The Future of Web Services

with Gregg Pollack
Orlando Ruby Users Group
December 13, 2007
The Big Problem with Web Services
What is REST really?
Why now?
How is Rails RESTful?
Gratuitous Code Demo
What Future of REST?
Human

Computer
API
(Application Programming Interface)
API
(Application Programming Interface)

Computer

Geocoding

Payment Gateway

Shipping
API
(Application Programming Interface)

Computer

Geocoding

Payment Gateway

Shipping
Computer to computer communication isn’t standardized.
What is REST?
Representational State Transfer (REST) was published by Roy Fielding in 2000 in his dissertation entitled "Architectural Styles and the Design of Network-based Software Architectures."
REST

Representational State Transfer

Roy Fielding published in 2000 in his dissertation entitled

“Architectural Styles and the Design of Network-based Software Architectures”

REST defined as

A style of software architecture for distributed hypermedia systems
REST
What makes up a RESTful software architecture?
What makes up a RESTful software architecture?

1. Resources (nouns)
What makes up a RESTful software architecture?

1. Resources (nouns)

2. All resources are addressable
What makes up a RESTful software architecture?

1. Resources (nouns)

2. All resources are addressable

3. Standard methods of interaction (verbs)
What makes up a RESTful software architecture?

1. Resources (nouns)

2. All resources are addressable

3. Standard methods of interaction (verbs)

4. Protocol which is:
   - Client / Server
   - Stateless
   - Cacheable
   - Layered
WWW
HTTP Protocol
Resources

Addressable

URI
(Uniform Resource Identifier)

HTTP Protocol
Resources

Addressable

URI
(Uniform Resource Identifier)

POST
GET
PUT
DELETE

HTTP Protocol

WWW
Resources

Addressable

URI
(Uniform Resource Identifier)

POST
GET
PUT
DELETE

HTTP Protocol

Browser / Server

WWW
Resources

Addressable

URI
(Uniform Resource Identifier)

POST
GET
PUT
DELETE

Browser / Server

HTTP Protocol

Stateless
What does REST have to do with Web Services?
What makes up a RESTful software architecture?

1. Resources (nouns)

2. All resources are addressable

3. A uniform interface of interaction (verbs)

4. Protocol which is:
   - Client / Server
   - Stateless
   - Cacheable
   - Layered
Why REST now?
Why REST now?
Why REST now?
Why **REST** now?

class HackingProgram

    def start_hacking
        ....
        end

    def list_games
        ....
        end

    def start_game(name)
        ....
        end

end

class WOPR

    def hello_professor_falken
        ....
        end

    def chess
        ....
        end

    def global_thermonuclear_war
        ....
        end

eend
RPC

(Remote Procedure Call)
RPC
(Remote Procedure Call)

XML

XML-RPC
RPC
(Remote Procedure Call)

XML

XML-RPC

SOAP
REST

(methods / verbs)

POST
GET
PUT
DELETE
How did Rails become RESTful?
How did **Rails** become **RESTful**?

**MVC**

- **Model** - Database tables & Domain Logic
- **View** - Display Logic (html, xml, or other)
- **Controller** - User Action Logic
A Rails Controller

class WOPR < ApplicationController

  # http://www.example.com/WOPR/hello_professor_falken
def hello_professor_falken
    ....
  end

  # http://www.example.com/WOPR/chess
def chess
    ....
  end

  # http://www.example.com/WOPR/global_thermonuclear_war
def global_thermonuclear_war
    ....
  end

end
Rails Best Practices

1. Keep your controllers as small as possible
Rails Best Practices

1. Keep your controllers as small as possible

2. Only use a single controller for each model
class Users < ApplicationController

  # GET /users/show/3
  def show
    ....
  end

  # POST /users/create
  def create
    ....
  end

  # POST /users/update/3
  def update
    ....
  end

  # POST /users/destroy/3
  def destroy
    ....
  end

end
<table>
<thead>
<tr>
<th>Actions</th>
<th>show</th>
<th>create</th>
<th>update</th>
<th>destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The actions listed are: show, create, update, destroy.
<table>
<thead>
<tr>
<th>Actions</th>
<th>show</th>
<th>create</th>
<th>update</th>
<th>destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>select</td>
<td>create</td>
<td>update</td>
<td>delete</td>
</tr>
<tr>
<td>Actions</td>
<td>show</td>
<td>create</td>
<td>update</td>
<td>destroy</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>SQL</td>
<td>select</td>
<td>create</td>
<td>update</td>
<td>delete</td>
</tr>
<tr>
<td>REST</td>
<td>get</td>
<td>post</td>
<td>put</td>
<td>delete</td>
</tr>
</tbody>
</table>
## Methods / Actions

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/users/show/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users/create</td>
</tr>
<tr>
<td>POST</td>
<td>/users/update/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users/destroy/3</td>
</tr>
</tbody>
</table>
### Methods / Actions

<table>
<thead>
<tr>
<th>Verb</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/users/show/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users/create</td>
</tr>
<tr>
<td>POST</td>
<td>/users/update/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users/destroy/3</td>
</tr>
<tr>
<td>Verb</td>
<td>Noun</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>GET</td>
<td>/users/show/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users/create</td>
</tr>
<tr>
<td>PUT</td>
<td>/users/update/3</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users/destroy/3</td>
</tr>
</tbody>
</table>
## Methods / Actions

<table>
<thead>
<tr>
<th>Verb</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/users/3</td>
</tr>
<tr>
<td>POST</td>
<td>/users</td>
</tr>
<tr>
<td>PUT</td>
<td>/users/3</td>
</tr>
<tr>
<td>DELETE</td>
<td>/users/3</td>
</tr>
</tbody>
</table>
class Users < ApplicationController

    # GET /users/show/3
    def show
        ....
    end

    # POST /users/create
    def create
        ....
    end

    # POST /users/update/3
    def update
        ....
    end

    # POST /users/destroy/3
    def destroy
        ....
    end

end
class Users < ApplicationController

  # GET /users/3
  def show
      ....
  end

  # POST /users/
  def create
      ....
  end

  # PUT /users/3
  def update
      ....
  end

  # DELETE /users/3
  def destroy
      ....
  end

end
In Rails

REST isn’t just how we code our web services
In Rails

REST isn’t just how we code our web services

It’s how we design our applications
Show me the code

(insert gratuitous code demo)
Browsers can’t do PUT and DELETE (YET)
You can even do nested resources!

/users/3/photos

/posts/5/comments

/users/4/posts/2/comments
Resource Based URLs are Search Engine Optimized

/book/agile-web-development
/planes/boeing-747
/author/gregg-pollack
play_global_thermonuclear_war
play_global_thermonuclear_war
process_credit_card_transaction
process_credit_card_transaction

POST credit_card_transaction
subscribe_to_mailing_list
subscribe_to_mailing_list

POST

Subscription
logout

Verb

Noun
REST forces you to create better code
The Future
New RESTful Software Architectures
Resource Oriented Architecture

RESTful Web Services
by Leonard Richardson & Sam Ruby
Open Social

RESTful software architecture for building social web application APIs
Gregg Pollack
Gregg@RailsEnvy.com
http://www.RailsEnvy.com

Additional Reading

RESTful Web Services
by Leonard Richardson & Sam Ruby

Agile Web Development with Rails
by Dave Thomas and David Heinemeier Hansson