Wikimedia Tech Talk
Unit testing for MediaWiki projects

Antoine “hashar” Musso

April 29th 2014 - 19:00-20:00 UTC

https://plus.google.com/events/cae6ng1m9o4mhdbpo10u5v05bvg

Testing frameworks

PHPUnit

QUnit

Selenium

PHPUnit : running

```
$ cd tests/phpunit && php phpunit.php
PHPUnit 3.7.30 by Sebastian Bergmann.

Configuration read from /mediawiki/core/tests/phpunit/suite.xml

..........FF.............................   61 / 8346 (  0%)
......SSSSSSSSEEEEIIIIIIIIII..........^C
```

PHPUnit : suites

Define sub groups of tests in `tests/phpunit/suites.xml`

```
<!-- Main tests -->
<testsuite name="includes">  <directory>includes</directory>  </testsuite>
<!-- Linting tests (ie AutoLoader, ResourceLoader, file names) -->
<testsuite name="structure"> <directory>structure</directory> </testsuite>
<!-- Load tests registered with hook UnitTestsList -->
<testsuite name="extensions">
     <file>suites/ExtensionsTestSuite.php</file>
    <file>suites/ExtensionsParserTestSuite.php</file>
     <file>suites/LessTestSuite.php</file>
</testsuite>
```

PHPUnit : MW extensions

Register your test files:

```
$wgHooks[ 'UnitTestsList' ][] = function ( &$files ) {
    $files += glob( __DIR__ . '/tests/*Test.php' );
    return true;
};
```

Run the suite defined previously

```
$ cd tests/phpunit
$ php phpunit.php --testsuite extensions
```

QUnit : running

```
$wgEnableJavaScriptTest = true;
```

Then head to your [[Special:JavaScriptTest]] and ...

own logo image.

Navigation

Main page
Community portal
Current events
Recent changes
Random page
Help

Toolbox

Upload file
Special pages

Running QUnit tests

< Special:JavaScriptTest

See testing documentation on mediawiki.org.

Choose a skin to run the tests with: Vector

MediaWiki JavaScript QUnit test suite ■ noglobals ■ notrycatch ■ debug ■ completenesstest ■ mwlogenv

Hide passed tests

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.77 Safari/535.7

Tests completed in 1384 milliseconds.
478 tests of 478 passed, 0 failed.

1. **jquery.autoEllipsis: -- Initial check (0, 1, 1)** Rerun
2. **jquery.autoEllipsis: Position right (0, 4, 4)** Rerun
3. **jquery.byteLength: -- Initial check (0, 1, 1)** Rerun
4. **jquery.byteLength: Simple text (0, 5, 5)** Rerun
5. **jquery.byteLength: Special text (0, 5, 5)** Rerun
6. **jquery.byteLimit: -- Initial check (0, 1, 1)** Rerun
7. **jquery.byteLimit: Plain text input (0, 2, 2)** Rerun
8. **jquery.byteLimit: Limit using the maxlength attribute (0, 3, 3)** Rerun
9. **jquery.byteLimit: Limit using a custom value (0, 3, 3)** Rerun
10. **jquery.byteLimit: Limit using a custom value, overriding maxlength attribute (0, 3, 3)** Rerun
11. **jquery.byteLimit: Limit using a custom value (multibyte) (0, 3, 3)** Rerun
12. **jquery.byteLimit: Limit using a custom value (multibyte) overlapping a byte (0, 3, 3)** Rerun
13. **jquery.byteLimit: Pass the limit and a callback as input filter (0, 3, 3)** Rerun

QUnit : MW extensions

```
$wgHooks['ResourceLoaderTestModules'][] = function(
     array &$testModules,
     ResourceLoader &$resourceLoader
) {
    $testModules['qunit']['ext.YourExtension.test'] = array(
        'scripts' = array(
            'tests/something.test.js',
        ));
 return true;
}
```

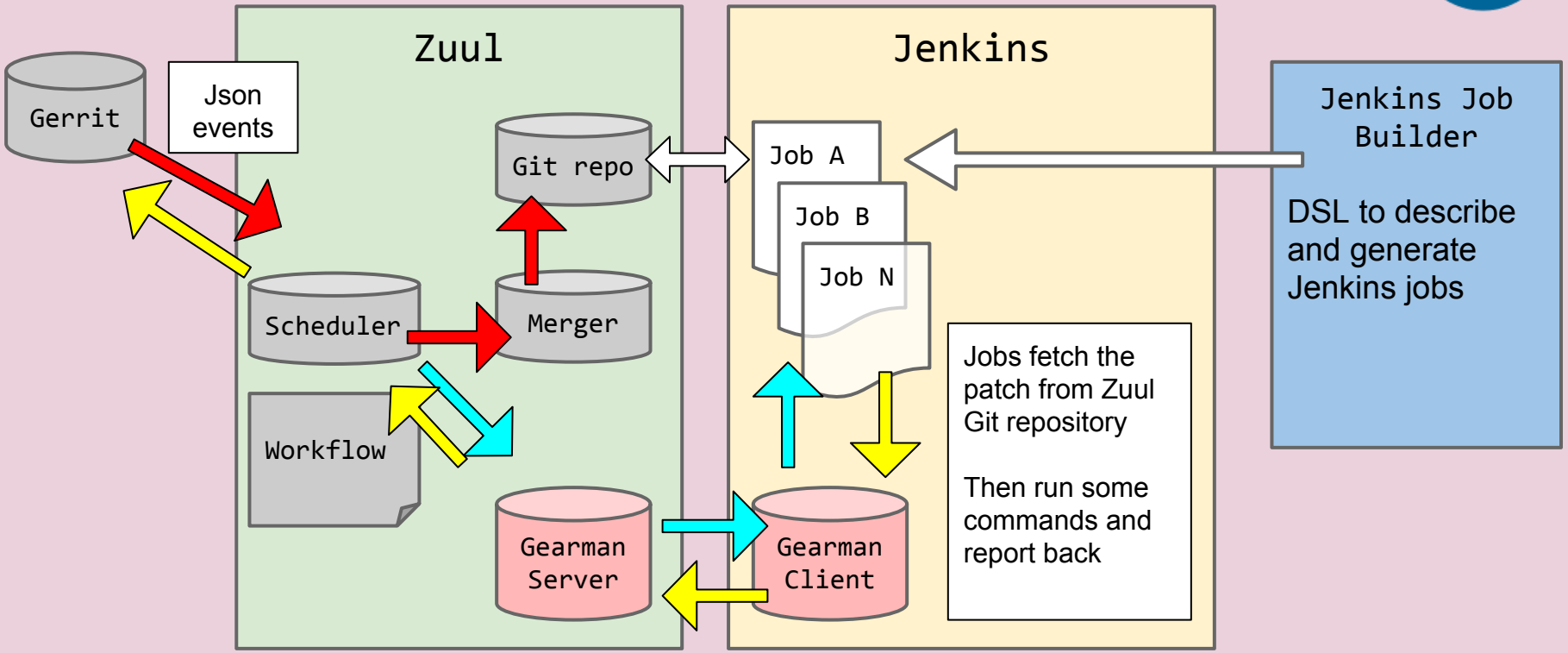
Selenium

Worth a Tech Talk on its own

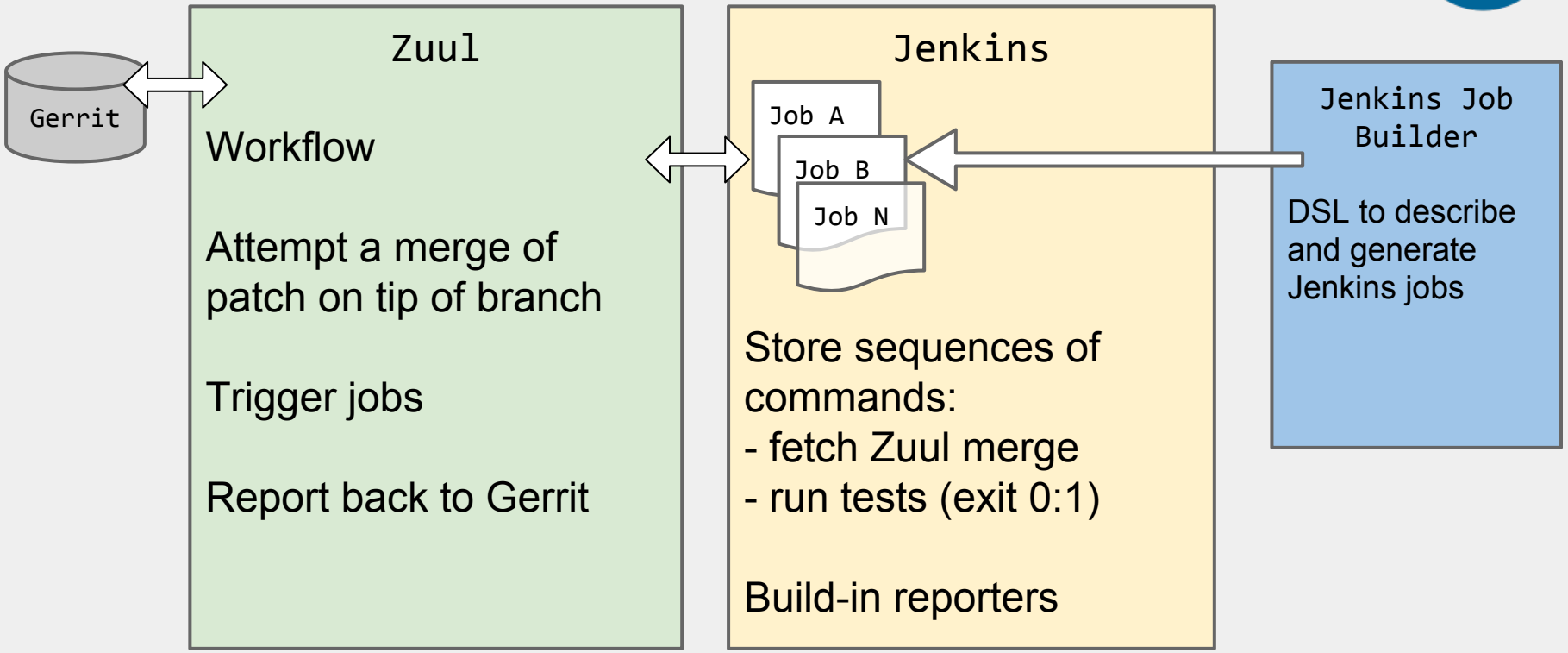
Drives the browser to replay a user browsing your web interface.

Ask on qa@lists.wikimedia.org

Test Infra


Zuul
Jenkins
Gerrit
Json events
Git repo
Job A
Job B
Job N
Jenkins Job Builder
DSL to describe and generate Jenkins jobs
Scheduler
Merger
Workflow
Jobs fetch the patch from Zuul Git repository
Then run some commands and report back
Gearman Server
Gearman Client

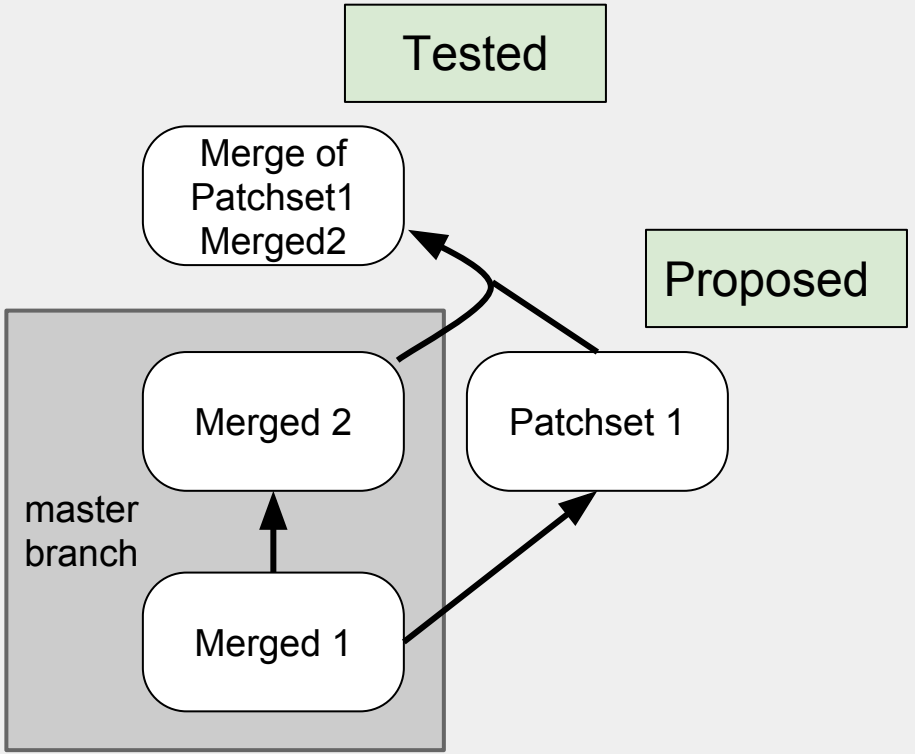
Test Infra (simplified)



Zuul merging

Patch is merged with the **tip of the branch**

If that fails: you need to rebase



Concerns

Cross repositories testing

MediaWiki extensions are tested against MediaWiki master

Extensions can depends on other extensions

mediawiki-config depends on two MW wmf branches

And so on..

Cross repositories testing

Examples:

- is that VisualEditor change still compatible with Parsoid in production ?
- would MediaWiki core patch break existing extensions?
- is an extension change compatible with both MediaWiki wmf branches ?

Barely tested

operations/mediawiki-config.git

MediaWiki conf for Wikimedia:

- almost no tests
- code barely testable

Outages could be avoided via automatic tests.

The “Unit” tests

Unit tests are supposed to:

- test a single function
- be fast

Our tests often:

- test far more than one function
- are slow
- because they do tons of things they don’t even want to test
    - Unexpected SQL
    - Localization cache

These are more link integration tests….

And integration tests are fine for some things, but too slow for everything!

Lack of tests helpers

Adding unit tests in MediaWiki should be easy

Often have to write test from scratch

Lack of helpers (ex: test an API query)

Mocking

Emulate the behavior of a real object

```
public function testNewFatalWithMessage() {
    $message = $this->getMockBuilder( 'Message' )
        ->disableOriginalConstructor()
        ->getMock();
    # $message is a fake instance of the Message class
    $status = Status::newFatal( $message );
    $this->assertEquals( $message, $status->getMessage() );
}
```

Mocking

```php
$db = $this->getMockBuilder( 'DatabaseMysql' )->disableOriginalConstructor()
    ->setMethods( array( 'fetchRow' ) )
    ->getMock();

$db->expects( $this->any() )->method( 'fetchRow' )
    ->with( $this->anything() )
    ->will( $this->onConsecutiveCalls(
        array( 'column' => 'value' ),
        false  # no more rows
    ));
$db->fetchRow();  # No database access. Returns: column: value
$db->fetchRow();  # No database access. Returns: false
```

Code coverage

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		5.17%	10976 / 212184		4.05%	502 / 12389		3.79%	46 / 1214
includes		6.35%	10111 / 159250		4.29%	480 / 11195		4.75%	46 / 969
languages		2.26%	790 / 34964		3.61%	15 / 415		0.00%	0 / 65
maintenance		0.47%	75 / 16100		0.94%	7 / 745		0.00%	0 / 172
resources		0.00%	0 / 993						
skins		0.00%	0 / 877		0.00%	0 / 34		0.00%	0 / 8

Legend

Low: 0% to 35% **Medium:** 35% to 70% **High:** 70% to 100%

Pretty low (see PHPUnit `@covers`)

How we can make testing easier

- Documentation in repository not on wiki
  - Much easier to keep up to date if you reject changes that don’t update appropriate docs
  - Easier to find for a new developer
- Push developers to Mediawiki Vagrant
  - Much more reproducible environment

How we can make testing faster

- Make it really easy to run just the tests you want to run
  - Single test
  - Single php file
  - Single extension
  - Single suite (in extension or core)
  - Some of this is already done
  - This is almost more a documentation problem then a speed one
- Speed up performance of Mediawiki Vagrant
  - rsync or nfs file system?
- Run PHPUnit in parallel somehow
- Find the slow parts of unit tests and route in mocks

How we can make testing sexy

- Graphs who's slope would make us feel better
  - Line coverage
  - File coverage
  - Branch coverage (not easy to find in PHP)
- Reports that we can use to target code most in need of testing
  - Cyclomatic complexity
  - N Path Complexity
  - Hand maintained list of scary bits of the code
- Reproducible randomized testing?
- Hamcrest?