5.0.25_de_dezarhivat_pe_C.zip (Dupa dezarhivare, in directorul 'jakarta-tomcat-5.0.25' pe C:\, se da variabilei de mediu 'CATALINA_HOME' valoarea '`c:\jakarta-tomcat-5.0.25`', se ruleaza serverul cu 'startup.bat' din subdirectorul '/bin', si se incarca pagina 'index.html' din subdirectorul 'webapps\laborator'. Pentru oprirea serverului se foloseste 'shutdown.bat' din subdirectorul '/bin')**

## *3.2. Arhitectura si dezvoltarea aplicatiilor Web bazate pe servlet-uri Java*

### 3.2.1. Introducere in *servlet-uri*

**O buna introducere in *servlet-uri*** este **sectiunea dedicata *servlet-urilor* din Tutorialul Java** care cuprinde, printre altele:

***Overview of Servlets***
- *Architecture of the Servlet Package*
- *A Simple Servlet*
- *The Example Servlets*

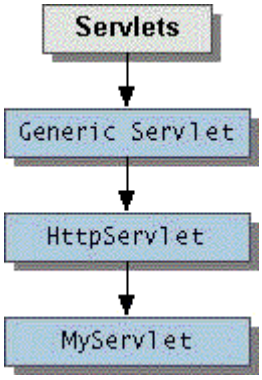
***Interacting with Clients***
- *Requests and Responses*
- *Handling GET and POST Requests*
- *Providing a Servlet Description*

***The Servlet Life Cycle***
- *Initializing a Servlet*
- *Destroying a Servlet*

***Saving Client State***
- *Session Tracking*
- *Using Cookies*

***Running Servlets***
- *Configuring Tomcat Servlets*
- *Configuring and Running Tomcat*
- *Configuring JSDK Servlets*

**Arhitectura pachetelor *servlet* este urmatoarea:**




Urmatorul ***servlet* Java** simplu **exemplifica tratarea cererilor HTML care utilizeaza metoda GET**.

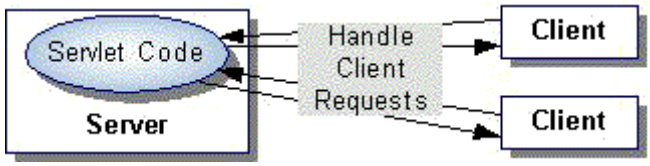
```
 1  import javax.servlet.*;
 2  import javax.servlet.http.*;
 3  import java.io.*;
 4
 5  public class SalutServletGet extends HttpServlet {
 6
 7     // Tratarea cererii "get" de la clienti
 8     protected void doGet(HttpServletRequest cerere,
 9                          HttpServletResponse raspuns)
10                          throws ServletException, IOException {
11        raspuns.setContentType("text/html");
12        PrintWriter out = raspuns.getWriter();
13
14        // Trimiterea paginii XHTML catre client
15
16        // Inceputul documentului XHTML
17        out.println( "<?xml version = \"1.0\"?>" );
18
19        out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
20           "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
21           "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
22
23        out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
24
25        // Sectiunea antet a documentului
26        out.println( "<head>" );
27        out.println( "<title>Un exemplu de servlet simplu</title>" );
28        out.println( "</head>" );
29
30        // Sectiunea corp a documentului
31        out.println( "<body>" );
32        out.println( "<h1>Bun venit la servlet-uri!</h1>" );
33        out.println( "</body>" );
34
35        // Sfarsitul documentului
36        out.println( "</html>" );
37
38        // Inchiderea fluxului pentru a incheia pagina
39        out.close();
40     }
41  }
```

**Raspunsul HTML generat de *servlet* este o pagina XHTML.**

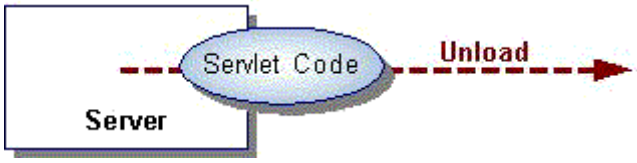
**Ciclul de viata al *servlet-urilor* Java:**

**- Serverul Web incarca si initializeaza** *servlet-ul*


- ***Servlet-ul* trateaza** zero sau mai multe **cereri de la client**




- **Serverul Web elimina** *servlet-ul* (unele servere fac asta doar atunci cand sunt oprite)




## *3.3. Instalarea si configurarea serverelor Web cu suport pentru servlet-uri*

### 3.3.1. Servere Web cu suport pentru *servlet-uri*

Mai multe **servere Web ofera suport pentru *servlet-uri* Java** (numit ***container*** **sau *motor* de *servlet-uri***).

Urmatoarea lista prezinta **cateva dintre acestea, si adresele Web** (URL-urile) la care pot fi ele gasite:

**- Apache Tomcat (implementare de referinta oficiala a specificatiilor *servlet* si JSP)**:
**http://jakarta.apache.org/**

**- JSWDK de la Sun (implementare de referinta oficiala a specificatiilor *servlet* si JSP)**:
**http://java.sun.com/products/servlet/download.html**

**- JRun de la Allaire (motor de *servlet-uri, plug-in* pentru servere Netscape, IIS, etc.)**:
**http://www.allaire.com/products/jrun/**

**- ServletExec de la New Atlanta(motor de *servlet-uri, plug-in* pentru diferite servere Web)**:
**http://newatlanta.com/**

**- LiteWebServer**:
**http://www.gefionsoftware.com/**

**- Java Web Server de la Sun (Server Web cu suport pentru *servlet* si JSP)**:
**http://www.sun.com/software/jwebserver/try/**

Pentru **documentatia claselor (API-urilor) *servlet* si JSP** pot fi utilizate urmatoarele adrese Web:

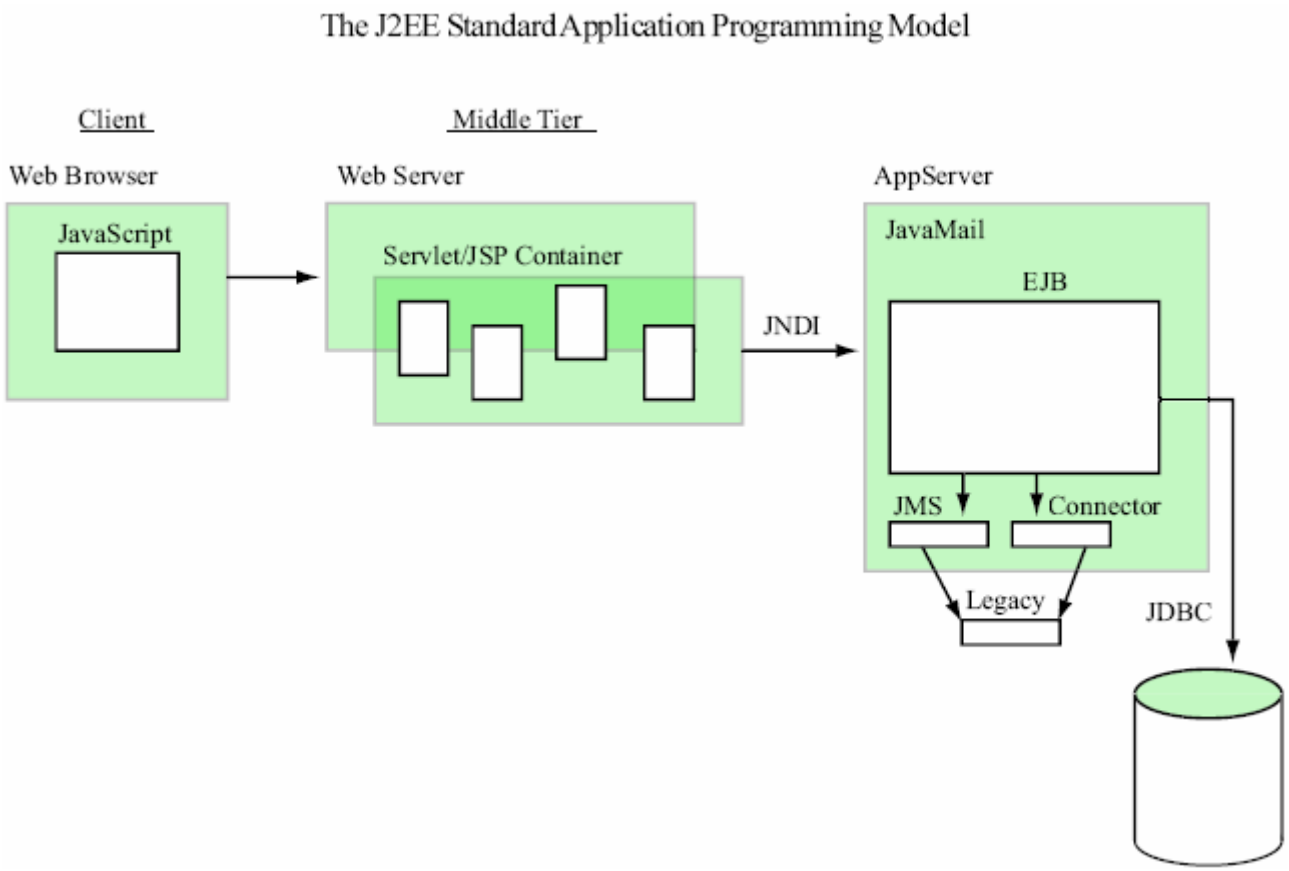
- **http://java.sun.com/products/jsp/download.html** - pentru **descarcarea (*download-ul*) API-urilor *servlet* si JSP**

- **http://java.sun.com/products/servlet/2.2/javadoc/** - pentru **accesul *on line* la API-ul *servlet***

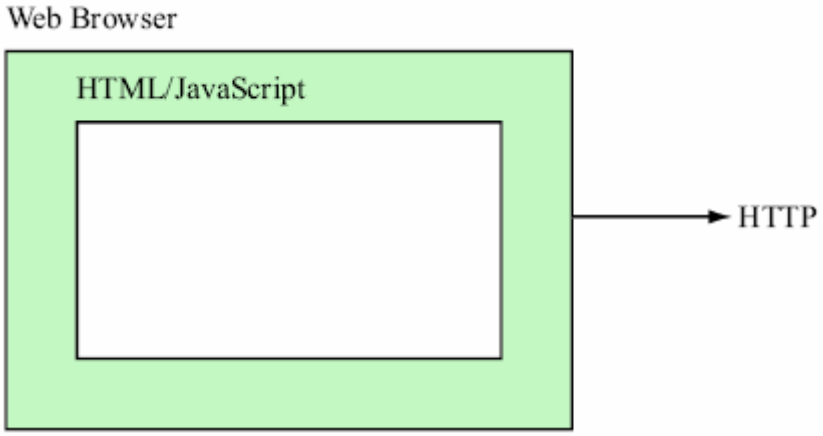
- **http://www.java.sun.com/j2ee/j2sdkee/techdocs/api/**- pentru **accesul la API-ul J2EE care include pachetele *servlet* si JSP**

## 6.3.2. Serverul Java Web Server de la Sun (cu suport pentru *servlet-uri*)

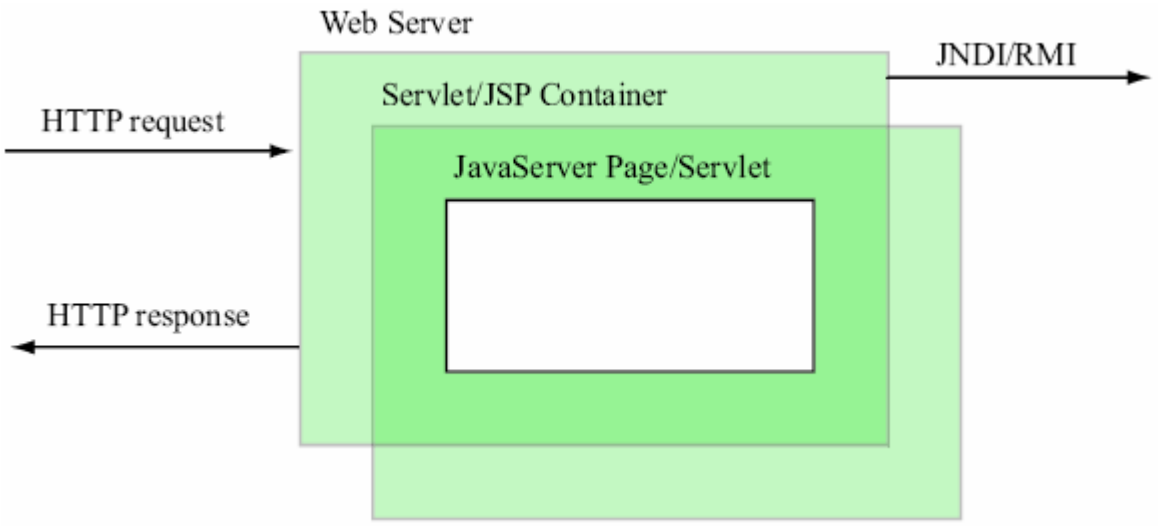
**Modelul standard in trei straturi (*three tier*) de programare a aplicatiilor J2EE** este urmatorul:




**Stratul client Web** (*client tier*) defineste **interfata cu utilizatorul** (*browser-ul* Web, client HTTP/HTML) este urmatorul:

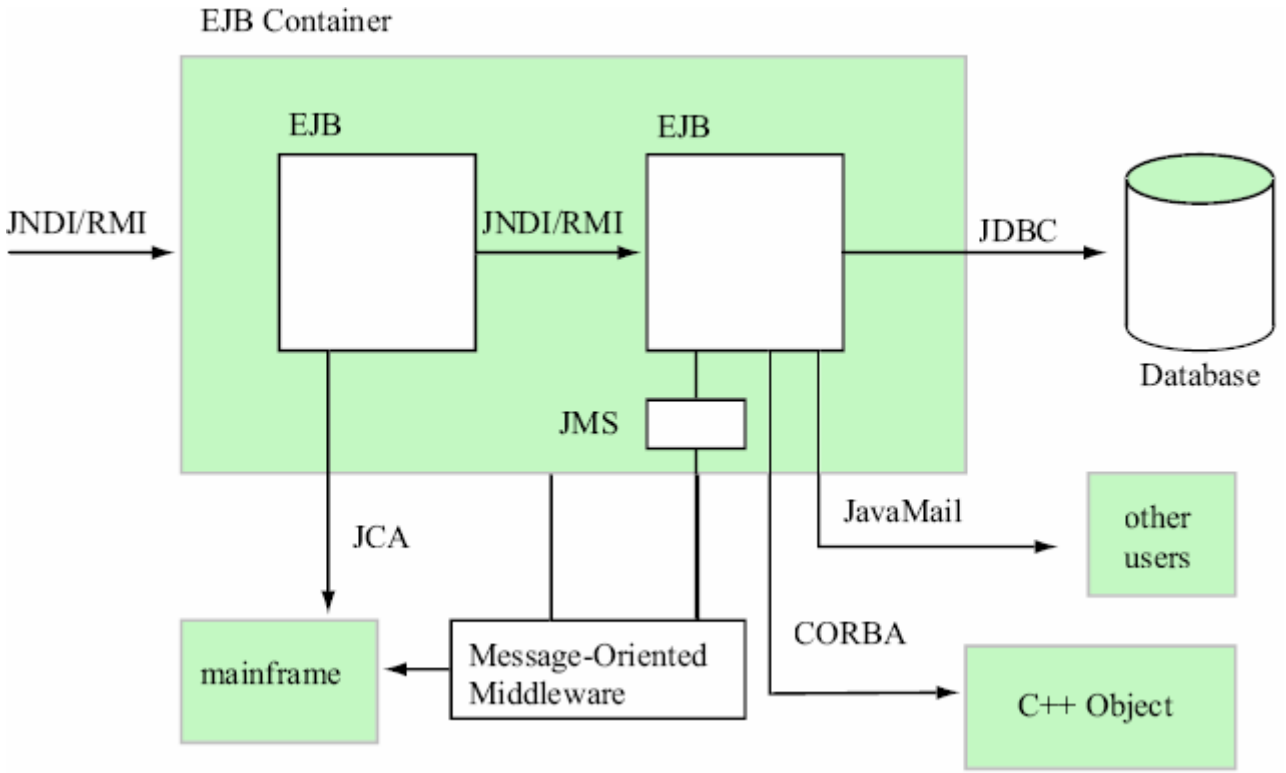


**Stratul server Web** (*middle tier*), **server HTTP care contine containere (motoare - *engines*) pentru *servlet-uri* si JSP-uri**, defineste **logica prezentarii** (genereaza si trimite catre client continut HTML, comunica cu aplicatia, etc.):




**JSP-urile (*JavaServer Pages*) sunt** o **combinatie de elemente HTML, date si *script-uri* JSP**, **convertite in *servlet-uri* in containerul *servlet*/JSP.**

**Stratul server de aplicatie** (*application server tier*), care include un container EJB (*Entreprise Java Beans*), defineste **logica aplicatiei Web** (*business logic*):



## 3.3.2. Serverul Web Tomcat de la Apache (cu suport pentru *servlet-uri*)

**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 server Web**, astfel incat **poate fi utilizat drept container de sine statator pentru a testa *servlet-uri* si a JSP-uri**.

Tomcat **poate fi configurat si pentru a servi** drept **container pentru *servlet-uri* si JSP-uri utilizat ca extensie a unor servere Web populare** cum ar fi ***Apache HTTP Server*** de la **Apache Software Foundation** sau ***Internet Information Services* (IIS)** de la **Microsoft**.

Tomcat este **integrat in implementarea de referinta** de la **Sun Microsystems** inclusa in **platforma Java pentru *e-business*** numita ***Java 2 Enterprise Edition* (J2EE)**.

**Ultima versiune de Tomcat poate fi obtinuta de la adresa Web**:

`http://jakarta.apache.org/site/binindex.cgi`

## 3.3.3. Instalarea serverului Web Tomcat de la Apache

**Pentru instalarea serverului Web Tomcat trebuie realizati urmatorii pasi**:

**1. Se obtine o versiune a distributiei serverului Web Tomcat.**

**In laborator se va utiliza versiunea 4.1.30 pentru Windows**. In acest caz **implementarea Tomcat** completa este **continuta in fisiere care incep cu:**

```
jakarta-tomcat-4.1.30
```

**Distributiile sunt oferite in fisiere arhiva** cu extensii **zip**, **exe, tar** comprimat **gz**, etc.

**2. Se extrage continutul distributiei intr-un director de pe *hard disk*.**

**In laborator s-au utilizat arhivele (distributiile):**

```
jakarta-tomcat-4.1.30.zip
jakarta-tomcat-5.0.25.zip
```

din care **s-au extras toate fisierele**, folosind caile definite in interiorul arhivei, **in directoarele:**

```
jakarta-tomcat-4.1.30\
jakarta-tomcat-5.0.25\
```

Din motive care tin de organizarea unitatilor de *hard disk* si de drepturile de acces **la calculatoarele din laborator a fost aleasa unitatea** de *hard disk* `d:\`.

Astfel, **calea completa a instalarii este:**

```
d:\jakarta-tomcat-4.1.30\
d:\jakarta-tomcat-5.0.25\
```

## 3.2.4. Configurarea serverului Web Tomcat de la Apache

**Pentru configurarea serverului Web Tomcat trebuie configurate variabilele de mediu** (*environment variables*) **`JAVA_HOME`** si **`CATALINA_HOME:`**

**1. Se configureaza variabila de mediu `JAVA_HOME`:**

**`JAVA_HOME`** trebuie sa indice directorul in care se afla instalarea Java curent utilizata. In cazul nostru, instalarea **`j2sdk`** versiunea **`1.4.2_04`** se afla pe unitatea **`c:\`**:

```
c:\j2sdk1.4.2_04
```

**2. Se configureaza variabila de mediu `CATALINA_HOME`:**

**`CATALINA_HOME`** trebuie sa indice directorul in care se afla instalarea Tomcat curent utilizata. **In cazul nostru**:

```
d:\jakarta-tomcat-4.1.30
```

iar in cazul utilizarii unitatii **`c:\`**:

```
c:\jakarta-tomcat-4.1.30
```

## 3.3.4. Lansarea, testarea si oprirea serverului Web Tomcat de la Apache

Dupa stabilirea variabilelor de mediu, se poate lansa serverul Tomcat.

**Se deschide o fereastra consola (*command prompt* sau *shell*) si se intra in** subdirectorul **`bin`** din directorul instalarii **`jakarta-tomcat-4.1.30`**. In directorul **`bin`** se afla fisierele **`startup.bat`** si **`shutdown.bat`** pentru lansarea si oprirea serverului Tomcat sub Windows, si fisierele **`startup.sh`** si **`shutdown.sh`** pentru lansarea si oprirea serverului Tomcat sub UNIX/Linux.

**Pentru a lansa serverul, se da comanda**:

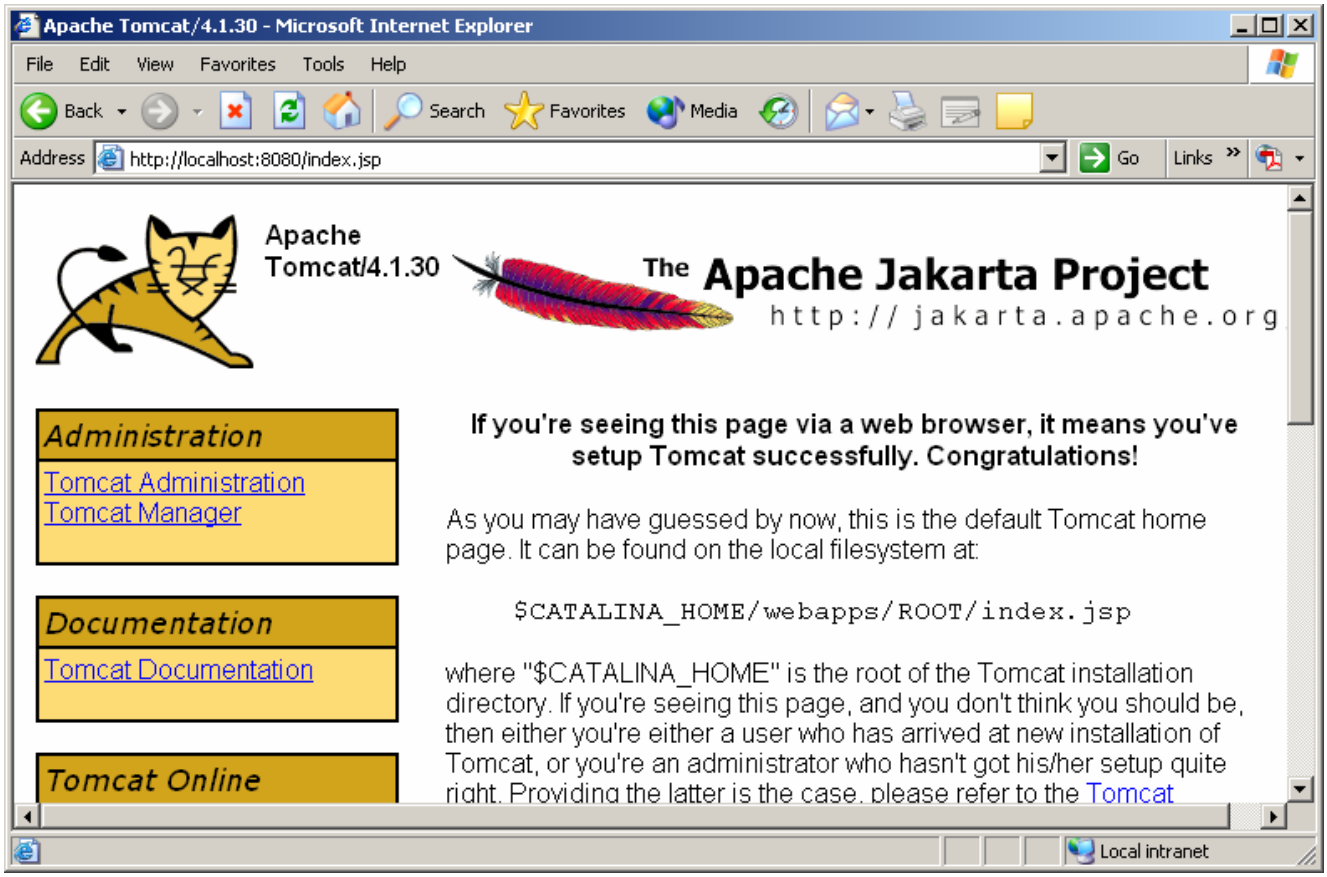
```
startup
```

care **lanseaza serverul Tomcat**, executat **pe portul TCP cu numarul 8080** pentru a preveni conflicte cu serverele Web standard (executate de obicei pe portul TCP cu numarul 80).

**Pentru a verifica faptul ca serverul Tomcat este executat si poate raspunde la cereri, se deschide un *browser* Web si se acceseaza** URL–ul:

`http://localhost:8080/`

**Continutul *browser-ului* va fi** urmatorul (**pagina documentatiei Tomcat**). Masina **`localhost`** indica *browser-ului* ca trebuie sa ceara pagina principala a serverului Tomcat de pe calculatorul local.




Masina IP **`localhost`** corespunde adresei IP **`127.0.0.1`**. Daca pagina principala a documentatiei serverului Tomcat nu e afisata, se poate incerca URL-ul:

**`http://127.0.0.1:8080/`**

Pentru a opri executia serverului Tomcat, in consola (*command prompt* sau *shell*) se da comanda:

```
shutdown
```

**In pagina Web a documentatiei Tomcat se afla urmatoarele adrese** (la sectiunea **Examples**):

- **Exemple de *servlet-uri* (tomcat 4.1.30)**

- **Exemple de *servlet-uri* (tomcat 5.0.25)**

- Exemple de JSP-uri

- Exemple de JSP-uri

## *3.4. Dezvoltarea servlet-urilor Java*

### 3.4.1. Instalarea *servlet-urilor* Java

***Servlet-urile*, JSP-urile si fisierele asociate lor sunt dezvoltate ca parte a unor aplicatii Web**. In mod normal, **aplicatiile Web sunt dezvoltate (instalate) in** subdirectorul **`webapps`** al instalarii **`jakarta-tomcat-4.1.30`**.

**O aplicatie Web are o structura de directoare prestabilita in care se plaseaza diferitele tipuri de fisiere** care fac parte din aplicatia Web.

Aceasta **structura de directoare poate fi creata** de administratorul serverului **in directorul `webapps`**, **sau** intreaga structura de directoare **poate fi arhivata intr-un fisier arhiva a aplicatiei Web**, cunoscut ca fisier **WAR** (*Web ARchive*), avand extensia **`.war`**. Daca un fisier WAR e plasat in directorul **`webapps`**, atunci, cand serverul Tomcat isi incepe executia, el extrage continutul fisierului WAR in structura corespunzatoare de subdirectoare ale directorului **`webapps`**.

**Structura de directoare a aplicatiei Web contine un director** numit ***context root*** (radacina a contextului), **directorul de cel mai inalt nivel al intregii aplicatii Web**, si mai multe subdirectoare:

| Director | Descriere |
|---|---|
| ***context root*** | **Directorul radacina al aplicatiei Web.** Toate documentele HTML, *servlet-urile*, JSP-urile si celelalte fisiere necesare unei aplicatii Web, cum ar fi imaginile si fisierele .class, sunt plasate in acest director sau in subdirectoarele lui.<br>**Numele** acestui director **este specificat de catre creatorul aplicatiei Web**.<br>**Pentru a structura aplicatia Web** pot fi create subdirectoare ale directorului *context root*. De exemplu, daca aplicatia Web foloseste mai multe imagini, ele pot fi plasate intr-un subdirector ale directorului *context root*.<br>**Exemplele din acest laborator sunt plasate in** directorul *context root* numit **`lab_servlet`**. |
| *context root*/ **`WEB-INF`** | Acest subdirector al directorului *context root* **contine descriptorul de dezvoltare a aplicatiei Web** (*Web application deployment descriptor*), numit **`web.xml`**. |
| *context root*/ `WEB-INF/`**`classes`** | Acest subdirector al directorului *context root* **contine fisierele cod de octeti (`.class`) ale *servlet-ului* si celelalte fisiere necesare aplicatiei Web**. Daca clasele *servlet-ului* sunt parte a unui pachet de clase (*package*), structura de directoare completa a pachetului de clase va incepe din acest director. |
| *context root*/ `WEB-INF/`**`lib`** | Acest subdirector al directorului *context root* **contine fisierele arhiva Java (`.jar`) ale *servlet-ului***. Fisierele JAR pot contine fisierele cod de octeti (**`.class`**) ale *servlet-ului* si celelalte fisiere necesare aplicatiei Web. |

**Configurarea directorului *context root* al aplicatiei Web in Tomcat** presupune crearea unui subdirector in directorul **`webapps`**. Cand Tomcat isi incepe executia, el creaza un *context root* pentru fiecare subdirector al **`webapps`**, folosind fiecare nume de subdirector ca nume de *context root*.

**Pentru a testa exemplele din acest laborator, se va crea un subdirector** **`lab_servlet`** **in** directorul **`webapps`** al instalarii serverului Tomcat.

Dupa configurarea directorului *context root*, **trebuie configurata aplicatia Web pentru a trata (*handle*) cererile venite de la clienti**. Aceasta configurare se face **intr-un descriptor de dezvoltare** (*deployment descriptor*), care este stocat intr-un fisier XML numit **`web.xml`**.

Descriptorul de dezvoltare **specifica diferitii parametri de configurare** cum ar fi **numele utilizat pentru invocarea *servlet-ului*** (***alias-ul*** *servlet-ului*), **o descriere a *servlet-ului*, numele clasei *servlet-ului*** complet calificat si **o translatie a *servlet-ului*** (*servlet* ***mapping***) care reprezinta **calea sau caile care produc invocarea *servlet-ului* de catre containerul *servlet-ului***.

Pentru aceasta **trebuie creat fisierul `web.xml`**. Mai multe instrumente *software* de dezvoltare a aplicatiilor Web in Java pot crea fisierul **`web.xml`** in mod automat.

**Continutul** fisierului **`web.xml`** **pentru primul exemplu din acest laborator** este prezentat in continuare.

```
 1  <!DOCTYPE web-app PUBLIC
 2     "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
 3     "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
 4
 5  <web-app>
 6
 7     <!-- Descrierea generala a aplicatiei Web -->
 8     <display-name>
 9        Exemple pentru laboratorul de servlet-uri
10     </display-name>
11
12     <description>
13        Aceasta este aplicatia Web in care sunt
14        exemplificate serlvet-urile si JSP-urile.
15     </description>
16
17     <!-- Definitii de servlet-uri -->
18     <servlet>
19        <servlet-name>salut1</servlet-name>
20
21        <description>
22           Un exemplu de servlet simplu care trateaza o cerere HTTP get.
23        </description>
24
25        <servlet-class>
26           SalutServletGet
27        </servlet-class>
28     </servlet>
29
30     <!-- Translatii servlet-uri -->
31     <servlet-mapping>
32        <servlet-name>salut1</servlet-name>
33        <url-pattern>/salut1</url-pattern>
34     </servlet-mapping>
35
36  </web-app>
```

Liniile 1–3 specifica **tipul de document al descriptorului de dezvoltare** al aplicatiei Web si **locatia DTD-ului** pentru acest fisier XML.

Elementul **`web-app`** (liniile **5–36**) defineste configuratia fiecarui *servlet* in aplicatia Web si translatia (*servlet mapping*) pentru fiecare *servlet*.

Elementul **`display-name`** (liniile 8–10) specifica un nume care poate fi afisat catre administratorul serverului pe care este instalata aplicatia Web.

Elementul **`description`** (liniile 12–15) specifica o descriere a aplicatiei care poate fi afisata catre administratorul serverului.

Elementul **`servlet`** (liniile **18–28**) descrie un *servlet*.

Elementul **`servlet-name`** (linia 19) este **numele pe care l-am ales pentru *servlet*** (**`salut1`**).

Elementul **`description`** (liniile 21–23) specifica **o descriere pentru un anumit *servlet***. Din nou, aceasta poate fi afisata catre administratorul serverului pe care este instalata aplicatia Web.

Elementul **`servlet-class`** (liniile 25–27) specifica **numele complet calificat** (incluzand calea corespunzatoare pachetului de clase din care face parte) **al *servlet-ului* compilat**. Astfel, *servlet-ul* **`salut1`** este definit de clasa **`SalutServletGet`** (clasa care face parte din pachetul implicit, local).

Elementul **`servlet-mapping`** (liniile **31–34**) specifica translatia *servlet-ului*, prin intermediul elementelor **`servlet-name`** si **`url-pattern`**.

Elementul **`url-pattern`** (linia 33) ajuta serverul sa determine ce cereri sunt trimise catre *servlet* (**`salut1`**). Aplicatia Web va fi instalata ca parte a *context root-ului* **`lab_servlet`**. Astfel, **URL-ul relativ pe care il furnizam *browser-ului* pentru a invoca *servlet-ul* in acest exemplu este**:

```
/lab_servlet/salut1
```

unde **`/lab_servlet`** specifica ***context root-ul* care ajuta serverul sa determine aplicatia Web care trateaza cererea** iar **`/salut1`** specifica **schema URL (*pattern-ul* URL) care este translatata** in *servlet-ul* **`salut1`** **pentru a trata cererea**.

Se observa ca **serverul pe care se afla *servlet-ul* nu este specificat aici**, desi **este posibil sa se faca asta, folosind**:

```
http://localhost:8080/lab_servlet/salut1
```

**Daca nu sunt specificate in mod explicit serverul si numarul de port ca parte a URL-ului**, *browser-ul* presupune ca rutina de tratare a formularului (*servlet-ul* specificat in proprietatea **`action`** a elementului **`form`**) se afla pe acelasi server si la acelasi numar de port de unde *browser-ul* a descarcat (*downloaded*) pagina Web care contine respectivul element **`form`** (formular).

**Exista mai multe formate de scheme URL** care pot fi utilizate. **Schema URL `/salut1` necesita o potrivire exacta a schemei.**
Dar pentru o aplicatie Web se pot specifica si **translatii de structura de directoare** (*path mappings*, translatii de cale), **translatii de extensie** si **un *servlet* implicit**.

O **translatie de cale** incepe cu un **`/`** si se incheie cu un **`/*`**. De exemplu, **schema URL:**

```
/lab_servlet/example/*
```

indica faptul ca orice cale URL incepand cu **`/lab_servlet/example/`** va fi trimisa catre *servlet-ul* care are schema URL respectiva.

O **translatie de extensie** incepe cu un **`*.`** si se incheie cu un nume de extensie. De exemplu, **schema URL:**

```
*.jsp
```

indica faptul ca **orice cerere pentru un fisier cu extensia va fi trimis catre *servlet-ul* care trateaza cereri JSP**.

De fapt, **serverele cu container JSP au o translatie implicita** a extensiei **`.jsp`** catre un *servlet* care trateaza cererile JSP.

**Schema** URL **`/`** reprezinta ***servlet-ul* implicit pentru aplicatia Web**. Ea este **similara documentului implicit al unui server Web** (de exemplu, daca se specifica URL-ul **`www.elcom.pub.ro`** in *browser-ul* Web, documentul primit de la serverul Web aflat la adresa **`www.elcom.pub.ro`** este documentul implicit **`index.html`**).

**Daca schema URL corespunde *servlet-ului* implicit al unei aplicatii Web, acel *servlet* este invocat pentru a se da un raspuns implicit clientului**. Acest mecanism poate fi util personalizarii continutului Web trimis catre utilizatori specifici.

**Acum suntem pregatiti pentru a plasa fisierele noastre in directoarele corespunzatoare pentru a completa dezvoltarea primului *servlet***. Trei fisiere trebuie plasate in directoare potrivite: **`SalutServletGet.html`**, **`SalutServletGet.class`** si **`web.xml`**.

**In directorul `webapps`** al instalarii **`jakarta-tomcat-4.1.30`** **se creaza subdirectorul `lab_servlet`** - *context root-ul* aplicatiei Web.

**In acest director se creaza subdirectoarele `servlets`** si **`WEB-INF`**.

**Se plaseaza fisierele HTML pentru *servlet-uri*** in directorul **`servlets`**. In cazul nostru, **se copiaza fisierul `SalutServletGet.html`** in directorul **`servlets`**.

**In directorul `WEB-INF` se creaza subdirectorul `classes`**.

**Se copiaza fisierul `web.xml`** in directorul **`WEB-INF`**, **si fisierul `SalutServletGet.class`** in directorul **`classes`**.

Astfel, **structura de directoare si fisiere a aplicatiei Web**, aflata in directorul **`webapps`** al instalarii **`jakarta-tomcat-4.1.30`**, **este urmatoarea** (numele fisierelor sunt scrise inclinat):

```
lab_servlet/
   servlets/
      SalutServletGet.html
   WEB-INF/
      web.xml
      classes/
         SalutServletGet.class
```

## 3.4.2. Compilarea *servlet-urilor* Java

**Pentru a compila *servlet-ul* trebuie fie utilizata optiunea `-classpath`, fie adaugata direct variabilei de mediu CLASSPATH urmatoarea cale**:

```
      c:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;c:\jakarta-tomcat-
4.1.30\common\lib\jspengine.jar
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **c:**, si

```
      d:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;d:\jakarta-tomcat-
4.1.30\common\lib\jspengine.jar
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **d:**.

Acest lucru se poate face, **de exemplu, cu comanda**:

```
      > set CLASSPATH=c:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;
      c:\jakarta-tomcat-4.1.30\common\lib\jspengine.jar
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **c:**, si

```
> set CLASSPATH=d:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;
d:\jakarta-tomcat-4.1.30\common\lib\jspengine.jar
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **d:**.

Astfel, **pentru compilarea *servlet-ului* poate fi folosita urmatoarea secventa de comenzi**:

```
> set CLASSPATH=c:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;
c:\jakarta-tomcat-4.1.30\common\lib\jspengine.jar
> javac -d WEB-INF\classes SalutServletGet.java
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **c:**, si

```
> set CLASSPATH=d:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;
d:\jakarta-tomcat-4.1.30\common\lib\jspengine.jar
> javac -d WEB-INF\classes SalutServletGet.java
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **d:**.

**Alternativa** este:

```
> javac -d WEB-INF\classes -classpath
c:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;c:\jakarta-tomcat-
4.1.30\common\lib\jspengine.jar SalutServletGet.java
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **c:**, si

```
> javac -d WEB-INF\classes -classpath
d:\jakarta-tomcat-4.1.30\common\lib\servlet.jar;d:\jakarta-tomcat-
4.1.30\common\lib\jspengine.jar SalutServletGet.java
```

**daca instalarea `jakarta-tomcat-4.1.30`** se afla pe *hard disk-ul* **d:**.

### 3.4.3. Invocarea *servlet-urilor* Java

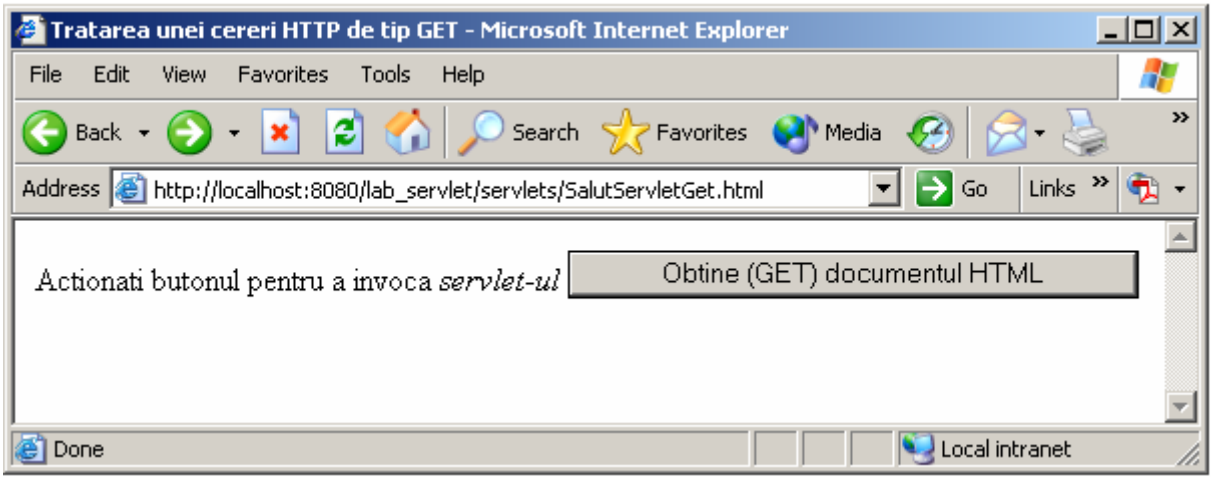
**Dupa plasarea fisierelor** in directoarele potrivite, **se porneste serverul Web Tomcat**, **se deschide *browser-ul*** Web si **se acceseaza urmatoarea adresa Web** (URL):

**`http://localhost:8080/lab_servlet/servlets/SalutServletGet.html`**

**pentru a se incarca pagina** Web **`SalutServletGet.html`** in *browser-ul* Web.

```
 1  <?xml version = "1.0"?>
 2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 4
 5  <!-- SalutServletGet.html -->
 6
 7  <html xmlns = "http://www.w3.org/1999/xhtml">
 8  <head>
 9     <title>Tratarea unei cereri HTTP de tip GET </title>
10  </head>
11
12  <body>
13     <form action = "/lab_servlet/salut1" method = "get">
14
15        <p><label>Actionati butonul pentru a invoca <i>servlet-ul</i>
16           <input type = "submit" value = "Obtine (GET) documentul HTML" />
17        </label></p>
18
19     </form>
20  </body>
21  </html>
```

**Se obtine in browser**:




Apoi **<u>se actioneaza</u>** butonul **Obtine (GET) documentul HTML** pentru a **<u>se invoca *servlet-ul*</u>**.

In *browser* **se obtine urmatoarea pagina HTML (generata dinamic de catre *servlet*)**:


Invocarea acestui *servlet* poate fi incercata **din mai multe *browser-e* Web** pentru a observa faptul ca **rezultatul este acelasi indiferent de *browser-ul* folosit**.

De fapt, **fisierul HTML nu e neaparat necesar pentru a invoca acest *servlet***.

**<u>O cerere GET poate fi trimisa catre server direct prin utilizarea URL-ului folosit pentru a cere pagina Web</u>**. In acest caz, se editeaza in campul **Address** (**Location**) al *browser-ului*, urmatorul URL:

`http://localhost:8080/lab_servlet/salut1`

obtinandu-se: