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 preqrequisite 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 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.

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 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

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 Tomcat with the following startup with the following JVM options: `-Xmx6g -Xms6g -XX:PermSize=256m -XX:MaxPermSize=512m -Dorg.ihtsdo.otf.tcc.datastore.bdb-location=<location of the berkeley-db folder>`. This is how these parameters are set based upon your operating system:

1. Windows - use the command `--JvmOptions` to set the above options. For more information on command-line parameters for Tomcat 7 on a Windows machine, please see this link [http://tomcat.apache.org/tomcat-7.0-doc/windows-service-howto.html#Command_line_parameters].

2. Linux/ Unix - add the given parameters as `JAVA_OPTS` in the `catalina.sh` file, which is located in the `/bin` folder within Tomcat.

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

Here is a sample `tomcat-users.xml` file that will setup a user "admin" with a password "password" and privileges to access the Tomcat Web Application Manager:

```
<?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>
```

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.

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. 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. Before deployment the app server must be properly configured with adequate memory and access to the Berkeley database folder. The 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.

Please see the Tomcat Setup section for instructions on how to configure your Tomcat server. Here are the instructions for deploying the .war file to your Tomcat server based upon how you're launching Tomcat:

1. From the command line: save the following xml file in CATALINA_HOME/conf/Catalina/localhost as otf.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context
antiJARLocking="true" docBase="/Users/dylangrald/tcc-sprint2/OTF-Query-Servic
path="/otf"/>
```

   execute the `startup` bat or 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 CATALINA_HOME 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`

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

You should see the following in the Apache Tomcat log when the data

```
Sep 20, 2013 2:58:19 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performanc
Sep 20, 2013 2:58:19 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Sep 20, 2013 2:58:19 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Sep 20, 2013 2:58:19 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 473 ms
Sep 20, 2013 2:58:19 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Sep 20, 2013 2:58:19 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.42
Sep 20, 2013 2:58:19 PM org.apache.catalina.startup.HostConfig deployDescriptor
INFO: Deploying configuration descriptor /Library/Tomcat/apache-tomcat-7.0.42/c
Sep 20, 2013 2:58:22 PM org.apache.catalina.util.SessionIdGenerator createSecur
INFO: Creation of SecureRandom instance for session ID generation using [SHA1PR
Sep 20, 2013 2:58:22 PM org.ihtsdo.otf.tcc.datastore.BdbTerminologyStore <init>
INFO: org.ihtsdo.otf.tcc.datastore.bdb-location set. Starting from location: /U
Sep 20, 2013 2:58:22 PM org.glassfish.jersey.server.ApplicationHandler initiali
INFO: Initiating Jersey application, version Jersey: 2.2 2013-08-14 08:51:58...
Sep 20, 2013 2:58:22 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: NidCidMap readOnlyRecords: 812
Sep 20, 2013 2:58:22 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: NidCidMap mutableRecords: 0
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /Library/Tomcat/apache-tomcat-7.0.42/
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /Library/Tomcat/apache-tomcat-7.0.42/
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /Library/Tomcat/apache-tomcat-7.0.42/
```

```
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /Library/Tomcat/apache-tomcat-7.0.42/w
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /Library/Tomcat/apache-tomcat-7.0.42/w
Sep 20, 2013 2:58:23 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Sep 20, 2013 2:58:23 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Sep 20, 2013 2:58:23 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3860 ms
```

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 takes a couple minutes to open the database, then you will see the following output in the log file:

```
Sep 20, 2013 2:59:32 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: mutable maxMem: 224000
Sep 20, 2013 2:59:32 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: mutable shared cache: true
Sep 20, 2013 2:59:32 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: readOnly maxMem: 224000
Sep 20, 2013 2:59:32 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: readOnly shared cache: true
Sep 20, 2013 2:59:32 PM 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. 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.

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

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 2. `KindOfQueryExample` output

```
                        <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
```

Lucene Resource

Once the client has been deployed, you can conduct a Lucene serach at `DEFAULT_HOST/otf/query-service/lucene/{desired Lucene query text}`. If the desired Lucene query text includes spaces or other characters not permitted in URLs, you can 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.

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 show 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.

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 shutdown Tomcat, ensure that `Released shutdown permit` prints to the console, and restart Tomcat. If restarting Tomcat doesn't solve the issue, the issue may be resolved by restarting your computer.

Tomcat Shutdown

When you shutdown your Tomcat server, you should see the following output:

```
Sep 20, 2013 4:27:29 PM org.apache.catalina.core.StandardServer await
INFO: A valid shutdown command was received via the shutdown port. Stopping the
```

```
Sep 20, 2013 4:27:29 PM org.apache.coyote.AbstractProtocol pause
INFO: Pausing ProtocolHandler ["http-bio-8080"]
Sep 20, 2013 4:27:29 PM org.apache.coyote.AbstractProtocol pause
INFO: Pausing ProtocolHandler ["ajp-bio-8009"]
Sep 20, 2013 4:27:29 PM org.apache.catalina.core.StandardService stopInternal
INFO: Stopping service Catalina
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Database sync to disk...
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Database sync complete.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down dbWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Awaiting termination of dbWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down changeSetWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Awaiting termination of changeSetWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: BdbCommitManager is shutdown.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.model.cc.lucene.LuceneManager close
INFO: Shutting down luceneWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.model.cc.lucene.LuceneManager close
INFO: Awaiting termination of luceneWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.model.cc.lucene.LuceneManager closeR
INFO: Shutting down luceneWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.model.cc.lucene.LuceneManager closeR
INFO: Awaiting termination of luceneWriterService.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Shutting down NidDataFromBdb executor pool.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Awaiting termination of NidDataFromBdb executor pool.
Sep 20, 2013 4:27:29 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: Termination NidDataFromBdb executor pool.
Sep 20, 2013 4:27:30 PM org.ihtsdo.otf.tcc.datastore.temp.LogWithAlerts info
INFO: bdb close finished.
Sep 20, 2013 4:27:30 PM org.apache.coyote.AbstractProtocol stop
INFO: Stopping ProtocolHandler ["http-bio-8080"]
Sep 20, 2013 4:27:30 PM org.apache.coyote.AbstractProtocol stop
INFO: Stopping ProtocolHandler ["ajp-bio-8009"]
Sep 20, 2013 4:27:30 PM org.apache.coyote.AbstractProtocol destroy
INFO: Destroying ProtocolHandler ["http-bio-8080"]
Sep 20, 2013 4:27:30 PM org.apache.coyote.AbstractProtocol destroy
INFO: Destroying ProtocolHandler ["ajp-bio-8009"]
```