Jeeves Service Tester

Unit Testing of GeoNetwork XML Protocols

Just van den Broecke

just@justobjects.nl

SVN: geocat.ch/trunk/test/services_junit

XML Protocol Test

The general idea

1. Send XML Request
2. Receive XML Response
3. Check Response Values
4. Outcome: SUCCES, FAIL, UNKNOWN

Example

```
<test>
    <request url="/eng/xml.harvesting.run">
        <params>
            <id>1234</id>
        </params>
    </request>

    <response>
        <response>
            <id status="ok">1234</id>
        </response>
    </response>
</test>
```

Aims

- Quick edit/test/debug cycle
- Minimize test script writing
- Reuse test scripts
- Use existing tools (JUnit+XMLUnit)
- Automation - nightly build/test

Issues

- Run/invoke Jeeves locally (in IDE)
- Stateful protocol sequences
  for example: reuse response values
- Need (fuzzy) XML response checks
  for example: current date/time vals
- Variables in requests
  for example: harvest URIs

Run Jeeves Locally

LocalJeeves.java

```
JeevesEngine jeeves = new JeevesEngine();
String appPath = GN_HOME + "/web/geonetwork/";
String baseUrl = "/geonetwork";
String configPath = appPath + "WEB-INF/";
jeeves.init(appPath, configPath, baseUrl);

// Make session with all permissions
UserSession session = new UserSession();
session.authenticate("0", "local", "Local", "User",
                     ProfileManager.ADMIN);
                     .
                     .
```

Invoke Jeeves Directly

LocalJeeves.java

```
static public Element dispatch(String anURL, Element params)
{
   try
   {
     LocalServiceRequest serviceReq =
                  LocalServiceRequest.create(anURL, params);

     jeeves.dispatch(serviceReq, session);
     if (serviceReq != null)
     {
       return serviceReq.getResult();
     }
   } catch (Throwable t)
   {
      err("Error in service req", t);
   }

   return null;
}
```

Writing Tests - Example

HarvestTest.java

```
public class HarvestTest extends JeevesTest
{
	public static final String URL_GEOTASKSERVER=
				"http://gbs02.geotaskserver.com/home_gs/services/MessageService";

	@Test
	public void testGNHarvesterLifeCycle() throws Exception
	{
		doTest("gn-harvester-add.xml");
		doTest("gn-harvester-get-inactive.xml");
		doTest("harvester-run-inactive.xml");
		doTest("harvester-start.xml");
		doTest("gn-harvester-get-active.xml");

		doTest("harvester-stop.xml");

		// Inactive status
		doTest("gn-harvester-get-inactive.xml");
		doTest("harvester-remove.xml");
	}

	@Test
	public void testCGPHarvesterLifecycle() throws Exception
	{
		setVariable("url", URL_GEOTASKSERVER);
		doTest("cgp-harvester-add.xml");
```

Writing Tests - Example

gn-harvester-add.xml

```
<test>
    <request url="/eng/xml.harvesting.add">
        <params type="geonetwork">
            <site>
                <name>gn.fao</name>
                <host>www.fao.org</host>
            </site>
        </params>
    </request>

    <response>
        <node type="geonetwork" id="${id}">
            <site>
                <name>gn.fao</name>
                <uuid>${uuid}</uuid>
                <account>
                    <use>false</use>
                    <username/>
                    <password/>
                </account>
                <host>www.fao.org</host>
                <port>80</port>
                <servlet>geonetwork</servlet>
            </site>
            <options>
                <every>90</every>
                <oneRunOnly>false</oneRunOnly>
                <status>inactive</status>
            </options>
              .
              .
        </node>
    </response>
</test>
```

The Magic in doTest()

JeevesTest.java

```
public void doTest(String aFileName, DifferenceListener differenceListener) throws Exception
{
      // Load the file containing the test
      Element testElm = loadXMLFile(aFileName);

      // Get request-fragment
      Element reqElm = testElm.getChild(TAG_REQUEST);

      // Expand possible symbolic XmlVars
      setRequestVars(reqElm);

      // Do the Jeeves dispatch
      Element result = LocalJeeves.dispatch(reqElm);

      // Expected elm is first child of response-fragment
      Element expectedRspElm = (Element) testElm.getChild(TAG_RESPONSE).getChildren().get(0);
      processResponseVars(expectedRspElm, result);

      // Compare the expected XML and actual response XML using XMLUnit
      String strExpected = getString(expectedRspElm);
      String strResult = getString(result);
      Diff diff = new Diff(strExpected, strResult);

      if (differenceListener != null)
      {
            diff.overrideDifferenceListener(differenceListener);
      }
      assertXMLEqual("comparing test xml to control xml", diff, true);
}
```

Working with Variables

Set in requests (step 1: XML)

```
<test>
  <request url="/eng/xml.harvesting.add">
    <params type="cgp">
      <site>
       <name>geocat.ch</name>
       <url>${url}</url>
       <icon>cgp.gif</icon>
      </site>
            <searches>
                <search>
                    <latNorth>${latNorth}</latNorth>
                    <latSouth>${latSouth}</latSouth>
                    <lonEast>${lonEast}</lonEast>
                    <lonWest>${lonWest}</lonWest>
                </search>
            </searches>
```

Working with Variables

Set in requests (step 2: Java)

```
@Test
public void testCGPHarvesterBboxSearch() throws Exception
{
     setVariable("url", URL_GEOTASKSERVER);
     setVariable("latNorth", "48");
     setVariable("latSouth", "44");
     setVariable("lonEast", "11");
     setVariable("lonWest", "4");

doTest("cgp-harvester-add-search-bbox.xml");
          .
          .
```

Working with Variables

Reuse from response - Step 1

```
<test>
    <request url="/eng/xml.harvesting.add">
        <params type="geonetwork">
            <site>
                <name>gn.fao</name>
                <host>www.fao.org</host>
            </site>
        </params>
</request>

<response>
    <node type="geonetwork" id="${id}">
        <site>
            <name>gn.fao</name>
            <uuid>${uuid}</uuid>
```

Working with Variables

Reuse from response - Step 2

```
<test>
    <request url="/eng/xml.harvesting.remove">
        <params>
            <id>${id}</id>
        </params>
    </request>

    <response>
        <response>
          <id status="ok">${id}</id>
        </response>
    </response>
</test>
```

Running Tests

1. Directly in IDE - standard JUnit
2. SuiteRunner.java - see AllTests.java
3. Ant
    A. SuiteRunner.java
    B. Ant JUnit Task with doc. (in progress)

Further Work

- Review
- Run with remote webservice
  basically renders generic XML tester
- Ant tasks, in particular reporting
- Backport to GN trunk ?
- Ideas ?