

Programmazione WEB server-side JSP

TECNICO SUPERIORE PER LO
SVILUPPO SOFTWARE
2013

Il container JSP

Il container JSP gestisce automaticamente ognuna di queste forme sulla **base** della data in cui è avvenuta l'ultima modifica a ciascun file.

In risposta ad una richiesta HTTP , il container controlla se il codice sorgente .jsp è stato modificato dall'ultima compilazione del codice sorgente .java. In caso affermativo il container ritraduce il codice sorgente JSP per produrre la nuova versione del codice sorgente Java.

Il container JSP

In realtà il container (nel nostro caso Tomcat 7) ha eseguito svariate operazioni in seguito alla richiesta del browser:

- ha convertito la pagina JSP in una Servlet (classe Java)
- ha compilato la classe in bytecode
- ha istanziato la classe nel servlet container
- ha esaudito la richiesta restituendo la risposta al browser.

Tutto questo accade **solo alla prima richiesta della pagina**, perché alle successive richieste Tomcat avrà già la pagina compilata in “memoria”, ottenendo quindi un notevole miglioramento delle prestazioni.

Le pagine JSP infatti sono molto performanti rispetto alle altre tecnologie (ASP e PHP) che ad ogni richiesta interpretano il codice lato server senza compilarlo.

Il container JSP

Risulta evidente di come sia molto più semplice scrivere una JSP che una Servlet. Esaminando meglio il [codice generato](#), notiamo come le istruzioni Java che noi abbiamo inserito e le istruzioni di scrittura dei tag HTML sono stati copiati nel metodo `_jspService()`. Nel corso delle successive richieste il *container* avrà già in memoria la servlet creata dalla pagina JSP e richiamerà sempre il metodo `_jspService()` sull'istanza conservata.

In conclusione, alla base di tutto ci sono comunque le classi Java, solo che anche un web designer senza troppa esperienza in programmazione Java, potrà realizzare le proprie applicazioni con uno sforzo minimo perché il *container* farà tutto il lavoro di conversione.

A questo punto si potrebbe pensare di realizzare la nostra applicazione solo con pagine JSP, potrebbe essere una soluzione valida, ma è nella realizzazione delle Servlet e dei componenti Bean che risiedono le maggiori differenze tra le Java Web Application e le altre tecnologie *tradizionali* come ASP o PHP.

Preparazione dell'ambiente

In questa lezione vedremo come preparare la nostra macchina all'esecuzione delle pagine JSP.

Le soluzioni che verranno proposte non sono le uniche possibili. La piattaforma J2EE è a specifiche aperte, di conseguenza sono disponibili sul mercato innumerevoli ambienti di sviluppo sia commerciali che gratuiti.

Nella scelta degli strumenti mi sono orientato su prodotti gratuiti e open source però allo stesso tempo con potenzialità notevoli, paragonabili ad ambienti commerciali:

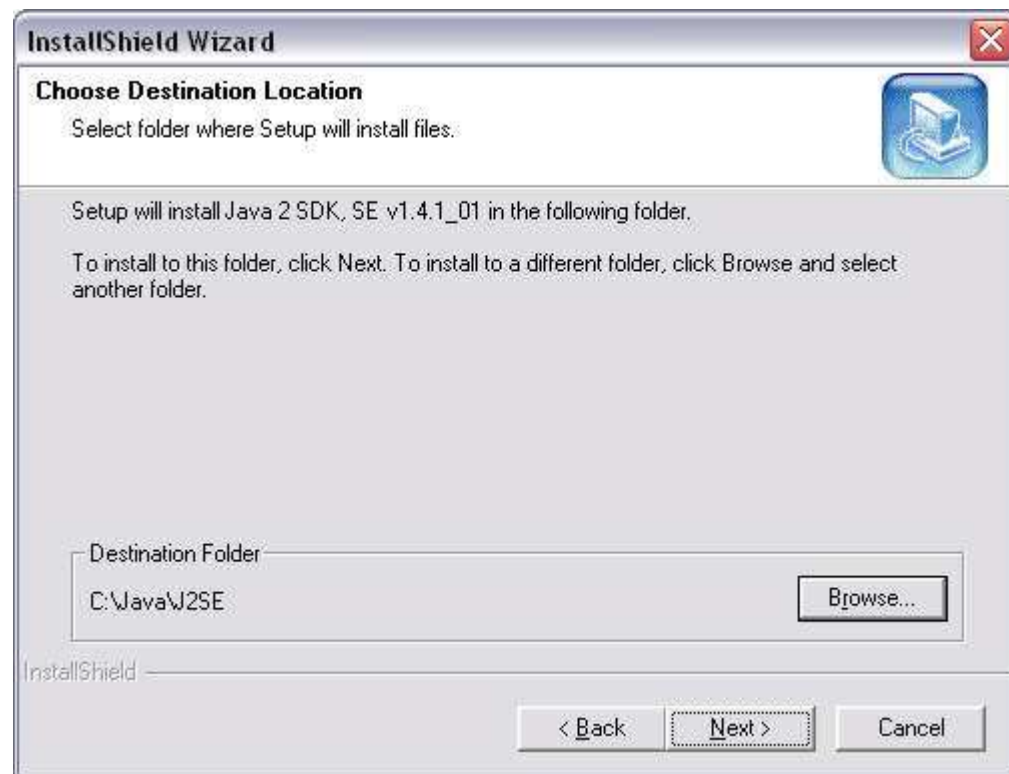
J2SE: sono indispensabili per la compilazione e l'esecuzione delle classi Java

- **Apache Tomcat:** un Application Server molto valido che integra un servlet container ed un Web Server, volendo lo si può integrare anche in Apache e IIS di Microsoft.

- **MySQL:** un ottimo server di database molto prestante, gratuito per l'utilizzo sul Web (le procedure per la sua installazione saranno descritte nelle lezioni successive).

Installazione di J2SE

Dopo aver scaricato il pacchetto di Java 2 Standard Edition dal sito della Sun Microsystem (prelevate anche i file di help utili per i riferimenti al linguaggio Java), fare doppio click per eseguire l'installazione. Selezionare il percorso in cui installarlo (personalizzatelo come da figura C:\Java\J2SE):

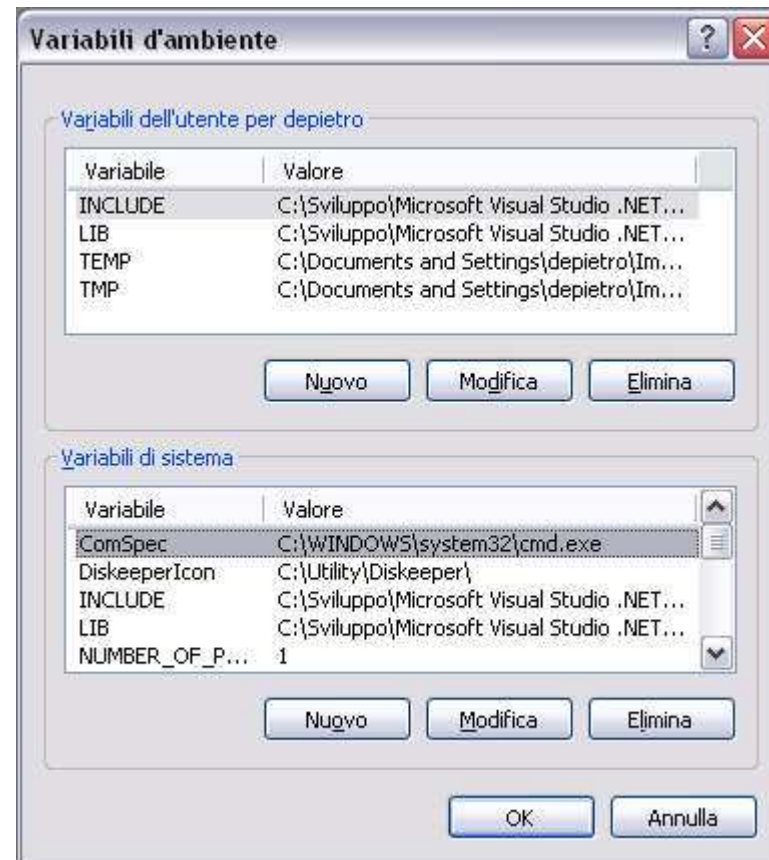


Andate avanti con l'installazione selezionando tutti i componenti da installare.

A questo punto bisognerà impostare le variabili d'ambiente.

Andare su **Pannello di Controllo | Sistema** selezionare la tabella **Avanzate** premere il pulsante

Variabili d'ambiente:



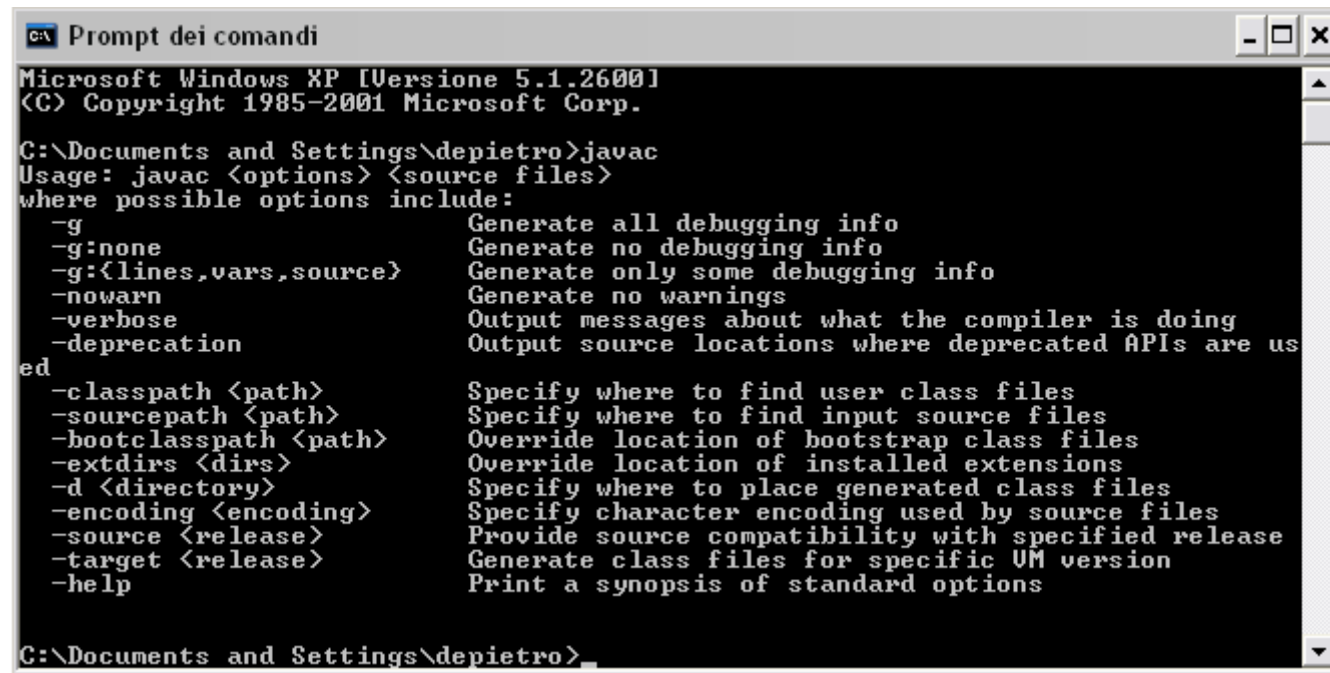
Premere il pulsante **Nuovo** nel frame delle **Variabili di sistema**:

È importante inserire nella casella di testo **Valore variabile** il percorso completo della cartella in cui si è installato lo JDK.

Poi bisogna selezionare **path** delle variabili di sistema e premere il pulsante **Modifica**:



Aggiungere alla fine del **Valore variabile** la stringa evidenziata in figura (compreso il punto e virgola). Premere OK ed il gioco è fatto. Per testare che il tutto sia configurato a dovere, aprire una finestra DOS dagli accessori e digitare il comando **javac**, dovrebbe presentarsi qualcosa di simile:



```
C:\Documents and Settings\depietro>javac
Usage: javac <options> <source files>
where possible options include:
    -g                Generate all debugging info
    -g:none           Generate no debugging info
    -g:{lines,vars,source}  Generate only some debugging info
    -nowarn           Generate no warnings
    -verbose          Output messages about what the compiler is doing
    -deprecation      Output source locations where deprecated APIs are used
    -classpath <path> Specify where to find user class files
    -sourcepath <path> Specify where to find input source files
    -bootclasspath <path> Override location of bootstrap class files
    -extdirs <dirs>    Override location of installed extensions
    -d <directory>    Specify where to place generated class files
    -encoding <encoding> Specify character encoding used by source files
    -source <release> Provide source compatibility with specified release
    -target <release> Generate class files for specific VM version
    -help             Print a synopsis of standard options

C:\Documents and Settings\depietro>
```

Se viene visualizzato un messaggio di comando non riconosciuto, verificate di aver eseguito correttamente le operazioni e soprattutto i percorsi delle cartelle.

Installazione di Tomcat

Tomcat è un progetto dell'Apache Software Foundation ed è quello più utilizzato tra gli sviluppatori di tutto il mondo. Esistono varie versioni di Tomcat:

- Tomcat v7.x fa riferimento alle specifiche delle Servlet v3.0 e delle JSP v2.2
- Tomcat v6.x fa riferimento alle specifiche delle Servlet v2.5 e delle JSP v2.1
- Tomcat v5.x fa riferimento alle specifiche delle Servlet v2.4 e delle JSP v2.0
- Tomcat v4.x fa riferimento alle specifiche delle Servlet v2.3 e delle JSP v1.2
- Tomcat v3.x fa riferimento alle specifiche delle Servlet v2.2 e delle JSP v1.1

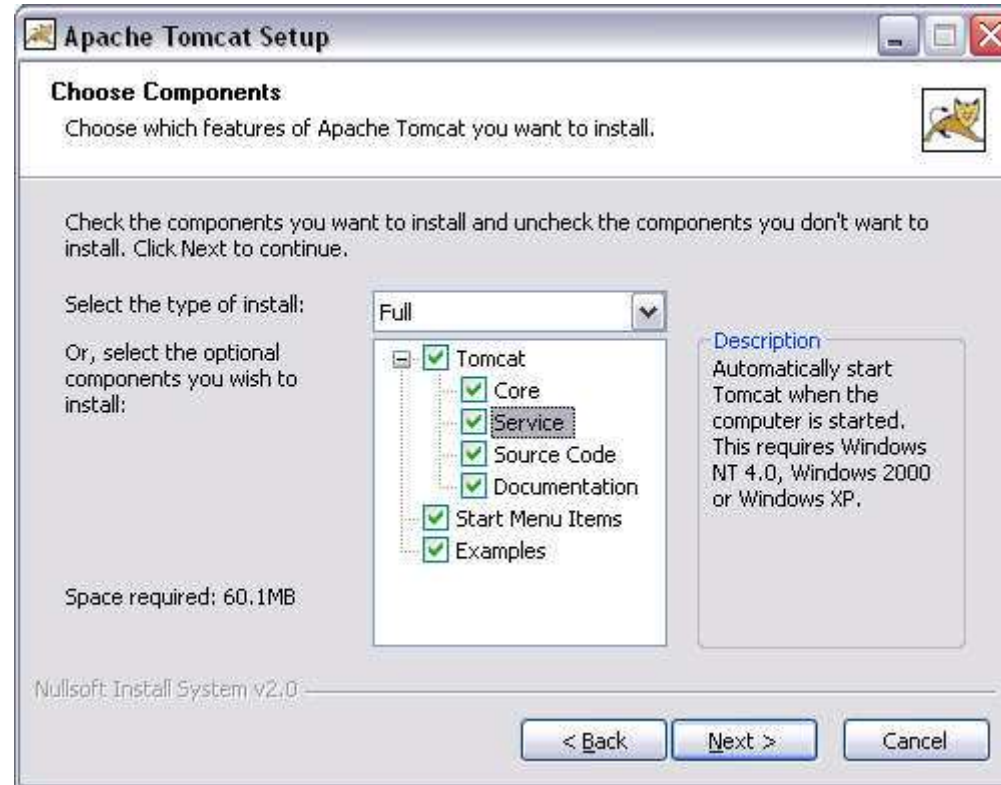
La seguente installazione si riferisce alla versione di Apache Tomcat 7.

Scaricate il file .exe all'indirizzo seguente

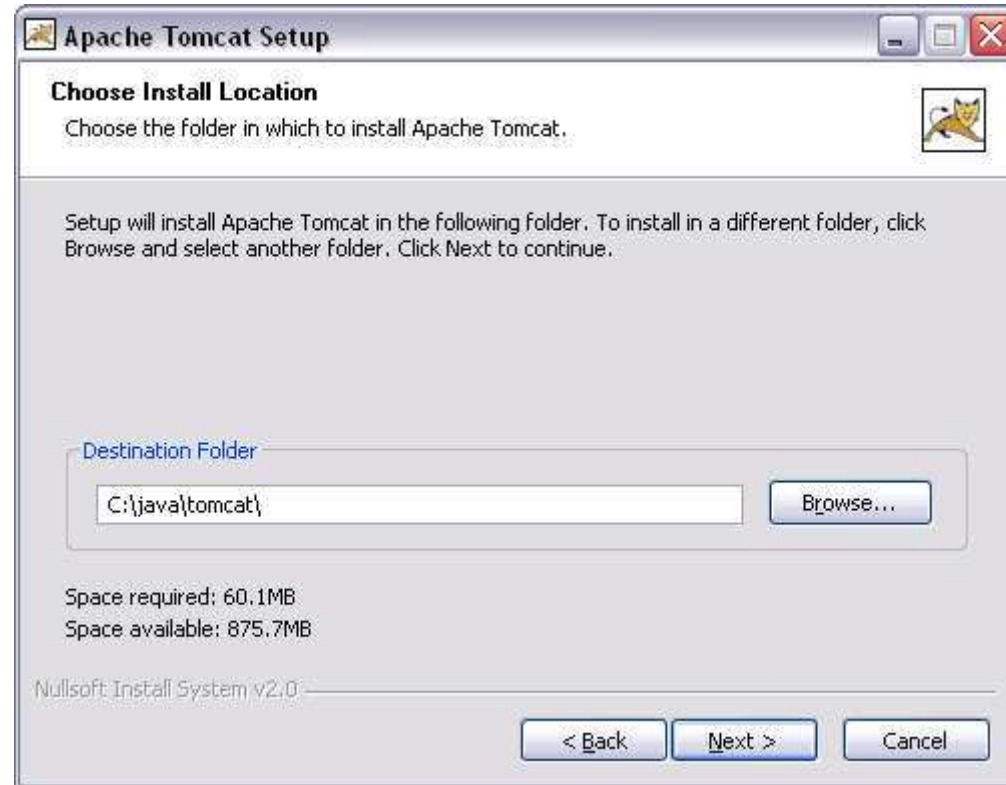
<http://tomcat.apache.org/download-70.cgi>

Assicuratevi di installare l'applicativo in versione Full di modo che venga installato come servizio di Windows e possa quindi partire automaticamente all'avvio del sistema:

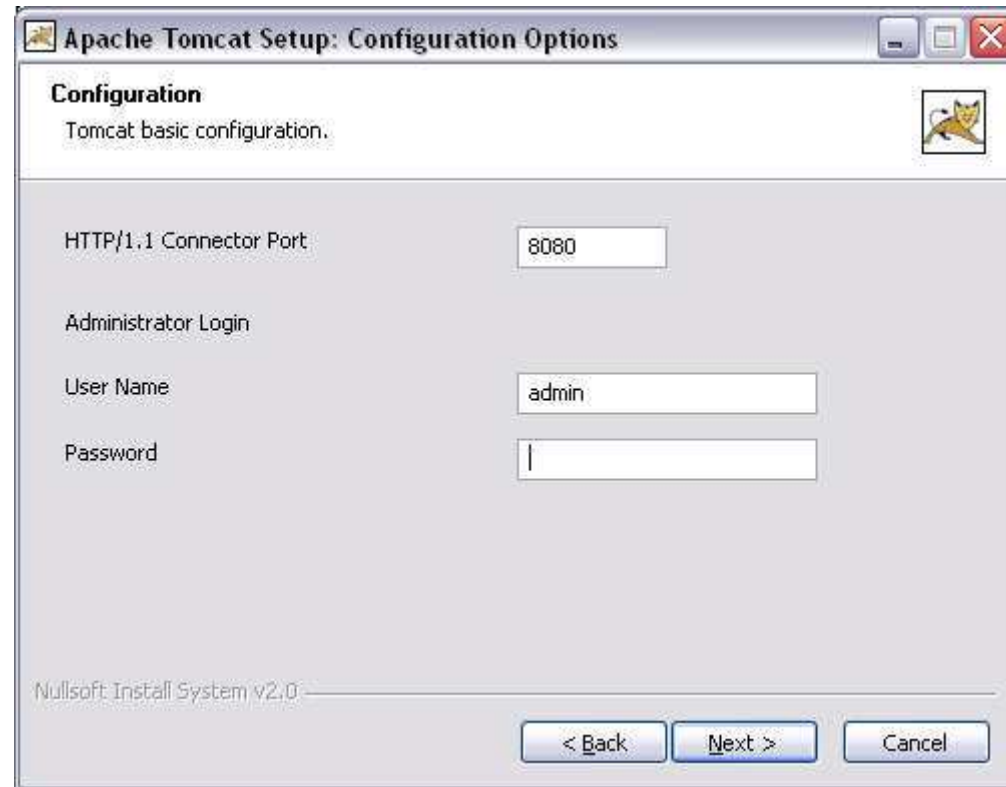
Installazione di Tomcat



Installazione di Tomcat



Installazione di Tomcat



The screenshot shows the 'Apache Tomcat Setup: Configuration Options' window. The title bar includes the text 'Apache Tomcat Setup: Configuration Options' and standard window controls. The main content area is titled 'Configuration' with the subtitle 'Tomcat basic configuration.' and a small Tomcat logo. It contains four configuration fields: 'HTTP/1.1 Connector Port' with the value '8080', 'Administrator Login' with the value 'admin', and 'User Name' and 'Password' fields which are currently empty. At the bottom, there is a status bar that reads 'Nullsoft Install System v2.0' and three buttons: '< Back', 'Next >', and 'Cancel'.

Apache Tomcat Setup: Configuration Options

Configuration
Tomcat basic configuration.

HTTP/1.1 Connector Port: 8080

Administrator Login: admin

User Name:

Password:

Nullsoft Install System v2.0

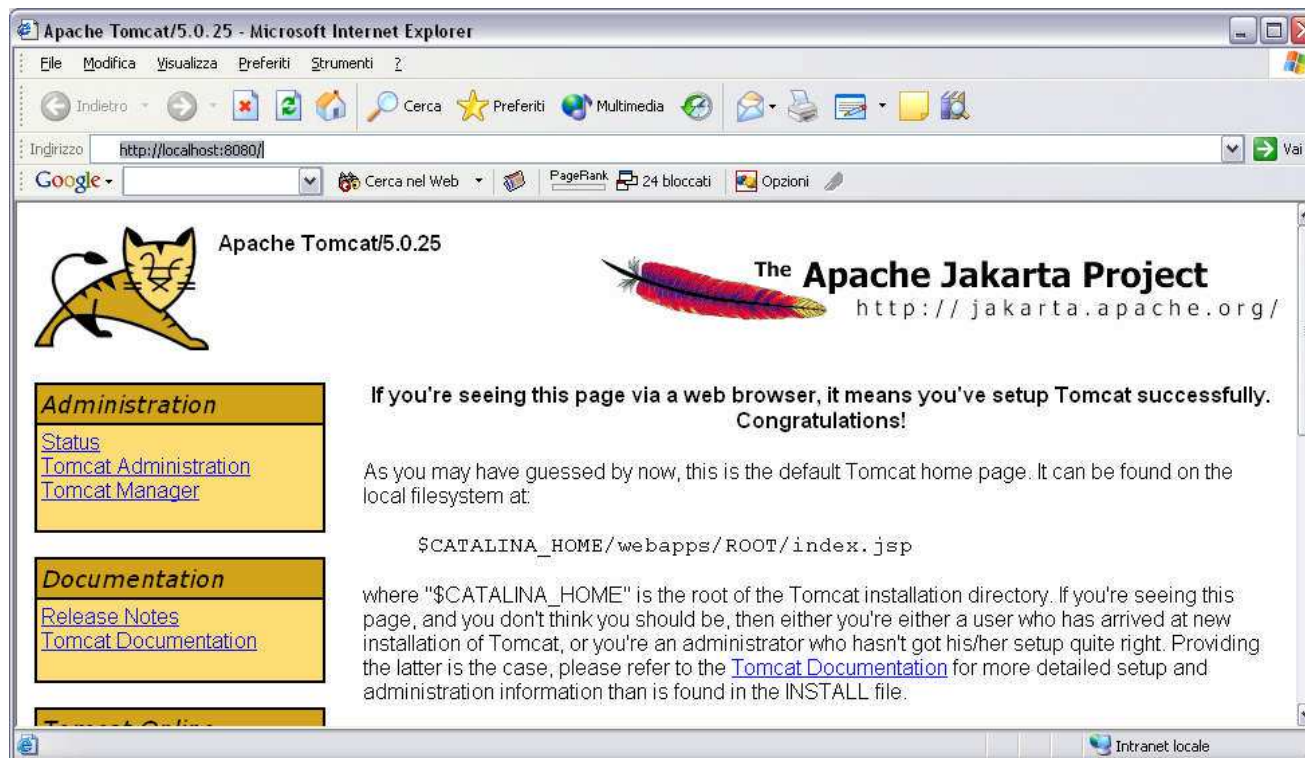
< Back Next > Cancel

Installazione di Tomcat



Installazione di Tomcat

Per verificare che il tutto sia andato a buon fine, aprite il browser e digitate l'url <http://localhost:8080/> , dovrebbe aprirsi la seguente pagina:



Da notare che Tomcat lavora sulla porta **8080** e non crea alcun conflitto con gli altri Web Server che di solito lavorano sulla porta 80.

Installazione di Tomcat

Creiamo una Web Application

Se guardiamo nella directory di installazione di Tomcat, noteremo la seguente struttura:

- *bin* contiene gli script per eseguire e fermare Tomcat
- *common* contiene le librerie accessibili a Tomcat e alle *web application*
- *conf* contiene i file di configurazione di Tomcat
- *logs* contiene i file di log
- *server* contiene le librerie accessibili solamente a Tomcat (Catalina)
- *shared* contiene le classi e le librerie comuni a tutte le *web application*
- *temp* contiene file temporanei di Tomcat
- *webapps* contiene le *web application*
- *work* contiene i file temporanei per ciascuna *web application*. È qui che vengono create le servlet generate dalle JSP.

CorsoJSP

All'interno della directory *webapps* saranno presenti già alcune Web Application predefinite ed in più una cartella di nome ROOT. Questa è la cartella che contiene la radice del sito web gestito da Tomcat, la pagina *index.jsp* che troviamo al suo interno è la risorsa che abbiamo aperto durante il test del web server.

Ora creiamo la nostra prima Web Application. Secondo le specifiche J2EE, una WA deve avere una determinata struttura a directory. Creiamo quindi una nuova cartella di nome *CorsoJSP* (sarà il nome della nostra WA) all'interno di *webapps* e diamole la seguente struttura:

```
\CorsoJSP\  
\CorsoJSP\WEB-INF\  
\CorsoJSP\WEB-INF\web.xml  
\CorsoJSP\WEB-INF\classes\  
\CorsoJSP\WEB-INF\lib\
```

CorsoJSP

CorsoJSP diventerà la radice della nostra applicazione, tutto ciò che sarà inserito al suo interno sarà reso pubblico e quindi disponibile a tutti gli utenti. La risorsa che Tomcat aprirà di default sarà un file di nome *index.htm* o *index.jsp*. E' opportuno organizzare l'applicazione in più sottocartelle (cercando di associare ogni sottolivello ad una sottostruttura logica della nostra applicazione), ogni sottocartella dovrà contenere un file *index.jsp* o *.htm* così da permettere una navigazione lineare per livelli.

CorsoJSP

Differente invece sarà la funzione della cartella WEB-INF che non sarà esposta dall'Application Server e quindi diventa il luogo ideale per riporre le classi (all'interno di *classes*), gli archivi *jar* (all'interno di *lib*) o le risorse che dovranno essere protette dagli utenti esterni.

Importante è la funzione del *Deployment Descriptor* ovvero il file `web.xml` (un file xml editabile con qualsiasi editor testuale), che conterrà la descrizione delle informazioni e delle risorse aggiuntive necessarie al funzionamento della WA.