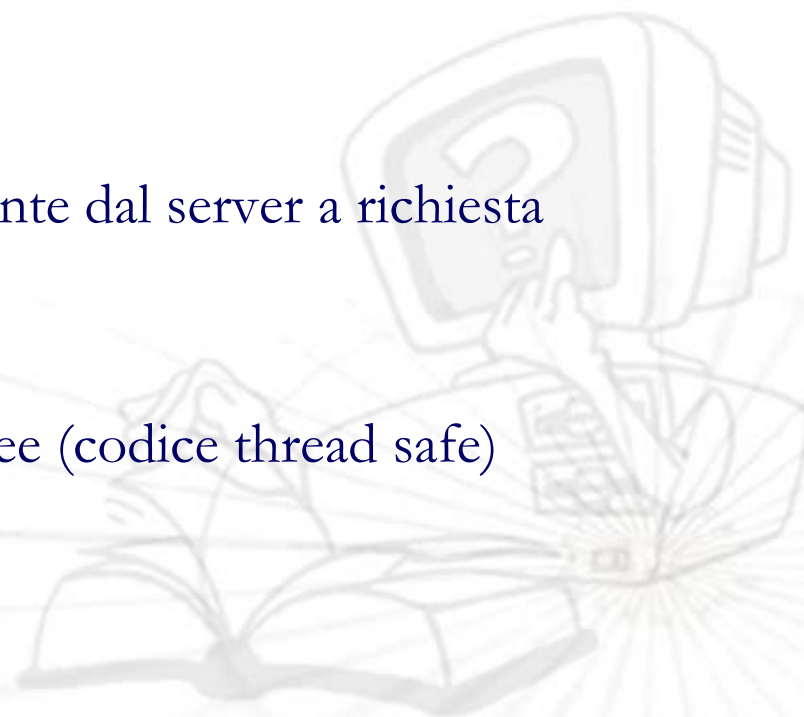


SERVLET

Introduzione alla programmazione di Servlet

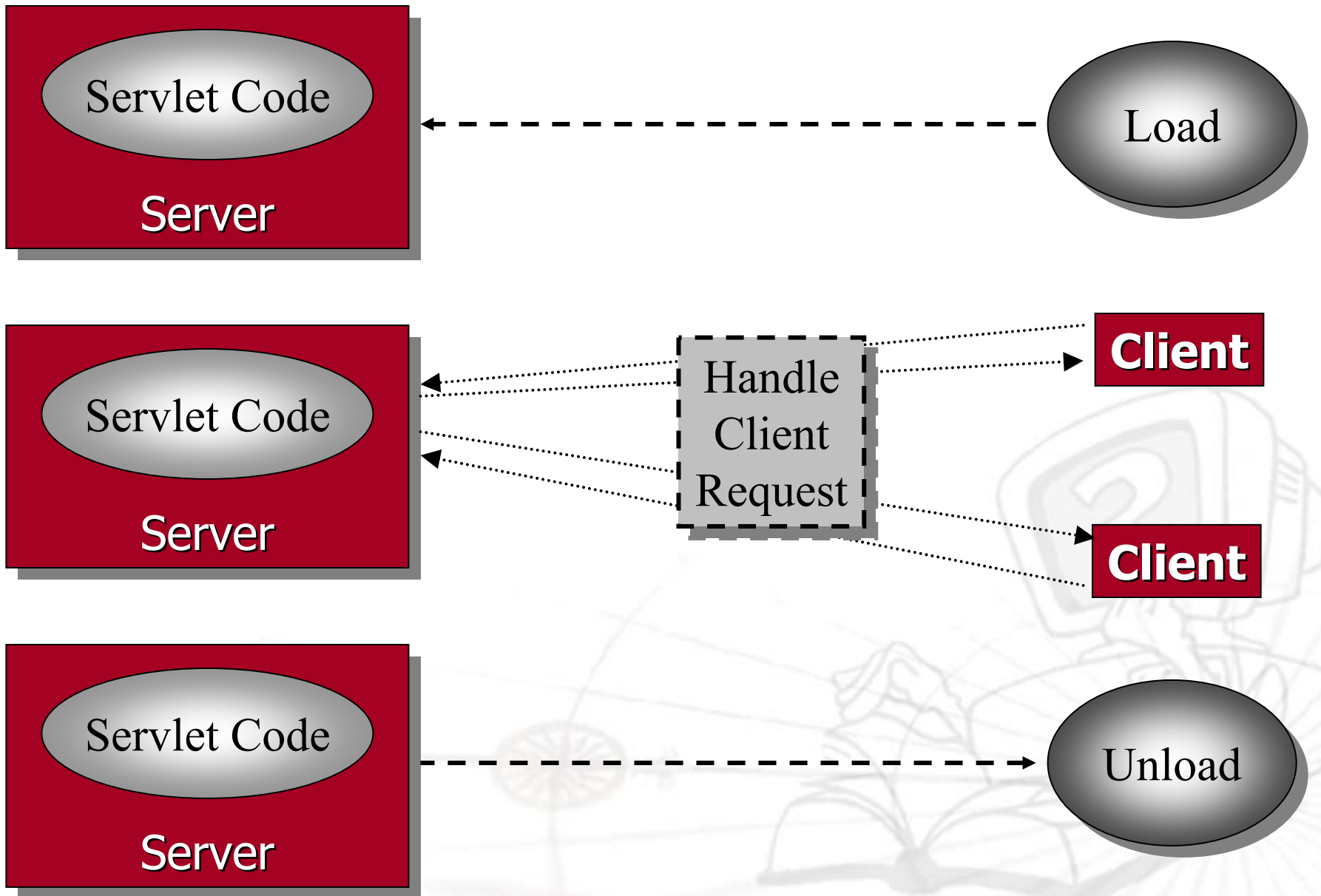


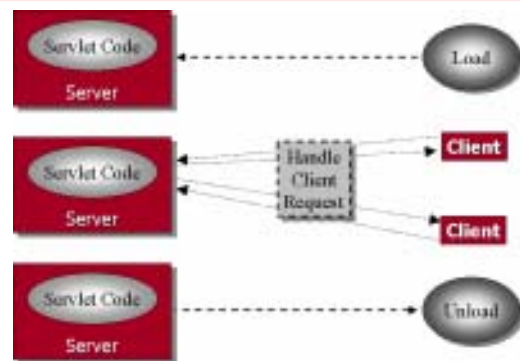
- Estensioni del server scritte in Java utilizzando la Servlet API
- La Servlet API è diventata un'estensione standard di Java, dalla versione 1.2
- Si tratta di moduli java caricati dinamicamente dal server a richiesta
- Possono gestire richieste multiple simultanee (codice thread safe)



- Supportati da una moltitudine di Server Web:
 - Apache (via jserv e/o tomcat)
 - Netscape FastTrack 2.0, Enterprise 2.0, 3.0
 - Microsoft IIS 2.0, IIS 3.0
 - Lotus Domino Go Webserver
 - IBM Internet Connection Server
- Spesso indirizzati tramite il prefisso servlet nella URL
`http://hostname/servlet/Servlet.class[?argomenti]`
- Il prefisso è configurabile nel Server e può essere più di uno

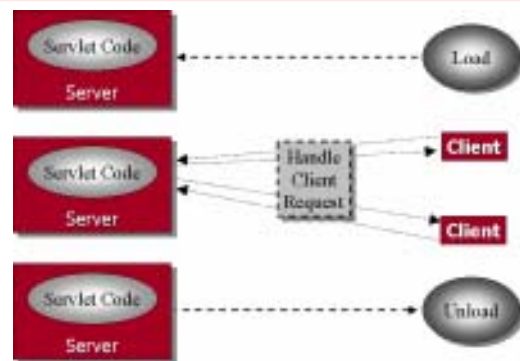






- Il Server carica il servlet
- Manda in esecuzione il metodo *init*
 - viene eseguito un'unica volta (può non essere thread-safe)
 - durante la init non sono gestite richieste

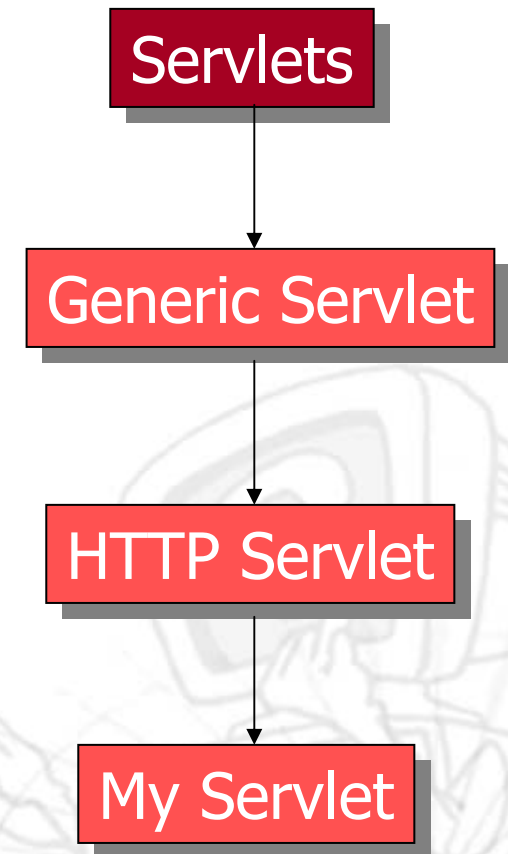
- Il Servlet gestisce richieste tramite il metodo *service*
 - gestisce la richiesta e produce la risposta
 - deve essere *thread-safe*; in generale verrà chiamato più volte in concorrenza
 - per non gestire la concorrenza tra thread, il servlet deve implementare l'interfaccia SingleThreadModel



• I Servlet accettano richieste finché non sono rimossi dal servizio. Quando questo accade:

- viene chiamato il metodo destroy
- non può andare in concorrenza con altri destroy
- può andare in concorrenza con altre richieste di service

- Definita nel package `javax.servlet.http`
- Nessuna assunzione su:
 - come un servlet sia caricato dal server
 - ambiente in cui il server gira (server internals)
 - il protocollo usato per la trasmissione
- Conseguenza: estrema portabilità
- La principale astrazione è l'interface Servlet
- Tutti i servlet implementano questa interfaccia direttamente o estendendo `HttpServlet`
- L'implementazione di un servlet avviene ridefinendo i metodi dell'interfaccia Servlet



La maggior parte dei metodi riceve due oggetti: una `ServletRequest` e una `ServletResponse`

- la classe ***ServletRequest*** incapsula le comunicazioni dal client al server
- la classe ***ServletResponse*** incapsula le comunicazioni dal server al client



Consente al Servlet di accedere a:

- nomi/valori passati dal client
 - il tipo di protocollo (solitamente http)
 - l'indirizzo del client
 - l'input stream
-
- `HttpServletRequest` è una sottoclasse di `ServletRequest` con metodi specifici per HTTP



Consente al Servlet di:

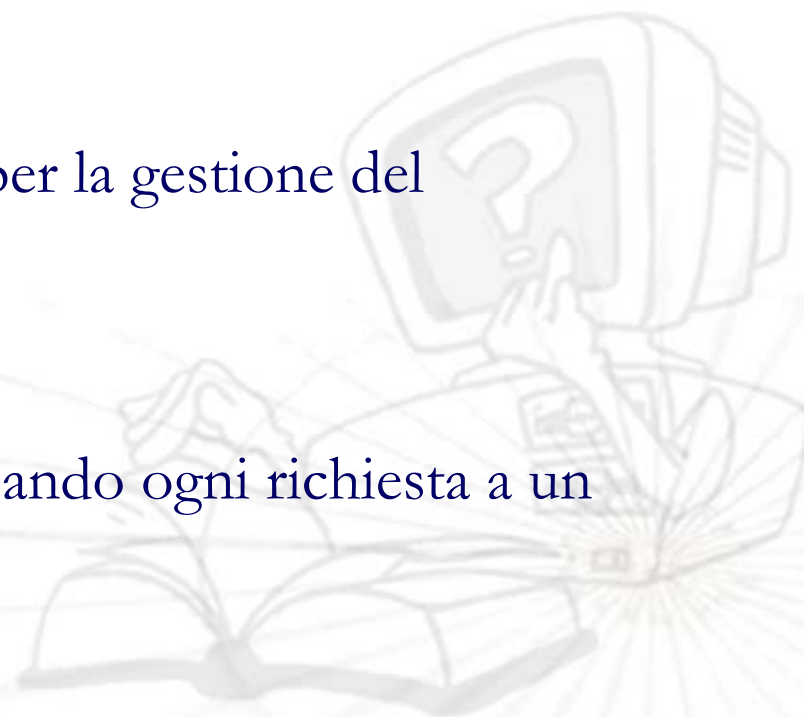
- settare content length della risposta
- settare mime type della risposta
- accedere all'output stream ServletOutputStream
- HttpServletResponse è una sottoclasse di ServletResponse con metodi specifici per HTTP



- I servlet HTTP hanno metodi e oggetti aggiuntivi per gestire funzionalità di session-tracking.
- Queste API servono per mantenere lo stato tra servlet e client attraverso connessioni diverse.



- Può essere definito implementando la interfaccia `javax.servlet.Servlet`
- In generale estende la classe `javax.servlet.http.HttpServlet`
- `HttpServlet` fornisce un framework per la gestione del protocollo HTTP
- Gestisce le richieste HTTP/1.1, passando ogni richiesta a un metodo specifico



- Per default una estensione di HttpServlet deve avere metodi thread-safe

- Per gestire una richiesta alla volta:

```
public class SurveyServlet extends HttpServlet
```

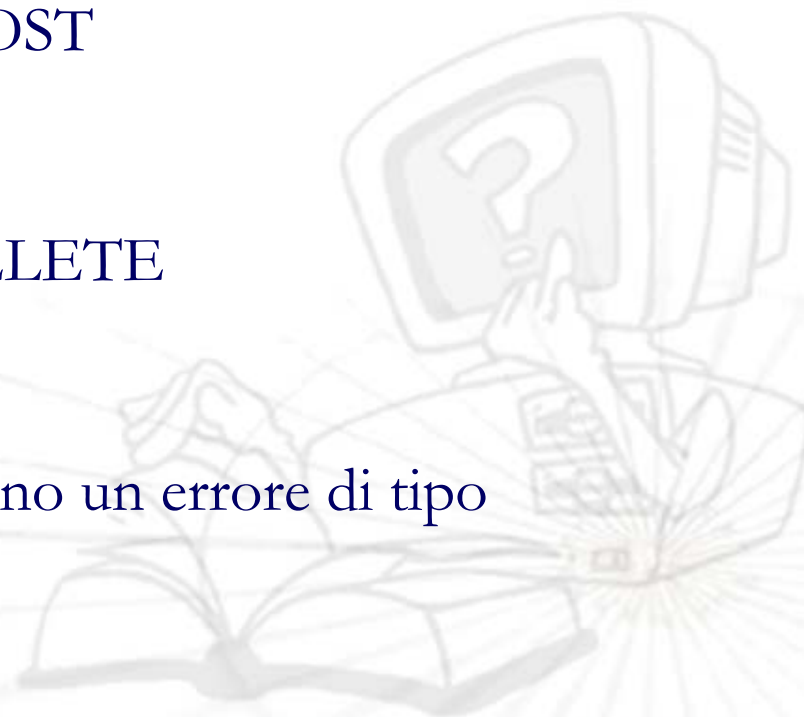
```
implements SingleThreadModel {
```

```
// SingleThreadModel non richiede di definire alcun metodo
```

```
}
```



- Un Servlet derivato da HttpServlet, deve ridefinire i metodi http che intende gestire:
 - doGet, per gestire i metodi HTTP GET ed HEAD
 - doPost, per gestire le richieste di POST
 - doPut, per gestire le richieste PUT
 - doDelete per gestire le richieste DELETE
- Per default questi metodi restituiscono un errore di tipo BAD_REQUEST (400)



- Per ogni metodo HTTP:
 - `getParameterNames` accede alla lista dei nomi
 - `getParameterValues` accede ai parametri per nome
 - `getQueryString` consente il parsing manuale della QUERY_STRING (solo per HTTP GET)
- Per i metodi POST, PUT e DELETE di HTTP:
 - per dati testuali: `BufferedReader` restituito da `getReader`
 - per dati binari: `ServletInputStream` restituito da `getInputStream`



```
Enumeration paramNames = request.getParameterNames();

while(paramNames.hasMoreElements()) {

    String paramName = (String)paramNames.nextElement();
    out.println("..." + paramName + "\n<TD>");

String[] paramValues = request.getParameterValues(paramName);

    if (paramValues.length == 1) {

        String paramValue = paramValues[0];
        if (paramValue.length() == 0)
            out.print("<I>No Value</I>");
        else
            out.print(paramValue);

    } else {

        out.println("<UL>");
        for (int i=0; i<paramValues.length; i++) {
            out.println("<LI>" + paramValues[i]);
        }
        out.println("</UL>");

    }

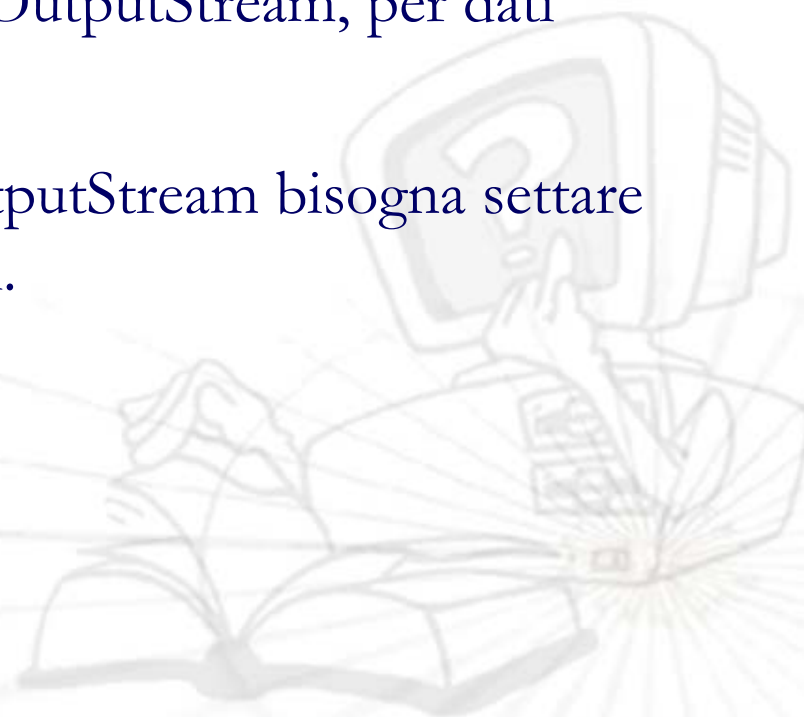
}
```



```
Enumeration headerNames = request.getHeaderNames();  
while(headerNames.hasMoreElements()) {  
    String headerName = (String)headerNames.nextElement();  
    out.println("<TR><TD>" + headerName);  
    out.println(" <TD>" + request.getHeader(headerName));  
}
```

La risposta può essere restituita usando:

- il `Writer` restituito dalla `getWriter`, per dati testuali;
- l'`OutputStream` restituito dalla `getOutputStream`, per dati binari;
- Prima di accedere a `Writer` o a `OutputStream` bisogna settare l'HTTP header relativo a tipo di dati.

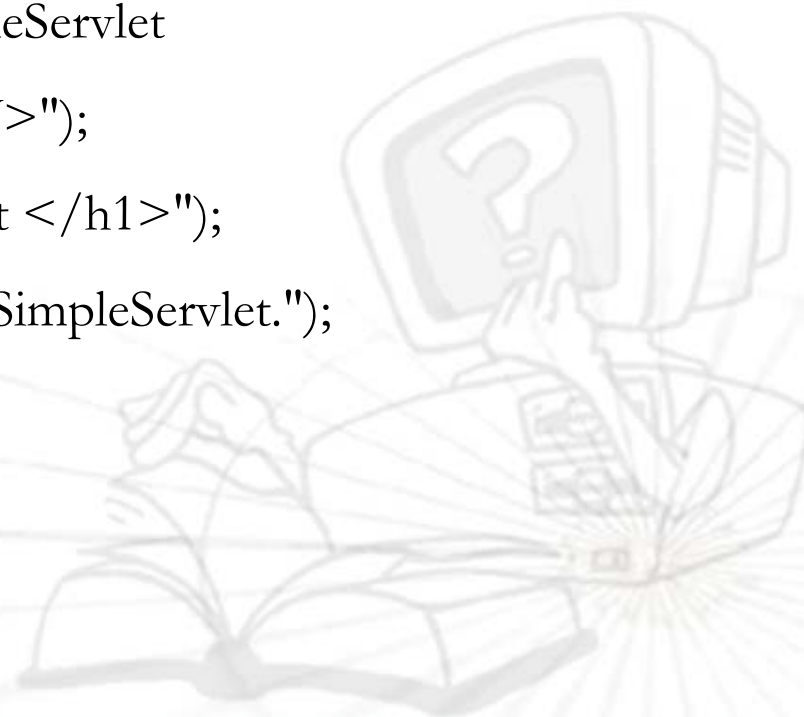


Esempio di Accesso alle Variabili CGI

```
String[][] variables = {  
    { "AUTH_TYPE", request.getAuthType() },  
    { "CONTENT_LENGTH", String.valueOf(request.getContentLength()) },  
    { "CONTENT_TYPE", request.getContentType() },  
    { "DOCUMENT_ROOT", getServletContext().getRealPath("/") },  
    { "PATH_INFO", request.getPathInfo() },  
    { "PATH_TRANSLATED", request.getPathTranslated() },  
    { "QUERY_STRING", request.getQueryString() },  
    { "REMOTE_ADDR", request.getRemoteAddr() },  
    { "REMOTE_HOST", request.getRemoteHost() },  
    { "REMOTE_USER", request.getRemoteUser() },  
    { "REQUEST_METHOD", request.getMethod() },  
    { "SCRIPT_NAME", request.getServletPath() },  
    { "SERVER_NAME", request.getServerName() },  
    { "SERVER_PORT", String.valueOf(request.getServerPort()) },  
    { "SERVER_PROTOCOL", request.getProtocol() },  
    { "SERVER_SOFTWARE", getServletContext().getServerInfo() } };
```



```
public class SimpleServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req,  
        HttpServletResponse res) throws ServletException, IOException {  
        res.setContentType("text/html");  
        PrintWriter out = res.getWriter();  
        out.println("<HEAD><TITLE> SimpleServlet  
Output</TITLE></HEAD><BODY>");  
        out.println("<h1> SimpleServlet Output </h1>");  
        out.println("<P>This is output is from SimpleServlet.");  
        out.println("</BODY>");  
        out.close();  
    }  
}
```



Il Servlet scrive i dati del form in un file e risponde all'utente con un messaggio:

```
<html> <head><title>JdcSurvey</title></head> <body>  
<form action=http://localhost/servlet/survey method=POST>  
    <input type=hidden name=survey value=Survey01Results>  
    <BR><BR>How Many Employees in your Company?<BR>  
    <BR>1-100<input type=radio name=employee value=1-100>  
    <BR>100-200<input type=radio name=employee value=100-200>  
    <BR>200-300<input type=radio name=employee value=200-300>  
    <BR>300-400<input type=radio name=employee value=300-400>  
    <BR>500-more<input type=radio name=employee value=500-more>  
    <BR><BR>General Comments?<BR>  
    <BR><input type=text name=comment>  
    <BR><BR>What IDEs do you use?<BR>  
    <BR>JavaWorkShop<input type=checkbox name=ide value=JavaWorkShop>  
    <BR>J++<input type=checkbox name=ide value=J++>  
    <BR>Cafe'<input type=checkbox name=ide value=Cafe'>  
    <BR><BR><input type=submit><input type=reset>  
</form> </body> </html>
```



```
public void doPost(HttpServletRequest req, HttpServletResponse res)

    res.setContentType("text/html");

    PrintWriter toClient = res.getWriter();

    String surveyName = req.getParameterValues("survey")[0];

    FileWriter resultsFile = new FileWriter(resultsDir + System.getProperty("file.separator") + surveyName + ".txt", true);

    PrintWriter toFile = new PrintWriter(resultsFile);

    toFile.println("<BEGIN>");

    Enumeration values = req.getParameterNames();

    while(values.hasMoreElements()) {

        String name = (String)values.nextElement();

        String value = req.getParameterValues(name)[0];

        if(name.compareTo("submit") != 0) {

            toFile.println(name + ": " + value);

        }

        toFile.println("<END>");

        resultsFile.close();

        toClient.println("<html>");

        toClient.println("<title>Thank you!</title>");

        toClient.println("Thank you for participating");

        toClient.println("</html>");

        toClient.close();
```



La gestione delle sessioni avviene tramite:

- Cookies
- URL Rewriting
- Hidden form fields:

`<INPUT TYPE="HIDDEN" NAME="session" VALUE="...">`

- La Servlet API contiene dei metodi che utilizzano Cookies e URL Rewriting

```
HttpSession session = request.getSession(true);

Integer accessCount = new Integer(0);

if (session.isNew()) {
    heading = "Welcome, Newcomer";
} else {
    heading = "Welcome Back";

    Integer oldAccessCount = (Integer)session.getValue("accessCount");
    if (oldAccessCount != null) {
        accessCount = new Integer(oldAccessCount.intValue() + 1);
    }
}

session.putValue("accessCount", accessCount);

out.println("<HTML>..." + "...<H2>Information on Your Session:</H2>\n" + session.getId() + "\n"
+ accessCount + "\n" + new Date(session.getCreationTime()) + "\n" + new
Date(session.getLastAccessedTime()) ....
```

