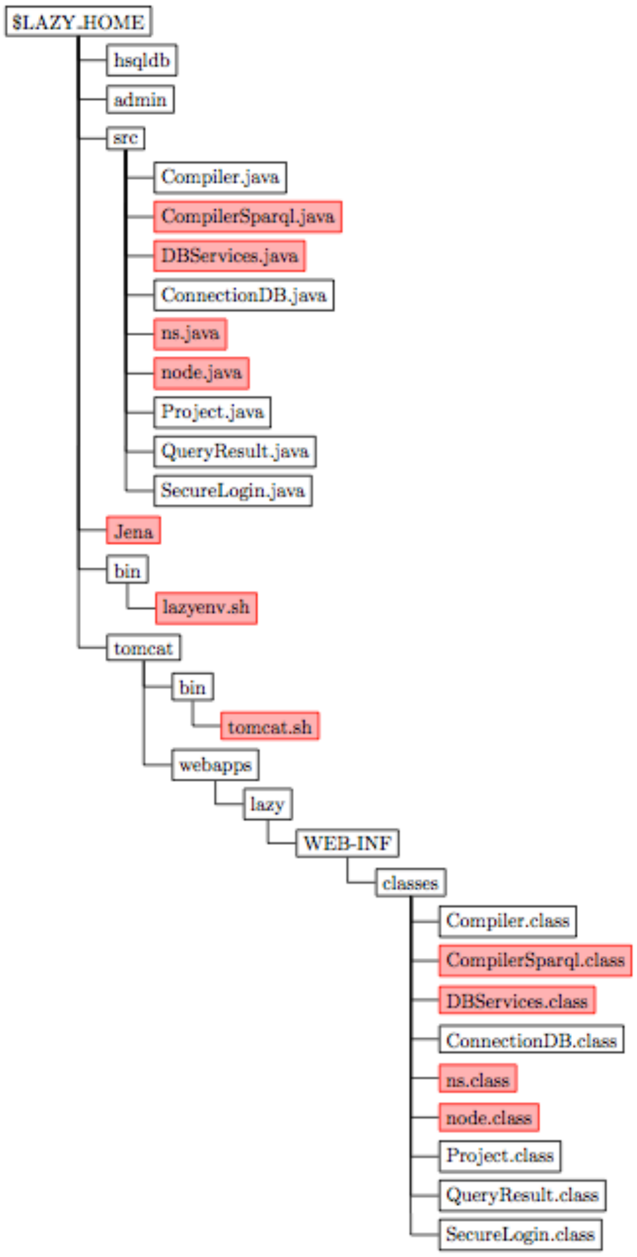
Implementation on Lazy to access SAPRQL endpoints

Lin ZHANG

This paper points out which files in Lazy system have been modified in order to render Lazy system access SPARQL endpoints.

All implementation is based on Lazy Hypertext view system[1].

The following directory structure tree is all modified files in the package, including new files and modified files, marked with red.



[1] http://sourceforge.net/projects/lazy-x/

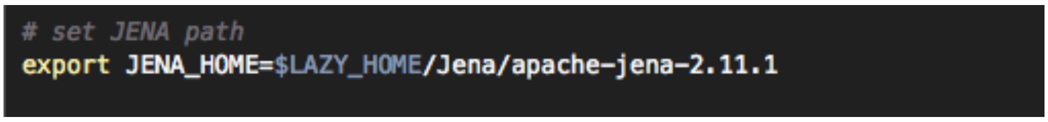
Import and configure Jena

Jena is a framework made up of different APIs to interacting together to process RDF data. The latest version could be downloaded from https://jena.apache.org/download/index.cgi.

Please put all uncompressed Jena files under Lazy root directory $LAZY_HOME.

1. Set Jena path into Lazy environment

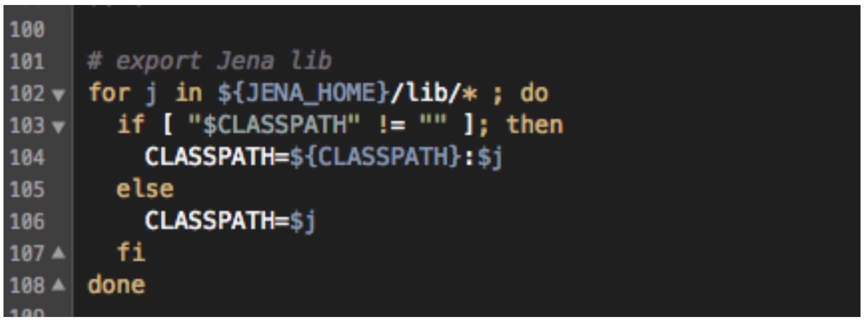
Put all unzipped Jena files under $LAZY_HOME, then set Jena path into Lazy environment. Add the following lines into $LAZY_HOME/bin/lazyenv.sh



```
# set JENA path
export JENA_HOME=$LAZY_HOME/Jena/apache-jena-2.11.1
```

2. Load all of Jena libs when execute runServer.sh

Add the following lines into:
$LAZY_HOME/tomcat/bin/tomcat.sh



```

# export Jena lib
for j in ${JENA_HOME}/lib/* ; do
  if [ "$CLASSPATH" != "" ]; then
    CLASSPATH=${CLASSPATH}:$j
  else
    CLASSPATH=$j
  fi
done
```

Java implementation

All source code are under directory $LAZY_HOME/src. If you want to execute the program correctly, please put all compiled files *.class under directory $LAZY_HOME/tomcat/webapps/lazy/WEB-INF/classes

CompilerSparql.java

New java class, similar to compiler.java, aiming to parse Lazy node to access SPARQL endpoints.

ns.java

void compileProject() —
Node server will distinguish the project to access different data sets — database or sparql endpoints — invoke either compiler or sparql compiler. This is a temporary implementation.

Node.java

public static Node getNodeDefinition() —
Fill node.preSQL, node.itemSQL, node.postSQL, according to nodetype — either “sparql” or default.

static void doQuery() —
If this is a sparql node and the executed query is item query (neither pre query or post query), send this query to sparql endpoint. The program will read node.collection, try to analyze whether the collection is a URL start with “http://”, then decide to send this query to database or endpoints.

static String formatSparqlDisplayString() —
New method, used to delete ‘,’and ‘” from node.items for displaying sparql content

static static String beautifyDisplay() —
New method. this method is used to cut off prefix string from content URI

DBServices.java

public static ResultSet execSparql() —
New method, execute sparql query to endpoints, return a resultset.

Some potential improvement

Project type problem:

When a project is started with “SPARQL”, the node server will take this project as a sparql project and will invoke SparqlCompiler, such as “SPARQL_DBpedia” and “SPARQL_DrugBank”.
The better way to deal with this is to add a field in project definition page, named “project type”, used to distinguish a project is used to access either database or sparql endpoints.

Related method: ns.compileProject() node.doQuery()

SPARQL node prefix problem:

As we all know, each sparql query has a list of prefix. Sparql queries access a same endpoint may share some common prefix. Now the prefix list is defined at the end of node definition. In the future, the project definition page could provide users a field to write down common prefix. Users only need to define unique prefix for each node.

Related class: CompilerSparql.java