

Le JavaServer Pages - Lezione 2

Preparazione dell'ambiente

A cura di Giuseppe De Pietro (depietro_giuseppe@yahoo.it)

Contenuti

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).

Vedremo come installarli e configurarli in ambiente Windows (prometto di scrivere a breve la procedura per Linux). Per Tomcat eseguiremo una configurazione di base (tralasciando le funzionalità avanzate a cui dedicherò dei capitoli nel corso delle successive lezioni).

Effettueremo un test per verificare che il tutto funzioni ed infine realizzeremo una semplice applicazione che visualizza la somma di due numeri inseriti dall'utente.

Installazione di J2SE

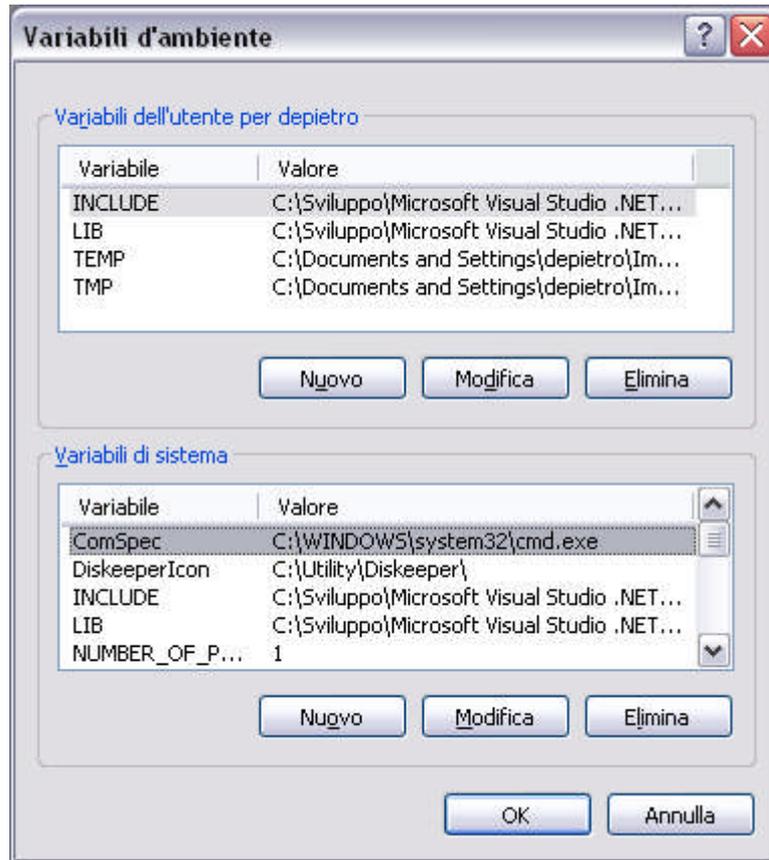
Dopo aver scaricato il pacchetto di Java 2 Standard Edition dal sito della Sun Microsystem (prelevate anche i file di help utili per i riferimenti al linguaggio Java), fare doppio click per eseguire l'installazione. Selezionare il percorso in cui installarlo (personalizzatelo come da figura C:\Java\J2SE):



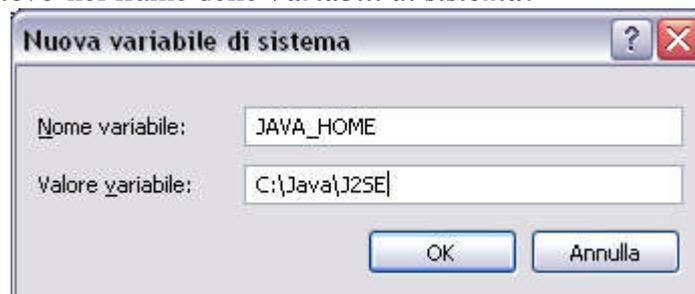
Andate avanti con l'installazione selezionando tutti i componenti da installare.

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:

```

Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

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 v5.x** fa riferimento alle specifiche delle **Servlet v2.4** e delle **JSP v2.0**

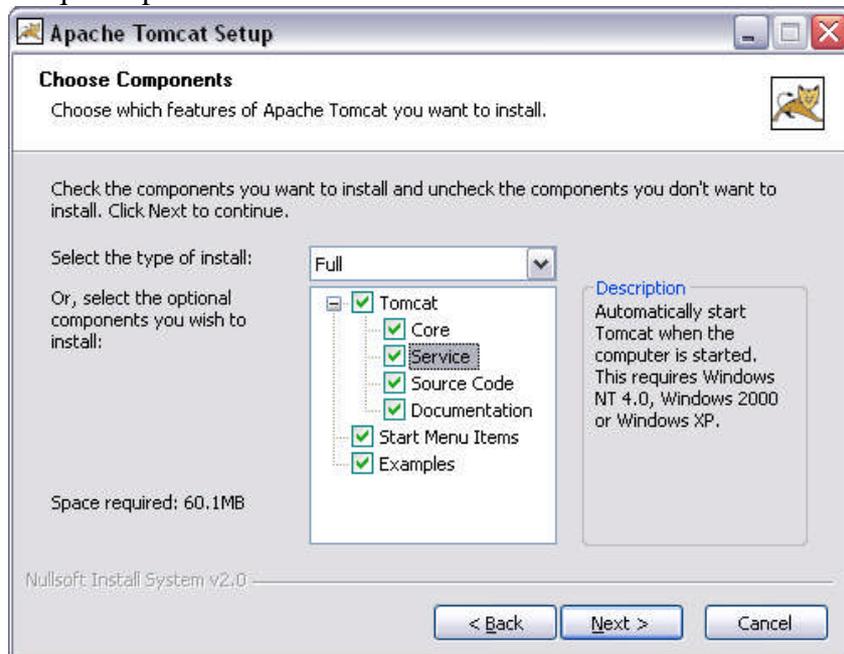
§ Tomcat v4.x fa riferimento alle specifiche delle Servlet v2.3 e delle JSP v1.2

§ Tomcat v3.x (obsoleto) fa riferimento alle specifiche delle Servlet v2.2 e delle JSP v1.1

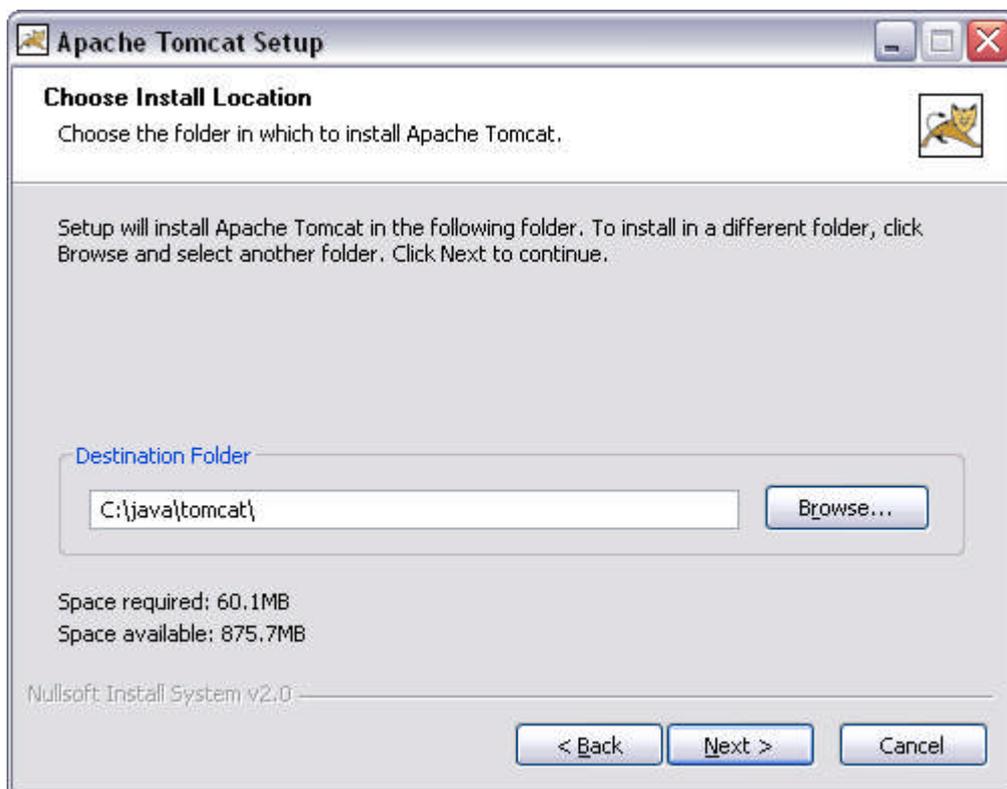
La seguente installazione si riferisce alla versione di Apache Tomcat 5.0.25 (probabile che quando leggerete la guida ci siano versioni successive, le operazioni dovrebbero essere le stesse).

Scaricate il file .exe all'indirizzo seguente <http://jakarta.apache.org/site/binindex.cgi>.

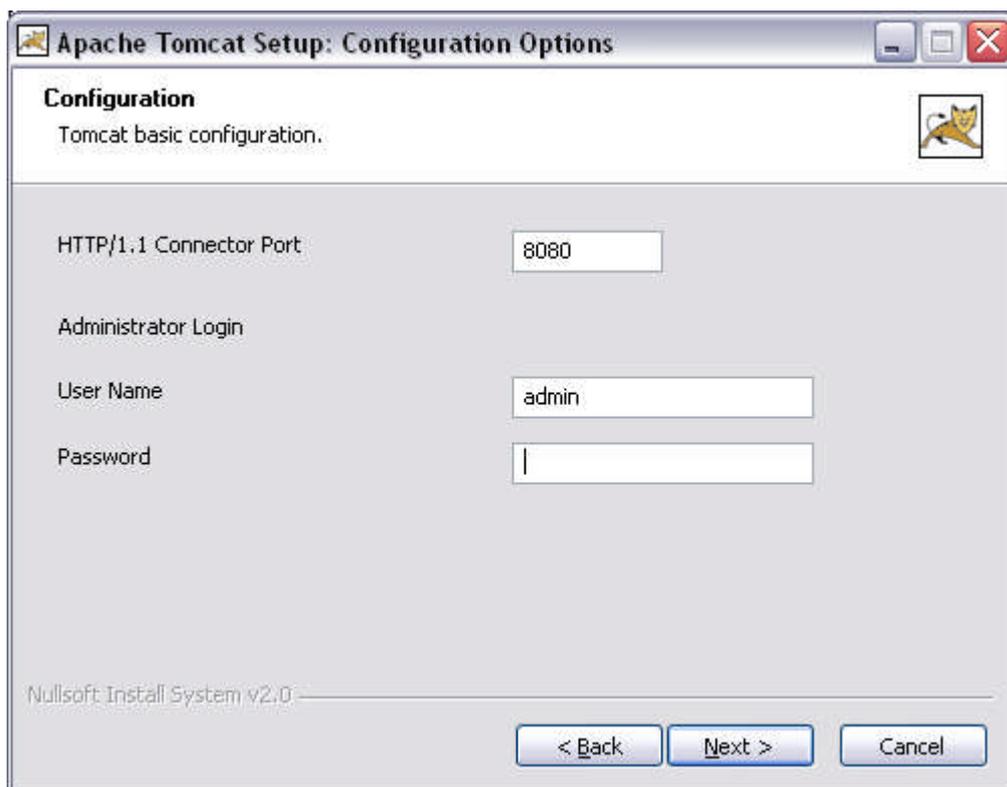
Assicuratevi di installare l'applicativo in versione Full di modo che venga installato come servizio di Windows e possa quindi partire automaticamente all'avvio del sistema:



Come percorso selezionate il seguente:



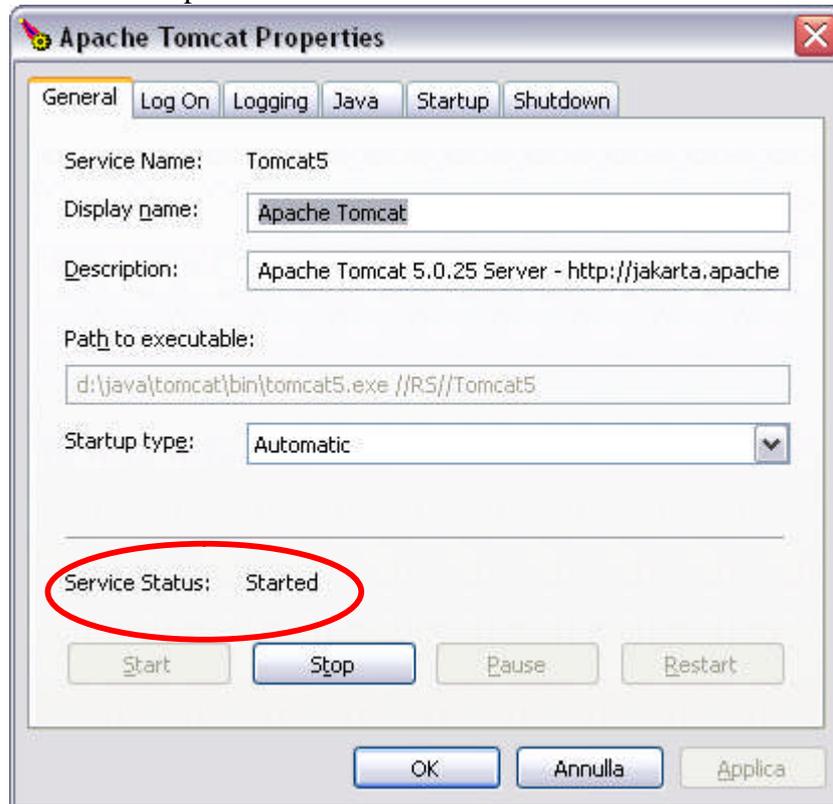
Impostato il nome utente ed una password di amministratore:



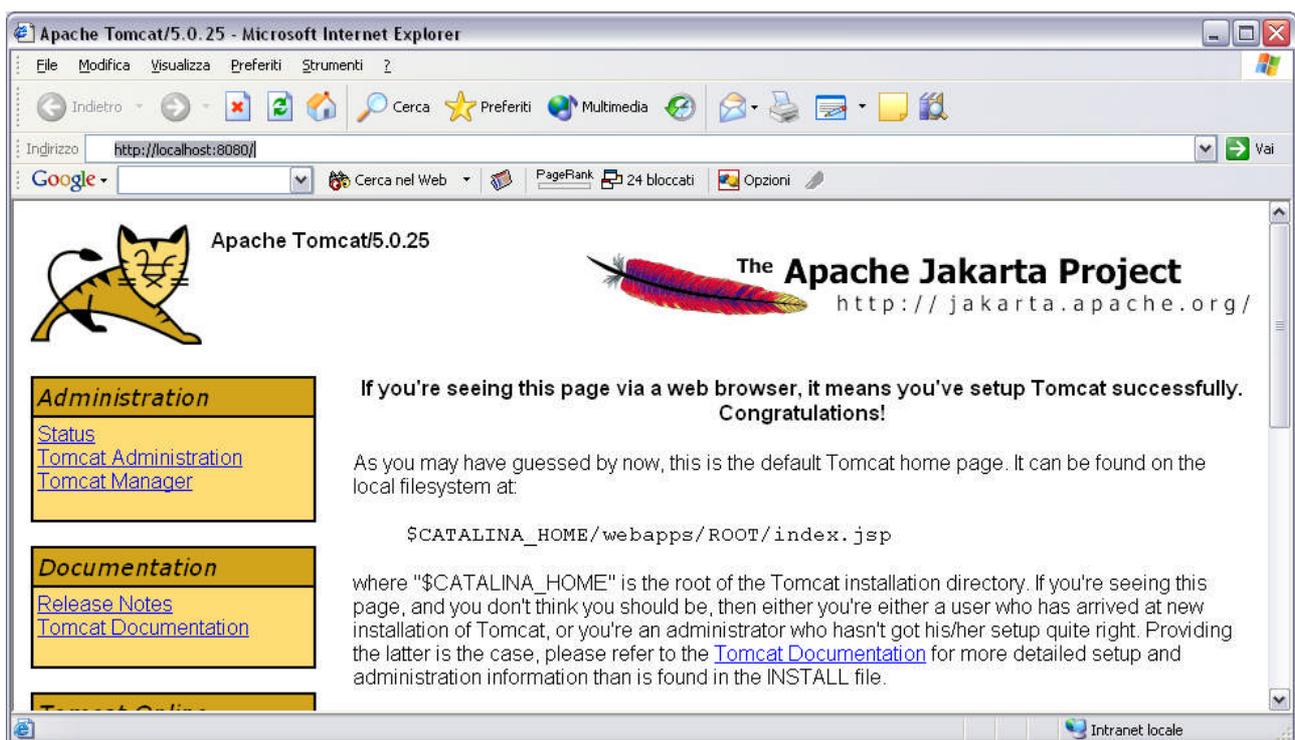
Al termine dell'installazione, avviate il servizio utilizzando l'Apache Tomcat Properties



Assicuratevi che il servizio sia partito:



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.

Creiamo una Web Application

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

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

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 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.

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. In seguito analizzeremo le funzionalità avanzate di questo file, per ora dovrà avere la seguente struttura:

```
<?xml version="1.0" encoding="UTF-8"?>  
<!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>  
</web-app>
```

Ora realizziamo una semplice pagina html che chiederà l'inserimento di due valori numerici ed il server dovrà risponderci con una pagina che visualizzi la somma dei due numeri inviati. I file utilizzati saranno *index.htm* e *somma.jsp* che saranno collocati nella cartella *CorsoJSP* all'interno di *webapps*.

Di seguito si riporta il codice della pagina *index.htm*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="IT">
  <head>
    <title>Corso JSP - Lezione 2</title>
    <meta http-equiv="description" content="Invio di due valori numerici"/>
  </head>
  <body>
    <form method="post" action="somma.jsp">
      <fieldset>
        <legend>Inserisci due valori numerici</legend>
        <label for="valore1">Valore 1:</label>
        <input type="text" id="valore1" name="valore1"><br>
        <label for="valore2">Valore 2:</label>
        <input type="text" id="valore2" name="valore2"><br>
        <input type="submit" value="Invia valori">
      </fieldset>
    </form>
  </body>
</html>
```

Il valore dell'attributo *action* del tag *form* definisce la pagina che dovrà intercettare i valori. La pagina *somma.jsp* sarà così composta:

```
<%@ page language="java" import="java.lang.*,java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="IT">
  <head>
    <title>Somma di due numeri</title>
  </head>
  <body>
    <h1>Somma di due numeri</h1>
    <br/>
    <%
String risultato;
// si preleva i due valori
String valore1=request.getParameter("valore1");
String valore2=request.getParameter("valore2");
// si convertono i valori stringa in valori numerici decimali e si sommano
try
{
  int somma=Integer.parseInt(valore1)+Integer.parseInt(valore2);
  risultato=valore1 +" + "+ valore2 +" = "+ Integer.toString(somma);
} catch (Exception e) {
  //si verifica quando i valori sono errati
  risultato="Valori numerici errati";
}
//si stampa il messaggio
out.write(risultato);
%>
  </body>
</html>
```

La pagina *somma.jsp* preleva i valori inviati dal client grazie al metodo *getParameter* dell'oggetto implicito *request* (nelle lezioni successive analizzeremo nel dettaglio le componenti delle JSP). I valori passati saranno ovviamente in formato stringa, quindi dovremo convertirli in interi utilizzando il metodo *parseInt* della classe *Integer*. Dopodichè costruiremo la stringa di risposta (variabile risultato).

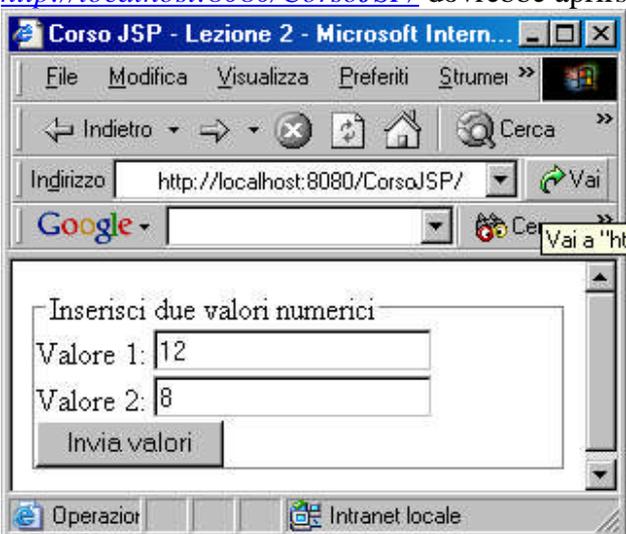
Il costrutto *try...catch* serve per gestire gli errori che potrebbero riscontrarsi nelle istruzioni contenute nel blocco *try*. In Java la gestione degli errori avviene in una modalità intrinsecamente diversa da linguaggi usuali come può essere il linguaggio C o Visual Basic.

Con il linguaggio C i codici di errori solitamente sono valori numerici di cui il programmatore può dimenticarsi di controllarne il valore (spesso per pigrizia).

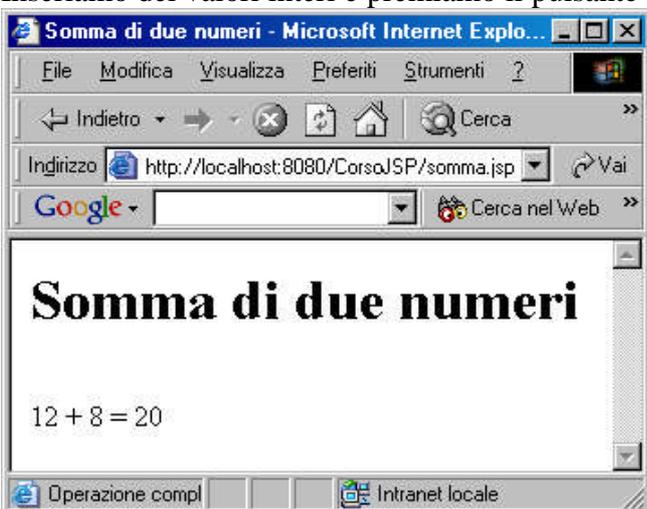
Con le eccezioni del linguaggio Java il programmatore è "obbligato" a gestire le eccezioni o comunque deve dichiarare esplicitamente di non volerle gestire, in questo modo viene sempre reso coscio che una parte di codice potrebbe generare delle eccezioni e qui sta la robustezza del linguaggio Java.

Nel nostro caso i valori inviati potrebbero essere dei caratteri alfanumerici, o valori decimali (l'algoritmo presentato gestisce solo la somma di interi) che causerebbero errori nell'esecuzione della pagina, introducendo il *try..catch* ci assicuriamo sempre il controllo della nostra applicazione anche in caso di errore.

Dopo aver creato i file proviamo che il tutto funzioni. Apriamo il browser al seguente indirizzo <http://localhost:8080/CorsoJSP/> dovrebbe aprirsi la seguente pagina:



Inseriamo dei valori interi e premiamo il pulsante "Invia valori" dovremmo ottenere:



Ora abbiamo l'ambiente configurato correttamente e siamo pronti ad approfondire gli elementi delle JSP.

Per editare i documenti JSP basta un qualsiasi editor testuale, ma è preferibile usare ambienti di sviluppo che facilitano il compito di stesura del codice.

Per chi utilizza già Dreamweaver è avvantaggiato dal fatto che l'ambiente di Macromedia supporta le pagine JSP. Ma per avere una gestione completa delle Java Web Application bisogna orientarsi su prodotti scritti per l'ambiente Java.

Un ottimo prodotto commerciale è il JBuilder di Borland che permette la realizzazione di progetti Java e progetti Web secondo le specifiche J2EE supportando numerosi Application Server.

Mentre in ambito Open Source c'è il progetto di IBM Eclipse (scaricabile gratuitamente all'indirizzo <http://www.eclipse.org/downloads/index.php>, consiglio la versione 3.0RC2 quella da me utilizzata) che integrato con tool commerciali e gratuiti permette l'edit (con funzione di completamento automatico del codice) e il deploy delle java Web Application. Segnalo tra i plug-in commerciali il MyEclipse Enterprise WorkBench (<http://www.myeclipseide.com>) e tra quelli open source l'ottimo Lombok (http://forge.objectweb.org/project/showfiles.php?group_id=97) scaricate la versione per Eclipse 3.0 RC2 (ce ne sono tante!).

Segnalo anche gli ottimi prodotti della Sun Microsystems, Sun One Studio evoluzioni del progetto Forte for Java.