Testing PHP Web Applications with Cucumber

Kevin Olbrich, Ph.D.
http://www.icontact.com

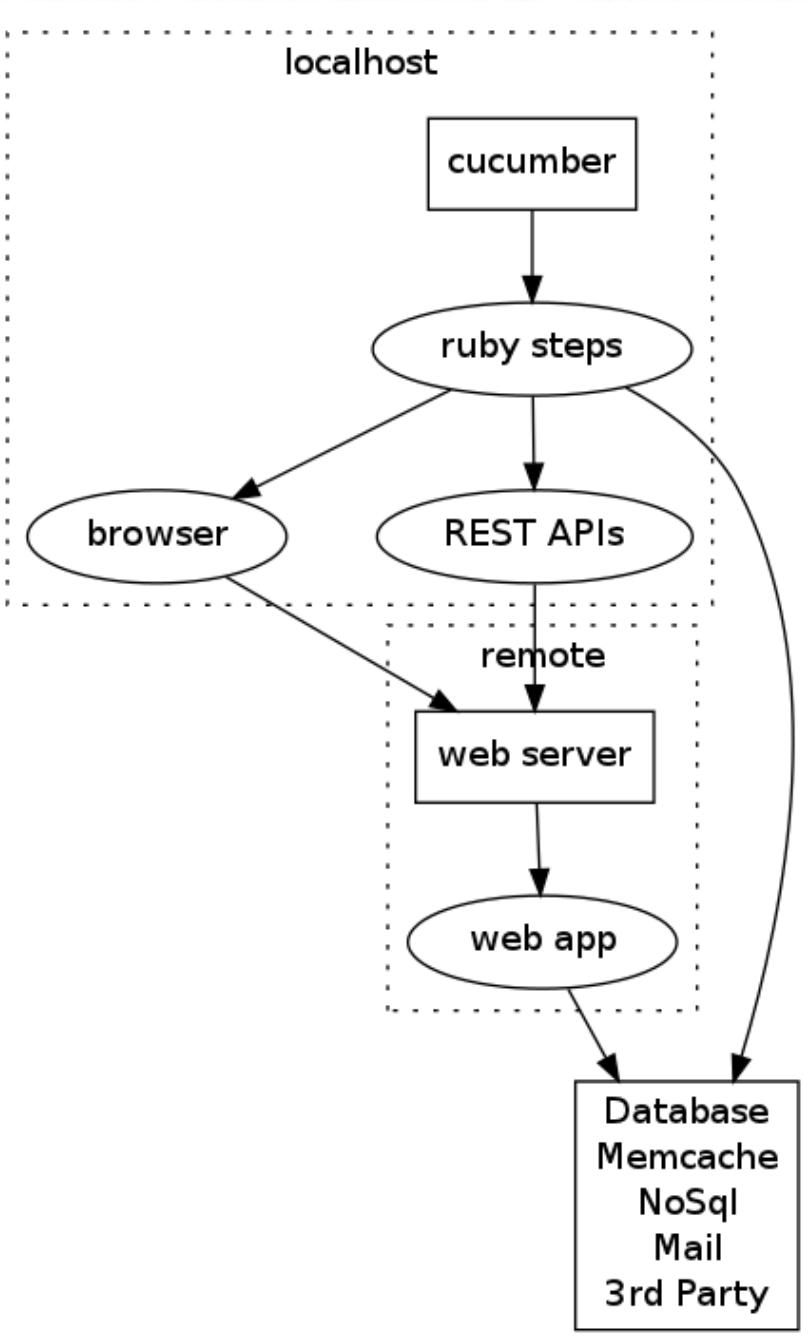
What is Cucumber?

- A Behavior Driven Development (BDD) Framework
- Describe behavior in plain text
- Write code to test the implementation
- Write code to implement behavior
- Lather, rinse, repeat

```
Feature: a simple blog
  As a user of the Website
  I want a simple blog
  So that I can see information about various topics

Scenario: I visit the blog index
  When I go to "blog/index"
  Then the page title should be "Blog Index"
  And the page should contain 1 "table" tag

Scenario: I create a blog entry
  When I go to the "blog/new" page
  Then the page title should be "Create Blog Entry"
  And the page should contain 1 "form" tag
  When I fill "Title" with "my awesome blog title"
  And I fill "Body" with "lorem ipsum, blah, blah, blah..."
  And I press "Create"
  Then the page title should be "Blog: my awesome blog title"

@remote
Scenario: I view a blog entry
  Given the "blogs" table is empty
  And a blog entry exists:
    | id    | 1                                |
    | title | yet another awesome blog post |
    | body  | I can haz blog post?             |
  When I go to "blog/show/1"
  Then the page title should be "Blog: yet another awesome blog post"
  And the page should contain "I can haz blog post?"

```

In Browser Testing

- Cucumber can drive:
  - Selenium
  - Watir
  - HtmlUnit (headless)


localhost
cucumber
ruby steps
browser
REST APIs
remote
web server
web app
Database
Memcache
NoSql
Mail
3rd Party

In Browser Testing

- Pros
  - Good representation of user experience
  - Supports javascript
  - Can be integrated into CI process (headless)
  - Human readable scenarios
  - Cross-browser testing
  - Living documentation of use cases

In Browser Testing

- Cons
  - Slow
  - Brittle
  - Can only interact with web app through web UI or REST API
- Gotchas
  - CAPTCHAs

Functional Testing - Why?

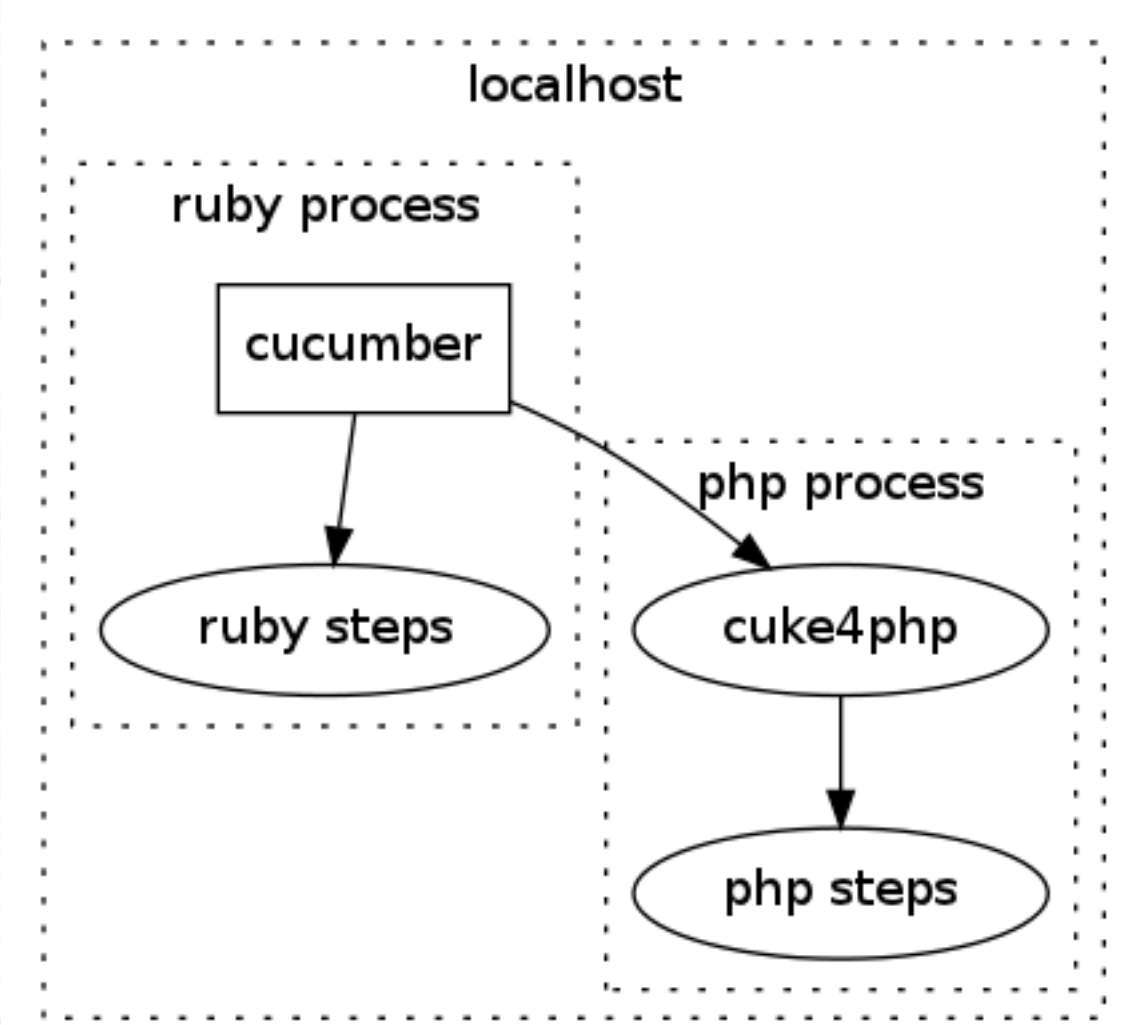
- Test behavior of high level business logic
- Test separately from user interface

Functional Testing - What?

```
1  Feature: Blogs
2    In order to manipulate blog posts
3    As a user
4    I want to be able to manipulate blog entries
5
6  Scenario: create a blog post
7    Given a blog model with:
8      | title | a blog title |
9      | body  | a blog body  |
10   When the model is saved
11   Then there should be 1 blog
```

Functional Testing - How?

- Cuke4php
  - Wire Protocol Server
  - Step definitions written in PHP


localhost
ruby process
cucumber
ruby steps
php process
cuke4php
php steps

```php
<?php

class BlogSteps extends CucumberSteps {

    /**
     * Before Hook
     * delete the contents of the 'blogs' table in preparation for a test run
     */
    public function beforeCleanDB() {
        DB::delete("blogs")->execute();
    }

    /**
    * Given /^a blog model with\:$/
    **/
    public function stepABlogModelWith($aTable) {
        $this->blog = ORM::factory('blog');
        foreach ($aTable as $aRow) {
            $this->blog->$aRow[0] = $aRow[1];
        }
    }

    /**
    * When /^the model is saved$/
    **/
    public function stepTheModelIsSaved() {
        $this->blog->save();
    }

    /**
    * Then /^there should be (\d+) blog$/
    **/
    public function stepThereShouldBeNBlog($sCount) {
        $this->assertEquals(intval($sCount),$this->blog->count_all());
    }


}
```

Functional Testing

- Pros
  - Clearly document behavior of business logic
  - Access to mocking
  - Faster than in-browser testing
- Cons
  - Slower than unit testing
  - No interprocess communication between Ruby & PHP

Remote Instrumentation - Why?

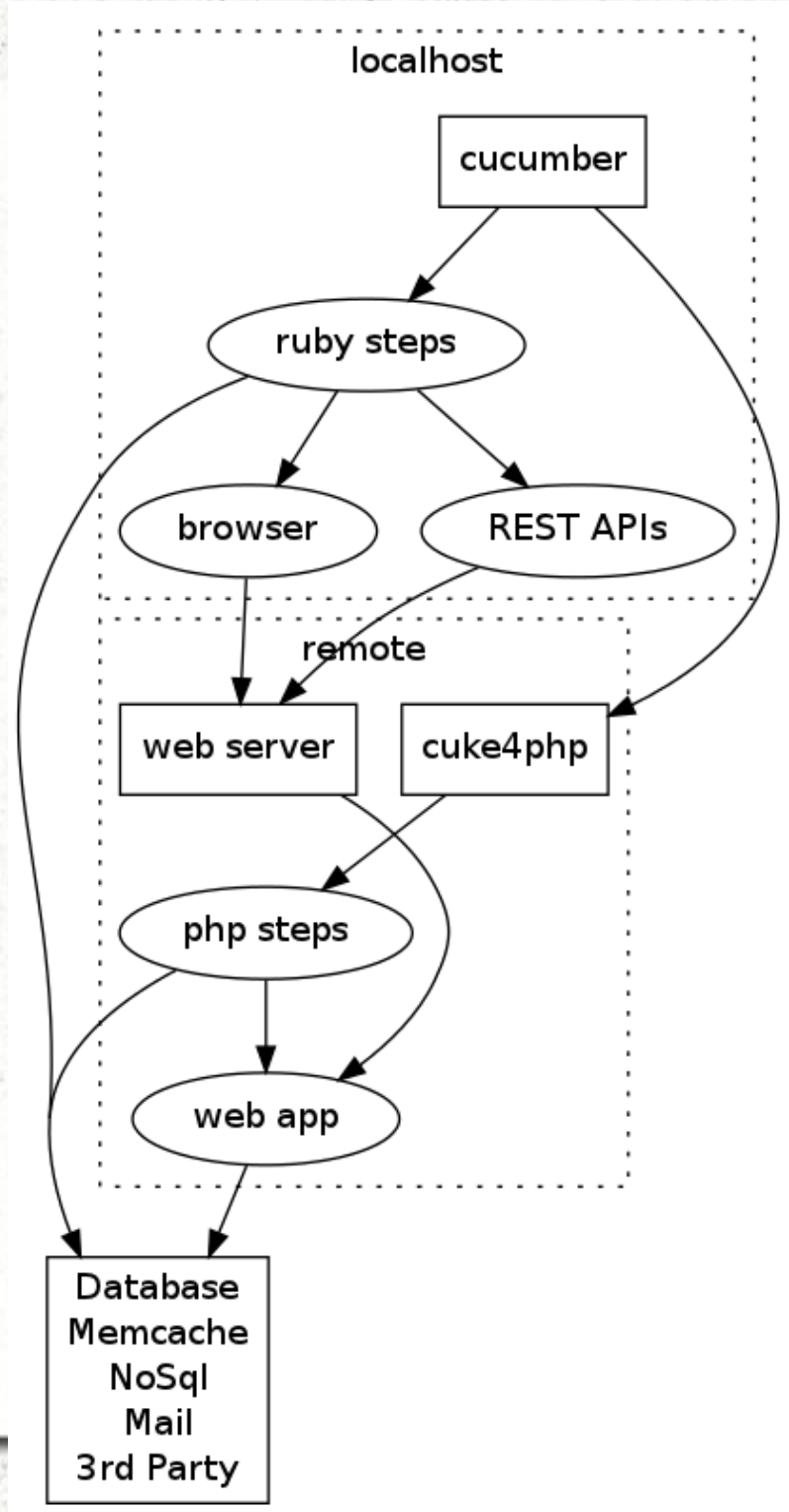
- How to quickly setup state for tests without using the UI?
- Cuke4php server listening on remote machine
- Has direct access to database and PHP codebase

Remote Instrumentation – What?

- Test account creation
- Bypassing CAPTCHAs
- Reseting database to known state
- Creating/Deleting datasets

Remote Instrumentation – How?

- Separately package and deploy cuke4php server to remote machine (never to production)
- Configure cucumber to use it


localhost
cucumber
ruby steps
browser
REST APIs
remote
web server
cuke4php
php steps
web app
Database
Memcache
NoSql
Mail
3rd Party

Code Coverage

- How do we get it?
  - PHPCoverage
  - Instrument main entry point for app
  - Run tests
  - Deinstrument
  - Generate coverage report

Spike PHPCoverage

spike SOURCE

Cucumber Demo App

Summary

Overall Code Coverage	**64.33%**	
Total Covered Lines of Code	**101**	(lines of code that were executed)
Total Missed Lines of Code	**56**	(lines of executable code that were not executed)
Total Lines of Code	**157**	(lines of executable code)
Total Lines	**691**	(includes comments and whitespaces)
Total Files	**22**	(count of included source code files)

Details

File Name	Lines				Code Coverage ▼
	Total	Covered	Missed	Executable	
/vagrant/app/application/classes/controller/welcome.php	13	3	0	3	100%
/vagrant/app/application/views/blog/new.php	14	0	0	0	100%
/vagrant/app/application/views/blog/show.php	4	2	0	2	100%
/vagrant/app/application/views/welcome.php	24	0	0	0	100%
/vagrant/app/application/classes/model/blog.php	6	0	0	0	100%
/vagrant/app/application/views/cuke4php.php	32	0	0	0	100%
/vagrant/app/application/views/remote.php	29	0	0	0	100%
/vagrant/app/application/views/browser.php	38	0	0	0	100%
/vagrant/app/application/views/coverage.php	7	0	0	0	100%
/vagrant/app/application/config/database.php	64	35	1	36	97.22%
/vagrant/app/application/bootstrap.php	131	30	1	31	96.77%
/vagrant/app/application/classes/controller/blog.php	50	15	2	17	88.24%
/vagrant/app/application/views/layout/page.php	32	4	2	6	66.67%
/vagrant/app/index.php	112	11	15	26	42.31%

Links

- Supporting Source code
  - git://github.com/olbrich/verify-ati-2011.git
- Cucumber
  - http://cukes.info
- Cuke4php
  - git://github.com/olbrich/cuke4php.git