



Tomcat & Servlet

Programmazione in Ambienti
Distribuiti

V 1.2 © Marco Torchiano 2005

Contenuti

- Tomcat
 - Applicazioni Web
 - ◆ Struttura
 - ◆ Sviluppo
 - ◆ Deployment
 - Servlet
 - JSP
 - Uso delle sessioni
-

Tomcat

- Tomcat è un contenitore di Servlet
 - Sviluppato come prodotto open-source dalla Apache Foundation
 - È in grado di
 - ♦ Funzionare come HTTP server
 - ♦ Attivare Servlet
 - in corrispondenza di specifici URL
 - ♦ Compilare ed attivare Java Server Pages (JSP)
 - Versione
 - ♦ In queste slide si fa riferimento a Tomcat 5.0
-

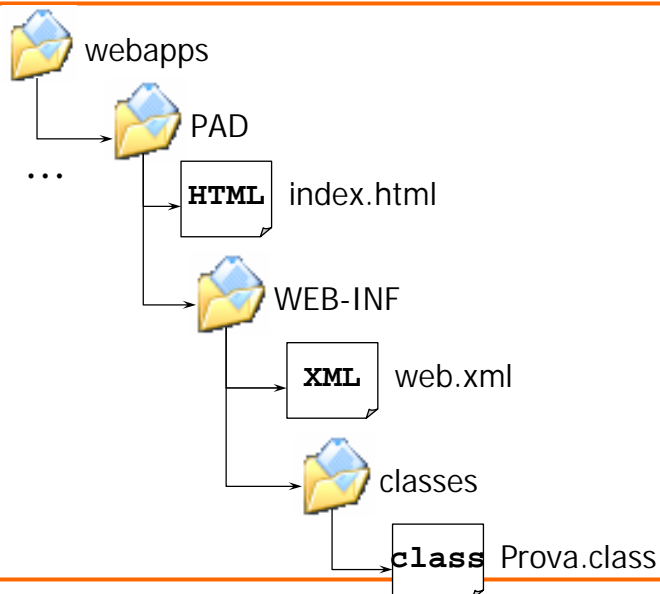
Installazione

- Tomcat è di solito installato (in Windows) nel folder **Programmi\Apache Software Foundation\Tomcat ?.**
 - Questo folder è indicato con **\$TOMCAT_HOME**
 - All'interno di questo folder ci sono:
 - ♦ **conf**: i file di configurazione
 - ♦ **bin**: gli eseguibili
 - ♦ **common**: le librerie (es. servlet.jar)
 - ♦ **webapp**: le applicazioni web
 - ♦ **log**: i file di log
-

Applicazioni web

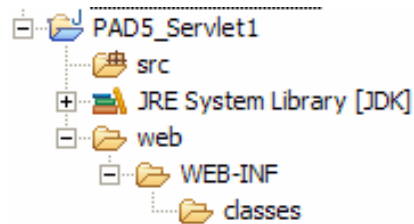
- L'applicazione web è l'unità di *deployment* del software per i contenitori di servlet (Java application server)
- Le applicazioni web sono composte da:
 - ♦ Pagine HTML
 - ♦ Altri file (es. Immagini)
 - ♦ Java Server Pages
 - ♦ Servlet (**WEB-INF\classes\...**)
 - ♦ Librerie di supporto (**WEB-INF\lib\...**)
 - ♦ Descrittore (**WEB-INF\web.xml**)

Struttura



Sviluppo con Eclipse (1)

- Creare un progetto Java
 - ♦ "Create separate source and output folders"
- Creare un nuovo folder "web"
- Creare un sotto-folder "WEB-INF"
- Creare un sotto-sotto-folder "classes"



Sviluppo con Eclipse (2)

- In Project Properties/ Java Build Path
 - ♦ Default output folder
 - NomeProgetto/web/WEB-INF/classes
 - ♦ Add External Jar
 - \$TOMCAT_HOME/common/lib/servlet-api.jar
- Nel folder web/WEB-INF
 - ♦ creare il file web.xml

index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html >
<head><title>Prova servlet</title></head>
<body>
<h1>Prova</h1>
<br>
Questo &grave; un
<a href="Prova" >link</a>. <br>
</body>
</html >
```

Servlet (1)

```
public class Prova extends HttpServlet {
    public void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    { doGet(request, response); }
    public void doGet(
        HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException
    { /* ... */ }
}
```

Servlet (2) – doGet()

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html><head><title>");
out.println("Prova servlet</title></head>");
out.println("<body bgcolor=\"white\">");
out.println("<h1>Prova servlet</h1>");
out.println("<href=\"index.html\">home</a>");
out.println("</body></html>");
```

web.xml

```
<web-app>
<display-name>PAD</display-name>
<description>Esempio PAD</description>
<servlet>
  <servlet-name>Prova</servlet-name>
  <servlet-class>Prova</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Prova</servlet-name>
  <url-pattern>/Prova</url-pattern>
</servlet-mapping>
</web-app>
```

Deployment

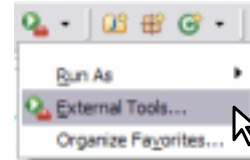
- Uso del Tomcat Manager
 - ♦ `http://localhost:8080/manager/html`
- Deployment da directory local
 - ♦ Context path: nome applicazione
 - ♦ Directory URL: la directory di base
- Deployment remoto
 - ♦ WAR file to deploy

Deployment Locale - Esempio



The screenshot shows the 'Deploy' form in the Tomcat Manager interface. It has a title bar 'Deploy' and a subtitle 'Deploy directory or WAR file located on server'. The form contains three input fields: 'Context Path (optional):' with the value '/PAD', 'XML Configuration file URL:', and 'WAR or Directory URL:' with the value 'file://C:\DSEKE\workspace\PAD5_Servlet1\w'. A 'Deploy' button is located below the third field. A yellow callout bubble points to the 'Context Path' field with the text '/PAD'. Another yellow callout bubble points to the 'WAR or Directory URL' field with the text 'file://D:\workspace\Progetto\web'.

Deployment Remoto

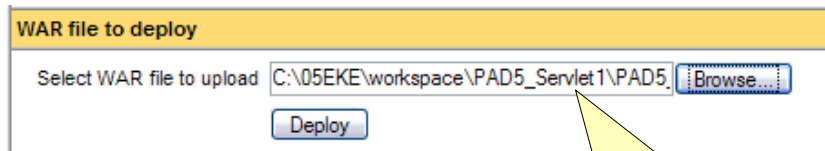
- Creare un file WAR:
 - ♦ Scegliere "External Tools..."
 - ♦ Scegliere Program + New
 - ♦ Location:
 - `${env_var: JAVA_HOME}\bin\jar.exe`
 - ♦ Working directory
 - `${project_loc}\web`
 - ♦ Arguments
 - `-cf ..\PAD.war *`



War

- Dopo la creazione del WAR è necessario fare un refresh  del progetto per vederlo
- Oppure, nell'external tool
 - ♦ Selezionare il tab  Refresh
 - ♦ Attivare: Refresh resources upon completion.
 - ♦ Scegliere: The project containing the...

Deployment Remoto - Esempio



WAR file to deploy

Select WAR file to upload C:\05EKE\workspace\PAD5_Servlet1\PAD5. Browse...

Deploy

C: \workspace\Progetto\Progetto.war

Re-deployment

- Applicazioni locali
 - ◆ Modificando i file (da Eclipse) si opera sui file visti da Tomcat
 - ◆ Per rendere attiva una modifica è sufficiente fare un "reload" dell'applicazione
 - Applicazioni remote
 - ◆ Fare un Undeploy
 - ◆ Rifare un deploy del War
-

Java Server Pages

- Sono "equivalenti" ai servlet
- Hanno una sintassi simile all'HTML con dei tag speciali `<% e %>` che racchiudono codice Java.
- Tomcat internamente converte i file JSP in servlet che generano l'HTML ed eseguono i frammenti Java
 - ♦ `$TOMCAT_HOME\work\Catalina\localhost...`
 - ♦ `...\Context\org\apache\jsp`

JSP Esempio

Direttive

```
<%@ page contentType="text/html" language="java"%>
<%@ page import="java.util.*" %>
<html >
<head><title>Prova JSP</title></head>
<body>
<%
int i;
for(i=0; i<10; ++i){
    %>
    Riga n. <%=i%><br>
    <%
} %>
</body></html >
```

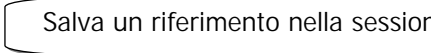
Inserisce il risultato dell'espressione nell'HTML

JSP Oggetti predefiniti

- request
 - ◆ L'oggetto HttpServletRequest
 - response
 - ◆ L'oggetto HttpServletResponse
 - session
 - ◆ L'oggetto HttpSession
 - application
 - ◆ L'oggetto ServletContext
-

Uso delle sessioni - Login

```
<%  
String name = request.getParameter("user");  
String pass = request.getParameter("pass");  
User user = User.authenticate(name, pass);  
  
if(user==null){  
%>Error<%  
}else{  
    session.setAttribute("user", user);  
%>  
Wel come
```



Uso delle sessioni – Post-login

```
<%  
User user =  
    (User)session.getAttribute("user");  
if(user==null){  
    response.sendRedirect(response  
        .encodeRedirectURL("index.html"));  
}else{  
%>  
Ciao <%=user.getName()%>
```

Recupera il riferimento all'utente dalla sessione

Rinvia alla pagina di login in caso di errore

FAQ

- Java non riconosce "servlet"
 - ◆ Aggiungere:
 - import javax.servlet.*;
 - import javax.servlet.http.*;
- Java non trova javax.servlet....
 - ◆ Includere
 - \$TOMCAT_HOME/common/lib/servlet-api.jar
- Tomcat non si lascia "uccidere"
 - ◆ Andare su **http://localhost:8080/admin**
 - Uid = admin; pwd = tomcat
 - Click su commit changes