ICAT interoperability

python and the ICAT WebService

19. April 2012 | Christian Felder

JCNS | Scientific Computing

ICAT interoperability

Table of contents

Part I: Motivation

Part II: Development Environment

Part III: Interoperability

ICAT interoperability

Part I: Motivation

19. April 2012 | Christian Felder

JCNS | Scientific Computing

Motivation

- catalogize measurements at the JCNS
- instruments follows Jülich Munich Standard
  - python
  - Qt
  - TACO/TANGO
- ICAT
  - WebService on top of a relational DBMS written in Java
  - platform independent
  - programming language independent (?)

integrate the ICAT WebService into the measurement software ⇒ python interface

Development Environment

Server side

- Operating System: CentOS 6
- Java SE 6 Update 31
- Java EE 6 SDK Update 4
- Oracle Database Express Edition 11g Release 2
- Oracle Database 11g Release 2 (11.2.0.3) JDBC Drivers (Oracle Java Database Connector - ojdbc6.jar)
- Apache Ant 1.8.3
- Apache Maven 3.0.4
- ICAT 4.0 (r1583)
- TOPCAT (r144)

Development Environment

Client side

- Java SE 6
- python 2.6.4
- suds package
  - lightweight soap-based WebService client
  - object-like API
  - runtime WSDL encoding/decoding

Interoperability

One Example

prerequisites:

- ICAT root user
- facility rule for the root user (CRUD)
- facility with shortname JCNS created

Interoperability

Interface: uk.icat3.sessionsbeans.manager.BeanManagerLocal

```
EntityBaseBean get(String sessionId, String query,
                   Object primaryKey) throws ...

  String sessionId    ICAT Session Id
  String query        type of object(s) to retrieve
  Object primaryKey   unique identifier for an object
```

Interoperability
Java

```
String facilityId = "JCNS";
Facility f = (Facility) icat.get(sessionId, "Facility",
                                             facilityId);
```

- returns a Facility Object
- generic interface - can return different type of objects
- needs explicit typecast

Interoperability
Python

```
#!/usr/bin/env python
from suds.client import Client

_URL = "http://replaceme.com:8080/ICATService/ICAT?wsdl"
_PASSWD = "your_icat_root_password!"

_c = Client(_URL)
_sessionId = _c.service.login("root", _PASSWD)

_facilityId = "JCNS"
_f = _c.service.get(_sessionId, "Facility", _facilityId)
```

'You have provided an instance of an incorrect PK class for this find operation. Class expected : class **java.lang.String**, Class received : class **com.sun.org.apache.xerces.internal.dom.ElementNSImpl**.'

Interoperability
Problems

related to this example: get a facility

- get method of the ICAT Service expects `java.lang.String` as primaryKey
  → no Java Datatypes in other languages, e. g. python
- weakly typed variable cannot be marshaled/unmarshaled

  `Object primaryKey`

  ⇒ python string type will be mapped to ElementNSImpl

Interoperability
Improvements

- keep existing generic interface → backwards compatibility
- add additional methods with strong typed variables, e.g. (`primaryKey`) ⇒ marshaling/unmarshaling works for other languages

Interoperability

Example: icat3-exposed - uk.icat3.sessionbeans.ICAT

```
@WebMethod
public EntityBaseBean getFacility(
        @WebParam(name = "sessionId") String sessionId,
        @WebParam(name = "primaryKey") String primaryKey
) throws ...

  return beanManagerLocal.get(sessionId, "Facility",
                             primaryKey);
```

Interoperability

Example: icat3-exposed - uk.icat3.sessionbeans.ICAT

or for complex queries, e.g. `INCLUDE` - keep query String:

```
@WebMethod
public EntityBaseBean getFacility(
        @WebParam(name = "sessionId") String sessionId,
        @WebParam(name = "query") String query,
        @WebParam(name = "primaryKey") String primaryKey
) throws ...

  return beanManagerLocal.get(sessionId, query,
                             primaryKey);
```

Thanks for your attention

Christian Felder

Jülich Centre for Neutron Science
Scientific Computing

c.felder@fz-juelich.de