

The future of Ruby on Rails : REST

thegiive in Taiwan OSDC 2007

Who am I ?

- thegiive Chen
- A Ruby on Rails Guy (寶石眾?)
- Ruby on Rails consultant of SocialPicks.com
- My Blog : <http://lightyror.thegiive.net/>
- My Email : thegiive at gmail.com

Outline

- What is REST ?
- Why REST ?
- REST on Rails
- How to REST ?
- Q & A
- take a REST

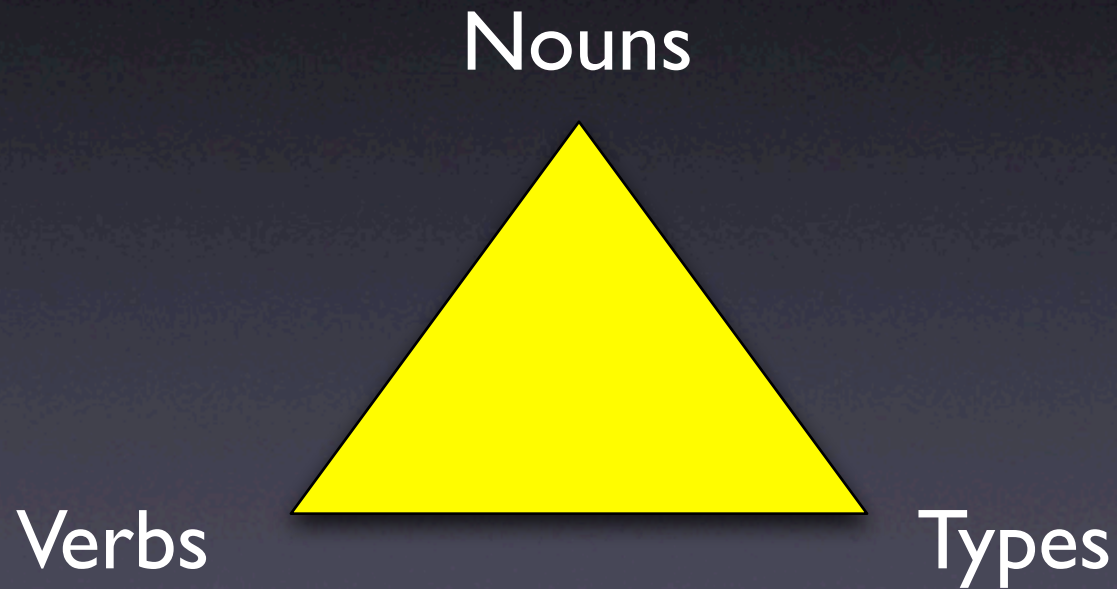
What is REST?

- REpresentational State Transfer
- Presented by Roy Fielding
 - 《Architectural Styles and the Design of Network-based Software Architectures》

Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

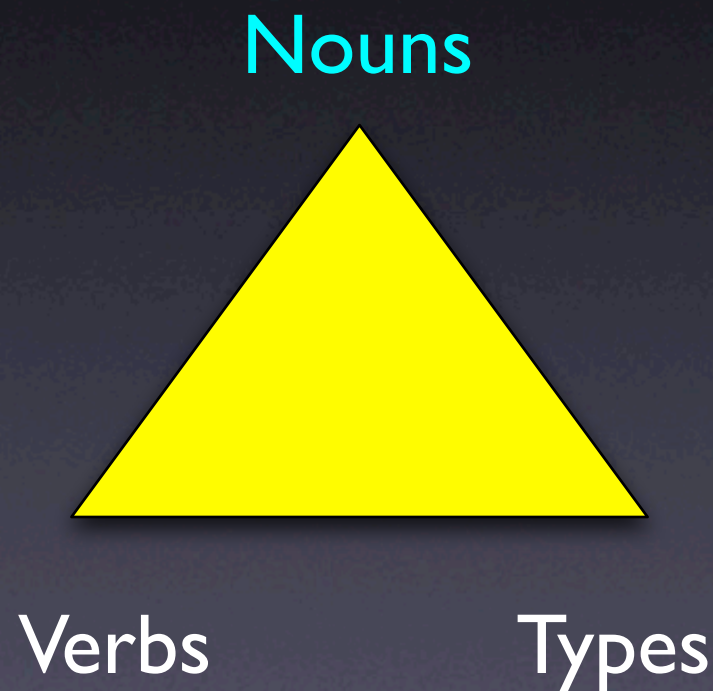
哩析勒供蝦？

REST is about Resource



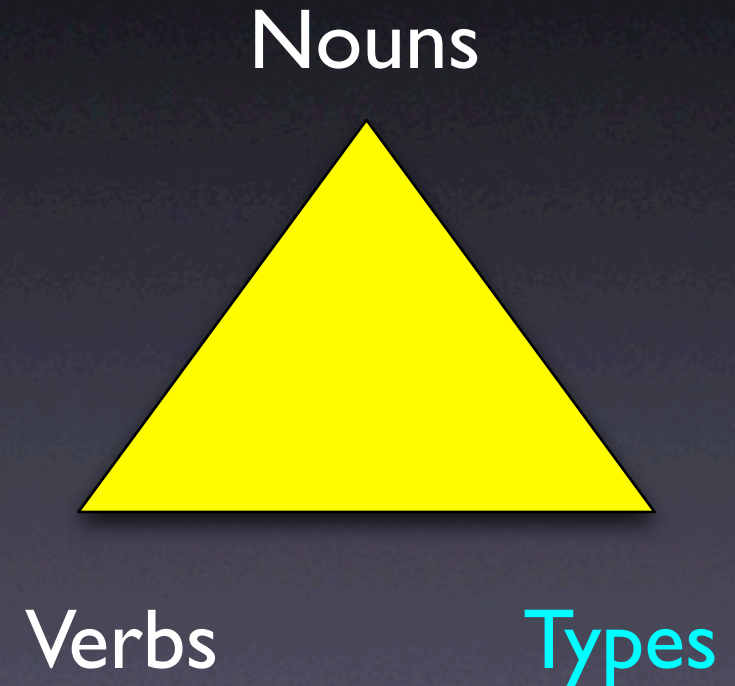
Nouns

- A unique location to present the resource
- Unconstrained
- URI
 - <http://www.abc.com/>



Content Types

- Constrained
- Format of the resource
 - XML, HTML, JPG... etc

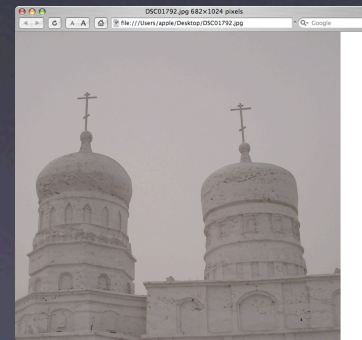


```
<xml>
<place>東北太陽島</place>
<date>20060304</date>
<size>1024x768</size>
</xml>
```

<http://abc.com/photo/xml/123/>



A Resource



<http://abc.com/photo/jpg/123/>

<http://abc.com/photo/html/123/>

Something wrong?

- It is the same resource
- Use **three URI** to describe three format
 - <http://abc.com/photo/xml/123/>
 - <http://abc.com/photo/html/123/>
 - <http://abc.com/photo/jpg/123/>

```
<xml>
<place>東北太陽島</place>
<date>20060304</date>
<size>1024x768</size>
</xml>
```

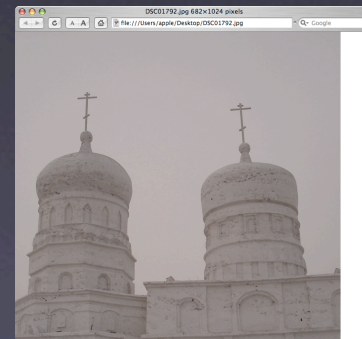
<http://abc.com/123>



A Resource



<http://abc.com/123>



<http://abc.com/123>

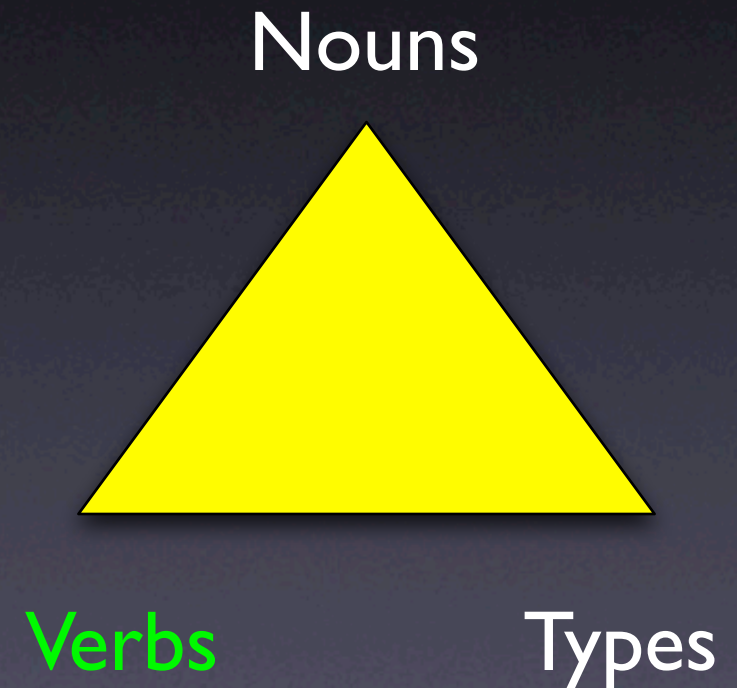


How to do that?

- Sub-file name
 - <http://abc.com/photo/123.xml>
- HTTP Request Header
 - In browser, request header is */*
 - Just create a request like **Accept: text/xml**

Verbs

- Constrained
- HTTP 1.1 has four type of request methods
 - GET, POST, PUT, DELETE
- RFC2616



CRUD

- A resource <http://abc.com/user/>
 - Show : <http://abc.com/user/show/>
 - Update : <http://abc.com/user/update/>
 - Delete : <http://abc.com/user/destroy/>

Something wrong?

- We operator the same resource
- Use different URI to operator resource

HTTP Verb	REST-URL	Action	No REST URL
GET	/users/1	show	GET /users/show/1
DELETE	/users/1	destroy	GET /users/destroy/1
PUT	/users/1	update	POST /users/update/1
POST	/users	create	POST /users/create

REST is ?

- An Architectural Style
- REST is not a standard
 - No W3C REST Specification
 - No REST developer's toolkits

REST **use** standards

- HTTP
- URL
- Resource Representation Standard
 - XML / HTML / GIF / JPEG
- MIME Type Standard
 - text/xml, text/html, image/gif, image/jpeg

Why REST ?

- Browser Support
- Clear application design
- Different Response format
- Cache
- Stateless

DRY

- I want to show the User id number 1
 - Use HTTP **GET** method
 - Use URI <http://abc.com/user/show/1>
- GET and Show is the same
- DELETE and Destroy is too

Simple

- Web programmer know GET and POST
- No more web service spec
- Amazon and Ebay's web service report
 - 60% customer use REST

HTTP is designed for
REST.

RoR 1.2 : REST on Rails

- scaffold_resource script
- Model
- Controller
- View
- **Route**
- RESTful Client : ActiveSupport

scaffold_resource

- RESTful scaffold generate script
- `./script/generate scaffold_resource`
- generate Model, View, Controller, Routing

Model

- Single table is easy to REST
- Relationship has unique Model
 - `has_many :through`

```
class Group < ActiveRecord::Base
  has_many :memberships
  has_many :users, :through => :memberships
end

class User < ActiveRecord::Base
  has_many :memberships
  has_many :groups, :through => :memberships
end

class Membership < ActiveRecord::Base
  belongs_to :group
  belongs_to :user
end

class MembershipsController < ActionController::Base
  # POST /memberships?group_id=1&user_id=2
  def create() end

  # DELETE /memberships/3
  def destroy() end
end
```

Code from iHower's Blog

Controller

- Easy to REST
 - respond_to
 - URL-Methods

respond_to

- Single resource and different format

```
respond_to do |format|  
  format.html { }  
  format.xml { render :xml => @user.to_xml }  
end
```

default format

```
respond_to do |wants|  
  wants.text  
  wants.html  
  wants.js  
  wants.ics  
  wants.xml  
  wants.rss  
  wants.atom  
  wants.yaml  
end
```

URL Method in Controller

UserController

ControllerName_url(id) → <http://abc.com/user/>

ControllerNameS_url → <http://abc.com/users/>

```
redirect_to :controller => "users",  
            :action => "show",  
            :id => @user.id
```



```
redirect_to user_url(@user)
```

URL Method in View

- There is a lot of View helpers to REST

`link_to "New", new_user_path => New`

`link_to "Edit", edit_user_path(@user) => Edit`

How to Delete and PUT?

- Browser don't support delete and put
- Use **POST** and **Javascript** to implement

```
form_for(:project, :url => user_path(@user),  
  :html => { :method => :put }) do |f| ..
```



```
<form action="/users/1" method="post">  
  <div style="margin:0;padding:0">  
<input name="_method" type="hidden" value="put" />  
  </div>
```

```
link_to "Destroy", user_path(@user),  
      :method => :delete
```



```
<a href="/users/1"  
  onclick="var f = document.createElement('form');  
  f.style.display = 'none'; this.parentNode.appendChild(f);  
  f.method = 'POST'; f.action = this.href;  
  var m = document.createElement('input');  
  m.setAttribute('type', 'hidden');  
  m.setAttribute('name', '_method');  
  m.setAttribute('value', 'delete'); f.appendChild(m); f.submit();  
  return false;">Destroy</a>
```

View Method	HTTP Verb	Path	Action
users_path	GET	/users	index
user_path(l)	GET	/users/l	show
new_user_path	GET	/users/new	new
edit_user_path(l)	GET	/users/l;edit	edit
users_path	POST	/users	create
users_path(l)	PUT	/users/l	update
users_path(l)	DELETE	/users/l	destroy

Routing

- `respond_to` format
- HTTP request method config

Respond_to Format

- sub-file name
 - <http://abc.com/user/1.xml>
- HTTP Request Header
 - Accept field

```
curl -H "Accept: application/xml" -i -X GET  
http://abc.com/users/1
```



```
HTTP/1.1 200 OK  
Connection: close  
Date: Sat, 30 Dec 2006 17:31:50 GMT  
Set-Cookie:  
_session_id=4545eabd9d1bebde367ecbadf015bcc2;  
path=/  
Status: 200 OK  
Cache-Control: no-cache  
Server: Mongrel 0.3.13.4  
Content-Type: application/xml; charset=utf-8  
Content-Length: 160
```

How to config routing?

- `map.resources :users`
- `map.resources :sprints, :controller =>`
`"ontrack", :path_prefix => "/"`
`ontrack/:project_id", :name_prefix =>`
`"ontrack_"`

Want more?

```
map.resources :users do |users|  
  users.resources :blogs  
end
```



<http://abc.com/users/1/blogs>

Active Resource

- REST Client
- For development REST-Client Web Service
- Operation Resource as ActiveRecord
- Use HTTP four method and XML

Demo

How to REST?

Q & A

Take a **REST!!!**

This Slice is
license by **Attribution-NonCommercial-
ShareAlike 2.5 Taiwan**