



MAX

360Flex - Unconference

Daniel Wanja | Flex with Ruby on Rails



1. Intro
2. Create a Rails app
3. Rails Overview
4. Rails Console
5. Rails Architecture
6. REST
7. Flex HttpService
8. Rails Resource
9. Flex ActiveResource
10. Rails Nested Resource
11. Rails Nested Attributes
12. Rails Bulk-API

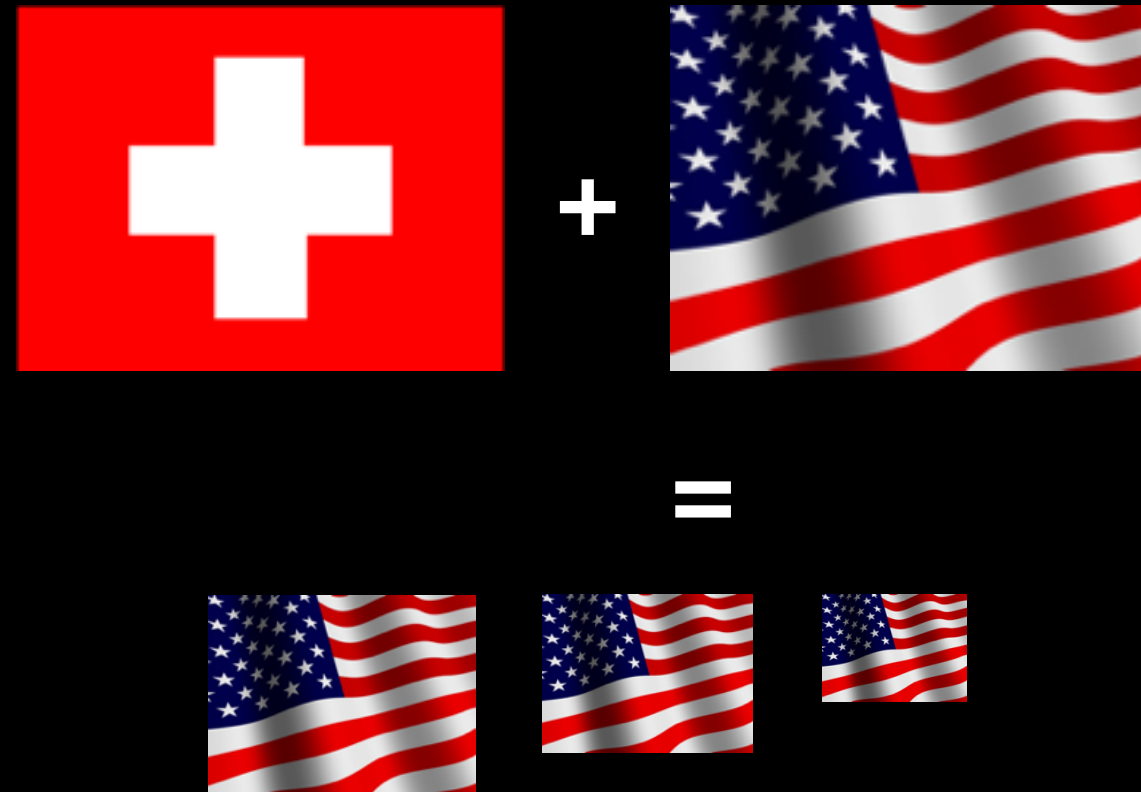
Bonus Topic as time permits

13. Zlib
14. AMF
15. Pusher
16. Gotcha's
17. Rails and the cloud
18. Heroku
19. Installing Rails

INTRO

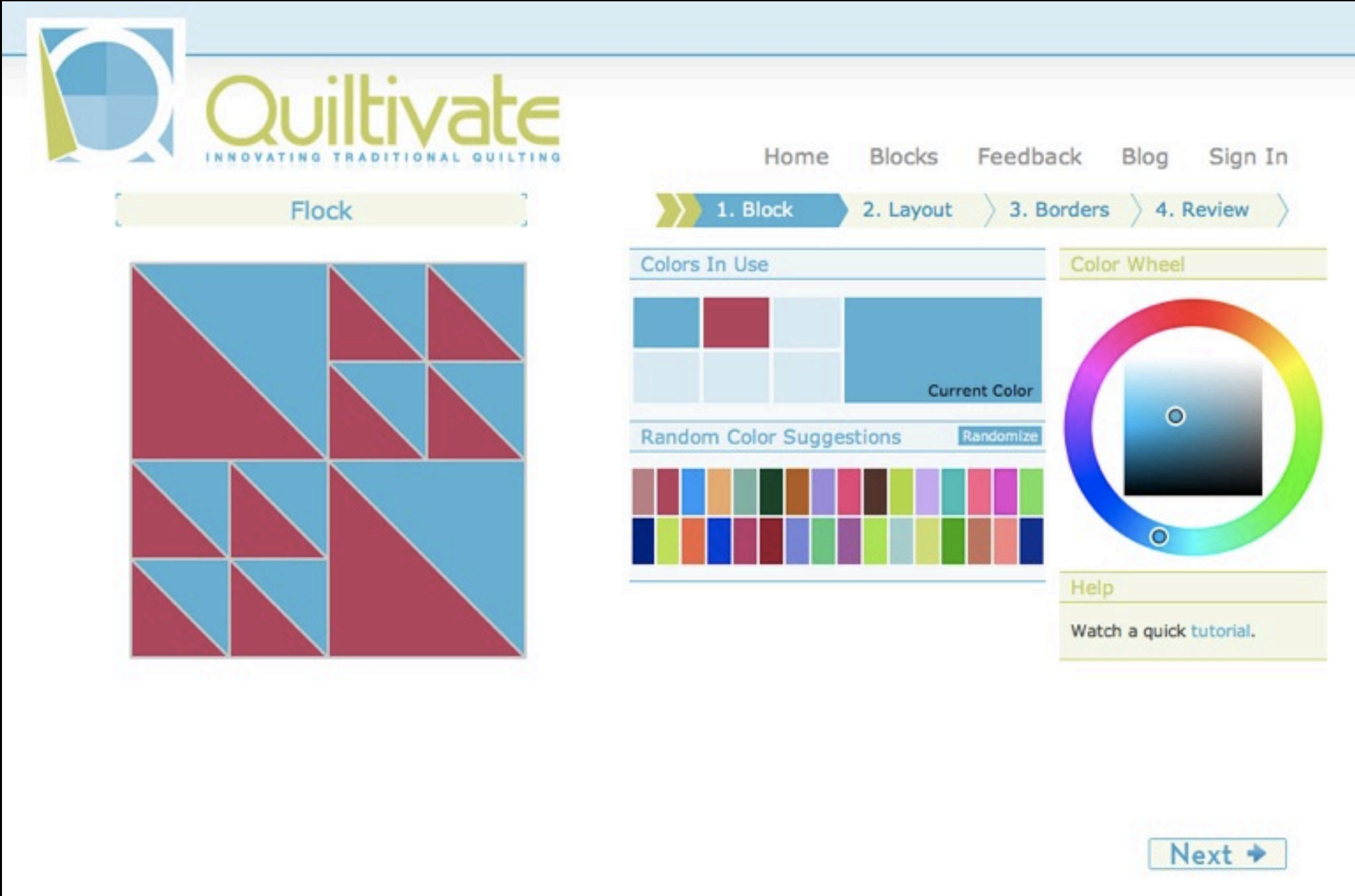
Hi...I'm Daniel!

- d@n-so.com
- @danielwanja
- onrails.org
- appsden.com
- flexonrails.com
- <http://github.com/danielwanja>





<https://www.quiltivate.com>



<http://vault.ncaa.com>



<http://vault.theacc.com/>

The screenshot displays the ACC Vault website interface. At the top, the ACC Vault logo is on the left, and the GEICO logo is on the right. Below the logo, there are tabs for "Basketball" and "Football". The main header shows the game "Florida State Seminoles 33 - 44 Virginia Tech Hokies". Below this, a video player shows a coach on the sidelines. To the left of the video is a "GAMES" section with a list of recent games: "2010 Florida State vs. Virginia Tech", "2009 Georgia Tech vs. Clemson", "2008 Boston College vs. Virginia Tech", "2007 Virginia Tech vs. Boston College", and "2006 Wake Forest vs. Georgia Tech". To the right of the video is an "OTHER FEATURES" section with "ACC GAME OF THE WEEK" (Virginia Tech vs. Miami) and a "FEATURED POLL". Below the video is a "HIGHLIGHTS" section with a list of highlights: "2006 Georgia Tech vs. Clemson" and "2010 Florida State vs. NC State". At the bottom, there is a "Championship" section with details about the game on Friday, December 3, 2010, at Bank of America Stadium, Charlotte, NC. The text describes Virginia Tech's victory and mentions QB Tyrod Taylor. A large GEICO advertisement is visible on the right side of the page.

ACC VAULT NETWORK

GEICO geico.com

Basketball Football

SEARCH

Florida State Seminoles 33 - 44 Virginia Tech Hokies VT

OTHER FEATURES

ACC GAME OF THE WEEK
Virginia Tech vs. Miami
November 12, 2008 • Watch Now

CLICK HERE FOR FEATURED POLL VOTE

Features

- 2011 profile on the Seminole's EJ Manuel, who recorded a pass efficiency rating of 138.07 in his first two seasons as Christian Ponder's fill-in, while completing 67.3 percent of his passes for 1,678 yards and 6 TDs.
- 2011 profile on Clemson's new Sophomore QB, Tajh Boyd, a talented pass and run threat who emerged from spring practice as the starter for the Tigers.
- 2011 profile of UNC's Quinton Coples, a national honors candidate at Defensive End.
- 2011 profile of Maryland Sophomore QB Danny O'Brien, who threw for the 3rd-most TD passes (22) by a rookie in ACC history and is the centerpiece of a Terrapin offense which will have four of six starters.

2010 Florida State vs. Virginia Tech
Virginia Tech clinched their third ACC title in four years and secured a berth in the Orange Bowl with a 44-33 win over Florida State. This game marked the 11th straight win by the Hokies after

2009 Georgia Tech vs. Clemson
Georgia Tech held off Clemson in a 39-34 shootout to win the 2009 ACC Championship game. This game featured plenty of offense, as neither team punted throughout the entire

2008 Boston College vs. Virginia Tech
Virginia Tech won the ACC title and a bid to the Orange Bowl with a 30-12 win over Boston College. Hokies' QB Tyrod Taylor threw for only 84 yards but managed to use his scrambling

2007 Virginia Tech vs. Boston College
Virginia Tech captured the ACC title and an Orange Bowl berth with a 30-16 win over Boston College. This marked the 2nd time in 3 years that Virginia Tech made an appearance in

2006 Wake Forest vs. Georgia Tech
In a low-scoring defensive battle, Wake Forest

2006 Georgia Tech vs. Clemson
#89 WR James Johnson some how comes up with an amazing touchdown catch. (James Johnson, Georgia Tech)

2010 Florida State vs. NC State
#16 QB Russell Wilson finds a big hole in the Florida State defense and sneaks it in from 10 yards out for a touchdown. (Russell Wilson, NC

Overview Highlights Play by Play Filter Plays

Championship

Friday, December 3, 2010
Bank of America Stadium, Charlotte, NC

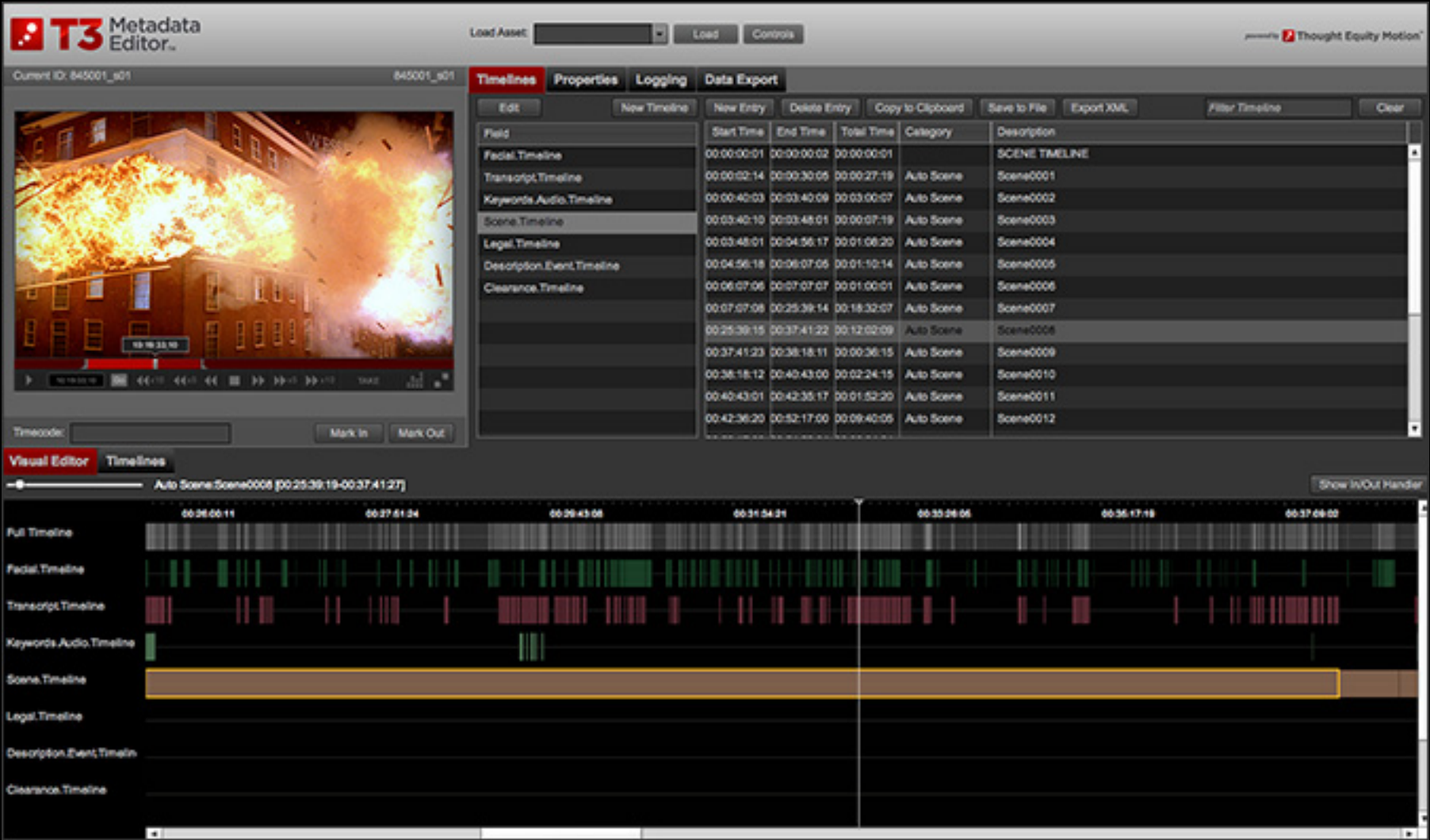
Virginia Tech clinched their third ACC title in four years and secured a berth in the Orange Bowl with a 44-33 win over Florida State. This game marked the 11th straight win by the Hokies after a rough 0-2 start to the season. Virginia Tech QB Tyrod Taylor (ACC Player of the Year) broke VT's single-season record for touchdown passes after completing 18 of 28 passes for 263 yards and 4 total touchdowns.

GEICO

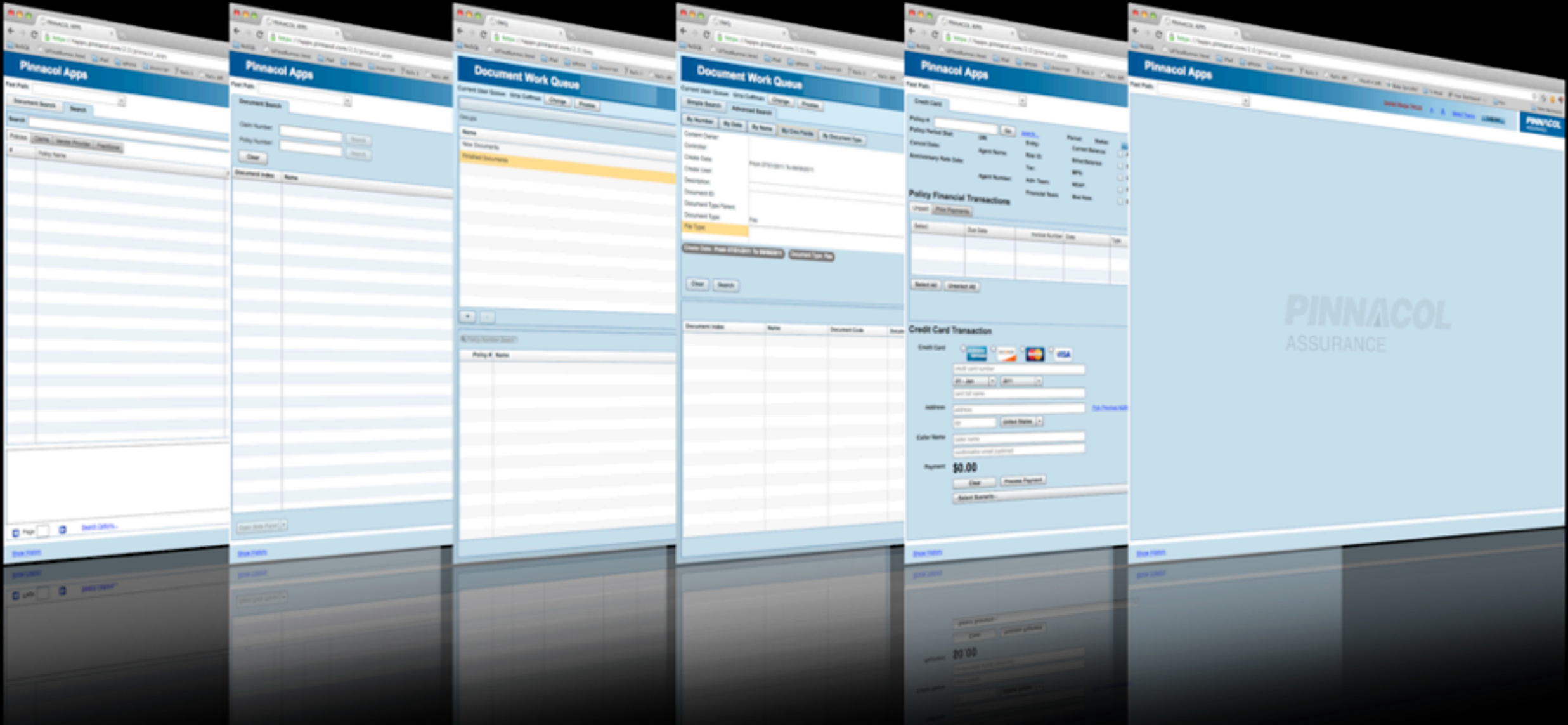
I'm here to save you money.

Click for a FREE car insurance quote.

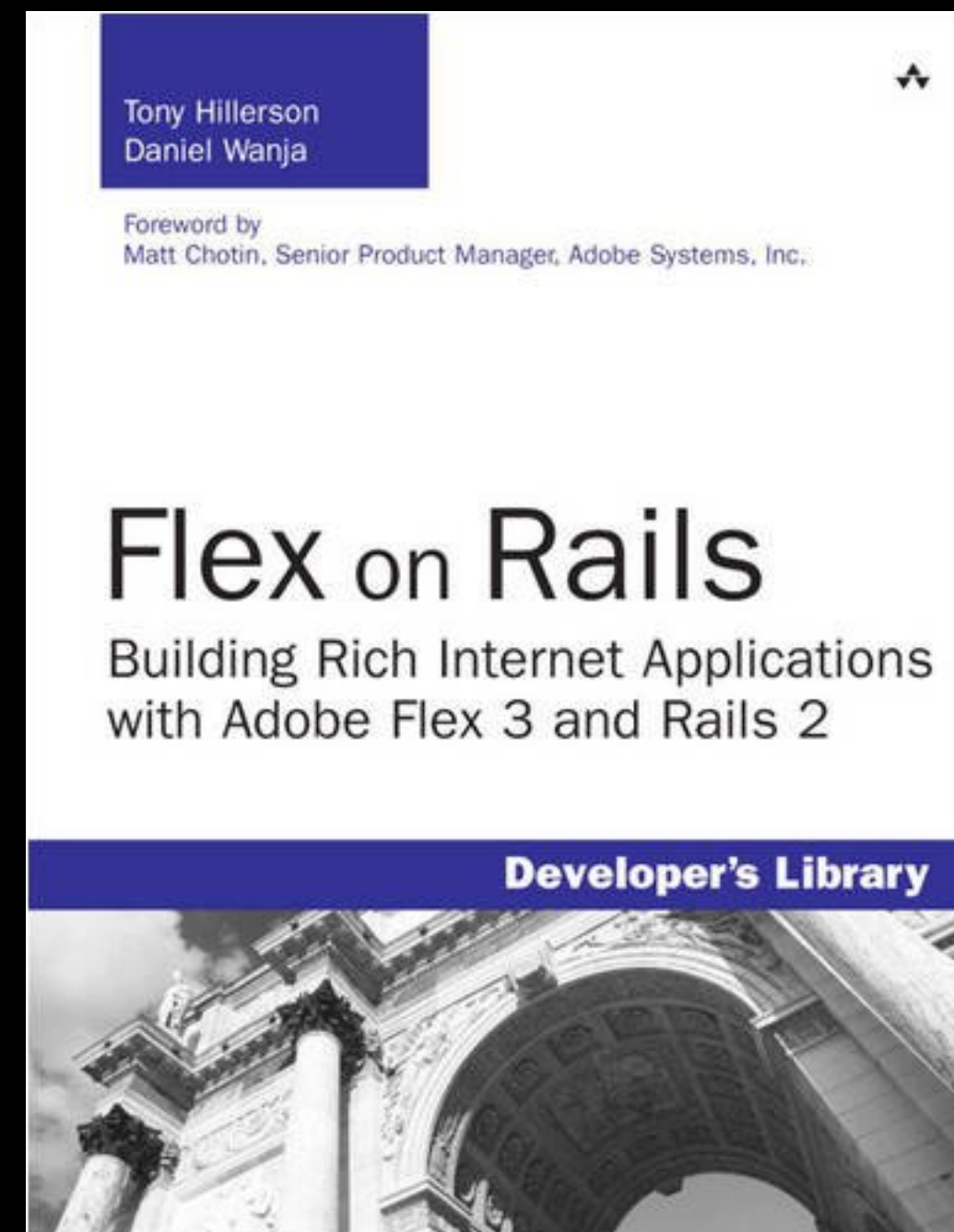
T3 Metadata Editor



Pinnacol -> Flex + Rails



And 1/2 a book...on what else than “Flex on Rails”

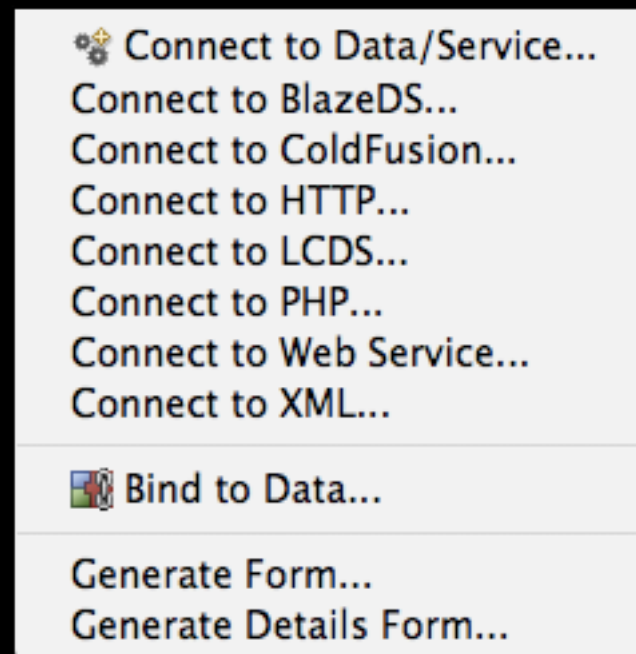


FLEX + RAILS

- Well Flex is cool, Rails is Cool...
- For the Rails developer: for advanced visualization, product configurator, admin tool, fast data entry, ...
- For the Flex developer: if you need to store data, secure some assets (i.e. S3), push notification, ...
- So let's check out how we can use them together
- Assumptions: you are a Flex developer...
- So I'll show a bit more of Rails

Different Options to Connect Flex to a Server

- **HTTPService: XML/JSON/AMF**
- Sockets/WebSockets
- URLLoader
- Lifecycle Data Services
- SOAP
- Flash Builder Data Menu



- We'll mainly focus on HTTPService tonight, as it's a good fit for Rails

RAILS

Create a Rails app

```
$ rails new demo1  
$ cd demo1  
$ rails generate scaffold post title:string body:text  
$ rake db:migrate  
$ rails s
```

<http://localhost:3000/posts>

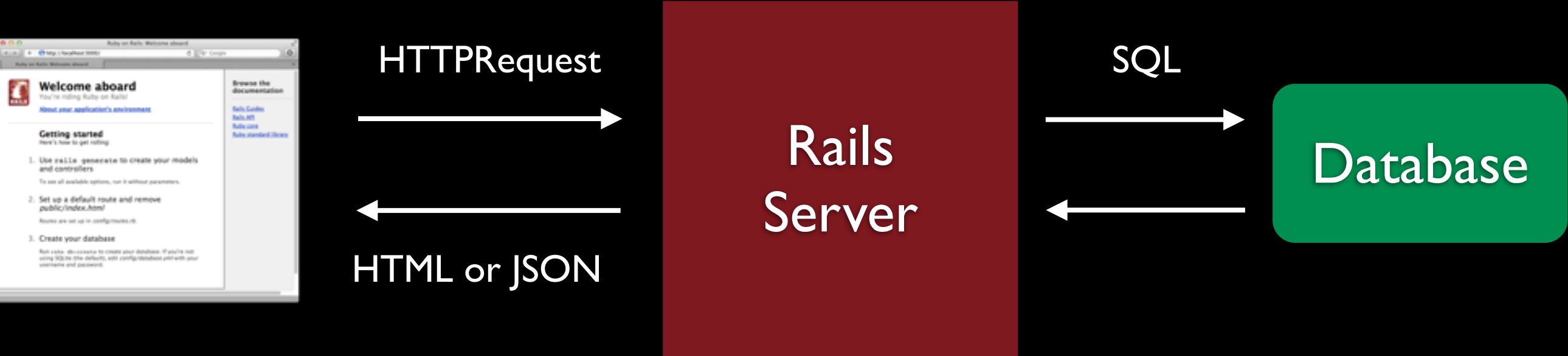
<http://localhost:3000/posts.json>

```
$ rails c
> Post.count
=> 1

> Post.first
=> #<Post id: 1, title: "Demo 1", body: "Getting started with Rails
in 5 minutes", created_at: "2011-09-11 17:17:36", updated_at:
"2011-09-11 17:17:36">

> Post.create(title: "Ruby Rocks!", body:"This is the content")
=> #<Post id: 2, title: "Ruby Rocks!", body: "This is the content",
created_at: "2011-09-11 20:34:43", updated_at: "2011-09-11 20:34:43">
```


RAILS OVERVIEW



ActionView

ActionController

ActionDispatch

ActiveSupport

ActionMailer

ActiveRecord / ActiveModel

ActionDispatch

1. Handle request

2. Routes matches URL

ActionController

3. Filters are invoked

4. Actions is called

ActionView

5. View is rendered

6. Response sent to client

Session deserialization
Browser Standards Mode
Cookie Abstraction Management
Deserialization of Content
Types like JSON, XML
Reloading
Handling IP Spoofing
Routing
Browser Caching (ETags, Last_Modified)
Content Negotiation (Mime)
Caching

ActionDispatch - under the hood

Request	Flash	Rescue	Routing	Parameter Filter
Response	Head	Show Exceptions	Cache	Filter Parameters
Best Standards Support	Params Parser	Static	Headers	Upload
Callbacks	Reloader	Closed Error	Mime Negotiation	Uploaded File
Cookies	Remote Ip	Middleware Stack	Parameters	URL

ActionController - under the hood

Caching	Data Streaming	Implicit Render	Renderers	Streaming	Action Controller Error
Metal	Force SSL	Instrumentation	Rendering	Testing	Unknown Http Method
Middleware	Head	Mime Responds	Request Forgery Protection	UrlFor	Render Error
Compatibility	Helpers	Params Wrapper	Rescue	UrlWriter	Routing Error
Conditional Get	Hide Actions	Rack Delegation	Responder	Routing	Method Not Allowed
Cookies	Http Authentication	Redirecting	Session Management	Record Identifier	Missing File

Descendants Tracker	Basic Object	Configurable	Message Encryptor	Rescuable
FileUpdate Checker	Benchmarkable	Deprecation	Message Verifier	Secure Random
Log Subscriber	Buffered Logger	Gzip	Multibyte	String Inquirer
Notifications	Cache	Inflector	Option Merger	Xml Mini
Backtrace Cleaner	Callbacks	JSON	Ordered Hash	Safe Buffer
Base64	Concern	Memoizable	Ordered Options	

REST API

Rails and Flex - Via REST



```
$ rake routes
```

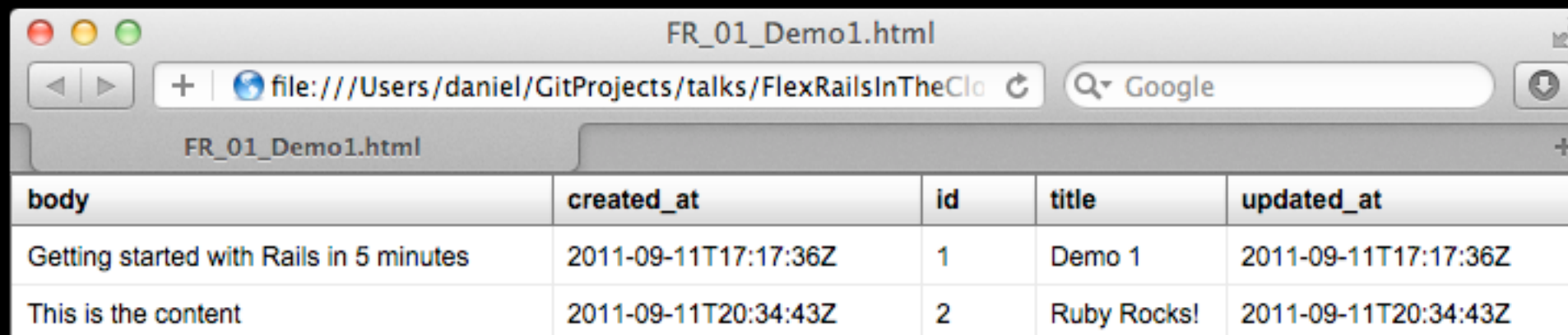
posts	GET	/posts(.:format)	{:action=>"index", :controller=>"posts"}
	POST	/posts(.:format)	{:action=>"create", :controller=>"posts"}
new_post	GET	/posts/new(.:format)	{:action=>"new", :controller=>"posts"}
edit_post	GET	/posts/:id/edit(.:format)	{:action=>"edit", :controller=>"posts"}
post	GET	/posts/:id(.:format)	{:action=>"show", :controller=>"posts"}
	PUT	/posts/:id(.:format)	{:action=>"update", :controller=>"posts"}
	DELETE	/posts/:id(.:format)	{:action=>"destroy", :controller=>"posts"}

HTTP SERVICE


```
<s:HTTPService id="service"
    url="http://localhost:3000/posts.json"
    resultFormat="text"
    result="service_resultHandler(event)"/>
```

```
service.send()
```

```
protected function service_resultHandler(event:ResultEvent):void
{
    var jsonData:Array = new JSONDecoder(event.result as String, true).getValue()
    list = new ArrayCollection(jsonData);
}
```



The screenshot shows a web browser window titled "FR_01_Demo1.html". The address bar shows the file path "file:///Users/daniel/GitProjects/talks/FlexRailsInTheClo". The browser displays a table with 5 columns: body, created_at, id, title, and updated_at. The table contains two rows of data.

body	created_at	id	title	updated_at
Getting started with Rails in 5 minutes	2011-09-11T17:17:36Z	1	Demo 1	2011-09-11T17:17:36Z
This is the content	2011-09-11T20:34:43Z	2	Ruby Rocks!	2011-09-11T20:34:43Z

- Index, Create, Update, Delete, Show -> That's 5 HTTPServices
- Create one class that wraps 5 HTTPServices for you
- Or use a Framework

ACTIVE RESOURCE

Flex ActiveResource Framework



- My try at a framework to use Rails
- Under active development
- Target 1.0 end of November
- REST, Nested Resources, Nested Attributes, BulkAPI support
- Custom Verbs (soon)

Flex ActiveResource Framework



- <https://github.com/danielwanja/activeresource>
- Move `active_resource.swc` to `/libs`
- No gem required on the Rails side


```
package model
{
    import active_resource.ActiveResource;

    dynamic public class Post extends ActiveResource
    {
        public function Post(attributes:Object=null)
        {
            super(attributes);
        }

        public var id:*;
        public var title:String;
        public var body:String;
        public var created_at:Date;
        public var updated_at:Date;

        /* static block */
        resource("posts", Post);
    }
}
```

```
// Find ALL
var call:AsyncToken = ActiveResource.findAll(Post)
call.addResponder(new AsyncResponder(resultHandler, faultHandler));

// Find One
var call:AsyncToken = ActiveResource.find(Post, 1)
call.addResponder(new AsyncResponder(resultHandler, faultHandler));
```

posts	mx.collections.ArrayCollection (@2b039701)
[0]	model.Post (@1d975ba1)
body	"the description..."
created_at	<Tue Oct 4 15:42:43 GMT-0700 2011> (@22151841)
id	1
title	"Rails rocks"
updated_at	<Tue Oct 4 15:42:43 GMT-0700 2011> (@22151821)
source	[] (@1fa0ee49)

id	title	body	created_at
1	Rails rocks	the description...	Tue Oct 4 15:42:43 GMT-0700 2011

```
// Create
var post:Post = new Post();
post.title = "Flex Rocks!";
post.body = "Getting started with Rails in 5 minutes";
var call:AsyncToken = post.save();
call.addResponder(new AsyncResponder(resultHandler, faultHandler));
```

```
// Update
post.body = "Getting started with Flex in 5 minutes";
var call:AsyncToken = post.save();
call.addResponder(new AsyncResponder(resultHandler, faultHandler));
```

```
// Delete
var call:AsyncToken = post.destroy();
call.addResponder(new AsyncResponder(resultHandler, faultHandler));
```

NESTED RESOURCES

- More complex apps: Post has_many comments

```
$ rails generate scaffold comments post_id:integer email:string  
title:string body:text
```

- Model

```
class Post < ActiveRecord::Base  
  has_many :comments  
end
```

```
class Comment < ActiveRecord::Base  
  belongs_to :post  
end
```

```
> Post.first.comments.create(email:"d@n-so.com", title:"This is great",  
body:"long description...")
```


Nested Resources - nested routes

- routes.rb

```
Demo2::Application.routes.draw do
  resources :posts do
    resources :comments
  end
end
```

- rake routes

post_comments	GET	/posts/:post_id/comments(:format)	{:action=>"index", :controller=>"comments"}
	POST	/posts/:post_id/comments(:format)	{:action=>"create", :controller=>"comments"}
new_post_comment	GET	/posts/:post_id/comments/new(:format)	{:action=>"new", :controller=>"comments"}
edit_post_comment	GET	/posts/:post_id/comments/:id/edit(:format)	{:action=>"edit", :controller=>"comments"}
post_comment	GET	/posts/:post_id/comments/:id(:format)	{:action=>"show", :controller=>"comments"}
	PUT	/posts/:post_id/comments/:id(:format)	{:action=>"update", :controller=>"comments"}
	DELETE	/posts/:post_id/comments/:id(:format)	{:action=>"destroy", :controller=>"comments"}

Nested Resources - constrain child controller by parent id

- constrain Comment controller by params[:post_id]

```
class CommentsController < ApplicationController
  before_filter :get_post

  def index
    @comments = @post.comments.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @comments }
    end
  end

  private

  def get_post
    @post = Post.find(params[:post_id])
  end
end
```

Nested Resources - Flex ActiveResource

- Comments are nested by given post

Create

```
comment.save({nestedBy:post})
```

```
[POST] "/posts/1/comments.json" // http.headers={X_HTTP_METHOD_OVERRIDE:'put'};
```

Update

```
comment.save({nestedBy:post})
```

```
[POST] "/posts/1/comments/3.json"
```

Delete

```
comment.destroy({nestedBy:post})
```

```
[POST] "/posts/1/comments/3.json" // http.headers={X_HTTP_METHOD_OVERRIDE:'delete'};
```

Index

```
ActiveResource.find(Comment, 1, {nestedBy:post})
```

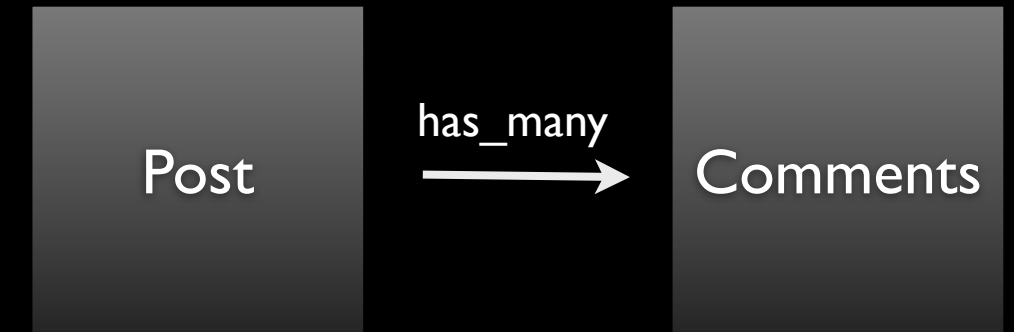
```
[GET] "/posts/1/comments.json"
```

NESTED ATTRIBUTES

- Another approach is nested attributes
- introduced with the advent of JavaScript
- Reduces HTTPService requests
- Update the Parent and the children in one request
- Can be deeply nested

- Enabled at Active Record Level

```
class Post < ActiveRecord::Base
  has_many :comments
  accepts_nested_attributes_for :comments, :allow_destroy => true
end
```



- HTTPService needs to send parent and children as nested attributes

```
params = { :post => {
  :id => 1, :title => "Demo 1", :body => "Getting started with Rails in 5 minutes",
  :comments_attributes => [
    { :id => 15, :title => 'This doesn't change' },
    { :id => 16, :title => 'This one is delete', :_destroy => '1' },
    { :title => 'This is a new one' }
  ]
}
```

- Controller doesn't change (as simple resource)

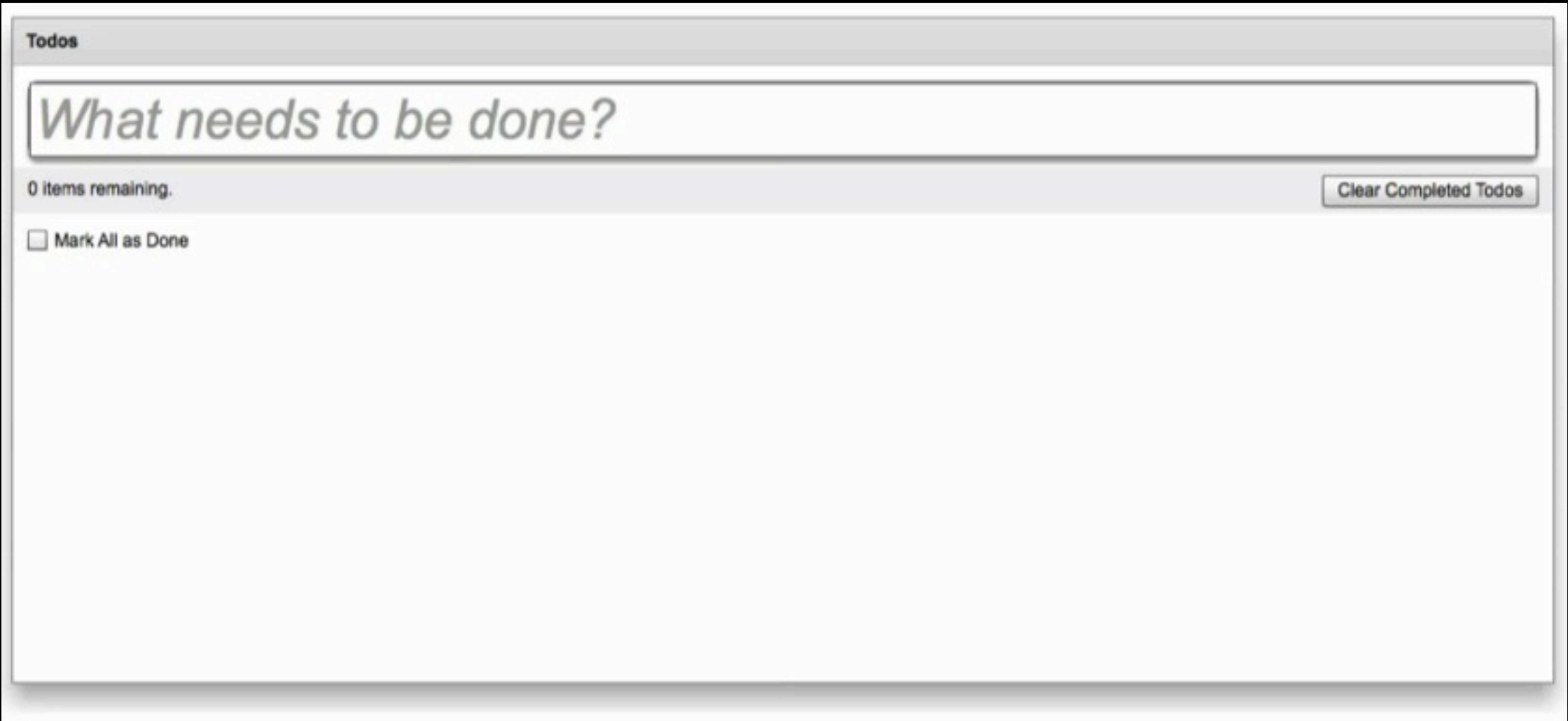
```
post.update(params[:post])
```

- Children are saved with the parent

```
post.comments.getItemAt(0).body = "changed comment"  
post.comments.getItemAt(1)._destroy = true  
post.comments.addItem(new Comment({title:"fancy", body:"Another description"}))  
  
post.save({nestedAttributes: ['comments']})
```

BULK API

- Rails: https://github.com/drogus/bulk_api
- Flex: https://github.com/danielwanja/bulk_data_source_flex
- Will certainly be merged into the Flex ActiveResource Framework



Create the Rails application

```
$ rails new todos
```

Edit the Gemfile and add

```
gem 'bulk_api'
```

Setup the bulk_api, add a todo resource, migrate the database and start the server

```
$ bundle install  
$ rails generate bulk:install  
$ rails g scaffold todo title:string done:boolean  
$ rake db:migrate  
$ rails s
```

Ok, we are done on the Rails side.


```
package resources
{
    import bulk_api.BulkResource;

    [RemoteClass(alias="Todo")]
    public dynamic class Todo extends BulkResource
    {
        public function Todo(attributes:Object=null) {
            super(attributes);
        }

        resource("todos", Todo)
    }
}
```

Create

```
var todo:Todo = new Todo({title:newTodo.text, done:false})
var call:AsyncToken = BulkResource.create(Todo, {todos:new ArrayCollection([todo])});
call.addResponder(new AsyncResponder(addTodoResult, faultHandler));

private function addTodoResult(event:ResultEvent, token:Object=null):void
{
    todos.addItem(event.result.todos.getItemAt(0));
    newTodo.text = "";
}
```

Update

```
var call:AsyncToken = BulkResource.update(Todo, {todos:new ArrayCollection(todos)});
```

Delete

```
var done:Array = [];
for each (var todo:Todo in todos) {
    if (todo.done) done.push(todo.id);
}
var call:AsyncToken = BulkResource.destroy(Todo, {todos:new ArrayCollection(done)});
```

ZLIB

```
require 'zlib'
File.open('data_to_compress.zlib', 'w') do |f|
  z = Zlib::Deflate.new(9) # 9 = Z_BEST_COMPRESSION
  f.write z.deflate(s.to_xml, Zlib::FINISH)
  z.close
end
```

Thanks to the @theaboutbox

<http://www.onrails.org/2007/11/27/flash-utils-bytearray-compressing-4-1mb-to-20k>

- http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/utils/CompressionAlgorithm.html

```
import flash.utils.ByteArray;

private function loadData():void {
    var loader:URLLoader = new URLLoader();
    loader.dataFormat = "binary";
    loader.addEventListener(Event.COMPLETE, completeHandler);
    var request:URLRequest = new URLRequest("data.zlib");
    loader.load(request);
}

private function completeHandler(event:Event):void {
    var loader:URLLoader = URLLoader(event.target);
    var ba:ByteArray = loader.data;
    ba.uncompress();
    var s:String = ba.toString();
    var xml:XML = new XML(s);
}
```

AMF

Ruby and AMF

- Rails: add RocketAMF and rails3-amf to Gemfile

```
gem 'RocketAMF', :git => 'git://github.com/warhammerkid/rocket-amf.git'  
gem 'rails3-amf', :git => 'git://github.com/warhammerkid/rails3-amf.git'
```

- Rails: in application.rb add the class mapping

```
module Demo4  
  class Application < Rails::Application  
    # ...  
    config.rails3amf.class_mapping do |m|  
      m.map :as => 'model.Post', :ruby => 'Post'  
      m.map :as => 'model.Comment', :ruby => 'Comment'  
    end  
  end  
end
```


Ruby and AMF

- Flex Declare the Value Objects

```
package model
{
    [RemoteClass(alias="model.Post")]
    public class Post
    {
        public var id:int;
        public var title:String;
        public var body:String;
        public var created_at:Date;
        public var updated_at:Date;
        public var comments:Array = [];
    }
}
```

```
package model
{
    [RemoteClass(alias="model.Comment")]
    public class Comment
    {
        public var id:int;
        public var title:String;
        public var body:String;
    }
}
```

- Rails: Add :amf support to the PostController

```
class PostsController < ApplicationController
  respond_to :html, :json, :amf

  def index
    @posts = Post.all
    respond_with(@posts) do |format|
      format.amf { render amf: @posts.to_amf(:include => [:comments]) }
    end
  end
end
```

- Flex Declare the Value Objects

```
<s:RemoteObject id="postsController"
    destination="ruby"
    source="PostsController"
    endpoint="http://localhost:3000/amf">
    <s:method name="index" result="indexResultHandler(event)" fault="faultHandler(event)" />
    <s:method name="create" result="createResultHandler(event)" fault="faultHandler(event)" />
    <!-- + update, delete, show, ... -->
</s:RemoteObject>
```

- Call the Rails controller via AMF

```
postsController.index.send();
```

- Typed Objects are Returned

```
protected function indexResultHandler(event:ResultEvent):void
{
    posts = new ArrayCollection(event.result as Array)
}
```

- All Posts are loaded including Comments

Create new Post

Posts

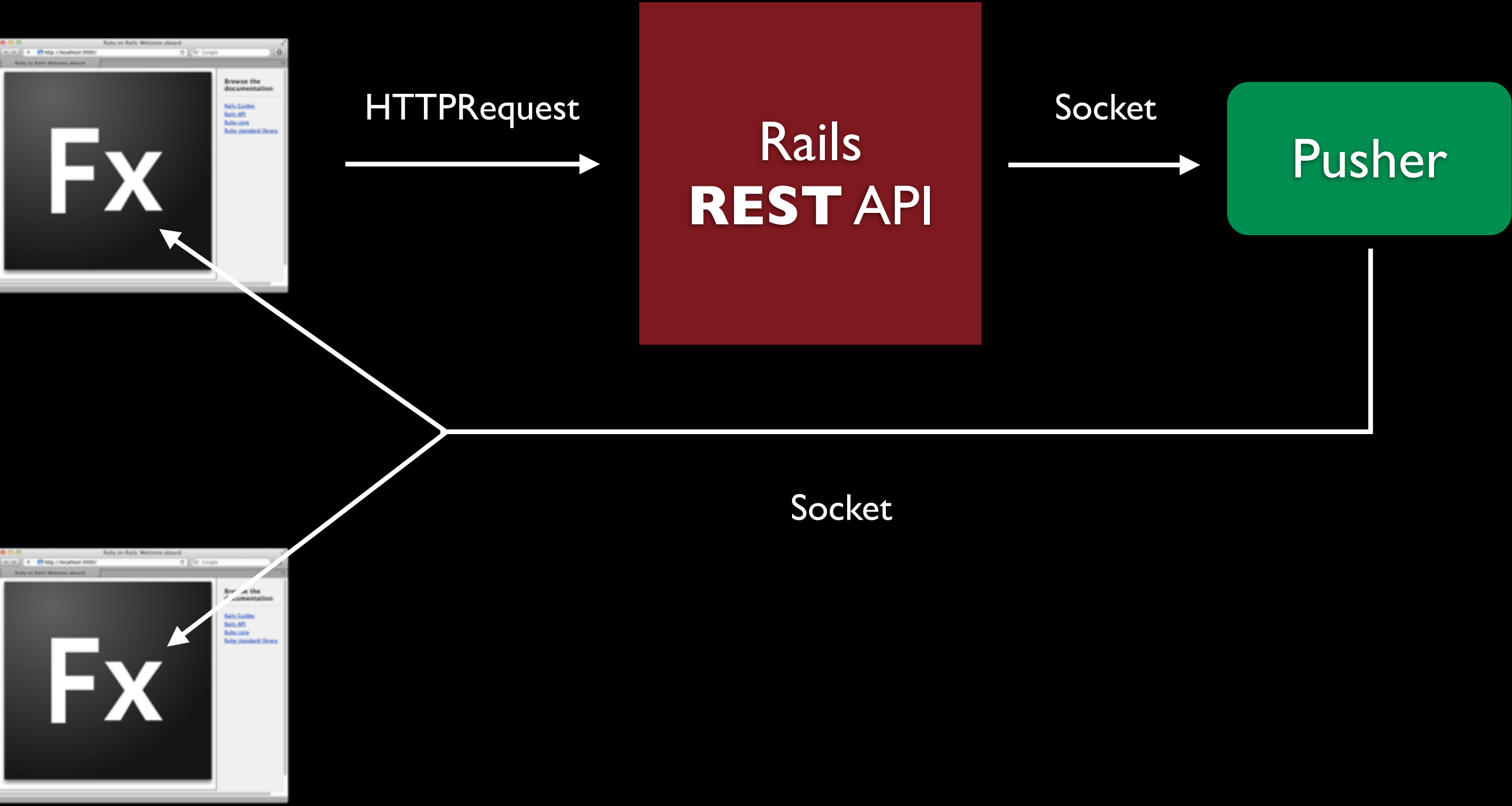
body	comments	created_at	id	test	title	updated_at
Getting started with Rails in 5 minutes	[object Object]	Sun Sep 11 11:17:36 GMT-0600 2011	1	tested	Demo 1	Sun Sep 11 11:17:36 GMT-0600 2011
This is the content		Sun Sep 11 14:34:43 GMT-0600 2011	2		Ruby Rocks!	Sun Sep 11 14:34:43 GMT-0600 2011

Comments

body	created_at	email	id	post_id	title	updated_at
long description...	Sun Sep 11 15:20:04 GMT-0600 2011	d@n-so.com	1	1	This is great	Sun Sep 11 15:20:04 GMT-0600 2011

PUSHER

WebSockets and Pusher



Publish/Subscribe uses

- messaging
- screen synchronization
- notification
- inbox/queue count update
- system wide alarms

- Add realtime features in minutes!
- Free sandbox (20 connections and 100,000 messages per day)
- \$49/month: 500 Max Connections + 1 millionMessages per day
- Why use pusher.com: easy setup
- Alternatives: <http://socket.io/>, pywebsocket, EM-WebSocket, ..

Pusher and Rails?

- Rails is not needed, can be useful (security, control)
- Can be Flex-->Pusher-->Flex (not secure)

Add Pusher to your Rails app

- Add the pusher Gem to your Gemfile

```
gem 'pusher'
```

- Signup at <http://pusher.com/>
- Get your pusher api key from the website
- config/environments/development.rb

```
require 'pusher'  
Pusher.app_id = 6182  
Pusher.key = 'a0b74a20a5d8df2db432'  
Pusher.secret = '057c1aac0e15defa3a55'
```

```
$ rails g controller Message hello
```

- Implement a notification method

```
class MessageController < ApplicationController

  def hello
    Pusher['test_channel'].trigger('greet', {
      :greeting => "Hello there!"
    })
  end

end
```

- <https://github.com/smakinson/Pusher-ActionScript-Library> ("not quite ready for use")
- <https://github.com/y8/websocket-as>
- Implement listener method

```
private var pusher:Pusher;
private var channel:Channel;
protected function setup():void
{
    pusher = new Pusher('a0b74a20a5d8df2db432', "pusherexample" );
    channel = pusher.subscribe('test_channel');
    channel.bind('greet', gotData);
}

protected function gotData(data:Object):void {
    Alert.show(ObjectUtil.toString(data));
}
```

GOTCHA'S

- authentication_token
- crossdomain.xml
- error_handling

THE CLOUD

- Many provider
- Amazon -> Roll you own
- Engineyard -> Professional help
- **Heroku** -> Takes care of tons for you
- CloudFoundry
- ...

HEROKU

- Install Heroku (as a gem)

```
$ gem install heroku  
$ heroku create myapp --stack cedar
```

- Create a Rails app

```
$ rails new myapp  
$ cd myapp  
$ rails generate scaffold post title:string body:text
```

- In Gemfile add Postgress to Production and use sqlite only in development

```
group :production do  
  gem 'pg'  
end  
  
group :development do  
  gem 'sqlite3'  
end
```

- Add code the first time

```
$ git init  
$ git add .  
$ git commit -m "Initial commit"  
$ git remote add heroku git@heroku.com:myapp.git  
$ git push heroku master  
$ heroku run rake db:migrate
```

git push deploys the app!

- Add code subsequent time

```
$ git add .  
$ git commit -m "Subsequent commit"  
$ git push heroku master
```

- The App is in the Cloud

<http://myapp.herokuapp.com/posts>

INSTALL

Installing Rails

- OSX: \$ gem install rails
- Windows: RAILSINSTALLER (<http://railsinstaller.org/>)

LINKS

These slides

- <https://github.com/danielwanja/talks/tree/master/FlexWithRubyOnRails/slides>

ActiveResource

- <https://github.com/danielwanja/activeresource>

Bulk API

- https://github.com/drogus/bulk_api
- https://github.com/danielwanja/bulk_data_source_flex

RestfulX

- <http://restfulx.github.com/>

Rocket/Rails AMF

- <https://github.com/warhammerkid/rocket-amf>
- <https://github.com/warhammerkid/rails3-amf>
- <https://github.com/rubyamf/rocketamf>

Pusher

- <http://pusher.com>
- <https://github.com/smakinson/Pusher-ActionScript-Library>

Heroku

- <http://heroku.com>

Q&A

