How To Use the OpenMRS API

Within the OpenMRS Webapp

Everything is delivered from the database in the form of objects: Patient, User, Concept, etc java objects.

You can fetch/save these objects using what we call "services" (PatientService, UserService, ConceptService).

You can get the services at any point by calling Context.getPatientService(), .getUserService(), etc.

To print out the names of all the patients in the system:

```
List<Patient> patients = Context.getPatientService().getAllPatients();
for (Patient patient : patients) {
  System.out.println("Patient: " + patient.getGivenName() + " " + patient.getFamilyName());
}
```

In an External Application using web services

See Web Services

In an External Application

We no longer test this use case so you may experience difficulties trying to use this approach. Please see Web Services as an alternative.

The openmrs-api-xxx.jar file can be used in stand-alone java applications. There are three simple steps you need to follow:

1. Choose an option below for including the openmrs jar(s) and dependencies
2. Fire up OpenMRS by calling: Context.startup(<connection.url>, <connection.username>, <connection.password>, getProperties());
3. You need to surround any units of work with Context.openSession() (which startup() calls for you) and Context.closeSession()
4. Authenticate into OpenMRS by calling: Context.authenticate(username, password);
5. You should be able to call into the Context and work with the services.

Example

```
public static void main(String[] args) {
  File propsFile = new File(OpenmrsUtil.getApplicationDataDirectory(), "openmrs-runtime.properties");**
  Properties props = new Properties();
  OpenmrsUtil.loadProperties(props, propsFile);
  Context.startup("jdbc:mysql://localhost:3306/db-name?autoReconnect=true", "openmrs-db-user", "3jknfjkn33ijt",
props);
  try {
    Context.openSession();
    Context.authenticate("admin", "test");
    List<Patient> patients = Context.getPatientService().getPatients("John");
    for (Patient patient : patients) {
      System.out.println("Found patient with name " + patient.getPersonName() + " and uuid: " + patient.
getUuid());
    }
    ...
  }
  finally {
    Context.closeSession();
  }
}
```

(the first 4 lines only have to be done once at startup. The rest are per request)

Using the API 1.8.0+ Jar (maven based)

OpenMRS introduced Maven in 1.8. The easiest way to use the OpenMRS API is to also use maven for your project. Once you do that, simply use these in your pom.xml

```
<dependency>
        <groupId>org.openmrs.api</groupId>
        <artifactId>openmrs-api</artifactId>
        <version>${openMRSVersion}</version>
        <type>jar</type>
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.openmrs.api</groupId>
        <artifactId>openmrs-api</artifactId>
        <version>${openMRSVersion}</version>
        <type>test-jar</type>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.openmrs.web</groupId>
        <artifactId>openmrs-web</artifactId>
        <version>${openMRSVersion}</version>
        <type>jar</type>
        <scope>provided</scope>
        </dependency>
<dependency>
        <groupId>org.openmrs.web</groupId>
        <artifactId>openmrs-web</artifactId>
        <version>${openMRSVersion}</version>
        <type>test-jar</type>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.openmrs.test</groupId>
        <artifactId>openmrs-test</artifactId>
        <version>${openMRSVersion}</version>
        <type>pom</type>
        <scope>test</scope>
</dependency>
<dependency>
        <groupId>org.openmrs.tools</groupId>
        <artifactId>openmrs-tools</artifactId>
        <version>${openMRSVersion}</version>
</dependency>
<properties>
        <openMRSVersion>1.8.1</openMRSVersion>
</properties>
```

Using the API Jar 1.7 and below

The openmrs-api-xxx.jar file can be used in stand-alone java applications. There are three simple steps you need to follow:

1. Download/build the openmrs api jar file "openmrs-api.****.jar" and include it on your classpath.
2. Include the dependent libraries on your classpath, which you can get from openmrs-trunk/api/pom.xml.

- See Overriding OpenMRS Default Runtime Properties for what data you can provide to openmrs

OpenMRS startup environment

In order to start OpenMRS with all it's modules (especially necessary if "mandatory" modules as with OpenMRS 1.6 are present), it is not sufficient to simply include the required jar files. Choose one of the following approaches to get your app started:

1. Start the external app with the same user, that Tomcat and the OpenMRS webapp are using.
2. If you are using a different user, specify the Global Property `module_repository_folder` with an absolute path to the modules directory of the OpenMRS webapp.
3. Set the working (start) dir of your external app to the OpenMRS base folder of the webapp. (This should usually be the place where the openmrs-runtime.properties is located.)
4. Overwrite the `application_data_directory` of the Overriding OpenMRS Default Runtime Properties

Note that scheduled tasks can/may/will fail during initialization.

Demarcating start and end of API work

- When coding against the database API, domain objects (Patients, Users, etc.) provided from the API are only guaranteed to be valid between calls to context.openSession() and context.closeSession().
    - You needn't worry about this within a web application environment as long as you are using a filter such as `org.openmrs.web.OpenmrsFilter` (since the filter marks the boundaries around each HTTP request)
    - You cannot use domain objects *across* transactions (or requests) (e.g., if you loaded a `Patient` object on one web page, you must reload that object, before using it on subsequent pages)
- It is the developer's responsibility to ensure that context.closeSession() is called (if not within webapp environment) to release precious resources (*even* in the event of an exception)
- These transaction boundary calls are *lightweight* -- i.e., there is little penalty for calling them