OxOOL Module Development HOWTO

This manual will simply explain the file structures in a standard OxOOL module repository and how to develop an OxOOL module.

Default OxOOL Module File Structure

When you use `oxool-module-maker` to generate a template module repository, by default it will use the module template defined in the package `oxool-dev`. The important file and folder description in the template is described below:

- `ModuleConfiguration.md` : Description of module configuration.
- `module.xml.in` : Module XML file template, it will be used to generate module XML file when executing `configure`. If you need to change some configurations after generating a module, you should follow the instructions in `ModuleConfiguration.md` to edit this file and re-run `autogen.sh` and `configure`.
- `module.spec.in` : It is used to generate RPM spec file.
- `debian/*` : It is used to generate DEB file.
- `admin/*` : If this module has a backend administration page you should put it in this folder. The name shown in the admin page is defined in the `<adminItem>` tag in `module.xml.in`, or assign `--adminItem` when running `oxool-module-maker`. See README.md in the generated git repository for reference.
- `src/*` : Module C++ source files. It will be compiled generated .so file for OxOOL to load.
- `html/*` : Module HTML frontend files. The default page is `index.html`.
- `test.sh` : When developing and testing a module, this file is used to pass the module XML file to OxOOL to load. See "OxOOL Module Compiling Manual" for detail.

C++ Class Methods in an OxOOL module

In the module repo generated by `oxool-module-maker`, by default there is a `Module.cpp` under `src/` folder. In this `Module.cpp` there is the base class of this module, named by the module name. Besides constructors and destructors, there are some methods as well:

- `getVersion()` : Get version number.
- `initialize()` : Initialize.
- `handleRequest()` : Handle requests from client (frontend). You should implement here to handle the requests from the frontend web client.
- `handleAdminRequest()` : Handle requests from admin pages. You should implement here to handle the requests from admin pages.
- `handleAdminMessage()` : Handle websocket messages from backend admin pages.

Revision #1
Created 2023-09-05 02:49:25 UTC by Jeff Huang
Updated 2025-01-20 02:58:22 UTC by Jeff Huang