Service-Oriented Architectures and Web Services:
Course Tutorial Notes

Serguei A. Mokhov
Computer Science and Software Engineering
Faculty of Engineering and Computer Science
Concordia University
mokhov@cse.concordia.ca

July 28, 2009

Abstract
This document presents a number of quick-step instructions to get started on writing mini-service-oriented web services-based applications using NetBeans 6.5.x, Tomcat 6, GlassFish 2.1, and Java 1.6 primarily in Fedora 9 Linux with user quota restrictions. While the tutorial notes are oriented towards the students taking the SOEN691A course on service-oriented architectures (SOA) at Computer Science and Software Engineering (CSE) Department, Faculty of Engineering and Computer Science (ENCS), other may find some of it useful as well outside of CSE or Concordia. The notes are compiled mostly based on the students’ needs and feedback.

Contents
1 Introduction .................................................. 2
  1.1 Linux ......................................................... 2
  1.1.1 Accounts ................................................. 2
  1.1.2 Java 1.6 .................................................... 2
  1.1.3 NetBeans 6.5.1 ........................................... 3

2 Configuring NetBeans and GlassFish for BPEL .................................. 3

3 Step-by-Step Environment Setup .............................................. 4

4 Step-by-Step Simple Application and Web Service Creation and Testing ........ 14

5 BPEL Composite Applications ............................................. 24

6 Conclusion .......................................................... 24
  6.1 See Also ..................................................... 24
List of Figures

1 Terminal Window ........................................... 4
2 Setting up Java 1.6 as a Default in the Terminal .......... 5
3 Setting up ~/home to the Group Directory ............... 5
4 NetBeans 6.5.1 Start-up Screen .......................... 7
5 NetBeans: Services → Server → GlassFish V2 .......... 8
6 Right-click GlassFish V2 → Properties .................. 9
7 GlassFish Admin Console Login Screen ................. 11
8 Downloading Additional Libraries in a Terminal with wget ........................................ 12
9 List of Components and Shared Libraries Installed in GlassFish .................. 13
10 “Java EE” → “Enterprise Application” ................... 14
11 NetBeans Programming Projects Location .............. 15
12 A1’s Example Server and Client Settings ............... 16
13 A1 Project Tree ........................................... 17
14 New Login Web Service ................................... 19
15 A1 Project Tree after Login Web Service Creation ...... 20
16 Adding a Web Method login() .............................. 21
17 Implementing a Simple Web login() Method for Quick Unit Testing ................. 22
18 Unit-testing Page for the Login WS ...................... 23
19 login() Web Method Invocation Trace ........................ 25
20 Creating a New Web Services Client in the Client Application Package from a Project ........................................ 26
21 Selecting the Service to Create a Client For from the Project ................ 26
22 Creating a New Web Services Client Nearly Done. Notice the URL ........... 27

1 Introduction

1.1 Linux

We are using Fedora 9 Linux during the labs. For your own work you can use any platform of your choice, e.g. Windows or MacOS X on your laptops. You will have to do the install and configuration of NetBeans, Java, Tomcat and so on there.

On ENCS Windows the software was not made readily available (in particular more recent NetBeans with the ALL option, and Tomcat 6 [Apa09]).

1.1.1 Accounts

Under UNIX, disk space (for a sample account acs691a1) would be accessible under e.g. /groups/a/ac_as691_a1 Under Windows, that path would be \filer-groups\groups\a\ac_soen691a_1 (“S:” drive). There is a 1GB storage space there and your in-school work related to the assignments and courses can be put there, as the generated data files can be large at times.

1.1.2 Java 1.6

Java 1.6 is not a default Java in ENCS. You need to make it default. In order to use this version all you need to prepend:
in your path. To do so there are simple instructions:

People using tcsh:

[serguei@lucid ~] % setenv PATH /encs/pkg/jdk-6/root/bin:$PATH
[serguei@lucid ~] % rehash
[serguei@lucid ~] % java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)
[serguei@lucid ~] %

People using bash:

bash-2.05b$ export PATH=/encs/pkg/jdk-6/root/bin:$PATH
bash-2.05b$ rehash
bash: rehash: command not found
bash-2.05b$ java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)
bash-2.05b$

You can avoid typing the above commands to set the PATH each time you open a terminal under Linux by recording it in ~/.cshrc. If you do not have this file in your home directory you can create one with the following content (e.g. using vim [MC07]):

set path=( /encs/pkg/jdk-6/root/bin $path )

or copy an example from [Mok09] and update the path to include the above directory first. Thus, next time when you login and open terminal, Java 1.6 will always be your default. The same applies if you click on the NetBeans shortcut in the menu.

1.1.3 NetBeans 6.5.1

NetBeans [Sun09b] is accessible as a simple command netbeans or from the “Applications” → “Programming” → “NetBeans” menu with a corresponding icon.

2 Configuring NetBeans and GlassFish for BPEL

The ALL option typically installs GlassFish 2.1 [Sun09a] as well as Tomcat 6 bundled by default with NetBeans, as well as some of the components. This includes some of the BPEL [Wik09] components as well. To complete all the needed extensions for BPEL for GlassFish you’d need to download WSDL extensions and Saxon shared libraries and deploy them within your running GlassFish instance. Download libraries for BPEL SE [Ope09], specifically: wsdlextlib.jar and saxonlib.jar. That’s all you need for your setup in the lab. For your home computer you may need to download and install the actual BPEL service engine component from the same web page [Ope09], called bpelserviceengine.jar.
3  Step-by-Step Environment Setup

1. Login to Linux. If you never did before likely your default Window manager is GNOME.

2. Open up the terminal: “Applications” → “System Tools” → “Terminal”. The window similar to Figure 1 should pop-up.

![Figure 1: Terminal Window](image)

3. Configure your Java 1.6 to be the default as outlined in Section 1.1.2, and an example is shown in Figure 2.

4. In the same terminal window, change your HOME environment variable to that of your 1GB group directory. This will allow most portions of NetBeans to write the temporary and configuration files there by default instead of your main Unix home directory. I use a temporary directory of mine `/tmp/groups/s/sm_s691a_1` as an example – and you should be using the directory assigned to you with your group 1GB quota. An example to do so is very similar as to set up PATH, except it is a single entry. It is exemplified in Figure 3. Unlike PATH, it is *not* recommended to put these commands to change your HOME into `.cshrc`.

5. Create the following directories in your new HOME (your 1GB group directory):
Figure 2: Setting up Java 1.6 as a Default in the Terminal

```
[serguei@alfredo ~] % setenv PATH /encl/pkg/jdk-6/root/bin:$PATH
[serguei@alfredo ~] % rehash
[serguei@alfredo ~] % java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Server VM (build 14.0-b16, mixed mode)
[serguei@alfredo ~] %
```

Figure 3: Setting up HOME to the Group Directory

```
[serguei@alfredo ~] % setenv HOME /tmp/groups/s/s691a_1
[serguei@alfredo serguei] % cd
[serguei@alfredo ~] % pwd
/tmp/groups/s/s691a_1
[serguei@alfredo ~] %
```
mkdir .netbeans .netbeans-derby .netbeans-registration
ls -al

These directories will hold all the configuration and deployment files pertaining to NetBeans, the Derby security controller, and the personal domain for GlassFish operation. The overall content may easily reach 80MB in total disk usage for all these directories.

6. Disk usage, quota, and big files:

    quota
du -h
bigfiles

7. In your real home directory, remove any previous NetBeans et co. setup files you may have generated from the previous runs:

    .asadminpass
    .asadmintruststore
    .netbeans*
    .personalDomain*
(assuming no important data for you are saved there):

    \rm -rf .netbeans* .personalDomain* .asadmin*

8. In your real home directory create symbolic links (“shortcuts”) to the same NetBeans directories, so in case it all still goes to the group directory without impending your main quota:

    [serguei@alfredo ~] % pwd
    /nfs/home/s/serguei
    [serguei@alfredo ~] % ln -s /tmp/groups/s/sm_s691a_1/.netbeans* .netbeans*
    [serguei@alfredo ~] % ls -ld .netbeans*
    lrwxrwxrwx 1 serguei serguei 34 2009-07-11 08:05 .netbeans -> /tmp/groups/s/sm_s691a_1/.netbeans
    lrwxrwxrwx 1 serguei serguei 40 2009-07-11 08:05 .netbeans-derby -> /tmp/groups/s/sm_s691a_1/.netbeans-derby
    lrwxrwxrwx 1 serguei serguei 47 2009-07-11 08:05 .netbeans-registration -> /tmp/groups/s/sm_s691a_1/.netbeans-registration

9. Again, in the same terminal window launch NetBeans, by executing command netbeans &, and after some time it should fully start up without of any errors. You will be prompted to allow Sun to collect your usage information and register; it is recommended to answer “No” to both. And then you will see a left-hand-side (LHS) menu, the main editor page with the default browsed info, and the top menu of the NetBeans, as shown in Figure 4. This is NetBeans 6.5.1, the latest released by the project is 6.7, and it will look slightly different in some places, but overall it is more-or-less the same.

10. Navigate to the Services tab and expand the Server tree in the LHS menu. You should be able to see a GlassFish V2 entry there, as shown in Figure 5.
Figure 4: NetBeans 6.5.1 Start-up Screen
Figure 5: NetBeans: Services → Server → GlassFish V2
Figure 6: Right-click GlassFish V2 → Properties
11. Right-click on “GlassFish V2” and then “Properties”, as in Figure 6. Observe the “Domains folder” and “Domain Name”. If the folder points within your normal home directory, you have to change it as follows:

(a) Close the properties window.
(b) Right-click on “GlassFish V2” and then “Remove”. Confirm with “Yes” the removal.
(c) Right-click on “Servers” and then “Add Server...”.
(d) Select “GlassFish V2” and then “Next”, and “Next”.
(e) Then for the “Domain Folder Location” Browse or paste your group directory, e.g. 

`/tmp/groups/s/sm_s691a_1/.domain` in my case, notice where `.domain` is an arbitrary name of a directory under your group directory that is not existing yet, give it any name you like, and then press “Next”.

(f) Pick a user name and a password for the admin console (web-based) of GlassFish. The NetBeans default (of the GlassFish we removed) is ‘admin’ and ‘adminadmin’. It is strongly suggested however you do NOT follow the default, and pick something else. Do NOT make it equal to your ENCS account either.

(g) “Next” and “Finish”. Keep the ports at their defaults. Notice it may take time to restart the new GlassFish instance and recreate your personal domain you indicated in the group folder.

12. Right-click on GlassFish again and select “Start”. It may also take some time to actually start GlassFish; watch the bottom-right corner as well as the output window for the startup messages and status. There should be no errors. Apache Derby network service should have started.

13. Once started, right-click on GlassFish again, and select “View Admin Console”. You should see the GlassFish login window pop-up in the Firefox web browser, looking as shown in Figure 7.

14. To log in, use the username and password you created earlier in Step 11f.

15. In your group home terminal, download additional libraries from [Ope09]. You will only need 2 (`wsdlextlib.jar` and `saxonlib.jar`) out of typical 3, because the version installed in ENCS already includes the 3rd (`bpelserviceengine.jar`). You will likely need the 3rd file however, for your laptop or home desktop in Windows. You can either download them directly from the browser, or using the `wget` command, as shown in Figure 8.

16. In your GlassFish console web page, under “Common Tasks” → “JBI” → “Shared Libraries” you need to install the two libraries we downloaded (3 for your Windows laptop or home desktop) by clicking “Install” and following the steps by browsing to the directory where you downloaded the files and installing them. Then, once installed `sun-saxon-library` and `sun-wsdl-ext-library` should be listed under the “Shared Libraries”.

17. Make sure under “Components” you have `sun-bpel-engine`. Linux boxes in the labs should have it installed with the NetBeans, at home it’s the 3rd file – `bpelserviceengine.jar` that may need to be installed using the similar procedure as in the previous step. Roughly, how your “Components” and “Shared Libraries” should look like is in Figure 9.
Figure 7: GlassFish Admin Console Login Screen
Figure 8: Downloading Additional Libraries in a Terminal with `wget`
Figure 9: List of Components and Shared Libraries Installed in GlassFish
On this the environment setup should be complete. You will technically not need to repeat except if you remove all the files from your group directory.

4 Step-by-Step Simple Application and Web Service Creation and Testing

1. Go to the “Projects” tab in NetBeans.
2. Then “File” → “New Project”.
3. Choose “Java EE” → “Enterprise Application”, as shown in Figure 10 and then “Next”.

4. Give the project properties, like Project Name to be “A1”, project location somewhere in your group directory, e.g. as for me shown in Figure 11 and then “Next”.

5. In the next tab, you can optionally enable “Application Client Module” for an example, and keep the rest at their defaults, e.g. as shown in Figure 12. Notice, I altered the client package Main class to be in soen691a.a1.Main. It is not strictly required in here as you can test your web services using web service unit testing tools built-into the IDE.
Figure 11: NetBeans Programming Projects Location
Figure 12: A1’s Example Server and Client Settings
6. Click “Finish” to create your first project with the above settings. You should see something that looks like as shown in Figure [13] after some of the tree elements expanded.

7. Under A1-war, create a package, called soen691a by right-clicking under “A1” → “Source Packages” → “New” → “Java Package” → “Package Name”: soen691a. Then “Finish”.

8. Create a “Web Service” under that package, by right-click on the newly created package → “New” → “Web Service” → “Web Service Name” → Login, as shown in Figure [14].

9. The LHS project tree if expanded would look like shown in Figure [14].

10. Right-click on Login WS, and select “Add Operation...” and create a web method login(), as shown in Figure [16].

11. After the web method login() appears as a stub inside the Login class with return false; by default. For quick unit testing of the new method, implement it with some test user name and password as shown in Figure [17] which will later be replaced to be read from the XML file.

12. Perform a simple unit test for the web method. Your GlassFish must be running and you have to “start” your project by deploying – just press the green angle “play” button. You should see a “Hello World” page appearing in your browser.

13. Then, under “A1-war” → “Web Services” → “Login” right-click on Login and select “Test Web Service”. It should pop-up another browser window (or tab) titled something like “LoginService Web Service Tester” with a pre-made form to test inputs to your web method(s), as shown in Figure [18].

14. Fill-in the correct test values that we defined earlier for login and press the “login” button. Observe the exchanged SOAP XML messages and the true value returned as a result, as shown in Figure [19].

Then try any wrong combination of the username and password and see that it returns false. This completes basic verification of your web service – that is can be successfully deployed and ran, and its method(s) unit-tested on the page.

15. Java-based client callee of a web service has to be defined e.g. as a WS client, as shown in earlier screenshots as “A1-app-client”, which has a Main.main() method. In that method you simply invoke the desired service by calling its web method after a number of instantiations. It may look like you are calling a local method of a local class, but, in fact, on the background there is a SOAP message exchange, marshaling/demarshaling of data types, etc. and actually connection to a web service, posting a request, receiving and parsing HTTP response, etc. all done by the middleware.

Steps:

(a) Right-click “A1-app-client” → “New” → “Web Service Client”. A dialog shown in Figure [20] should appear. Click “Browse”.

(b) Select your web service to generate a reference client for, as e.g. shown in Figure [21] and click “OK”.

17
Figure 13: A1 Project Tree
Figure 14: New Login Web Service
Figure 15: A1 Project Tree after Login Web Service Creation
Having selected the service to generate the WS client code for, you should see the URL, as shown in Figure 22 “Finish”, re-deploy (green “Play” button).

Then, in Main, import the generated code classes to invoke the service, as shown in Listing 1.

See also an example from DMARF [Mok06].

Relative path for loading XML can be found using System.getProperty(''user.dir'') to find out your current working directory of the application, which is actually relative to the config/ subdirectory in your personal domain folder, so it would be based on your deployment, but roughly:

System.getProperty("user.dir") + "./generated/....../users.xml"

where “......” is the path leading to where your users.xml and others actually are. You can configure Ant’s build.xml (actually build-impl.xml and other related files for deployment to copy your XML data files into config/ automatically.

Loading and querying XML with SAX is exemplified in TestNN with MARF [CMt09 The09], specifically at these CVS URLs:

http://marf.cvs.sf.net/viewvc/marf/apps/TestNN/
http://marf.cvs.sf.net/viewvc/marf/marf/src/marf/Classification/NeuralNetwork/
Figure 17: Implementing a Simple Web `login()` Method for Quick Unit Testing
package soen691a.a1;

import soen691a.Login;
import soen691a.LoginService;

/**
 * @author serguei
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        LoginService service = new LoginService();
        Login login = service.getLoginPort();

        // ...
        // Must be false
        boolean success = login.login("wrongusername", "wrongpassword");
        // Must be false
        success = login.login("wrongusername", "pa$3T");
        // Must be false
        success = login.login("userTest", "wrongpassword");
        // Must be true
        success = login.login("userTest", "pa$3T");
        // ...
    }
}

Listing 1: Invoking a Web Service from a Plain Java Class
Do not validate your XML unless you specified a DTD schema (not necessary here), just make sure your tags are matching, properly nested, and closed.

5 BPEL Composite Applications

GlassFish is needed for BPEL (while the previous could be done with Tomcat 6). E.g. tutorial from NetBeans:

http://www.netbeans.org/kb/61/soa/loanprocessing.html

Similarly, there are good application samples available in the betbeans to start the process of a BPEL composite application: “New” → “Samples” → “SOA”; specifically “Travel Reservation Service” and “BPEL BluePrint 1”.

6 Conclusion

Please direct any problems and errors with these notes or any other constructive feedback to mokhov@cse.concordia.ca.

6.1 See Also

- GlassFish website [Sun09a].
- Unix commands [Mok05].
login Method invocation

Method parameter(s)

| Type            | Value       |
|-----------------|-------------|
| java.lang.String | userTest    |
| java.lang.String | pa$$+3$$T  |

Method returned

boolean: "true"

SOAP Request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns2="http://soen691a/">
  <S:Body>
    <ns2:login xmlns:ns2="http://soen691a/"
      xmlns:username="http://soen691a/"
      xmlns:password="http://soen691a/"
     username="userTest"
      password="pa$$+3$$T"
    />
  </S:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns2="http://soen691a/"
  xmlns:username="http://soen691a/"
  xmlns:password="http://soen691a/"
>
  <S:Body>
    <ns2:loginResponse xmlns:ns2="http://soen691a/"
                        xmlns:username="http://soen691a/"
                        xmlns:password="http://soen691a/"
                        username="userTest"
                        password="pa$$+3$$T"
      username="userTest"
      password="pa$$+3$$T"
    />
  </S:Body>
</SOAP-ENV:Envelope>
```
Figure 20: Creating a New Web Services Client in the Client Application Package from a Project

Figure 21: Selecting the Service to Create a Client For from the Project
Figure 22: Creating a New Web Services Client Nearly Done. Notice the URL

- ENCS help: [http://www.encs.concordia.ca/helpdesk/](http://www.encs.concordia.ca/helpdesk/)

- An example of the XML parsing application, TestNN with MARF [CMt09][The09] using the built-in SAX parser.

References

[Apa09] Apache Foundation. Apache Jakarta Tomcat. [online], 1999–2009. [http://jakarta.apache.org/tomcat/index.html](http://jakarta.apache.org/tomcat/index.html)

[CMt09] Ian Clement, Serguei A. Mokhov, and the MARF Research & Development Group. TestNN – Testing Artificial Neural Network in MARF. Published electronically within the MARF project, [http://marf.sf.net](http://marf.sf.net) 2002–2009. Last viewed April 2008.

[MC07] Bram Moolenaar and Contributors. Vim the editor – Vi Improved. [online], 2007. [http://www.vim.org/](http://www.vim.org/)

[Mok05] Serguei A. Mokhov. UNIX commands, revision 1.4. [online], 2003 – 2005. [http://users.encs.concordia.ca/~mokhov/comp444/tutorials/unix-commands.pdf](http://users.encs.concordia.ca/~mokhov/comp444/tutorials/unix-commands.pdf)

[Mok06] Serguei A. Mokhov. On design and implementation of distributed modular audio recognition framework: Requirements and specification design document. [online], August 2006. Project report, [http://arxiv.org/abs/0905.2459](http://arxiv.org/abs/0905.2459), last viewed May 2009.
[Mok09] Serguei A. Mokhov. A .cshrc example, 2000–2009. \url{http://users.ens.concordia.ca/~mokhov/comp346/.cshrc}

[Ope09] OpenESB Contributors. BPEL service engine. [online], 2009. \url{https://open-esb.dev.java.net/BPELSE.html}

[Sun09a] Sun Microsystems. Sun GlassFish: Open web application platform. [online], 1994–2009. \url{http://www.sun.com/glassfish}

[Sun09b] Sun Microsystems. NetBeans 6.5.1. [online], July 2004–2009. \url{http://www.netbeans.org}

[The09] The MARF Research and Development Group. The Modular Audio Recognition Framework and its Applications. SourceForge.net, 2002–2009. \url{http://marf.sf.net}, last viewed December 2008.

[Wik09] Wikipedia. Business Process Execution Language (BPEL) — Wikipedia, the free encyclopedia. [Online; accessed 14-July-2009], 2009. \url{http://en.wikipedia.org/w/index.php?title=Business_Process_Execution_Language&oldid=302021294}
Index

API
A1-war, 17
false, 17
HOME, 4, 5
Login, 17, 19, 24
login(), 17, 21, 22, 25
Main, 14, 21
Main.main(), 17
PATH, 3, 4
path, 3
soen691a, 17
soen691a.a1.Main, 14
System.getProperty("user.dir"), 21
TestNN, 21, 27
true, 17

Files
.asadminpass, 6
.asadmintruststore, 6
.cshrc, 4
domain, 10
.netbeans*, 6
.personalDomain*, 6
./encs/pkg/jdk-6/root/bin, 3
./tmp/groups/s/sm_s691a_1, 4
./tmp/groups/s/sm_s691a_1/.domain, 10
./cshrc, 3
bpelserviceengine.jar, 3, 10
build-impl.xml, 21
build.xml, 21
config/, 21
saxonlib.jar, 3, 10
users.xml, 21
wsdlextlib.jar, 3, 10

Frameworks
Distributed MARF, 21
MARF, 21, 27

Java, 1, 2

Libraries
Distributed MARF, 21
MARF, 21, 27

Tools
bash, 3
netbeans, 3
netbeans &, 6
vim, 3
wget, 10, 12

Distributed, 21

MARF, 21, 27