Introduction to Git and GitHub

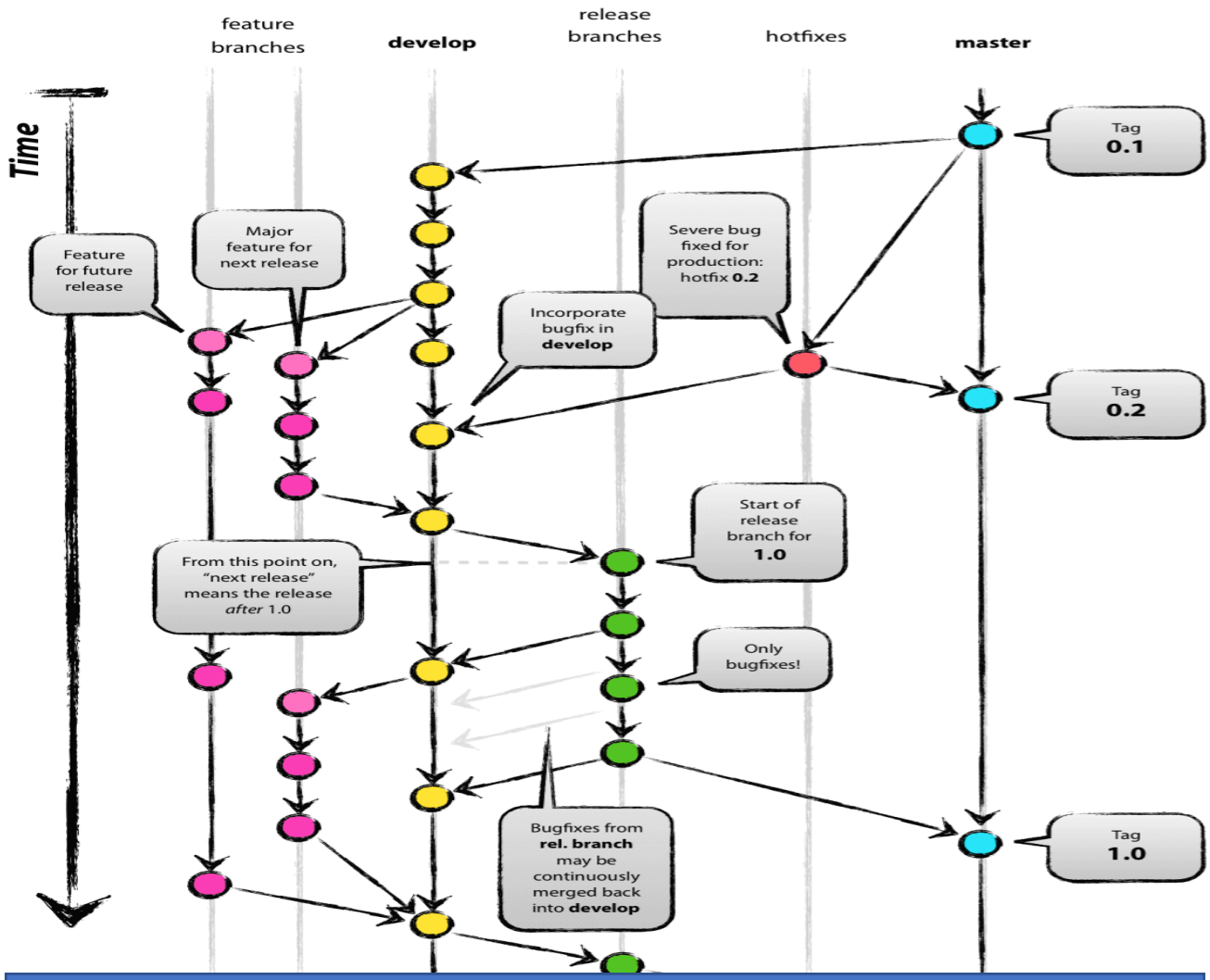
Tools for collaboratively managing your source code.

This Is Not a Tutorial About Git

There are many tutorials online.

What is Git?

- Git is a collaborative, distributed version control system.
  - Everyone has their own branch of the code in a local repository.
  - Each branch has a unique ID
  - You can work entirely separately and never give back....
- If you want to work collaboratively, you have to combine (merge) branches.
- Teams need a strategy for how to merge their branches.
- You can have multiple collaborative branches as well as personal branches.
  - ”Master”, “Feature”, “Test”, “Release”



See http://nvie.com/posts/a-successful-git-branching-model/

1000 Words about Previous Picture

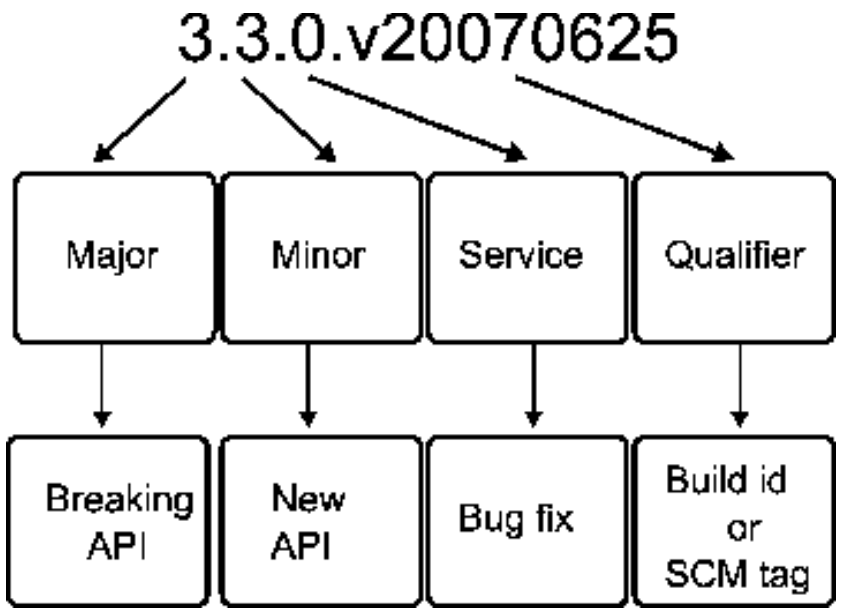
Branch	Description
Master	All other branches trace back to here. Final releases are here. Must always build and pass all tests.
Develop	Code for next version of Master. Integration Branch. Everyone's code goes back here. Must always build and pass all tests.
Feature	Working branches with code not ready for integration. May have 1 or more developers. Goes away when merged back into Develop.
Release	Code that is preparing to go back to the Master. Only bug fixes.
Hotfix	Code that fixes a bug discovered in Master that must be fixed immediately. Merged back to both Master and Develop branches.

Only the Master and Develop branches live forever!
Only the Master branch is continuous.

What Is the Right Strategy for Your Team?

- The above is just one example.
  - It works well with continuous integration while allowing feature development.
  - But it could be a lot of overhead for a small team.
- Your team decides its own branch and merge strategy.
- Decide on the team discussion list.
- Have a way for coming to a conclusion that is public.
- You later decide you have made a mistake.
  - If so, discuss that on the your team discussion list as well.
  - Come to a conclusion as a team.
  - Then change.

A Word about Semantic Versioning



See http://semver.org/.

Image courtesy https://www.tomaz.me/2013/10/28/libcloud-and-the-road-to-1-0-release.html

GitHub

What Is GitHub?

- A public repository for open source code that is managed with Git.
- Tools for helping you manage your code and your community.
- And more
  - https://guides.github.com/
- GitHub also integrates with JIRA and other online tools
  - Connect a git commit to a JIRA issue.

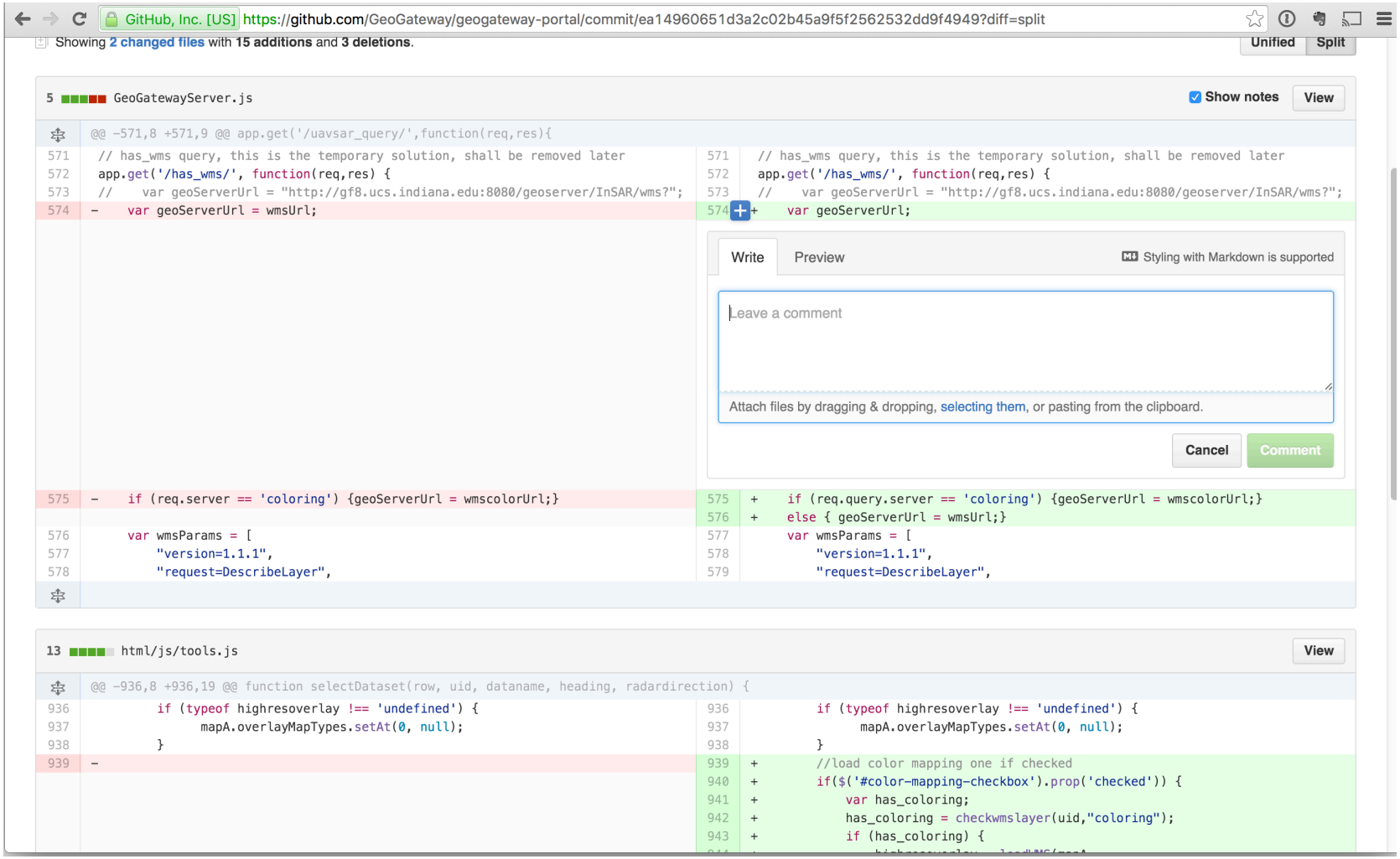
GitHub Issues

- See https://guides.github.com/features/issues/ for a full guide.
- Use this feature to discuss your project.
- Every code commit must be associated with a specific issue.
  - Include the issue number (#xxx) in your commit message.
  - See also https://help.github.com/articles/closing-issues-via-commit-messages/
- Create [VOTE] and [DISCUSS] issues for the milestone assignments ("releases").
  - Votes are +1, 0, -1
- Explore other features, see which ones are useful for your team

Pull Requests

- Notifies others of changes to a common branch.
  - Initiate reviews
- If you want to contribute a patch to a code branch that you don't have write access to, use a **pull request**.
- For simplicity in this class, each team has full access to its project repository.
  - But not the other team's repository.
- As we go along, you may want to mix and match code between teams.
  - Deciding to accept a contributed pull request to your code base is a team decision.

Code Review


GitHub, Inc. [US] https://github.com/GeoGateway/geogateway-portal/commit/ea14960651d3a2c02b45a9f5f2562532dd9f4949?diff=split
Showing 2 changed files with 15 additions and 3 deletions.
Unified
Split
5 GeoGatewayServer.js
Show notes
View
@@ -571,8 +571,9 @@ app.get('/uavsar_query/',function(req,res){
// has_wms query, this is the temporary solution, shall be removed later
app.get('/has_wms/', function(req,res) {
// var geoServerUrl = "http://gf8.ucs.indiana.edu:8080/geoserver/InSAR/wms?";
- var geoServerUrl = wmsUrl;
+ var geoServerUrl;
Write
Preview
Styling with Markdown is supported
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Cancel
Comment
- if (req.server == 'coloring') {geoServerUrl = wmscolorUrl;}
+ if (req.query.server == 'coloring') {geoServerUrl = wmscolorUrl;}
+ else { geoServerUrl = wmsUrl;}
var wmsParams = [
"version=1.1.1",
"request=DescribeLayer",
13 html/js/tools.js
View
@@ -936,8 +936,19 @@ function selectDataset(row, uid, dataname, heading, radardirection) {
if (typeof highresoverlay !== 'undefined') {
mapA.overlayMapTypes.setAt(0, null);
}
-
+ //load color mapping one if checked
+ if($('#color-mapping-checkbox').prop('checked')) {
+ var has_coloring;
+ has_coloring = checkwmslayer(uid,"coloring");
+ if (has_coloring) {

GitHub Pages and Wikis for Documentation

- Good code documents itself, but…
- https://guides.github.com/features/pages/
- https://guides.github.com/features/wikis/
- Use these to describe your project.
  - Minimally, anything the instructors need to know to check your milestones.
- Typically,
  - Use Pages for well-organized, mostly static content
  - Use Wikis for more free-form pages and page organizations.

Announcements

- Announce your project milestones
  - https://guides.github.com/activities/citable-code/
- This gives you a Document Object Identifier (DOI)
- Useful for citing code
  - “This result was produced by this specific version of our code”

Some “Apache Way” Lessons

- Community over code.
- Discuss issues publically in an archived, citable manner.
- Assign yourself to issues.
  - Volunteer
- Cite the issue(s) associated with each commit.
- Review pull requests for code bases you can’t write to
  - Patches -> Apache
- Call votes on important decisions
  - Team policies with git branches, code review, issue organization, agile policies
  - Software releases
  - Granting write access to important branches.
- Make and announce your source code releases.
- And be prepared for what happens next
  - Documentation, build systems, bug handling, code licensing, code attributions, …