Le basi di J2EE

Servlets e Tomcat

Emanuele Rosso

12/05/2004

Scaletta

- Introduzione
- Web dinamico
- Le basi dei servlet
  - API
  - Ciclo di vita
  - Esempio
- Tomcat: installazione e configurazione

L'evoluzione del Web

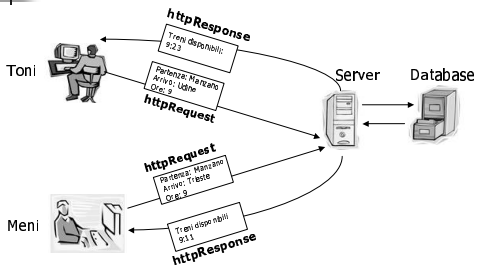
- Web statico: documenti statici e nessuna interazione col client
- Web dinamico:
  - Programmi CGI, che risiedono sul server, accettano le richieste, utilizzano le risorse lato server e generano come risposta una pagina html
  - Script lato client, programmi interpretati che consentono di alleggerire parte del lavoro del server, in quanto permettono a piccole applicazioni di essere eseguite sul client

L'evoluzione del Web

Un esempio:

Toni abita a Manzano e deve essere a Udine alle 10. Va sul sito delle ferrovie, inserisce la stazione di partenza e di arrivo e l'orario e vengono visualizzati tutti i treni che Toni può prendere. Meni abita sempre a Manzano ma deve andare a Trieste. Inserisce tutte le informazioni e ottiene un elenco di treni per Trieste. Come è possibile che due persone diverse ricevano informazioni diverse dallo stesso sito?

L'evoluzione del Web



L'evoluzione del Web

- Perché il sito dispone di un' **applicazione Web** che accetta le richieste degli utenti e genera una risposta pertinente per ciascuna interrogazione
- Poiché l'applicazione Web crea questi documenti al volo, si dice che restituisce un contenuto dinamico.

L'evoluzione del Web

- Quando bisogna costruire pagine Web dinamicamente? Ad esempio:
  - La pagina Web si basa su dati inviati dal client (Toni e Meni)
  - La pagina Web si ricava da dati che cambiano frequentemente (previsioni del tempo)
  - La pagina Web usa informazioni provenienti da database o altre risorse server-side

7

J2EE

- J2EE (Java 2 Enterprise Edition): specifica che definisce:
  - Le parti di un'applicazione d'impresa
  - I servizi che essa richiede
  - Come comunicano queste parti.
- J2EE è composta da 13 tecnologie (JSP, Servlet, JMS, JDBC, JNDI, JTS, JTA, JavaMail, JAF, RMI, EJB, XML, Java IDL)

8

Servlet

- Un servlet è:
  - Un programma Java che gira su un server
  - Costruisce pagine Web dinamiche
  - Agisce come intermediario tra le richieste provenienti dal Web browser o altri HTTP client e database o applicazioni sul server HTTP
- Cosa fa più precisamente?
  - Riceve una richiesta HTTP
  - Costruisce una pagina Web
  - Prepara e spedisce la risposta HTTP

9

Tomcat

- E' un server Web gratuito in grado di gestire richieste per HTML, JSP e servlet
- E' un contenitore di servlet
- Consente l'esecuzione dei servlet
- Si può utilizzare sul proprio desktop

10

La struttura base dei servlet

- I servlet sono scritti in Java
- Importano le classi in `java.io`, `javax.servlet` e `javax.servlet.http`
- estendono `HttpServlet`
- sovrascrivono i metodi `doGet` o `doPost`, che hanno 2 argomenti:
  - `HttpServletRequest`, che gestisce i dati in ingresso
  - `HttpServletResponse` che gestisce i dati in uscita e ti permette di ottenere un `PrintWriter`
- `doGet` e `doPost` lanciano le eccezioni `ServletException` e `IOException`

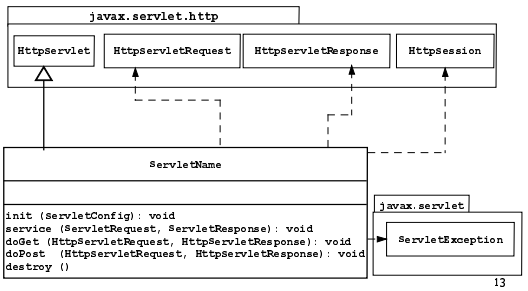
11

Commenti

- I servlet sono framework: il programmatore di un servlet deve 'riempire' i metodi `doGet` o `doPost`
- `doGet` e `doPost` sono metodi hook, come quelli di Template Method
- Il browser genera una richiesta GET quando l'utente inserisce un URL, clicca su un link o invia un form con attributo `method="get"`
- Il browser genera una richiesta POST quando l'utente invia un form con attributo `method="post"`

12

UML dei servlet



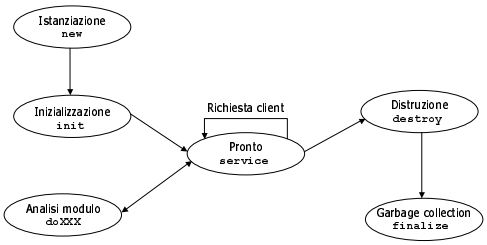
13

Il ciclo di vita del servlet

- Quando il servlet è creato viene invocato il suo metodo **`init`**
- Ogni richiesta dell'utente genera un thread che chiama il metodo **`service`** dell'istanza precedentemente creata
- Il metodo **`service`** chiama **`doGet`** o **`doPost`**
- Infine se il server decide di eliminare un servlet chiama il metodo **`destroy`**

14

Ciclo di vita del servlet



15

Esempio di servlet

- Vediamo un semplice esempio di servlet che crea una tabella di potenze di 2
- Non è un servlet dinamico, ma è più che sufficiente come inizio!
- Ci servono:
  - Un file HTML (**`index.html`**)
  - Un file Java (**`PowersOf2.java`**)
  - Il deployment descriptor (**`web.xml`**)

16

index.html

```
<html>
   <head>
     <title>Esempio di servlet</title>
   </head>
   <body>
     <h1>Potenze di 2</h1>
     <p>Clicca <a
  href="servlet/esempio">qui</a>
        per vedere il servlet </p>
   </body>
</html>
```

17

PowersOf2.java (1/2)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PowersOf2 extends HttpServlet
{
  public void doGet(HttpServletRequest request,
                HttpServletResponse response)
                  throws IOException, ServletException
  {
     response.setContentType("text/html");
     ServletOutputStream out=response.getOutputStream();
     out.print("<html>");
     out.print("<head><title>Potenze di 2</title></head>");
     out.print("<body>");
     out.print("<center>");
```

18

PowersOf2.java (2/2)

```
  out.print("<h2>Ecco le potenze di 2</h2>");
  out.print("</center>");
  out.print("<table border ='2' align='center'>");
  out.print("<th>Esponente</th><th>2^Esponente</th>");
  for(int i=0;i<=9;i++)
      {
            out.print("<tr><td>"+ i + "</td>");
            out.print("<td>" + Math.pow(2,i) + "</td>");
            out.print("</tr>");
      }
      out.print("</table></body></html>");
      out.close();
  }
}
```


web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>
  <servlet>
    <servlet-name>Example</servlet-name>
    <servlet-class>PowersOf2</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Example</servlet-name>
    <url-pattern>/servlet/esempio</url-pattern>
  </servlet-mapping>
</web-app>
```


Spiegazione

- Le applicazioni Web utilizzano un file web.xml
  - descrive le risorse lato server che le compongono
  - contiene 2 elementi:
    - `servlet` assegna il nome al servlet
      - sottoelementi `servlet-name` (nome del servlet) e `servlet-class` (nome della classe)
    - `servlet-mapping` assegna l'URL al servlet
      - sottoelementi `servlet-name` (nome del servlet) e `url-pattern` (URL)

Spiegazione

- Il file **`index.html`** contiene un link che 'punta' all'URL **`servlet/esempio`** (è quello associato al servlet Example)
- Il file **`PowersOf2.java`** ha la struttura standard di un servlet
  - Il metodo **`setContentType`** di **`HttpServletResponse`** indica che il documento ritornato è un file HTML
  - Il metodo **`getOutputStream`** di **`HttpServletResponse`** consente di ottenere un oggetto **`ServletOutputStream`** per scrivere nel file HTML

Compilazione

- Devo compilare il file Java, ma le classi **`javax.servlet`** e **`javax.servlet.http`** non sono nell'edizione di Java standard!
- Devo settare la variabile d'ambiente CLASSPATH:
  - Vai in Pannello di controllo -> Sistema -> Avanzate -> Variabili d'ambiente
  - Aggiungi una nuova variabile d'ambiente e come valore metti il path delle classi **`javax.servlet`** e **`javax.servlet.http`**, che di solito sono in **`$TOMCAT_HOME/common/lib/servlet.jar`**

Dove metto i file?

- Posso metterli in `$TOMCAT_HOME/webapps/` così

```
+-- $TOMCAT_HOME/webapps/
      |
      +--Example/
           |
           +-- index.html
           +-- WEB-INF/
                 |
                 +-- web.xml
                 +-- classes/
                       |
                       +-- PowersOf2.class
                 +-- lib/
```

Verifichiamo

- Riavviamo Tomcat
- Richiesta HTTP a http://localhost:8080/Example e …

Ecco le potenze di 2

Esponente	2^Esponente
0	1.0
1	2.0
2	4.0
3	8.0
4	16.0
5	32.0
6	64.0
7	128.0
8	256.0
9	512.0

25

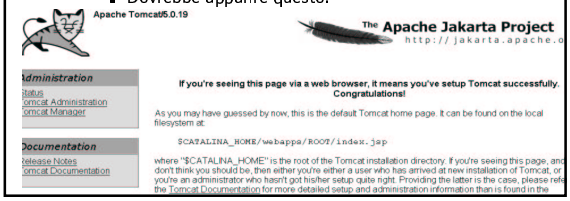
Configurazione di Tomcat

- Scarica e installa il Java Software Development Kit
- Scarica un server:
  - Vai alla pagina http://jakarta.apache.org/site/binindex.cgi,
  - Scegli l'ultima release build di Tomcat
- Configura il server
  - Setta la variabile JAVA_HOME:
    - Apri col notepad il file `install_dir/bin/catalina.bat`
    - Dopo le righe di commento scrivi `set JAVA_HOME=C:\jdk`

26

Configurazione di Tomcat

- Testing:
  - Double-click su `install_dir/bin/startup.bat`
  - Apri un browser e inserisci http://localhost:8080
  - Dovrebbe apparire questo:



28

Le alternative

- Tomcat non è l'unico contenitore di servlet a disposizione
- Per gestire compiti di elaborazione superiori possiamo orientarci sui server di applicazioni
- Sono software che gestiscono l'elaborazione delle applicazioni

Perché usare Tomcat?

- Perché usare un server sul proprio desktop, piuttosto che utilizzare un server remoto?
  - E' più facile da testare
  - E' più facile fare il debug
  - E' più facile il restart
  - Benchmark più affidabili
  - E' tutto sotto il tuo controllo
  - E' più facile da installare

29

I vantaggi dei servlet

- Efficienti: usano i thread e rimangono in memoria centrale
- Conveniente: gestiscono parsing e decodifica automatica dei dati di un form, lettura e settaggio header, gestione cookie e sessioni e poi conosciamo già Java!
- Portabili: usano Java
- Economici: Tomcat è gratis!
- Sicuri: Controllo dei limiti dell' array e altre protezioni della memoria sono una parte centrale di Java

30

Configurazione di Tomcat

- Il problema:
  - Se usi una vecchia versione di Windows (Windows 98 / ME o più vecchia) devi cambiare i DOS memory settings per gli script di startup e shutdown:
    - Right click su install_dir/bin/startup.bat
    - Seleziona Properties
    - Seleziona Memory
    - Cambia Initial Environment da Auto a 2816
    - Ripeti il processo per install_dir/bin/shutdown.bat

31

Gestione dello stato

- Se volete (e se c'è tempo) ne parliamo
- Alcune keyword:
  - Stato
  - Sessione
  - Cookie
  - Session tracking

32

Riassunto

- Web dinamico
- Le basi dei servlet
  - Api
  - Ciclo di vita
  - Esempio
- Tomcat: installazione e configurazione

33

Bibliografia

- M. Hall, L. Brown, *Core Servlets and JavaServer Pages,* Prentice Hall, 2004
- J. Annunziato, S. F. Kaminaris, *Imparare JavaServer Pages in 24 ore,* Tecniche nuove, 2001
- http://jakarta.apache.org/site/binindex.cgi
- http://java.sun.com

34

Gestione dello stato

- HTPP è un protocollo privo di informazioni sullo stato
  - I sever si preoccupano di un client in particolare solo fino al momento in cui soddisfano la richiesta
  - HTTP non dispone di meccanismi per distinguere un client dal successivo
- Esempio: carrello della spesa online da consegnare alla fine alla cassa. Il server deve essere in grado di distinguere quali richieste provengono da un utente in modo da inserire gli articoli corretti nel carrello

35

Gestione dello stato

- Sessione: interazione comprensiva di informazioni sullo stato di un client con un sito Web
- Tre meccanismi per la gestione dello stato:
  - I cookie
  - Oggetto session
  - Codifica dell'url

36

I cookie

- Perché si usano?
  - Identificare un utente durante una sessione di e-commerce
  - Ricordare username e password
  - Personalizzare siti
  - Focalizzare la pubblicità
- Attenzione: Problemi di privacy (alcuni utenti li disabilitano, potrebbero scomparire)

37

I cookie

- Mandare cookie al client:
  - Creare un oggetto cookie
  - Settare il tempo di permanenza del cookie nel disco col metodo **`setMaxAge`**
  - Mettere il cookie nell'HTTP response header usando **`response.addCookie`**
- Leggere il cookie dal client
  - Chiamare **`request.getCookies`**, in modo da ottenere un array di cookie
  - Chiamare **`getName`** su ognuno dei cookie dell' array finchè non viene trovato il cookie di interesse

38

Session tracking

- E' una tecnica molto semplice, posta ad un livello più alto dei cookie, su cui si basa. Come si usa?
  - Accedere all'oggetto session associato alla richiesta corrente: **`HTTPSession session = request.getSession`**
  - Accedere all'informazione associata alla sessione: **`SomeClass value = (SomeClass) session.getAttribute ("someIdentifier")`**
  - Memorizzare informazione in una sessione: **`session.setAttribute ("someIdentifier", value)`**

39

Session tracking

- Rimuovere una sessione: con **`removeAttribute`** rimuovi un valore, con invalidate un'intera sessione

40