Servlet e JDBC

Programmazione in Rete e Laboratorio

Matteo Baldoni
Dipartimento di Informatica
Universita` degli Studi di Torino
C.so Svizzera, 185 I-10149 Torino

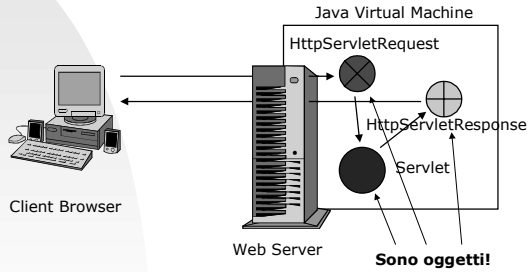
baldoni@di.unito.it
http://www.di.unito.it/~baldoni/didattica

Servlet e Web Server

Servlet:

- estensioni del Java API
- permettono di scrivere applicazioni indipendenti eseguite all'interno del Web server
- possono servire richieste in parallelo
- mantengono la sessione di lavoro con il singolo utente
- efficienti e semplici da utilizzare rispetto ai programmi CGI

Servlet e Web Server



Servlet e Web Server

- Una servlet è una classe Java (estensione di HttpServlet).
- Un Web Server può ospitare più servlet (con relativi alias)
- Quando il processo Web Server viene lanciato, genera un thread Java che inizializza l'oggetto Servlet (metodo init), questo *persiste* per tutta la durata per processo Web Server
- Ogni servlet è un thread all'interno del Web Server (vs CGI dove viene eseguito un processo esterno)

Servlet

- Ogni volta che il Web Server riceve una richiesta per una servlet S:
  - identifica la servlet S
  - invoca il metodo service dell'oggetto Servlet S, per far eseguire le operazioni richieste
- Il metodo service rappresenta l'interfaccia del secondo livello di applicazione. Al termine del metodo, S rimanda una risposta sotto forma di pagina HTML
- Il Web server spedisce la pagina al browser chiamante

Richieste

ES:
http://nbbaldoni.di.unito.it/servlets/splppp0001.Registration?cognome1=Baldoni&nome1=Matteo

- La classe HttpServletRequest offre i metodi per
  - determinare i parametri di una richiesta (cognome1, nome1)
  - estrarre i valori speficati per i parametri nella richiesta (Baldoni, Matteo)
- metodi gestiti: POST (doPost), GET (doGet)

Servlet e Web Server

- Apache, Netscape, Microsoft o Oracle Web Server
- le servlet erano inizialmente solo integrate nel Java Web Server della Sun
- JSP: *Java Server Pages*, analogo di ASP (Active Server Pages), progetto Tomcat
- Modulo ApacheJServ

7

Un esempio: AccessCounter.java

- Si vuole realizzare un contatore di accessi
- Nella realizzazione si deve tenere conto che piu` utenti potrebbero accedere contemporaneamente alla stessa pagina

8

AccessCounter.java

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

public class AccessCounter extends HttpServlet {

  int counter = 1;

  public void service(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {

      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
  […]

}
```

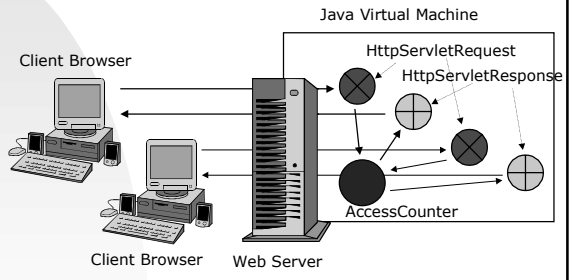
9

AccessCounter.java

```
[…]
      out.println("<HEAD><TITLE>" +
                  "Contatore degli accessi a questa pagina" +
                  "</TITLE></HEAD>");
      out.println("<BODY>");

      out.println("<H2>Contatore accessi a questa pagina</H2>");

      int localCounter;
      synchronized(this) {
          localCounter = counter++;
      }

      out.println("<P>Questa pagina e` stata visitata " +
                              localCounter + " volte.");

      out.println("</BODY>");
[…]
```

10

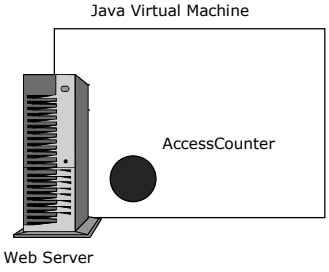
AccessCounter.java



11

AccessCounter.java

- Anche quando non vi sono accessi al server l'oggetto (la servlet) di tipo AccessCounter permane nella Java Virtual Machine del server stesso.



12

Un esempio: Registration.java

- Analizzare le informazioni dei componenti di un gruppo inserite tramite una form ed inviate mediante il metodo POST
- Se non sono rilevati errori spedire una mail con il numero del gruppo (assegnato in modo automatico) e spedire le varie infomazioni raccolte tramite mail
- Se sono rilevati errori nell'immissione dei dati nella form segnalarli tramite una pagina html

registration.html

```
<FORM ACTION="https://baldo.di.unito.it:8080/registration/servlets/Registration"
      METHOD="post">
[…]
<INPUT TYPE="text" NAME=cognome1 VALUE="" SIZE=40>
[…]
<INPUT TYPE="text" NAME=nome1 VALUE="" SIZE=40>
[…]
<INPUT TYPE="text" NAME=email1 VALUE="" SIZE=40>
[…]
<INPUT TYPE="text" NAME=cognome2 VALUE="" SIZE=40>
[…]
<INPUT TYPE="text" NAME=nome2 VALUE="" SIZE=40>
[…]
<INPUT TYPE="text" NAME=email2 VALUE="" SIZE=40>
[…]
<TEXTAREA NAME=commenti COLS=50 ROWS=5></TEXTAREA> <br>
[…]
<INPUT TYPE="submit" VALUE="Inoltra">  
<INPUT TYPE="reset" VALUE="Cancella">  
</FORM>
```


Registration.java

```
public class Registration extends HttpServlet {

  […]

  int counter;

  public void service(HttpServletRequest req,
                      HttpServletResponse res)
         throws ServletException, IOException {

     res.setContentType("text/html");
     PrintWriter out = res.getWriter();

[…]

}
```

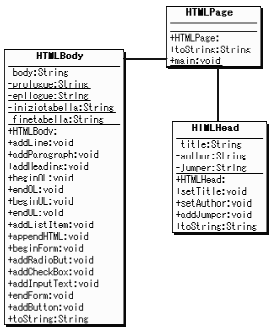
Viene utilizzato sia per l'invio con POST sia con GET

MIME type della risposta

PrintWriter per l'output

HTMLPage

- In una servlet la pagina HTML di risposta da inviare all'utente è costruita mediante una serie di "stampe" ma questo puo` risultare pesante. Una soluzione a questo è data da JSP, un'altra è definirsi in proprio una classe che permetta di definire oggetti di tipo pagina HTML e che si occupino di inserire tutti i comandi HTML necessari alla formattazione di una pagina HTML



Registration.java

```
[…]
HTMLPage page = new HTMLPage();
[…]

String cognome1 = req.getParameter("cognome1").trim();
String nome1    = req.getParameter("nome1").trim();
String email1   = req.getParameter("email1").trim();
[…]

int localCounter;
synchronized(this) {
    localCounter = counter++;
    saveState();
}

[…]
out.println(page);
[…]
```

Attenzione questa classe l'ho definita io per facilitare la creazione di una pagina HTML

Estrazione dei parametri inviati alla servlet dalla form

Incremento del contatore per il numero del gruppo

counter è **una variabile condivisa** dai vari thread delle diverse richieste della stessa servlet

Output della pagina HTML

Registration.java

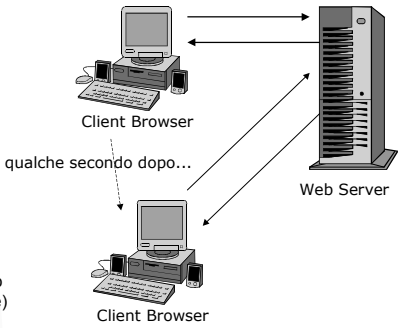
```
public void init(ServletConfig config)
        throws ServletException {
    super.init(config);
    try {
        FileReader fileReader =
            new FileReader("/tmp/InitDestroyCounter.initial");
        BufferedReader bufferedReader =
            new BufferedReader(fileReader);
        String initial = bufferedReader.readLine();
        counter = Integer.parseInt(initial);
        return;
    }
    catch(FileNotFoundException ignored) { }
    catch(IOException ignored) { }
    catch(NumberFormatException ignored) { }
    counter = 1;
}

public void destroy() {
    saveState();
}
```

Questo metodo è eseguito quando la servlet è creata (una volta sola alla prima richiesta)

Questo metodo è eseguito automaticamente prima dello shutdown del Web Server

HTTP È Stateless

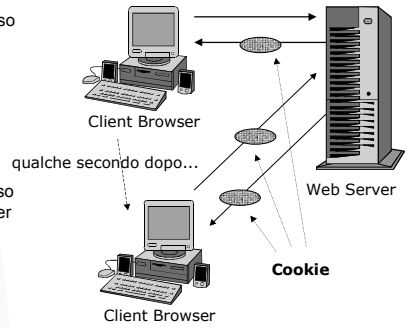
- Una connessione HTTP è stateless
- Non vi è *nessuna relazione* tra una connessione ed una successiva
- Come trattare informazioni che si vorrebbero persistenti? (esempio il carrello della spesa on-line)



19

HTTP È Stateless: Cookie!

- È possibile fare uso dei cookie per memorizzare le informazioni che vogliamo rendere persistenti
- I cookie sono memorizzati presso il client dal browser
- I cookie sono memorizzati in forma di stringhe



20

PesonalAccessCounter.java With Cookie

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

public class PersonalAccessCounterWithCookie extends HttpServlet {
  public void service(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      HTMLPage page = new HTMLPage();
      page.header.setTitle("Contatore personale degli accessi");
      page.header.setAuthor("Matteo Baldoni");
      page.body.addHeading(2, "Contatore personale");
      […]
      page.body.addParagraph("L'utente ha visitato questa pagina " +
                             ival + " volte.");
      out.println(page);
  }
}
```

21

Pesonal Access Counter With Cookie

```
[…]
Cookie[] cookies = req.getCookies();
int ival = 1;
Cookie cookie = null;
if (cookies.length > 0) {
    int i = 0; boolean found = false;
    while (i < cookies.length && !found) {
        cookie = cookies[i];
        if ((cookie.getName()).equals("PersonalCounter")) {
            ival = (new Integer(cookie.getValue())).intValue() + 1;
            cookie.setValue(ival+"");
            found = true;
        }
    }
    if (!found) {
        cookie = new Cookie("PersonalCounter", "1");
    }
}
else {
    cookie = new Cookie("PersonalCounter", "1");
}
res.addCookie(cookie); […]
```

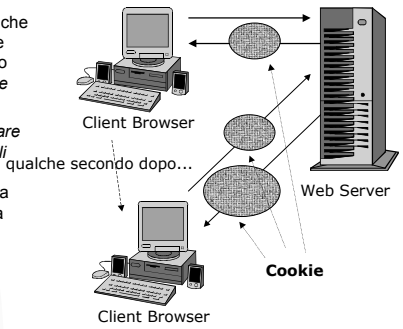
I cookie possono essere prelevati presso l'oggetto HttpServletRequest

Aggiunta di un cookie all'oggetto HttpServletResponse

22

HTTP È Stateless: HttpSession

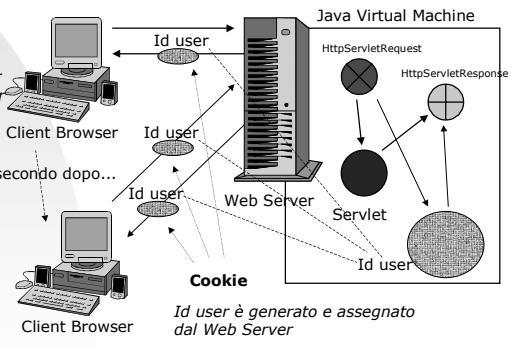
- Se le informazioni che si vogliono rendere persistenti possono *crescere facilmente*
- Perche` non *sfruttare la persistenza degli oggetti presso il server*? Questa è la soluzione proposta con la classe HttpSession



23

HTTP È Stateless: HttpSession

Solo un cookie con lo Id user viaggia dal server al browser mediante un cookie



Id user è generato e assegnato dal Web Server

24

Pesonal Access Counter With Session

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

public class PersonalAccessCounter extends HttpServlet {

  public void service(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {

      HttpSession session = req.getSession(true);

      res.setContentType("text/html");
      PrintWriter out = res.getWriter();

      […]

      out.println(page);
  }
}
```

Creazione di un oggetto di tipo HttpSession se non esiste, previevo dello stesso dalla Java Virtual Machine, altrimenti

Pesonal Access Counter With Session

```
[…]
      HTMLPage page = new HTMLPage();
      page.header.setTitle("Contatore personale degli accessi");
      page.header.setAuthor("Matteo Baldoni");

      page.body.addHeading(2, "Contatore personale");

      Integer ival = (Integer) session.getValue("counter");
      if (ival == null)
          ival = new Integer(1);
      else
          ival = new Integer(ival.intValue() + 1);

      session.putValue("counter", ival);

      page.body.addParagraph("L'utente " +
                             session.getId() +
                             " ha visitato questa pagina " +
                             ival + " volte.");
[…]
```

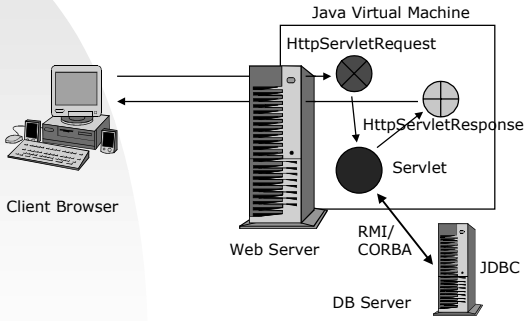
Prelievo di un ***oggetto*** dal proprio HttpSession

Memorizzazione di un ***oggetto***

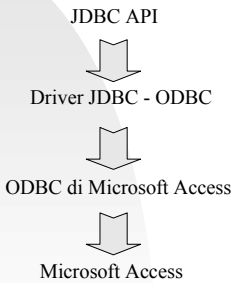
HttpSession: vantaggi

- Piu` facili da gestire
- Solo l'identificativo dell'utente viene memorizzato tramite cookie
- Si possono memorizzare veri e propri oggetti mentre con la tecnica dei cookie si deve trasformare l'informazione da rendere persistente in stringa da spedire al browser
- Sfruttano la persistenza degli oggetti presso il Web Server

Internet 3-level Architecture



JDBC



- Interfaccia di programmazione Java per la connessione e l'uso di database
- Permette di scrivere un'interfaccia SQL unica per i principali DB: Oracle, MS Access, Sybase, ...
- Sfrutta la security di Java e quella del database attraverso lo specifico driver

Bibliografia

- Paolo Malacarne. Java Servlet. Apogeo, 2001. ISBN 88-7303-870-0. 21,69 euro.
- Giuseppe Naccarato. Java database e programmazone client/server. Apogeo, 2001. ISBN 88-7302-773-9. 35,64 euro.