CODING & DEVELOPMENT | LINDSAY GAINES | FEBRUARY 8 2013

Managing Code and Configuration with Update Functions

(and staying sane)

* staying sane not guaranteed

MONKI

Audience

Who is this talk for?

- written custom modules for Drupal 7
- worked on a site with multiple deployments
  - dev
  - staging
  - live
- used or created install profiles and update functions

MONKII

Problem

Need to change a setting or enable a module

This change will need to be performed:

- local dev copy
- another developer's dev copy
- staging
- live!

That's a lot of clicking.

MONKII

Risks

What risks might we run into?

- setting doesn't live with the code
- future deployments may miss out
- can't run automated tests (SimpleTest)
- forget to update a deployment
  - staging site for a feature branch
  - Bob's dev instance (poor Bob!)

MONKII

This is Bob

Update functions to the rescue!

Put your trust in hook_update_N()

- works for any configuration stored in the database
- all Drupal devs should know to run updates
- configuration lives with the code in version control
- all deployments get it when updated
- config changes performed almost instantly
- your mouse will not be destroyed by excessive clicking
- Bob keeps his job

MONKII

Update functions

Introduction

- live in a module's **.install** file
- update module between major versions
- can be used to add / remove fields from content types
- provide default values for new Drupal variables
- even enable modules
- system table keeps track of current module versions so that update functions only get run once
- not just for modules - can also live in **install profiles**!

MONKII

Update functions

Function naming

mymodule_update_7000()

- update from Drupal 6 module to Drupal 7

mymodule_update_7001() ⇐ (start here)
mymodule_update_7002()

- standard update function iterations
- last update (i.e. 7002) stored in the system table

Update functions

Function naming

mymodule_update_7100()

- first update to get db ready for version 7.x-1.*

mymodule_update_7200()

- first update to get db ready for version 7.x-2.*
- updating from 6.x-2.* to 7.x-2.* gets all 70xx and 72xx updates, but **skips 71xx**

Update functions

StaySane example module

staysane_update_7001()

- very basic
- only creates / updates variables
- returns a translated message to the admin
- message will probably not been seen using

  drush updatedb

Update functions

StaySane example module

staysane_update_7002()

- restructure existing variables
- includes its own **.module** file for access to module functions
- Drupal can't guarantee that everything from the module is set up yet

MONKII

Update functions

Batch updates

- need to modify a large set of data
- use the **$sandbox** argument to store data
- populate **$sandbox**['#finished'] with a message when complete

See the Batch API for more information.

MONKII

Update functions

Failures

- must throw exceptions in case of failure
  - DrupalUpdateException
  - try to provide meaningful message to admins

- database errors
  - PDOException should be thrown automatically

Update functions

Module installation

- update functions are **not** run
  - if they really need to be run, call them yourself
  - think carefully about this!
- system table records module version at the last available update function

Install profiles

What are they?

- similar to modules, but living in **/profiles**
- typically many dependencies
- during install all required modules are enabled

They can set up common parts of a Drupal site:

- content types, taxonomies
- user roles and permissions
- core and module configuration settings

MONKII

Install profiles

... are very useful!

- designed for Drupal "distributions"
- you can package
  - a set of modules
  - a selection of compatible themes
  - an install profile

The install profile would provide:

- sensible defaults
- additional install steps for various settings

MONKII

Install profiles

How is a custom site like a distribution?

- deployed in multiple places (dev, staging, live)
- needs the same modules enabled everywhere

Other benefits

- site configuration in install profiles is portable
- no longer need database exports full of test data getting imported for each new deployment
- clean installs everywhere

Install profiles

Standard install profile

- creates *Full HTML* and *Filtered HTML* text filters
- sets up some default blocks
- creates *Page* and *Article* content types
- adds fields to content types
- sets sitewide variables
- creates taxonomy
- enables and sets default themes

Install profiles

A few other interesting features

- add additional steps to Drupal install

More information

How to write an Install Profile

Install Profile API

Custom modules

What do we do with them?

- completely new functionality
- custom content types
- changes to behaviour of existing modules
- bind functionality from other modules into new and custom features
- ???
- profit!

MONKI

Custom modules

Update functions

- concentrate on generic functionality
- avoid
  - settings changes
  - non-mandatory content changes

Example custom module

StaySane

Simple image carousel module.

Configurable settings

- max items to display (default: **5**)
- javascript animation speed (default: **300ms**)
- javascript animation easing (default: **linear**)

Scenario A

Update our configuration

We're using the StaySane module on a new site. A month after deployment, we want to increase the maximum items to display to **6**.

We could write an update function for **staysane.install**:

```
function staysane_update_7003() {
  variable_set('staysane_max_items', 6);
}
```

MONKII

Scenario A

What about everyone else?

- this change is forced on every site that uses StaySane
- other sites will have to stop updating StaySane

Bummer?

- it's ok to provide a default when the module is first enabled
- it's **not a good idea** to change a default value in an update function

MONKII

How do we manage this change?

In an install profile!

Thankfully, we used an install profile when we set up this site! **We can put our configuration changes there.**

sanity.install:

```
sanity_update_7003() {
  variable_set('staysane_max_items', 6);
}
```

This works the same as a module update function, but is a part of your site, **not the module**.

MONKII

Update function best practices

Modules

- changes that are necessary for **all sites** that use that module (even if there is only one)

Install Profiles

- configuration changes that are necessary for all of your **deployments**, but don't belong in a module update

This is a great reason to use an install profile on every site!

Writing configuration updates

Follow the code

- a lot of modules store configuration in **variables** table
- use a tool like Variable Changes module
  - module lets you "backup" your **variables** table
  - make your configuration change in the backend
  - view the Variable Changes report
  - put the new values in a **variable_set()**
- more advanced configuration may mean reading through the module's code:
  - admin form submit functions

MONKI

What works

What things work well in an update function?

- anything easy to do programmatically with the Drupal API
- **variable_set()**
- enabling modules
- basic content type and field modifications
- granting and revoking permissions from user roles
- any database query

What doesn't work

What things are very hard to do in an update function?

- things that are hard to do programmatically
- anything that relies on specific node ids that might be different across environments
- complex changes to content types
- enabling or placing blocks
- anything with ctools or that has ever been touched by merlinofchaos (Views, Panels, Pages, etc)

MONKII

Make it work!

Features / ctools

- both support 'exportables' in certain contexts
- learning curve is high
- can be worth it in some situations for all of the benefits of using update functions

MONKII

Testing update functions

Update functions make "one way" changes

Most important tool for testing will be a database backup and restore tool.

- phpmyadmin
- Sequel Pro
- drush
- or even simple bash scripts

MONKII

Testing update functions

Testing workflow

A nice generic workflow for testing update functions:

1. create all the test data your update function will need
2. take a backup/snapshot of the database
3. use dd() or another file-output debug command
4. run **update.php** or **drush updatedb**
5. examine your debug output
6. restore the db from the backup and repeat!

MONKII

Drupal 8

No more update functions in install profiles

- install profiles are being re-engineered
- only used during install, then never used again
- this means no more update functions!

What do we do then?

- enable a special module with the install profile
- put configuration updates in this module
- or use new Configuration Management!

Best Practices

To make your life easier

- proper use of update functions
- **must** use version control
- always run **update.php** or **drush updatedb** after updating the code from version control
- use the Devel module for debug output during testing
- look into one-click deployments (Springloops, Beanstalk), they will save you money and headaches
- learn drush! (it has lots of great tools for working with install profiles)

MONKII

Going deeper

Recommended reading

Drush Make and Install Profiles with Drupal 7

Profiler on d.o

Configuration Manager on d.o

MONKII

Thank you!

My name is **Lindsay Gaines** (aka **aethr** on d.o)
from **Monkii**

I hope this talk has been interesting and informative!

Please remember to evaluate this session:

http://sydney2013.drupal.org/managing-code-and-configuration-update-functions-and-staying-sane

You can find the examples from this talk on drupal.org:

http://drupal.org/sandbox/aethr/1903934
http://drupalcode.org/sandbox/aethr/1903934.git

MONKII

I hope you enjoyed DrupalCon Sydney!

MONKII