Query Service

IHTSDO

Copenhagen K
Gammeltorv 4, 1.
1457
Denmark

Table of Contents

Project overview

The OTF-Query-Services module is a component of the IHTSDO Open Terminology Tooling Framework that implements queries that can be conducted against a Berkeley SNOMED database. The project illustrates examples of how queries can be executed from within Java or from a REST client server. The input parameters and syntax of the queries is derived from XQuery FLWOR expressions [http://en.wikipedia.org/wiki/FLWOR].

Project requirements

A working knowledge of Maven is a prerequisite to understanding how to build and work with the query service. More information on Maven is available at the Apache Maven [http://

maven.apache.org] site. Maven supports project aggregation in addition to project inheritance through its module structure. See Maven's Guide to Working with Multiple Modules [http://maven.apache.org/guides/mini/guide-multiple-modules.html] and Maven's Introduction to the POM [http://maven.apache.org/guides/introduction/introduction-to-the-pom.html] for more information about Maven modules, project inheritance, and project aggregation.

The minimum software requirements for the project are as follows:

1. Apache Maven 3.1.0 [http://maven.apache.org/download.cgi]

2. Java 1.7 [http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html] or higher

If you plan to perform queries using the REST service, you will need to install an Apache Tomcat 7.0 server [http://tomcat.apache.org/download-70.cgi].

Setup

Maven

In order to build the Query Service project, developers must either install Maven 3.1.0 to use from the command line, use an IDE that has Maven 3.1.0 already integrated (IntelliJ IDEA, Netbeans), or add a plugin to their IDE that adds Maven 3.1.0 support. Instructions on installing Maven for use from the command line are available at Maven's download site [http://maven.apache.org/download.cgi]. Integration information for IntelliJ IDEA is available from the Jetbrains wiki for creating and importing maven projects [http://wiki.jetbrains.net/intellij/Creating_and_importing_Maven_projects]. Integration information for Netbeans is available from Netbean's Maven wiki page [http://wiki.netbeans.org/Maven].

Please note that the project build requires the most recent version of Maven, 3.1.0, and Java 1.7 or higher. Ensure that you are running the minimum requirements for Maven and Java by entering the command `mvn --version`, which will output the Maven and Java versions running on your machine. If the output displays a Java version earlier than 1.7, download it here [http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html] and ensure that JAVA_HOME and PATH variables are set to point at JDK 1.7 or higher.

Once Maven is configured, you will need to download JavaFX artifacts to your local repository by running the command `$ mvn org.codeartisans.javafx:javafx-deployer-maven-plugin:1.2:install` from a directory that is not a Maven project.

To access the artifact dependencies necessary to build the project, the Maven `settings.xml` file must be appropriately configured. More information about the `settings.xml` file and its location is available on Maven's Settings Reference [http://maven.apache.org/settings.html] web page. As noted in the Settings Reference page, locate and edit the settings.xml in the directory `{user.home}/.m2/` .

For build profiles other than the default build, developers will need an account to download a SNOMED CT berkeley database in the IHTSDO's repository. Credentials can be requested from Rory Davidson. All of the other artifacts required for the project are open source resources from Apache. Below is a copy of a `settings.xml` file that allows a user to conduct the default build. In order to perform build profiles other than the default build, enter IHTSDO Maestro credentials into the username and password fields.

Example 1. Example settings.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
"http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.
     <servers>
         <server>
             <id>maestro</id>
             <username>username</username>
             <password>password</password>
          </server>
      </servers>
      <mirrors>
         <mirror>
             <id>maestro</id>
             <mirrorOf>external:*</mirrorOf>
             <name>IHTSDO Maestro</name>
 <url>https://mgr.servers.aceworkspace.net/apps/ihtsdo-archiva/repository/all/<
         </mirror>
     </mirrors>
</settings>
```

Java

The Query service project requires Java 1.7 or higher. To give Java more heap space, you should set the following JAVA_OPTS: `-J-Xss2m -J-Xms1g -J-Xmx2g -J-XX:PermSize=1600m`. If you're using the command line, these options can be set with the command `JAVA_OPTS set -J-Xss2m -J-Xms1g -J-Xmx2g -J-XX:PermSize=1600m`. Within Netbeans, these options should be added to the `netbeans_default_options` in the `netbeans.conf` file.

Tomcat setup

Apache Tomcat 7.0 can be downloaded here [http://tomcat.apache.org/download-70.cgi]. Configure the following properties in `.bash_profile`:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.7.0_45.jdk/Contents/Hom
export CATALINA_HOME="/Applications/apache-tomcat-7.0.42"
export CATALINA_OPTS="-Xms4g -Xmx6g -XX:MaxPermSize=512m -Dorg.ihtsdo.otf.tcc.r
-Duser.home=${CATALINA_HOME}/temp/bdb"
```

Ensure that you replace This link [http://wolfpaulus.com/jounal/mac/tomcat7] gives explicit instructions on how to configure your machine to launch Tomcat from the command line with `startup.sh` and `shutdown.sh` scripts in OS X.

If you want to run Tomcat within Netbeans, this link [http://technology.amis.nl/2012/01/02/installing-tomcat-7-and-configuring-as-server-in-netbeans/] gives directions on how to download Apache 7.0 and configure it to run within Netbeans.

In order to gain access to the Tomcat Web Application Manager, you need to configure a Tomcat user with the appropriate privileges by editing the `tomcat-users.xml` file that is stored in `CATALINA_HOME/conf/`. Here is a sample `tomcat-users.xml` file that will setup a user named "admin" with a password "password" and privileges to access the Tomcat Web Application Manager:

Example 2. Sample tomcat-users.xml

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>

  <role rolename="manager"/>
  <user username="admin" password="password" roles="tomcat,manager,manager-gui,

  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
</tomcat-users>
```

If you want to perform REST queries longer than 9000 characters, then edit the `maxHttpHeaderSize` parameter by conducting the following two changes:

1. Edit the `server.xml` file in the directory `CATALINA_HOME/conf` to include your desired `maxHttpHeaderSize`as follows:

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"
           maxHttpHeaderSize="15000"/>
```

2. Edit the `param-value` for the `httpMaxHeaderSize` in the `web.xml` file in the directory `OTF-Query-Services/query-service/src/main/webapps/WEB-INF` as shown:

```
<init-param>
    <param-name>httpMaxHeaderSize</param-name>
    <param-value>15000</param-value>
</init-param>
```

Project checkout

The Query service project can be checked out anonymously from GIT by entering the prompt `$ git clone https://github.com/IHTSDO/OTF-Query-Services.git` in an empty directory folder.

Query service build profiles

The query service defines five build profiles described in the following sections. For more information on build profiles, see Maven's Introduction to build profiles [http://maven.apache.org/guides/introduction/introduction-to-profiles.html]. The default build is the only build profile that can be conducted without acquiring IHTSDO Maestro credentials.

A developer can execute the builds from the command line, using the appropriate command described below, or from an IDE that supports Maven by selecting the desired build profile.

Default build profile

The default build profile consists of the modules that build when no profile is specified. By default the following modules are built:

1. Client

2. JAXB objects

This profile will provide a sufficient build to test the query client with the provided settings.xml file.

The artifacts required for the default build are located in the IHTSDO public Maven repository, and a user can perform the default build without IHTSDO Maestro credentials.

The default build can be conducted from the console with the command `$ mvn clean install`.

Query Service build profile

This profile will build the query service and dependent modules. This project has more dependencies, including dependencies in the IHTSDO Maven repository, which require a user account.

The command for this build is `$ mvn clean install -P query-service`.

Integration tests build profile

The integration tests build profile adds the integration tests module to the build. The integration tests are not part of the default build profile because they have an external dependency on a Berkeley SNOMED database that is rather large, and downloading and opening this database may not be necessary for all types of development. Omitting this module from the default build profile makes the default build rapid.

Assuming all of the required dependencies are installed, the build time for the integration tests module takes about 1 min 20 sec on a high-spec developers laptop, while the other modules in this project take between 0.5 and 5 seconds. To build this from the console, use the command `$ mvn clean install -P integration-tests`.

Documentation build profile

The documentation build profile adds the integration tests module and the documentation module to the build when the build profile id *documentation* is activated. Generation of documentation depends of proper execution of the integration tests module, and therefore is removed from the default build profile secondary to the resource requirements and build time of the integration tests module.

Execute the documentation profile build with the command `$ mvn clean install -P documentation`.

All build profile

Perform all of the goals listed in the above build profiles with the command `$ mvn clean install -P all`.

Query service repository structure

The query service top-level project is the query-parent project that defines the root of the repository structure. The query service repository holds a maven multi-module project, which manages the sources and documents for the project.

Maven Modules

Within the top level project are six maven modules (subprojects), some of which are only built when a particular build profile is activated.

1. Client
   - demonstrates how to connect to the query service REST server using a Tomcat 7.0 [http://tomcat.apache.org] REST client
   - group id: org.ihtsdo.otf
   - artifact id: query-client
   - directory: query-client
   - build profiles: default, all, query-service, integration-tests, documentation
2. Service
   - implements a Tomcat 7.0 [http://tomcat.apache.org] REST service for querying
   - group id: org.ihtsdo.otf
   - artifact id: query-service
   - directory: query-service
   - build profiles: all, query-service, integration-tests, documentation
3. Implementation
   - implementation of queries against Terminology Component Chronicle service
   - group id: org.ihtsdo.otf
   - artifact id: query-implementation
   - directory: query-implementation
   - build profiles: all, query-service, integration-tests, documentation
4. JAXB objects
   - generates Java data display objects derived from running the JAXB xjc against the the underlying implementation
   - group id: org.ihtsdo.otf
   - artifact id: query-jaxb-objects
   - directory: query-jaxb-objects
   - build profiles: all, query-service, integration-tests, documentation
5. Integration tests
   - conducts tests of queries against a SNOMED CT database
   - group id: org.ihtsdo.otf
   - artifact id: query-integration-tests
   - directory: query-integration-tests
   - build profiles: all, integration-tests, documentation
6. Documentation

- handles compilation of documentation fir Query Services project
- group id: org.ihtsdo.otf
- artifact id: query-documentation
- directory: query-documentation
- build profiles: all, documentation

Deploy to app server

The default build command generates a `.war` file that can be deployed to an app server. Please see the Tomcat Setup section for instructions on how to configure your Tomcat server. In order to avoid causing lock errors when opening the berkeley database, undeploy the otf app before redeploying. In order to undeploy the app, choose "undeploy" from the Mananger App (http://localhost:8080/manager/html) or from within the Tomcat server in your IDE. Here is a link [http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html] to more information about the Tomcat 7 Manager App. Based upon how you're using Tomcat, these are the instructions for deploying the `.war` file to your Tomcat server:

1. From the command line: once you have built the project from the command line: save the following xml file in the directory `{CATALINA_HOME}/conf/Catalina/localhost/` as `otf.xml` and replace the text {OTF Query Service Home} with the directory where you saved the OTF-Query-Service project on your machine.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <Context
   antiJARLocking="true" docBase="{OTF Query Service Home}/query-service/target/
   path="/otf"/>
   ```

   Then, execute the `startup.bat` or `startup.sh` file, depending on your operating system. In OS X, in order to execute `$ bash startup.sh`, you must first enter the command `$ chmod +x /bin/*.sh` from the CATALINA_HOME directory to make all of the `.sh` files executable. Test the deployment of the war with the following command with the following command from the query-client folder `$ mvn exec:java -Dexec.mainClass="org.ihtsdo.otf.query.rest.client.examples.HelloExample" package` or type the url http://localhost:8080/otf/query-service/hello/frank into a browser.

2. From Netbeans: right-click the query-service project, click "Run," and select the Tomcat 7 server.

Deploying the app creates a folder at `{user.home}/app-server` and ensures that you have the updated version of the berkeley database. You should see the following in the Tomcat localhost log when the Maven build is successful and the database has been opened:

```
Oct 28, 2013 11:43:05 AM org.apache.catalina.core.ApplicationContext log
INFO: Embedded maven build succeeded: 0
Oct 28, 2013 11:43:05 AM org.apache.catalina.core.ApplicationContext log
INFO: Finised database dependency setup: 0 min, 20 sec
Oct 28, 2013 11:43:09 AM org.apache.catalina.core.ApplicationContext log
INFO: Released storeSemaphore for init.
Oct 28, 2013 11:43:09 AM org.apache.catalina.core.ApplicationContext log
INFO: Finised database startup: 0 min, 24 sec
```

Then, it will open the berkeley database at the configured location. Please see the Tomcat Setup section for more information on how to set this location. It can take up to a couple minutes to open the database, then you will see the following output in the Tomcat Catalina log file:

```
Index: /Users/dylangrald/app-server/berkeley-db/lucene/descriptions
Oct 28, 2013 11:43:05 AM org.ihtsdo.otf.tcc.datastore.BdbTerminologyStore <init
INFO: org.ihtsdo.otf.tcc.datastore.bdb-location set. Starting from location: /U
Index: /Users/dylangrald/app-server/berkeley-db/lucene/refex
!## maxMem: 6174015488 heapSize: HEAP_6000
setup dbRoot: /Users/dylangrald/app-server/berkeley-db
absolute dbRoot: /Users/dylangrald/app-server/berkeley-db
Oct 28, 2013 11:43:05 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: NidCidMap readOnlyRecords: 952
Oct 28, 2013 11:43:05 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: NidCidMap mutableRecords: 0
Oct 28, 2013 11:43:09 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: mutable maxMem: 4224000
Oct 28, 2013 11:43:09 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: mutable shared cache: true
Oct 28, 2013 11:43:09 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: readOnly maxMem: 4224000
Oct 28, 2013 11:43:09 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: readOnly shared cache: true
Oct 28, 2013 11:43:09 AM org.ihtsdo.otf.tcc.datastore.BdbTerminologyStore <init
INFO: Database setup complete
```

Test the deployment of the `.war` by running the `HelloExample` in a web browser at the URL `{default host}/otf/query-service/hello/frank` . Please see the Query Client section below for instructions on how to run the `HelloExample` and `KindOfQueryExample`.

Module overview

Query Client

This query client module demonstrates how to connect to the query service REST server using an Apache Tomcat 7.0 [http://tomcat.apache.org] REST client and provides examples of queries that can be constructed in Java.

There are several example programs, including the `HelloExample`, `KindOfQueryExample`, and `LuceneExample`.

HelloExample

The `HelloExample` program, located in package, `org.ihtsdo.otf.query.rest.client.examples` is a simple program that sends a hello request to the rest server. If you haven't deployed the war to your Tomcat server, you can still run the `HelloExample` with the url `http://api.snomedtools.com/query-service/hello/frank`. If you deployed the war to your Tomcat server, you can run the hello example in the following ways:

1. Enter the url `http://localhost:8080/otf/query-service/hello/frank` in a browser
2. Run the command `$ mvn exec:java -Dexec.mainClass="org.ihtsdo.otf.query.rest.client.examples.HelloExample" package` from the query-client folder
3. Run the main method of `HelloExample.java`, located in the query-client module, in an IDE

Running the hello example should produce the output:

```
200

hello frank.
```

KindOfQueryExample

The `KindOfQueryExample` program, located in package `org.ihtsdo.otf.query.rest.client.examples`, performs a simple "kind of" query using the rest server and returns data display objects with the results. This example demonstrates the parameters required for a query and how to run queries using the REST service.

The structure of a query is defined by:

- a `ViewCoordinate` that defines what version of the terminology to query against, as well as other information such as the preferred language for results.
- a FOR that defines the set of components to iterate over
- a LET that defines references to concept specifications or other view coordinates used by where clauses.
- a WHERE that defines the where clauses for the query
- a RETURN that defines that type of components to return (concepts, descriptions, etc).

The following parameters are required for a query:

1. Let declarations
2. Where clauses

If a ViewCoordinate is not specified, then the Snomed inferred latest will be used. The default FOR set is all concepts. And the default RETURN value is the fully specified description version of the components in the result set.

This KindOfQueryExample's main method will setup a kind-of query that will return concepts that are a kind-of SNOMED concept "allergic asthma." If a server is not specified, a default server is chosen from the QueryProcessorForRestXml class, located in the package: `org.ihtsdo.otf.query.rest.client`.

Run this example in the following ways:

1. If you haven't deployed the war to a Tomcat server, you can still run the example with the command `$ java -cp query-client/target/query-client-1.1-SNAPSHOT-jar-with-dependencies.jar org.ihtsdo.otf.query.rest.client.examples.KindOfQueryExample http://api.snomedtools.com` from the top level directory of the OTF-Query-Service.
2. If you're running Tomcat from the command line, navigate to the query-client folder and enter the command `$ mvn exec:java -Dexec.mainClass="org.ihtsdo.otf.query.rest.client.examples.KindOfQueryExample` package
3. If you're running Tomcat from an IDE, such as Netbeans, then run the main method of `KindOfQueryExample.java` in the `org.ihtsdo.otf.query.rest.client.examples package`.

Below is an example output that is generated by a successful run (xml formatting was added after the fact to make the results display better in this document).

Example 3: KindOfQueryExample output

```
            <componentNid>-2135975835</componentNid>
            <primordialComponentUuid>7ddc3edc-d49d-3f0f-b5ab-81efb44e5e62</primordi
         </conceptAttributes>
         <conceptReference>
            <definitionalState>UNDETERMINED</definitionalState>
            <nid>-2135975835</nid>
            <text>Extrinsic asthma with asthma attack (disorder)</text>
            <uuid>7ddc3edc-d49d-3f0f-b5ab-81efb44e5e62</uuid>
         </conceptReference>
         <primordialUuid>7ddc3edc-d49d-3f0f-b5ab-81efb44e5e62</primordialUuid>
         <viewCoordinateUuid>7dcb2c4a-fdb7-4b1b-bd50-a1e981837326</viewCoordinateU
         <refexPolicy>REFEX_MEMBERS_AND_REFSET_MEMBERS</refexPolicy>
         <relationshipPolicy>DESTINATION_RELATIONSHIPS</relationshipPolicy>
         <versionPolicy>ACTIVE_VERSIONS</versionPolicy>
      </theResults>
      <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
         <descriptionList />
         <destinationRelationshipList />
         <mediaList />
         <refsetMemberList />
         <conceptAttributes>
            <additionalIdList />
            <annotationList />
            <versionList />
            <componentNid>-2135452099</componentNid>
            <primordialComponentUuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</primordi
         </conceptAttributes>
         <conceptReference>
            <definitionalState>UNDETERMINED</definitionalState>
            <nid>-2135452099</nid>
            <text>Saw dust asthma</text>
            <uuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</uuid>
         </conceptReference>
         <primordialUuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</primordialUuid>
         <viewCoordinateUuid>7dcb2c4a-fdb7-4b1b-bd50-a1e981837326</viewCoordinateU
         <refexPolicy>REFEX_MEMBERS_AND_REFSET_MEMBERS</refexPolicy>
         <relationshipPolicy>DESTINATION_RELATIONSHIPS</relationshipPolicy>
         <versionPolicy>ACTIVE_VERSIONS</versionPolicy>
      </theResults>
   </ns4:result-list>
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma with status asthmaticus (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Atopic asthma
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma with status asthmaticus (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Non-IgE mediated allergic asthma
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma without status asthmaticus (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma with asthma attack (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Allergic-infective asthma (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma without status asthmaticus (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma with asthma attack (disorder)
Oct 16, 2013 11:38:10 AM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Saw dust asthma
```

Query Implementation

This query implementation module implements queries against the Terminology Component Chronicle service, which can be forked here [https://github.com/IHTSDO/OTF-Versioning-Service]. The pure Java implementation of query capabilities is demonstrated in the integration tests and may be called independently of the REST service. The query implementation is called by the REST service to perform queries.

Query Integration Tests

This module performs integrations tests against a SNOMED CT database. These tests demonstrate examples of how queries can be constructed within Java.

Query Service

This module implements a Tomcat 7.0 [http://tomcat.apache.org] REST service for querying. Once you have deployed the Query Service to Tomcat, you can utilize the following resources to create queries using the REST service.

Lucene Resource

Once the client has been deployed, you can conduct a Lucene search at `{host}/otf/query-service/lucene/{desired Lucene query text}`. If the desired Lucene query text includes spaces or other characters not permitted in URLs, encode the text here [http://www.url-encode-decode.com] by entering the text, selecting UTF-8 from the drop-down menu, and clicking the "Url encode" button.

Once you have deployed the war file, search for descriptions matching the text "extrasystole" by entering the URL `localhost:8080/otf/query-service/lucene/extrasystole` . The search outputs the following results, which were formatted:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns4:result-list xmlns:ns4="http://display.object.jaxb.otf.ihtsdo.org" xmlns:ns
   <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
      <authorReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483619</nid>
         <text>;user</text>
         <uuid>f7495b58-6630-3499-a44e-2052b5fcf06c</uuid>
      </authorReference>
      <fxTime>
         <time>1217487600000</time>
      </fxTime>
      <moduleReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483519</nid>
         <text>SNOMED CT core</text>
         <uuid>1b4f1ba5-b725-390f-8c3b-33ec7096bdca</uuid>
      </moduleReference>
      <pathReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483549</nid>
         <text>SNOMED Core</text>
         <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
      </pathReference>
      <statusString>ACTIVE</statusString>
      <viewCoordinateUuid>0c734870-836a-11e2-9e96-0800200c9a66</viewCoordinateU
      <componentNid>-2142320111</componentNid>
```

```
      <primordialComponentUuid>7891de6b-8dd5-3e07-b20f-555ebd869dec</primordialC
      <typeReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483625</nid>
         <text>Fully specified name</text>
         <uuid>00791270-77c9-32b6-b34f-d932569bd2bf</uuid>
      </typeReference>
      <initialCaseSignificant>true</initialCaseSignificant>
      <language>en</language>
      <text>ECG: extrasystole (finding)</text>
   </theResults>
   <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="n
      <authorReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483619</nid>
         <text>user</text>
         <uuid>f7495b58-6630-3499-a44e-2052b5fcf06c</uuid>
      </authorReference>
      <fxTime>
         <time>1012464000000</time>
      </fxTime>
      <moduleReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483519</nid>
         <text>SNOMED CT core</text>
         <uuid>1b4f1ba5-b725-390f-8c3b-33ec7096bdca</uuid>
      </moduleReference>
      <pathReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483549</nid>
         <text>SNOMED Core</text>
         <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
      </pathReference>
      <statusString>ACTIVE</statusString>
      <viewCoordinateUuid>0c734870-836a-11e2-9e96-0800200c9a66</viewCoordinateUu
      <componentNid>-2142320119</componentNid>
      <primordialComponentUuid>b4836b5d-da3e-3dbf-b937-48b414b43950</primordialC
      <typeReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483620</nid>
         <text>Synonym</text>
         <uuid>8bfba944-3965-3946-9bcb-1e80a5da63a2</uuid>
      </typeReference>
      <initialCaseSignificant>true</initialCaseSignificant>
      <language>en</language>
      <text>ECG: extrasystole</text>
   </theResults>
   <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="n
      <authorReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483619</nid>
         <text>user</text>
         <uuid>f7495b58-6630-3499-a44e-2052b5fcf06c</uuid>
      </authorReference>
      <fxTime>
         <time>1154329200000</time>
      </fxTime>
      <moduleReference>
```

```
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483519</nid>
            <text>SNOMED CT core</text>
            <uuid>1b4f1ba5-b725-390f-8c3b-33ec7096bdca</uuid>
         </moduleReference>
         <pathReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483549</nid>
            <text>SNOMED Core</text>
            <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
         </pathReference>
         <statusString>ACTIVE</statusString>
         <viewCoordinateUuid>0c734870-836a-11e2-9e96-0800200c9a66</viewCoordinateU
         <componentNid>-2142319952</componentNid>
         <primordialComponentUuid>3b413fc4-6bd8-38be-a69b-8be034e1595e</primordial
         <typeReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483620</nid>
            <text>Synonym</text>
            <uuid>8bfba944-3965-3946-9bcb-1e80a5da63a2</uuid>
         </typeReference>
         <initialCaseSignificant>false</initialCaseSignificant>
         <language>en</language>
         <text>Electrocardiogram: extrasystole</text>
      </theResults>
      <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
         <authorReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483619</nid>
            <text>user</text>
            <uuid>f7495b58-6630-3499-a44e-2052b5fcf06c</uuid>
         </authorReference>
         <fxTime>
            <time>1154329200000</time>
         </fxTime>
         <moduleReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483519</nid>
            <text>SNOMED CT core</text>
            <uuid>1b4f1ba5-b725-390f-8c3b-33ec7096bdca</uuid>
         </moduleReference>
         <pathReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483549</nid>
            <text>SNOMED Core</text>
            <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
         </pathReference>
         <statusString>ACTIVE</statusString>
         <viewCoordinateUuid>0c734870-836a-11e2-9e96-0800200c9a66</viewCoordinateU
         <componentNid>-2142319960</componentNid>
         <primordialComponentUuid>d2c81732-2561-37a1-ba39-67e3cc2b986f</primordial
         <typeReference>
            <definitionalState>NECESSARY</definitionalState>
            <nid>-2147483625</nid>
            <text>Fully specified name</text>
            <uuid>00791270-77c9-32b6-b34f-d932569bd2bf</uuid>
         </typeReference>
         <initialCaseSignificant>false</initialCaseSignificant>
```

```
      <language>en</language>
      <text>Electrocardiogram: extrasystole (finding)</text>
   </theResults>
   <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
      <authorReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483619</nid>
         <text>user</text>
         <uuid>f7495b58-6630-3499-a44e-2052b5fcf06c</uuid>
      </authorReference>
      <fxTime>
         <time>1012464000000</time>
      </fxTime>
      <moduleReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483519</nid>
         <text>SNOMED CT core</text>
         <uuid>1b4f1ba5-b725-390f-8c3b-33ec7096bdca</uuid>
      </moduleReference>
      <pathReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483549</nid>
         <text>SNOMED Core</text>
         <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
      </pathReference>
      <statusString>ACTIVE</statusString>
      <viewCoordinateUuid>0c734870-836a-11e2-9e96-0800200c9a66</viewCoordinateU
      <componentNid>-2142319964</componentNid>
      <primordialComponentUuid>c58822fc-ebe3-3c70-9d4a-9d5b3a18dbe9</primordial
      <typeReference>
         <definitionalState>NECESSARY</definitionalState>
         <nid>-2147483620</nid>
         <text>Synonym</text>
         <uuid>8bfba944-3965-3946-9bcb-1e80a5da63a2</uuid>
      </typeReference>
      <initialCaseSignificant>true</initialCaseSignificant>
      <language>en</language>
      <text>ECG: extrasystole</text>
   </theResults>
</ns4:result-list>
```

If this URL doesn't display the results listed above, please see the Troubleshooting Tomcat section.

Query Resource

Once the `.war` has been deployed, you can utilize the Query Resource to develop queries at `{host}/otf/query-service/query?`. Queries are constructed by stringing together the VIEWPOINT, FOR, LET, WHERE, and RETURN objects encoded in UTF-8 in the following manner: `{host}/otf/query-service/query?VIEWPOINT=<viewpoint xml>&FOR=<for xml>&LET=<let xml>&WHERE=<where xml>&RETURN=<return xml>`. Here is a link [http://www.url-encode-decode.com/urlencode] to an online URL encoder. Note that the objects need to be encoded individually and strung together with & symbols. The LET and WHERE parameters are required in order to conduct a query, while the VIEWPOINT, FOR, and RETURN parameters are optional. These optional parameters can be set to the default options using one of the following: `{host}/otf/query-service/query?VIEWPOINT=&FOR=&LET=<let xml>&WHERE=<where xml>&RETURN=` or `{host}/otf/query-service/query?VIEWPOINT=null&FOR=null&LET=<let xml>&WHERE=<where xml>&RETURN=null`

. This link [http://localhost:8080/otf/query-service/query?VIEWPOINT=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22+standalone%3D%22yes%22%3F%3E%3Cns2%3Asimple-view-coordinate+xmlns%3Ans2%3D%22http%3A%2F%2Fapi.chronicle.jaxb.otf.ihtsdo.org%22%3E%3CallowedStatus%3EACTIVE%3C%2FallowedStatus%3E%3CclassifierSpecification%3E%3Cdescription%3EIHTSDO+Classifier%3C%2Fdescription%3E%3Cuuid%3E7e87cc5b-e85f-3860-99eb-7a44f2b9e6f9%3C%2Fuuid%3E%3C%2FclassifierSpecification%3E%3CcontradictionPolicy%3ELAST_COMMIT_WINS%3C%2FcontradictionPolicy%3E%3ClangSort%3ERF2_LANG_REFEX%3C%2FlangSort%3E%3ClanguagePreferenceOrderList%3E%3Cdescription%3EUnited+States+of+America+English+language+reference+set+%28foundation+metadata+concept%29%3C%2Fdescription%3E%3Cuuid%3Ebca0a686-3516-3daf-8fcf-fe396d13cfad%3C%2Fuuid%3E%3C%2FlanguagePreferenceOrderList%3E%3ClanguageSpecification%3E%3Cdescription%3EUnited+States+of+America+English+language+reference+set+%28foundation+metadata+concept%29%3C%2Fdescription%3E%3Cuuid%3Ebca0a686-3516-3daf-8fcf-fe396d13cfad%3C%2Fuuid%3E%3C%2FlanguageSpecification%3E%3Cname%3ESnomed+Inferred+Latest%3C%2Fname%3E%3Cprecedence%3EPATH%3C%2Fprecedence%3E%3CrelAssertionType%3EINFERRED%3C%2FrelAssertionType%3E%3CviewPosition%3E%3Cpath%3E%3Corigins%3E%3Cpath%3E%3CpathConceptSpecification%3E%3Cdescription%3EWorkbench+Auxiliary%3C%2Fdescription%3E%3Cuuid%3E2faa9260-8fb2-11db-b606-0800200c9a66%3C%2Fuuid%3E%3C%2FpathConceptSpecification%3E%3C%2Fpath%3E%3CtimePoint%3E9223372036854775807%3C%2FtimePoint%3E%3C%2Forigins%3E%3CpathConceptSpecification%3E%3Cdescription%3ESNOMED+Core%3C%2Fdescription%3E%3Cuuid%3E8c230474-9f11-30ce-9cad-185a96fd03a2%3C%2Fuuid%3E%3C%2FpathConceptSpecification%3E%3C%2Fpath%3E%3CtimePoint%3E9223372036854775807%3C%2FtimePoint%3E%3C%2FviewPosition%3E%3C%2Fns2%3Asimple-view-coordinate%3E&FOR=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22+standalone%3D%22yes%22%3F%3E%3Cns2%3AforCollection+xmlns%3Ans2%3D%22http%3A%2F%2Fquery.jaxb.otf.ihtsdo.org%22%3E%3CforCollectionString%3ECONCEPT%3C%2FforCollectionString%3E%3C%2Fns2%3AforCollection%3E&LET=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22+standalone%3D%22yes%22%3F%3E%3Cns2%3AletMap+xmlns%3Ans2%3D%22http%3A%2F%2Fquery.jaxb.otf.ihtsdo.org%22%3E%3Cmap%3E%3Centry%3E%3Ckey%3Eallergic-asthma%3C%2Fkey%3E%3Cvalue+xsi%3Atype%3D%22ns4%3AsimpleConceptSpecification%22+xmlns%3Ans4%3D%22http%3A%2F%2Fapi.chronicle.jaxb.otf.ihtsdo.org%22+xmlns%3Axsi%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema-instance%22%3E%3Cdescription%3EAllergic+asthma%3C%2Fdescription%3E%3Cuuid%3E531abe20-8324-3db9-9104-8bcdbf251ac7%3C%2Fuuid%3E%3C%2Fvalue%3E%3C%2Fentry%3E%3Centry%3E%3Ckey%3EIs+a%3C%2Fkey%3E%3Cvalue+xsi%3Atype%3D%22ns4%3AsimpleConceptSpecification%22+xmlns%3Ans4%3D%22http%3A%2F%2Fapi.chronicle.jaxb.otf.ihtsdo.org%22+xmlns%3Axsi%3D%22http%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema-instance%22%3E%3Cdescription%3EIs+a+%28attribute%29%3C%2Fdescription%3E%3Cuuid%3Ec93a30b9-ba77-3adb-a9b8-4589c9f8fb25%3C%2Fuuid%3E%3C%2Fvalue%3E%3C%2Fentry%3E%3C%2Fmap%3E%3C%2Fns2%3AletMap%3E&WHERE=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22+standalone%3D%22yes%22%3F%3E%3Cns2%3Awhere+xmlns%3Ans2%3D%22http%3A%2F%2Fquery.jaxb.otf.ihtsdo.org%22%3E%3CrootClause%3E%3CletKeys%3EIs+a%3C%2FletKeys%3E%3CletKeys%3Eallergic-asthma%3C%2FletKeys%3E%3CsemanticString%3EREL_TYPE%3C%2FsemanticString%3E%3C%2FrootClause%3E%3C%2Fns2%3Awhere%3E&RETURN=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22+standalone%3D%22yes%22%3F%3E%3Cns2%3AreturnTypes+xmlns%3Ans2%3D%22http%3A%2F%2Fquery.jaxb.otf.ihtsdo.org%22%3ECONCEPT_VERSION%3C%2Fns2%3AreturnTypes%3E] provides an example of a query using the Query Resource. The construction of the LET and WHERE objects for this query is shown in Java class org.ihtsdo.otf.query.rest.client.examples.SimpleQueryExample. Here are the formatted results:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns4:result-list xmlns:ns4="http://display.object.jaxb.otf.ihtsdo.org" xmlns:ns
   <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
```

```
            <descriptionList />
            <destinationRelationshipList />
            <mediaList />
            <refsetMemberList />
            <conceptAttributes>
               <additionalIdList />
               <annotationList />
               <versionList />
               <componentNid>-2145791857</componentNid>
               <primordialComponentUuid>904794b3-1536-382f-9323-f7eb3c7e343f</primordi
            </conceptAttributes>
            <conceptReference>
               <definitionalState>UNDETERMINED</definitionalState>
               <nid>-2145791857</nid>
               <text>Extrinsic asthma</text>
               <uuid>904794b3-1536-382f-9323-f7eb3c7e343f</uuid>
            </conceptReference>
            <primordialUuid>904794b3-1536-382f-9323-f7eb3c7e343f</primordialUuid>
            <viewCoordinateUuid>e7113695-0e84-431b-bcb4-e6e210626105</viewCoordinateU
            <refexPolicy>REFEX_MEMBERS_AND_REFSET_MEMBERS</refexPolicy>
            <relationshipPolicy>DESTINATION_RELATIONSHIPS</relationshipPolicy>
            <versionPolicy>ACTIVE_VERSIONS</versionPolicy>
         </theResults>
         <theResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="
            <descriptionList />
            <destinationRelationshipList />
            <mediaList />
            <refsetMemberList />
            <conceptAttributes>
               <additionalIdList />
               <annotationList />
               <versionList />
               <componentNid>-2145526668</componentNid>
               <primordialComponentUuid>b714e0a9-33f0-30ad-a795-8fb9437b4af6</primordi
            </conceptAttributes>
            <conceptReference>
               <definitionalState>UNDETERMINED</definitionalState>
               <nid>-2145526668</nid>
               <text>Non-IgE mediated allergic asthma</text>
               <uuid>b714e0a9-33f0-30ad-a795-8fb9437b4af6</uuid>
            </conceptReference>
            <primordialUuid>b714e0a9-33f0-30ad-a795-8fb9437b4af6</primordialUuid>
            <viewCoordinateUuid>e7113695-0e84-431b-bcb4-e6e210626105</viewCoordinateU
            <refexPolicy>REFEX_MEMBERS_AND_REFSET_MEMBERS</refexPolicy>
            <relationshipPolicy>DESTINATION_RELATIONSHIPS</relationshipPolicy>
            <versionPolicy>ACTIVE_VERSIONS</versionPolicy>
         </theResults>
      </ns4:result-list>
```

JAXB Objects

This module generates Java data display objects derived from running the JAXB xjc against the the underlying implementation. This XML Schema Document is then compiled using JAXB schemagen to generate Java files. The .xsd document may be similarly useful for generating JSON for JavaScript developers. Future versions of the project will provide more complete examples using the JAXB generated data display objects to reduce client dependencies on other libraries.

Documentation

This project uses Docbook 5 [http://www.docbook.org] for generating documentation. Docbook generation is integrated with Maven using Docbkx Tools [http://code.google.com/p/docbkx-tools/], and follows in the footsteps of other development projects such as Sonatype's Maven book [http://blog.sonatype.com/people/2008/04/writing-a-book-with-maven-part-i/] and JBoss's community documentation [https://www.jboss.org/pressgang/jdg]. More details on how to contribute documentation will come in subsequent versions of this documentation.

Troubleshooting Tomcat

You may receive an ENV_LOCKED error when trying to start the Tomcat server. This occurs if you try to redeploy the `.war` when the database has not yet been initialized. The best fix for this is to undeploy the "otf" application, shutdown Tomcat, ensure that `Released shutdown permit` prints to the console, and restart Tomcat. If restarting Tomcat doesn't solve the issue, the ENV_LOCKED error may be resolved by restarting your computer.

Tomcat Shutdown

When you shutdown your Tomcat server, you should see the following output in the Tomcat Catalina log file:

```
Interrupting: Nid data service 4
Interrupting: Nid data service 1
Interrupting: Nid data service 3
Interrupting: Nid data service 2
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Database sync to disk...
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Database sync complete.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down dbWriterService.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
Interrupting: Nid data service 6
Interrupting: Nid data service 5
Interrupting: Nid data service 4
Interrupting: Nid data service 4
INFO: Awaiting termination of dbWriterService.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down changeSetWriterService.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Awaiting termination of changeSetWriterService.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: BdbCommitManager is shutdown.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down NidDataFromBdb executor pool.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Awaiting termination of NidDataFromBdb executor pool.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Termination NidDataFromBdb executor pool.
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts warnin
WARNING: org.apache.lucene.store.AlreadyClosedException: this IndexWriter is cl
Oct 28, 2013 11:50:15 AM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: bdb close finished.
Oct 28, 2013 11:50:16 AM org.apache.catalina.startup.HostConfig deleteRedeployR
INFO: Undeploying context [/otf]
```

And you should see the following in the Tomcat localhost log file:

```
Oct 28, 2013 11:50:15 AM org.apache.catalina.core.ApplicationContext log
INFO: Destroy ChronicleServletContainer
Oct 28, 2013 11:50:15 AM org.apache.catalina.core.ApplicationContext log
INFO: Aquired storeSemaphore for destroy.
Oct 28, 2013 11:50:15 AM org.apache.catalina.core.ApplicationContext log
INFO: Released storeSemaphore for destroy.
```