Web Apps that Mesh

developing organic, universal network apps with
Ruby on Rails

What is REST?

REST is

an Architectural Style

REST is

a Universal API

REST !== HTTP+XML

with HTTP,
every request is already
an operation on a
URL-addressed
resource

HTTP is a good fit for RESTful development

web developers rejoice

Loose Coupling

we already know how it works

the API docs won’t be a tome

REST requires a certain disconnect between the interface
(which is REST-oriented)
and the implementation
(which is usually object-oriented).

- Paul Prescod

redefine your interface
in terms of
standard URL
operations instead of
method calls

HTTP request
becomes a
method call

URI == noun

one URI per object

HTTP method == verb

all the benefits of HTTP

- ubiquitous standard
- stable, linkable URL’s
- caching
- proxying (security & load-balancing)
- content-based filtering

The Downside

no native callbacks

synchronous only

HTTP rides on TCP/IP

unpredictable timing

Universal Access

external, public-facing, open applications benefit most

RESTful CRUD
HTTP methods

create == POST	GET /people/new then POST /people/create
read == GET	GET /people/8 or GET /people/8;edit
update == PUT	PUT /people/8
delete == DELETE	DELETE /people/8

;term

a verb or modifier

;edit

editable representation of the resource

;complete

commit a complex, multi-faceted transaction

today's web browsers
GET & POST only

pollute the URI's with verbs

strict	compatible
PUT /people/8	POST /person/8?_method=PUT
DELETE /people/8	POST /person/8?_method=DELETE

authentication

like all auth in Rails, it's up to you

HTTP is stateless

REST is stateless

sessions are stateful

require an authentication request
to establish context

HTTP provides

authentication headers

example: *simple_http_auth*

data format?

there is no
REST-standard
data format

Simple Data

REST is more than Rails

what about others that are not at our convention?

respond appropriately

example:

ActionController#respond_to

instead of creating your own XML, try

- RSS - evolving lists & notifications
- JSON - client-side manipulation
- YAML - human & machine friendly plain text
- XHTML - embedded data

focus on the majority case

XHTML

XHTML can facilitate
both human & machine use

beauty of structural truth

find the wizardry
in mark-up

XHTML is easier

- to understand; fewer exceptions
- to style; strict mode rendering
- to process; valid XML data

make the app work

then add client-side behaviors & style

spiders & other server-side consumers will not run scripts, so don't hide your content in there

semantic mark-up

tags express meaning,
not just layout

but "meaning" is
a relative term

can we have some standards for semantics?

yes, Micro Formats

standards for semantic mark-up

right now it's geeky; bridge that gap

XOXO: lists everywhere

enumerable containers

hCard: it’s a social network

the contact page becomes a vCard

XFN: an open social networking mechanism

critical mass replaces proprietary social sites build the ultimate view

Piles of Partials vs. Monolithic Beasts

Many Blocks vs. Big Blob

encapsulate partials with a selectable parent

<div>

meaningful <div>'s
support rich CSS design

encapsulate major blocks with multiple nested <div>'s

now that’s a joyful,
background-layering,
CSS playground

empower the stylers
& scripters

logical, predictable
classes & id's

example tool: ~~*dom_id.rb*~~ *simply_helpful* simplify application reuse

support themes

simple, semantic views make skinning a joy

example: *theme_support_plugin*

Loose Coupling

we already know how it works

the API docs won’t be a tome

Universal Access

external, public-facing, open applications benefit most

Simple Data

REST is more than Rails

everyone is not at our convention

Micro Formats

standards for semantic mark-up

right now it's geeky; bridge that gap

presented by Mars

http://marsorange.com/