Getting Started with jUDDI

Alan Vinh
Phil Bonderud

Preface

Some instructions where pieced together various documents to create installation guide for installing JUDDI with Tomcat 5.0, MySQL, and Eclipse 3.2. Your install can be tested by using the JUDDI validation tool as well as creating a simple Web Service in Eclipse to access, store, and retrieve information.

Edited by Kurt Stam, Tom Cunningham

Requirements

Using Windows XP Professional

```
Tomcat-5.5.20  (http://tomcat.apache.org/download-55.cgi)
          Binary Distributions, Core, using Windows Service Installer

MySQL-5.0.26-win32  (http://dev.mysql.com/downloads/mysql/5.0.html)
          Windows (x86) file.

JDK 5.0 Update 10 (http://java.sun.com/javase/downloads/index.jsp)
              http://java.sun.com/javase/downloads/index_jdk5.jsp
          Includes Java Runtime Environment (JRE)
          Use Windows Online Installation download

jUDDI0.9rc4  ( http://ws.apache.org/juddi/releases.html)


Eclipse 3.2.1 http://www.eclipse.org
              eclipse-SDK-3.2.1-win32.zip
```

Install JDK 5.0

Download program, run installation, and accept default installation path. Install the full program.

After successful installation create/modify the JAVA_HOME environmental variable. Locate/Identify the full path to the new Java JDK 5.0 download. The default is C:\Program Files\Java\jdk1.5.0_10.

Click Start, Settings, Control Panel, System. On the Advanced tab click the Environment Variables button. Edit JAVA_HOME and add this to the end of the existing line (if it exists, otherwise create a new variable with this information and without the leading semicolon):

```
  ; C:\Program Files\Java\jdk1.5.0_10
```

Install Apache-Tomcat-5.5.20

Install and read the documentation for the program. Throughout the Tomcat documentation you will see references to $CATALINA_HOME. When you install Tomcat, a folder will be created which will hold the various subfolders and files. The default folder settings for Tomcat is:

```
C:\Apache Software Foundation\Tomcat 5.5\blah blah blah
```

The root folder for Tomcat, which is also called $CATALINA_HOME, in this example is:

```
C:\Apache Software Foundation\Tomcat 5.5
```

The default settings for Tomcat's installation should work fine, however this scenario included installing the full program. The setup steps are:

- Obtain the program files listed above.
- The first prompt is simply to click Next.
- Agree/Accept the terms.
- Select Full install and click Next. A custom install (default) will probably work however within the environment used for this tutorial a full installation was selected.
- Accept the default destination folder: C:\Program Files\Apache Software Foundation\Tomcat 5.5 and click Next
- Leave the default port (8080) as is for now. Change the User Name and/or enter a password if you desire. You will need this information to access. Tomcat's management tools. Entering a password is optional. Click Next.
- Verify the path to the Java installation you just performed. If you accepted Java's default settings you should see: C:\Program Files\Java\jre1.5.0_10. Click Install.
- Accept the final defaults and click Finish. Tomcat should start and the readme file may provide additional information for your reading pleasure.

Tomcat's toolbar icon looks like this: Right clicking on it enables you to stop and restart Tomcat as needed. Usually, when Tomcat is running the center of the icon is a green rectangle. Tomcat's documentation states that its icon itself is not an accurate tool for determining whether the server is running. You will need to right click the icon to be sure.

Set your $CATALINA_HOME environmental variable to point to the root folder of where you installed Tomcat. If you are running more than one version of Tomcat on your home computer you may want to read up on $CATALINA_BASE. $CATALINA_BASE may be necessary, instead of $CATALINA_HOME, when multiple versions of Tomcat exist. It is up to you to determine whether this scenario applies to your system and this document assumes that only this version of Tomcat is used.

To set the environmental variable for this root Tomcat folder, click on Start, Settings, Control Panel, and then System. In the Advanced tab click on the Environment Variables button. At the bottom of the Environment Variables dialogue box (for System variables) create a new Variable named $CATALINA_HOME and give it a path to the root Tomcat folder. Using the scenario above my Variable value is C:\Program Files\Apache Software Foundation\Tomcat 5.5.

Check your work: open a browser and enter the URI http://localhost:8080/. If your installation was successful you should see the Tomcat logo, program version number, and some additional information. If you see a different set of information (other than an error) then you have another server running instead of Tomcat. Disable or uninstall the other server and try again.

Wherever this document references $CATALINA_HOME in this scenario it means "C:\Program Files\Apache Software Foundation\Tomcat 5.5." Therefore $CATALINA_HOME\blah means C:\Program Files\Apache Software Foundation\Tomcat 5.5\blah.

Install MySQL

Download the Windows setup file named mysql-5.0.26-win32, unzip the item, and run the setup program. Select:

- Run the setup program, click Next
- Setup Type: Complete, click Next
- Accept default destination folder and click Install.
- Skip Sign-up (optional)

Configuring:

- Detailed Configuration, click Next
- Developer Machine, click Next
- Multifunctional Database, click Next
- MySQL Server Instance Configuration, InnoDB Tablespace Settings, accept
- default, click Next
- Decision Support (DSS)/OLAP (default) , click Next
- Enable TCP/IP Networking and Enable Strict Mode (yes – default on both) ,
- click Next
- Standard Character Set (default) , click Next
- Install As Windows Service (default) and check the box "Include Bin
- Directory in Windows PATH, click Next
- Modify Security Settings (default) and enter the root password as: 123456. The effect of checking "Enabling root access from remote machines" has not yet been used and in this tutorial the box was left unchecked. Do not
- create an Anonymous Account.
- Execute

If configuration fails and you use a firewall, try disabling the firewall and re-executing.

Install MySQL JDBC Driver

Unzip the MySQL file mysql-connector-java-5.0.4 into its own folder. Open the unzipped mysql-connector-java-5.0.4 folder and copy the jar file (named mysql-connector-java-5.0.4-bin) to $CATALINA_HOME\common\lib.

Install JUDDI

Ignore all instructions that are posted on the website for JUDDI, they are not for this version. There are a lot of independent steps. If you miss a step, you will have to review the entire jUDDI installation process again and locate the error. **Important** If you are not following the naming conventions used in this tutorial you must amend the content below to match your MySQL details and it is up to you to discover what needs to be changed. Good luck! I recommend following these instructions for now and altering them later, after you are more comfortable with the environment.

1) Get jUDDI Program Files
Download the program file labeled juddi-2.0rc5 and drop it into C:\JUDDI. Unzip it using the default name. Once unzipped your jUDDI files will be located at C:\JUDDI\juddi-2.0rc5. The relevance of this is simply so that your path matches the paths shown in this document. Later we will use documentation and scripts that come with the downloaded file. The URI for JUDDI is: http://ws.apache.org/juddi/releases.html.

Copy the C:\JUDDI\juddi-2.0rc5\JUDDI .war (not jar) file into the $CATALINA_HOME\webapps directory.

2) Copy/Paste juddi folder into Tomcat
Navigate to the JUDDI webapp folder. Within the JUDDI webapp folder copy the subfolder "juddi" and paste it into the $CATALINA_HOME\webapps folder. There is a JUDDI validation program which will help diagnose errors, but we're not ready to use it just yet.

3) Create log4j.properties
To enable log4j, you need to create the file $CATALINA_HOME\webapps\juddi\WEB-INF\classes\log4j.properties. In that file enter only this information:

```
#
# set the log file to $CATALINA_HOME\webapp\juddi\WEB-INF\{HOME}/juddi.log
# and not the ${PWD}/juddi.log
#
log4j.appender.LOGFILE.File=/opt/tomcat/logs/juddi.log
```

Save and close the file.

4) Edit juddi.properties
Amend the file juddi.properties at which is located at $CATALINA_HOME\webapps\juddi\WEB-INF\juddi.properties match the email address you used within insert_publishers.sql. This should match the domain name used for the email address within JUDDI's insert_publisher.sql script.

# The UDDI Operator Contact Email Address
juddi.operatorEmailAddress = yourEmail@aDomain.com

5) Edit server.xml
Next, edit the server.xml file which is located at $CATALINA_HOME\conf\server.xml and append the following to the file, immediately above the </Host> tag.

```
<Context path="/juddi" docBase="juddi"
debug="5" reloadable="true" crossContext="true">
<Logger className="org.apache.catalina.logger.FileLogger"
prefix="localhost_juddiDB_log" suffix=".txt"
timestamp="true"/>
```

```
<Resource name="jdbc/juddiDB" auth="Container" type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000"
username="juddi" password="juddi" driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/juddi?autoReconnect=true"/>

</Context>
```

6) Edit juddi-user.xml
Finally, edit the juddi-users.xml file, located at $CATALINA_HOME\webapps\juddi\WEB-INF\juddi-users.xml and change the password for user juddi to 123456.

```
<user userid="juddi" password=123456">
```

8) Create a "juddi" database in MySQL : Use your preferred mysql command line client to create a database named "juddi" (mysql> create database juddi). Make sure that the user that you intend to use with this example has permissions to create tables, select, update, and delete on the new database you have created. Unlike in prior versions of juddi, there is no need to to run a script to create the tables, tables will be created the first time the database is used in the following step.

10) Test Your Work:
Restart Tomcat. Right click the Tomcat icon, stop the service, then start the service. Open your Internet Explorer browser and enter the URI: http://localhost:8080/juddi.
Click on Validate and scroll for errors noted in red. If you have set up your juddiDB datasource correctly as seen in Step 5, the validation jsp will propagate the database with the necessary tables and default data.

Juddi is happy now. "Juddi is happy" is how writers express that jUDDI is correctly configured. If you see any errors in your JUDDI validation, rework these instructions.

Test your setup by creating a simple web service in eclipse 3.2

Install Eclipse 3.2.1

Obtain Eclipse. If using the Eclipse website to download the program choose these options. http://www.eclipse.org Next, pick a mirror and download the program. You must use version 3.2.1.

Unzip the program into C:\eclipse

Other than configuring a few settings within Eclipse itself this completes the installation process.

Establishing an SFTP Connection within Eclipse 3.2 to Repos using Plug-in

Eclipse Memory

Depending on the number of Eclipse plug-ins you install you may need to manually edit the Eclipse.ini file located in the root folder of your Eclipse installation. You may see this step repeated in other documentation but only one edit of this file is necessary. The values used below were sufficient for handling a larger number of plug-ins than those required for TCSS-460. The test environment for creating this documentation has 1.5 gigs of RAM. While creating a simple Web Service, several errors were resolved by increasing the minimal and maximum memory values shown below. You are free to experiment with these values and set them to anything that works for your environment. It is possible to set these values on the command line, when starting Eclipse, but it is up to you to discover how.

Open Eclipse.ini for editing. Within this file are the default minimum (Xms) and maximum (Xmx) memory settings for Eclipse. Increase the numeric values for these settings. Adding additional content, in addition to these three lines, may cause Eclipse to ignore the entire file. Settings that worked well for theauthor of this document were:

```
-vmargs -Xms512m -Xmx1024m
```

Save the Eclipse.ini file, close your editor, and make certain that the extension was not changed.

Eclipse Workspaces

Start Eclipse 3.2 and select or create a workspace. Eclipse will automatically create a workspace folder if you choose a workspace name that does not exist.

Eclipse Plug-ins and the Web Tools Platform

Web Services in Eclipse requires a plug-in called WTP (Web Tools Platform). Additional plug-ins will also be installed with this step. You are free to determine which are not absolutely necessary for this

project or to simply grab all that this tutorial suggests.

- Open Eclipse
- Select Help – Software Updates – Find and Install
- Search for updates of the currently installed features, click Finish
- Select a mirror, click OK
- Accept all terms and allow all updates to take place
- If an error occurs during these updates, repeat these steps.
- Allow Eclipse to restart the workspace if prompted

Installing new plug-ins:

- Select Help – Software Updates – Find and Install
- Search for new features to install, click Next
- Select Callisto Discovery Site and click Finish
- Select a mirror and click OK
- Expand the list for Callisto Discovery Site
- As you select plug-ins, errors will appear at the top of the dialogue by which indicate that additional downloads are required. When this error appears click the right-side button: Select Required, so that all other dependent files are downloaded as well.
- As you add plug-ins the help list within Eclipse should also expand as well.
- If you have ample memory feel free to select all plug-ins except for C/C++ development. Otherwise start with Java Development, Web and J2EE
- Development (WTP) and expand your functionality later. It is up to you to learn how to use these features. Remember to use the "Select Required" button to resolve plug-in file dependency errors.
- Click Next
- Accept all terms, click Next
- Click Finish.
- If an error occurs during these updates, repeat these steps.
- Allow Eclipse to restart the workspace if prompted

You may also repeat the first part of this process to look for additional updates of the new installed features (optional).

Eclipse Workspaces

Start Eclipse 3.2.1 and select or enter a new workspace name. Eclipse will automatically create a workspace folder if you choose a workspace name that does not exist.

1) Create a Tomcat server in the Eclipse workbench by following the instructions at the following link:
http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.jst.ws.axis.ui.doc.user/tasks/ttomcatserv.html

The following steps are done using the Eclipse IDE along with the WTP plugin:

2) Create a "Dynamic Web Project" by doing the following inside of Eclipse:
- File->New->Dynamic Web Project

- Give the project an appropriate name and select the "Finish" button. We'll call it "myApp" as a reference in this document.

3) Create your Java class with the public methods that you will be using as your Web Service Interfaces - this is called bottom up web design. Do this as follows:

- RIGHT click on the "myApp" project and select New->Class

- Give an appropriate package name (e.g. my.org.name) and class name (e.g. MyWebClass) and select the "Finish" button.

- Finish coding up the public methods for the "MyWebClass" class.
For this example use the following simple Java file:

- For this exercise 'build automatically' is selected. To enable this feature click Project then select Build Automatically.
- Make sure your project has all the required jar files in its "Java Build Path", i.e. do the following: RIGHT click on your project and select "Properties", then under "Java Build Path" add all the required jar files needed the compile your class.

- Make sure you build your project so that your class is available in the output folder (e.g. build/classes/...), i.e. do the following to create the class files: Project->Build Project

4A) RIGHT click on the "MyWebClass.java" [myApp/srs/my.or.name/MyWebClass.java] file and do the following to create the WSDL and the test client service:

- Web Services->Create Web Service

- In the "Web Services" window, check the "Generate a proxy" and "Test the Web service" boxes move the upper slider to the top position (Test Service) then select Next->Next.
-On the Test Web Service Window click Launch. If you have a local copy of Tomcat running, shut it down, wait 1 – 2 minutes, then click Launch. The Web Services Explorer will open within a new browser window in order for you to test your service (view WSDL Binding Details). Within the left side Navigator window, click on any of the methods. After clicking on a method the main window changes to enable you to Invoke a WSDL operation. In the space provided enter an appropriate parameter and click "go." In the Status window below the results of your transaction will appear – check for correctness. To view the underlying SOAP messages click "Source" within the Status window.
- Click Finish

4B) RIGHT click on the "MyWebClass.java" [myApp/srs/my.or.name/MyWebClass.java] file and do the following to create the WSDL and the test service:
- Web Services->Create Web Service

- In the "Web Services" window, move the upper slider to the Start Service position and the bottom slider to the Test Service position and then select Next->Next.
-On the Test Web Service Window click Launch. If you have a local copy of Tomcat running, shut it down, wait 1 – 2 minutes, then click Launch. The Web Services Explorer will open within a new browser window in order for you to test your service (view WSDL Binding Details). Within the left side Navigator window, click on any of the methods. After clicking on a method the main window changes to enable you to Invoke a WSDL operation. In thespace provided enter an appropriate parameter and click "go." In the Statuswindow below the results of your transaction will appear – check for correctness. To view the underlying SOAP messages click "Source" within the Status window.
- Click Finish

5) Once the "Web Services Explorer" window come up, you can test your interfaces. Some users have found that testing the interfaces using this mechanism may result in peculiar behaviors. For these users, what works better is to migrate your project to Tomcat by doing the following on the target machine:

5.1 – If running, stop Tomcat on the target machine/platform.

5.2 - Create a directory with the SAME name as your project, in this case it's "myApp", under the following directory -> TOMCAT_HOME/webapps which should result in something like TOMCAT_HOME/webapps/myApp

5.3 - Copy all contents under the "WebContent" folder for your project from your development machine (e.g. C:\EclipseWorkspace\myApp\WebContent*.*) to the target machine under TOMCAT_HOME/webapps/myApp.

5.4 - If your Eclipse output folder is defaulted to the "build" folder, then copy the "classes" folder for your project from your development machine (e.g. C:\EclipseWorkspace\myApp\build\classes) to the target machine under TOMCAT_HOME/webapps/myApp/WEB-INF.

5.5 - You should now have a similar directory structure as the following on your target machine:
--> TOMCAT_HOME/webapps/myApp/META-INF
--> TOMCAT_HOME/webapps/myApp/WEB-INF
--> TOMCAT_HOME/webapps/myApp/WEB-INF/classes
--> TOMCAT_HOME/webapps/myApp/wsdl

5.6 - Eclipse can create the client application project which you can use to test your Web Services, it will have the same name as your project's name with the word "Client" appended to it, for this discussion it would be "myAppClient". The code for the client can be used to create your own client classes to access your Web Service Interfaces, so you can make good use of it.
We are going to take steps similar to those in step 4B with a couple differences.
-RIGHT click on the "MyWebClass.java" [myApp/srs/my.or.name/MyWebClass.java]
-Select Web Services->Create Web Service
- In the "Web Services" window, move the upper slider to the Start Service position and the bottom slider to the Test Service position and then select Finish. Eclipse will create the myAppClient files (including some JSP files) and directory structure automatically. Eclipse will also open a Web Services Test Client window in which you may test your Web Service using Eclipse. Move on to step 8.7 to learn how to run this client within your local version of Tomcat.

5.7 - Repeat steps 5.2-5.5 and migrate the client application [including the JSP files located in the sampleMyWebClassProxy folder] (e.g. myAppClient) to TOMCAT/webapps/myAppClient too.

6) Start Tomcat on the target machine and Tomcat should recognize your new web applications located under the TOMCAT_HOME/webapps directory. You can access and test the "myApp" Web Services by running the "myAppClient" JSPthat was created by Eclipse. The typical URL for the client JSP is as follows (notice that it uses your application name and class name):
http://localhost:8080/myAppClient/sampleMyWebClassProxy/TestClient.jsp

7) If you make changes to your "MyWebClass" using Eclipse copy the "MyWebClass.class" file from your development machine (e.g. C:\EclipseWorkspace\myApp\build\classes\myPackageName\...\MyWebClass.class) to the target machine (e.g. TOMCAT_HOME/webapps/myApp/WEB INF/classes/myPackageName/.../MyWebClass.class). Restart Tomcat on the target machine and retest until everything is in order and your Web Services Interfaces are working as required.

8) At this point, using your Java class, you have successfully created your Web Services Interfaces and their WSDL document - the WSDL document lives under the TOMCAT_HOME/webapps/myApp/wsdl directory and can be accessed using the browser via a URL similar to the following: http://localhost:8080/myApp/services/MyWebClass?wsdl

Using the following URL will show you some other WSDL interfaces created by Eclipse: http://localhost:8080/myApp/services

Publishing the WSDL document into the UDDI registry...

Next we want to publish the "myWebClass" WSDL into a UDDI registry/server. In this example we will be using the jUDDI server that was installed on our target machine. If the jUDDI server was installed correctly, its main page should be at a URL similar to the following:
http://localhost:8080/juddi

By looking in the "controller.jsp" file that resides under the TOMCAT_HOME/webapps/juddi/console directory, we can see that the "inquiry" URL is something like "http://localhost:8080/juddi/inquiry"; - we will need this URL later in the wizard.
*Note: Your local version of Tomcat must be running in order for these next steps to execute successfully.
1) Bring up Eclipse and do: run->Launch the Web Services Explorer

2) In the Web Services Explorer window, click on the "UDDI Main" link.

3) For the "Inquiry URL" field enter the jUDDI inquiry URL that you got from the "controller.jsp" file, e.g. http://localhost:8080/juddi/inquiry.

4) Give the jUDDI server an appropriate name for the "Registry Name" field, e.g. "My jUDDI", then click on the "Go" button.

5) In the "Registry Details" screen, under "Other Actions", you can click on the "Add To Favorites" link to add your jUDDI registry server to the drop

down list (for future usage).

6) In the "Registry Details" screen, under "Other Actions", click on the "Publish" link to bring up the "Publish" screen. Use the following options to publish the "MyWebApp" services:

- Publish -> Service Interface
- Publication format -> simple
- Publish URL -> http://localhost:8080/juddi/publish
- User ID -> myUserId (this is the user ID that you have allowed to use the jUDDI database)
- Password ->
- WSDL URL -> platform:/resource/myApp/WebContent/wsdl/MyWebClass.wsdl
- Name -> My Application
- Description -> My Web Class Services

For the "WSDL URL" field, you must use the browse button to select the myApp project and Eclipse should be able to find the WSDL document from that project and import it into that field.

Notice that the "Publish URL" is the URL for the jUDDI server's "publish" address (i.e. where we're publishing the WSDL document) and NOT where the MyWebClass service interface lives (i.e. not http://localhost:8080/myApp/services/MyWebClass).

Searching for the Service...

1) Bring up Eclipse and do: run->Launch the Web Services Explorer

2) In the Web Services Explorer window, click on the "UDDI Main" link.

3) For the "Inquiry URL" field enter the jUDDI inquiry URL that you got from the "controller.jsp" file, e.g. http://localhost:8080/juddi/inquiry or use the drop down list if you had saved it already. Click the "Go" button.

4) In the "Registry Details" screen, under "Other Actions", click on the "Find" link to bring up the "Find" screen.

5) Use the following options in the "Find" screen to find your service:

- Search for -> Service Interfaces
- Type of search -> simple
- Name -> My Application (or whatever "Name" you used in step 6 above)

Click on the "Go" button and the information for the "MyWebClass" services should be displayed. Notice that you can "Edit" certain fields such as the "WSDL URL" field. You might want to use the actual URL for the Web Services such as http://my.host.machine.name:8080/myApp/services/MyWebClass?wsdl (in this tutorial: http://localhost:8080/myApp/services/MyWebClass?wsdl)instead of the default one listed in the MySql database, e.g. "platform:/resource/myApp/WebContent/wsdl/MyWebClass.wsdl