# OpenMRS SDK Step By Step Tutorials

This tutorial will introduce you to OpenMRS SDK (Software Development Kit). It assumes that you have basic understanding of OpenMRS modular architecture (you can learn about it by reading Technical Overview).

To get started ensure that you have installed the latest version of OpenMRS SDK and Oracle JDK 8. You can find instruction how to do that on OpenMRS SDK wiki pages.

In this tutorial we will walk step by step through common use cases and scenarios.



## Why use the SDK

The question is why do I need to use the SDK for development, instead of installing Tomcat and using my own configuration.

The SDK is the tool for setting multiple self contained servers that can be run independently, at times at the same time by running each available server on a different port specifying the -Dport variable. You can even mix the servers, having those running JDK 1.7 for 1.11.x and JDK 1.8 2.x alongside each other.

The SDK creates an indpendent Tomcat instance with its own database.

# Setting up dev environment

To set up your dev environment, you have to execute 4 steps:

1. Create server
2. Add developed project to watched projects on server
3. Create debugging configuration in IDE
4. Run server

We will walk through these steps and create environment to develop and debug REST module on top of OpenMRS 2.0.0 platform.

To create server, run:

```
mvn openmrs-sdk:setup -DserverId=webservices-dev -Dplatform=2.0.0
```

SDK will fetch artifacts, and prompt you for debug port:

> *If you want to enable remote debugging by default when running the server,*
> *specify the port number here (e.g. 1044). Leave blank to disable debugging.*
> *(Do not do this on a production server) (default: 'no debugging'):*

Type default value '1044' and push enter. Specifying debug port is essential to setup remote debugging configuration. You will be asked what database you want to use:

> *Which database would you like to use?:*
> *1) H2*
> *2) MySQL 5.7 (requires pre-installed MySQL 5.7)*
> *3) MySQL 5.7 in SDK docker container (requires pre-installed Docker)*
> *4) Existing docker container (requires pre-installed Docker)*

In this tutorial we will use most simple solution: H2, so answer with '1'. You can find more information about other options in OpenMRS SDK Setting up servers docs.

Last prompt will ask you for path to JDK:

> *Which JDK would you like to use to run this server?:*
> *1) JAVA_HOME (currently: /usr/lib/jvm/java-8-oracle/jre)*
> *2) /usr/lib/jvm/java-7-oracle/jre*
> *4) Other...*

If you have properly configured JAVA_HOME, answer '1'. If not, you need to answer with '4' and type path to JDK on your machine. Please note that OpenMRS platform 2.x requres JDK 8 to run.

Now, when server is created you need to get source of REST module. To do that, in your workspace directory run:

```
mvn openmrs-sdk:clone -DgroupId=org.openmrs.module -DartifactId=webservices.rest
```

SDK will ask you for your GitHub credentials, then fork repository and clone it. Enter newly created 'openmrs-module-webservices.rest' directory and add project to watched projects with

```
mvn openmrs-sdk:watch -DserverId=webservices-dev
```

From now on, every time you run 'webservices-dev' server, SDK will automatically execute steps:

1. Build webservices.rest module with 'mvn clean install -DskipTests'
2. Deploy webservices.rest module to webservices-dev server, equivalent of manually running:

   ```
   mvn openmrs-sdk:deploy -DserverId=webservices-dev
   ```

So you will always have module compiled from your sources on 'webservices-dev' server.

Now it's time to create debugging configuration in IDE. Recommended way is to create remote debugging configuration on adress 'localhost:1044' (1044 is debug port you specified during server creation). This way, you can run instance from command line:

```
mvn openmrs-sdk:run -DserverId=webservices-dev
```

Remote debugging has important advantage over running server from IDE - you can debug multiple projects with single server run (but only one at the same time), and run debugging only when you need to.

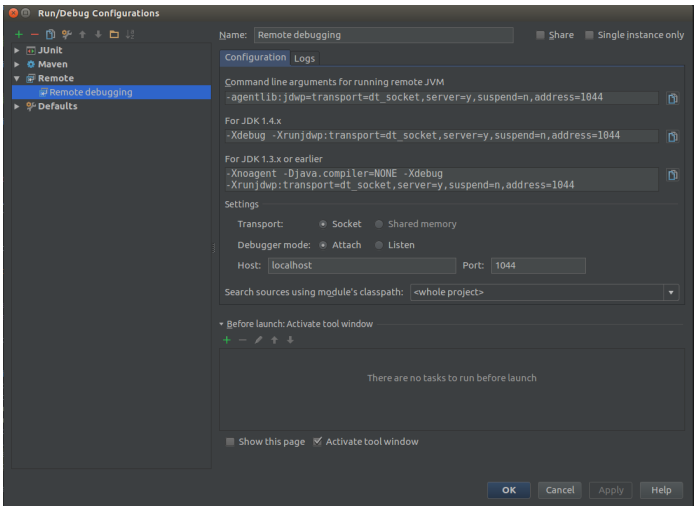
Alternatively, you can create direct Maven configuration in IDE to run:

```
mvn openmrs-sdk:run -DserverId=webservices-dev -Dfork=false
```

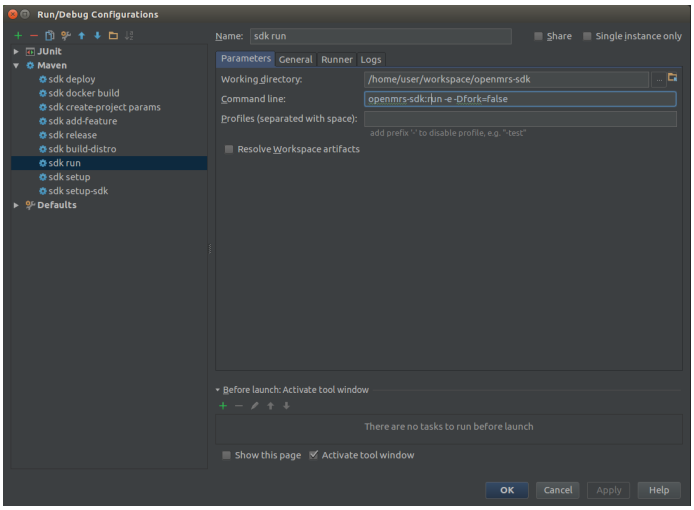
Steps to create debugging configuration depend on IDE you are using, but it should be quite easy to find.

Example screens of configurations for IDEA IntelliJ:

Remote debugging




Direct debugging




This way you can set up development environment for any module and openmrs-core. Remember that you have to provide all required modules by currently developed module.

## Developing with the Reference Application

You can easily develop/ debug modules with the Reference Application just like with the platform. The steps are as follows.

1. Create a distribution server.
2. Select Reference Application version.
3. Set debug port.
4. Connect with database.
5. Add a module as watched in the selected server.

As before, create a server by running

```
mvn openmrs-sdk:setup
```

This will start the SDK in the wizard mode. When prompted, give a name to the server you're creating.

Next, you will be prompted to select server type.

```
You can setup the following servers:
1) Distribution
2) Platform
Which one do you choose? [1/2]:
```

Make a Distribution server by entering 1. Next you'll be prompted to select the distribution version.

```
You can deploy the following versions of distribution:
1) Reference Application 2.6-SNAPSHOT
2) Reference Application 2.5
3) Reference Application 2.4
4) Reference Application 2.3.1
5) Reference Application 2.2
6) Other...
Which one do you choose? [1/2/3/4/5/6]:
```

Select the latest Reference Application version by entering 1. The rest of the steps are the same as explained under **Setting up dev environment**. The watched modules will be built from source and deployed to the selected server every time you run that server.

## Creating new distribution

OpenMRS distribution consist of platform and set of modules. SDK distribution is specified in properties file, by convention named **openmrs-distro.properties**. Minimal valid distro file contains 3 fields:

> *name= { name of distribution }*
> *version= { version of distribution }*
> *war.openmrs= { version of openmrs platform }*

This is sufficient to create new instance of platform with specified version, but probably you will want to have some modules as well.

Let's create basic distribution for development of Open Web Apps  for OpenMRS. To serve OWA we need  OWA module and REST module, because most of Web Apps want to consume OpenMRS REST API. To introduce module to distro configuration, we need to add new line in format: 'omod.{module id}={module version}'. Module id by convention matches module's Maven Artifact ID. For OWA module it is simply 'owa', for REST module it is 'webservices.rest'. We will use platform 2.0 version.  Let's name our distribution 'OWA development'.

To sum up, our openmrs-distro.properties file will look like this:

> *name= OWA development*
> *version= 1.0*
> *war.openmrs= 2.0.0*
> *omod.owa=1.6.2*
> *omod.webservices.rest=2.17-SNAPSHOT*

SDK automatically assumes that modules Maven Group ID is 'org.openmrs.module', but you can declare another Group ID

> *omod.event={ version of Event module }*
> *omod.event.groupId=org.openmrs*

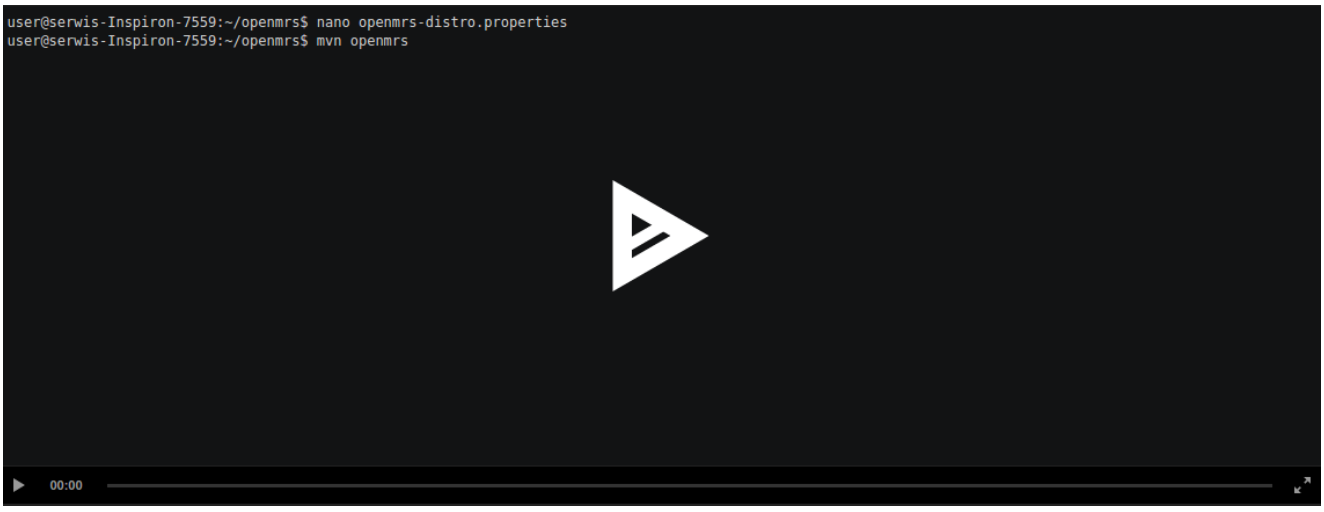
You can declare if your distribution supports H2 as database by adding line:

> *db.h2.supported=true*

Now, when we have distro configuration file, you can setup server from directory containing **openmrs-distro.properties** with command:

```
mvn openmrs-sdk:setup
```

You can see these steps executed on asciinema:




## Replicating environment for troubleshooting

With knowledge from previous tutorials, you can pretty easily create environment to reproduce application behaviour. Person who found error needs to provide basic information about server setup - version of platform, installed modules, database provider. Sometimes it is necessary too replicate state of database tables too. This tutorial will explain how effectively replicate environment with OpenMRS SDK.

To allow others to reproduce your server configuration, you have to create **openmrs-distro.properties** file with modules deployed on failing server. Since SDK 3.4.x you can go to your server directory `{user directory}/openmrs/{name of server}` and copy this file to share it with others. If you are using older version of SDK or standalone OpenMRS, you need to manually create this file as described in Creating new distribution.

### Creating MySQL dump file

Best way to replicate database state is to create SQL dump with mysqldump. Mysqldump will require database name, user name and password. You can run it like this:

```
mysqldump -u {username} -p{password} {name of database} > dump.sql
```

To create dump from dockerized MySQL database, you need to use mysqldump from within the container. We will show it on example of creating dump of MySQL database in SDK container. First, ensure that SDK container **openmrs-sdk-mysql-v3-2** is started:

```
docker start openmrs-sdk-mysql-v3-2
```

in next step, run mysqldump from within container:

```
docker exec -it openmrs-sdk-mysql-v3-2 mysqldump -u root -pAdmin123 {name of database, equal to name of server}
> dump.sql
```

Username **root** and password **Admin123** are standard credentials for MySQL instance in container created by SDK. Creating dumps from custom container looks analogously, you just have to replace container name and credentials.

After command execution, you will have **dump.sql** file in your working directory.

### Setting up server

Now, if you want to replicate environment of other developer, and you are provided **openmrs-distro.properties** and **dump.sql,** you can do that in one step:

```
mvn openmrs-sdk:setup -DdbSql=dump.sql -Drun
```

from directory where you have those 2 files. flag **-Drun** makes SDK automatically run server after successful setup. Go to http://localhost:8080/openmrs/ to complete the setup.

## Related articles

- OpenMRS SDK documentation