



UNDERSTANDING VERSION CONTROL

Git and SVN

Overview

- What problems does version control solve?
- How does version control work?
- Compare and contrast two popular systems
 - SVN
 - Git
- Answer questions

Audience survey

- How many use a VCS?
- Which one(s)?
- If none: why not?
 - Going to try to overcome all objections
 - Goal: I want you to all leave as version control converts!

Problems solved by version control

- More than one person working on a project
- Usable history of your code's evolution
- Multiple versions of a project (1.x, 2.x, etc.)
- Foundation for all kinds of best practices
 - Unit testing
 - Documentation generation
 - Code standards

“The Hard Way”

- Regularly make archives of your whole project folder
- Keep individual archived copies of each file
 - MyClass.php : MyClass.php.bak
 - MyClass.php.bak.yyyymmddhhmmss
- Pretend there isn't a problem and do nothing
 - Hope for the best

“The Right Way”

- Programming usually offers more than one “right” way
 - Not using version control is (almost) always wrong
 - Using any system is better than none at all
- Very low learning curve to get started
 - Some are put off by the complexity of available systems
 - Start slow and master the tool over time
- Understand how version control works
 - It is not a magic black box

Disclaimer

- This is not a comprehensive version control tutorial
 - Demonstration of daily workflow
 - See links for many excellent tutorials and references
- This will be a high-pressure sales pitch
 - I REALLY want everyone to use version control
- My VCS history (bias)
 - Visual SourceSafe, Subversion, Git (avoided CVS)
 - Honorable mention: Sourcegear Vault, MS TFS
- Lots of time allowed for questions

Understanding the terminology

- All version controls systems share common attributes
- We will focus on SVN and Git
 - Applies equally to Mercurial, Bazaar, etc
- Stop me if you have any questions

Commit

- Basic unit of work in a version control system
- Can be as big or small as you like
 - One small change, several changes, whole day's work
 - Smaller is better
- Once you commit, your work is safe

Commit message

- Include a message describing your commit
- Extremely valuable on teams or inherited code
 - Even useful for reviewing your own year-old code
- Can reference bug ID numbers for tracking
- Can trigger emails to team members or RSS feed
- <http://stopwritingramblingcommitmessages.com/>

Branch

- Analogy to trees: trunk and branches
 - All branches lead back to the trunk
- Diverge from the main code base temporarily (or longer)
- Many strategies for branching
 - Per release version
 - Per development stage
 - Per feature
- Can be merged back with the “trunk”

Tags

- Reference to a fixed point in time
- Compared to a branch which is actively developed

Merge

- Combine branches together
- Several strategies for merging
 - Normal merge includes history
 - “Compressed” merge
 - Cherry pick merge
- Some version control systems handle this better than others
 - *cough* Git *cough*

Diff

- See the difference between two versions of file
- You can also diff branches and folders
- Visual vs. text diff
 - Some even do binary diffs (images)
- Dozens of available programs (free/commercial)
 - <http://www.kaleidoscopeapp.com/>
 - Beyond Compare (windows)
 - Gvimdiff
 - Will demonstrate some later

Subversion

- Client/server model
 - Conceptually similar to using FTP
 - Companies like centralization and control
- Requires a connection to the SVN server for all tasks
- Sequential version numbers 1, 2, 3 . . .
- Works at the individual file level

Subversion Pros/Cons

- PRO Very mature and widely supported
 - Many tools integrate with SVN
 - Lots of hosting options
 - Tons of books and documentation
- PRO Easier to sell to corporate decision makers (usually)
- CON Can't work disconnected – this is huge
- CON Weaker at branching/merging

Git

- Distributed model
 - No central server by default – but there can be if you want
- Work disconnected
 - Commits, diffs, logs, branching/merging
 - Entire history is local
 - Efficient method for storage (hashing objects and storing only one time)
- SHA1 version IDs
 - F5737c51c4b645617fa3017389e873628eec0edc, scary!
- Works at the project level

Git Pros/Cons

- PRO: Work disconnected (worth repeating)
- PRO: FAST!
- PRO: Excellent branching support
 - Extreme example: <http://github.com/mootools/mootools-more/network>
- PRO: Clean directory structure (no .svn in each folder)
- CON: Not recognized by spell check



Questions?

- Are you convinced yet?



DEMO

Quotes

- "A popular misconception is that distributed systems are ill-suited for projects requiring an official central repository. Nothing could be further from the truth. Photographing someone does not cause their soul to be stolen. Similarly, cloning the master repository does not diminish its importance."
- "A small project may only need a fraction of the features offered by such a system, but using systems that scale poorly for tiny projects is like using Roman numerals for calculations involving small numbers."

Final questions / discussion

- Very easy to migrate from SVN to Git
- You can use version control on your own even if the team doesn't buy in right away
- Start tomorrow (or tonight)!

Links

- <http://gitref.org/>
 - Quick reference assembled by the GitHub team
- <http://tom.preston-werner.com/2009/05/19/the-git-parable.html>
 - Interesting essay walking you through the creation of a git-like system
- <http://book.git-scm.com/>
 - Git community driven manual/tutorial "work in progress"
- <http://www-cs-students.stanford.edu/~blynn/gitmagic/>
 - One developer's passionate discussion of his experience with Git, also a good intro
 - Source of quotes on preceding slide
 - good appendix on Git shortcomings
- <http://progit.org/book/>
 - Apress published book, authored by core Git contributor and Github developer, freely available online
- <http://gitcasts.com>
 - Videos for visual learners