Baker to Builder:
From CakePHP to Rails

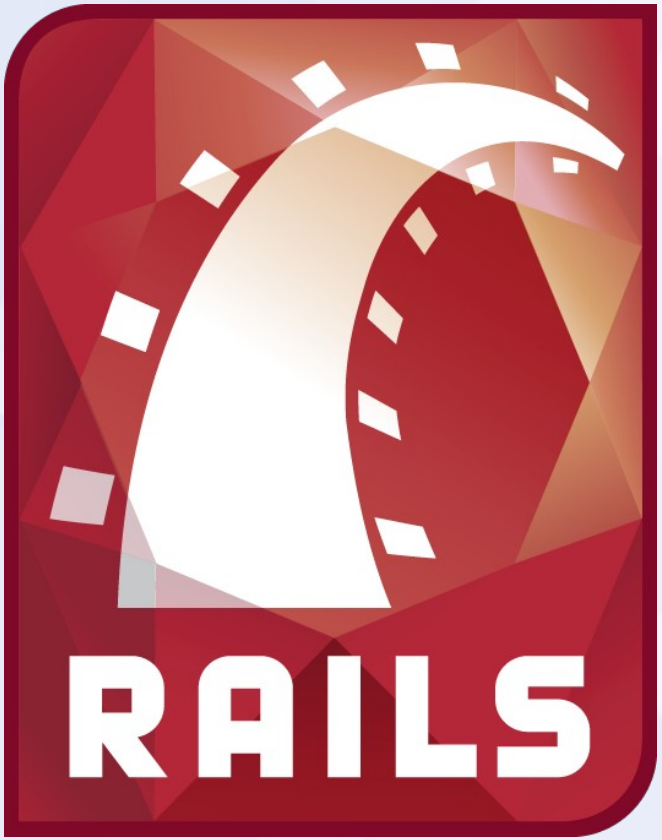
Matt Huggins

My Background

- 2003: started learning PHP
- 2007: self-taught CakePHP while developing Black Book Singles.
- 2008: started at Challenge Games as CakePHP developer.

My Background

- 2010: started at Food on the Table with no Ruby or Rails experience.
- 2011: forked & patched several gems on Github.
- 2011: Black Book Singles in Rails 3.1.



Why CakePHP?

- Already familiar with PHP.
- Writing every project from scratch is not the way to go.
- Features
  - Follows MVC principles.
  - Provides out of the box ORM.
- Has a following: support & documentation.

Familiar CakePHP Structure

- Object-oriented
- Controllers
  - Callbacks (beforeFilter, beforeRender)
  - Helpers (Text, Time, Html, Javascript, etc.)
  - Components (“controller helpers”)
- Models
  - Relationships (hasMany, belongsTo, etc.)
- Views
  - Partials
- Plugins

Why Rails?

- Joined Food on the Table team to help with front-end code & Facebook integration.
- Weaseled my way into backend code.

Benefits of Rails over CakePHP

1. Overall structure of framework
2. Ruby language benefits
3. Plugin integration
4. Deployment

Benefit #1:
Overall Structure of Rails Framework

Objects in CakePHP Controllers

```
class AppController extends Controller {
    public function __construct() {
        // append required components, helpers, and models to each inherited controller
        $this->components = array('Session', 'RequestHandler', 'GoogleAnalytics.Tracker',
                                  'DebugKit.Toolbar');
        $this->helpers = array('Profile', 'Javascript', 'GoogleAnalytics.Tracker');
        $this->uses = array('Location');

        // set admin routing properly
        if (isset($this->scaffold) && $this->scaffold === 'admin') {
            $this->scaffold = Configure::read('Routing.admin');
        }

        parent::__construct();
    }
}
```

- Define list of components, helpers, and models used within controller.
- Specify any additional controller config (e.g.: admin scaffolding).

Objects in Rails Controllers

```
def UsersController < ApplicationController
  include ControllerHelpers::UsersHelper
  include TextHelper

  # ...
end
```

- No need to specify models being used – just use them.
- Still need to include helpers being used in controllers.
  - View helpers in controllers goes against MVC principles.

Filters in CakePHP Controllers

```
class AppController extends Controller {
    public function beforeFilter() {
        requireUser();
        setLocation();
    }

    protected function requireUser() {
        $this->Auth->login();
    }

    protected function setLocation() {
        if (!$this->Session->check('Location')) {
            $location = $this->Location->findByIp($this->RequestHandler->getClientIP());
            if ($location) {
                $this->Session->write('Location', $location['Location']);
            }
        }
    }
}
```

- Only one beforeFilter method
  - all processing must occur here
- Called before every controller request
  - not just for certain actions

Filters in CakePHP Controllers

- Achieving action-specific beforeFilter must be done within the function itself.

```
class AppController extends Controller {
    public function beforeFilter() {
        requireUser();
        setLocation();
    }

    protected function requireUser() {
        if (in_array($this->action, array('index', 'show', 'logout'))) {
            $this->Auth->login();
        }
    }

    protected function setLocation() {
        if (in_array($this->action, array('index', 'new', 'edit'))) {
            if (!$this->Session->check('Location')) {
                $location = $this->Location->findByIp($this->RequestHandler->getClientIP());
                if ($location) {
                    $this->Session->write('Location', $location['Location']);
                }
            }
        }
    }
}
```

Filters in Rails Controllers

```
class ApplicationController < ActionController::Base
  protected

  def set_location
    if !session[:location]
      location = Location.find_by_ip(request.ip)
      session[:location] = location if location
    end
  end
end

def UsersController < ApplicationController
  before_filter :authenticate_user!, :only => [:show]
  before_filter :set_location, :only => [:index, :new, :edit]

  # ...
end
```

- No need to specify models being used (just any helpers).
- Any number of beforeFilter methods
  - code abstracted by utility
- Called before just the specified controller actions

Benefit #2:
Ruby Language Benefits

Ruby Blocks

- Closures
- Nothing similar in PHP
- Powerful tool, useful for many situations
- If you've used Ruby, you've used blocks

Ruby Blocks

- Loop through an array

```
list = [1, 2, 3, 4]

list.collect do |i|
  i ** 2
end
# [1, 4, 9, 16]

list.collect { |i| i ** 2 }
# [1, 4, 9, 16]
```

Ruby Blocks

- Define callbacks

```
def do_something(handler)
  puts 'About to call handler.'
  handler.call
  puts 'Handler was called.'
end

handler = Proc.new do
  puts 'SUCCESS! Inside my handler!'
end

do_something(handler)

# About to call handler.
# SUCCESS! Inside my handler!
# Handler was called.
```

Ruby Blocks

- Lambda scopes in Rails

```
class User < ActiveRecord::Base
  scope :gender, lambda { |sex| where(:gender => sex }
end

User.gender('m')

# [<User @id=1 @name='Joe' @gender='m'>,
   <User @id=3 @name='Bob' @gender='m'>,
   <User @id=7 @name='Tom' @gender='m'>]
```

Benefit #3:
Plugins

CakePHP Plugins

- Installation
  - Search the web
  - Download zip
  - Check files into code base
- Maintenance
  - Search the web
  - Download new version (if available)
  - Overwrite files in code base

Rails Plugins

- Installation
  - Add reference to Gemfile
  - bundle install
- Maintenance
  - bundle update

Benefit #4:
Deployment

CakePHP Deployment

- No implemented deployment strategy (until recently)
- Custom approach: mimic Rails
  - Numerically organized migrations
    - `001_create_users.sql`
    - `002_create_products.sql`
  - Update code on server via ssh
    - `svn update (or git pull)`
    - `mysql -uroot dbname < db/migrate/*`
  - Repeat steps for staging/production
  - Build bash script to automate

Rails Deployment

- Rails migrations automatically ordered with timestamps
  - `script/generate migration CreateUsers`
  - `rails g migration CreateUsers`
  - `rake db:migrate` runs migrations in order
- Capistrano Gem
  - Run `capify .` to create template `deploy.rb` script
  - Run `cap deploy` or `cap deploy:migrations`
- Capistrano/Multistage Gem
  - `deploy/staging.rb`
  - `deploy/production.rb`

Why Rails? (Then)

- Joined Food on the Table team to help with front-end code & Facebook integration.
- Weaseled my way into backend code.

Why Rails? (Now)

- Fast iteration
  - Rails code generation
- Accessibility of gems/plugins
  - Github
  - Bundler
- Documentation & support
- Environment management
  - rvm

It's Official

I'm a Rails convert.

Matt Huggins

http://www.matthuggins.com

http://blackbooksingles.com

http://www.foodonthetable.com