

Tomcat

- E' uno dei servlet engine più diffusi.
- Può funzionare sia come prodotto stand-alone, sia come modulo del web server Apache.
- Disponibile gratuitamente (per molti sistemi operativi tra cui Linux e Windows) dal sito:
<http://jakarta.apache.org/> (tramite il link “Tomcat”), oppure direttamente da <http://tomcat.apache.org/>

Tomcat - funzionamento

- Rimane in attesa di richieste HTTP su una porta (default: 8080).
- Quando riceve la richiesta di una servlet, crea un'istanza della classe corrispondente.
- Inoltre ad ogni richiesta (compresa la prima) crea un thread che invoca il metodo appropriato della servlet per gestire la richiesta HTTP.
- L'istanza della classe viene eliminata in corrispondenza agli eventi di Stop, Reload, Remove, Shutdown di Tomcat.

Tomcat – Installazione in Windows

- Scaricare dalla sezione Downloads di `tomcat.apache.org` il file `.exe` dell'ultima versione di Tomcat (e.g., `apache-tomcat-5.5.12.exe`).
- Lanciare l'applicazione e seguire passo per passo il wizard.
- **N.B.:** per motivi di sicurezza, **non** installare Tomcat con l'account Administrator (o con un account con privilegi di amministratore).

Tomcat – Installazione in Linux (I)

- Scaricare dalla sezione Downloads di **tomcat.apache.org** il file **.tar.gz** dell'ultima versione (binary distribution) di Tomcat (e.g., **apache-tomcat-5.5.12.tar.gz**).
- Decomprimerlo in una directory con il comando **tar** (e.g., **tar -zxvf apache-tomcat-5.5.12.tar.gz**).
- Avviare tomcat tramite lo script **startup.sh** presente nella sottodirectory **bin** della directory principale.
- **N.B.:** per motivi di sicurezza, **non** installare Tomcat usando l'account **root**, ma creare un utente apposito (e.g., **tomcat**) e installare il software nella sua home directory.

Tomcat – Installazione in Linux (II)

- Dopo aver effettuato il login come root:

```
useradd tomcat      # crea l'utente tomcat
passwd tomcat      # cambia la password
su tomcat          # switch user
# scaricare il .tar.gz nella home di tomcat
# spostarsi nella home e decomprimere
cd; tar -zxvf apache-tomcat-5.5.12.tar.gz
# avviare Tomcat
apache-tomcat-5.5.12/bin/startup.sh
```

Tomcat – struttura delle directory

- **`$CATALINA_HOME`** (oppure **`$TOMCAT_HOME`**):
rappresenta la directory radice dell'installazione di Tomcat e contiene le seguenti directory:
 - **`bin`**
 - **`common`**
 - **`conf`**
 - **`logs`**
 - **`server`**
 - **`shared`**
 - **`temp`**
 - **`webapps`**
 - **`work`**

Tomcat – directory **bin**

- Questa directory contiene i file binari e gli script di amministrazione di Tomcat.
- Tra questi ve ne sono due fondamentali:
 - **startup.sh** (**startup.bat** in Windows): avvia Tomcat
 - **shutdown.sh** (**shutdown.bat** in Windows): arresta Tomcat
- Durante la fase di sviluppo di una web application è molto frequente dover arrestare e riavviare il servizio.

Tomcat – directory **common**

- In questa directory si trovano le classi relative a Tomcat ed alle applicazioni web pubblicate in **webapps**:

- **classes/**

- **.class**

- **lib/**

- **.jar**

- **servlet.jar** (o **servlet-api.jar**): da inserire nel classpath:

```
javac -classpath $TOMCAT_HOME/common/lib/servlet.jar <file.java>
```

- **tools.jar**

- ...

Tomcat – directory **conf**

- Questa directory contiene i file di configurazione di Tomcat.
- In particolare vi sono i seguenti file:
 - **server.xml**: parametri di configurazione generali di Tomcat (in caso di modifica è necessario riavviare il server).
 - **tomcat-users.xml**: informazioni sugli utenti di Tomcat.

Tomcat – `server.xml` (I)

- Struttura del file:
 - **<Server>** (tag principale)
 - **<Service>** (descrive la modalità di funzionamento, e.g., stand-alone)
 - **Connector** (porta TCP su cui Tomcat ascolta le richieste)
 - **Engine** (gestore delle richieste)
 - » **Host** (host virtuale)
 - » **Context** (applicazione web)
 - » **Realm**
 - » **Logger**

Tomcat – `server.xml` (II)

- Uno dei tag fondamentali è `<Context>` in quanto consente di definire il “contesto” di un’applicazione web:

`<Context`

`path="/examples"`

`docBase="examples"`

`debug="0"`

`reloadable="true">`

URL



Posizione nel file system

Tomcat – `server.xml` (III)

- Il tag `<Logger>` consente di definire le caratteristiche e la posizione dei file di log:

```
<Logger
```

```
  className="org.apache.catalina.logger.filelogger"  
  directory="logs" prefix="localhost_log."  
  suffix=".txt" timestamp="true" />
```

Tomcat – tomcat-users.xml (I)

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat"
    roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="admin" password="adminpwd"
    roles="admin,manager"/>
</tomcat-users>
```

N.B.: le password degli utenti sono inserite in chiaro!

Tomcat – directory logs

- Questa directory contiene i “registri” dell’attività di Tomcat.
- Sono presenti molti file il cui nome è definito dal tag **<Logger>** nel file di configurazione **server.xml**. Ad esempio alcuni possibili formati sono:
`localhost_log.AAAA-MM-GG.txt`
`localhost_admin_log.AAAA-MM-GG.txt`
`localhost_users_log.AAAA-MM-GG.txt`
- Cosa viene registrato?
 - eventi relativi al server Tomcat (avvio, arresto, ...);
 - eventuali errori;
 - ...

Tomcat – directory server

- Questa directory contiene:
 - le classi interne di Tomcat (in **classes/** e **lib/**);
 - due applicazioni web per facilitare la gestione di Tomcat:
 - **webapps/admin**
 - **webapps/manager**

Tomcat – directory **shared**

- Tutte le classi condivise dalle applicazioni installate su Tomcat sono contenute nelle seguenti cartelle di questa directory:
 - **classes/**
 - **lib/**

Tomcat – directory temp

- Temp è una directory di lavoro dove Tomcat memorizza dei file temporanei durante la sua attività.
- Questa directory è essenziale per il funzionamento di Tomcat e non va rimossa (anche se appare vuota).

Tomcat - directory webapps

- Directory contenente le applicazioni web:
 - scritte dagli utenti;
 - predefinite con l'installazione di Tomcat (e.g., gli esempi contenuti in **examples/**).

Tomcat – directory work

- Anche questa è una directory di lavoro.
- In particolare Tomcat la utilizza per memorizzare temporaneamente le servlet generate dalle pagine scritte con la tecnologia JSP (Java Server Pages).

Esempio – Ciao, mondo!

- Fasi dello sviluppo:
 - scrittura del codice in locale (**CiaoMondo.java**);
 - compilazione in locale (**CiaoMondo.class**) – passo opzionale;
 - scrittura del file **web.xml** (deployment descriptor file);
 - Scrittura di una pagina HTML di presentazione che contenga il link alla servlet – passo opzionale;
 - deployment e Test su un'installazione locale di tomcat – passo opzionale;
 - copia dei file sul server (latoserver.dimi.uniud.it) nelle opportune directory tramite **scp**;
 - compilazione sul server (se la compilazione è avvenuta in locale, si può provare a copiare sul server solo il file **CiaoMondo.class**, evitando questo passo);
 - Arresto e riavvio della propria applicazione web tramite il Tomcat Manager.

Il codice

```
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>");
    }
}
```

Compilazione in locale (ambiente Windows)

- Supponendo di aprire un prompt del DOS, digitare i seguenti comandi:
 - `cd <percorso della directory che contiene CiaoMondo.java>`
 - `javac -classpath "C:\Programmi\Apache Software Foundation\Tomcat 5.0\common\lib\servlet-api.jar" CiaoMondo.java`
- Viene prodotto il file **CiaoMondo.class**

Il file 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>
```

Un file `index.html` con il link alla servlet

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    <TR>
      <TD>
        <A HREF="servlet/Primo">La mia prima servlet</A>
      </TD>
    </TR>
  </TABLE>
</BODY>

</HTML>
```


Deployment in locale (I)

Ambiente Windows

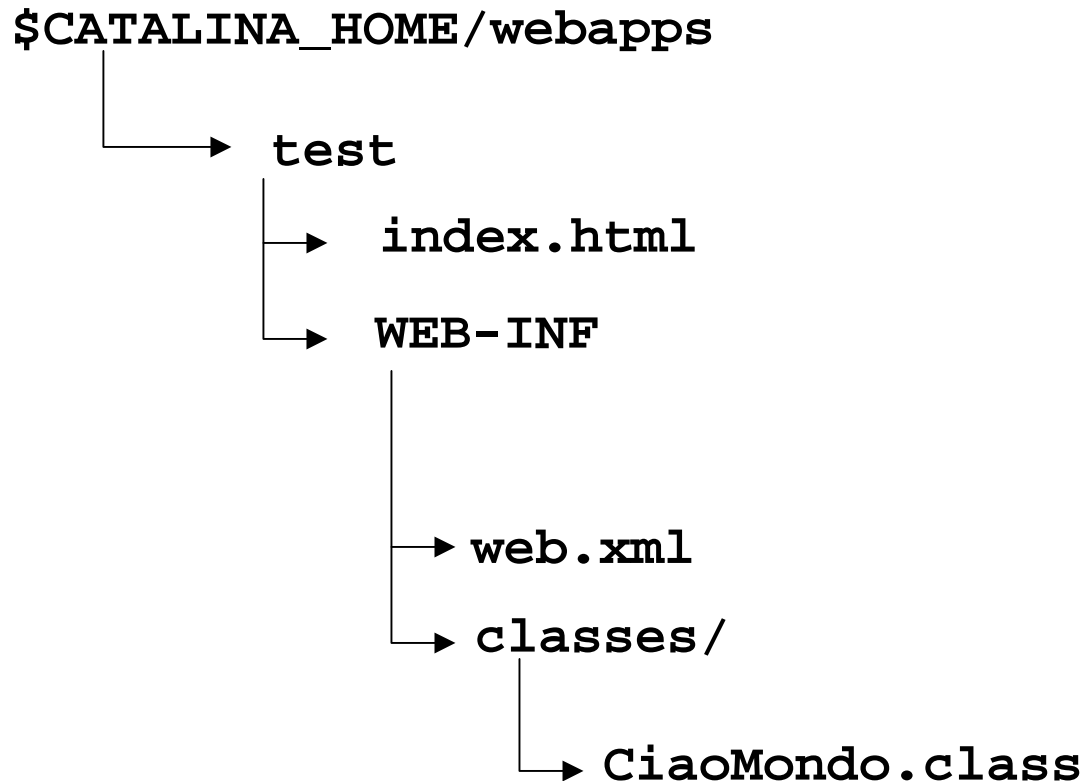
- Dove copiare i file?
 - supponiamo di aver installato Tomcat in
C:\Programmi\Apache Software Foundation\Tomcat 5.5;
 - creiamo una cartella **test** all'interno di
C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps;
 - creiamo una cartella **WEB-INF** all'interno di
C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps\test;
 - creiamo una cartella **classes** all'interno di
C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps\test\WEB-INF.

Deployment in locale (II)

Ambiente Windows

- Copiamo **index.html** (se lo abbiamo creato) in **C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps\test.**
- Copiamo **web.xml** in **C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps\test\WEB-INF;**
- Copiamo **CiaoMondo.class** in **C:\Programmi\Apache Software Foundation\Tomcat 5.5\webapps\test\WEB-INF\classes;**

Schema della struttura delle directory create



Deployment in locale (III)

Ambiente Windows

- Arrestiamo e riavviamo il servizio (tramite gli script **shutdown.bat** e **startup.bat**, oppure l'apposita applicazione nella tray-bar di Windows).
- Testiamo la nostra applicazione:
 - <http://localhost:8080/test/> (facciamo click sul link nella pagina che compare sul browser);
 - <http://localhost:8080/test/servlet/Primo> (test diretto senza passare dalla pagina HTML).

Deployment remoto (tramite **scp**) - I

- Sintassi di **scp**:

```
- $ man scp
```

```
...
```

```
NAME
```

```
scp - secure copy (remote file copy program)
```

```
SYNOPSIS
```

```
scp [-pqrVBC46] [-F ssh_config] [-S program] [-P port]  
    [-c cipher] [-i identity file] [-o ssh_option]  
    [[user@]host1]:file1 [...] [[user@]host2]:file2
```

```
...
```

```
The options are as follows:
```

```
...
```

```
-r Recursively copies entire directories
```

Deployment remoto (tramite `scp`) - II

- `scp index.html <nome utente>@latoserver.dimi.uniud.it:/home/<nome utente>/servlets/`
- `scp web.xml <nome utente>@latoserver.dimi.uniud.it:/home/<nome utente>/servlets/WEB-INF/`
- `scp CiaoMondo.class <nome utente>@latoserver.dimi.uniud.it:/home/<nome utente>/servlets/WEB-INF/classes/`

Deployment remoto (tramite client grafico)

- Per i sistemi operativi della famiglia Windows esistono dei client grafici per le operazioni di copia remota.
- Un client grafico gratuito è disponibile all'indirizzo <http://www.coreftp.com>
- Le operazioni di copia sono notevolmente semplificate dall'interfaccia grafica (supporta anche la copia sicura, come **scp**).

Deployment remoto su `latoserver.dimi.uniud.it`

- Copiare i file su `latoserver.dimi.uniud.it` (tramite `scp` o `CoreFTP`) nelle seguenti directory:
 - `index.html` in `~/servlets/`
 - `web.xml` in `~/servlets/WEB-INF`
 - `CiaoMondo.class` in `~/servlets/WEB-INF/classes/`
- Se non si dispone del file `.class`:
 - copiare `CiaoMondo.java` nella propria home;
 - Compilarlo con il comando

```
javac -classpath /home/tomcat/jakarta-tomcat-4.1.27/common/lib/servlet.jar ~/CiaoMondo.java
```
 - Copiare il file compilato in `~/servlets/WEB-INF/classes/`

Deployment remoto su `latoserver.dimi.uniud.it`

- Collegarsi a `http://latoserver.dimi.uniud.it:8080/`
- Fra i link a sinistra della home page di Tomcat cliccare su “Tomcat Manager”
- Inserire lo username tomcat e la password comunicata a lezione
- Compare una pagina web che elenca tutte le applicazioni registrate su Tomcat
- Scrollare la pagina fino a individuare la propria (`/users/<nome utente>`)
- Cliccare prima su Stop, poi su Start (**non cliccare su Remove**)
- Comportatevi bene: non arrestate, avviate o rimuovete le applicazioni degli altri utenti.

Test finale

- Collegatevi a

http://latoserver.dimi.uniud.it:8080/users/<nome_utente>/

e cliccate sul link per testare la servlet

- Oppure collegatevi a

http://latoserver.dimi.uniud.it:8080/users/<nome_utente>/servlet/Primo

per testare la servlet direttamente senza passare dalla pagina **index.html**.

Un esempio più complesso

- Supponiamo di voler scrivere una variante dell'applicazione CiaoMondo che stampi un messaggio personalizzato in base al contenuto di un campo di un form.
- Più precisamente scriveremo una pagina HTML contenente un form che chiederà all'utente di inserire il proprio nome.
- Alla pressione del pulsante di invio del form, verrà chiamata una servlet che leggerà il valore inserito dall'utente e genererà una pagina personalizzata.

Il form: nome.html

```
<HTML>
<HEAD>
<TITLE>Nome</TITLE>
</HEAD>
<BODY>
  <FORM METHOD="post" ACTION="servlet/Secondo">
    Qual &egrave; il tuo nome?
    <INPUT TYPE="text" NAME="nome">
    <P>
    <INPUT TYPE="submit" VALUE="Invia &gt;&gt;">
  </FORM>
</BODY>
</HTML>
```

La nuova servlet: CiaoPersonalizzato.java

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

public class CiaoPersonalizzato extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        String nome=req.getParameter("nome");
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Ciao, "+
            (nome.length()==0 ? "sconosciuto" : nome)+"!</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<STRONG>Ciao, "+
            (nome.length()==0 ? "sconosciuto" : nome)+"!</STRONG>");
        out.println("</BODY></HTML>");
    }
}
```

Modifiche da apportare a 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>SecondoTest</servlet-name>
```

```
        <servlet-class>CiaoPersonalizzato</servlet-class>
```

```
    </servlet>
```

```
    <servlet-mapping>
```

```
        <servlet-name>SecondoTest</servlet-name>
```

```
        <url-pattern>/servlet/Secondo</url-pattern>
```

```
    </servlet-mapping>
```

```
</web-app>
```

Modifiche da apportare a `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>
<BODY>
  <TABLE>
    <TR>
      <TD>
        <A HREF="servlet/Primo">La mia prima servlet</A>
      </TD>
    </TR>
    <TR>
      <TD>
        <A HREF="nome.html">Una servlet un po' più complessa</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Esercizi

- Eseguire il deployment ed il test di **CiaoPersonalizzato** (aggiornando anche i file **web.xml** e **index.html**) su **latoserver.dimi.uniud.it**
- **CiaoPersonalizzato** può essere richiamata direttamente (senza passare dal form) tramite l'URL
http://latoserver.dimi.uniud.it:8080/users/<nome utente>/servlet/Secondo?
Perché?
- Apportare le dovute modifiche a **CiaoPersonalizzato.java** affinché funzioni anche con una richiesta diretta (senza passare dal form).
- Scrivere un'applicazione web che generi una pagina HTML con la tabellina Pitagorica.