2.4. Developing with Eclipse

This document describes developing Metro WebServices on Eclipse. The instructi

2.4.1. Setup

This is one time setup.

1. After starting Eclipse, select the J2EE perspective: **Windows>Open Perspective>Others>J2EE**
2. In the lower window you should see a tab with label `Servers`. Select the tab and right click in the window and select **new>Server**.
3. To download the GlassFish server, select ***Download additional server adapters***. Accept the license and wait for Eclipse to restart.
4. After Eclipse has restarted, you can create a new GlassFish V2 Java EE5 server.
5. In the creation dialog select ***Installed Runtimes*** and select the directory where your GlassFish installation resides.

2.4.2. Create a Metro Web Services Endpoint

1. To create the HelloWorld service, create a new dynamic Web project. Give it a name (e.g. helloworld) and select as target runtime GlassFish
2. **HelloWorld.java**

```
package sample;

import javax.jws.WebService;

@WebService
public class HelloWorld {
    public String hello(String param){
    return param + ", World";
    }
}
```

3. Deploy the service by selecting the project and select ***Run as>Run on server***.
4. Check in the server Window that the helloworld project has a status of *Synchronized*. If this is not the case, right-click in the server window and select publish.
5. You can check that the GlassFish server is started and contains the Web service by going to the GlassFish admin console (localhost:4848)

See Arun's screen cast, it talks about the above steps.

2.4.3. Creating Web Service Client using Wsimport CLI

1. Create a new project for the HelloWorld client (an ordinary Java project suffices).
2. Select Add Glassfish v2 as Server Runtime in Build Path.
    - **Right click->BuildPath->Add Library->ServerRuntime->Glassfish v2**
3. Open a command window and go into the source directory of that project in Eclipse. For example, if the Eclipse workspace is in path

```
c:\home\vivekp\workspace
```

and the name of the project is HelloWorldClient, then you need to go to

```
c:\home\vivekp\workspace\helloworld\src
```

. In this directory execute

```
wsimport -keep http://localhost:8080/helloworld/HelloWorldService?wsdl
```

. On Linux or with Cygwin on Windows, you need to escape the ? by using \? instead.

4. Select refresh in the project view to see the generated files.
5. Now you can create the client class HelloWorldClient
6. You can execute the client, by selecting the HelloWorldClient in the package explorer of Eclipse and selecting Run>Java Application. In the console window of Eclipse, you should see "Hello World".

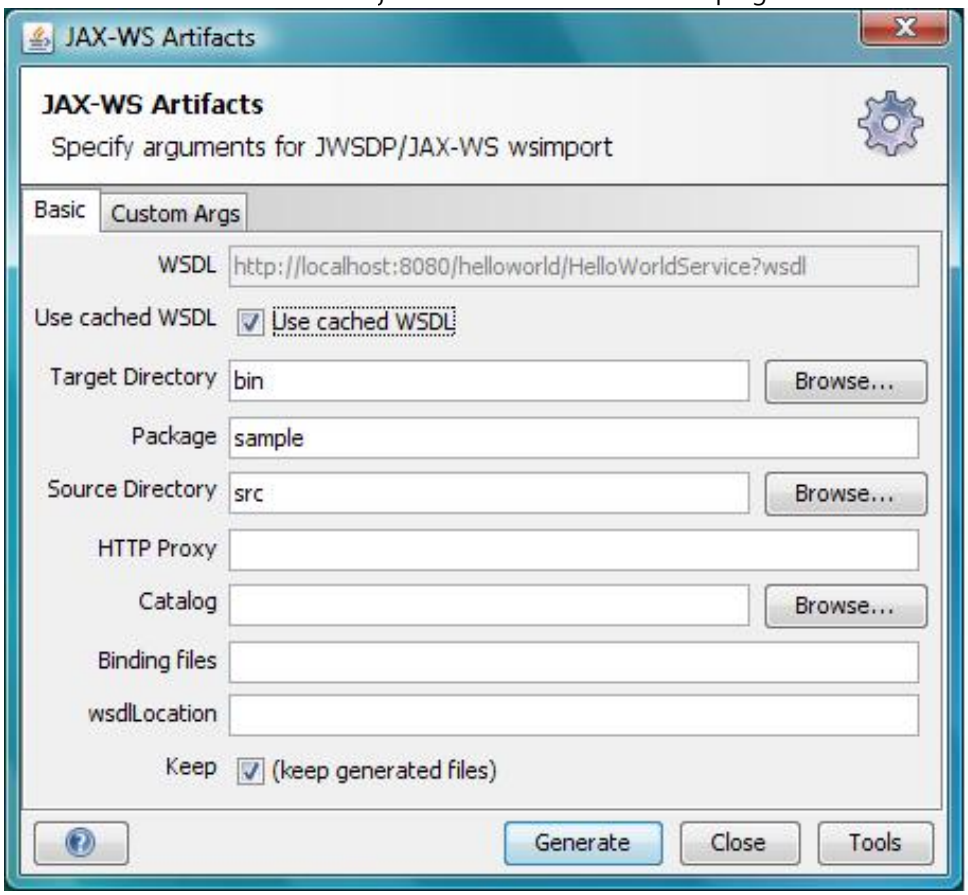
2.4.4. Creating Web Service Client using Wsimport Ant Task

You can pretty much avoid steps 3 - 5 above by using an Ant build.xml file.

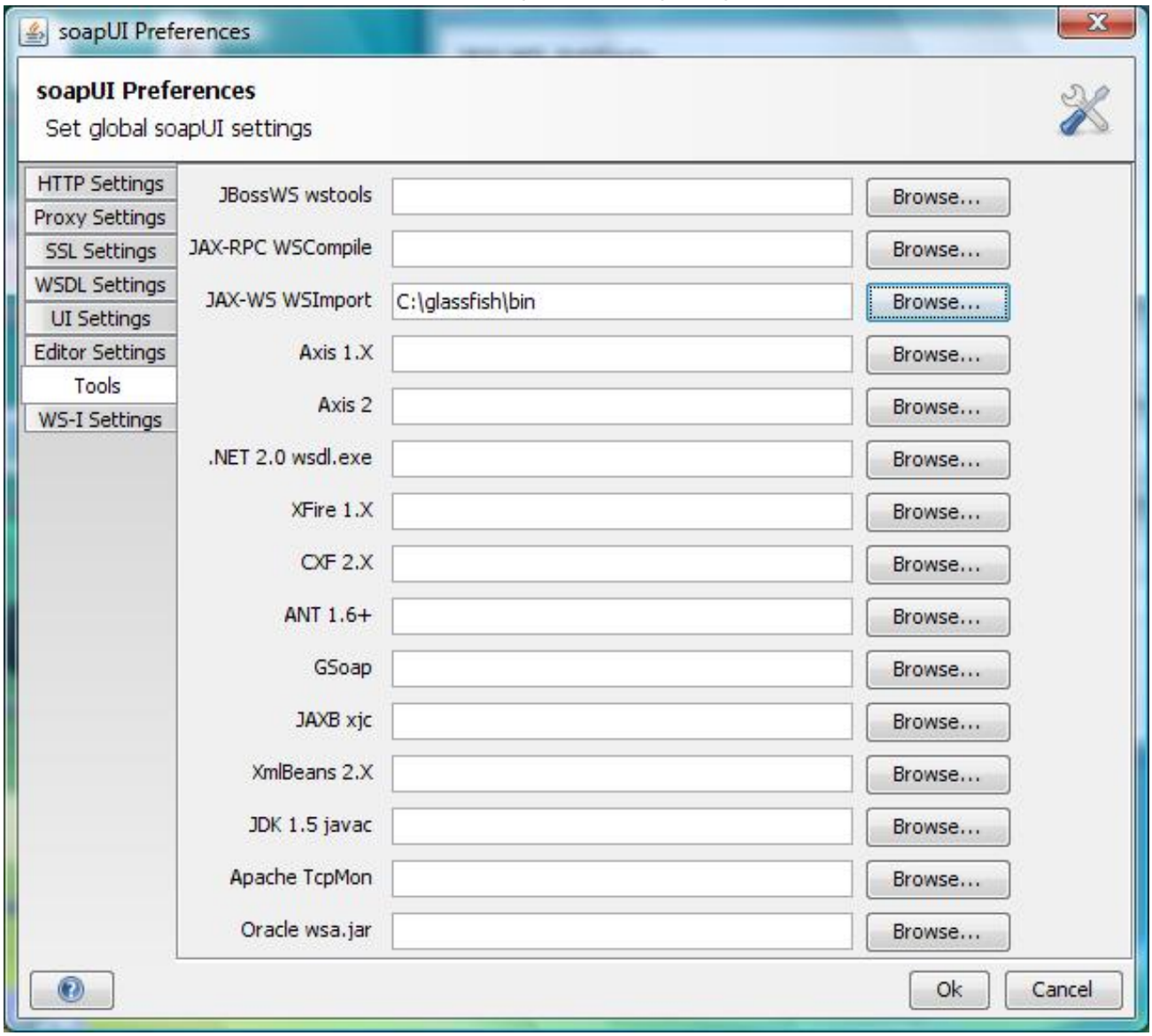
1. Select helloworldclient in Package Exp and create a new file build.xml
2. In this file (build.xml) copy the sample ant build script
3. Then select build.xml in the package explorer, then **Right Click->Run As->Ant Build...**
4. Invoke client target, it will run wsimport ant task and generate the client side stubs
5. Invoke run to invoke the endpoint and run the client or you can execute the client, by selecting the HelloWorldClient in the package explorer of Eclipse and selecting **Run>Java Application**. In the console window of Eclipse, you should see "Hello World".

2.4.5. Creating Web Service Client using SOAP UI Plugin

- Inside Eclipse, install SOAP UI Plugin
- Select "Help"/"Software Updates"/"Find and Install..."
- Press the "New Remote Site" button and add http://www.soapui.org/eclipse/update/site.xml as the plugin URL
- Select Finish and the follow the dialogs to install the soapUI feature
- Create a new project for the HelloWorld client (an ordinary Java project suffices).
- Select Add Glassfish v2 as Server Runtime in Build Path.
- **Right click->BuildPath->Add Library->ServerRuntime->Glassfish v2**
- Select the project and **Right Click->Soap UI->Add SOAPUI Nature**, SOAP UI WebService item will be added in Project Explorer
- Select HelloWorldPortBinding and **Right Click->GenerateCode->JAX-WS Artifacts**
- Enter the appropriate info in the JAX-WS Artifacts window



- Click Tools and enter the location of JAX-WS Wsimport, for example c:\glassfish\bin



- Click OK
- Then click Generate on JAX-WS Artifacts window, it will display a dialog box that the operation was successful. Switch back to Java Perspective, then refresh the src folder and you can see the wsimport generated classes
- Now implement your client code

HelloWorldClient.java

```
package sample;

public class HelloWorldClient {

   /**
    * @param args
    */
    public static void main(String[] args) {
        //Create Service
        HelloWorldService service = new HelloWorldService();

        //create proxy
        HelloWorld proxy = service.getHelloWorldPort();

        //invoke
        System.out.println(proxy.hello("hello"));
    }
}
```

- You can execute the client by selecting the HelloWorldClient in the package explorer of Eclipse and selecting Run>Java Application. In the console window of Eclipse, you should see "Hello World".

You can also use Wsimport and Wsgen Maven2 tools. For details see here. Netbeans offers an easy to use a comprehensive Metro tooling choice. On Eclipse you can use SOAP UI or ant build script or CLI or even Maven based tools, which does not look bad. There is RFE on Eclipse and looks like it is being looked at. For the Quality Of Service features (WS-* features) it is little difficult as manually creating/modifying WSIT configuration is hard, so we will need an equivalent of the WSIT Plugin in NetBeans for Eclipse. Please let us know if you are willing to write a WSIT plugin for Eclipse.

Built onMon Apr 26 14:48:00 PDT 2010