Migrating to Git

The OpenMRS community is currently in the process of evaluating Git as a replacement for SVN.

GitHub

There is an OpenMRS organization on GitHub (http://github.com/openmrs) used to host community OpenMRS projects. All bundled modules managed within git are kept under this organization with the naming convention "openmrs-module-*<moduleid>*". All community-supportes modules are welcome and encouraged to be kept under the OpenMRS organization as well. Other organizations and authors who maintain OpenMRS modules under other accounts within github are encouraged to use the same naming convention.

Naming Conventions

When using Git for your project, please use the following naming conventions:

- Use all lowercase.
- For modules, use the name "openmrs-module-*moduleid*" (for example: `openmrs-module-htmlformentry`); for contributions, use the name "openmrs-contrib-*contribname*" (for example: `openmrs-contrib-standalone`).
- Please include "OpenMRS module" or "OpenMRS Contrib" somewhere in your project's README.

When creating a new module or contrib, please e-mail code@openmrs.org to request a module ID or contribution name, including your OpenMRS ID, a description of your module/contribution, and your proposed module ID or contribname. The conversation with the "code" e-mail group is used to avoid duplicates and maximize consistency of naming; you should expect a module ID to be decided within one day of your request. Eventually, we hope to create a more automated process for ensuring uniqueness of module identification & awareness of existing modules.

Getting Started with GitHub

To get started with Git and GitHub you should read the documentation for your OS:

- http://help.github.com/mac-set-up-git/
- http://help.github.com/win-set-up-git/
- http://help.github.com/linux-set-up-git/

These pages explain adding SSH keys (better than passwords), and configuring your local git to use your GitHub credentials, e.g.

```
git config --global github.user username
git config --global github.token 0123456789yourf0123456789token
```

Repository conversion

The tool that Github recommends is svn2git. It expects your project to be organized in the standard SVN way of *trunk*, *branches* and *tags*, so you should reorganize the project before conversion if necessary.

Copy the list of authors from Github Authors txt into an authors.txt file and from that folder, start with:

```
$ mkdir openmrs-module-yourmoduleid
$ cd openmrs-module-yourmoduleid
$ svn2git http://svn.openmrs.org/openmrs-modules/yourmoduleid --authors ~/authors.txt
```

Be patient. The svn2git command can run for a while without any visual feedback. You can open another terminal and peek at the contents of the directory (most initial content goes into the .git folder which is hidden by default).

If you get an error from svn2git, it's possible that one or more users are missing from the authors.txt file. Checkout a local copy of the module from svn and list out all authors who contributed to the module:

```
$ svn log --quiet | grep "^r" | awk '{print $3}' | sort | uniq
```

Make sure each of these users has an entry in the authors.txt file. If anyone is missing, then find their corresponding git name and e-mail (if you cannot find these, e-mail the dev mailing list and ask for help).

NB Other alternatives to svn2git include;

1. **svn-git**

   Git comes with a pre-installed tool called **svn-git** see https://aboullaite.me/migration-from-svn-to-git-a-developer-guide/

   svn2git is just a wrap up tool around git's **svn-git ,** and it makes using **svn-git** more easier and flexible .However in some cases , it may be better to use svn-git directly

2. **The github Importer**

   The github Importer is also a great and easy to use tool to use for conversion and migration of repos from svn to git

   see https://help.github.com/en/articles/importing-a-repository-with-github-importer

Once the complete project is converted from svn to git format, you need to get it up to GitHub.

1. Create a new repo in Github (or ask for one for the OpenMRS org from code@openmrs.org)
   a. Name: openmrs-module-yourmoduleid
   b. Description: a one-line description of what the module does (look at others for examples)
   c. Make it public
   d. Initialize with a README
   e. Add .gitignore for maven
2. Edit the module's pom.xml to use the github address

```
        <scm>
                <connection>scm:git:https://github.com/openmrs/openmrs-module-yourmoduleid/</connection>
                <developerConnection>scm:git:https://github.com/openmrs/openmrs-module-yourmoduleid/<
/developerConnection>
                <url>https://github.com/openmrs/openmrs-module-yourmoduleid/</url>
        </scm>
```

3. Change the local repo you just converted to point at that one github repo "origin" (below)

```
$ git remote add origin https://github.com/openmrs/openmrs-module-yourmoduleid.git
```

```
$ git pull origin master
```

4. Assuming that your module uses the OpenMRS license and it wasn't already in subversion's root, add it:

```
wget https://raw.github.com/openmrs/openmrs-core/master/license.txt
```

5. Commit your changes locally

```
$ git add .
$ git commit -m "migrating from svn to github"
```

6. Push the code to github (below)

```
$ git push
```

Push the tags manually using the following command.

```
$ git push --tags
```

7. Add default teams to the new github repository:
    a. Full Committers
    b. Partial Committers
    c. Repo Owners

8. Under the Admin section for the new github repository, make sure Wiki & Issues are unchecked (we don't want wiki & issues going into github; rather, we want people using wiki.openmrs.org and jira.openmrs.org)

Retiring the SVN project

It's important to clearly indicate to other developers that the SVN project is no longer the active repository for the project. After successfully loading the code into github, rather than delete the project, you should:

1. **Add a text file called *MIGRATED*** to the SVN project root that contains the URL of the new repository location (example)
2. **Remove trunk and branches folders** (previous tags may be left in SVN)

```
# If you just completed running svn2git, start by stepping out of the git repo
$ cd ..


# Check out the top of the svn project
$ svn co https://svn.openmrs.org/openmrs-modules/yourmoduleid --depth=immediates
$ cd yourmoduleid


# Add a MIGRATED file and remove trunk & branches folders
$ echo "https://github.com/openmrs/openmrs-module-yourmoduleid/" > MIGRATED
$ svn add MIGRATED
$ svn rm trunk
$ svn rm branches


# Commit changes to subversion
$ svn commit -m "migrated to github"
```

Leaving the root folder with a MIGRATED text file pointing to the new location will allow any persons with old links to find the new location of the code.

As a courtesy, please notify OpenMRS developers with an e-mail to the developers mailing list (example).

Mavenization

It's preferable that older modules be mavenized prior to migrating to git, since it significantly reduces the size of the repository, making it easier for low-bandwidth developers to clone the repository locally. If your module is not yet mavenized, see Converting old code to Maven.

For Mavenized projects, IDE specific files should not be included in the repository. Git supports an ignore file similar to SVN called *.gitignore*, and so for a typical mavenized module project this will look like:

```
.classpath
.idea
.project
.settings/*
api/.classpath
api/.project
api/.settings/*
api/target/*
omod/.classpath
omod/.project
omod/.settings/*
omod/target/*
```

Mavenized projects do not include JAR files, but if your project was previously a normal project with JAR files, then these will still exist in the repository history, and will make your new Git repository much larger than it needs to be. However JAR files can be removed from the history of a project using the method described on this page:

```
# Remove all JAR references from current branch history
git filter-branch --index-filter 'git rm --cached --ignore-unmatch *.jar'

# Push to origin repo
git push origin master --force

# Cleanup local objects
rm -rf .git/refs/original/
git reflog expire --expire=now --all
git gc --prune=now
git gc --aggressive --prune=now
```

Your pom.xml file also needs updated to refer to github instead of svn in the "scm" tags.

Authors

During the conversion, user accounts for the OpenMRS SVN repository can be mapped to personal GitHub accounts. This is done using a text file which lists account mappings as follows:

```
johnd = John Doe <john@openmrs.org>
```

If authors list is not specified all the authors email id's will be randomly generated with the git hashes. The following command can generate a list of SVN accounts from an existing SVN repository:

```
$ svn -q log http://svn.openmrs.org/openmrs-modules/YOURMODULE | grep -E "r0-9+ | .+ |" | awk '{print $3}' |
sort | uniq > ~/authors.txt
```

We have a shared authors file. Please use and add to this as needed: Github Authors txt!

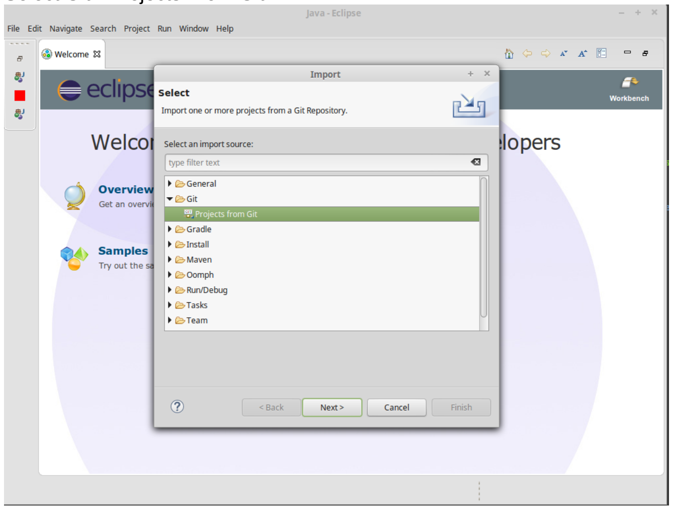
IDE integration

Git has a relatively easy to use command line interface. However all the major Java IDEs have some support for Git:

Eclipse

- EGit (available from Eclipse Marketplace)
    - Use this to clone a repository from GitHub and then import as a Maven project.
        - To clone a repository from GitHub:
            - From Eclipse, select File->Import

  - Select Git->Projects From Git



Import

Select

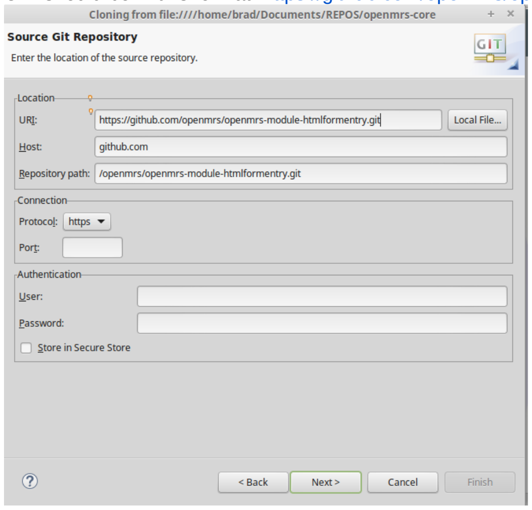
Import one or more projects from a Git Repository.

Select an import source:

type filter text

- General
- Git
  - Projects from Git
- Gradle
- Install
- Maven
- Oomph
- Run/Debug
- Tasks
- Team

  - Select URI...
  - URI should be in this format: https://github.com/openmrs/openmrs-module-htmlformentry.git



Cloning from file:////home/brad/Documents/REPOS/openmrs-core

Source Git Repository

Enter the location of the source repository.

Location

URI: https://github.com/openmrs/openmrs-module-htmlformentry.git

Host: github.com

Repository path: /openmrs/openmrs-module-htmlformentry.git

Connection

Protocol: https

Port:

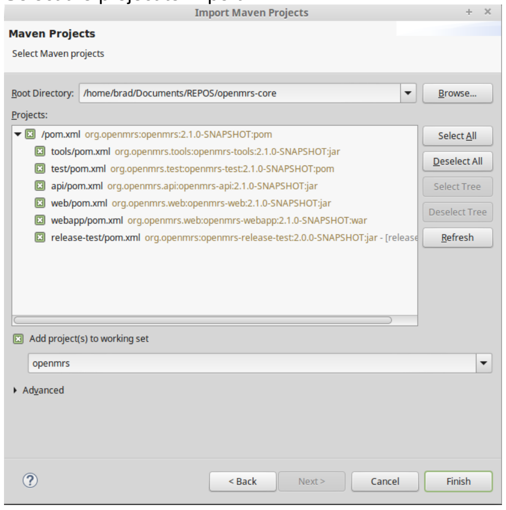
Authentication

User:

Password:

Store in Secure Store

  - Once you've cloned the repository, skip the rest of the import because you want to import this as a Maven project
- To import as a Maven project:
  - Select File->Import
  - Select Maven->Existing Maven Projects
  - Select the project to import



Import Maven Projects

Maven Projects

Select Maven projects

Root Directory: /home/brad/Documents/REPOS/openmrs-core

Projects:

- /pom.xml org.openmrs:openmrs:2.1.0-SNAPSHOT:pom
  - tools/pom.xml org.openmrs.tools:openmrs-tools:2.1.0-SNAPSHOT:jar
  - test/pom.xml org.openmrs.test:openmrs-test:2.1.0-SNAPSHOT:pom
  - api/pom.xml org.openmrs.api:openmrs-api:2.1.0-SNAPSHOT:jar
  - web/pom.xml org.openmrs.web:openmrs-web:2.1.0-SNAPSHOT:jar
  - webapp/pom.xml org.openmrs.web:openmrs-webapp:2.1.0-SNAPSHOT:war
  - release-test/pom.xml org.openmrs:openmrs-release-test:2.0.0-SNAPSHOT:jar - [release

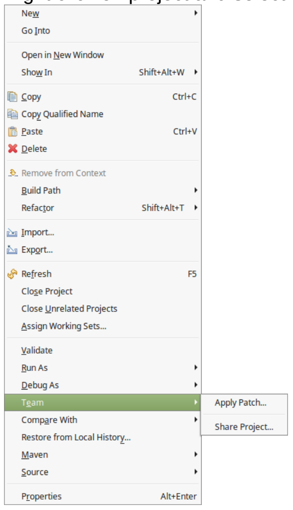
Add project(s) to working set

openmrs

Advanced

- To associate this project with Git:

        - Right click on project and select Team->Share Project



        - Select Git
        - Check "Use or create repository in parent folder of project"
        - If the HOME environmental variable is not set it will give warning:The following directory will be used for GIT user configuration and as a default repository location. If it is your home directory (e.g. C:\Users\Tom) click OK or create the HOME environmental variable pointing to your home directory and restart Eclipse.
- Maven SCM Connector for Git (available from Eclipse Marketplace)
    - Doesn't create an EGit managed project

NetBeans

- Support through plugin

IntelliJ IDEA

- Built-in support for Git and GitHub

Best Practices

Here's the set of best practices to follow in distributed OpenSource projects.