HOW-TO :

Création d'un Web Service Java avec AXIS 1.2

Déploiement sous Jboss 4

Création automatique d'un client à partir du WSDL

Tests du webservice avec JUnit

1 - Installation ECLIPSE 3.0.2 et MyEclipse sous Windows 2000/XP si ce n'est déjà fait

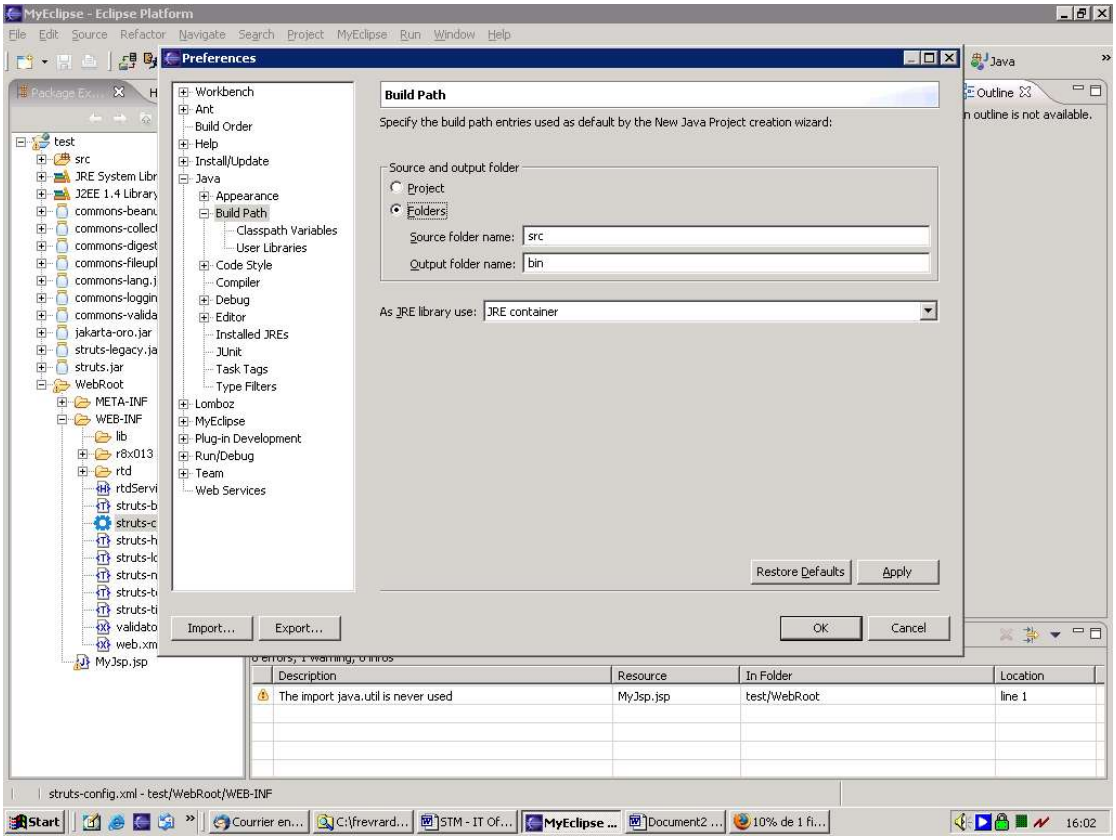
Plugins qui seront nécessaire :
MyEclipse
Webservice Client Wizard (Wsdl2Java)
VSS

Pré-requis :
eclipse-SDK-3.0.2-win32.zip2.zip
lavadora_winxp-1.0.0.zip
axis-bin-1_2RC3.zip
org.vssplugin_1.6.0.zip

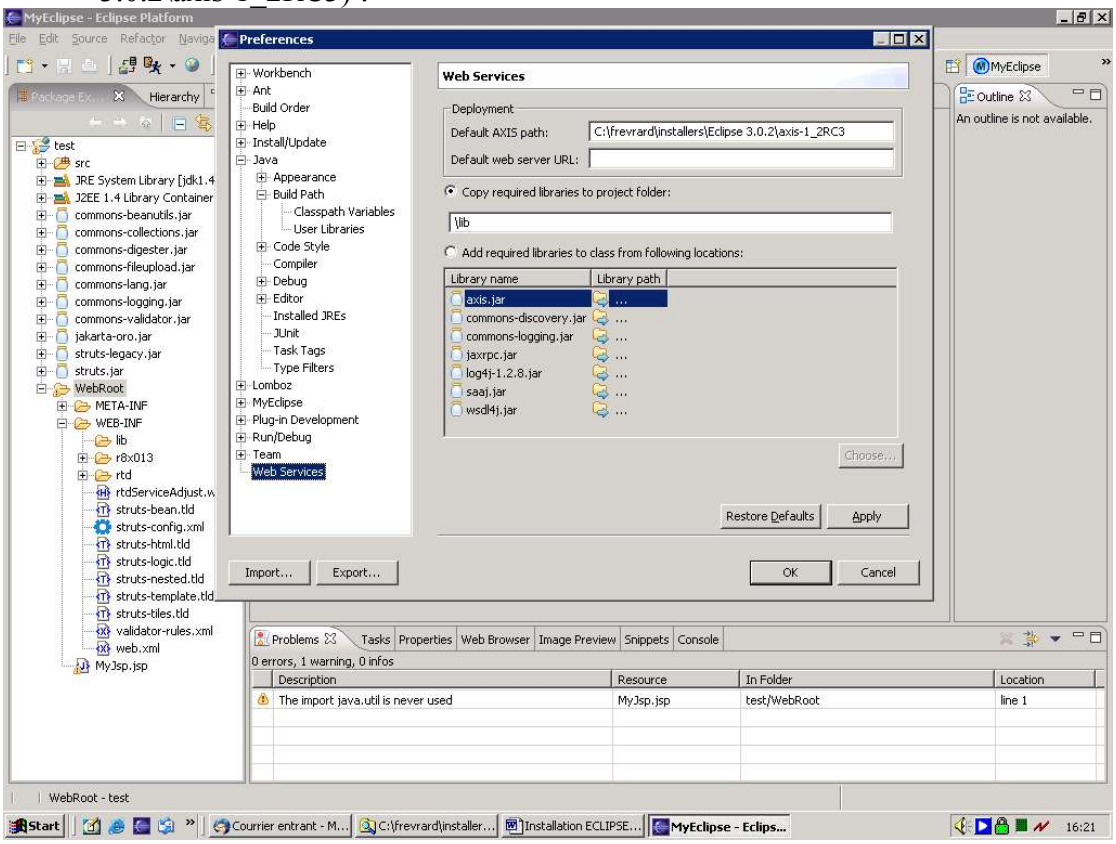
Ces packages sont disponibles sous \\Solutions\Automation-Fab App\ECLIPSE\Eclipse 3.0.2

installation

1. décompacter eclipse-SDK-3.0.2-win32.zip2.zip et le copier dans C:\applications\eclipse3.0.2
2. décompacter lavadora_winxp-1.0.0.zip et copier le contenu des repertoires plugins et features dans les repertoires respectifs plugins et features situés sous C:\applications\eclipse3.0.2
3. décompacter org.apache.axis_1.2beta3.zip dans un repertoire quelconque (ici C:\frevrard\installers\Eclipse 3.0.2\axis-1_2RC3) sous C:\applications\eclipse3.0.2
4. décompacter org.vssplugin_1.6.0.zip et copier le contenu des repertoires plugins et features dans les repertoires respectifs plugins et features situés sous C:\applications\eclipse3.0.2
5. Lancer MyEclipse à partir du bureau Windows
6. Dans le menu Préférence -> suivre la configuration suivante :

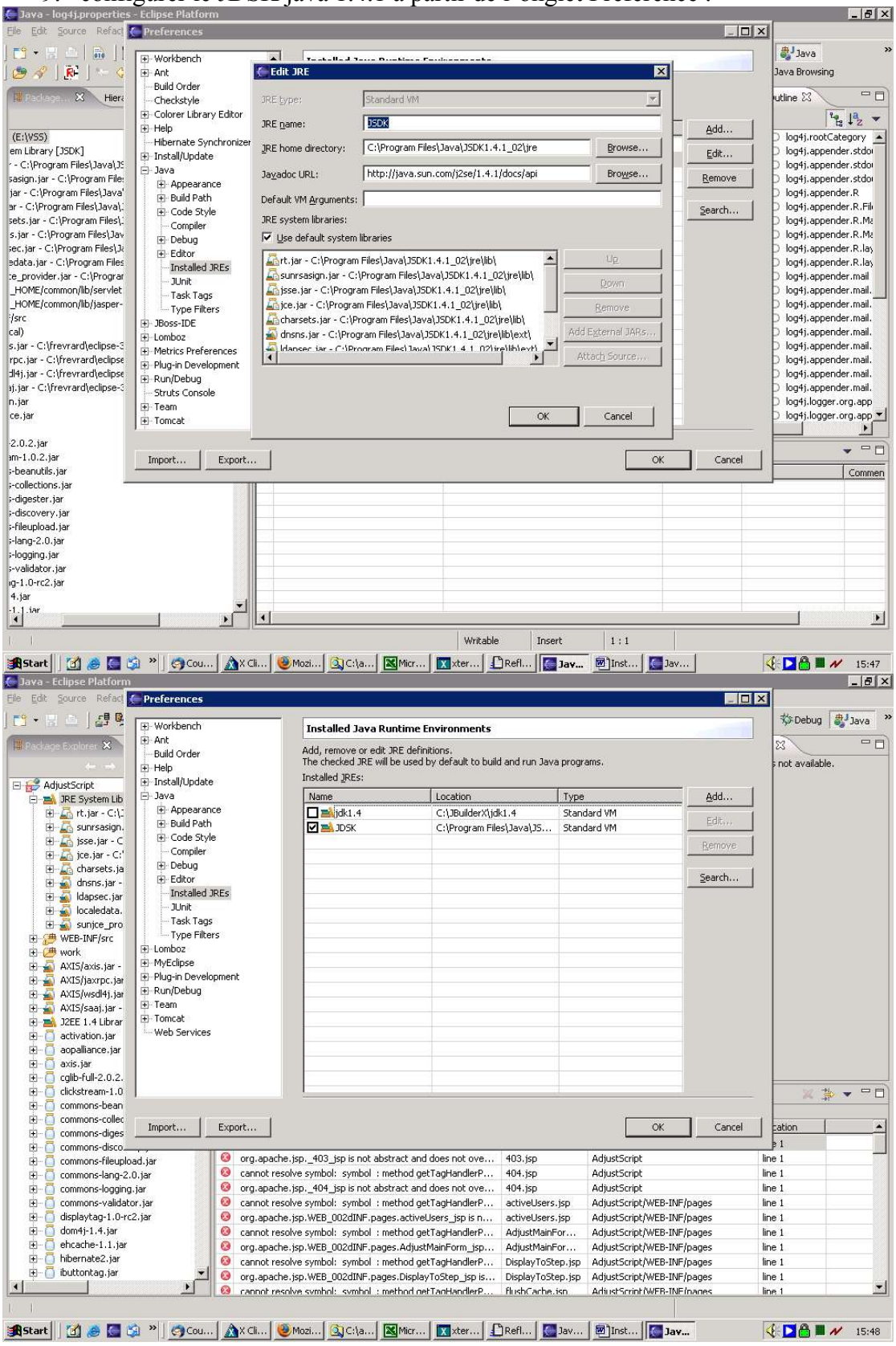


7. Sous le menu Préférence -> configurer le wizard Axis de la sorte en fonction de la localisation des librairies Axis (ici C:\frevrard\installers\Eclipse 3.0.2\axis-1_2RC3) :



8. Cliquer sur ‘Ok’

9. configurer le JDSK java 1.4.1 a partir de l'onglet Préférence :



10. Cliquer sur Ok pour terminer l'installation, relancer Eclipse. **Il est maintenant possible d'utiliser l'ensemble des fonctionnalités STRUTS, HIBERNATE de MyEclipse ainsi que de la génération automatique du client AXIS à partir d'un fichier WSDL (Fichier de description d'un WebService).**

2 – Création du WebService Java déployé sous Jboss

Liens très interessants :

Description de java2wsdl et ses arguments :
http://ws.apache.org/axis/java/reference.html

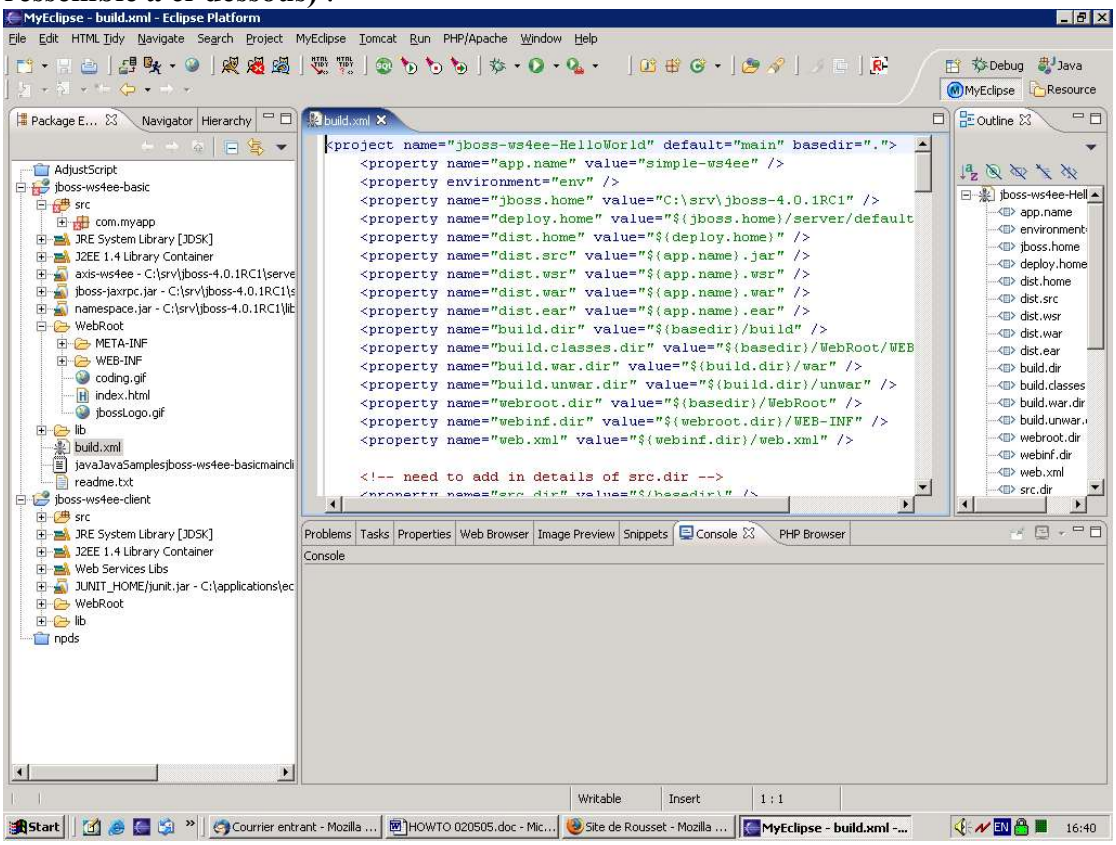
How-to utilisé pour la création du webservice :
http://www.csd.abdn.ac.uk/~bscharla/teaching/mtp_software/jboss/jboss-ws4ee-HelloWorld.shtml

Référence pour le descripteur de deploiement wsdd :
http://www.osmoticweb.com/axis-wsdd/

Console d'administration de JBoss :
http://localhost:8080/web-console/

Console de visualisation des ws de JBoss :
http://localhost:8080/ws4ee/services

Créer un projet Web sous MyEclipse nommé jboss-ws4ee-basic (la structure ressemble a ci-dessous) :



Incoporer dans le java build path du projet les librairies suivantes provenant de Jboss pour la génération du webservice :

```
C:/srv/jboss-4.0.1RC1/server/default/deploy/jboss-ws4ee.sar/axis-
ws4ee.jar
C:/srv/jboss-4.0.1RC1/server/default/lib/jboss-jaxrpc.jar
C:/srv/jboss-4.0.1RC1/lib/namespace.jar
```

Créer le fichier build.xml pour les taches de compilation ANT, très pratique pour compiler le projet sur d'autres configurations d'outils de développement (j'ai modifié ce fichier par rapport à loriginal car avec eclipse on utilise la compilation automatique, on n'a pas besoin de le compiler avec ant car c'est mieux ainsi. Par contre, on se sert de ant pour créer le war et le fichier wsdl avec encodage RPC/Literal). Ce fichier fait une copie des classes et fichier nécessaires à la création d'un war dans le répertoire 'build' qui sera automatiquement créé dans votre projet, il sera cependant nécessaire de modifier certaines chose (surligné en gras) :

```
<project name="jboss-ws4ee-HelloWorld" default="main" basedir=".">
      <property name="app.name" value="simple-ws4ee" />
      <property environment="env" />
      <property name="jboss.home" value="C:\srv\jboss-4.0.1RC1" />
      <property name="deploy.home" value="${jboss.home}/
server/default/deploy" />
      <property name="dist.home" value="${deploy.home}" />
      <property name="dist.src" value="${app.name}.jar" />
      <property name="dist.wsr" value="${app.name}.wsr" />
      <property name="dist.war" value="${app.name}.war" />
      <property name="dist.ear" value="${app.name}.ear" />
      <property name="build.dir" value="${basedir}/build" />
      <property name="build.classes.dir" value="${basedir}/
WebRoot/WEB-INF/classes" />
      <property name="build.war.dir" value="${build.dir}/war" />
      <property name="build.unwar.dir" value="${build.dir}/unwar" />
      <property name="webroot.dir" value="${basedir}/WebRoot" />
      <property name="webinf.dir" value="${webroot.dir}/WEB-INF" />
      <property name="web.xml" value="${webinf.dir}/web.xml" />

      <!-- need to add in details of src.dir -->
      <property name="src.dir" value="${basedir}" />

      <!-- the other jars for axis, servlets, etc are added to the
classpath
-->
      <path id="base.libraries">
            <pathelement location="${build.classes.dir}" />
            <fileset dir="${jboss.home}/lib">
                  <include name="*.jar" />
            </fileset>
            <fileset dir="${jboss.home}/server/default/lib">
                  <include name="*.jar" />
            </fileset>
            <fileset dir="${jboss.home}/server/default/deploy/jboss-
ws4ee.sar">
                  <include name="*.jar" />
            </fileset>
      </path>

      <path id="client.classpath">
            <pathelement location="${build.classes.dir}" />
            <fileset dir="${jboss.home}/client">
                  <include name="*.jar" />
            </fileset>
      </path>
```

```
	<target name="clean">
		<delete dir="${build.dir}" />
	</target>

	<target name="validate">
		<available property="classpath_id" value="base.libraries"
file="${jboss.home}/server/default/lib/javax.servlet.jar" />
	</target>

	<target name="fail_if_not_valid" unless="classpath_id">
		<fail message="jboss.home=${jboss.home} is not a valid
JBoss dist directory " />
	</target>

	<target name="prepare" depends="clean, validate,
fail_if_not_valid">
		<mkdir dir="${build.dir}" />
		<mkdir dir="${build.war.dir}" />

		<!-- automatic bluid par eclipse mkdir dir="${build.
classes.dir}" / -->
		<!-- mkdir dir="${build.classes.dir}/client" / -->
		<!-- mkdir dir="${build.classes.dir}/WEB-INF" / -->
		<!-- mkdir dir="${build.classes.dir}/WEB-INF/classes" /
-->
		<!-- mkdir dir="${build.classes.dir}/META-INF" / -->
		<!-- set the classpath for compiling files -->

		<property name="classpath" refid="${classpath_id}" />
		<!-- now copy across the files -->
		<!-- <copy file="main/testHelloWorld.java"
todir="${build.classes.dir}"/>  -->
		<!--
		<copy todir="${build.classes.dir}">
			<fileset dir="${src.dir}/WebRoot">
				<include name="**/*" />
			</fileset>
		</copy>

		<copy todir="${build.classes.dir}/WEB-INF">
			<fileset dir="${src.dir}/WebRoot/WEB-INF">
				<include name="**/*" />
			</fileset>
		</copy>
		-->
	</target>

	<target name="compile" depends="prepare">
		<!--
		<javac srcdir="main" destdir="${build.classes.dir}"
classpath="${build.classes.dir}" debug="on" optimize="off"
deprecation="on">
			<classpath path="${classpath}" />
		</javac>
		-->
		<!--
		<copy todir="">
			<fileset dir="${src.dir}/main/src">
				<include name="main/src/testHelloWorld.*" />
			</fileset>
		</copy>
		-->
	</target>

	<target name="javadoc" depends="prepare">
		<!-- automatic
```

```
            <javadoc sourcepath="src" packagenames="*"
destdir="${javadoc.home}" />
            -->
      </target>

      <target name="wsdl" depends="compile">
            <!-- mkdir dir="${build.classes.dir}/WEB-INF/wsdl" / -->
            <java classname="org.apache.axis.wsdl.Java2WSDL"
fork="yes" dir=".">
                  <classpath path="${classpath}" />
                  <!--
                  <arg value="-lhttp://
localhost:8080/ws4ee/services/HelloWorld" />
                  <arg value="-sHelloWorld" />
                  -->
                  <arg value="-lhttp://localhost:8080/simple-
ws4ee/exactpath/jse/" />
                  <arg value="-sHelloWorld" />
                  <arg value="-oWebRoot/WEB-INF/wsdl/hello.wsdl" />
                  <arg value="-yRPC" />
                  <arg value="-uLITERAL" />
                  <arg value="com.myapp.HelloWorld" />
            </java>
      </target>

      <!-- now build the war file -->
      <target name="war" depends="wsdl">
            <war destfile="${build.war.dir}/${dist.war}"
webxml="${webinf.dir}/web.xml">
                  <fileset dir="${webroot.dir}" />
                  <webinf dir="${webinf.dir}">
                        <exclude name="web.xml" />
                        <include name="*.xml" />
                        <include name="wsdl/*.*" />
                  </webinf>
                  <classes dir="${build.classes.dir}">
                        <include name="com/myapp/*.class" />
                  </classes>
            </war>
            <!-- copy file="${build.dir}/${dist.war}" todir="${build.
classes.dir}" / -->
      </target>

      <!-- for ease of checking that all of the contents are correct
-->
      <target name="unwar" depends="war">
            <mkdir dir="${build.unwar.dir}" />
            <unwar src="${build.war.dir}/${dist.war}" dest="${build.
unwar.dir}" />
      </target>

      <target name="dist" depends="unwar">
            <copy file="${build.war.dir}/${dist.war}"
todir="${deploy.home}" />
      </target>

      <target name="run_client">
            <!-- copy file="${src.dir}/com/myapp/log4j.xml"
todir="${build.classes.dir}" / -->
            <java classname="com.myapp.DIIClient" fork="yes" dir=".">
                  <classpath refid="client.classpath" />
            </java>
      </target>

      <target name="main" depends="run_client">
```

```
      </target>

</project>
```

Ensuite créer le package com.myapp ainsi que les classes suivantes :

```
package com.myapp;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface HelloWorld extends Remote {
      public String getHelloWorld(String name) throws
RemoteException;
}
```

puis :

```
package com.myapp;

import java.rmi.RemoteException;
import javax.xml.rpc.*;
import javax.xml.rpc.server.ServiceLifecycle;

public class HelloWorld_Impl implements HelloWorld, ServiceLifecycle
{


    public String getHelloWorld(String name)throws RemoteException {
      return "Hello world to " + name;
    }

      /* (non-Javadoc)
       * @see javax.xml.rpc.server.ServiceLifecycle#init
(java.lang.Object)
       */
      public void init(Object arg0) throws ServiceException {
            // TODO Auto-generated method stub

      }

      /* (non-Javadoc)
       * @see javax.xml.rpc.server.ServiceLifecycle#destroy()
       */
      public void destroy() {
            // TODO Auto-generated method stub

      }

      /* (non-Javadoc)
       * @see HelloWorld#getHelloWorld(java.lang.String)
       */

}
```

puis la classe de test sommaire :

```
package com.myapp;

import javax.xml.namespace.QName;
import javax.xml.rpc.Call;
import javax.xml.rpc.Service;
import javax.xml.rpc.JAXRPCException;
```

```
import javax.xml.rpc.ServiceFactory;
import javax.xml.rpc.ParameterMode;

public class DIIClient {

      // modified from sun j2ee jaxrpc example

      private static String endpoint = "http://localhost:8080/simple-
ws4ee/exactpath/jse";
      private static String qnameService = "HelloWorldService";
    private static String qnamePort = "HelloWorld";

    private static String ENCODING_STYLE_PROPERTY =
         "javax.xml.rpc.encodingstyle.namespace.uri";
    private static String NS_XSD =
        "http://www.w3.org/2001/XMLSchema";
    private static String URI_ENCODING =
         "http://schemas.xmlsoap.org/soap/encoding/";

    public static void main(String[] args) {

        System.out.println("Endpoint address = " + endpoint);

        try {
            ServiceFactory factory = ServiceFactory.newInstance();
            Service service = factory.createService(new QName
(qnameService));

            QName port = new QName(qnamePort);

            Call call = service.createCall(port);
            call.setTargetEndpointAddress(endpoint);

            call.setProperty(Call.SOAPACTION_USE_PROPERTY,
                new Boolean(true));
            call.setProperty(Call.SOAPACTION_URI_PROPERTY, "");
            call.setProperty(ENCODING_STYLE_PROPERTY, URI_ENCODING);
            QName QNAME_TYPE_STRING = new QName(NS_XSD, "string");
            call.setReturnType(QNAME_TYPE_STRING);


            call.setOperationName(new QName(endpoint,
                "getHelloWorld"));
            call.addParameter("String_1", QNAME_TYPE_STRING,
                ParameterMode.IN);
            String[] params = { "jboss!" };

            String result = (String)call.invoke(params);
            System.out.println(result);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Nous allons créer le fichier log4j.xml dans le package com.myapp contenant, pas vraiement nécessaire pour la suite mais utile :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<!--
================================================================================
-->
```

```
<!--
-->
<!--  Log4j Configuration
-->
<!--
-->
<!--
================================================================================
-->

<!-- $Id: log4j.xml,v 1.24 2004/05/06 16:15:03 tdiesler Exp $ -->

<!--
   | For more configuration infromation and examples see the Jakarta
Log4j
   | owebsite: http://jakarta.apache.org/log4j
 -->

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/"
debug="false">

   <!-- ================================== -->
   <!-- Preserve messages in a local file -->
   <!-- ================================== -->

   <!-- A time/date based rolling appender -->
   <appender name="FILE"
class="org.jboss.logging.appender.DailyRollingFileAppender">
      <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
      <param name="File" value="C:\java\JavaSamples\jboss-ws4ee-
basic\main\client\client.log"/>
      <param name="Append" value="false"/>

      <!-- Rollover at midnight each day -->
      <param name="DatePattern" value="'.'yyyy-MM-dd"/>

      <!-- Rollover at the top of each hour
      <param name="DatePattern" value="'.'yyyy-MM-dd-HH"/>
      -->

      <layout class="org.apache.log4j.PatternLayout">
         <!-- The default pattern: Date Priority [Category] Message\n
-->
         <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>

         <!-- The full pattern: Date MS Priority [Category]
(Thread:NDC) Message\n
         <param name="ConversionPattern" value="%d %-5r %-5p [%c] (%
t:%x) %m%n"/>
          -->
      </layout>
   </appender>

   <!-- A size based file rolling appender
   <appender name="FILE"
class="org.jboss.logging.appender.RollingFileAppender">
     <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
     <param name="File" value="C:\java\JavaSamples\jboss-ws4ee-
basic\main\client\client.log"/>
     <param name="Append" value="false"/>
     <param name="MaxFileSize" value="500KB"/>
     <param name="MaxBackupIndex" value="1"/>

     <layout class="org.apache.log4j.PatternLayout">
```

```
      <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
    </layout>
  </appender>
  -->

  <!-- ================================ -->
  <!-- Append messages to the console -->
  <!-- ================================ -->

  <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
    <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="Target" value="System.out"/>
    <param name="Threshold" value="INFO"/>

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] Message\n
-->
      <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c
{1}] %m%n"/>
    </layout>
  </appender>

  <appender name="JSR77" class="org.apache.log4j.FileAppender">
    <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="Append" value="false"/>
    <param name="File" value="${jboss.server.home.dir}/
log/jsr77.log"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d{ABSOLUTE} %-5p [%c
{1}] %m%n"/>
    </layout>
  </appender>

  <!-- ====================== -->
  <!-- More Appender examples -->
  <!-- ====================== -->

  <!-- Buffer events and log them asynchronously
  <appender name="ASYNC" class="org.apache.log4j.AsyncAppender">
    <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <appender-ref ref="FILE"/>
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="SMTP"/>
  </appender>
  -->

  <!-- EMail events to an administrator
  <appender name="SMTP" class="org.apache.log4j.net.SMTPAppender">
    <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="Threshold" value="ERROR"/>
    <param name="To" value="admin@myhost.domain.com"/>
    <param name="From" value="nobody@myhost.domain.com"/>
    <param name="Subject" value="JBoss Sever Errors"/>
    <param name="SMTPHost" value="localhost"/>
    <param name="BufferSize" value="10"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="[%d{ABSOLUTE},%c{1}] %
m%n"/>
    </layout>
  </appender>
  -->
```

```
   <!-- Syslog events
   <appender name="SYSLOG"
class="org.apache.log4j.net.SyslogAppender">
     <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
     <param name="Facility" value="LOCAL7"/>
     <param name="FacilityPrinting" value="true"/>
     <param name="SyslogHost" value="localhost"/>
   </appender>
   -->

   <!-- Log events to JMS (requires a topic to be created)
   <appender name="JMS" class="org.apache.log4j.net.JMSAppender">
     <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
     <param name="Threshold" value="ERROR"/>
     <param name="TopicConnectionFactoryBindingName"
value="java:/ConnectionFactory"/>
     <param name="TopicBindingName" value="topic/MyErrorsTopic"/>
   </appender>
   -->

   <!-- Log events through SNMP
   <appender name="TRAP_LOG"
class="org.apache.log4j.ext.SNMPTrapAppender">
     <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
     <param name="ImplementationClassName"
value="org.apache.log4j.ext.JoeSNMPTrapSender"/>
     <param name="ManagementHost" value="127.0.0.1"/>
     <param name="ManagementHostTrapListenPort" value="162"/>
     <param name="EnterpriseOID" value="1.3.6.1.4.1.24.0"/>
     <param name="LocalIPAddress" value="127.0.0.1"/>
     <param name="LocalTrapSendPort" value="161"/>
     <param name="GenericTrapType" value="6"/>
     <param name="SpecificTrapType" value="12345678"/>
     <param name="CommunityString" value="public"/>
     <param name="ForwardStackTraceWithTrap" value="true"/>
     <param name="Threshold" value="DEBUG"/>
     <param name="ApplicationTrapOID"
value="1.3.6.1.4.1.24.12.10.22.64"/>
     <layout class="org.apache.log4j.PatternLayout">
           <param name="ConversionPattern" value="%d,%p,[%t],[%c],%
m%n"/>
     </layout>
   </appender>
   -->

   <!-- ================ -->
   <!-- Limit categories -->
   <!-- ================ -->

   <!-- Limit the org.apache category to INFO as its DEBUG is verbose
-->
   <category name="org.apache">
      <priority value="INFO"/>
   </category>

   <!-- Limit the org.jgroups category to WARN as its INFO is verbose
-->
   <category name="org.jgroups">
      <priority value="WARN"/>
   </category>

   <!-- Limit apache axis to INFO as its DEBUG is even more verbose
-->
```

```
   <category name="org.apache.axis">
      <priority value="INFO"/>
   </category>

   <!-- Limit JBoss categories
   <category name="org.jboss">
      <priority value="INFO"/>
   </category>
   -->

   <!-- Limit JBoss webservice category -->
   <category name="org.jboss.webservice">
      <priority value="DEBUG"/>
   </category>


   <!-- Decrease the priority threshold for the org.jboss.varia
category
   <category name="org.jboss.varia">
     <priority value="DEBUG"/>
   </category>
   -->

   <!-- Show the evolution of the DataSource pool in the logs
[inUse/Available/Max]
   <category
name="org.jboss.resource.connectionmanager.JBossManagedConnectionPool
">
     <priority value="TRACE" class="org.jboss.logging.XLevel"/>
   </category>
   -->

   <!--
      | An example of enabling the custom TRACE level priority that
is used
      | by the JBoss internals to diagnose low level details. This
example
      | turns on TRACE level msgs for the org.jboss.ejb.plugins
package and its
      | subpackages. This will produce A LOT of logging output.
   <category name="org.jboss.system">
     <priority value="TRACE" class="org.jboss.logging.XLevel"/>
   </category>
   <category name="org.jboss.ejb.plugins">
     <priority value="TRACE" class="org.jboss.logging.XLevel"/>
   </category>
   -->

   <!--
       | Logs these events to SNMP:
           - server starts/stops
           - cluster evolution (node death/startup)
           - When an EJB archive is deployed (and associated verified
messages)
           - When an EAR archive is deployed

   <category name="org.jboss.system.server.Server">
     <priority value="INFO" />
     <appender-ref ref="TRAP_LOG"/>
   </category>

   <category
name="org.jboss.ha.framework.interfaces.HAPartition.lifecycle">
     <priority value="INFO" />
     <appender-ref ref="TRAP_LOG"/>
   </category>
```

```
   <category name="org.jboss.deployment.MainDeployer">
     <priority value="ERROR" />
     <appender-ref ref="TRAP_LOG"/>
   </category>

   <category name="org.jboss.ejb.EJBDeployer">
     <priority value="INFO" />
     <appender-ref ref="TRAP_LOG"/>
   </category>

   <category name="org.jboss.deployment.EARDeployer">
     <priority value="INFO" />
     <appender-ref ref="TRAP_LOG"/>
   </category>


   -->


   <!-- ====================== -->
   <!-- Setup the Root category -->
   <!-- ====================== -->

   <root>
      <appender-ref ref="CONSOLE"/>
      <appender-ref ref="FILE"/>
   </root>

   <!-- Clustering logging -->
   <!-- Uncomment the following to redirect the org.jgroups and
      org.jboss.ha categories to a cluster.log file.

   <appender name="CLUSTER"
class="org.jboss.logging.appender.RollingFileAppender">
     <errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
     <param name="File" value="${jboss.server.home.dir}/
log/cluster.log"/>
     <param name="Append" value="false"/>
     <param name="MaxFileSize" value="500KB"/>
     <param name="MaxBackupIndex" value="1"/>

     <layout class="org.apache.log4j.PatternLayout">
       <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
     </layout>
   </appender>
   <category name="org.jgroups">
     <priority value="DEBUG" />
     <appender-ref ref="CLUSTER"/>
   </category>
   <category name="org.jboss.ha">
     <priority value="DEBUG" />
     <appender-ref ref="CLUSTER"/>
   </category>
   -->

</log4j:configuration>
```

Ensuite nous allons créer les descripteurs de deploiement pour Jboss qui devront être placés sous WebRoot/WEB-INF, pour avoir des infos là-dessus, cf. liens interessants ci-avant :

/jboss-ws4ee-basic/WebRoot/WEB-INF/jaxrpc-mapping.xml

```
<?xml version='1.0' encoding='UTF-8' ?>
<java-wsdl-mapping
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                http://www.ibm.com/webservices/xsd/j2ee_jaxrpc_mapping
_1_1.xsd"
  version="1.1">
  <package-mapping>
    <package-type>com.myapp</package-type>
    <namespaceURI>http://myapp.com/HelloWorld</namespaceURI>
  </package-mapping>
</java-wsdl-mapping>
```

/jboss-ws4ee-basic/WebRoot/WEB-INF/jboss-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss-web PUBLIC "-//JBoss//DTD Web Application 2.2//EN"
"http://www.jboss.org/j2ee/dtd/jboss-web.dtd">

<jboss-web>

   <!-- Resource references -->

   <!-- EJB References -->

</jboss-web>
```

/jboss-ws4ee-basic/WebRoot/WEB-INF/web.xml

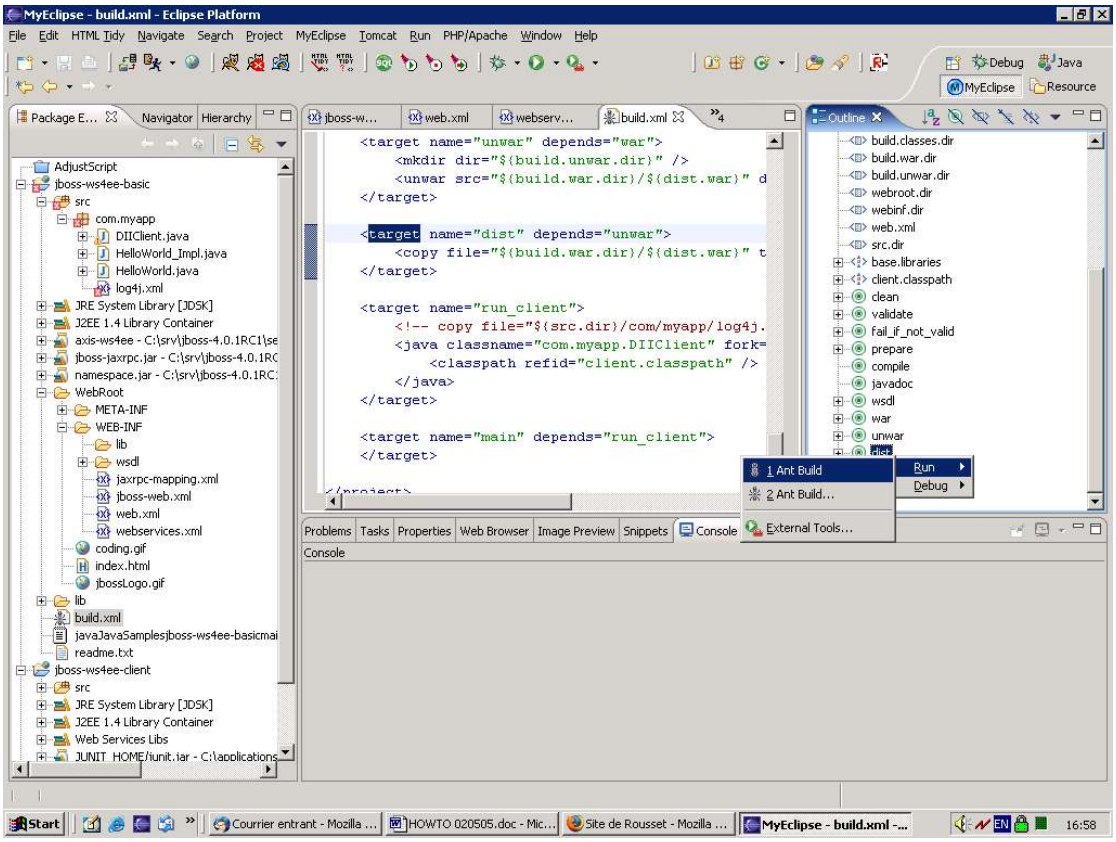
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
       xmlns="http://java.sun.com/xml/ns/j2ee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
     version="2.4">

      <servlet>
            <servlet-name>HelloWorldServlet</servlet-name>
            <servlet-class>com.myapp.HelloWorld_Impl</servlet-class>
      </servlet>

      <servlet-mapping>
      <servlet-name>HelloWorldServlet</servlet-name>
      <url-pattern>/exactpath/jse</url-pattern>
   </servlet-mapping>

   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
   </welcome-file-list>

</web-app>
```

/jboss-ws4ee-basic/WebRoot/WEB-INF/webservices.xml

```
<?xml version='1.0' encoding='UTF-8' ?>
<webservices
   xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:impl="http://com.myapp/ws4ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                 http://www.ibm.com/webservices/xsd/j2ee_web_services
 1 1.xsd"
   version="1.1">
```

```
      <webservice-description>
         <webservice-description-name>HelloWorldService</webservice-
description-name>
         <wsdl-file>WEB-INF/wsdl/hello.wsdl</wsdl-file>
         <jaxrpc-mapping-file>WEB-INF/jaxrpc-mapping.xml</jaxrpc-
mapping-file>
         <port-component>
            <port-component-name>HelloWorld</port-component-name>
            <wsdl-port>HelloWorld</wsdl-port>
            <service-endpoint-
interface>com.myapp.HelloWorld</service-endpoint-interface>
            <service-impl-bean>
               <servlet-link>HelloWorldServlet</servlet-link>
            </service-impl-bean>
         </port-component>
      </webservice-description>
</webservices>
```

Voila le webservice est créé, lancer JBoss (ligne de commande dans le répertoire de l'installation de Jboss). Puis compiler le war et générer le fichier wsdl en selectionnant le fichier build.xml et en cliquant sur la tache 'dist' dans la vue 'outline' de ant :

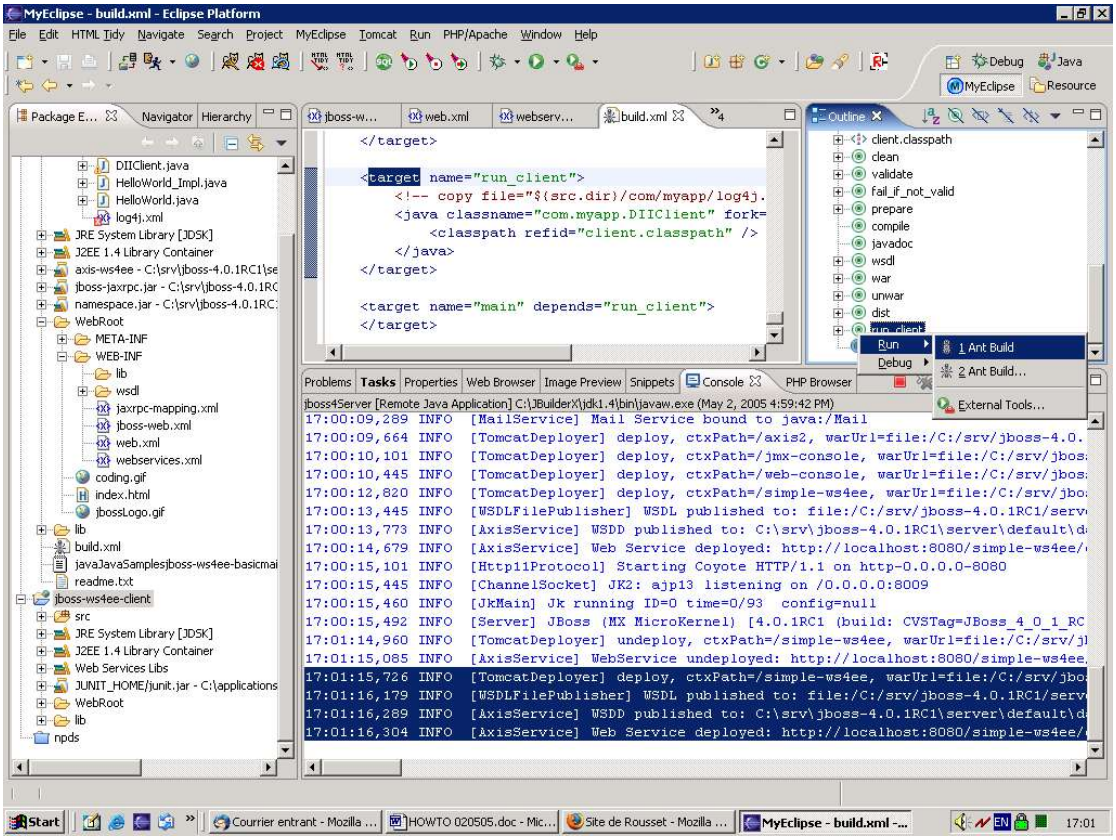


Le log montre la trace suivante :

```
clean:
validate:
fail_if_not_valid:
prepare:
    [mkdir] Created dir:
C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-basic\build
    [mkdir] Created dir:
C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-basic\build\war
compile:
```

```
wsdl:
war:
      [war] Building war:
C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-
basic\build\war\simple-ws4ee.war
      [war] Warning: selected war files include a WEB-INF/web.xml
which will be ignored (please use webxml attribute to war task)
unwar:
    [mkdir] Created dir:
C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-basic\build\unwar
    [unwar] Expanding: C:\applications\eclipse3.0.2\workspace\jboss-
ws4ee-basic\build\war\simple-ws4ee.war into
C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-basic\build\unwar
dist:
     [copy] Copying 1 file to C:\srv\jboss-
4.0.1RC1\server\default\deploy
BUILD SUCCESSFUL
Total time: 3 seconds

17:01:15,726 INFO  [TomcatDeployer] deploy, ctxPath=/simple-ws4ee,
warUrl=file:/C:/srv/jboss-
4.0.1RC1/server/default/tmp/deploy/tmp45318simple-ws4ee-exp.war/
17:01:16,179 INFO  [WSDLFilePublisher] WSDL published to:
file:/C:/srv/jboss-4.0.1RC1/server/default/data/wsdl/simple-
ws4ee.war/hello.wsdl
17:01:16,289 INFO  [AxisService] WSDD published to: C:\srv\jboss-
4.0.1RC1\server\default\data\wsdl\simple-
ws4ee.war\HelloWorldService.wsdd
17:01:16,304 INFO  [AxisService] Web Service deployed:
http://localhost:8080/simple-ws4ee/exactpath/jse
```

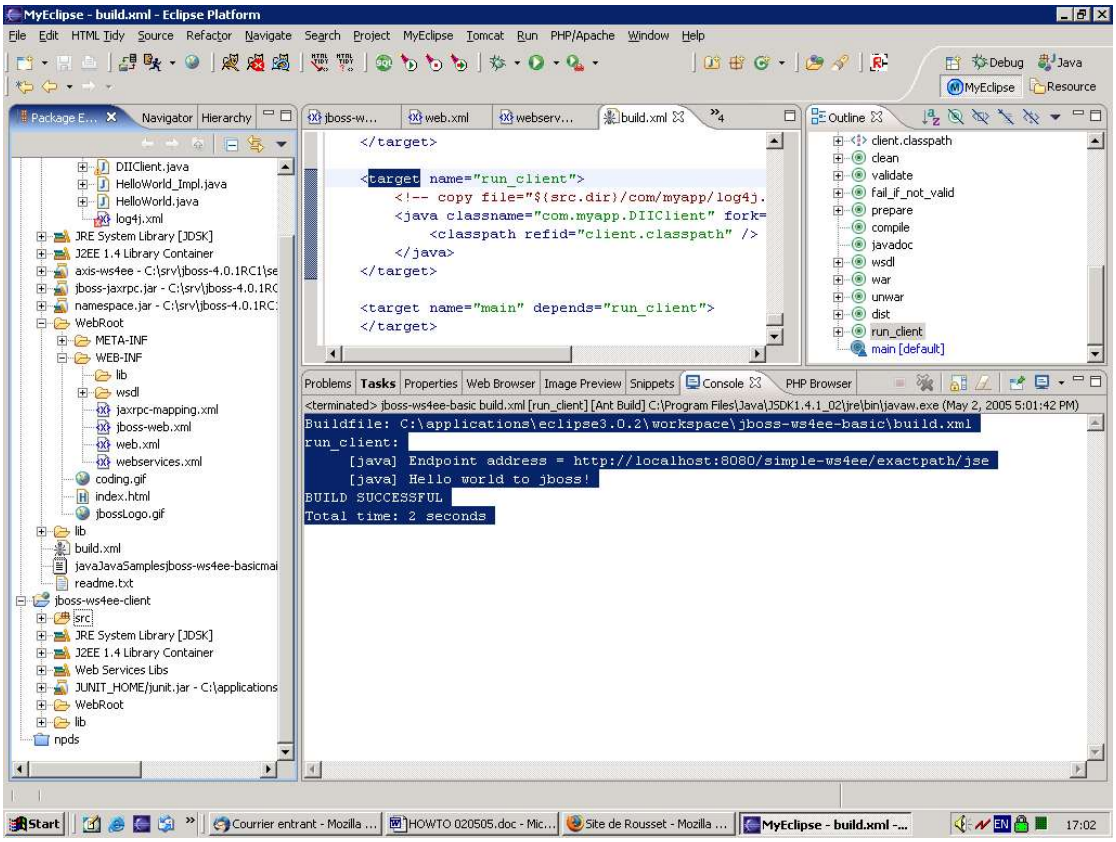
Maintenant que le webservice est deployé nous allons faire un test sommaire en selectionnant le fichier build.xml et en cliquant sur la tache 'run_client' dans la vue 'outline' de ant :



Le log généré montre :

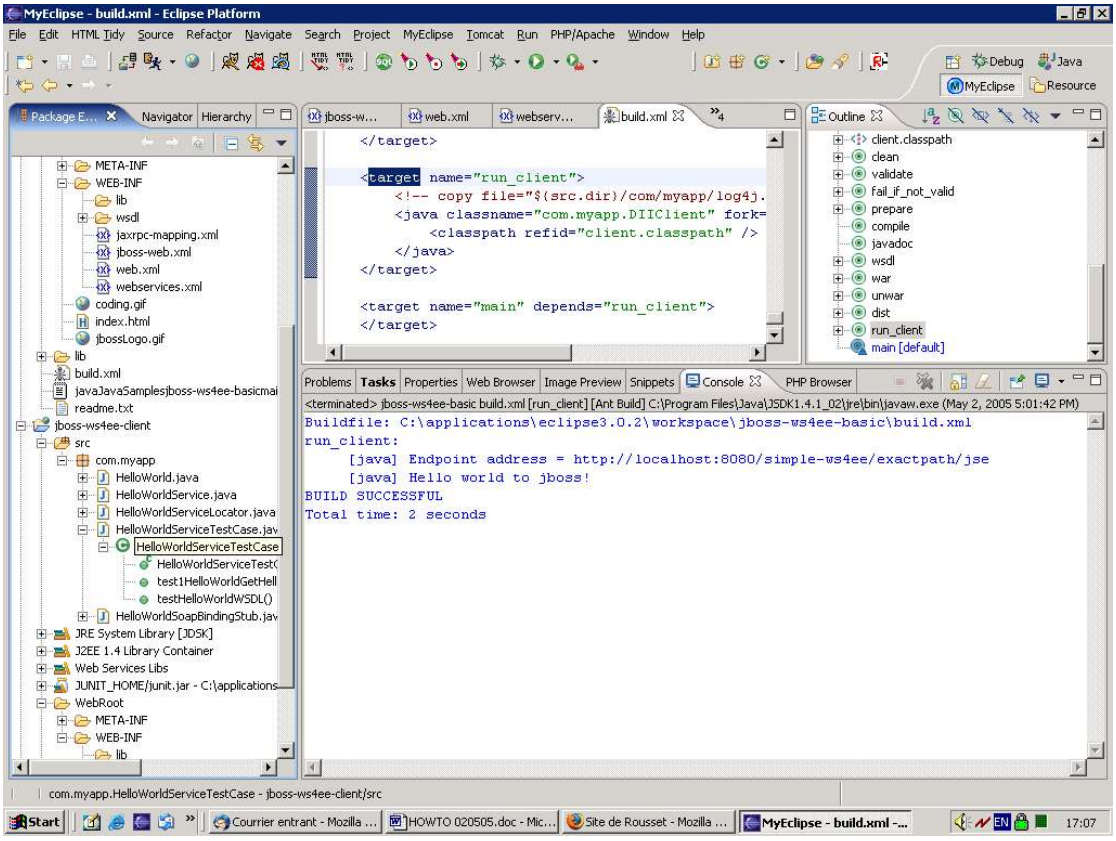
```
Buildfile: C:\applications\eclipse3.0.2\workspace\jboss-ws4ee-
basic\build.xml
run_client:
     [java] Endpoint address = http://localhost:8080/simple-
ws4ee/exactpath/jse
     [java] Hello world to jboss!
BUILD SUCCESSFUL
Total time: 2 seconds
```

Nous allons maintenant générer un client plus complet avec JUNIT, il faut pour cela créer un projet MyEclipse de type Web nommé 'jboss-ws4ee-client', sa structure doit ressembler à celle-ci :

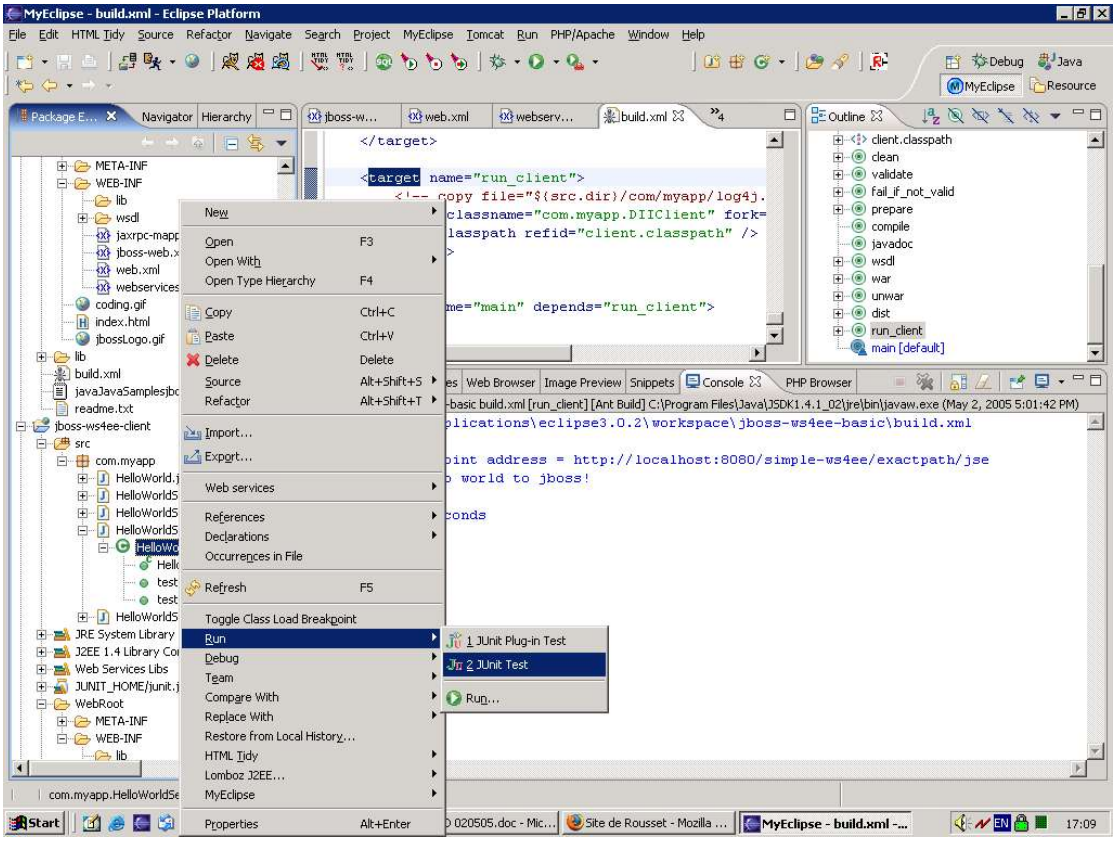


On récupere le fichier wsdl deployé à partir de la console Jboss http://localhost:8080/ws4ee/services ou en le copiant directement depuis le projet ws vers notre projet client dans le repertoire /jboss-ws4ee-client/WebRoot/WEB-INF/hello.wsdl par exemple (son emplacement n'est pas important il faut simplement le copier dans notre projet client). Ensuite on clic-droit sur ce fichier /jboss-ws4ee-client/WebRoot/WEB-INF/hello.wsdl et on selectionne 'Axis-WebService -> WSDL2java' (grâce au plugin lavadora_winxp-1.0.0).

Les classes de test sont ainsi créées :



Faire un clic-droit sur la classe /jboss-ws4ee-client/src/com/myapp/HelloWorldServiceTestCase.java et cliquer ensuite sur run -> junitTest :



Le test est lancé et affiché dans la console eclipse, c'est fini.