# Using the UI Framework in Your Module

The OpenMRS UI Framework is packaged in a module which provides the *framework*. To use it, you need to create a module of your own that depends on it and uses it.



## Depending on the UI Framework module

Its recommended that you use a mavenized module and that you generate it with the Module Maven Archetype.

To refer to the UI Framework in your module, you just need a standard module dependency on *uiframework* module:

1. In your module's root pom.xml you need something like this: (you want to get the latest released version number from http://mavenrepo.openmrs.org/nexus/index.html#nexus-search;quick~uiframework-api )

```
<dependencyManagement>
        <dependencies>
                <!-- Depends on uiframework module -->
                <dependency>
                        <groupId>org.openmrs.module</groupId>
                        <artifactId>uiframework-api</artifactId>
                        <version>3.2.2</version>
                        <type>jar</type>
                        <scope>provided</scope>
                </dependency>
                ...
        </dependencies>
</dependencyManagement>
```

2. In your module's omod/pom.xml you need something like this:

```
<dependency>
        <groupId>org.openmrs.module</groupId>
        <artifactId>uiframework-api</artifactId>
</dependency>
```

3. And your module's omod/src/main/resources/config.xml needs something like this:

```
<require_modules>
        <require_module>
                org.openmrs.module.uiframework
        </require_module>
</require_modules>
```

Of course any administrator who installs your module will also have install the UI Framework module, just like with any other module dependency.

## Beans in webModuleApplicationContext.xml

99% of of the time you'll want to have your module use the standard conventional configuration, by defining a single bean in webModuleApplicationContext.xml

```
<bean class="org.openmrs.ui.framework.StandardModuleUiConfiguration">
                <property name="moduleId" value="yourmoduleid"/>
       </bean>
```

The standard configuration looks for:

- Java page controllers in org.openmrs.module.yourmoduleid.page.controller
- Groovy page views in your omod's web/module/pages
- Java fragment controllers in org.openmrs.module.yourmoduleid.fragment.controller
- Groovy fragment views in your omod's web/module/fragments

### Folders and Packages

In the 99% case where you're using this standard configuration, you should create the folders:

- (yourmodule)/omod/src/main/webapp/pages
- (yourmodule)/omod/src/main/webapp/fragments
- (yourmodule)/omod/src/main/webapp/resources

And you should create your page and fragment controller packages like:

- org.openmrs.module.(yourmoduleid).page.controller at (yourmodule)/omod/src/main/java/org/openmrs/module/(yourmoduleid)/page/controller
- org.openmrs.module.(yourmoduleid).fragment.controller at (yourmodule)/omod/src/main/java/org/openmrs/module/(yourmoduleid)/fragment/controller

## Development mode

The UI Framework supports a "development mode" that helps you iterate rapidly on web-layer functionality, by automatically recompiling controllers as you edit them, and automatically reloading views. Note that development mode only automatically recompiles *controllers* (for pages or fragments), not other classes. But you should be iterating on your module's domain objects and service layer using the unit testing framework anyway. :-)

### OpenMRS SDK users

If your development environment is setup using the OpenMRS SDK, then this step is even easier for you. All you need to do to enable UI framework development mode is run

```
mvn openmrs-sdk:watch -DserverId=myserver
```

For more information on that see the Watch projects section of the SDK documentation.

### Manual configuration(Non SDK users)

If you are not currently using the SDK for development(which we strongly recommend that you do), then follow these instructions instead.

To enable development mode for a particular module, you need to set an argument at runtime before starting up OpenMRS.

The primary means for doing this has been to set a VM argument on the JRE you are running OpenMRS in. (See section on Setting a VM argument below)

If you set the following VM argument, it will be picked up by the view and controller providers configured by the StandardModuleUiConfiguration bean for yourmoduleid.

```
-DuiFramework.development.yourmoduleid="/path/to/root/of/mavenized/yourmoduleid"
```

As of version 3.3 of the uiframework module, there some alternative options available for configuring development mode:

1. You can now specify any of the runtime arguments via either VM arguments (as described above), or as properties in your OpenMRS runtime properties file
2. You can continue to specify each individual module that you wish to put into development mode by listing it and the path to the code for it explicitly as described above
3. For developers who want to more easily enable development mode for several modules that are checked out into a common location on the filesystem, you can specify

```
uiFramework.developmentFolder=/path/to/parent/folder/for/modules
uiFramework.developmentModules=comma,separated,module,ids,to,include
```

This will work only if you have your code checked out into a directory named either <moduleId> or openmrs-module-<moduleId>

As an example, lets say that I manage all of my OpenMRS code in a single directory: /home/openmrs/code

Then, within that directory I have various modules cloned that I work on:

- /home/openmrs/code/openmrs-module-coreapps
- /home/openmrs/code/openmrs-module-reportingui
- /home/openmrs/code/openmrs-module-htmlformentryui
- /home/openmrs/code/openmrs-module-appointmentschedulingui

If I set the following in my openmrs-runtime.properties file:

uiFramework.developmentFolder=/home/openmrs/code

Then starting up my system, if any modules with moduleId in coreapps, reportingui, htmlformentryui, or appointmentschedulingui are started, then development mode will be enabled for these.

However, if at the moment I am only working on coreapps and reportingui, and so I only want to enable development mode for these, then I can further limit this with the following in my openmrs-runtime.properties file:

uiFramework.developmentFolder=/home/openmrs/code

uiFramework.developmentModules=coreapps,reportingui

This will also work as if my code is checked out under simply the moduleId (i.e. /home/openmrs/code/coreapps)

### Setting a VM argument

- In Eclipse you can do this from the JRE tab of your jetty:run Run Configuration.  In Eclipse, you need to specify this as:  -DuiFramework...
- In Netbeans you can do this by adding it to the *jetty:run custom goal*. In detail, download/open the OpenMRS core source in Netbeans, then open the *openmrs-webapps* subproject. Right click on the webapps subproject, select *Custom/Goals*... This will bring up the *Run Maven* popup. Type *jetty:run* in the Goals and the VM argument line in the *Properties* text area of the goal by typing the following: *uiFramework.development.yourmoduleid=/path/to/root/of/mavenized/yourmoduleid*.

**Troubleshooting:** It is reported that in unix you may need to delete " from the path and escape whitespaces with \ if any.