Web application con Java ed Apache Tomcat

Programmazione 3 e Laboratorio

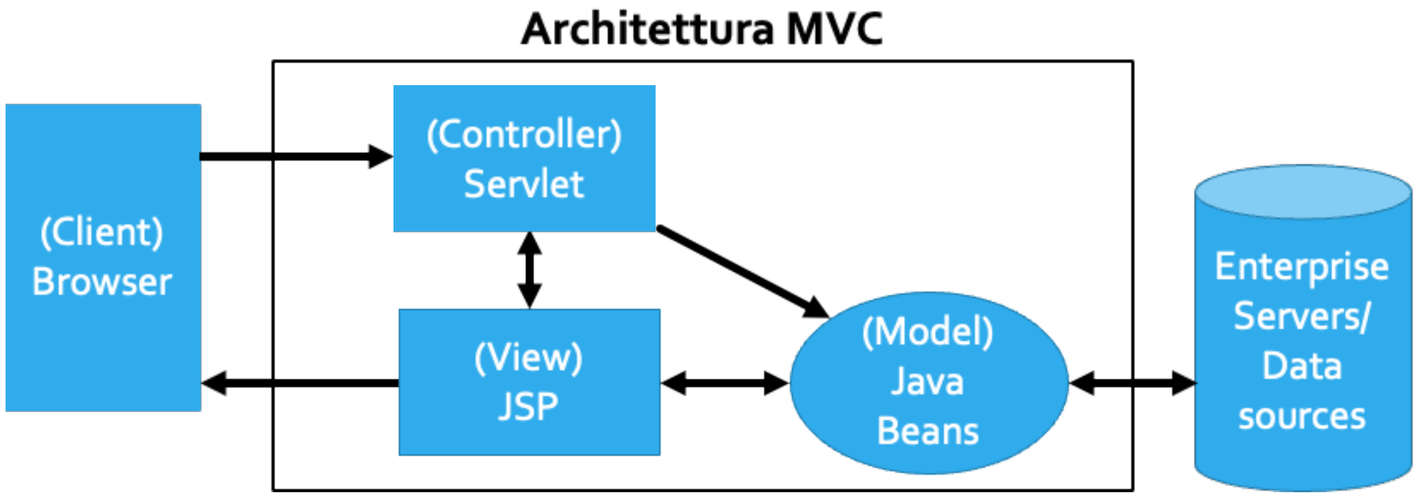
Proff. Angelo Ciaramella - Emanuel Di Nardo

Sommario

- Apache Tomcat
  - Componenti di Apache Tomcat
- Web Application
  - Componenti di una Web Application
- Configurazione ambiente IntelliJ
  - Server Apache
  - Database
- Esempio di Registrazione e Login
- Tomcat Web Application Manager

Apache Tomcat

- Apache Tomcat è un *Application Server* basato su Java Servlet
- Permette di creare applicativi Client-Server
- Tomcat implementa diverse specifiche:
  - Java Servlet
  - JavaServer Pages (JSP)



Componenti di Tomcat

- **Catalina**
  - Contenitore di *Servlet* Java
  - Fornisce l’effettiva implementazione di Tomcat e delle specifiche della Servlet
- **Coyote**
  - Supporta il protocollo HTTP 1.1
  - Processa le richieste effettuare al web server
- **Jasper**
  - Processa il codice Java/templating all’interno delle JSP e le invia a Catalina
  - Cerca cambiamenti avvenuti nei file JSP e li ricompila

Web Components

- I *Web Components* sono in grado di accedere ai servizi offerti da un *web container come:*
  - Smistamento delle richieste
  - Sicurezza
  - Concorrenza
  - Gestione della memoria

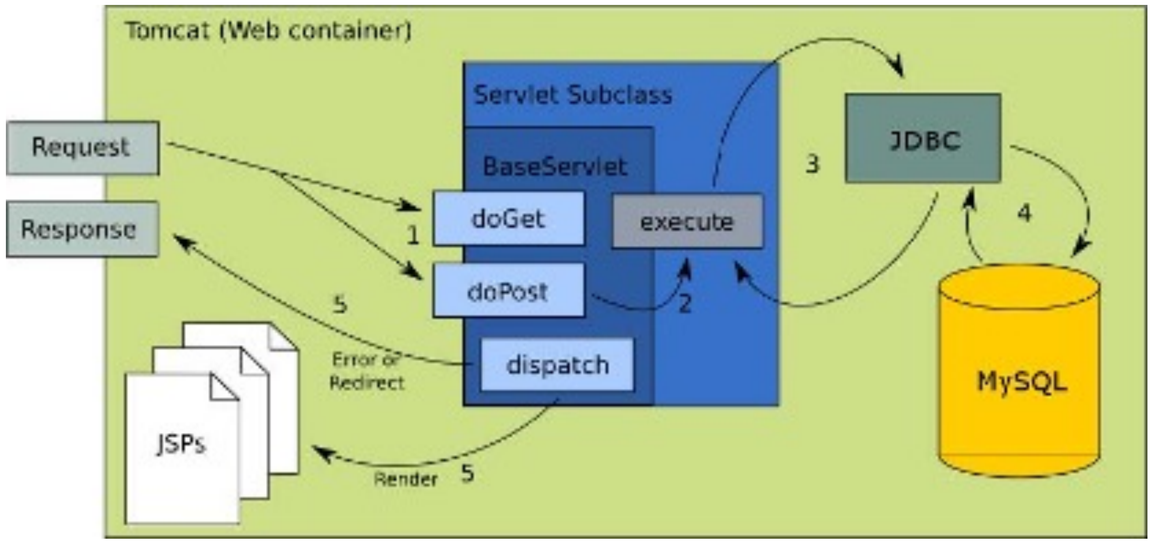
Java Servlet (Controller - Backend)

- Le funzionalità principali sono:
  - Elaborazione o memorizzazione di dati provenienti da form HTML
  - Generazione di contenuti dinamici (pagine Web) a seconda dei parametri della richiesta

- Comunicazione Client-Server con protocollo HTTP

Java Server Page (View - Frontend)

- JSP è una tecnologia web per lo sviluppo di WA
- Fornisce contenuti dinamici in formato HTML o XML
- Composto da un insieme di *tag* con cui possono essere invocate funzioni e/o codice java
- Rappresentazione ad alto livello di una Servlet



JavaBeans (Model)

- Classe che incapsula molti oggetti in un singolo oggetto
- Deve essere serializzabile
- Ha un costruttore e metodi getter, setter
- Rappresentazione POJO di entità (tabelle) presenti sul database

Vantaggi	Svantaggi
Controllo delle proprieta', eventi e metodi dei beans esposti ad altre applicazioni.	E' soggetto ad essere instanziato con un stato invalido avendo un costruttore nullo.
Registra eventi da altri oggetti e puo' generare eventi da poter inviare ad altri oggetti.	Sono oggetti intrinsecamente mutabili mancando cosi del vantaggio offerto dagli oggetti immutabili.
Impostazioni di configurazione di un bean possono essere memorizzati in modo persistente.	Il possedere molti metodi getter e setter puo' portare una quantita' immensa di boilerplate code.

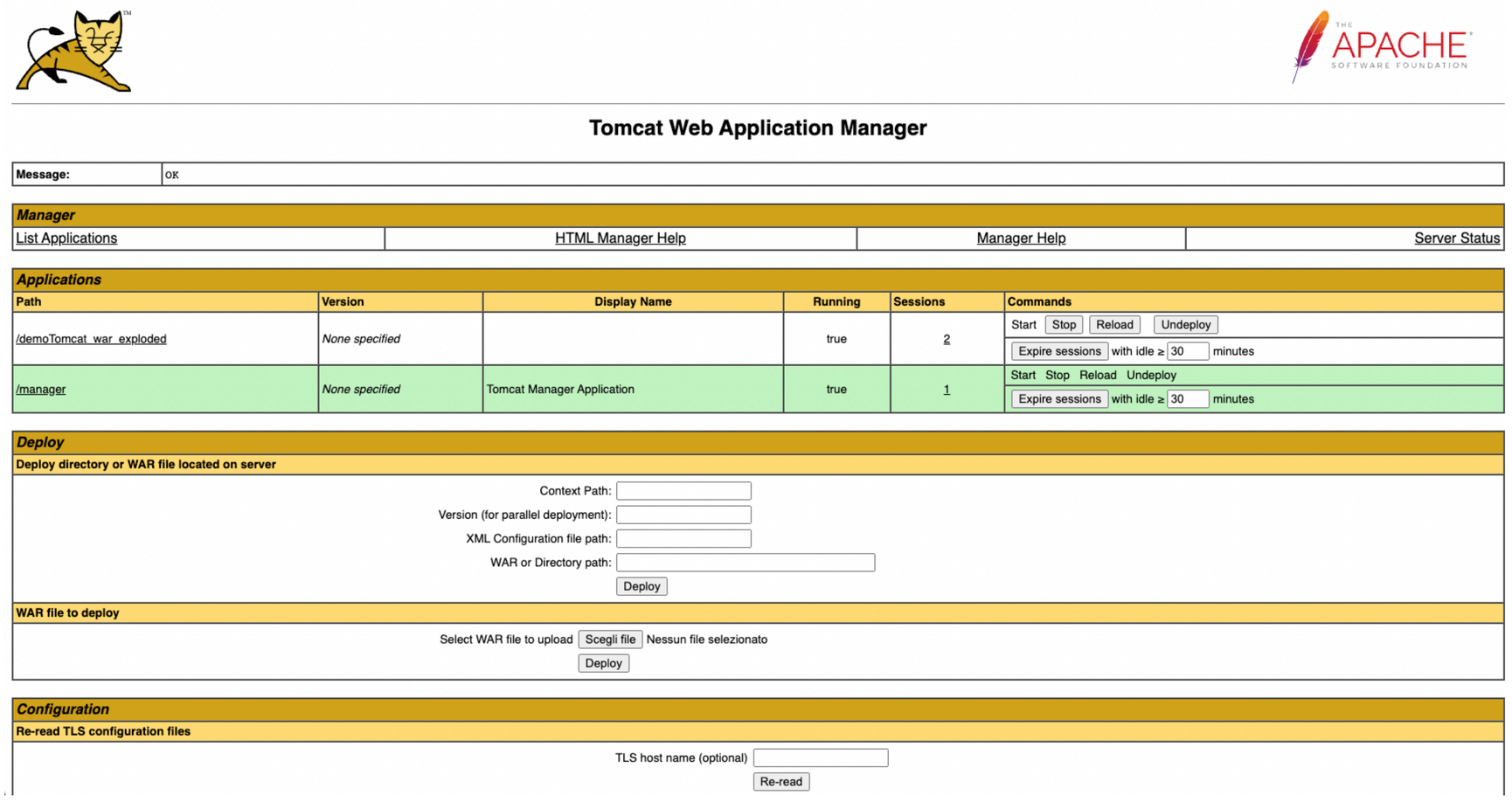
Tomcat

- Download: https://tomcat.apache.org/index.html

- Tomcat era parte di JavaEE. Il progetto JavaEE è ora chiamato JakartaEE
  - Di conseguenza tutte le librerie ora saranno contenute nei package jakarta

Tomcat Manager

- Nel file di configurazione tomcat-users.xml inserire:
  - <user username="admin" password="admin" roles="manager-script,admin-gui,manager-gui"



Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/demoTomcat_war_exploded	None specified		true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path:
Version (for parallel deployment):
XML Configuration file path:
WAR or Directory path:
Deploy

WAR file to deploy

Select WAR file to upload Scegli file Nessun file selezionato
Deploy

Configuration

Re-read TLS configuration files

TLS host name (optional)
Re-read

Directory webapp

- Contiene **WEB-INF**: configurazione dell'applicazione
  - web.xml: descrittore della web application
    - Descrive le servlets e gli altri componenti
      - Tag <*servlet*> per dare un nome alla servlet e assegnarle una classe
      - Tag <*servlet-mapping*> definisce un url per raggiungere la servlet
    - Da Java 6 EE è possibile configurare le servlet tramite *annotation*
  - lib: vanno inserite le librerie di jakarta.servlet

- Contiene tutte le jsp ed i file html
- La prima pagina eseguita è quella chiamata *index*

Servlet

- Ogni Servlet è una classe java
  - Si consiglia di inserire la parola Servlet alla fine del nome della classe
    - es. *HelloServlet.java*
- Tre modi per creare una servlet:
  - Implementare l'interfaccia **Servlet**
  - Estendere la classe **GenericServlet**
  - Estendere la classe **HttpServlet** (preferibile)
- Gestiscono i metodi HTTP (GET, POST, PUT, DELETE) tramite metodi java:
  - GET -> doGet
  - POST -> do Post

Servlet

- Ogni metodo http ha due argomenti di input:
  - **HttpServletRequest**: responsabile di tutte le richieste effettuate alla pagina. Contiene i parametri di input della richiesta (request.getParameter)
  - **HttpServletResponse**: processa la risposta alla richiesta HTTP
- Per rendere la Servlet raggiungibile dall'esterno è necessario utilizzare l'annotation **@WebServlet**

```
@WebServlet(name = "helloServlet", value = "/hello-servlet")
```

  - Due parametri di input:
    - *name*: nome della servlet
    - Value: url-mapping

Servlet

- Per essere reindirizzati verso un’altra pagina utilizziamo il RequestDispatcher()
  - Lo troviamo già inizializzato in request. Prende come argomento un url oppure un file html
  - E’ eseguito dalla forward

```
request.getRequestDispatcher("hello_friend.html").forward(request, response);
```

JSP

- Tramite le JSP possiamo creare pagine web che a cui è possibile passare parametri tramite il backend ed eseguire codice java al suo interno.
- Diverse tipologie di tag:
  - Espressioni <%= %>: Permettono di inserire variabili

```
<%= request.getAttribute("firstname") %>
```

  - Scriptlets <% %>: Permette di inserire grandi porzioni di codice java

```
<% if (!request.getAttribute("firstname").equals("Emanuel")) { %>
    <p>Per favore scrivi il nome corretto</p>
<% } else { %>
    Firstname: <%= request.getAttribute("firstname") %><br />
    Lastname: <%= request.getAttribute("lastname") %>
<% } %>
```

JSP

- Può sembrare interessante utilizzare codice Java per renderizzare le pagine web
- E’ assolutamente **SCONSIGLIATO** (da Oracle) utilizzare gli *scriptlets*!
  - **Riusabilità**: non si possono riutilizzare
  - **Sostituibilità**: non possono essere dichiarati astratti
  - **OO-Paradigm:** non è possibile utilizzare l’ereditarietà
  - **Testabilità**: non possono essere testati tramite unit-test
  - **Manutenibilità**: difficile manutenere

https://www.oracle.com/technical-resources/articles/javase/code-convention.html

JSP

- Processare tutti gli elementi lato server, in modo da inviare al frontend le elaborazioni terminate.
- Per elaborazioni on-page utilizzare i metodi classici del pagine web dinamiche (Javascript con Ajax)
- Utilizzare librerie alternative per il *templating:*
  - JSP Standard Tag Library (JSTL)
  - Expression Language (EL)

JSTL e EL

- Aggiungiamo la libreria JSTL
  - Tramite maven:

```
<dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>2.0.0</version>
</dependency>
```

- Per Tomcat 10.0.x utilizzare la versione 2.0.0
- Per Tomcat 10.1+ utilizzare la versione 3.0.0

  - Aggiungere alle pagine jsp il seguente taglib:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

- Fino alla versione 2.0.0 utilizzare l'URI specificato
- Dalla versione 3.0.0 l'URI diventa:
  - jakarta.tags.core

- Per altre funzionalità inserire gli altri taglib a disposizione:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
```

JSTL e EL

- Leggere dalle variabili:
  - <c:out value="${var_name}" />
  - ${var_name} (scorciatoia)
- Settare variabili:
  - <c:set var="nome_variabile" value="valore">
- Control Flow:
  - <c:if> Condizione, non supporta condizioni multiple
  - <c:choose> Supporta condizioni multiple
    - Ogni condizione è verificata da <c:when>, l'ultima condizione può essere <c:otherwise>
- For, while do-while:
  - <c:forEach>

JSTL e EL

- I tag vanno sempre chiusi!
- Come in html presentano due tipi di chiusure:
  - Se non è previsto un tag di chiusura (elemento singolo) si usa /> : <c:out />
  - Se è previsto un tag di chiusura (raggruppa più elementi) si utilizza </tag>: </c:choose>

HTML - CSS - JS

- E’ possibile utilizzare qualunque framework per lo styling delle pagine:
  - Bootstrap: https://getbootstrap.com/
  - Foundation: https://get.foundation/
  - React: https://it.reactjs.org/
- Di conseguenza è possibile utilizzare qualunque libreria javascript di nostro interesse
  - jQuery: https://jquery.com/
  - Datatables: https://datatables.net/
  - ...

Esercizio

- Utilizzando il database BankAccount costruire
  - una semplice pagina web che permetta di creare un nuovo conto corrente sul database
    - Nome
    - Cognome
    - Numero di conto
    - Bilancio
  - una pagina che permette di vedere le informazioni del correntista