

How to Code Modules

For this tutorial, we'll work together and create a simple FAIMS module. For illustrative purposes, we're going to recreate a simple module available from the FAIMS demo server—the "Oral History" module.

We're going to start by showing you how to make a basic but fully functional version of the module that can be made only by constructing a relevant module.xml. Then, after you've run that through the FAIMS-tools and produced a fully functioning but simple version, we'll teach you some ways to add complexity and functionality by modifying the necessary files directly.

1.1 Introduction to Module.xml

Earlier sections introduced the basic necessary files of a module. You should already have some idea what these were:

1. a Data Definition Schema (data_schema.xml).
2. a User Interface Schema (ui_schema.xml)
3. a User Interface Logic file (ui_logic.bsh)
4. a Translation (Arch16n) file (faims.properties)
5. A server-side validation file (validation.xml)
6. CSS Files for styling the modules (ui_??)

You've already learned that you don't have to design and code all of these files from scratch. Instead, you'll create one file that serves as a set of instructions for the FAIMS Tools to create the necessary files for you. That one file is called module.xml.

1.2 The Module Creation Process

Before getting into the nitty-gritty specifics of how to structure module.xml, it's useful to review what the overall process of creating a module will look like. Below is an overview of the steps you'll follow.

Quiz 1.1 Test your knowledge: a
gentle descent into module.xml

1. What are some programs you can use when modifying module.xml? Which programs should you avoid? If you don't remember, review the previous section, "Setting Up Your Development Environment."

Back on your Ubuntu install in the virtual machine, open up a Terminal window using the Ubuntu Dash program. You can read how here: https://help.ubuntu.com/community/UsingTheTerminal#In_Ubuntu.

The Terminal is Ubuntu's command line tool that we'll use to navigate the file system and run commands and programs, so this is probably a good time to make sure you're acquainted with it before proceeding.

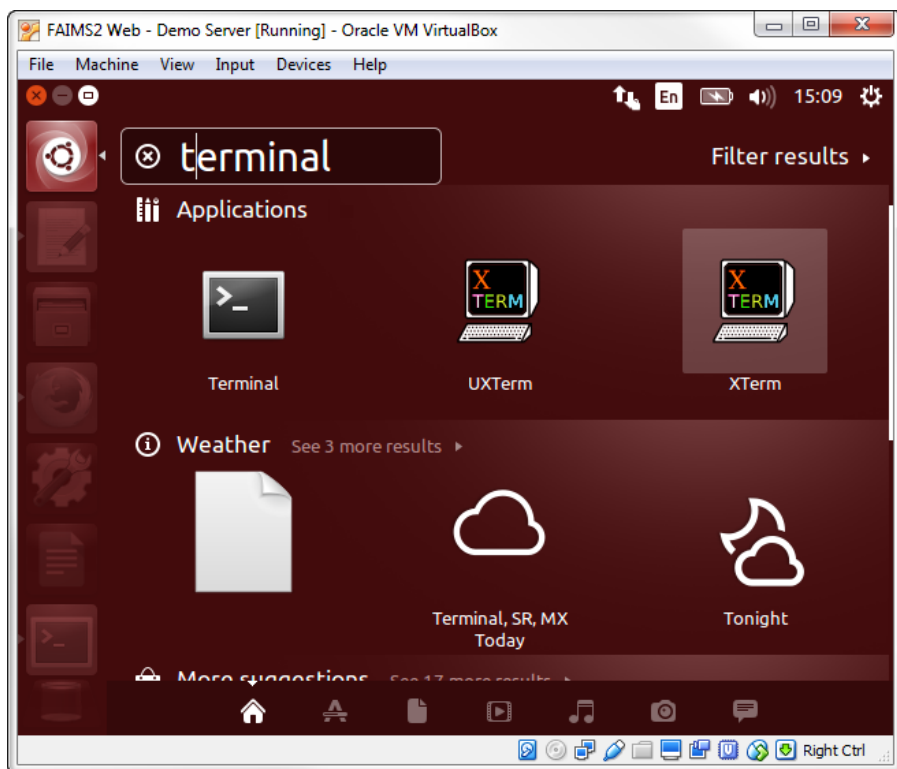


Figure 1.1 How to open the terminal using Ubuntu

Start your chosen text editor and open the “module.xml” file. If you’re using the text editor we recommended for the virtual machine, you can simply get it from the directory where you installed the FAIMS-Tools (usually FAIMS-Tools/generators/christian/module.xml). If you’re using a text editor on your main machine, you’ll need to export module.xml outside the virtual machine and work from there.

Before you make any edits, it’s a very good idea to save an extra copy of module.xml file as “moduleTemplate.xml.” The version you’re going to develop your module in and run through FAIMS-Tools needs to be named “module.xml,” but if you want to do any other module development in the future you’re going to want to have a blank lying

around to work with. Once you’ve made your backup copy, you don’t have to worry about messing around in `module.xml`.

Once you’ve finished coding, editing, and troubleshooting `module.xml` (and don’t worry; we’ll explain how to do all of that in the coming section), the instructions are complete and you’re ready to generate your module. From the same directory you originally found `module.xml` in, run the command `do so` (below) from the command line terminal. You won’t have to type the rest of a command: just hit `tab` and the terminal will try to guess¹.

```
$ cd ~/FAIMS-Tools/generators/christian
$ ./generate.sh
```

Code 1.1 BASH shell commands to run the generator.

The “`./generate.sh`” command will look through “`module.xml`” and create the necessary files that the FAIMS server needs to create a module.

Navigate to the newly created “module” folder and you’ll see that the script created 6 new files. That wasn’t so hard, was it?

¹ The first command could be typed with fewer characters `cd ~/F<tab>g<tab>c<tab>m<tab>`

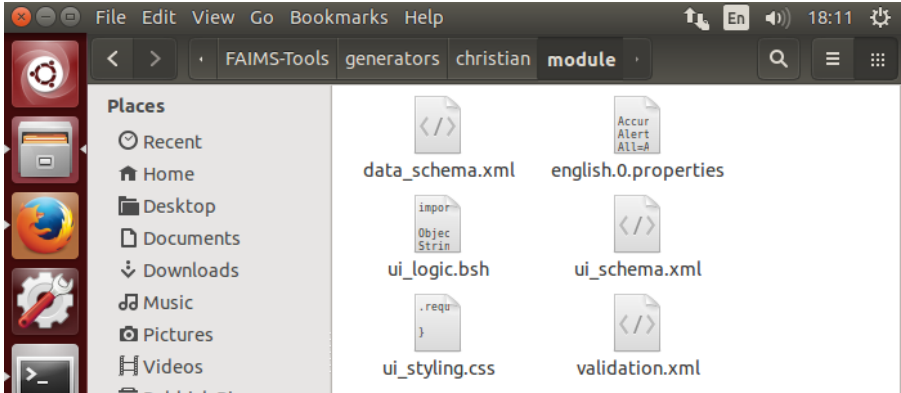


Figure 1.2 The files generated by the generator

In the web browser, on your Ubuntu installation, open up the FAIMS server and select the “Modules” tab. On the “Modules” tab, click on the “Create Module” button.

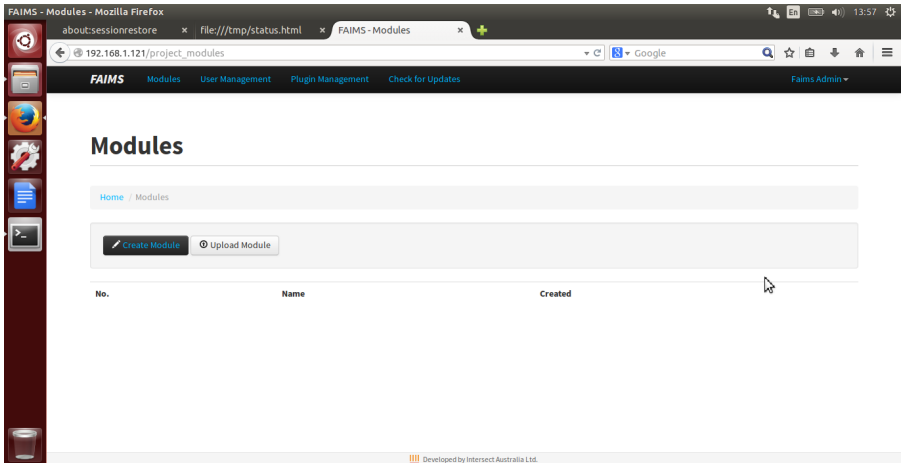


Figure 1.3 Modules page on the FAIMS server.

In a web browser, load up the web interface for the FAIMS server and log in.

The “Create Module” page offers a number of fields that allow you to describe the module name (required), version, year, description, author, and more. Give your module the name “Simple Sample Module”. On the right hand side of the screen are several boxes with file upload buttons (marked “browse”). For each of these boxes, click “browse” and attach the module necessary files you just created. Once again, the files for each box are:

1. Data Schema: data_schema.xml
2. UI Schema: ui_schema.xml
3. Validation Schema: validation.xml
4. UI Logic: ui_logic.bsh
5. Arch16n (optional, since it’s only useful if you designed your module to support interchangeable terms, but it can be ugly if you leave it out): arch16n.properties or english.0.properties
6. CSS (optional, but recommended): ui_styling.css

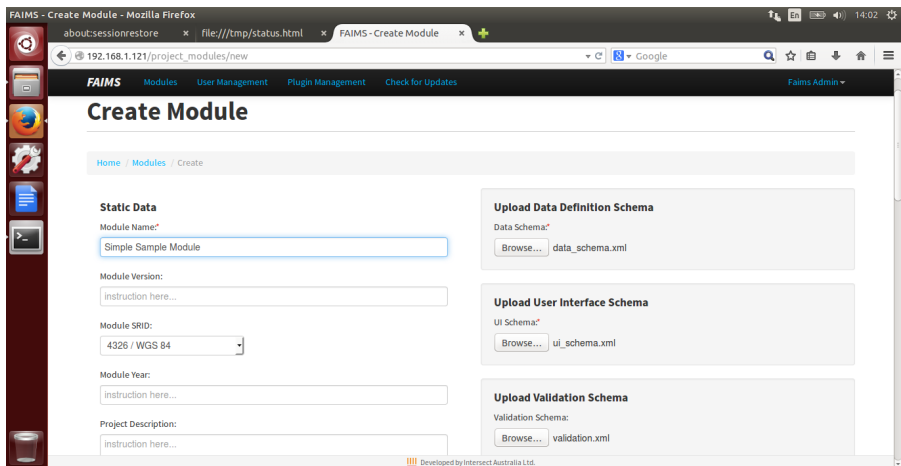


Figure 1.4 Filling in the create module page.

Click the “Submit” button at the bottom of the page to have the FAIMS Server compile your module.

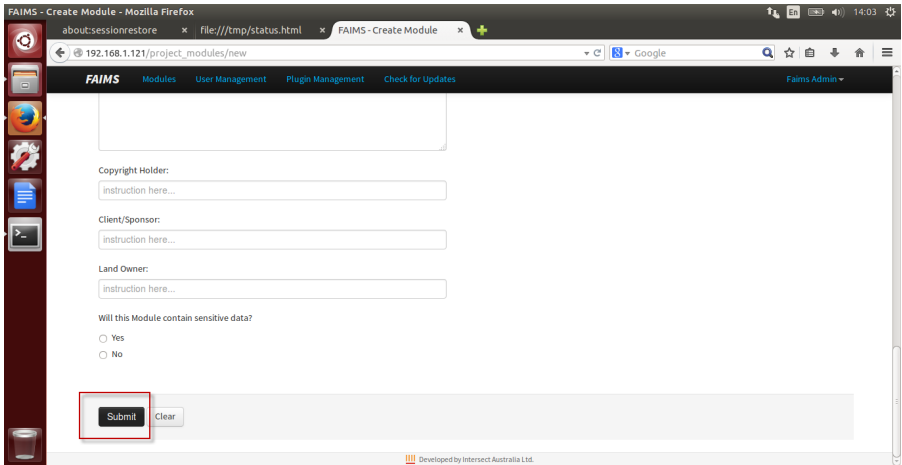


Figure 1.5 The important submit button.

Once the server creates the module, you'll be directed back to the main "Module" tab.

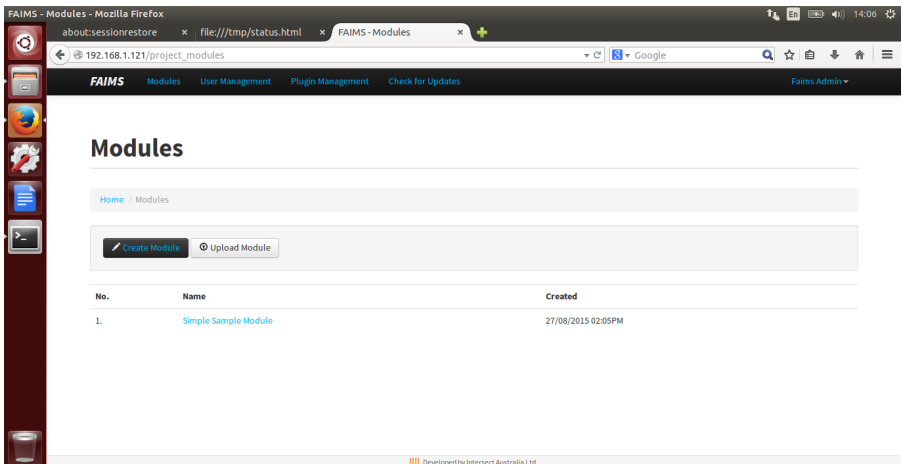


Figure 1.6 Now the module appears.

If you click on the module name, you'll see a screen where you can manage the module, including editing the module metadata, schemas, add user accounts, and downloading or exporting your module. You

may also browse any records uploaded from mobile devices using the “Module Details” area and “Search Entity Records” button. Near the bottom of the page, you can delete the module. For more information on deploying or deleting modules from the server, see FAIMS Handout 103 here: <https://www.fedarch.org/resources/handouts.pdf>

Now, open up the FAIMS mobile app on your Android device. Click on the three vertical dots in the upper right to open the “Settings” menu.

Some Samsung devices may show the menu differently, such as the Samsung S III. A Nexus 7 tablet shows it in the top right corner.

In the Setting menu, you’ll see the option to select a specific FAIMS server to connect to. By default, this is set to the FAIMS Demo server in Sydney, Australia. Click on the dropdown and you’ll see that you can set the server address manually (through the “New Server” option) or, if you’re connected to the same network as the server, you can use the “Auto Discover Server” to expedite the server setup. We’ll assume that your Android device is currently on the same wifi network as your server, so choose “Auto Discover Server” and then hit the “Connect” button. If auto-detection does not see your server, you may also select “New Server” from the dropdown list and enter the IP address manually. The IP address is visible in the address bar of your internet browser when you are connected to the FAIMS server and will look similar to “192.168.1.133”. Often, the default port of 80 will work.

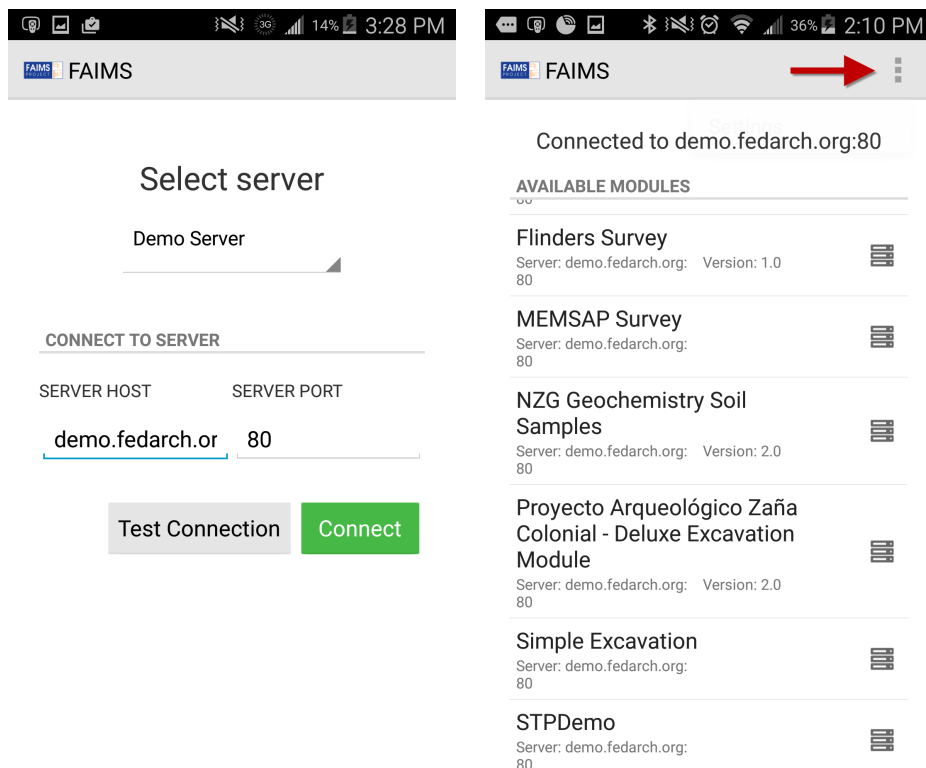




Figure 1.7 Now the module appears.

After a few seconds to a minute, the FAIMS app will find your server. Once connected, you will see a list of modules available to use, either locally on your device (designated by a blue phone icon) and those available for download from the server (designated by a black server icon). Once you have downloaded a module, it will show both icons so long as you are still connected to a server that also has the module.

A screenshot of a file upload confirmation screen. The screen has a light gray background with a thin black border. In the center, there is black text displaying the file's name, file path, and state.

```
name: media/image8.png
file: media/image8.png
state: unknown
```

Tap on the item “Simple Sample Module” from the list, which is the one we just uploaded to the server, and the FAIMS app will copy it to your device. A sidebar with the module metadata will appear. You can browse that quickly, and then tap the “Load Module” button.

A screenshot of a file upload confirmation screen, similar to the one above. It has a light gray background with a thin black border. In the center, there is black text displaying the file's name, file path, and state.

```
name: media/image39.png
file: media/image39.png
state: unknown
```

Once the module has loaded, you’ll be presented with the first tab group of the module, in this case, a user login screen. User logins allow

FAIMS to track which users are responsible for which changes. As any archaeologist who has had to decipher and track down which initials belonged to the field worker who sloppily wrote them on a level sheet or artifact bag can tell you, automatically attaching user names to record can make life much easier when going through field collections and forms back at the office or lab. You can add more users via the FAIMS server by selecting the module and clicking the “Edit Users” button.

name: media/image92.png

file: media/image92.png

state: unknown

name: media/image107.png

file: media/image107.png

state: unknown

- 1 And there you have it. That's essentially what making a module with a `module.xml` file will look like.
- 2 Of course, as you've probably realized, the part we glossed over—designing and troubleshooting `module.xml`—was also the tricky part. Fortunately, it's not that tricky after all.