Java Development for iRODS

Java Development Update
iRODS User Group Meeting 2015
Mike Conway
What is the deal?

- It may appear confusing from the outside, DICE shifted to DFC and the Consortium picked up the core server.
- Python developed by iPlant and picked up by Antoine in Consortium.
- Java and PHP maintained by me at DFC, for DFC development, with PHP ‘catch as catch can’. I’m now 10% Consortium so I can do some support work.
Client Dev Going Forward

- Part of 4.2+ strategy
- Components
  - Migrate functionality back to server as coarse-grained operations, less chatty
  - Standard representation of iRODS protocol (e.g. Avro or ProtoBuf) with generated object bindings
  - Phased migration across all clients
    - (PHP, Python, Java, C++)
DICEData Intensive Cyber Environments

iRODS and related projects from DICE

Chapel Hill, NC  http://www.irods.org  unc.dice@gmail.com

Filters  Find a repository...

+ New repository

DFC-Mobile

JavaScript  0  1

forked from matt1123/iRODS-App
Mobile Client for DFC based on Cordova
Updated an hour ago

DICE-Maven

DICE Maven Repository
Updated 17 hours ago

jargon

Java  10  7

Jargon core libraries
Updated 17 hours ago

iroids-cloud-browser

JavaScript  2  0

(incubation) iRODS cloud browser, follow on to iDrop
Updated 20 hours ago

People

Teams

Owners
4 members - 40 repositories

Admins
2 members - 0 repositories

Collaborators
3 members - 2 repositories

Create new team

Invite someone
Let’s Build Jargon

- git clone https://github.com/DICE-UNC/jargon.git
- cd jargon
- mvn install -Dmaven.test.skip=true
Best we can do for now, sorry. Working with Consortium to do more friendly deployment automation.

For now we host Maven Repo using GitHub web server

https://github.com/DICE-UNC/DICE-Maven
Example POM using Jargon - set GitHub repo

```xml
<repository>
    <id>dice.repository snaps</id>
    <name>dice.repository.snapshots</name>
    <releases>
        <enabled>true</enabled>
    </releases>
    <snapshots>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
        <checksumPolicy>warn</checksumPolicy>
    </snapshots>
</repository>

<repository>
    <id>dice.repository</id>
    <name>dice.repository</name>
    <url>https://raw.githubusercontent.com/DICE-UNC/DICE-Maven/master/releases</url>
    <releases>
        <enabled>true</enabled>
    </releases>
    <snapshots>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
        <checksumPolicy>warn</checksumPolicy>
    </snapshots>
</repository>
```
Example POM using Jargon - set Jargon coordinates

```xml
<dependency>
  <groupId>org.jboss.resteasy</groupId>
  <artifactId>resteasy-spring</artifactId>
  <version>${resteasy.version}</version>
</dependency>
<dependency>
  <groupId>org.irods.jargon</groupId>
  <artifactId>jargon-core</artifactId>
  <version>${jargon.version}</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.8.2</version>
  <scope>test</scope>
</dependency>
```
Setting Up Testing

- mvn install without skip flag will run all tests
- A test server needs to be configured with expected resources, users, see the setup scripts for stand-alone and federated setup in jargon/test-scripts
- Settings for maven in home/.m2/settings.xml to match configured grid, see sample-maven-settings.xml in jargon/test-scripts
- mvn install will create a testing.properties file in src/test/resources and launch the testing
- See: https://github.com/DICE-UNC/jargon/wiki/Setting-up-unit-tests
Quick View

- IRODSAccount defines target grid and identity

- IRODSSession
  - provides hooks to retrieve a service factory and return it
  - provides a cache of expensive things, like configuration and server profile information

- IRODSSProtocolManager
  - pluggable source of connections (one per request, pool, cache)
Session

- Based on Hibernate session,
- Each connection stateful, ThreadLocal
Factory and Services

- IRODSAccessObjectFactory provides a way to obtain service objects
- DAO -> AO
FileFactory provides java.io

- IRODSFile, InputStream, OutputStream, etc
<beans:bean id="irodsConnectionManager"
    class="org.irods.jargon.core.connection.IRODSSimpleProtocolManager"
    factory-method="instance" init-method="initialize" destroy-method="destroy" />

<beans:bean id="irodsSession"
    class="org.irods.jargon.core.connection.IRODSSession" factory-method="instance">  
    <beans:constructor-arg
type="org.irods.jargon.core.connection.IRODSProtocolManager" ref="irodsConnectionManager" />
</beans:bean>

<beans:bean id="irodsAccessObjectFactory"
    class="org.irods.jargon.core.pub.IRODSSession"
    factory-method="instance" ref="irodsSession" />
</beans:bean>
Let’s See Some Code

```java
IRODSAccount irodsAccount = testingPropertiesHelper
    .buildIRODSAccountFromTestProperties(testingProperties);

IRODSAccessObjectFactory accessObjectFactory = irodsFileSystem
    .getIRODSAccessObjectFactory();

String targetIrodsCollection = testingPropertiesHelper
    .buildIRODSCollectionAbsolutePathFromTestProperties(
        testingProperties, IRODS_TEST_SUBDIR_PATH);

String dataObjectAbsPath = targetIrodsCollection + '//' + testFileName;

DataTransferOperations dto = accessObjectFactory
    .getDataTransferOperations(irodsAccount);

dto.putOperation(  
    localFileName,  
    targetIrodsCollection,  
    testingProperties  
        .getProperty(TestingPropertiesHelper.IRODS_RESOURCE_KEY),  
    null, null);
```
Jenkins - We’re in Consortium CI