Spago
Distribuzione dei Servizi su EJB

Redatto da Angelo Bernabei

INDICE DEI CONTENUTI

Scopo del Documento

In questo documento vengono presentate alcune linee guida su come impostare i progetti sviluppati con **Spago** utilizzando Jboss, in particolare viene approfondita l'eventualità di eseguire i servizi (*action* e *pages*) all'interno del container EJB di Jboss. Per lo sviluppo del caso di test è stato utilizzato il plug-in di Eclipse *MyEclipse*, ma le considerazioni sono del tutto generali e possono essere estese a qualsiasi altro ambiente di sviluppo integrato.

E' quindi possibile separare l'applicazione in due, la parte di presentazione e la parte dei servizi, attualmente questo viene fatto dal framework adottando un patter di *EJB Facade,* con lo scopo di eseguire le *action* e le *pages* all'interno del container EJB per sfruttarne le caratteristiche transazionali e di loadbalancing specifiche della tecnologia EJB. Per non gravare sulle prestazioni sono state utilizzate le interfacce locali per accedere ai metodi pubblici dell'EJB.

Informazioni sulla versione

Versione/Release n° :	1.1	**Data Versione/Release :**	29/10/2004
Descrizione modifiche:	Correzioni formali		

1 Caso di test

Come primo passo realizzo una semplice pagina JSP e una Action che realizzano un moltiplicatore per due di un numero per avere un semplice caso di test su cui operare.

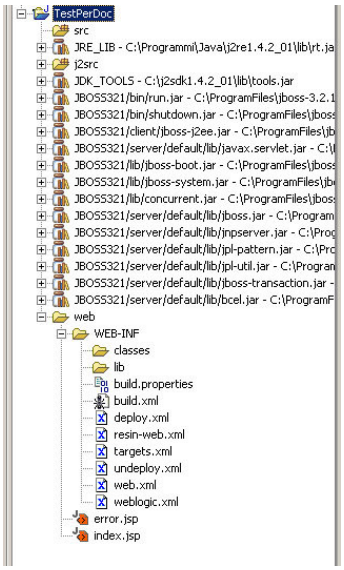
Ecco i passi che è necessario eseguire in sequenza:

Passo 1, creare un progetto per la parte web. Dal menu di Eclipse *File.new* selezionare la voce *Project*, apparirà una finestra dove sarà possibile selezionare la tipologia del progetto,

Passo 2, selezionando la tipologia *J2EE* troviamo sulla destra della finestra *Web module project*, selezioniamo e premiamo il bottone next.

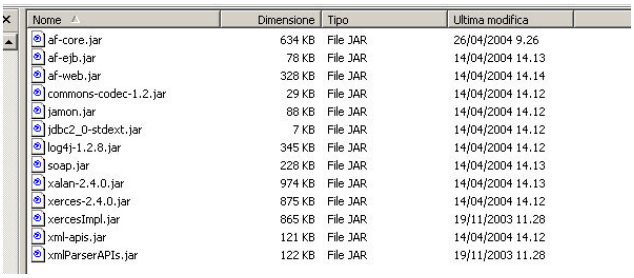
Passo 3, a questo punto viene richiesto il nome del progetto e alcune informazioni aggiuntive: nome del contesto, directory dei sorgenti e directory web; inseriamole e premiamo il bottone finish.

Su Eclipse sarà presente un nuovo progetto con la seguente struttura:



Una volta creato un progetto è necessario configurare l'operazione di deploy dal menù *Add and Remove Project Deployment* accessibile con il bottone destro direttamente sull'icona del progetto. Dalla finestra di configurazione sarà sufficiente indicare su quale server desideriamo effettuare il deploy. Il server dovrà essere già stato opportunamente configurato sulla configurazione di MyEclipse.

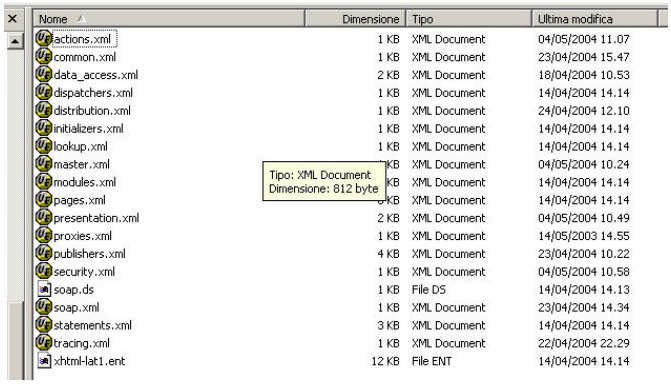
A questo punto è necessario inserire le librerie del framework (nella directory /web/WEB-INF/lib del progetto) e i suoi file di configurazione:



Nome	Dimensione	Tipo	Ultima modifica
af-core.jar	634 KB	File JAR	26/04/2004 9.26
af-ejb.jar	78 KB	File JAR	14/04/2004 14.13
af-web.jar	328 KB	File JAR	14/04/2004 14.14
commons-codec-1.2.jar	29 KB	File JAR	14/04/2004 14.12
jamon.jar	88 KB	File JAR	14/04/2004 14.12
jdbc2_0-stdext.jar	7 KB	File JAR	14/04/2004 14.12
log4j-1.2.8.jar	345 KB	File JAR	14/04/2004 14.12
soap.jar	228 KB	File JAR	14/04/2004 14.13
xalan-2.4.0.jar	974 KB	File JAR	14/04/2004 14.13
xerces-2.4.0.jar	875 KB	File JAR	14/04/2004 14.12
xercesImpl.jar	865 KB	File JAR	19/11/2003 11.28
xml-apis.jar	121 KB	File JAR	14/04/2004 14.12
xmlParserAPIs.jar	122 KB	File JAR	19/11/2003 11.28

aggiungendo le librerie al classpath del progetto di Eclipse.

Ecco i vari file di configurazione sotto la directory */web/WEB-INF/conf*:



Aggiungere il file web.xml allegato alla distribuzione del framework.

E' inoltre necessario accertarsi che la directory dove Eclipse posiziona il file .class generati durante la fase di compilazione sia */web/WEB-INF/classes*, questa operazione può essere fatta dalla finestra delle properties del progetto.

Si rimanda alla documentazione del framework la descrizione in dettaglio dei singoli file di configurazione xml.

Definiamo il servizio di test configurando i seguenti file xml:

action.xml

```
<ACTION class="com.engiweb.test.Sommatore" distributed="TRUE" name="Sommatore" scope="REQUEST">
        <CONFIG />
</ACTION>
```

presentation.xml

```
<MAPPING business_name="Sommatore" business_type="ACTION" publisher_name="SommatoreP" />
```

publischer.xml

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

distribution.xml

```
<DISTRIBUTIONS>
        <DISTRIBUTION business_type="ACTION" business_name="Sommatore"
distribute_coordinator_class="com.engiweb.framework.dispatching.distribute.DistributeCoordinator" />
</DISTRIBUTIONS>
```

Il framework è in grado di rimettere l'esecuzione dei servizi ad un EJB indipendentemente da come questi sono stati sviluppati, ma è necessario gestire la fase di deploy dell'applicativo in modo differente, aggiungendo un secondo progetto che contiene la configurazione dell'EJB e le librerie aggiuntive del framework. Per come funziona il classloader di Jboss, è necessario che le classi dei servizi siano posizionate all'interno di questo secondo progetto, che saranno comunque anche visibili dalla parte

web. E' inoltre necessario aggiungere nel file web.xml l'indicazione in quale file jar è presente il progetto EJB:

```
<ejb-local-ref id="EJBLocalRef">
        <ejb-ref-name>ejb/af/SessionFacadeEJBLocalHome</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <local-home>
           com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBLocalHome
        </local-home>
        <local>
           com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBLocal
        </local>
        <ejb-link>EJBFrameWork.jar#SessionFacadeEJB</ejb-link>

  </ejb-local-ref>
```

2 Progetto EJB

Per creare questo progetto utilizziamo il wizard messo a disposizione da *MyEclipse*, selezionando *l'EJB Module Project* dalla finestra di creazione di un nuovo progetto. Il progetto creato è molto semplice, sarà quindi necessario inserire le librerie *af-ejb.jar*, *af-core.jar* e *jamon.jar* nella cartella *src/META-INF* creata dal wizard, inserire i sorgenti nella cartella *src* e aggiungere i file di configurazione jboss.xml, ejb-jar.xml e beans.xml all'interno della cartella *META-INF*.

Eseguite queste operazioni è possibile configurare il deploy con le stesse modalità descritte per il progetto precedente.

La classe seguente implementa l'action:

```
package com.engiweb.test;


import com.engiweb.framework.base.SourceBean;
import com.engiweb.framework.dispatching.action.AbstractAction;


/**
 * @author bernabei
 */
public class Sommatore extends AbstractAction{



        public void service(SourceBean request, SourceBean response) throws Exception {
                String primo=(String)request.getAttribute("primo");
                String secondo=(String)request.getAttribute("secondo");
                int risultato=Integer.parseInt(primo)+ Integer.parseInt(secondo);
                ValueObjectTest vo=new ValueObjectTest();
                vo.setMsg(Integer.toString(risultato));
                response.setAttribute("risultato",vo);
        }
}
```

E' stata creata anche la seguente classe di utilità:

```
package com.engiweb.test;

/**
 * @author bernabei
 */
public class ValueObjectTest {
private String msg="";
/**
 * @return
 */
public String getMsg() {
        return msg;
}

/**
 * @param string
 */
public void setMsg(String string) {
        msg = string;
}
public String toString(){
        return msg;
}
}
```

Il risultato sarà pubblicato tramite la seguente pagina JSP indicata nel file di configurazione dei publisher:

```
<%@ page
                        contentType="text/html; charset=ISO-8859-1"
                        pageEncoding="ISO-8859-1"
                        session="true"
                        import="com.engiweb.framework.base.*,com.engiweb.test.ValueObjectTest"
%>

<%
ValueObjectTest risultato=null;
ResponseContainer responseContainer=ResponseContainerAccess.getResponseContainer(request);
if (responseContainer!=null){
        SourceBean sb =         responseContainer.getServiceResponse();
        if (sb!=null) risultato = (ValueObjectTest)sb.getAttribute("risultato");
}
```

```
if (risultato==null) risultato=new ValueObjectTest();
%>
<html>
	<head>
		<title>Sommatore...</title>
	</head>
	<body>
	<form name="forem" method="post" action="./servlet/AdapterHTTP?ACTION_NAME=Sommatore">
		<center>SOMMATORE...</center>
		Primo Numero : <input type="text" name="primo">
		Secondo Numero : <input type="text" name="secondo">
		<input type="submit" class="submit" value="Somma">
		</form>
		Risultato= <input type="text" name="risultato" value='<%=risultato.getMsg()%>'>
	</body>
</html>
```

A questo punto verifichiamo la correttezza del lavoro eseguendo una prova lanciando JBoss ed effettuando il deploy dei due progetti.

Scrivendo la seguente URL nel browser appare la pagina dove è presente il risultato:

http://localhost::8080/web/servlet/AdapterHTTP?ACTION_NAME=Sommatore&primo=2&secondo=2&NEW_SESSION=TRUE

3 *Appendice*

Ecco I file di configurazione che devono essere posizionati all'interno della cartella src/META-INF:

beans.xml

```
<beangroup>
    <bean>
        <beanClass>com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBBean</beanClass>
        <type>Session</type>
    </bean>
</beangroup>
```

ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar id="ejb-jar_ID">
	<display-name>AFDemoEJB</display-name>
	<enterprise-beans>
		<session id="AdapterEJB">
			<ejb-name>AdapterEJB</ejb-name>
			<home>com.engiweb.framework.dispatching.ejbchannel.AdapterEJBHome</home>
			<remote>com.engiweb.framework.dispatching.ejbchannel.AdapterEJB</remote>
			<ejb-class>com.engiweb.framework.dispatching.ejbchannel.AdapterEJBBean</ejb-class>
			<session-type>Stateful</session-type>
			<transaction-type>Bean</transaction-type>
```

```
		</session>
		<session id="SessionFacadeEJB">
			<ejb-name>SessionFacadeEJB</ejb-name>
			<local-home>com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBLocalHome</local-home>
			<local>com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBLocal</local>
			<ejb-class>com.engiweb.framework.dispatching.ejbchannel.SessionFacadeEJBBean</ejb-class>
			<session-type>Stateless</session-type>
			<transaction-type>Bean</transaction-type>
		</session>
		<session id="BusinessDelegateEJB">
			<ejb-name>BusinessDelegateEJB</ejb-name>
			<local-home>com.engiweb.framework.dispatching.ejbchannel.BusinessDelegateEJBLocalHome</local-home>
			<local>com.engiweb.framework.dispatching.ejbchannel.BusinessDelegateEJBLocal</local>
			<ejb-class>com.engiweb.framework.dispatching.ejbchannel.BusinessDelegateEJBBean</ejb-class>
			<session-type>Stateless</session-type>
			<transaction-type>Bean</transaction-type>
		</session>
	</enterprise-beans>
</ejb-jar>
```

jboss.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 3.0//EN" "http://www.jboss.org/j2ee/dtd/jboss_3_0.dtd">
<jboss>
	<unauthenticated-principal>nobody</unauthenticated-principal>
	<enterprise-beans>
		<session>
			<ejb-name>AdapterEJB</ejb-name>
			<jndi-name>ejb/af/AdapterEJBHome</jndi-name>
		</session>
		<session>
			<ejb-name>SessionFacadeEJB</ejb-name>
			<local-jndi-name>ejb/af/SessionFacadeEJBLocalHome</local-jndi-name>
		</session>
	</enterprise-beans>
	<resource-managers></resource-managers>

</jboss>
```