ICAT Client Tutorial Practical Session

Richard Tyer and Glen Drinkwater Metataxa Limited

Introduction

The aim of this session is to provide some practical experience using the ICAT API with developers on hand to assist if necessary. The session will work through the steps required to perform some simple, but representative, use cases with the ICAT API.

Prerequisites

The session will use the Java programming language and will use the NetBeans Platform. It should be noted that due to the standards compliant nature of the ICAT API, it is possible to interact with ICAT API from any platform and language that has the corresponding web services libraries available, although this is outside the scope of this workshop.

Your machines should have already had the Java SE development kit and the NetBeans platform installed on them in which case you can skip the rest of this section.

- Installing the JDK: Please go to http://java.sun.com/javase/downloads/index.jsp and download the latest version of the Java development kit. The exact version is unimportant, but it should be version 6.x. After agreeing to the licensing agreement, the download should now begin and you should install the software as appropriate for your platform. Further instructions can be found at: http://java.sun.com/javase/6/webnotes/install/index.html
- Installing the NetBeans platform: Unfortunately there is an intermittent bug in the wsimport tool within the current release version of NetBeans, so you should download the last release of the previous branch 6.0.1. This can be found at http://download.netbeans.org/netbeans/6.0/final/ As you will need the web service development tools, please select the 'All' NetBeans IDE Download Bundle. The download should now begin and you should install the software as appropriate for your platform. Further instructions can be found at: http://www.netbeans.org/community/releases/60/install.html#installation

Creating a NetBeans Project

We need to create an project within NetBeans to contain the configuration and files for this workshop. To do this, start up the NetBeans IDE then:

- From the File menu select 'New Project'
- Choose category: Java and Project: Java Application, then select Next
- Choose a suitable name and location for your project. Leave the tick boxes 'Set as Main Project' and 'Create Main Class' ticked
- Select Finish

Compiling the WSDL

The WSDL defines the service interface endpoints and the corresponding remote proxy, along with the business delegates and the exceptions. This can be done using the wsimport tool (see https://jax-ws.dev.java.net/jax-ws-ea3/docs/wsimport.html) from either the command line or ant. The NetBeans platform includes this tool and we shall use NetBeans as an interface to it.

- Within the projects pane on the left hand side of the IDE, right click on your newly created project then select New then Web Service Client.
- A wizard entitled 'New Web Service Client' will appear. In the top section 'Specify the WSDL file of the Web Service', select the 'WSDL URL' radio button and enter the value:

  https://facilities01.esc.rl.ac.uk/ICATService/ICAT?wsdl
- Leave the other settings unaltered and select 'Finish'
- Click 'Yes' to assert that you trust the server certificate and wsimport should start.

If you go back to the Projects pane and expand 'Web Service Resources' then 'ICAT' then 'ICAT Service' and finally 'ICATPort' then you will see all the methods within the ICAT web services API.

Importing the server certificate

Before we can import the WSDL and call the ICAT API web services, we need to assert that we trust the server hosting the ICAT API web services. This is achieved by using a Java keystore that contains the public key corresponding to the server certificate. This can be downloaded from:

http://eminerals.dl.ac.uk/cacerts-facs01.jks

Once the file is downloaded, we need to configure NetBeans to use this instead of the default keystore shipped with the JDK. To do this:

- Within the projects pane on the left hand side of the IDE, right click on your newly created project then select 'Properties'
- In the 'Categories' pane on the left hand side, select 'Run'
- In the 'VM Options' text field add: -Djavax.net.ssl.trustStore=<*location of your JKS file*> , e.g. -Djavax.net.ssl.trustStore= /home/rty/cacerts-facs01.jks
- Click 'OK' to close the Project Properties window

AuthDetails Singleton

For the purpose of this workshop, we will do most of the work within the Main class for brevity. However in order to avoid having to display your federal ID and password in the event that you require assistance, we will however create another class to hold these. To do this:

- Within the Projects pane on the left hand side of the IDE, right click on the default package. This will be the entry under 'Source Packages' and will by default be the name of the project in lower case without spaces.

- Select 'New' then 'Java Class'. Enter 'AuthDetails' for ClassName then click 'Finish'
- Enter the following code edited to contain your federal ID and password

```
public class AuthDetails {
   private final static String username=   <your federal ID>   ;
   private final static String password=   <your password>   ;
   private static AuthDetails authDetails;
   private AuthDetails() {
   }

   public static AuthDetails getInstance() {
      if (authDetails==null) {
         authDetails = new AuthDetails();
      }

      return authDetails;
   }

   public String getUsername() {
      return username;
   }

   public String getPassword() {
      return password;
   }
}
```

- You will be able to then retrieve your federal ID and password within your other test classes using:

```
String username = AuthDetails.getInstance().getUsername();
String password = AuthDetails.getInstance().getPassword();
```

Getting an ICAT Instance

Return to your Main.java file which should already contain a main() method. Before we can call any of the ICAT API methods, we need to obtain a reference to the remote proxy for the service endpoints. This is achieved by instantiating an ICATService object which is a factory for the ICAT class that

proxies for the remote methods.

All the ICAT related classes live within the package uk.icat3.client, so add the appropriate import statement to the top of Main.java (between the package statement and the start of the Main class):

```
import uk.icat3.client.*;
```

Now within your main() method add the code to get a reference to the remote proxy:

```
ICATService service = new ICATService();
ICAT icat = service.getICATPort();
```

Obtaining a session ID

The first step in any workflow with the ICAT is usually obtaining a session ID string that is then used in subsequent web service calls to authenticate the user. For the sake of this tutorial, we will be rather lax with respect to exception handling and put our calls to the ICAT API within a try catch construct that catches java.lang.Exception.

The call to ICAT API to obtain a session ID is:

```
String login(String username, String password)
```

so add the following code to your main() method:

```
try {
   String sid = icat.login(username, password);
   System.out.println(   ICAT Returned session ID:     + sid);

} catch (Exception e) {
   System.err.println(e);
}
```

Now that the code is complete, you can compile it by pressing F9. Assuming that the compilation is fine, the code can be executed by pressing F6. The tail of the output should look similar to:

```
Compiling 1 source file to /home/rty/NetBeansProjects/ICATTutorial/build/classes
compile-single:
run-single:
Session ID = 14383bdc-685b-438e-c418-86bb4c7f1e56
BUILD SUCCESSFUL (total time: 16 seconds)
```

Searching for investigations by keywords

Now that you have a session ID, you can search the ICAT catalogue for investigations that contain

keywords of interest to you. Use this methods to obtain and display a list of investigations that contain your favourite keywords:

List<Investigation> searchByKeywords(String sessionId, List<String> keywords)

Recall that matching investigations must match all keywords that you specify and that, using this method, keyword searches are case insensitive.

Try and call some of the methods on the Investigation business delegate objects, e.g. getId(), getTitle() etc. If you obtain the list of keywords associated with an investigation found using the above method, why is it empty? What about the method:

List<Investigation> searchByKeywordsAll(String sessionId, KeywordDetails kd, int startIndex, int numberOfResults)

How can you perform case sensitive searches?

Listing datasets within a collection

Now that you have the investigation IDs for some interesting investigations, let's use the ICAT API to interrogate the catalogue to find its constituent datasets and data files. Some useful method are:

Investigation getInvestigation(String sessionId, Long investigationId)

Investigation getInvestigation(String sessionId, Long investigationId, InvestigationInclude includes)

Collection<Investigation> getInvestigations(String userId, Collection<Long> investigationIds, InvestigationInclude includes)

Note that getInvestigation(sid, investigationId) is equivalent to getInvestigation(sid, investigationId, InvestigationInclude.NONE), so you will need to use the latter two methods in order to have the dataset and data file attributes of the investigation populated. Please be judicious about how much data you pull back from the ICAT! Try InvestigationInclude.DATASETS_AND_DATAFILES for example.

You can obtain the list of datasets within an investigation using getDatasetCollection() and similarly get the list of data files within a dataset using getDatafileCollection(). Print out some of the key fields relating to the dataset and data files, e.g. their id's and names.

Downloading datasets and data files

Finally, now that you have the ID's of various datasets and data files, you can use ICAT to get a URL that will enable them to be downloaded using another tool or API (e.g. a web browser). Try out the following ICAT API methods:

String downloadDatafile(String sessionID, Long datafileID)

String downloadDataset(String sessionID, Long datafileID)

String downloadDatafiles(String sessionID, Collection<Long> datafileIDs)