LOD2 Demonstrator Development Installation Guide

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1.0	march 2011	Initial version	BVN

Contents

The demonstrator installation guide describes how to get, compile, install and deploy the LOD2 demonstrator. The early version of the demonstrator is the delivery of Workpackage WP1.4.

1 Required software

This section describes the setup of the necessary software to get the demonstrator working.

1.1 Java

The programming language Java. We install version 1.6.

```
sudo apt-get install openjdk-6-jdk
```

An alternative is

```
sudo apt-get install sun-java6-jdk
```

1.2 Apache Tomcat 6

A popular servet container of which we install version 6.

```
sudo apt-get install tomcat6
sudo apt-get install tomcat6-admin
```

In order to access the tomcat manager page an administrator has to be defined. Edit the tomcat-users.xml file:

```
sudo vim /etc/tomcat6/tomcat-users.xml
```

and add the next 2 lines

```
<role rolename="manager"/>
<user username="admin" password="admin" roles="manager"/>
```

Adapt the username and password to your conventions.

Restart tomcat

```
sudo service tomcat6 restart
```

to make the changes have effect.

Some installation locations:

- tomcat server : /usr/share/tomcat6
- tomcat webspace : /var/lib/tomcat6

1.3 Apache Maven

An easy to use build system.

```
sudo apt-get install maven2
```

1.4 Apache Subversion

Subversion is an open source version control system.

```
sudo apt-get install subversion
```

1.5 Eclipse

A IDE for software development.

```
sudo apt-get install eclipse
```

Note
Per default the latest version Galileo is installed. However it seems that many plugins for popular other tools do have problems.

1.6 OpenRDF framework

Download the openRDF developers kit from http://sourceforge.net/projects/sesame/files/Sesame%202/2.3.3/openrdf-sesame-2.3.3-sdk.tar.gz. Unpack and deploy the wars in tomcat.

1.6.1 Vaadin plugin

Vaadin is a JAVA framework to build web applications. See http://vaadin.com/.

1.6.2 Maven plugin

The installation in the Galileo version did not succeed yet.

2 Installing the LOD2 stack components

The LOD2 consortium has tried to collect all components together in one repository. All the components are present as debian packages [1].

2.1 Installation of the LOD2 repository

Browse to stack.lod2.eu. Select the http://stack.lod2.eu/deb/lod2/lod2repository_1.2_all.deb. Let the Ubutu package manager install it. This updates the package repository description.

Update now the repository content.

```
sudo apt-get update
```

2.2 Installation of the individual LOD2 components

As for now one has to install the components individually.

[1] Poolparty is not distributed via these package system.

2.2.1 Virtuoso

Virtuoso is an RDF database.

```
sudo apt-get install virtuoso-opensource
```

During the installing you will be asked to set the dba (root) password of the virtuoso database.

2.2.2 Ontowiki

An ontological authoring tool of RDF based knowledge.

```
sudo apt-get install ontowiki
```

After installation ontowiki is accessible locally in your browser as

```
http://localhost/ontowiki
```

Note
For the moment the mysql version of ontowiki is installed. This should be replaced with virtuoso variant.

3 Setting up a demonstrator project from scratch

The project setup, build process and dependency management is done via Maven (http://maven.apache.org/).

3.1 Creating an Hello World Vaadin application

The userguide for setting up a Vaadin Maven project is http://vaadin.com/wiki/-/wiki/Main/Using%20Vaadin%20with%20Maven.

The following maven call will create a hello world vaadin application

```
mvn archetype:generate -DarchetypeGroupId=com.vaadin -DarchetypeArtifactId=vaadin-archetyp
```

In the setup some dependencies have to be resovled manually by publishing som jars in your local maven repository.

3.2 Installation of the OpenRDF workbench

This is a workbench to inspect RDF stores with a SPAQRL interface. It shows that the LOD2 component virtuoso can be accessed via other RDF frontends.

Secondly the installation opens the access to the database via one of the defacto standard API's for RDF applications.

Activating the openrdf-sesame repository server in tomcat requires to setup the log directory. According to the manual the following property should be adapted: -Dinfo.aduna.platform.appdata.basedir=/my/log/directory.

in JAVA_OPTS or CATALINA_OPTS

I could not get it working hence I adopted the following solution. I created in the logs directory of the deployment space of tomcat6

```
cd /var/libs/tomcat6/logs
```

a new directory

```
sudo mkdir .aduna
```

make the directory public accessible

```
sudo chmod 777 .aduna
```

make a link from the tomcat6 server location

```
cd /usr/share/tomcat6
sudo ln -s /var/libs/tomcat6/logs/.aduna
```

3.3 Activating Virtuoso in the OpenRDF sesame workbench

The extended manual is found at

```
http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtSesame2HttpRepository
```

Download location: http://www.openlinksw.com/dataspace/dav/wiki/Main/VOSDownload
Bassically it consists of the following operations: NOTE: to be checked the directories.

1. download the virtuoso jars or find them in the distribution packages
2. copy some jars to the tomcat webapps openrdf-sesame and openrdf-workbench

   ```
   virt_sesame2.jar
   virtjdbc3.jar

   to the WEB-INF/libs/
   ```

3. copy xlst scripts with configuration information to the openrdf-sesame webapp

   ```
   cp create.xlst
      create-virtuoso.xlst

   to WEB-INF/transformations
   ```

3.4 Activating Virtuoso Sesame drivers in Maven project

Add the following dependencies to the pom.xml

```
<dependency>
        <groupId>virtuoso.sesame2</groupId>
        <artifactId>driver</artifactId>
        <version>2.0</version>
        <scope>compile</scope>
</dependency>
<dependency>
        <groupId>virtuoso</groupId>
        <artifactId>jdbc3</artifactId>
        <version>3.0</version>
        <scope>runtime</scope>
</dependency>
```

The files have to be registered manually in your maven repository.

```
mvn install:install-file -DgroupId=virtuoso.sesame2 -DartifactId=driver -Dversion=2.0 -Dpa
mvn install:install-file -DgroupId=virtuoso -DartifactId=jdbc3 -Dversion=2.0 -Dpackaging=j
```

3.5 Supporting Tomcat deployment with Maven

Extend the pom.xml with the following plugin

```
<plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>tomcat-maven-plugin</artifactId>
        <!--<version>1.0-beta-1</version>-->
        <version>LATEST</version>
        <configuration>
                <server>tomcat6</server>
                <url>http://localhost:8080/manager</url>
        </configuration>
</plugin>
```

It requires the tomcat server to be defined in your local maven settings as a server

In .m2/settings.xml

add the server specification

```
<server>
        <id>tomcat6</id>
        <username>admin</username>
        <password>admin</password>
</server>
```

This allows you to (re)deploy your application in tomcat.

```
mvn tomcat:deploy    // first time
mvn tomcat:redeploy  // when the project has been already been deployed.
```

3.6 Accessing the Maven repository of openRDF

This section decribes how the Sesame API can be integrated in the maven project by adding the following configuration to the project's pom.xml.

The Aduna maven repository containing the openRDF Sesame API is

```
<repository>
  <id>openRDF</id>
  <url>http://repo.aduna-software.org/maven2</url>
</repository>
```

The Aduna maven repository is structured so that one first has to import the dependencies before declaring the use of it. The section <dependencyManagement> has to be before the <dependencies> tag.

```
<dependencyManagement>
  <!-- For Sesame dependency one requires to import first the pom dependencies -->
  <dependencies>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
```

```
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-model</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-query</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-queryalgebra-evaluation</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-repository</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-rio-api</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.openrdf.sesame</groupId>
      <artifactId>sesame-rio-rdfxml</artifactId>
      <version>2.3.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

These import dependencies allow you to declare the project dependencies

```
<dependencies>
  ...
  <dependency>
    <groupId>org.openrdf.sesame</groupId>
    <artifactId>sesame-repository-api</artifactId>
    <version>2.3.3</version>
  </dependency>
```

```
  ...
</dependencies>
```

The Sesame openRDF library depends on the slf4j logger library. The maven repository is

```
<repository>
  <id>slf4j</id>
  <url>http://repo2.maven.org/maven2/org/slf4j/</url>
</repository>
```

3.7 Supporting Vaadin AddOns with Maven

Vaadin allows to extend the default widget set by custom build addons. Vaadin offers a Vaadin AddOn repository. In order to get them working via only Maven some conventions should be known. The online information demonstrates the Eclipse usage.

4 Publishing Debian Package

4.1 Setting up maven to build the debian packages

Follow the instructions of to set http://europatech.blogspot.com/search/label/debian

install dput (debian package put)

```
sudo apt-get install dput
```

configure dput in your homedir: add the next configuration info to .dput/dput.cf

```
[lod2]
fqdn = stack.lod2.eu
method = scp
login = packaging
incoming = /var/www/stack.lod2.eu/deb/mini-dinstall/incoming
```

install dch (debian package change)

```
sudo apt-get install devscripts dh-make
```

install fakeroot

```
sudo apt-get install fakeroot
```

Create the basic template for the application package. This is done by creating a directory with the name of the package in the form of <package>-<version>

```
cd target/
mkdir <package>-<version>
cd <package>-<version>
dh_make -c apache -n -s -i -e Bert.Van.Nuffelen@tenforce.com
```

Move the generated files to the maven soure directory.

```
mkdir src/deb-package
cp -r debian src/deb-package
```

Adapt the changelog to be of the form:

```
lod2demo (1.0) lod2; urgency=low
  * Initial Release .
  -- Bert Van Nuffelen (TenForce/LOD2) <Bert.Van.Nuffelen@tenforce.com>  Tue, 24 May 2011
```

where it is important to set the package distribution identifier to lod2 instead of unstable otherwise it will be blocked at the upload to the lod2 stack.

Add the build script build_debpkg.sh to src/deb-package

```
#!/bin/sh

MODULE_NAME=lod2demo_1.0

echo "Building Debian package for ${MODULE_NAME}"
echo

rm -rf ../../target/deb-pkg
mkdir -p ../../target/deb-pkg

# Extract the tarball to the package workspace
#tar xfz data.tar.gz --directory ../../target/deb-pkg

# copy war file to package workspace
cp ../../target/lod2demo-1.0.war ../../target/deb-pkg
# Add the Debian control files
cp -r debian ../../target/deb-pkg

# Build the package and sign it.
cd ../../target/deb-pkg
debuild --check-dirname-level 0 -b
```

The package contains a war file which must be installed at the appropriate place during the installation process of the debian package.

```
lod2demo-1.0.war /var/lib/tomcat6/webapps
```

The package is automatically signed by the debuild command at the end of the script.

Extend the pom.xml with

```
<profiles>
    <profile>
       <id>deb-pkg</id>
       <build>
          <plugins>
             <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-antrun-plugin</artifactId>
                <configuration>
```

```
                  <tasks>
                     <echo
                        message="Creating deb package">
                     </echo>
                     <exec
                        dir="${basedir}/src/deb-package"
                        executable="${basedir}/src/deb-package/build_debpkg.sh"
                        failonerror="true">
                     </exec>
                  </tasks>
               </configuration>
               <executions>
                  <execution>
                     <id>deb-pkg</id>
                     <phase>package</phase>
                     <goals>
                        <goal>run</goal>
                     </goals>
                  </execution>
               </executions>
            </plugin>
         </plugins>
      </build>
   </profile>
</profiles>
```

build the package

```
mvn package -Pdeb-pkg
```

debuild

```
use it debuild --check-dirname-level 0 -b
```