Versioning "develop" and "master"

Overview

Documentation on basic versioning strategy (as of March 2017)

Managing Versions

The develop branch of the git repository is where features are pushed and deployed to DEV for testing.

NOTE: When doing this process, make sure that eclipse is closed! This process can be run with a combination of cygwin and tortoise git.

The develop branch should always be one minor version ahead of master. For example

- master = 1.3.1
- develop = 1.3.2-SNAPSHOT

When it comes time to deploy features to PROD/UAT, the master branch needs to be updated/synchronized before that happens.

The basic process is this:

1. Pull latest changes for the repository.
2. Ensure the project is set to the "develop" branch.
3. Use "mvn versions:set" to change the versions of the develop branch to the next release version (e.g. 1.3.2)

```
cd /path/to/OTF-Mapping-Service
mvn versions:set
... when prompted, type in the next version (e.g. 1.3.2)
cd parent/
mvn versions:set
... when prompted type in the next version (e.g. 1.3.2)

... Note: If you make a mistake, use mvn versions:revert.
```

4. For mapping projects, pom-us.xml needs to be updated manually.
5. Commit the "develop" branch changes (but not need to push)
6. Switch to the "master" branch
7. Merge changes from "develop" to "master" (resolve any conflicts, rebuild code completely, confirm that ui deploys as expected)
8. Push the "master" branch (it now has the next release version)
9. Switch back to the "develop" branch.
10. Use "mvn versions:set" to change the versions to the next develop version

```
cd /path/to/OTF-Mapping-Service
mvn versions:set
... when prompted, type in the next version (e.g. 1.3.3-SNAPSHOT)
cd parent/
mvn versions:set
... when prompted type in the next version (e.g. 1.3.3-SNAPSHOT)
```

11. Commit the "develop" branch and push this change
12. Redeploy DEV to reflect the next dev version

NOTE: Confirm that pom-us.xml also gets versioned appropriately. If not, update manually.

NOTE: you may notice the complexity of running versions:set twice. This is because of the split nature of the parent and aggregator modules. It is recommended in the future that parent project be merged into the aggregator and removed. Thant involves:

1. Moving the dependencyManagement section from parent/pom.xml to pom.xml
2. Moving the pluginManagement section from parent/pom.xml to pom.xml
3. Removing "parent" from the modules list in pom.xml
4. Updating pom.xml to remove the "<parent>" section at the top (it is now the parent)
5. Also change pom.xml description to indicate that it is both parent and aggregator
    a. in particular, change the artifactId to be "...-parent" instead of "...-aggregator" (this is for the next part)
6. update all other pom.xml files in submodules to change the relativePath to the parent module
    a. from "../parent" to ".."
    b. OR from "../../parent" to "../.."

Managing Hotfixes

Occasionally, only a select few commits from develop need to be deployed on production as a hotfix. In these cases, the full merging from develop to master is inappropriate, and the following steps should be followed instead:

1. Switch to the "master" branch. Do a pull if you don't have the latest changes on the "master" branch.
2. Use CherryPick in TortoiseGit to merge the desired commit(s) from develop into master:
    a. Using TortoiseGit, select Show Log.
    b. In the top left-hand corner there will be a hyperlink saying "master". Hit that to open a dialog box showing other branches, and select the "develop" branch.
    c. Select the commit(s) to be included in the hotfix, right-click, and choose "Cherry pick this commit". Follow the prompts to in the next screen to merge this/these commits into master.
3. Use "mvn versions:set" to change the versions of the master branch. For hotfixes, the convention is to add an underscore+letter to the current version. e.g. If the current version is 1.5.2, then update the version to 1.5.2_a. If another hotfix is deployed before the next full merge from develop into master, increment the letter to 1.5.2_b, and so on.

    ```
    cd /path/to/OTF-Mapping-Service
    mvn versions:set
    ... when prompted, type in the version plus the hotfix suffix (e.g. 1.5.2_a)
    cd parent/
    mvn versions:set
    ... when prompted type in the version plus the hotfix suffix (e.g. 1.5.2_a)

    ... Note: If you make a mistake, use mvn versions:revert.
    ```

4. Commit the "master" branch changes and push.

NOTE: For hotfixes, no version changing is done on the develop branch. So in the above example, the develop version will remain at "1.5.2-SNAPSHOT".

NOTE: As with standard merges, confirm that pom-us.xml also gets versioned appropriately. If not, update manually.

Creating Release Notes Page

Whether for a full deployment or a hot fix, whenever code is versioned and deployed to production, create a release notes page to document the update.

1. Navigate to the main page for mapping tool release notes, here: Mapping Tool - Release Notes
2. Open a few of the existing release note pages listed in new tabs, for use as templates.
3. Hit the blue Create button on the top of the screen.
4. Using the other previously created pages for guidance, fill in the new page. Sections include:
    a. The title should be: ('US' if for NLM) Mapping Tool Version (VERSION) - (TODAY'S DATE in DD Mmm YYYY format) - Release Notes
    b. Tables for Defects Corrected, and/or New Features/Improvements.
        i. Copy/paste Ticket numbers and descriptions from Jira.
    c. Additional Notes, describing the release
    d. Module versions in this deployment
    e. Related dependencies for information
5. Once finished, hit the "Save" button to publish the page.
6. Copy the URL from the newly created page, and paste it on the top of the main Mapping Tool - Release Notes page. It will automatically format it to match the title of the new page.

References/Links

- n/a