

---

# Programmazione server-side: Java Servlet

Corso di ***Applicazioni Telematiche***

A.A. 2008-09 – Lezione n.12

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II

Facoltà di Ingegneria

---

---

# Cos'è una Servlet?

Una *Servlet* è un componente software  
scritto in Java,  
gestito da un “container”,  
che produce contenuto web dinamico

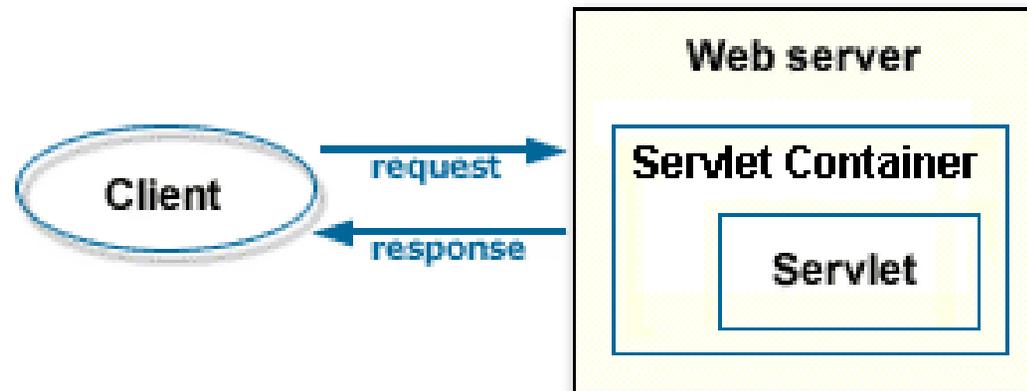
*(Java Servlet Specification, v. 2.4)*

---

---

## Cos'è una Servlet? (2)

Una Servlet interagisce con un web Client attraverso il paradigma di comunicazione request/response



---

# Servlet Container

- Il servlet container (o *servlet engine*) è un'estensione di un web server che fornisce l'ambiente di esecuzione ad una Servlet
  - Esempio di servlet container open-source:
    - Tomcat
      - Tomcat è un servlet container open-source interamente scritto in Java disponibile su:  
<http://tomcat.apache.org/>
-

---

# Installazione e configurazione di Tomcat

- Scaricare l'installer di Apache Tomcat per la propria piattaforma (Win32 o Linux)
    - Versione attuale (aprile 2006): 5.5.x
  - Installare Tomcat
    - Per convenzione indicheremo con "install\_dir" la directory base della installazione (es. *C:\wamp\Tomcat\Tomcat5.5*)
  - Installare le applicazioni "manager" ed "admin" per la gestione delle applicazioni web e del server stesso
    - vedi "Manager App HOW-TO" sul sito ufficiale
  - Abilitare la funzione di ri-caricamento automatico
    - Trasformare `<Context>` in `<Context reloadable="true">` in *install\_dir/conf/**context.xml*** (per tutte le applicazioni web) o nel file "Context XML descriptor" della specifica applicazione
-

---

## Installazione e configurazione di Tomcat (2)

- Abilitare l'invocazione delle servlet senza deployment descriptor (opzionale)
    - In *install\_dir/conf/web.xml* togliere il commento agli elementi *servlet* and *servlet-mapping* in modo da abilitare l'invoker servlet su un qualunque path del tipo */servlet/\**
    - Grazie a questa modifica è possibile copiare il bytecode .class della servlet in *WEB-INF/classes* e successivamente invocare la servlet tramite la URL *http://host/servlet/ServletName*
    - Utile per un rapido debugging delle servlet prima del deployment
  - In alternativa, se non si sceglie la strada indicata sopra, si devono sempre inserire le servlet sviluppate in un contesto di web application, opportunamente descritte nel deployment descriptor
    - Nel seguito inseriremo le servlet d'esempio nel Context */AT\_servlet* descritto in *install\_dir/conf/Catalina/localhost/AT\_servlet.xml*
-

---

# Context XML descriptor

- Il Context XML descriptor è un documento XML che contiene la descrizione del Context associato ad una web application
  - In particolare, sono specificati:
    - La base directory della applicazione
    - Il path anteposto al nome delle servlet della applicazione per identificarle univocamente
    - Se l'applicazione deve essere "reloadable" o non
  - La descrizione del Context di una applicazione è tipicamente memorizzata in un file XML  
*install\_dir/conf/[enginename]/[hostname]/[nome\_applicazione].xml*
  - Es: *install\_dir/conf/Catalina/localhost/AT\_servlets.xml*
-

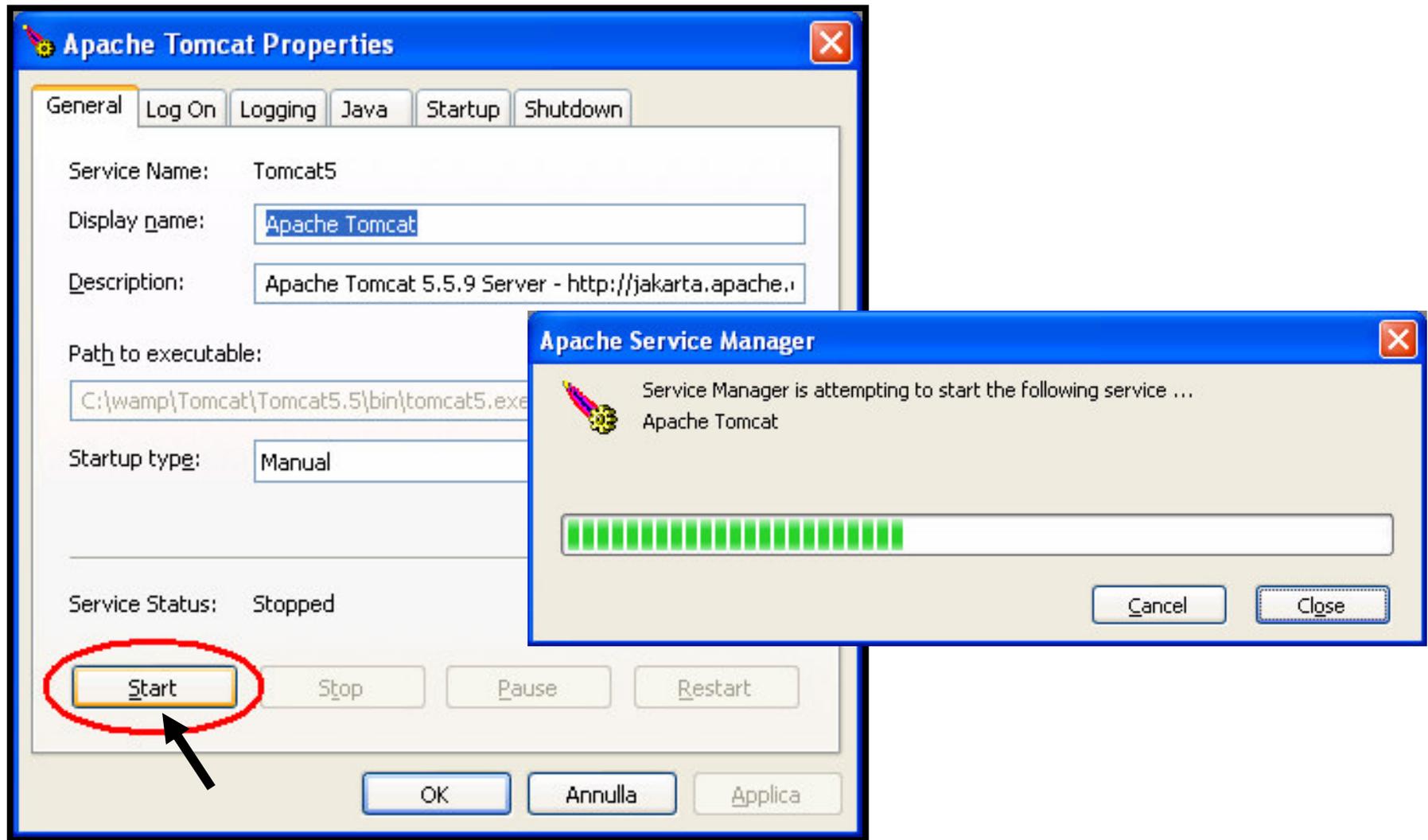
---

# AT\_servlets.xml Context

```
<?xml version="1.0" encoding="UTF-8"?>  
<Context docBase="C:\java\workspace\AT_servlets"  
  path="/AT_servlets" reloadable="true"/>
```



# Avvio di Tomcat





Apache Tomcat/5.5.9

The **Apache Jakarta Project**<http://jakarta.apache.org/>

### Administration

[Status](#)[Tomcat Administration](#)[Tomcat Manager](#)

### Documentation

[Release Notes](#)[Change Log](#)[Tomcat Documentation](#)

### Tomcat Online

[Home Page](#)[Bug Database](#)[Open Bugs](#)[Users Mailing List](#)[Developers Mailing List](#)[IRC](#)

### Examples

**If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!**

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.jsp
```

where "\$CATALINA\_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the [Tomcat Documentation](#) for more detailed setup and administration information than is found in the INSTALL file.

**NOTE:** This page is precompiled. If you change it, this page will not change since it was compiled into a servlet at build time. (See `$CATALINA_HOME/webapps/ROOT/WEB-INF/web.xml` as to how it was mapped.)

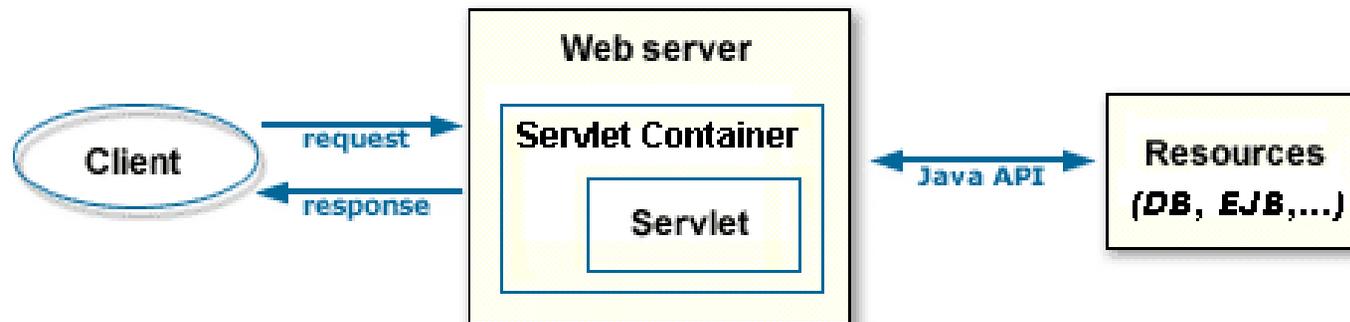
**NOTE:** For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation (including the Servlet 2.4 and JSP 2.0 API JavaDoc), and an introductory guide to developing web applications.

---

# Interazione client-servlet

1. Il Client fa una richiesta HTTP al web server
2. Il web server carica la servlet (solo la prima volta) e crea un thread per eseguirla
3. Il container esegue la servlet richiesta
4. La servlet genera la risposta
5. La risposta viene restituita al client



---

# Vantaggi delle servlet

- Sono molto più veloci dei CGI
  - Persistenti
    - una volta caricata, una servlet rimane in memoria e può ottimizzare l'accesso alle risorse attraverso caching, pooling, etc...
  - *Implementation independence*
    - usano una API standard supportata da molti web server
  - Vantaggi offerti dal linguaggio Java
    - *Platform independence, Object Oriented programming, Garbage Collection, ...*
-

---

# Servlet API

- E' un framework di classi Java che offre delle interfacce object oriented ad oggetti che incapsulano la comunicazione tra client e server (request, response)
  - I package `javax.servlet` e `javax.servlet.http` definiscono interfacce e classi base da cui un programmatore può derivare le proprie specifiche servlet
  - Una servlet deve implementare l'interfaccia `Servlet`
  - Una servlet può essere fatta derivare dalla classe `GenericServlet` o da `HttpServlet`, che implementa i metodi base (es. `doGet`, `doPost`, ecc...) per gestire l'interazione con un client via HTTP
-

---

# Servlet Lifecycle

## 1. Load & instantiation:

```
Servlet MyServlet = new HttpServlet();
```

## 2. Initialization:

```
MyServlet.init(ServletConfig);
```

## 3. Request Handling:

```
MyServlet.service(request, response);
```

## 4. End of service:

```
MyServlet.destroy();
```

---

---

## Servlet: init()

- Il metodo init() viene eseguito una volta soltanto per ciascuna servlet
  - Permette di accedere a risorse utili per la servlet
    - aprire connessioni ad un db, ottenere reference ad EJB, ecc...
-

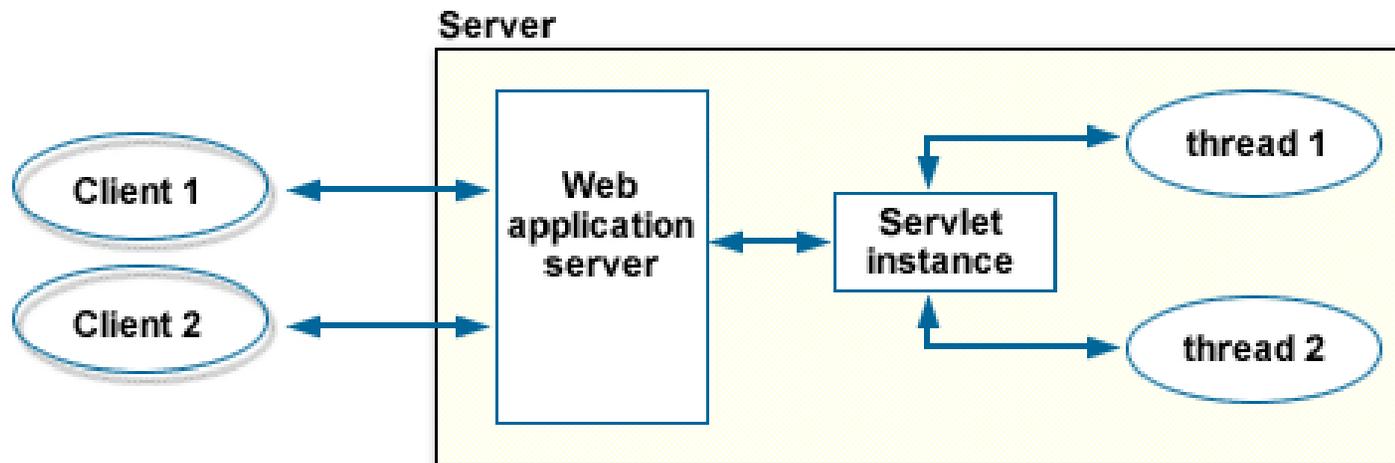
---

## Servlet: `service()`, `doGet()` e `doPost()`

- I metodi `service()`, `doGet()` e `doPost()` possono essere implementati dal programmatore
  - Il metodo `service()` per default delega l'esecuzione al metodo indicato nella richiesta HTTP del client
  - Le richieste HTTP più comuni sono POST e GET, gestite dai metodi della servlet `doPost()` e `doGet()` implementate dal programmatore
-

# Servlet Lifecycle Multithreading

- Viene creato un Thread per ogni richiesta
- Il thread può essere riutilizzato se lo stesso client richiede la stessa servlet



# Una servlet con un parametro di input

```
package it.unina.at;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Servlet01 extends HttpServlet {
    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        String nome = (String)request.getParameter("nome");
        HttpSession session = request.getSession(true);
        session.setAttribute("nome", nome);
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<p>Benvenuto, " + nome + "</p>");
        out.println("<p>La data di oggi è:");
        out.println("<b>" + new java.util.Date() + "</b></p>");
        out.println("</body></html>");
    }
}
```

---

# Struttura di una applicazione web

- Un'applicazione web è costituita da un insieme di pagine statiche, di servlet e di pagine JSP
  - In Tomcat un'applicazione web è associata ad un "Context"
  - La struttura di una applicazione web eseguibile in Tomcat è la seguente:
    - Nella root directory della applicazione:
      - \*.html, \*.jsp
    - Nella sotto-directory WEB-INF:
      - web.xml – il Deployment Descriptor
      - In WEB-INF/classes i file \*.class delle servlet (bytecode)
      - In WEB-INF/lib eventuali file di libreria \*.jar
-

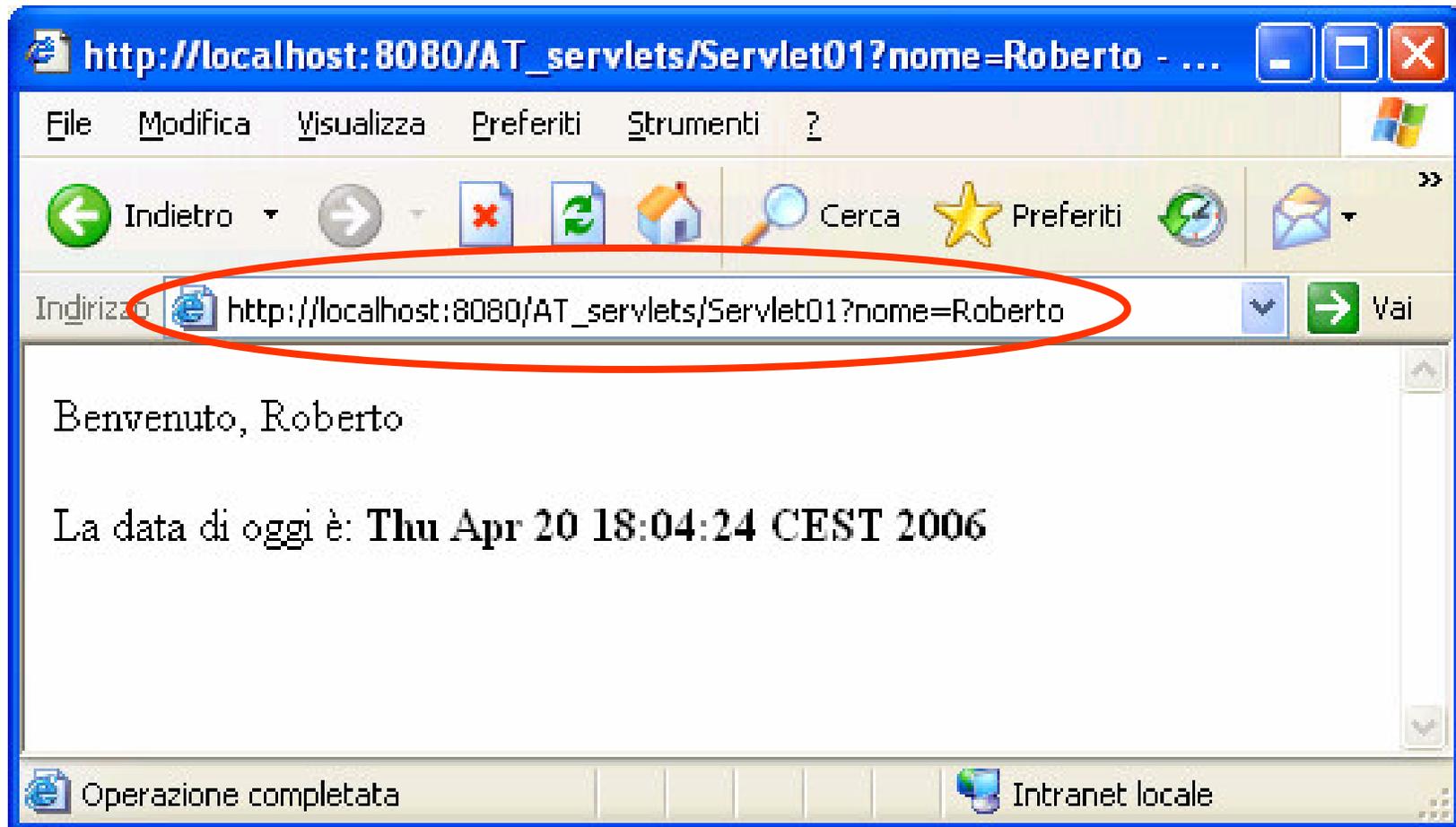
---

# Deployment descriptor web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>Servlet01</servlet-name>
    <servlet-class>it.unina.at.Servlet01</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Servlet01</servlet-name>
    <url-pattern>/Servlet01</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>
      index.jsp
    </welcome-file>
  </welcome-file-list>
</web-app>
```

---

# Prova della servlet



---

# Servlet API: HttpRequest interface

Permette di:

- Ottenere i parametri inviati dal client
  - Ottenere il riferimento alla sessione utente
  - Ottenere il flusso dei dati inviati dal client
  - Riconoscere l'utente autenticato
-

---

# Servlet API: HttpServletResponse interface

Permette di:

- Inviare dati al client come HTML oppure come flusso binario
  - Inviare codici di errore e codici di controllo nell'intestazione della response HTTP per controllare il comportamento del browser
-

---

Domande?

