Query Service

IHTSDO

Copenhagen K
Gammeltorv 4, 1.
1457
Denmark

Table of Contents

Maven setup

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 (Eclipse) 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 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 a JDK 1.7 or higher.

Furthermore, to debug queries, you will need to allocate more space for Java. To set these properties in the NetBeans IDE, add "-J-Xss2m -J-Xms1g -J-Xmx2g -J-XX:PermSize=1600m" to the netbeans_default_options in the netbeans.conf file.

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 artifacts in the IHTSDO's repository, which can be requested from Rory Davidson. Below is a copy of a settings.xml that allows a user to conduct the default build. In order to perform other build profiles, 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 setup

The Query service project requires Java 1.7 or higher.

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 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 manages the sources and documents for the project. Understanding Maven is a prerequisite to understanding how to build and work with the query service. More information on Maven is available at the Maven [http://maven.apache.org] web 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.

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

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 specifically 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.

These artifacts 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 required 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 when the build profile id *integration-tests* is activated. 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`.

Deploy to app server

The default build command generates a `.war` file that can be deployed to an app server. Before deployment the app server must be properly configured with adequate memory, and access to the Berkeley database folder. A .zip file of the Berkeley database folder can be accessed from the file releases section [https://csfe.aceworkspace.net/sf/frs/do/listReleases/projects.the_ihtsdo_terminology_open_tool/frs.test_data] of the Open Tooling Framework [https://csfe.aceworkspace.net/sf/projects/the_ihtsdo_terminology_open_tool/] website.

When the rest server starts (currently at first query), it opens the berkeleydb located in the folder berkeley-db, wherever that relative path is on your server. On my server, it is at:

/Users/kec/GlassFish_Server/glassfish/domains/ttk/config/berkeley-db

(the config directory of the domain appears to be the working directory) You can override the location by setting a system property on the server:

```
-Dorg.ihtsdo.otf.tcc.datastore.bdb-location=<the location of the berkeley-db fo
```

It writes diagnostic output when opening the database as follows:

```
INFO: QS_sprint2:_Query_rest_service was successfully deployed in 3,637 millise
INFO: org.ihtsdo.otf.tcc.datastore.bdb-location not set. Using default location
INFO: setup dbRoot: berkeley-db
INFO: absolute dbRoot: /Users/kec/GlassFish_Server/glassfish/domains/ttk/config/
INFO: NidCidMap readOnlyRecords: 812
INFO: NidCidMap mutableRecords: 0
```

And takes a few minutes to open. Inside the test data is an old format view coordinate... So it throws a serialization exception:

```
SEVERE: java.io.StreamCorruptedException: unexpected block data at
        java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1362) at
        java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:1989
        java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1913) a
        java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:179
```

Just ignore that… It will self heal.

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.

There are two example programs: `HelloExample` and `KindOfQueryExample`.

Setup

Apache Tomcat 7.0 can be downloaded here [http://tomcat.apache.org/download-70.cgi]. Configure Tomcat with the following startup parameters: `-Xmx6g -Xms6g -Dorg.ihtsdo.otf.tcc.datastore.bdb-location=/usr/NetBeansProjects/berkeley-db`. Replace the location "/usr/NetBeansProjects/berkeley-db" with the local directory of the berkeley database on your machine.

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. Include the startup parameters stated above in the VM Options.

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.

This program is simply to test connectivity and server status. A simple request of `http://api.snomedtools.com/query-service/hello/frank` should return the following to the console:

```
200

hello frank.
```

The java file contains comments on each line to show the setup of the client, sending the request, and processing the result. The program has a main() and therefore can be invoked from within most IDE's by right clicking on the file, and selecting a run or debug option. Please review the java file for more details.

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.

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 like 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 defined the where clauses for the query
- a RETURN that defines that type of components to return (Concepts, descriptions, etc).

This example's main method will setup a kind-of query that will return concepts that are kind-of SNOMED "allergic asthma" concepts. 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 with the following 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
```

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).

```
                        <time>1012464000000</time>
                    </fxTime>
                    <moduleReference>
                        <nid>-2147483534</nid>
                        <text>SNOMED Core</text>
                        <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
                    </moduleReference>
                    <pathReference>
                        <nid>-2147483534</nid>
                        <text>SNOMED Core</text>
                        <uuid>8c230474-9f11-30ce-9cad-185a96fd03a2</uuid>
                    </pathReference>
                    <statusString>ACTIVE</statusString>
                    <viewCoordinateUuid>bd58d1df-b3be-4c6a-9c01-89c7561fafe8</v
                    <authorityRef>
                        <nid>-2147483476</nid>
                        <text>SNOMED RT Id</text>
                        <uuid>740f47ef-caf3-3e4e-bdeb-39792b408f1c</uuid>
                    </authorityRef>
                    <denotation xsi:type="xs:string" xmlns:xs="http://www.w3.org
                        >D2-54262</denotation>
                </additionalId>
            </additionalIdList>
            <annotationList/>
            <versionList/>
            <componentNid>-2140298245</componentNid>
            <primordialComponentUuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</prim
        </conceptAttributes>
        <conceptReference>
            <nid>-2140298245</nid>
            <text>Saw dust asthma</text>
            <uuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</uuid>
        </conceptReference>
        <primordialUuid>7aa5d3d5-97a5-349e-b2f0-9a1638e8a176</primordialUuid>
        <viewCoordinateUuid>bd58d1df-b3be-4c6a-9c01-89c7561fafe8</viewCoordinat
        <refexPolicy>REFEX_MEMBERS_AND_REFSET_MEMBERS</refexPolicy>
        <relationshipPolicy>DESTINATION_RELATIONSHIPS</relationshipPolicy>
        <versionPolicy>ACTIVE_VERSIONS</versionPolicy>
    </theResults>
</ns4:result-list>
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma with status asthmaticus (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Atopic asthma
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma with status asthmaticus (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Non-IgE mediated allergic asthma
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma without status asthmaticus (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma with asthma attack (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Allergic-infective asthma (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Intrinsic asthma without status asthmaticus (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Extrinsic asthma with asthma attack (disorder)
Aug 28, 2013 10:11:24 PM org.ihtsdo.otf.query.rest.client.examples.KindOfQueryEx
INFO: Returned concept: Saw dust asthma
```


Example 2. `KindOfQueryExample` output

Query Implementation

This query implementation module implements queries against Terminology Component Chronicle service, and proved a pure java implementation of query capabilities that may be called independently of the REST service, and is also called by the REST service to perform queries.

Query Integration Tests

This module performs tests against a SNOMED CT database.

Query Service

This module implements a Tomcat 7.0 [http://tomcat.apache.org] REST service for querying.

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 enabled distributed editing with configuration mangament, and multiple distribution modalities. 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.