RAILSCONF EUROPE

CO-PRESENTED BY O'REILLY & Ruby Central

Rails and the Next Generation Web

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems

Craig.McClanahan@sun.com

Who Is This Guy?

- Long timer in the Java web development space
- Apache Tomcat
  - Original author, Catalina servlet container
- Apache Struts
  - Original developer, Struts 1.x
- JavaServer Faces
  - Co-specification Lead, JavaServer Faces 1.0
- Lots of conference speaking gigs

Who Is This Guy?

- And, for the ten months:
  - Developing a large scale application based on Rails
  - And liking it ...

Commercial

- Sun is holding a drawing for a **Sun Fire T1000** server:
  - World's most eco-efficient server
  - 8 core processor, 1.0GHz, 8GB, 2x73GB drives
- Drop your business card at the Sun booth:
  - Or, to one of the representatives at Sun talks
  - Limit one entry per attendee
  - Drawing on Wednesday, Sept. 19, 16:00
    - At the Sun booth

What We All Like About Rails

- Clean model / view / controller (MVC) architecture:
  - ActiveRecord for representing persistent data
  - RHTML / RXML / etc. templating
- Leverage extensibility of Ruby language:
  - Dynamically add functionality
  - “Domain specific languages”
- Incredibly productive way to build traditional server side web applications
  - With some Ajax icing on the cake

Is That All There Is?

- Traditional server side applications are still common
- But there is a new accelerating trend ...
  - *Web 2.0*?
    - Resonates with IT managers and bosses
    - Sneered at by developers
  - *Next Generation Web?*
    - The term I prefer
    - Evolution of existing practices ...
    - Towards a different architectural style

Three Tier Architectures

- Maybe not so new, but applied in interesting ways
- Rich Internet Applications:
  - Move application logic to the browser (or a plugin)
  - Exchange *data* with server, not *markup*
- Server Side Mashups:
  - UI is an aggregation of info from multiple sources
  - Some internal, some external
- Massively Scaled Applications:
  - Independently scale tiers as needed
  - Without reliance on Moore's Law

A Short Story

- Developers of Java web frameworks:
  - Are a pragmatic lot
  - *Use* frameworks they work on in day jobs
- Pace of innovation across the group is slowing down
  - Many will consider that a *Good Thing* :-)
- On the Struts developer list:
  - We asked ourselves, “what kinds of applications have you been building lately?”

A Short Story

- Common answers:
  - Client side Javascript with simple back end
    - Mostly transferring data, not markup
  - Client side mashups
    - Directly or using server side proxies
  - Large scale apps that need to *radically scale*
- See any common threads here?
  - Adding tiers to the application architecture
  - Making traditional webapp frameworks *legacy*

Architect Evaluating Frameworks Says:

Notice that old school web frameworks do not even get a mention here. But I am looking at forward facing technologies – not those facing towards the past.

But this is RailsConf Not JavaOne!

Can the same thing happen to Ruby on Rails?

What Happens If

Rails developers focus solely on two tier CRUD and server markup generation use cases?

What Happens If

Rails users leverage extensibility points in a way that limits functionality to two tier application environments?

Plugins With Common Characteristics

- acts_as_cached
- acts_as_state_machine
- will_paginate

Duck Typing

```
module Duck
  def quack
    “Quack!”
  end
end
```

```
class MyModel
  include Duck
...
end

MyModel.new.quack =>
   “Quack!”
```

Duct Tape Typing

```
module ActiveRecord
  Base.class_eval do
    def quack
      “Quack!”
    end
  end
end
```

```
class MyModel <
   ActiveRecord::Base
end


MyModel.new.quack =>
   “Quack!”
MyModel.find_by_name
  (“Daffy”) =>
  #<MyModel:xxxxx ...>
```

Plugins That Get It Right

- acts_as_authenticated
- paginator

Get Your Favorite Representations Here

- The *make_resourceful* plugin is a favorite of mine:
  - http://nex-3.com/posts/7
- Designed to DRY up RESTful controllers
  - Encapsulates common action logic
  - Before and after filters for customization
- Supports nested resources, if you like them

Get Your Favorite Representations Here

```
class PostController << ApplicationController
  make_resourceful do
    actions :create, :destroy, :edit, :index,
      :new, :update, :show
    publish :xml, :json, :yaml, :attributes => [
      :id, :title, :body,
      { :comments => [ :id, :body ] }
    ]
  end;
end
```

Get Your Favorite Representations Here

- Plugin is agnostic to underlying model functionality
- Abstract accessors separate API from implementation
  - current_object, current_objects
  - current_model, current_model_name, current_param
- Default accessors expects methods like find, save
  - But makes no assumption that ActiveRecord is used
  - Can be customized in model classes
  - We use this extensively with ActiveResource

What To See and Do In Berlin (and At Home)

- Rails framework developers:
  - ActiveResource needs a little love
  - But the framework is evolving just fine

What To See and Do In Berlin (and At Home)

- Rails framework developers:
  - ActiveResource needs a little love
  - But the framework is evolving just fine
- Rails extenders:
  - Leverage duck typing to provide functionality *without* assuming an underlying base class

What To See and Do In Berlin (and At Home)

- Developers using Rails:
  - Look for opportunities to expose application content as services (REST if you can, SOAP if you must)
  - Think of your applications as *internal mashups*, not as *monoliths*

What To See and Do In Berlin (and At Home)

- Developers using Rails:
  - Look for opportunities to expose application content as services (REST if you can, SOAP if you must)
  - Think of your applications as *internal mashups*, not as *monoliths*
- Attendees at RailsConf Europe:
  - Drink some beer and have some fun

Thank You!

Craig R. McClanahan
Senior Staff Engineer
Sun Microsystems, Inc.

Craig.McClanahan@sun.com