

Version Management with Subversion for Designers

the why, the how and some baby steps

Lena Doppel, lena_d
educating and site building
university for applied arts | cat-x media

Designers and software

Designers are carbon based life-forms who are made of 99% of the same stuff as coders, 1% diff: we are more visual

if a software tool is not reasonable easy we won't use it

if a software tool's interface isn't well crafted we will get confused and cranky using it

if the tools aren't beautiful enough we will stop having fun using them

It's ok to hunt for beautiful easy-to-use software as much as often as you can, you're not abnormal or crazy.

My personal why

I avoid using ugly software, for the same reason I don't wear ugly clothes or buy ugly furniture so I searched for a ,beautiful' way to include version control into my Drupal workflow

So what IS this Version Management?

A complicated, annoying, notoriously elusive secret science?

Probably a good idea but rather badly executed in the coding world - from the point of view of a designer?

The perfect version management for (most) creative people would be something like Apple's „Time Machine“ for their brains and files - but without the folders and with instant telephatic access

We are all waiting for the day someone finally invents this, till then we have to do it the hard - but at least pretty - way.

Version management...

aka revision control, version control, source control, code management....

the management of changes to documents stored as computer files in something called a ,repository‘

the tracking of changes to these files over time
aka ,versions‘ or ,revisions‘ usually identified by a number or letter code

the resolving of conflicts between changes in these files, either by choosing a file’s ,version a‘ over ,version b‘ or sometimes also by ,merging‘ conflicting file versions.

Subversion (SVN)

a version control system - yeah!

one that is not CVS - the version control system used by the Drupal community. Wikipedia: „SVN was started in 2000 as an effort to write a ... system which operated much like CVS but which fixed the bugs and supplied the features missing in CVS“

But then I couldn't care less: it was the first one that came up with a stylish app when I searched for a solution in 2008.

The why

using version management is more obvious
than you might think

Why use it (as a non-coder)?

to back up your site (in more than one place)

to keep track of the development (and growth) of a site without having to manually duplicate your drupal installation folders hourly/daily/weekly/monthly

to take (some) hassle out of updating

to have a synced relationship between the content of your sites folder and your database dumps

to collaborate with others in site development.

If you feel intimidated or scared

relax: most coders are not nearly as organized themselves as they want you to believe they are

buy a good book on software bloopers, read it, laugh out loud, then reconsider if version control couldn't still somehow be useful

use only what feels useful to you

maybe get a good insurance in case a client feels the need to sue you over lost site assets or down-time.

Why not to use it

keeping your stuff in order is a lot like keeping faith:
it's time consuming, might feel unnatural and the
outcome is usually heavily distorted by entropy

version control is no magical fix for the chaotically
inclined

if your virtual and real-world desks are a mess, version
control might help but it also might not (or you might
eventually stop using it)

version control doesn't mean less work, only a different
work-flow.

The how

some baby steps toward svn greatness

My Mac toolset (yours may differ)

Versions - desk app, Subversion client

Beanstalk - online repository and deployment server

Cyberduck - desk app, FTP client that syncs

ChronoSync - desk sync program

File Merge - to compare files, part of Apple Dev Tools

Onyx - desk app to accomplish some command line tasks, like showing invisible files.

The lingo

repository - contains your project's files and folders

working copy - the local copy where you change stuff

timeline - overview of changes

update / commit - download resp. upload changed files

checkout repository - for getting a local working copy

compare diff - compare changes in a file

blame - see whose fault it is.

Setting up a repository I

local or online repository? free beanstalk repository or paid subscription?

create a repository - if you use beanstalk and you're not developing software then don't use the suggested structure unless you really need it

if online: create user(s)

create a database dump with phpMyAdmin or Drupal's backup_migrate module and copy your whole Drupal installation including the dump to your local machine.

Setting up a repository 2

import the whole drupal installation (including database dump) into the repository

checkout your working copy to your local machine

look inside the folder of your working copy, it contains your files and the (invisible) Subversion-Information, it should be roughly twice the size of your Drupal installation

get some coffee, you're done (for now).

Version Control Strategy

Q: Which Drupal files do I really need to put under version control?

A: You should version control these files:

- your sites folder
- your database dump
- your .htaccess and robots.txt (but only if you made changes to them)

Q: So, why do you ,version control‘ drupal core?

A: It’s convenient if you want to deploy the whole site from beanstalk via ftp.

Working with a repository I

update your working copy, commit changes

change files, replace files, delete files, add files

replace a folder (like in the case of a contrib module update) by first deleting the old one it from your working copy, committing the change to the repository and then adding the new folder and commit again

don't replace folders directly, since version information is stored ,by folder' and you will mess up the structure and get unnecessary (,obstructed') conflicts.

Working with a repository 2

every time you commit changes to the repository a new version number will be issued

save meaningful comments with your versions (i.e. „before views 3.0 update“ and „after views 3.0 update“ instead of „views 3.0 update“)

if possible (and useful) include a db dump

during development I find it useful to also include a pdf print-out of permissions and modules settings.

Working with a repository 3

commit often f.e. while developing a theme, only if you commit a file a new version of it will be saved, subversion is not an automated backup system

commit every time you update your site, don't forget to make a database dump before you change any files, so you can revert to a working installation

don't forget to backup and test your site and database regularly, version control is no substitute for doing so

if you work in a team update your working copy before you begin to work and update often during work.

Reverting to an old version

file? folder? or whole site? local or from repository?

locally: revert | from repository: revert to revision

revert locally or identify the online version you want to revert a file or folder to

case of „site“: for a clean reinstall: revert to the desired revision then export the repository

check the export (install it to a staging or dev server)

you can revert „up and down the timeline“.

Resolving file conflicts

Version will resolve many conflicts by itself - you won't even notice

in case of a file conflict that Version cannot resolve you are told to first update your working copy

Then Version creates several temporary files you can mix and match changes from

If you're done use ,Action -> Mark as resolved' and Version will clean up the temporary files

Now you can commit your file again.

Resolving folder conflicts

SVN stores its information in a folder, if you simply replace a folder in your working copy you will get an ,obstructed'-error, because the SVN-information is now missing

simply reverting won't help, first delete the folder both online and offline, then you can add it again

if you completely messed up the folder structure: you can always delete your working copy and checkout a new one (but first save the part you were working on someplace to re-create and then re-commit the changes).

Where to go from here?

If you like it so far you could as well try to learn more about it ;-)

A visual guide to version control

<http://betterexplained.com/articles/a-visual-guide-to-version-control/>

Version control with subversion (O'Reilly online book)

<http://svnbook.red-bean.com/index.en.html> .

Links to Apps

Versions (Mac) - <http://versionsapp.com/>

TortoiseSVN (Win) - <http://tortoisesvn.tigris.org/>

Beanstalk - <http://beanstalkapp.com/>

Cyberduck (Mac) - <http://cyberduck.ch/>

ChronoSync (Mac) - <http://www.econtechologies.com/>

Onyx - <http://www.titanium.free.fr/>

I hope this was helpful

if all is lost, at least we will go down in style

doppel@cat-x.at,
lena_d on several Drupal sites and on twitter,
www.cat-x.at