

Note pratiche sullo sviluppo di servlet (I)

- Nel caso in cui sulla macchina locale (PC in laboratorio/PC a casa/Portatile) ci sia a disposizione un ambiente Java (con compilatore) e un editor/ambiente di sviluppo a cui si è abituati, è più comodo realizzare e testare le servlet in locale e poi trasferirle sul server finale (e.g., **latoserver.dimi.uniud.it**).
- Tuttavia è possibile lavorare direttamente su **latoserver.dimi.uniud.it** collegandosi tramite **ssh** (PuTTY sui sistemi Windows in laboratorio) e creando i vari sorgenti con gli editor **vi** o **pico**.

Note pratiche sullo sviluppo di servlet (II)

- Se si decide di sviluppare le servlet in locale esistono vari programmi che permettono di trasferire i file sulla macchina remota in modo sicuro:
 - scp;
 - CoreFTP (www.coreftp.com);
 - il programma PSFTP fornito assieme a PuTTY sui sistemi Windows in laboratorio.

Esempio: utilizzo di PSFTP

- Lanciare il programma (da menu, oppure direttamente dalla cartella di installazione di PuTTY:
C:\Programmi\PuTTY\PSFTP.EXE).
- A questo punto si hanno a disposizione i comandi:
 - **lcd <percorso>** (per spostarsi nel filesystem locale)
 - **!dir** (per visualizzare il contenuto della directory locale corrente)
 - **open <nome utente>@host** (per collegarsi come <nome utente> al server **host**)
 - **cd <percorso>** (per spostarsi nel filesystem remoto)
 - **dir** (per visualizzare il contenuto della directory remota corrente)
 - **put <nomefile>** per copiare sul server il file **<nomefile>**
 - **get <nomefile>** per copiare sulla macchina locale il file **<nomefile>**

Gli argomenti dei metodi `doGet/doPost`

- Osserviamo che le servlet sono dei veri e propri programmi Java che vengono invocati dal contenitore di servlet (e.g., Tomcat) per soddisfare le richieste dei client.
- L'interfaccia con il protocollo HTTP messa a disposizione del programmatore consiste nell'utilizzo dei metodi dei due argomenti di tipo **`HttpServletRequest`** e **`HttpServletResponse`**.

HttpServletRequest

- E' un'interfaccia, definita in **javax.servlet.http**, che aggiunge alla “soprinterfaccia” **ServletRequest** (per servlet generiche), definita in **javax.servlet**, metodi specifici per le richieste HTTP.
- Rappresenta la richiesta di un client verso una servlet.
- L'oggetto corrispondente viene creato dal container (e.g., Tomcat) al momento della richiesta e passato al metodo opportuno della servlet.
- Metodi fondamentali:
 - **getInputStream**: per leggere informazioni inviate dal client;
 - **getParameter**: per estrarre i parametri della richiesta.

HttpServletResponse

- E' un'interfaccia, definita in **javax.servlet.http**, che aggiunge alla “soprinterfaccia” **ServletResponse** (per servlet generiche), definita in **javax.servlet**, metodi specifici per le richieste HTTP.
- Rappresenta la risposta al client da parte di una servlet.
- L'oggetto corrispondente viene creato dal container (e.g., Tomcat) al momento della richiesta e passato al metodo opportuno della servlet.
- Metodi fondamentali:
 - **setContentType**: per specificare il tipo MIME del contenuto che verrà spedito al client;
 - **getWriter**: restituisce il flusso di dati verso il client.

Errori comuni

- Tomcat manager:
“**FAIL - Application at context path /test could not be started**”
Causa: errore di parsing - controllare il file **web.xml**.
- “**HTTP Status 404**”
Causa: servlet non trovata – controllare l’URL della servlet ed i link ad essa relativi – controllare che il tag **<url-pattern>** abbia un “/” iniziale (a differenza degli URL relativi che si usano nelle pagine HTML per richiamare le servlet).
- “**HTTP Status 500**”
Errore interno (eccezione) della servlet.

Un contatore di accessi

- Scriviamo una servlet che conta il numero di accessi e lo visualizza.
- Sfruttiamo il fatto che Tomcat, quando riceve la richiesta di una servlet, crea un'istanza della classe corrispondente.
- Utilizziamo una variabile d'istanza privata **i** nel modo seguente:
 - la azzeriamo all'inizio dell'esecuzione della servlet;
 - la incrementiamo ad ogni accesso.

Il sorgente Java (I)

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

public class Contatore extends HttpServlet {
    private int i;

    // init viene eseguito solo alla prima richiesta
    // della servlet.
    public void init() {
        i=0;
    }

    // continua nel lucido successivo ...
}
```

Il sorgente Java (II)

```
// ... continua dal lucido precedente
public void doGet(HttpServletRequest req,
                  HttpServletResponse res)
                  throws ServletException , IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Contatore di "+
                "accessi</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<STRONG>Accessi registrati "+
                "finora: "+(++i)+"</STRONG>");
    out.println("</BODY></HTML>");
}
}
```

```
}
```

Il metodo `init()`

- Il metodo `init()` di una servlet viene eseguito alla prima richiesta di quest'ultima.
- E' utile quindi per inserire il codice che inizializza la servlet.
- Nel nostro caso il metodo `init()` contiene soltanto l'assegnamento `i = 0` (che inizializza a zero il contatore di accessi).

Modifiche al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/Contatore">Un contatore di
          accessi</A>
      </TD>
    </TR>
  </TABLE>
</BODY>

</HTML>
```

Modifiche al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE web-app
```

```
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
    "http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app>
```

```
    ...
```

```
    <servlet>
```

```
        <servlet-name>Contatore</servlet-name>
```

```
        <servlet-class>Contatore</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

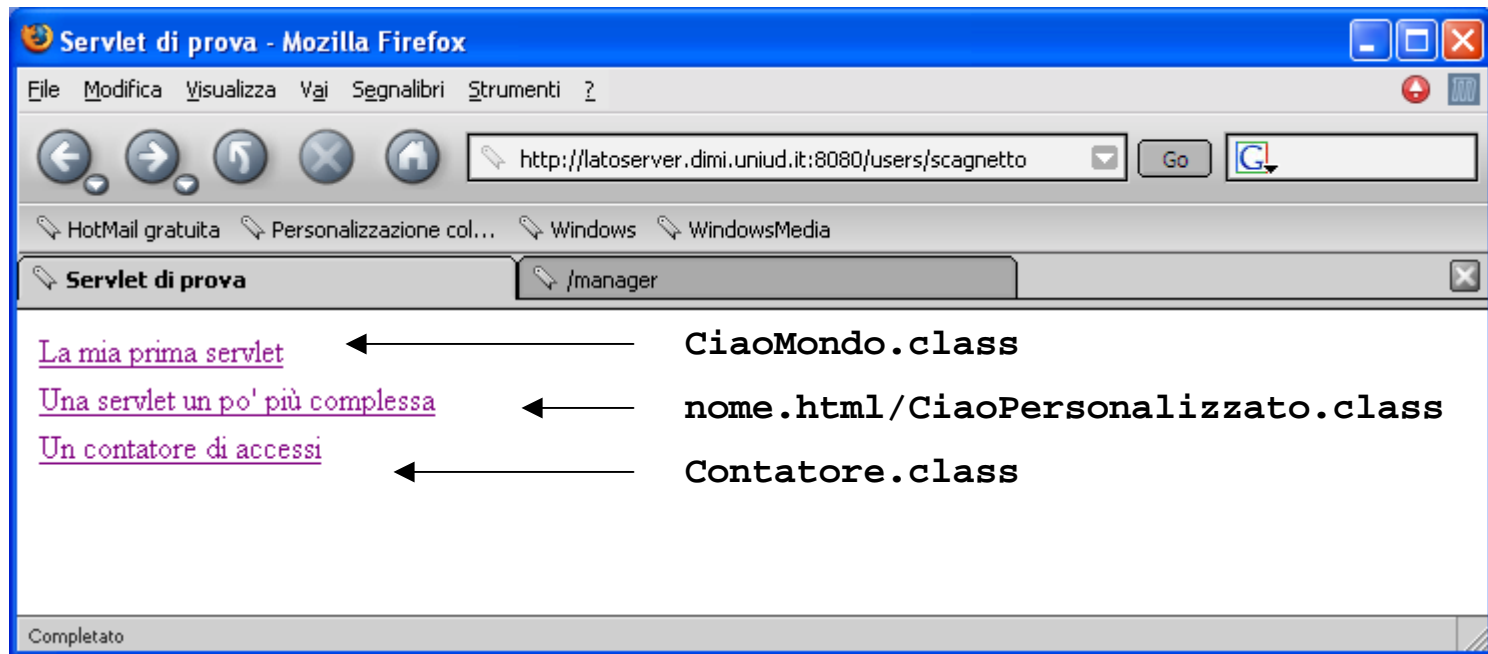
```
        <servlet-name>Contatore</servlet-name>
```

```
        <url-pattern>/servlet/Contatore</url-pattern>
```

```
    </servlet-mapping>
```

```
</web-app>
```

Aspetto della pagina `index.html`



Ciclo di vita di una Servlet

- Non caricata.
- Prima richiesta:
 - Caricata: il file **.class** viene caricato in memoria centrale.
 - Inizializzata: viene eseguito il metodo **init()**.
- In servizio: risponde alle richieste con il metodo opportuno (per le servlet http vengono richiamati i metodi **do<tipo richiesta>**, e.g., **doGet()** e **doPost()**).
- In attesa: rimane presente in memoria centrale, ma inattiva.
- In distruzione: viene eseguito il metodo **destroy()** e la servlet viene rimossa dalla memoria centrale.

Un contatore di accessi “persistente”

- Modifichiamo la servlet che conta il numero di accessi in modo che conservi tale valore anche quando l'applicazione web viene riavviata.
- Sfruttiamo i metodi **init()** e **destroy()**:
 - in **init()** leggiamo dal file il valore iniziale del contatore **i**;
 - in **destroy()** (al momento della rimozione dalla memoria della servlet) scriviamo sul file il valore corrente del contatore **i**.

Codice Java (I)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ContatorePersistente extends HttpServlet {
    final static String NOME_FILE="contatore.txt";
    private int i;
    private String path; // memorizzerà il percorso di NOME_FILE.
    public void init() {
        try {
            path=getServletContext().getRealPath("")+"/";
            DataInputStream f=new
                DataInputStream(new FileInputStream(path+NOME_FILE));
            i=f.readInt();
            f.close();
        } catch (IOException e) { i=0; }
    } // continua nel lucido successivo ...
}
```

Codice Java (II)

```
// ... continua dal lucido precedente
public void doGet(HttpServletRequest req,
                  HttpServletResponse res)
                  throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Contatore di "+
                "accessi</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<STRONG>Accessi registrati "+
                "finora: "+(++i)+"</STRONG>");
    out.println("</BODY></HTML>");
}
// continua nel lucido successivo ...
```

Codice Java (III)

```
// ... continua dal lucido precedente
public void destroy() {
    try {
        DataOutputStream f=
        new DataOutputStream
            (new FileOutputStream(path+NOME_FILE));
        f.writeInt(i);
        f.close();
    } catch (IOException e) { }
}
```

Il metodo **destroy()** viene invocato al momento della rimozione della servlet dalla memoria centrale.

Modifiche al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE web-app
```

```
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
    "http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app>
```

```
    ...
```

```
    <servlet>
```

```
        <servlet-name>Contatore2</servlet-name>
```

```
        <servlet-class>ContatorePersistente</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

```
        <servlet-name>Contatore2</servlet-name>
```

```
        <url-pattern>/servlet/ConPers</url-pattern>
```

```
    </servlet-mapping>
```

```
</web-app>
```

Modifiche al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/ConPers">Un contatore di accessi
          persistente</A>
      </TD>
    </TR>
  </TABLE>
</BODY>

</HTML>
```

Generare una tabella HTML a partire da un file di testo

- In questo esempio faremo generare alla servlet **StampaTabella** una tabella HTML 10x2 a partire dal contenuto di un file di testo.
- Il formato dei dati nel file di testo è il cosiddetto CSV (i.e., Comma Separated Value: valori separati da virgole).
- I dati della tabella saranno contenuti in un file di testo chiamato **tabella.txt** nella cartella **servlets** della propria home directory.

Il codice Java (I)

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

public class StampaTabella extends HttpServlet {

    public void doGet(HttpServletRequest req,
                      HttpServletResponse res)
                      throws ServletException , IOException,
FileNotFoundException {
    String filePath=getServletContext().getRealPath("")+"/tabella.txt";
    BufferedReader input = new BufferedReader(new FileReader(filePath));
    String linea = null;
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Tabella letta da file</TITLE></HEAD>");
```

Il codice Java (II)

```
out.println("<BODY>");
    out.println("<TABLE BORDER=\"1\">");

    while((linea = input.readLine()) != null) {
        out.println("<TR>");
        String[] celle = linea.split(",");
        out.println("<TD>"+celle[0]+"</TD>");
        out.println("<TD>"+celle[1]+"</TD>");
        out.println("</TR>");
    }

    input.close();
    out.println("</TABLE>");
    out.println("</BODY></HTML>");
}
}
```


Modifiche al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/Tabella">Stampa di una tabella
          letta da file</A>
      </TD>
    </TR>
  </TABLE>
</BODY>

</HTML>
```

Modifiche al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE web-app
```

```
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
    "http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app>
```

```
    ...
```

```
    <servlet>
```

```
        <servlet-name>Tabella</servlet-name>
```

```
        <servlet-class>StampaTabella</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

```
        <servlet-name>Tabella</servlet-name>
```

```
        <url-pattern>/servlet/Tabella</url-pattern>
```

```
    </servlet-mapping>
```

```
</web-app>
```

Note sul dispiegamento

- Ricordarsi di copiare nella cartella **servlets** della propria home directory su **latoserver.dimi.uniud.it** anche il file **tabella.txt**.
- Esempio di contenuto del file **tabella.txt**:

Riga 1 Colonna 1,Riga 1 Colonna 2

Riga 2 Colonna 1,Riga 2 Colonna 2

Riga 3 Colonna 1,Riga 3 Colonna 2

Riga 4 Colonna 1,Riga 4 Colonna 2

Riga 5 Colonna 1,Riga 5 Colonna 2

Riga 6 Colonna 1,Riga 6 Colonna 2

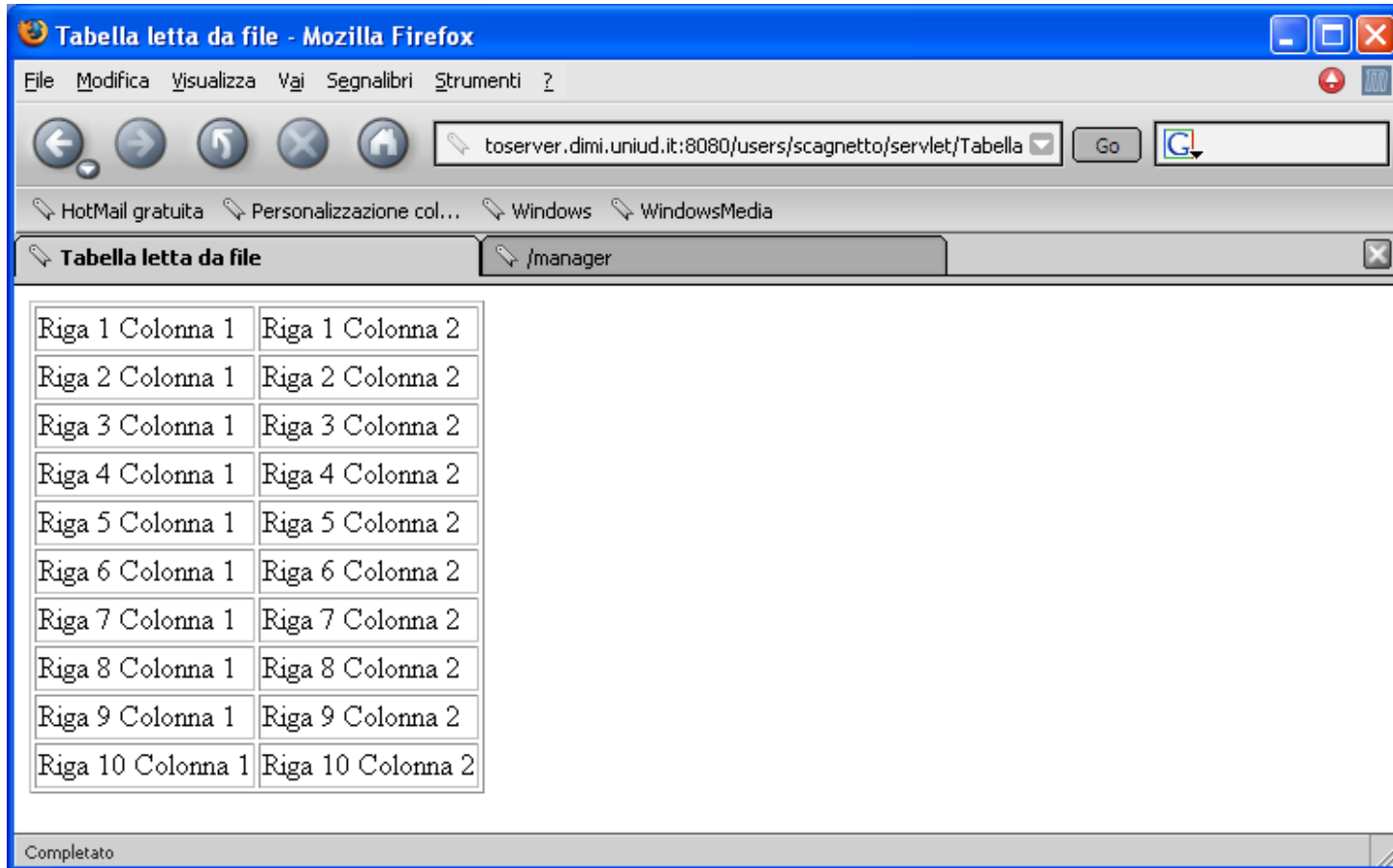
Riga 7 Colonna 1,Riga 7 Colonna 2

Riga 8 Colonna 1,Riga 8 Colonna 2

Riga 9 Colonna 1,Riga 9 Colonna 2

Riga 10 Colonna 1,Riga 10 Colonna 2

Esempio di esecuzione di StampaTabella



The screenshot shows a Mozilla Firefox browser window titled "Tabella letta da file - Mozilla Firefox". The address bar contains the URL "tosever.dimi.uniud.it:8080/users/scagnetto/servlet/Tabella". The browser displays a table with 10 rows and 2 columns. The status bar at the bottom indicates "Completato".

Riga 1 Colonna 1	Riga 1 Colonna 2
Riga 2 Colonna 1	Riga 2 Colonna 2
Riga 3 Colonna 1	Riga 3 Colonna 2
Riga 4 Colonna 1	Riga 4 Colonna 2
Riga 5 Colonna 1	Riga 5 Colonna 2
Riga 6 Colonna 1	Riga 6 Colonna 2
Riga 7 Colonna 1	Riga 7 Colonna 2
Riga 8 Colonna 1	Riga 8 Colonna 2
Riga 9 Colonna 1	Riga 9 Colonna 2
Riga 10 Colonna 1	Riga 10 Colonna 2

Esercizio

- Generalizzare la servlet che disegna la tabella in base al contenuto del file di testo **tabella.txt** in modo da funzionare con un numero arbitrario di colonne.