

Quality Assurance and the Drupal Development Process

Drupalcon DC 2009



Why "Quality Assurance"

Quality Assurance is only needed when

- the developer fails to create exactly what the client wants
- the code is not 100% bug-proof

So talking about QA is probably not appropriate for this conference as Drupal-biased sites always works and provide exactly what teh client wants.



What is "Quality Assurance"

The term "Quality Assurance" can include any and all of the following practices:

- User Stories
- User Acceptance Tests
- Automated testing (e.g., Selenium and/or Simpletest)
- Documentation (code comments and external docs)
- Development frameworks (Dev/QA/Live multi-sites)
- Multi-browser compatibility (IE 6/7/8, FF 2/3, Safari)
- User Accessibility Testing
- Usability Testing



QA is Good for Business

Greasing the wheels of communication

- Developers create the desired site
- Clients get the site they want
- Users have trouble-free experience
- Less time and money with QA built in from the beginning

QA is not an expense, it's an investment



User Stories are Key

- Capture user requirements throughout a project
- Written by a team incl. a developer and client rep.

As a [user role], I want to [goal], so I can [reason].

- Client-generated User Stories provide a simple and clear functional specification used by
 - Estimating (business development)
 - Developers
 - Quality Assurance team



User Acceptance Tests

Final testing stages by users of a new or changed system

Client-generated User Acceptance tests help to:

- Limit scope creep
- Ensure the goals of a project sprint are met

A client that is involved from the beginning helps to ensure the success of a project



When to Add QA to a Budget

- Small (under 50 hour) projects can often get by with User Acceptance Tests
- Medium-sized projects can build simple tests for key code segments identified by the Tech Lead
- Large (over 500 hour) projects need testing frameworks set up from the beginning
 - Required for post-sprint regressions tests
 - Difficult to get clients to invest \$\$ into frameworks that will guarantee the success of the project but don't show immediate results



When to use Simpletests

Always!

But in reality, particularly when a site is:

- Being developed by more than one person
- May be extended or maintained by other people than the original developers
- Expected to grow and transform through several iterations

Test-Driven Development (TDD) with Simpletest is fun and effective



SimpleTest



SimpleTest

Testing framework in Drupal 7 core

6.x-2.x API is compatible with HEAD

Allows for both unit tests, and functional testing with a cURL based browser.

Core and all pending patches for Drupal 7 are tested continually by testing.drupal.org



Where are we?

Drupal 6 core and a small percentage of contributed modules already have some test coverage.

Drupal 7 core has about 75%+ test coverage with over 10,000 assertions.



Save yourself time and clicking.

Tests need to be maintained like the rest of the code base.

Write focused tests dealing with discrete areas of functionality.

Tests are best written for code which is going to be maintained long term.



Code is gold, tested code is platinum.

Modules have a lifecycle beyond an individual website.

Lots of contributed modules don't have test coverage yet.



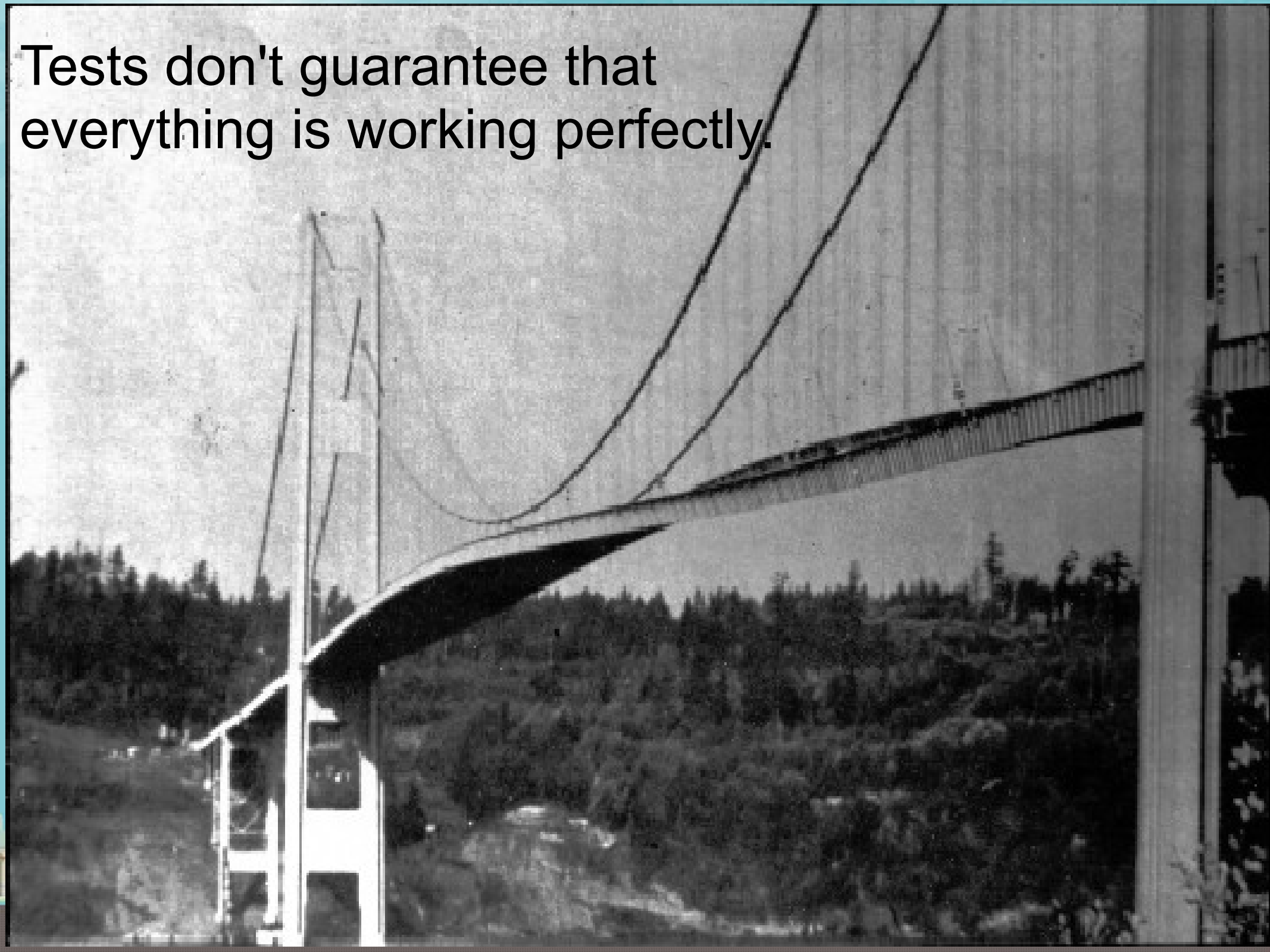
Get your patches committed faster.

- Commit **#171334** by **webchick** at 02:25
Drupal: /includes/common.inc 1.866
Drupal: /modules/simpletest/tests/common.test 1.27

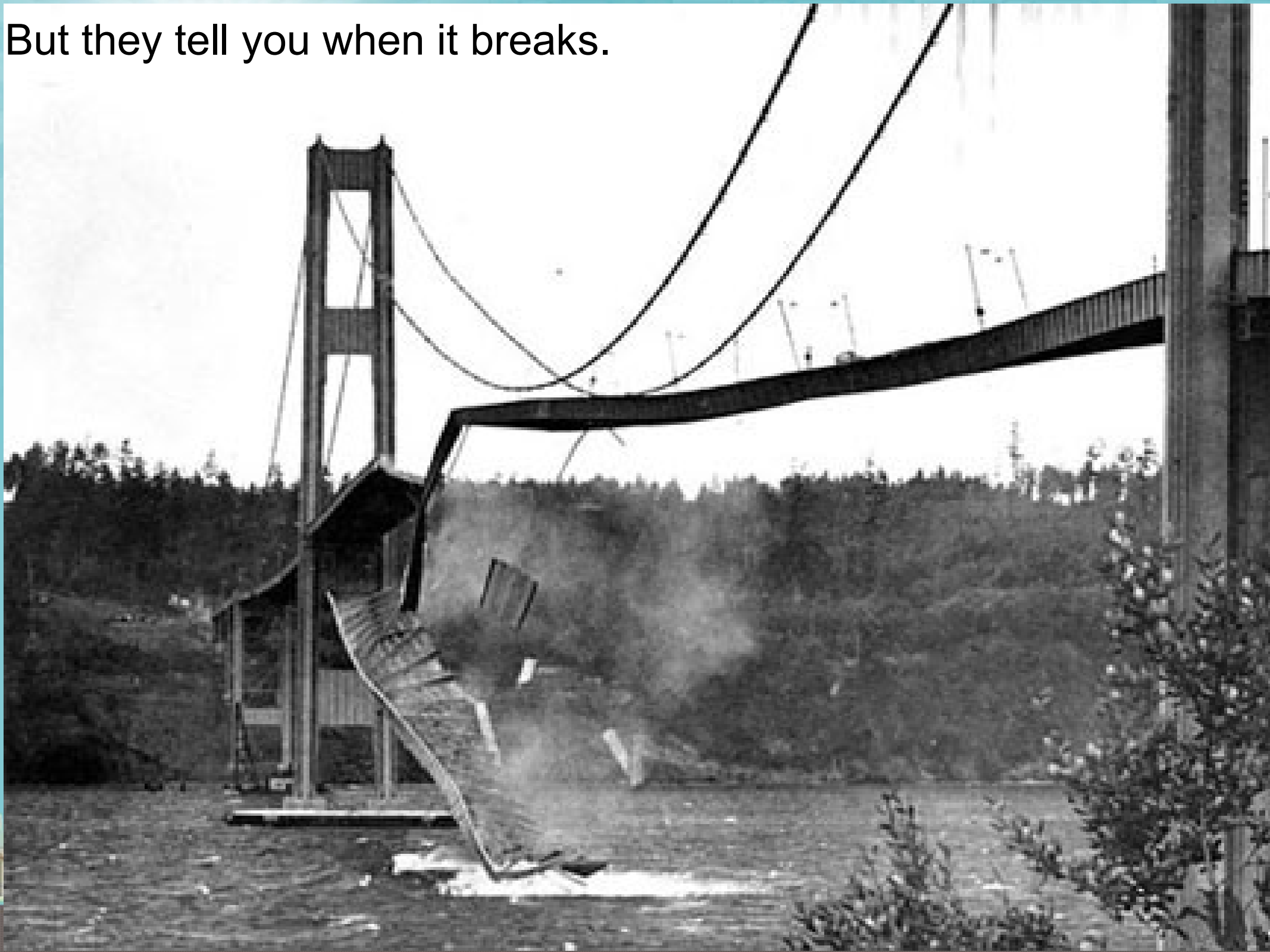
#369423 by nedjo: Fix drupal_write_record() bug with multi-field primary keys (with tests).



Tests don't guarantee that everything is working perfectly.



But they tell you when it breaks.



testing.drupal.org

Every patch in the queue for Drupal 7 is tested continually against HEAD.

Next steps:

Contributed modules!!

Code coverage



Find errors when they're introduced (or before).

#168

catch - February 22, 2009 - 21:08

Status: patch (code needs work) » patch (code needs review)

Attachment	Size
cache_node_load.patch	28.86 KB

Testbed results

cache_node_load.patch failed - [Request re-test](#) Failed: Invalid PHP syntax in modules/system/system.install. [Detailed results](#)

[delete](#) · [edit](#) · [reply](#)

#169

System Message - March 1, 2009 - 19:55

Status: patch (code needs review) » patch (code needs work)

The last submitted patch **failed testing**.

[delete](#) · [edit](#) · [reply](#)



Verify that they really are fixed.

#169

System Message - March 1, 2009 - 19:55

Status: patch (code needs review) » patch (code needs work)

The last submitted patch **failed testing**.

[delete](#) · [edit](#) · [reply](#)

#170

justinrandell - March 2, 2009 - 20:04

Status: patch (code needs work) » patch (code needs review)

getting back to this.

updated patch, conflict with update_N function only, no other changes.

Attachment	Size
cache_node_load.patch	28.85 KB

Testbed results

[cache_node_load.patch](#) re-testing

[delete](#) · [edit](#) · [reply](#)



Resources

- <http://testing.drupal.org/>
- <http://drupal.org/project/simpletest>
- <http://drupal.org/project/issues/3060/term/345>
- <http://drupal.org/simpletest> - documentation
- http://drupal.org/project/code_coverage
- <http://acquia.com/latest-drupal-test-results>
- <http://drupal.org/project/civicactions>
- http://civicactions.com/blog/Drupal_Developer_Tips_Getting_Most_out_Open_Source
- http://civicactions.com/blog/most_important_decision_developing_site_Contributed_vs_custom_development



Thank you

Fen Labalme <fen@civicactions.com>

Nathaniel Catchpole <catch@civicactions.com>

This presentation will be linked to:

<http://dc2009.drupalcon.org/session/quality-assurance-and-drupal-development-process>



Bonus Slides

Some additional information left off from the presentation



Custom vs. Generic Code

- Using contributed modules is generally a good idea
- However, if nothing is known about the module, the cost (in time and energy) can be as great or *greater* than writing from scratch
- Consider contributing back to Drupal by:
 - submitting patches to existing modules
 - writing simpletests for the modules you like and use
- Contributed modules have widely varying testing



Custom vs. Generic (more)

How should the custom vs. generic (and contributed back) code choice be made, and what are its impacts on long term quality?

Nedjo Rogers wrote two great articles addressing this:

<http://civicactions.com/blog/>

Drupal_Developer_Tips_Getting_Most_out_Open_Source

<http://civicactions.com/blog/>

most_important_decision_developing_site_Contributed_vs_custom_development



Developer Basics

Using SVN and Multi-site Effectively

- Developer Sandboxes and scripts like sync.sh that pull code, files and database from the server
- Server side DEVELOPMENT site that holds trunk - used by developers for integrated functional testing
- Server side QA site gets tagged code release and LIVE data for testing (using e.g., the pushdb script)
- Server side LIVE site gets tagged code release once it passes QA



What to Document

- User Stories
- User Acceptance Tests
- Information Architecture (sketch per sprint)
- Comments and clear code (The PIE principle: ***program intently and expressively***)

Complete documentation is considered harmful

- Document *just enough* as code and requirements change



