Spago Tutorial

Redatto da Gianfranco Boccalon
Daniela Butano

Indice

Informazioni sulla versione

Versione/Release n° :	1.1	**Data Versione/Release** :	25/10/2004
Descrizione modifiche:	L'esempio del tutorial viene ricondotto alla disponibilità del template Spago.war		
Versione/Release n° :	1.2	**Data Versione/Release** :	19/11/2004
Descrizione modifiche:	In occasione della pubblicazione della versione inglese del documento viene ristrutturato l'indice, inserito lo schema architetturale e vengono riportate alcune modifiche formali. L'esempio è stato modificato con il rilascio della classe PublishAction in spago-core1.0.1.		
Versione/Release n° :	1.3	**Data Versione/Release** :	13/12/2004
Descrizione modifiche:	Aggiornata la struttura di *master.xml*		

Scopo del documento

Il documento ha l'obiettivo di fornire una rapida introduzione alla configurazione di un'applicazione scritta con Spago a partire dal template di progetto *Spago.war* rilasciato in *startup-sample.zip*.

Riferimenti

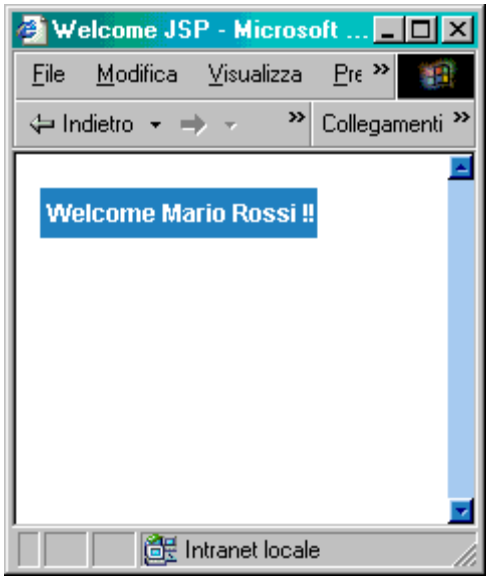
Per ulteriori informazioni sul framework Spago si rinvia ai seguenti documenti disponibili in http://spago.eng.it/docs_it/documentation/index.html :

- [1] *Spago Overview*
- [2] *Spago User Guide.*

1 Applicazione di esempio

L'applicazione scelta come esempio è composta da due pagine JSP: una di login e una di benvenuto, da presentare a login effettuato.

Verrà utilizzato un file di configurazione XML per censire gli utenti, cui si accederà tramite il framework Spago.



2 Prerequisiti Software

L'esempio prevede l'utilizzo di alcuni strumenti di sviluppo, elencati di seguito:

- **JDK 1.4.x** scaricabile dal sito *java.sun.com*
- **Tomcat 5.0.28** scaricabile dal sito *http://jakarta.apache.org/*
- **Eclipse 3.0** scaricabile dal sito *http://www.eclipse.org/*
- **Lomboz 3.0.1** scaricabile dal sito *www.objectlearn.com*

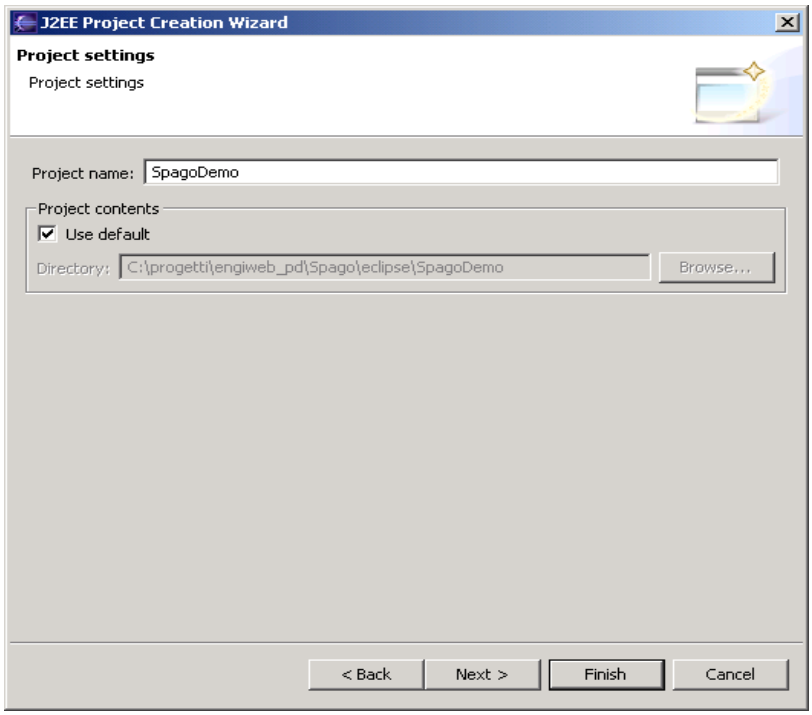
Si assume che la configurazione di questi strumenti sia già stata effettuata.

Inoltre è necessaria la versione distribuita di Spago scaricabile da *http://sourceforge.net/projects/spago*.

Utilizziamo per comodità il template di progetto Spago.war scaricabile da *http://sourceforge.net/projects/spago* in startup-sample.zip.

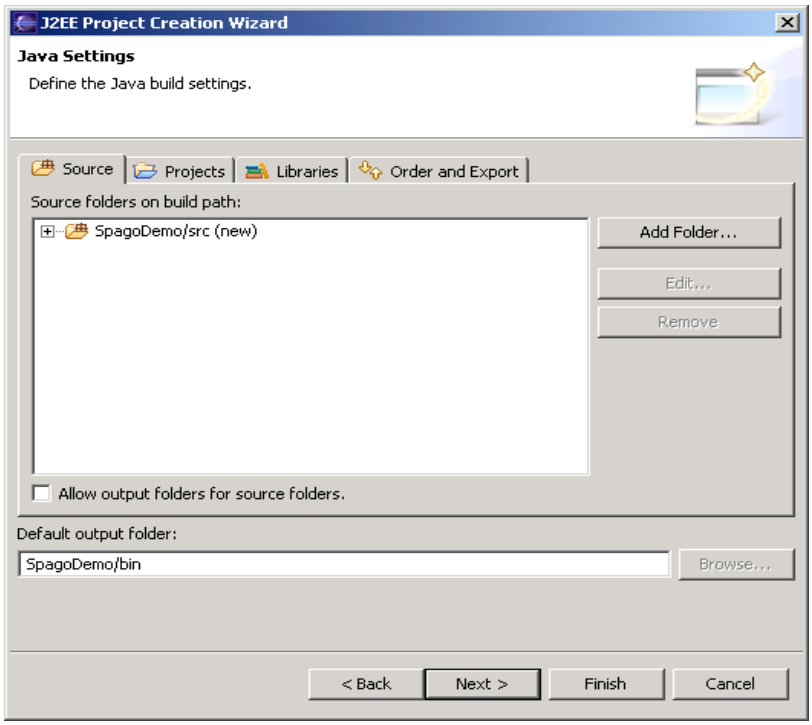
3 Creazione del progetto

Dopo aver avviato *Eclipse*, creare il progetto per l'applicazione d'esempio selezionando la voce *New->Lomboz J2EE Project*, apparirà il seguente wizard:



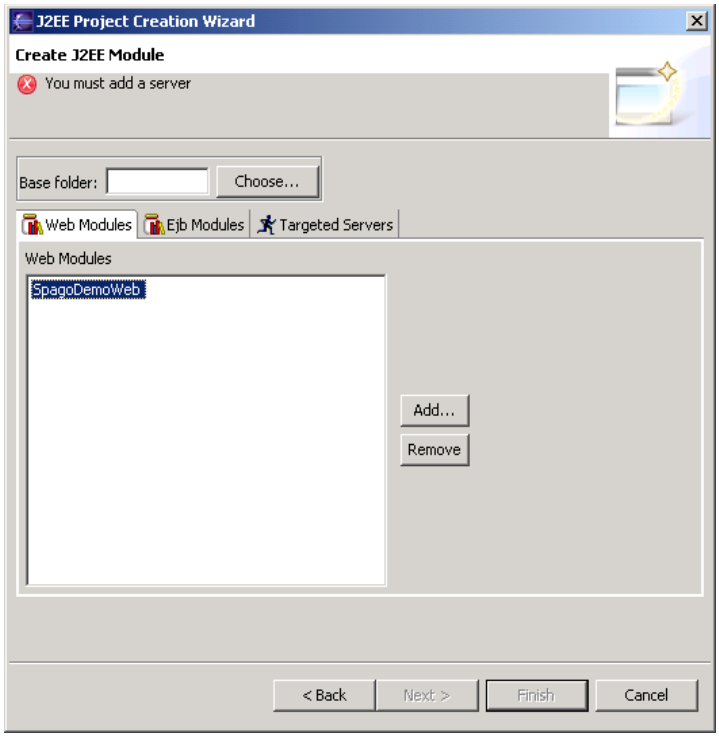
Impostare come nome di progetto "*SpagoDemo*".

Utilizzare le seguenti impostazioni per i "*Java settings*":

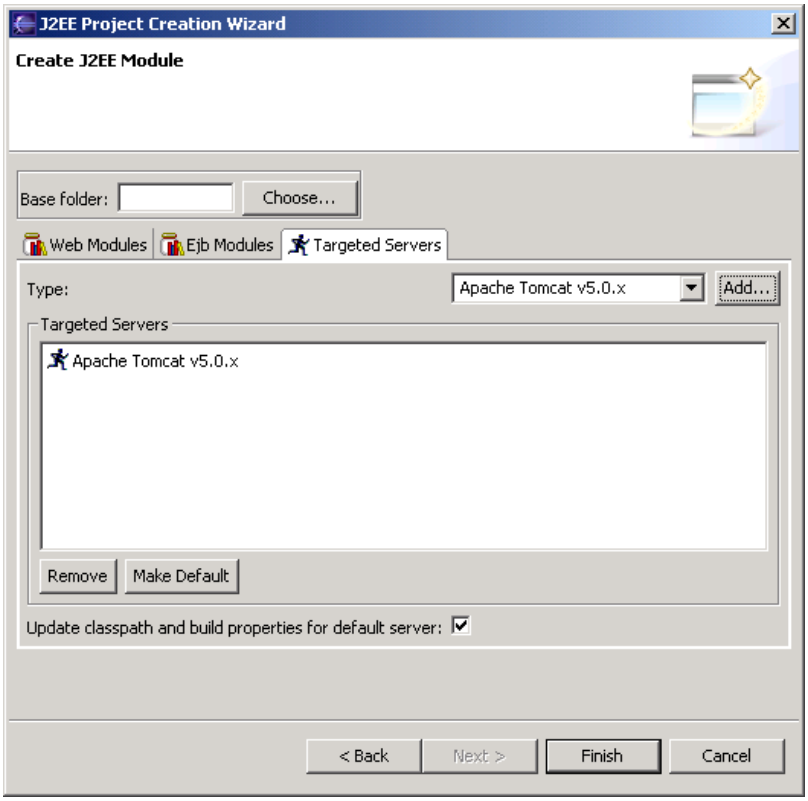


E' importante che l'output folder sia "***SpagoDemo/bin***".

Cliccare su *Next* e aggiungere un modulo web chiamandolo *SpagoDemoWeb*.



Aggiungere *Tomcat5.0.x* come application server per l'applicazione.



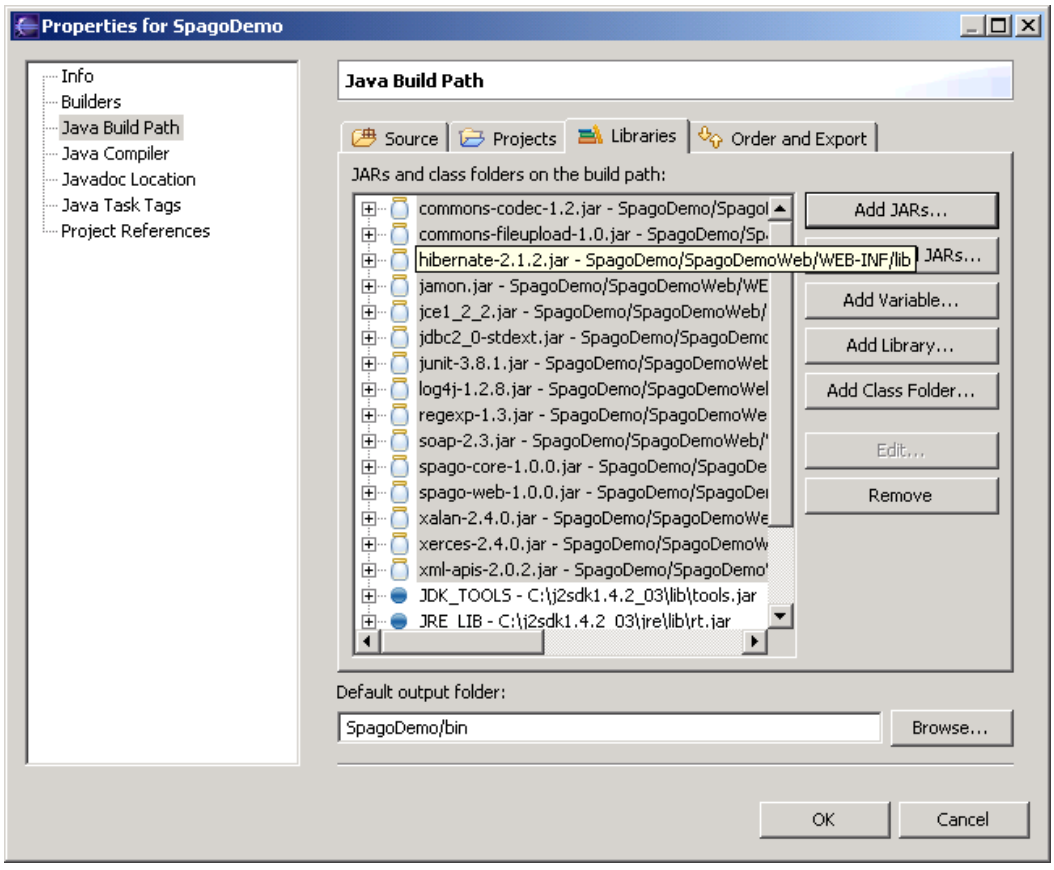
4 Configurazione del progetto

Nel seguito vediamo quali sono i passi necessari da seguire per la configurazione del progetto appena creato in modo che possa utilizzare il framework Spago.

Innanzitutto estraiamo i contenuti di *Spago.war* nella cartella *SpagoDemoWeb*. In questo modo abbiamo a disposizione tutti i file di configurazione, tutte le librerie esterne a Spago e quelle di Spago, i tag libraries, i fogli di stile, le pagine di gestione degli errori e delle eccezioni.

4.1 Librerie

Il progetto deve essere configurato in modo che possa utilizzare le librerie necessarie: si accede alle proprietà del progetto e sotto la voce "*Java Build Path*" si aggiungono tutti i jar presenti in *SpagoDemoWeb/WEB-INF/lib*.



4.2 Files di configurazione

Spago necessita di una serie di files di configurazione XML che sono presenti nella directory *WEB-INF/conf/spago* di distribuzione di Spago, che ora sono presenti nella cartella *WEB-INF/conf/spago* del progetto appena creato.

4.3 Web.xml

Il file *web.xml* contiene la configurazione corretta per l'adapter HTTP del framework, oltre alla configurazione delle tag libraries.

Nel caso in cui si vogliano salvare i file di configurazione in una directory esterna al progetto, togliere i commenti alla sezione seguente e modificare il valore del parametro *AF_ROOT_PATH* in modo che contenga il path della directory:

```
<init-param>
    <param-name>AF_ROOT_PATH</param-name>
    <param-value>C:\Progetti\eclipse\SpagoDemo\SpagoDemoWeb</param-value>
</init-param>
```

4.4 Tag Libraries

Spago contiene una tag library la cui definizione si trova nel file "*\WEB-INF\tld**spago.tld***".

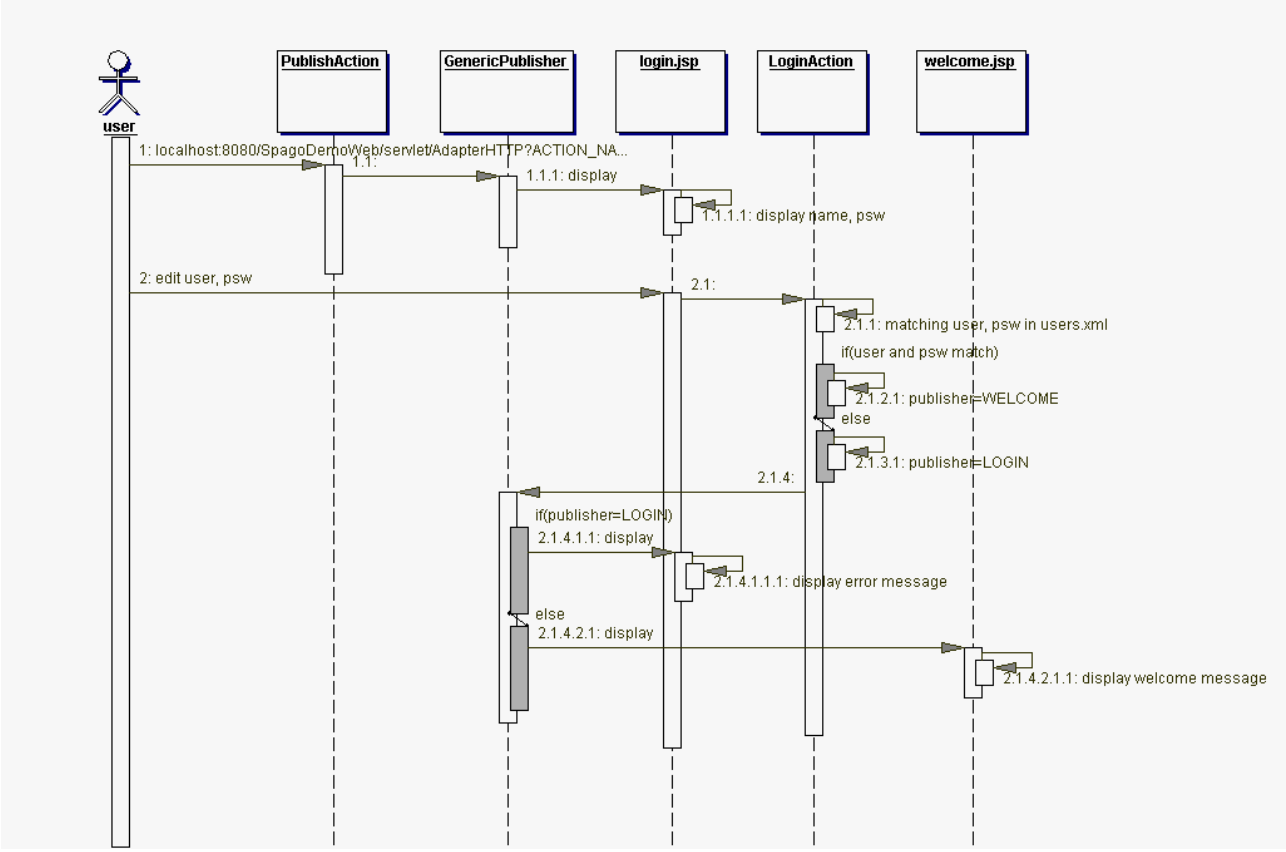
4.5 Fogli di stile, immagini e JSP

Spago ha alcune funzionalità di generazione automatica di form di liste che fanno uso di alcuni fogli di stile (css) e alcune immagini.

5 Architettura Applicativa

L'esempio qui sviluppato utilizza la modalità di dispatching ad *action* cui si fa corrispondere ad una richiesta di un servizio un singolo oggetto di business (in alternativa alla modalità a *module*, per la cui descrizione si rinvia alla documentazione in Riferimenti).

Lo schema dell'applicazione è riportato nel seguito.



6 Sviluppo dell'applicazione

Nel seguito sono descritti in sequenza i passi necessari per lo sviluppo dell'applicazione.

6.1 ACTION.XML

Le action utilizzate dall'applicazione sono due:

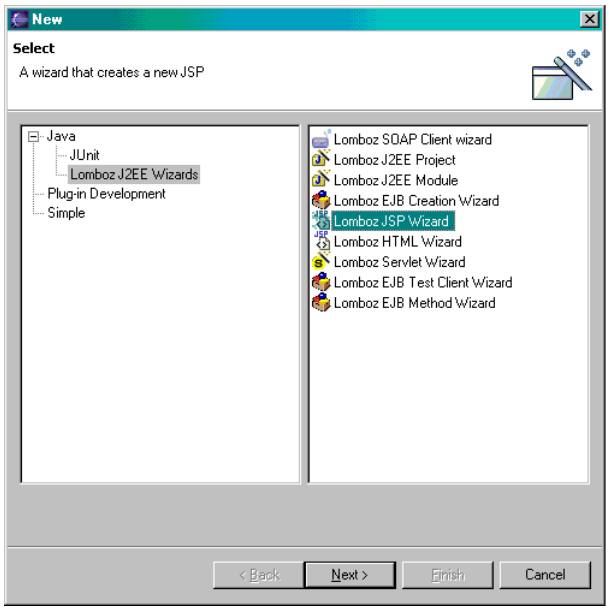
- *Login*: è l'action che effettua l'attività di autenticazione dell'utente.
- *Publish*: è l'action attraverso la quale pubblichiamo la prima pagina JSP. Spago ne fornisce un'implementazione.

Il file "*SpagoDemo\SpagoDemoWeb\WEB-INF\conf\spago**actions.xml***" andrà configurato come segue:

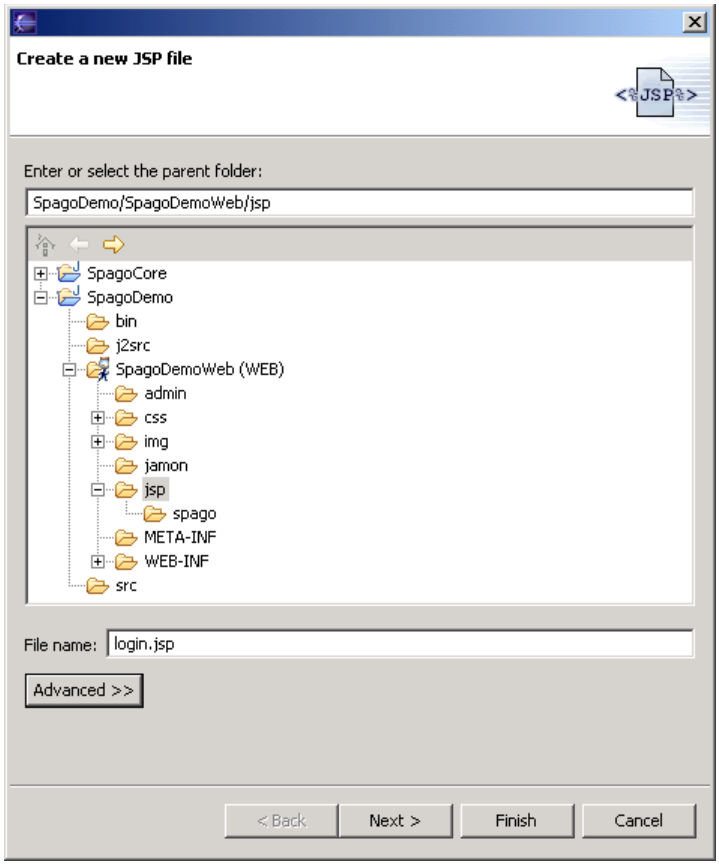
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ACTIONS>
      <ACTION name="LOGIN"
            class="it.eng.spago.demo.action.LoginAction"
            scope="REQUEST"/>
      <ACTION name="PUBLISH"
            class="it.eng.spago.dispatching.action.util.PublishAction"
            scope="REQUEST"/>
</ACTIONS>
```

6.2 PAGINA JSP DI LOGIN

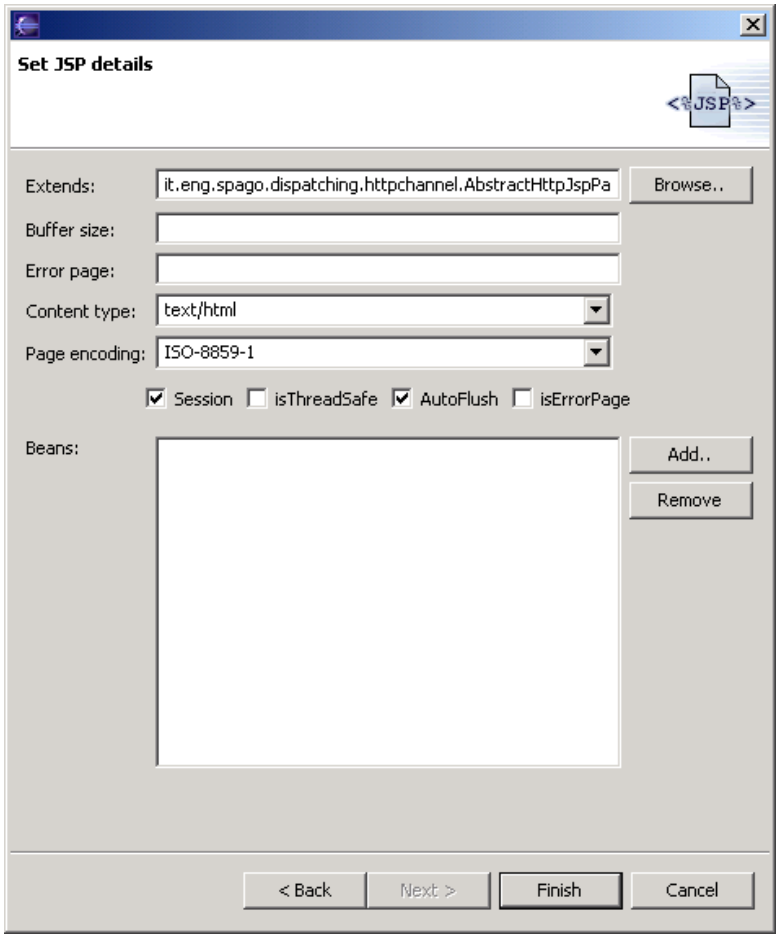
Per creare la pagina di Login si utilizza il wizard "*Lomboz JSP Wizard*", sul menu "*File->New*".



La pagina va creata nella directory "*SpagoDemo\SpagoDemoWeb\jsp*" e va chiamata *login.jsp*.



La pagina deve derivare da "*it.eng.spago.dispatching.httpchannel.**AbstractHttpJspPage***".



In conseguenza di un'anomalia del tool di sviluppo, dopo aver scelta ***AbstractHttpJspPage*** come classe da cui derivare la pagina, viene segnalato l'errore "invalid type". Per rimuoverlo è sufficiente scrivere qualcosa nel campo "*Buffer size*" e poi cancellare il testo scritto, prima di premere il bottone "*Finish*".

La pagina JSP è strutturata come segue:

```
<%@ page language="java"
  extends="it.eng.spago.dispatching.httpchannel.AbstractHttpJspPage" %>
<%@ taglib uri="spagotags" prefix="spago"%>
<!DOCTYPE HTML PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
      <link rel="stylesheet" type="text/css"
href="../css/spago/listdetail.css"/>
      <title>Login JSP</title>
</head>
<body bgcolor="#FFFFFF" onload="checkError();">
<spago:error/>

<FORM name="login" method=post
      action="../servlet/AdapterHTTP?ACTION_NAME=LOGIN">
<table>
      <tr><th class="LISTA" height="25" colspan="2"> Login</th></tr>
      <tr><td colspan="2" class="piccolo"> </td></tr>
      <tr>
            <td class="DETAIL">UserId</td>
            <td class="DETAIL">
              <input type="text" name="user" value="" maxlength="20" size="10">
            </TD>
      </tr>
      <tr>
            <td class="DETAIL">Password</td>
            <TD class="DETAIL">
              <input type="password" name="password" value="" maxlength="20"
                     size="10">
            </TD>
      </tr>
      <tr><td>
      </table>
      <table bgcolor="#ffffff">
      <tr><td colspan="2"><input type="submit" value="Invia"></td></tr>
      </table>
      </td></tr>
</table>
</form>

</body>
</html>
```

Alcune note importanti:

- La pagina discende da "*it.eng.spago.dispatching.httpchannel.**AbstractHttpJspPage***" : questo le permette di accedere al response container, alla response, etc.
- Tramite la riga *<%@ taglib uri="spagotags" prefix="spago"%>* si possono utilizzare i tag del framework.
- Al submit del form di login viene invocata l'action di login, come si può vedere dall'attributo *action="../servlet/AdapterHTTP?ACTION_NAME=LOGIN"* del form.

6.3 Action di Login

Creare l'action di login *it.eng.spago.demo.action.**LoginAction*** come segue:

```
package it.eng.spago.demo.action;

import it.eng.spago.base.SourceBean;
import it.eng.spago.configuration.ConfigSingleton;
import it.eng.spago.dispatching.action.AbstractAction;
import it.eng.spago.error.EMFErrorSeverity;
import it.eng.spago.error.EMFUserError;

public class LoginAction extends AbstractAction {

      public void service(SourceBean request, SourceBean response)
            throws Exception {
            // Reperisco i parametri presenti nel form
            String userName = (String)request.getAttribute("user");
            String password = (String)request.getAttribute("password");

            // Verifico se nel file di configurazione c'è il
            // corrispondente utente
            ConfigSingleton config = ConfigSingleton.getInstance();
            SourceBean userConfig = (SourceBean)
config.getFilteredSourceBeanAttribute("USERS.USER", "NAME", userName);

            if (userConfig == null)
                  getErrorHandler().addError(new
                    EMFUserError(EMFErrorSeverity.ERROR, ERROR_USER_NOT_FOUND));
            else if ((password != null) &&
                !password.equals((String)userConfig.getAttribute("password")))
                  getErrorHandler().addError(new
                    EMFUserError(EMFErrorSeverity.ERROR, ERROR_PASSWORD_WRONG));
            else {
                  response.setAttribute("name",
                        (String)userConfig.getAttribute("name"));
                  response.setAttribute("surname",
                        (String)userConfig.getAttribute("surname"));
            }

            response.setAttribute("PUBLISHER",
                  (getErrorHandler().getErrors().size() != 0) ? LOGIN_PUBLISHER
                                                      : WELCOME_PUBLISHER);
      }

      private final static int ERROR_USER_NOT_FOUND = 5000;
      private final static int ERROR_PASSWORD_WRONG = 5001;

      private final static String LOGIN_PUBLISHER = "LOGIN";
      private final static String WELCOME_PUBLISHER = "WELCOME";
}
```

L'action reperisce i parametri "*user*" e "*password*" dalla richiesta e verifica se nel file di configurazione degli utenti è presente l'utente specificato, con la password specificata. In caso contrario aggiunge un errore al gestore degli errori (*ErrorHandler*).

In caso di errori l'action rimanda la visualizzazione alla pagina di login, che riporterà all'utente le descrizioni degli errori.

6.4 CONFIGURAZIONE UTENTI

L'applicazione utilizza un nuovo file di configurazione chiamato "***users.xml***" da creare in "*SpagoDemo\SpagoDemoWeb\conf\spago*" strutturato come segue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<USERS>
    <USER name="Mario" surname="Rossi" password="prova1"/>
    <USER name="Giovanni" surname="Bianchi" password="prova2"/>
</USERS>
```

Perchè questo file sia visibile al configuratore del framework, va aggiunto in "*SpagoDemo\SpagoDemoWeb\conf**master.xml***" il riferimento al nuovo file tramite la riga evidenziata in grassetto:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<MASTER>
      <CONFIGURATOR
        path="/WEB-INF/conf/spago/data_access.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/actions.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/business_map.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/common.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/dispatchers.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/initializers.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/modules.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/pages.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/presentation.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/publishers.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/security.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/statements.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/tracing.xml"/>
    <CONFIGURATOR
        path="/WEB-INF/conf/spago/users.xml"/>
</MASTER>
```

6.5 CONFIGURAZIONE DELLE STRINGHE MULTI-LINGUA

Nella directory "*\WEB-INF**classes***" del template di progetto sono presenti i file di properties nei quali vengono censiti tutti i messaggi da visualizzare.

Copiare questi file nella cartella dei sorgenti del nuovo progetto "*SpagoDemo**src*". In ***messages_en_US.properties*** devono essere inseriti i messaggi di errore utilizzati dall'action qualora si verifichino situazioni di errore nella fase di login.

Le indicazioni su come utilizzare files di properties in linguaggio diverso dall'inglese sono in *Spago User Guide*.

Nel nostro esempio sono censiti due errori applicativi, come segue:

```
5000=Utente inesistente
5001=Password errata
```

6.6 PRESENTATION.XML

Il file "*SpagoDemo\SpagoDemoWeb\WEB-INF\conf\spago**presentation.xml***" contiene l'associazione tra le action ed i corrispondenti publisher. Nel nostro caso l'abbiamo configurato come segue:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PRESENTATION>
    <MAPPING business_name="LOGIN" business_type="ACTION"
             publisher_name="GenericPublisher"/>
    <MAPPING business_name="PUBLISH" business_type="ACTION"
             publisher_name="GenericPublisher"/>
</PRESENTATION>
```

6.7 PUBLISHERS.XML

Il file "*SpagoDemo\SpagoDemoWeb\WEB-INF\conf\spago**publishers.xml***" contiene le dichiarazioni dei publisher per i vari canali. In questo caso abbiamo censito i publisher per il solo canale HTTP.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<PUBLISHERS>
    <PUBLISHER name="GenericPublisher">
    <RENDERING channel="HTTP" type="JAVA" mode="FORWARD">
          <RESOURCES>
          <ITEM prog="0"
resource="it.eng.spago.presentation.DefaultPublisherDispatcher">
          <CONFIG>
          <CHECKS>
          <CHECK target="Login">
                <CONDITIONS>
                <PARAMETER name="PUBLISHER" scope="SERVICE_REQUEST"
value="LOGIN" />
                </CONDITIONS>
```

```
            </CHECK>
            <CHECK target="Login">
                  <CONDITIONS>
                  <PARAMETER name="PUBLISHER" scope="SERVICE_RESPONSE"
value="LOGIN" />
                  </CONDITIONS>
            </CHECK>
            <CHECK target="Welcome">
                  <CONDITIONS>
                  <PARAMETER name="PUBLISHER" scope="SERVICE_RESPONSE"
value="WELCOME" />
                  </CONDITIONS>
            </CHECK>
            </CHECKS>
            </CONFIG>
            </ITEM>
            </RESOURCES>
            </RENDERING>
      </PUBLISHER>
```

```
    <PUBLISHER name="Login">
        <RENDERING channel="HTTP" type="JSP" mode="FORWARD">
            <RESOURCES>
                <ITEM prog="0" resource="/jsp/login.jsp"/>
            </RESOURCES>
        </RENDERING>
    </PUBLISHER>
```

```
    <PUBLISHER name="Welcome">
        <RENDERING channel="HTTP" type="JSP" mode="FORWARD">
            <RESOURCES>
                <ITEM prog="0" resource="/jsp/welcome.jsp"/>
            </RESOURCES>
        </RENDERING>
    </PUBLISHER>
</PUBLISHERS>
```

In questo caso entrambe le action utilizzano un publisher di tipo Java, che ci permette di rendere dinamica la pagina con cui viene effettuata la presentazione.

Sono state censite anche le due pagine JSP per consentirne la visualizzazione.

6.8 Pagina JSP di benvenuto

Infine, ecco la pagina JSP di benvenuto: va creata sempre con il wizard di Lomboz e va chiamata "***welcome.jsp***".

```
<%@ page language="java"
extends="it.eng.spago.dispatching.httpchannel.AbstractHttpJspPage" %>

<!DOCTYPE HTML PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
```

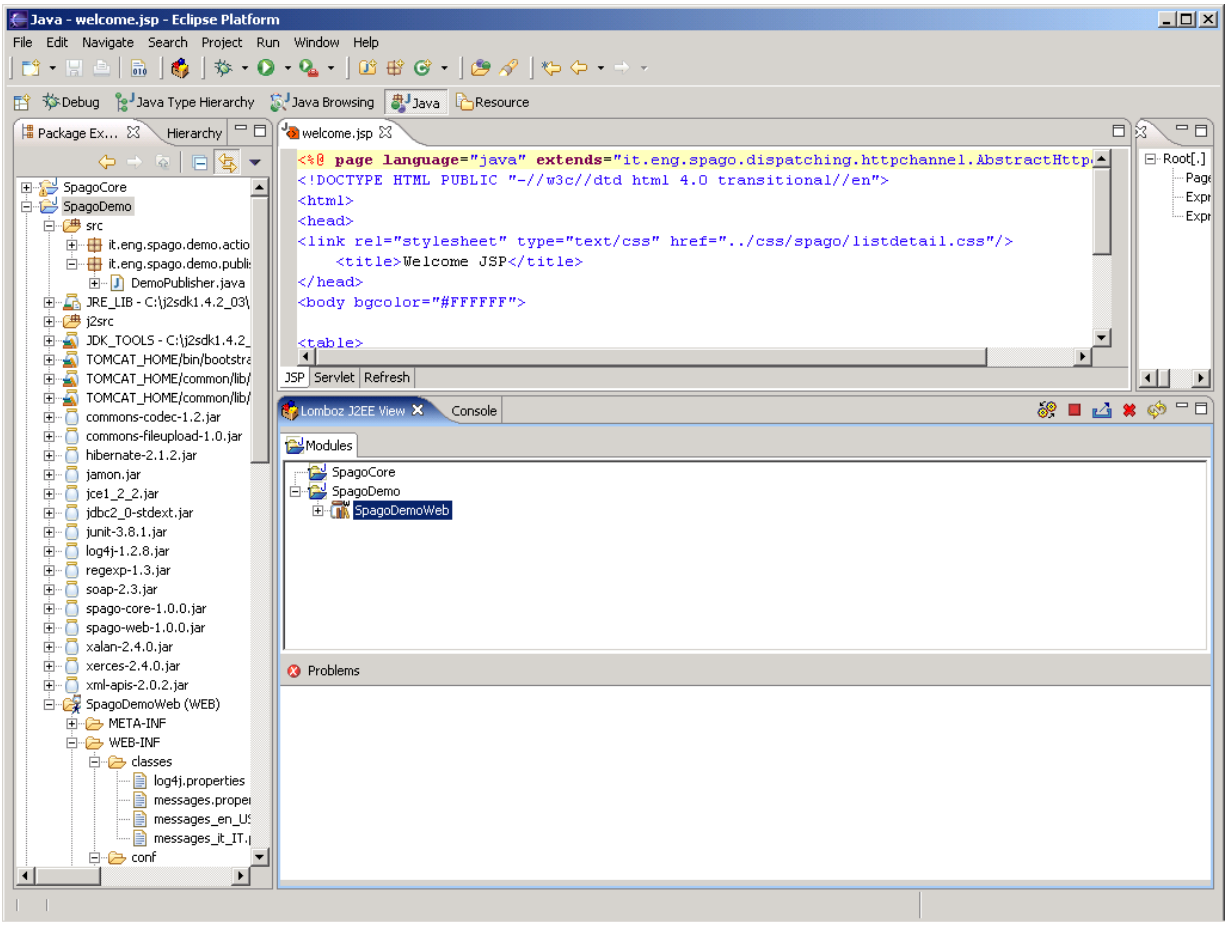
```
      <link rel="stylesheet" type="text/css"
href="../css/spago/listdetail.css"/>
      <title>Welcome JSP</title>
</head>
<body bgcolor="#FFFFFF">

<table>
      <tr><th class="LISTA" height="25" colspan="2"> Welcome
      <%= getServiceResponse(request).getAttribute("name") %>
      <%= getServiceResponse(request).getAttribute("surname") %> !!</th></tr>
</table>

</body>
</html>
```

7 Esecuzione dell'applicazione

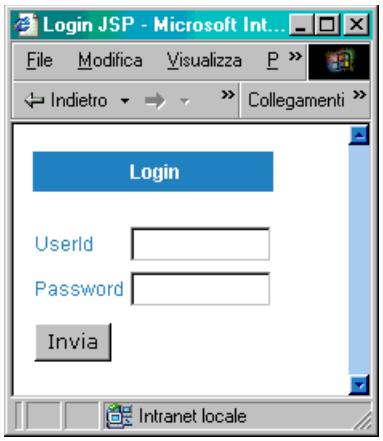
A questo punto si può effettuare il deploy dell'applicazione su Tomcat: dalla "*Java Perspective*", selezionare il bottone  (J2EE Project Outliner) che aprirà un pannello chiamato "*Lomboz J2EE View*". Da questo pannello effettuare il deploy dell'applicazione su Tomcat e avviarlo.



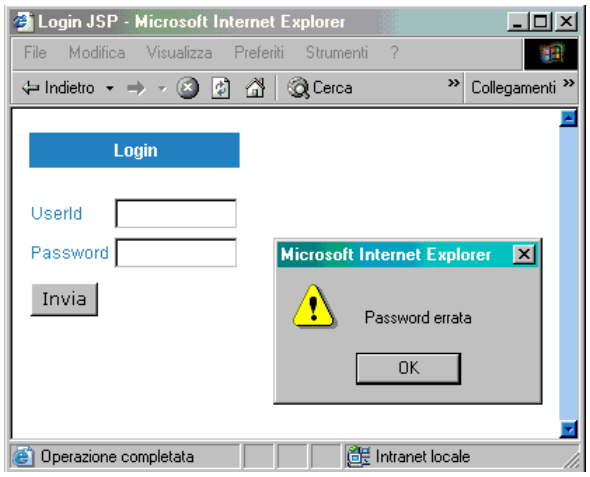
Avviato Tomcat, aprire il browser e digitare l'URL:

> http://localhost:8080/SpagoDemoWeb/servlet/AdapterHTTP?ACTION_NAME=PUBLISH&PUBLISHER=LOGIN&NEW_SESSION=TRUE

Verrà pubblicata la pagina di login:



In caso di dati errati inseriti nel form, verranno reperiti i corrispondenti messaggi d'errore dal corrispondente catalogo multilingua e presentati in una popup tramite una funzione Javascript. Ad esempio:



In caso di dati corretti verrà visualizzata la pagina di benvenuto che riporta il nome e cognome dell'utente appena autenticato.