ARCHER 1.0

ICAT

Developer's Guide

- Overview
- Software frameworks
- Development status

Overview

This guide gives an overview of the ICAT service and its use by developers.

The ICAT service provides an object oriented interface to the ICAT database. It allows calling programs to manipulate all the objects in the CCLRC Scientific Metadata Model: Studies, Investigations, Experiments, Datasets, Datafiles and Samples.

There are two implementations of the ICAT service interface:

- ICAT Hibernate implementation: A Java API, used by deploying other Java programs on the same machine
- ICAT web services: Generated by CXF/JAXB.

Development status

ICAT was developed at James Cook University, as part of the ARCHER project. It is released as open source under the GNU General Public License. At the time of writing, its future development and maintenance was unclear.

Java API

The definitive guide to accessing ICAT through the Java API is in the form of the Javadoc documentation. This documentation is available from the ARCHER website.

The included test cases provide typical examples of using ICAT. These are found at `common/src/test/java/au/edu/archer/services/icat`.

Web services

The precise definition of the web services provided, and their arguments, is the WSDL (Web Services Definition Language) file included in the source code package.

Software architecture and frameworks

ICAT uses a number of frameworks to facilitate the serialising of Java objects to and from the database, into XML, and then onto SOAP for the web services layer.

In crude terms, the frameworks used are as follows:

Database ← ***HyperJAXB*** → Java objects ← ***JAXB*** → XML ← ***CXF*** → SOAP

The layers and frameworks are built up as follows:

1. The Java objects and the Java API to access them is written by hand. The primary interface is called ICAT.
2. An XML schema document (.xsd) is written to describe the form of these objects when serialised.
3. JAXB uses this .xsd and the class definitions to automatically produce classes to serialise the objects to XML (`common/generated/src/main/java/au/edu/archer/schemas/icat`)
    a. HyperJAXB, a plugin to JAXB, adds Hibernate annotations to the generated classes.
4. A WSDL (Web Services Description Language) document is written to describe the Web Services to be provided

    a. CXF uses this WSDL, which imports the above XML schema document, to generate a set of classes that implement the web services. These marshal and de-marshal between XML and Java objects. These classes are created in `common/generated/src/main/java/au/edu/archer/services/icat/ws`.
    b. It also creates ICATService: a template for code that any client wishing to use the web services should instantiate.
    c. It also creates ICATPort, an interface representing the entry point for the web services that you should implement.
5. Classes are written to implement the ICAT interface:
    a. ICATHibernateImpl implements ICAT and contains the generated database access layer.
    b. ICATWsClientImpl implements ICAT, and contains an ICATService instance, which accesses the ICAT web service. This code is not part of ICAT, but would be used by a third party ICAT client.
6. ICATPortImpl implements ICATPort, and contains an ICATHibernateImpl.

Building ICAT

Directory	Contains
common	Where most of the code is.
webservice	Hand-written Spring configuration files.
src	Documentation in Maven structure.
clients/python	Python web service bindings generated by ZSI.
generate-code	The script that was used to generate the WSDL. Included only for reference.

ICAT is built with Maven. First, install the code into a Maven repository as follows:

```
$ mvn install
```

Compile it as follows:

```
$ mvn compile
```

A copy of the complete Maven repository is available for download from the ARCHER site.