Module Conventions

Naming Your Module

- Module *id* should be all lowercase without spaces. We prefer no breaks in the word (e.g., `htmlformentry`); however, you may use hyphens if absolutely necessary.
    - The Module *id* should correspond with the unqualified top-level package for classes in the module. So, org.openmrs.module.foo would have an id of foo.
    - Avoid using a module *id* that conflicts with existing modules or tables that already exist in the data model — e.g., "concept" and "form" would be undesirable module ids, since there are existing tables with these names.
    - Implementations of special functions should use dot notation: serialization.xstream, webservices.jaxws. (and this means a package of org.openmrs.module.serialization.xstream)
- Module *name* should be concise (ideally less than 20-25 letters), may contain spaces and upper case characters, but should avoid punctuation. It does not have to match the module id exactly.

Project Layout

If your module is on Github, see the the Git conventions here

Also try the OpenMRS SDK which can get you started quickly.

README conventions for modules on Github

Each module should contain a README file within the root folder named **README.md** (using markdown syntax).

Refer to

- https://github.com/openmrs/openmrs-core
- https://github.com/openmrs/openmrs-module-webservices.rest

for sample README.md files

Module Tables

- Module table names should begin with the module id — e.g., all tables added for a the "formentry" module should have table names beginning with "formentry_*".
- Be careful when naming your module to avoid conflicts with other modules or existing tables

Module Project Management

- While module developers are welcome to use their own ticketing system, we encourage module developers to use ?JIRA to track bugs and enhancement requests.
- Request a new JIRA project for your module by opening a ticket.
- Add milestones for your module to the Technical Roadmap using the naming convention: *<module name> Module x.x* — e.g., FormEntry Module 1.0. Make one milestone for the current version and one for the next version. Optionally, you can make a milestone entitled "*<module name> Module Someday*" for enhancements not anticipated in the next version.
- Module authors are responsible for keeping tickets and milestones for their module up to date

UI Conventions

- Module admin pages should fit in with other admin pages. Users like a standard look. Fancy js windows are cool but they have their place. Mimic the core jsp pages

Logging

- **Don't include a log4j.xml**: this will override the configuration in core and possibly break it. It also prevents system administrators from managing logging. If you need to change the log4j configuration then use the LogManager module. In a Mavenized module you can however include a log4j configuration file just for testing by placing it in *src/test/resources*.
- Read Java Conventions - Logging for more conventions on logging

See Also

- Creating Modules
- Creating Your First Module