Sviluppo di Applicazioni Web con Java 2 Enterprise Edition

Ivan Scagnetto

Dipartimento di Matematica e Informatica

http://www.dimi.uniud.it/scagnett

scagnett@dimi.uniud.it

Applicazioni Web (Web Application)

- Sono applicazioni utilizzabili mediante il web browser: l'HTML sostituisce la classica GUI.
- Permettono di eseguire dei programmi su un server.
- Esempi:
  - Web mail
  - Commercio elettronico

Java 2 Enterprise Edition

- E’ la release di Java che permette di costruire applicazioni Web.
- Componenti di J2EE:
  - **Servlet**
  - **JSP (Java Server Pages)**
  - **JDBC (Java DataBase Connectivity)**
  - Supporto per XML
  - RMI (Remote Method Invocation)
  - CORBA (Common Object Request Broker Access)
  - JNDI (Java Naming and Directory Interface)
  - JMS (Java Message Service)
  - JavaMail
  - EJB (Enterprise Java Beans)
  - …

Bibliografia

- Wrox team

  Professional Java Server Programming – J2EE 1.3 Edition

  Wrox 2001
- David Flanagan, Jim Farley, William Crawford

  Java Enterprise in a Nutshell (2nd Edition)

  O'Reilly 2002

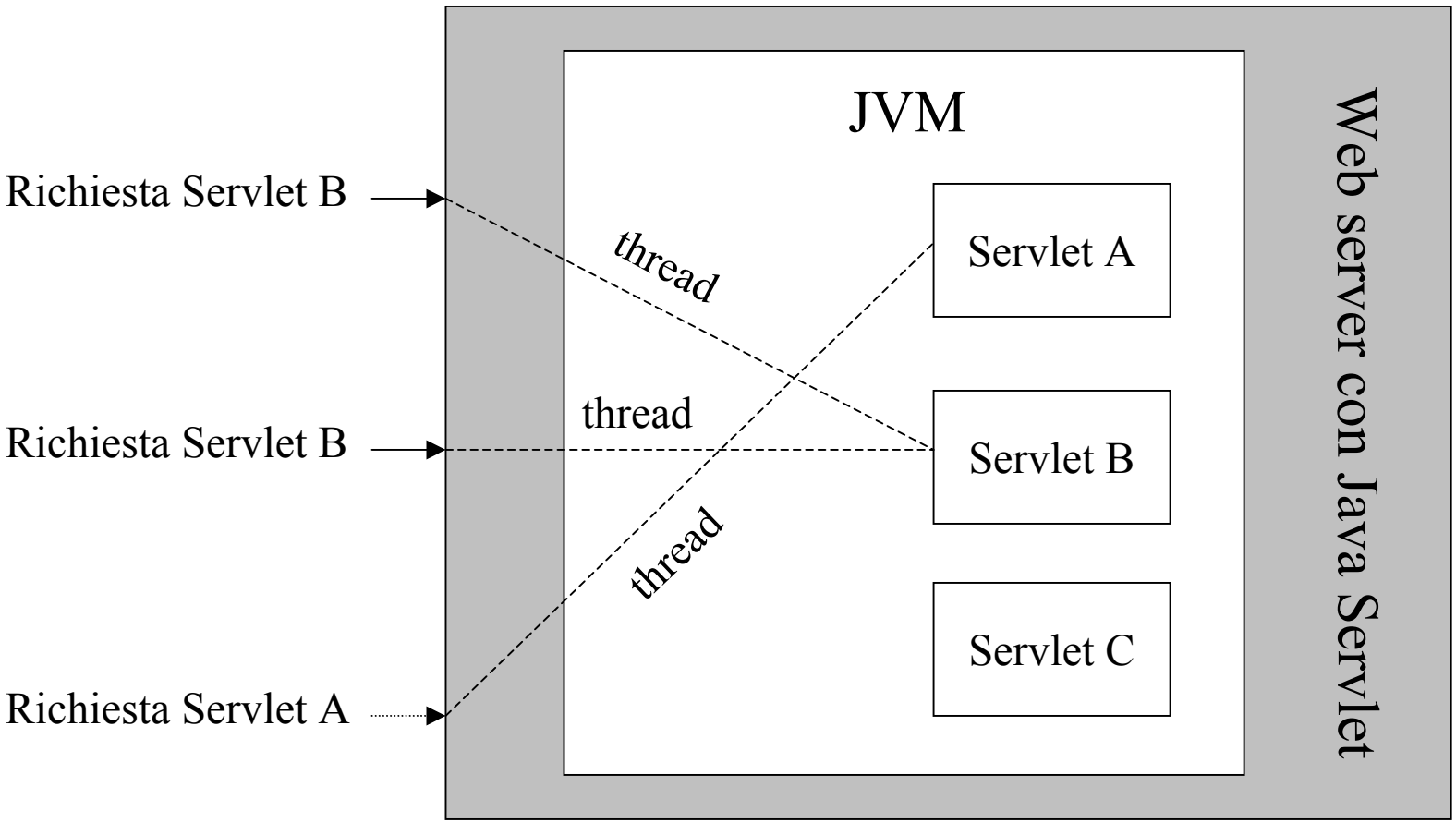
Java Servlet

- Una Java servlet è un'estensione del server, ovvero, una classe Java che può essere caricata dinamicamente per estendere le funzionalità del server.
- Di solito le servlet sono utilizzate nei web server (in modo da fornire un'alternativa agli script CGI).
- Le servlet girano in una JVM sul server, quindi sono sicure e portabili.
- A differenza delle applet **non richiedono** alcun supporto per Java nel web browser.

Vantaggi delle Servlet

- Tutte le richieste alle servlet vengono implementate come thread distinti all'interno dello stesso processo: ne derivano **efficienza** e **scalabilità**.
- Le servlet sono **portabili**:
  - da un sistema operativo ad un altro;
  - da un web server ad un altro (tutti i principali web server supportano le servlet).

Java Servlet e Web Server



Java Servlet: ambito di utilizzo

- In generale le Java servlet sono utilizzate come rimpiazzamento degli script CGI su un web server.
- Tuttavia nulla vieta di utilizzare tale tecnologia con altri tipi di server: FTP server, mail server ecc.

Supporto per le Java Servlet

- Per utilizzare le Java Servlet occorre:
  - JVM;
  - Servlet API (classi **`javax.servlet`** e **`javax.servlet.http`**), fornite in bundle con il JSDK o incorporate in alcuni web server;
  - un Servlet Engine; le tipologie disponibili sono:
    - standalone;
    - add-on;
    - embeddable.

Standalone Server Engine

- Si tratta di un server che fornisce un supporto nativo per le Java Servlet. Alcuni esempi sono i seguenti:
  - Sun's Java Web Server ("Jeeves"):
    http://java.sun.com/products/
  - World Wide Web Consortium's Jigsaw Server:
    http://www.w3.org/Jigsaw
  - O'Reilly's WebSite Professional:
    http://website.oreilly.com
  - Netscape's Enterprise Server:
    http://wp.netscape.com/enterprise/v3.6

Application Server Engine

- Un application server fornisce le primitive lato server per lo sviluppo di applicazioni di tipo enterprise. Fra quelli che supportano le servlet vi sono i seguenti:
  - BEA WebLogic Application Server:

    http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic
  - ATG's Dynamo Application Server 3:

    http://www.atg.com/

Add-on Servlet Engine

- Questa tipologia consiste in plug-in che aggiungono il supporto alle servlet per server esistenti e privi di tale supporto.
- Alcuni fra i più diffusi sono:
  - Java-Apache project's JServ module: http://java.apache.org/
  - IBM's WebSphere Application Server: http://www.software.ibm.com/webservers/
  - New Atlanta's ServletExec: http://www.newatlanta.com/

Embeddable Servlet Engine

- Questa tipologia consiste in una piattaforma che supporta le servlet e che può essere inserita in un'altra applicazione (il vero server).
- Alcuni esempi:
  - Sun's JavaServer Engine: http://java.sun.com/products/javaserverengine/
  - Jef Poskanzer's Acme.Serve: http://www.acme.com/java/software/Package-Acme.Serve.html

Scegliere un servlet engine

- Non tutti i servlet engine sono uguali.
- Prima di scegliere un particolare engine, conviene testarlo per vedere se supporta le funzionalità necessarie.
- La lista dei servlet engine disponibili è mantenuta aggiornata dalla Sun al seguente URL:

  http://java.sun.com/products/servlet/industry.html

Caratteristiche delle servlet

- Portabilità
- Potenza
- Efficienza e persistenza
- Sicurezza
- Eleganza
- Integrazione
- Estendibilità e flessibilità

Portabilità

- Essendo scritte in Java e basandosi su un'API standard, le servlet sono altamente portabili fra diversi sistemi operativi e diversi server: "**write once, serve everywhere**".
- Tuttavia la portabilità non è strettamente necessaria: le servlet devono girare solo sul server di sviluppo e produzione (cfr. la necessità di un applet di girare su tutti i client possibili).
- Le servlet non utilizzano la parte più soggetta ad errori e mal implementata di Java: l'AWT.

Potenza

- Le servlet possono sfruttare tutta la potenza delle API principali di Java:
  - networking,
  - multithreading,
  - manipolazione di immagini,
  - compressione dei dati,
  - connessione a basi di dati,
  - …

Efficienza e persistenza

- Una volta caricata nella memoria del server, una servlet vi rimane come istanza di un oggetto. Ogni successiva chiamata alla servlet è quindi servita in modo immediato.
- La permanenza in memoria come istanza di un oggetto permette ad una servlet di mantenere uno stato.

Sicurezza

- Le servlet ereditano la “type safety” e meccanismi come il “garbage collector” direttamente dal linguaggio Java.
- La mancanza dei puntatori in Java esclude problemi legati alla gestione esplicita della memoria (e.g., memory leak).
- Il meccanismo delle eccezioni protegge il server dagli errori a run-time (e.g., divisioni per zero).
- Il Java security manager consente di ottenere un ulteriore livello di sicurezza.

Eleganza

- La pulizia e modularità del codice delle servlet deriva direttamente dall'API delle servlet stessa.
- Infatti l'API contiene molte classi utili per il trattamento dei cookie, della sessione ecc.

Integrazione

- A differenza degli script CGI, le servlet sono strettamente integrate con il server.
- Ciò consente di utilizzare il server per compiti come i seguenti:
  - convertire i percorsi dei file,
  - effettuare dei log,
  - controllare le autorizzazioni d'accesso,
  - …

Estendibilità e flessibilità

- L'API delle servlet è stata progettata per essere facilmente estesa con nuove funzionalità.
- Inoltre, le servlet sono flessibili:
  - possono generare una pagina web completa,
  - possono essere incluse in una pagina statica con il tag **`<SERVLET>`** (server-side include),
  - la tecnologia Java Server Pages consente di inserire dei frammenti di codice delle servlet direttamente in una pagina statica (come in ASP o PHP).

Servlet API

- Le servlet usano le classi e interfacce di due package:
  - **`javax.servlet`** (servlet generiche indipendenti dal protocollo);
  - **`javax.servlet.http`** (servlet specifiche per il protocollo http).
- Ogni servlet deve implementare l'interfaccia **`javax.servlet.Servlet`**. Ciò solitamente avviene estendendo:
  - **`javax.servlet.GenericServlet`** (servlet generiche indipendenti dal protocollo);
  - **`javax.servlet.http.HttpServlet`** (servlet specifiche per il protocollo http).

Servlet API

- Le servlet non hanno il metodo **`main()`**.
- Il server invoca certi metodi specifici in risposta ad una richiesta:
  - **`service()`** viene chiamato ogni volta che una richiesta è inoltrata ad una servlet.
- Una servlet generica deve quindi fare un overriding del metodo **`service()`**.
- Una servlet HTTP invece esegue l'overriding dei metodi **`doGet()`** e **`doPost()`**. Il metodo **`service()`** in questo caso coordina l'inoltro delle richieste ai due metodi precedenti e non deve quindi essere modificato.

La prima servlet: Ciao, mondo!

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

public class CiaoMondo extends HttpServlet {
        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>Ciao, mondo!</TITLE></HEAD>");
                out.println("<BODY>");
                out.println("<STRONG>Ciao, mondo!</STRONG>");
                out.println("</BODY></HTML>");
        }
}
```

Tomcat

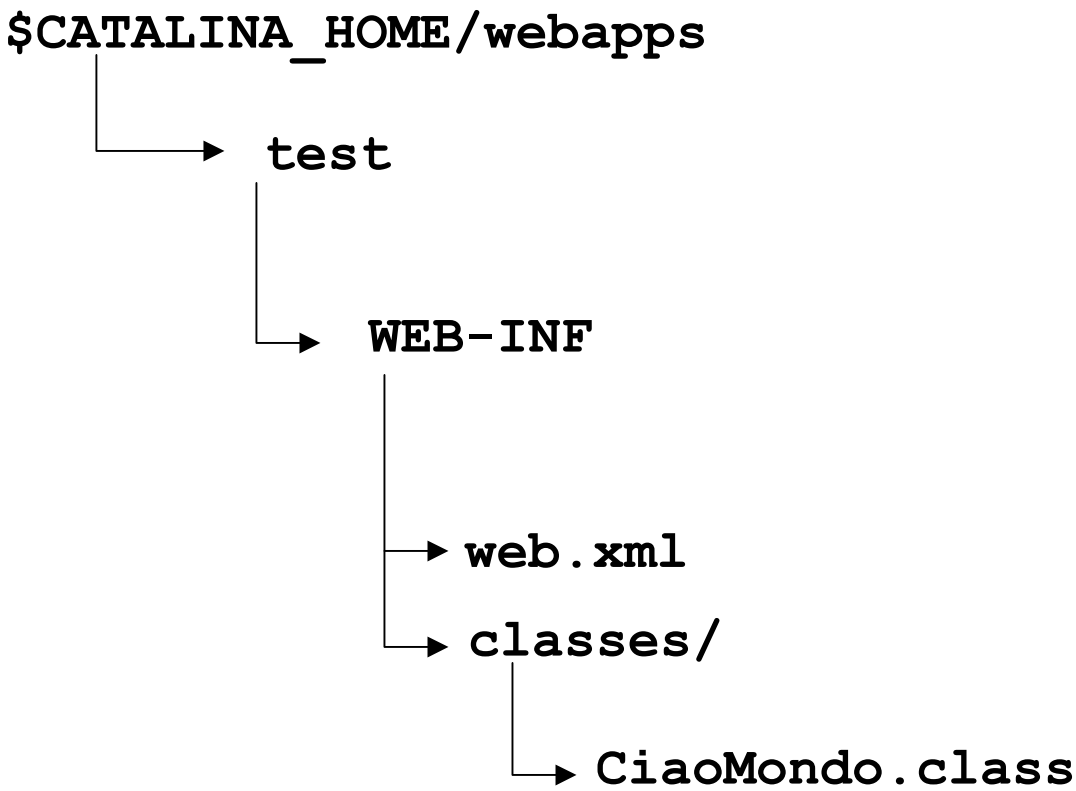
- Il servlet engine che useremo sarà Tomcat:
  http://jakarta.apache.org/
- Installato su latoserver.dimi.uniud.it sulla porta 8080:
  http://latoserver.dimi.uniud.it:8080

Deployment su Tomcat

- Compilare la servlet, ricordandosi di includere nel classpath (tramite la viariabile d'ambiente **`CLASSPATH`** oppure l'opzione **`-classpath`** di **`javac`**) l'archivio **`servlet-api.jar`** (o **`servlet.jar`**) presente nella directory (**`$CATALINA_HOME`** è la directory root dell'installazione di Tomcat):

  `$CATALINA_HOME/common/lib`
- Scrivere il deployment descriptor file **`web.xml`**.
- Copiare i file sul server.
- Fermare e riavviare il servizio.

Deployment su Tomcat



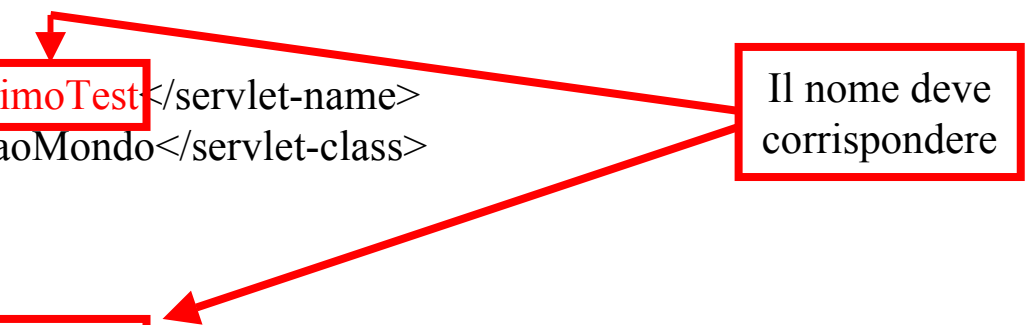
web.xml

```
<?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>PrimoTest</servlet-name>
      <servlet-class>CiaoMondo</servlet-class>
   </servlet>

   <servlet-mapping>
      <servlet-name>PrimoTest</servlet-name>
      <url-pattern>/servlet/Primo</url-pattern>
   </servlet-mapping>
</web-app>
```



Deployment su latoserver.dimi.uniud.it (I)

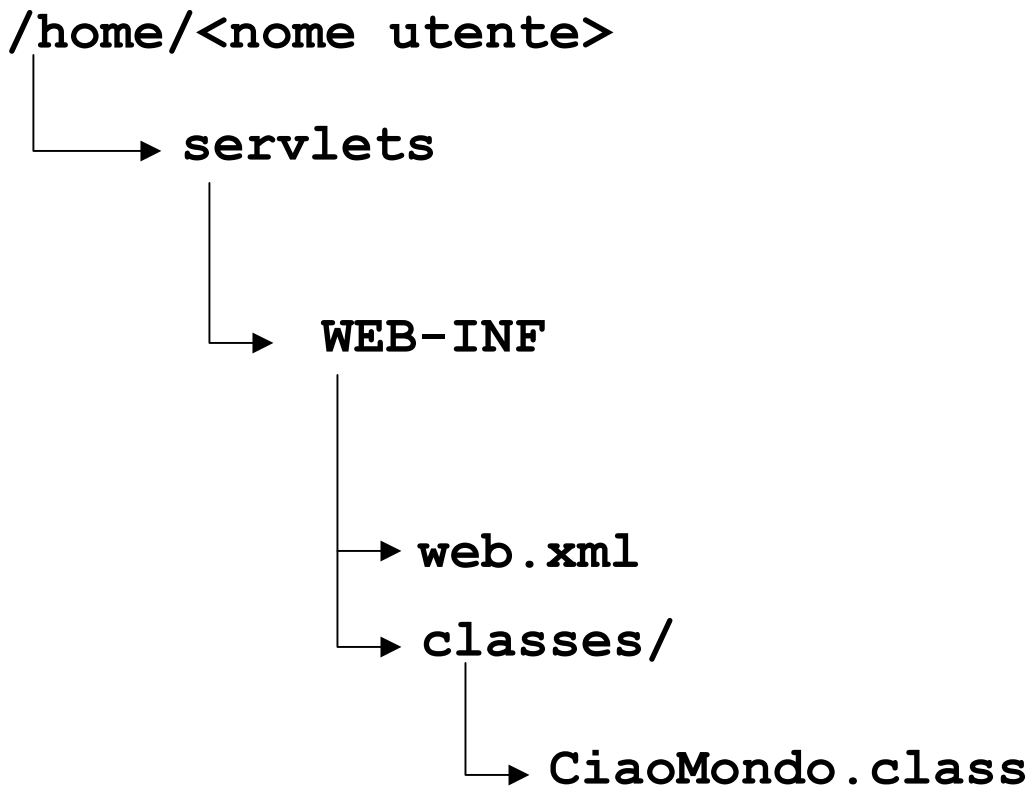
- Ogni utente nella propria home directory (**/home/<nome utente>**) ha una sottodirectory **servlets**:

```
drwxr-x--T 2 <nome utente> tomcat 4096 2006-10-10 11:42 servlets
```

- Tale directory va usata per rendere disponibili le proprie servlet a Tomcat, copiandovi i relativi file **.class** e il file **web.xml**.
- Se non esistono, creare le directory **WEB-INF** e **WEB-INF/classes**:

```
cd ~/servlets; mkdir WEB-INF; mkdir WEB-INF/classes
```

Deployment su latoserver.dimi.uniud.it (II)



Deployment su latoserver.dimi.uniud.it (III)

- Compilare la servlet (dalla directory in cui si è creato il sorgente):

  `javac CiaoMondo.java`

  oppure, se la variabile d'ambiente **`CLASSPATH`** non è definita correttamente:

  `javac -classpath /usr/share/jdk1.5.0_09/jre/lib/servlet-api.jar CiaoMondo.java`

- Spostare il file CiaoMondo.class in **`~/servlets/WEB-INF/classes`** con il comando

  `mv CiaoMondo.class ~/servlets/WEB-INF/classes`

- Creare (o copiare) il file **`web.xml`** in **`~/servlets/WEB-INF`**

Deployment su latoserver.dimi.uniud.it (IV)

- Accedere a **`http://latoserver.dimi.uniud.it:8080`** e cliccare sul link “Tomcat Manager” (autenticandosi con user e password forniti a lezione):

Tomcat Manager

Deployment su latoserver.dimi.uniud.it (V)

- Fra le varie applicazioni gestite dal Tomcat Manager, individuare quella relativa al proprio nome utente, fare clic su Stop e poi su Start (importante: non cliccare Undeploy e non interferire con le applicazioni degli altri utenti).
- Accedere alla servlet:

```
http://latoserver.dimi.uniud.it:8080/
<nome utente>/servlet/Primo
```