Tellus Front End Configuration

Wednesday, July 26, 2017 3:19 PM

Browser configuration: /conf/config.json
PHP configuration: /conf/config-php.json, Tellus.ini

Tellus Frontend Git Workflow

Tuesday, October 10, 2017 2:59 PM

Git repo: https://gitlab.factset.com/app-qa-automation/tellus

Using repo

- Please fork the repo for development, which might help with notification for changes and code review.
- Create a personal branch in your fork, and make code changes in the branch -- not in master branch
- Then submit Pull Request to merge into upstream (App QA Automation)'s master branch

To fork the repo:

- Go to https://gitlab.factset.com/app-qa-automation/tellus
- Click **Fork**
- Clone your fork:

```
git clone git@gitlab.factset.com:<your-name>/tellus.git

cd tellus
```

Installing Dependencies

Steps to install:

1. Install composer if not already: https://getcomposer.org/Composer-Setup.exe
2. Install log4php, as a dependency in htdocs/composer.json (run bellow commands from cmd)
    a. C:
    b. cd \tellus\htdocs
    c. composer install

Pulling server changes

- Before pulling server changes, fetch meta-data first:

```
git fetch
```

- Download server changes:

```
git pull
```

Synching with Upstream (app-qa-automation group)

- Add upstream repo -- only **ONE TIME**:

```
git remote add upstream https://gitlab.factset.com/app-qa-automation/tellus
```

- Check that is was added:

```
git remote -v
```

- Open Git Bash.
- Change the current working directory to your local project.

- Check out your fork's local master branch.

  `git checkout master`

- Before merging upstream changes, fetch the branches and their respective commits from the upstream repository. Commits to master will be stored in a local branch, upstream/master.

```
git fetch upstream
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/ORIGINAL OWNER/ORIGINAL REPOSITORY
 * [new branch]      master     -> upstream/master
```

- To merge the changes from upstream's master branch into your current local branch:

  `git merge upstream/master`

- To merge from another upstream branch (e.g. v1.2):

```
git checkout v1.2
git merge upstream/v1.2
```

Standards

Versioning Standards

Use semantic versioning: http://semver.org/
Git repo initially started with version 1.1.0 in master branch.

Commit Message Standards

http://thief.factset.com/idiomatic-git/commit-standards.html

Branches

- **master branch:** release branch. It has latest pre-release (e.g. QA), or release.
- **v*<major>*.*<minor>* branch:** Version v*<major>*.*<minor>*
  - Merge developer codes during pre-release and for release
  - patch after release of version v*<major>*.*<minor>*.0. For example: v1.2.

Release Workflow

TIP: Release git tag should be done in the **main git repo**, NOT in the forks.

Future Minor Release Workflow

Suppose that current production release is v1.1.0, and you are developing for upcoming v1.2.0:

- Create branch v1.2 as feature branch from v1.1 branch
- Create and your personal feature branch from v1.2 branch, for example, *<username>*.1.2 or *<username>*.*<featurename>*
  - `git checkout v1.1`
  - `git branch jdoe.1.2`
  - `git checkout jdoe.1.2`
- Develop in your personal development branch
- When you are done, merge to v1.2 branch
  - `git checkout v1.2`
  - `git merge jdoe.1.2`

- Change package.json version (e.g. 1.2.0-qa.1 for tellusdevb01, 1.2.0 for tellusb01)
- When v1.2 (QA pre-release or production release) is ready for release, merge to master branch -- "code freeze":
  - `git checkout master`
  - `git merge v1.2`
- For pre-release, create Git tag like 1.2.0-qa.1
- For production release, create Git tag like 1.2.0
- If this is a fork, then submit Merge Request
- Build master branch (in Jenkins, for example)
- Deploy master branch to server:

tellusdevb03	Devel
tellusdevb01	QA
tellusb01	Production

Emergency Patch Release Workflow

Suppose that you need to make a small emergency patch for production as version 1.1.1 (e.g. while production version is version 1.1.0), then,

- Create your personal patch branch *<jdoe>*.1.1.1 or something similar
- Develop in personal branch
- Merge to v1.1 branch when you are done
- Change package.json version (e.g. 1.1.1)
- Create tag like 1.1.1
- If this is a fork, then submit Merge Request
- Build and release v1.1 branch

  Note: As master might be newer version (e.g. 1.2.0-qa.1 for QA release), one cannot merge version 1.1.1 to it.

If you are also working on a newer feature development branch (e.g. v1.2), then you may also need to merge the patch to that branch.

Merge Request Notes

You can Accept the Merge Request yourself (which will merge into app-qa-automation / tellus), when there is no problem after code review.

Please make sure that the target branch (e.g. v1.2) is same as the source branch.

Branch Cleanup

You can delete your personal branches when you are done

Configurations

/conf/config.json is renamed to config.example.json in source
On server, create config.json from config.example.json for server-specific configuration; config.example.json is example for all servers.

New parameter in /conf/config.json:
To display/hide ScripterBranch field:
EnableScripterBranch: true|false