

]

OpenResty: Your Relational Cloud

OpenResty: Your *Relational* Cloud

☺ *agentzh@yahoo.cn* ☺

agentzh (章亦春)

2008.9

☺ Why OpenResty?

agentzh's @boss have *strong opinions...*



laser: I want a web OS for pure JavaScript sites!

ting: I want a thin web *interface* for our **Pg cluster**!

agentzh: ...

OpenResty was *born* since then...

OpenResty's *birthday*,
as seen on the *#eeee* IRC channel...

(2007-11-17 01:59:33 PM)agentzh: just created a project for **OpenAPI** on openfoundry :)

(2007-11-17 02:07:09 PM)mbot: NOTICE: agentzh start to commit to /export/m1/svn/hack

(2007-11-17 02:07:10 PM)mbot: NOTICE: *created* the **openapi** project

(2007-11-17 02:08:44 PM)mbot: NOTICE: [**openapi**] initial checkin

☺ Originally *Inspired* by Jifty's REST plugin

The screenshot shows a Mozilla Firefox browser window with the title "Jifty::Plugin::REST - search.cpan.org - Mozilla Firefox". The address bar shows the URL "http://search.cpan.org/~sartak/Jifty-0.80408". The page displays the CPAN logo and a navigation menu with links: Home, Authors, Recent, News, Mirrors, FAQ, and Feedback. A search bar is present with the text "in All" and a "CPAN Search" button. The main content area shows the search results for "Shawn M Moore > Jifty-0.80408 > Jifty::Plugin::REST" with a "permlink" link. Below this, it says "Module Version: 1.01" with a "Source" link. The page is organized into sections: "NAME" (Jifty::Plugin::REST), "DESCRIPTION" (A RESTian web services API for your Jifty app.), and "USAGE" (Add the following to your site_config.yml). On the right side, there is a "Download:" section with a link to "Jifty-0.80408.tar.gz", a "Dependencies" link, and an "Annotate this POD" link. At the bottom right, there is a "CPAN RT" section with a table showing "New 7" and "Open 1", and links for "View Bugs" and "Report a bug". The status bar at the bottom shows "Done".

Jifty::Plugin::REST - search.cpan.org - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://search.cpan.org/~sartak/Jifty-0.80408

CPAN

Home Authors Recent News Mirrors FAQ Feedback

in All CPAN Search

Shawn M Moore > Jifty-0.80408 > Jifty::Plugin::REST [permlink](#)

Module Version: 1.01 [Source](#)

NAME
DESCRIPTION
USAGE

NAME ↑

Jifty::Plugin::REST

DESCRIPTION ↑

A RESTian web services API for your Jifty app.

USAGE ↑

Add the following to your site_config.yml

Download:
[Jifty-0.80408.tar.gz](#)

[Dependencies](#)

[Annotate this POD](#)

CPAN RT

New	7
Open	1

[View Bugs](#)
[Report a bug](#)

Done

...but have a *BIGGER* ambition.

(06:31:53 PM)ting: ft.. 我怎么感觉我们在设计的是一个**网络操作系统**。。...赫赫。。

(06:40:36 PM)agentzh: oh

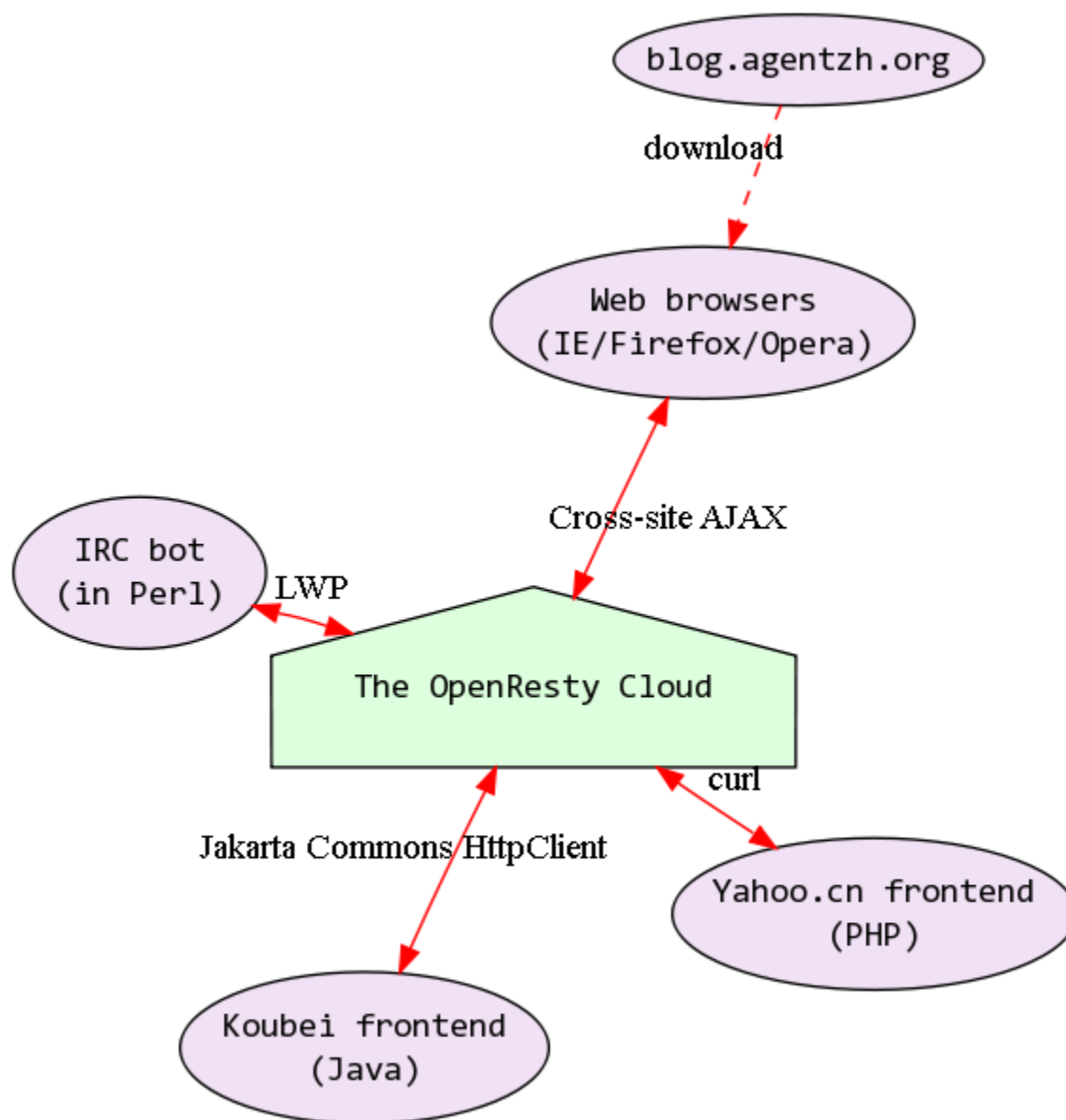
(06:40:42 PM)agentzh: laser's *vision*?

(06:40:44 PM)agentzh: *grin*

😊 So **What** is *OpenResty* then?

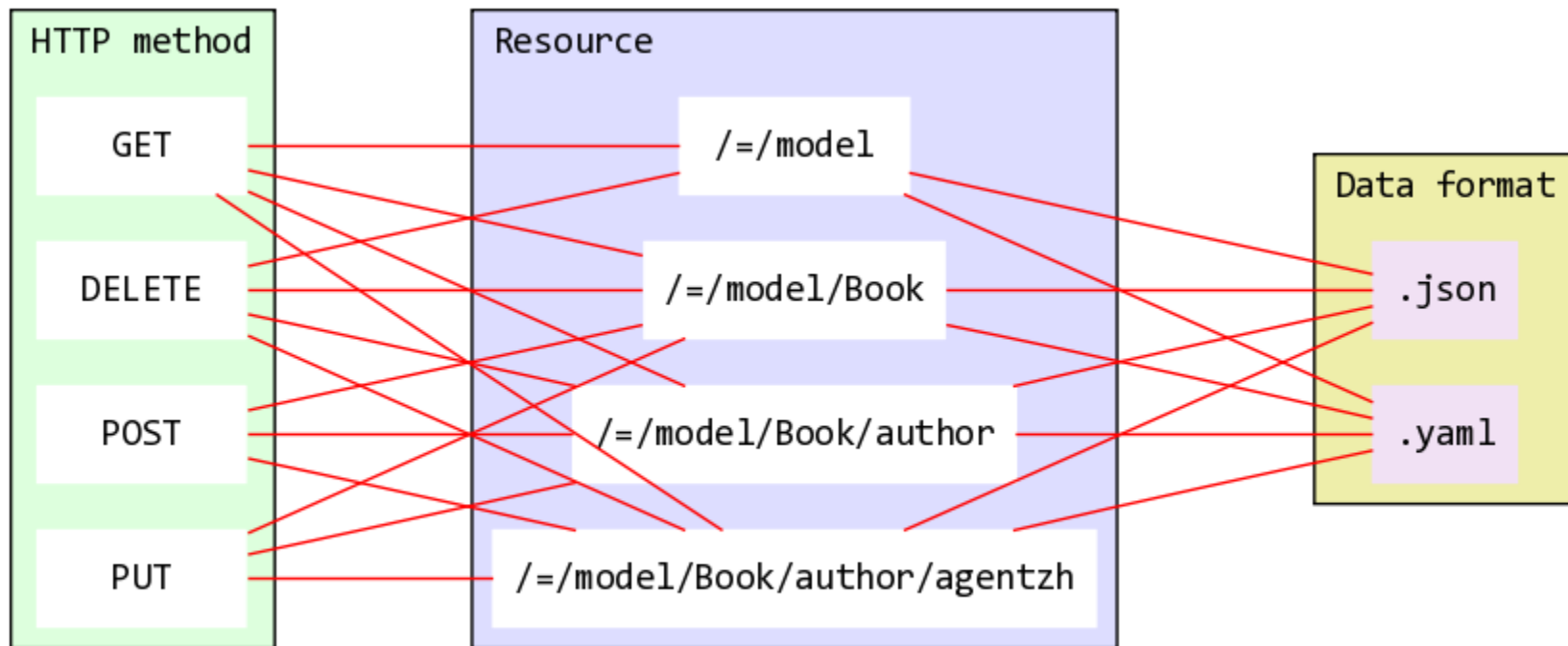
OpenResty is a

- ✓ *general-purpose* **RESTful** web service platform
- ✓ REST **wrapper** for *relational* databases
- ✓ Virtual web *runtime* for **RIA**
- ✓ "*meta* web site" supporting **other sites**
via web services
- ✓ handy *web database* which can be accessed
from **anywhere**

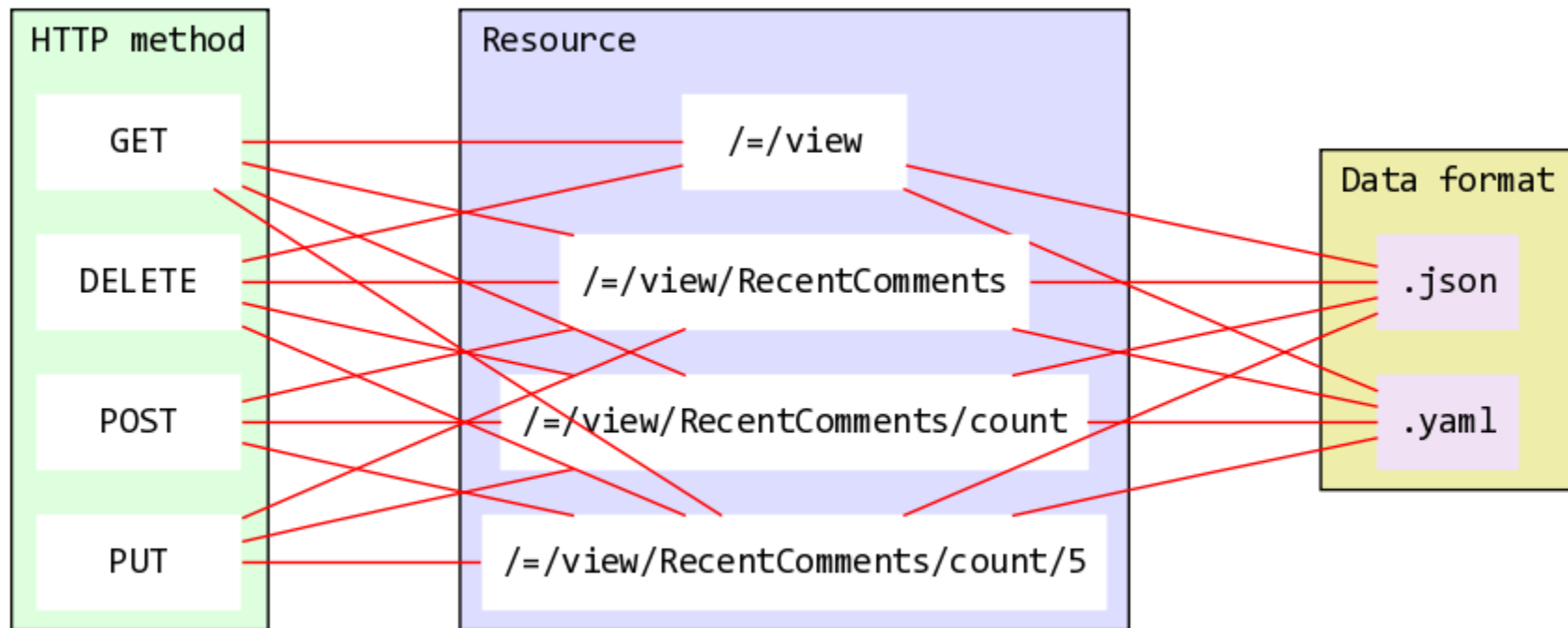


The OpenResty cloud within Yahoo! China

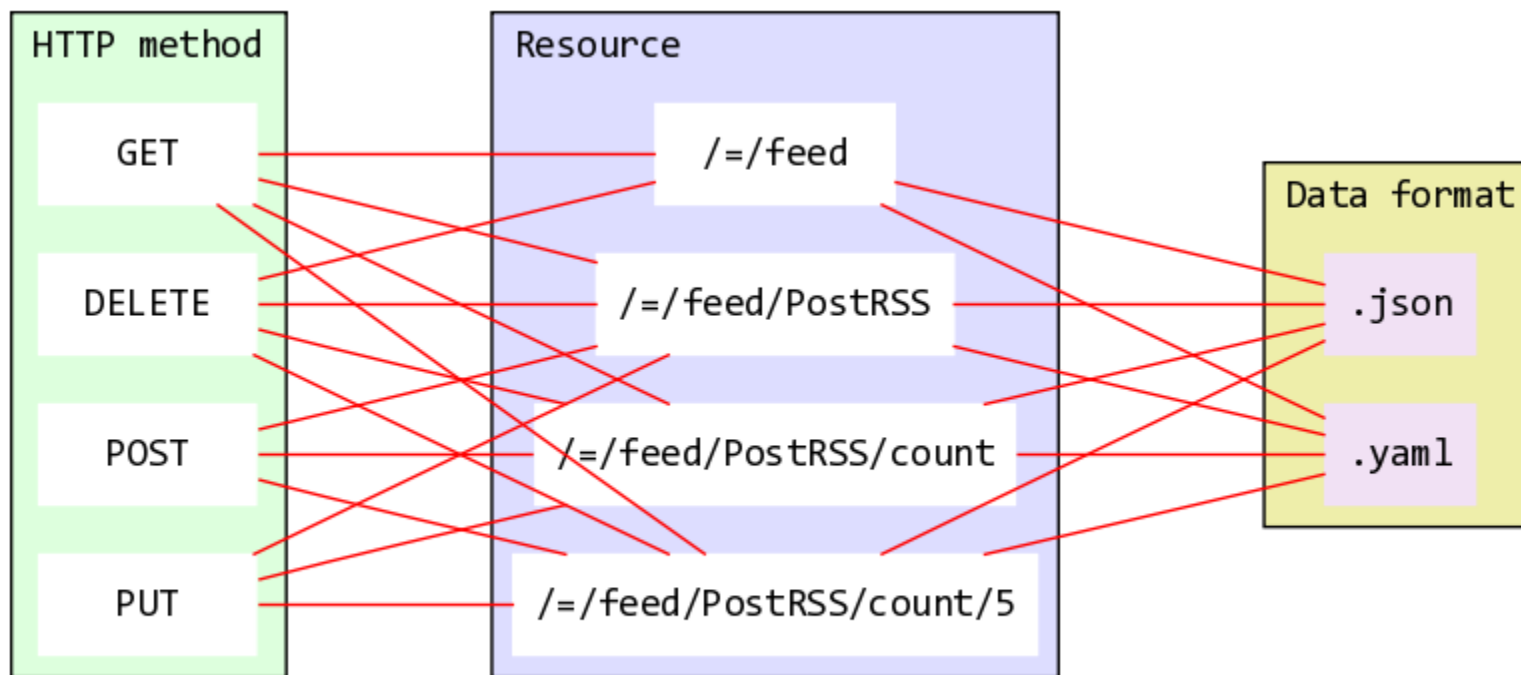
☺ agentzh loves combinations,
so he designed his API *that way*.



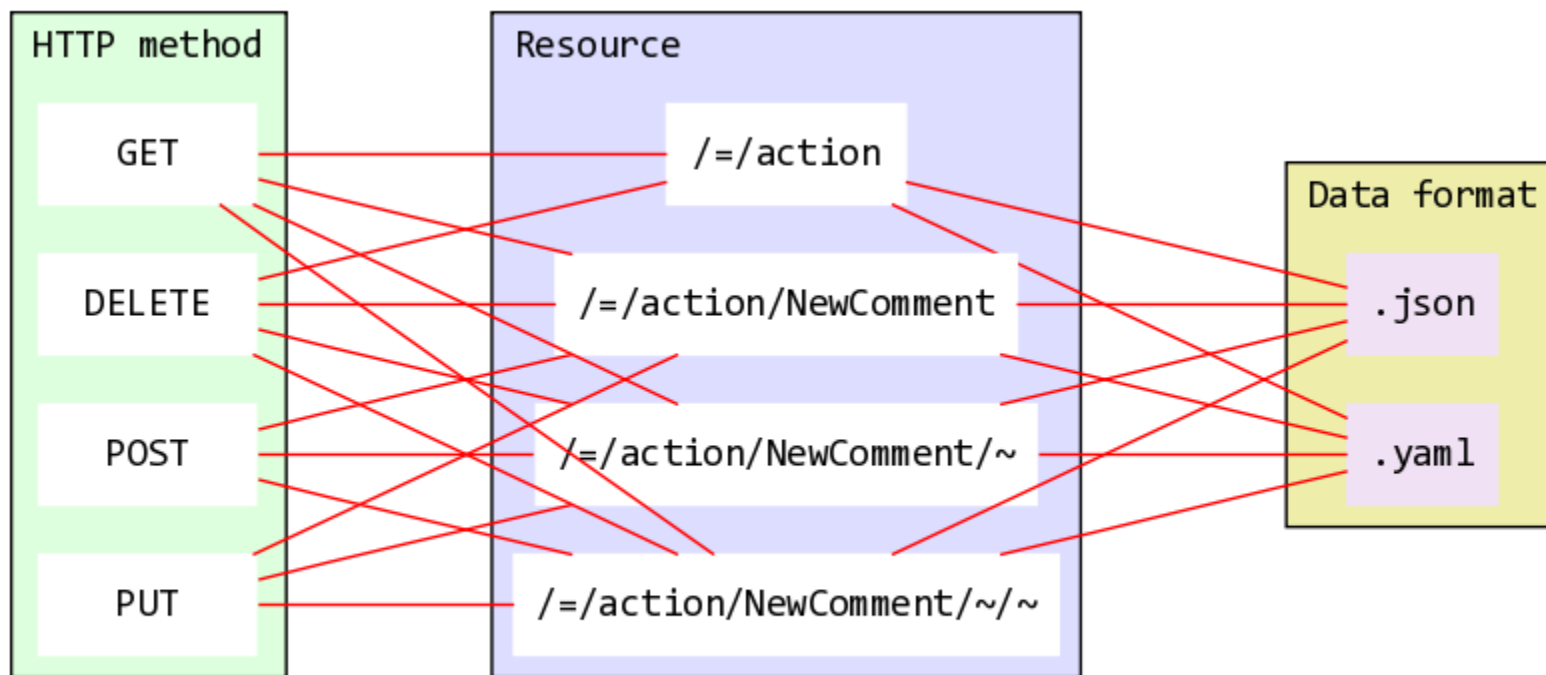
Combinations in OpenResty's Model API



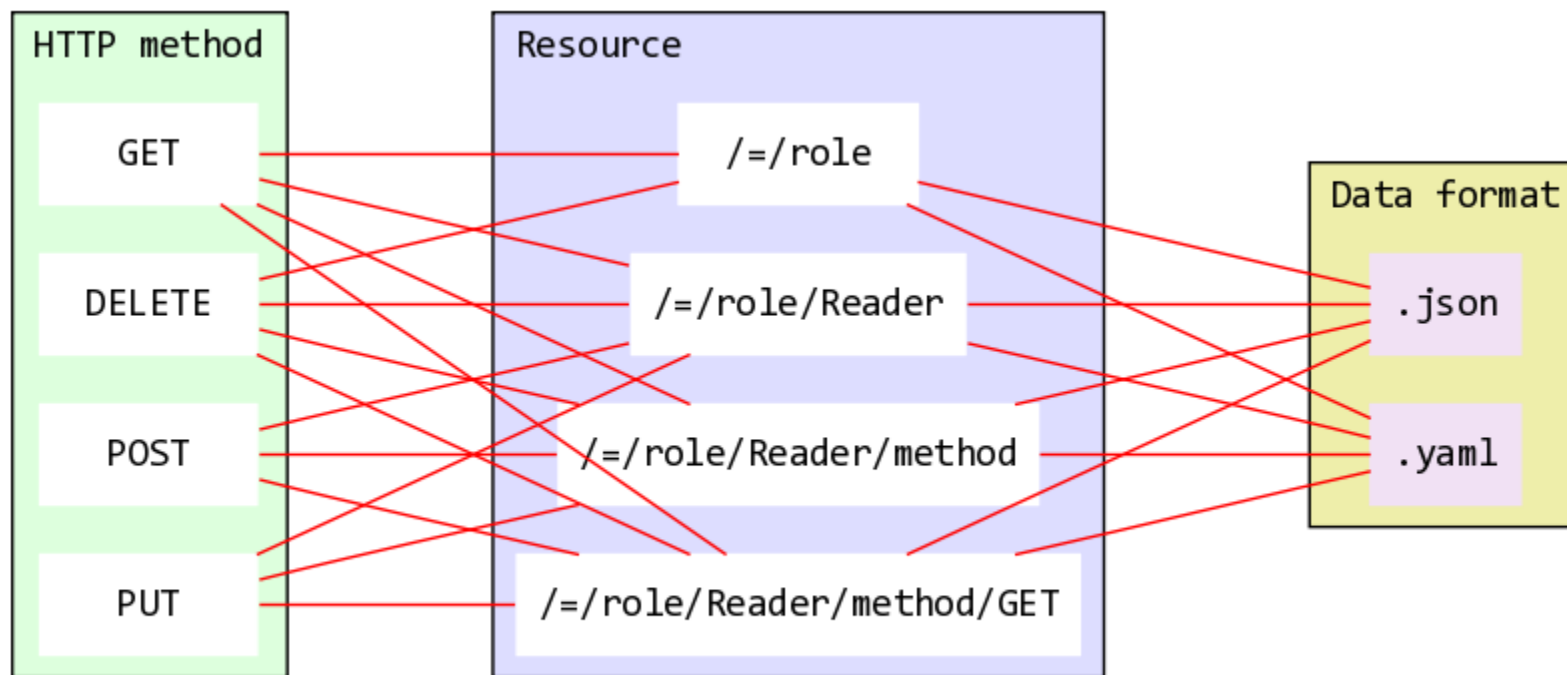
Combinations in OpenResty's View API



Combinations in OpenResty's Feed API



Combinations in OpenResty's Action API



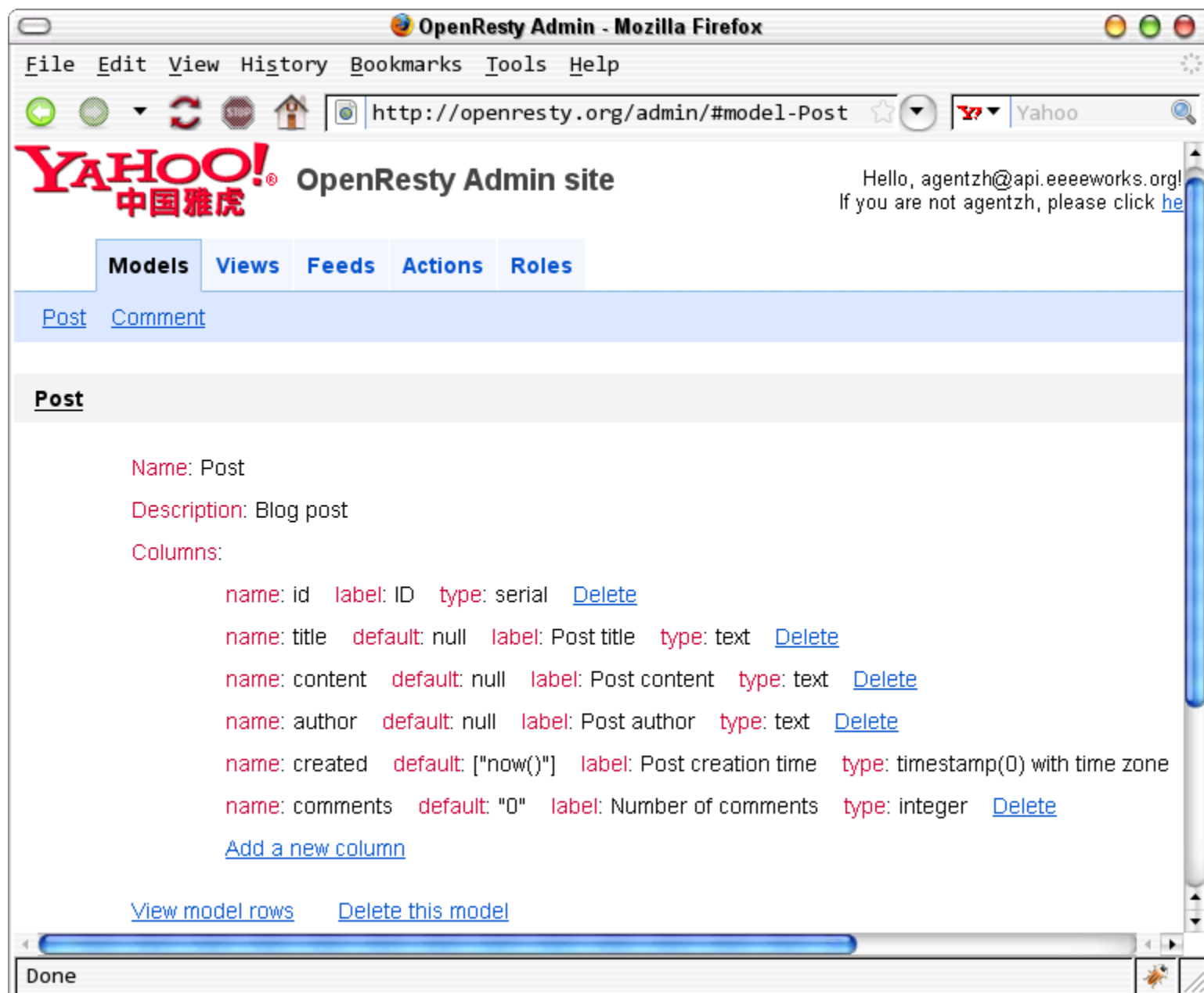
Combinations in OpenResty's Role API

☆ **Models:** Your database *tables*

(02:38:03 PM)laser: **model** means *table* in RDBMS world, right?

(02:38:08 PM)agentzh: yup

(02:38:14 PM)laser: and **column** means *column*



Let's *create* a Post **model** for my blog...

POST **/=/model/Post** HTTP/1.0

Content-Type: **text/json**

Content-Length: 111

```
{  
  "description": "Blog post",  
  "columns": [  
    { "name": "title", "label": "Post title", "type": "text" },  
    { "name": "content", "label": "Post content", "type": "text" },  
    { "name": "author", "label": "Post author", "type": "text" },  
    { "name": "created", "default": ["now()"],  
      "type": "timestamp (0) with time zone",  
      "label": "Post creation time" }  
  ],  
}
```

Insert some rows?

POST /=/model/Post/~/~ **HTTP/1.0**

Content-Type: text/json

Content-Length: 92

```
{  
  "title": "My first post",  
  "content": "Blah blah blah...",  
  "author": "agentzh"  
}
```

Get my *rows* back! ...with *paging*...

GET **/=/model/Post/~/?_offset=0&_limit=100** **HTTP/1.0**

How about *posts* written by *agentzh*?

GET **/=/model/Post/author/agentzh** **HTTP/1.0**

Posts with the title containing the word "OpenResty"?

GET **/=/model/Post/title/OpenResty?_op=contains** **HTTP/1.0**

Okay...I want to *add* a new **column** named "comments"

POST **/=/model/Post/~** HTTP/1.0

Content-Type: text/json

Content-Length: 43

```
{  
  "name": "comments",  
  "label": "Number of comments to this post",  
  "type": "integer",  
  "default": 0  
}
```

Err...let me see *all* the **columns** of the model?

GET **/=/model/Post/~ HTTP/1.0**

...or *just* the definition of the **author** column?

GET /=/model/Post/author **HTTP/1.0**

Change the column name?

PUT /=/model/Post/author **HTTP/1.0**

Content-Type: text/json

Content-Length: 43

{ "name": "sender" }

Hmm...I'd *remove* this **column** from the model!

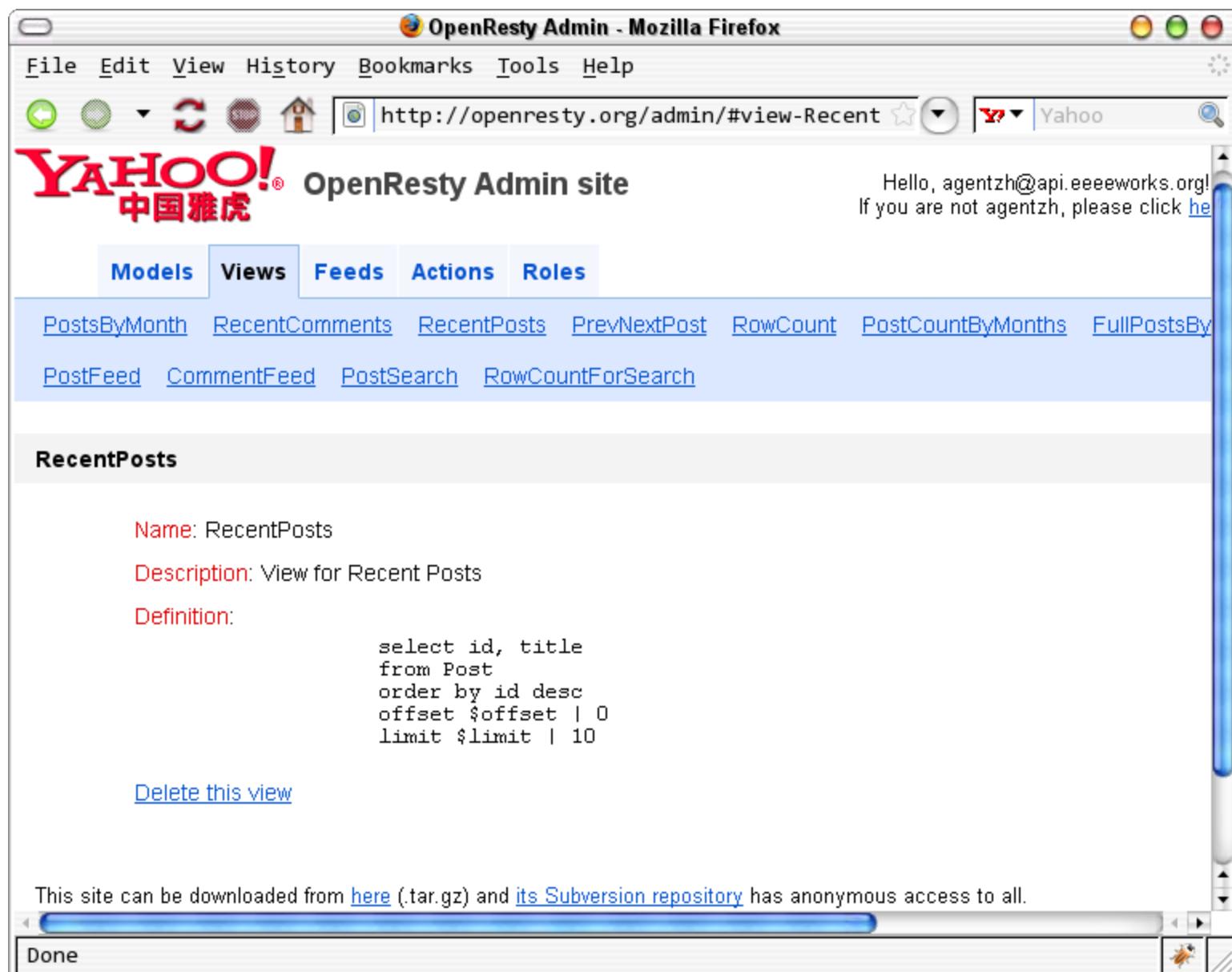
DELETE /=/model/Post/sender **HTTP/1.0**

☆ **Views:** Your custom *SQL* queries

(09:02:20 PM)laser: **View** is a very *cool* invention

(09:02:30 PM)laser: heh

(09:02:52 PM)laser: **SQL** is a *4th* generation language



POST /=/view/~ HTTP/1.0

Content-Type: text/json

Content-Length: 312

```
{  
  "name": "CommentsForAuthor",  
  "description": "Comments for the specified author",  
  "definition": "  
    select Comment.sender as guest,  
      Comment.body as comment_body  
  from Comment, Post  
  where Comment.id = Post.id and Post.author = $author"  
}
```

Get all the *comments* for the author agentzh

GET /=/view/CommentsForAuthor/author/agentzh **HTTP/1.0**

Or equivalently

GET /=/view/CommentsForAuthor/~/?author=agentzh **HTTP/1.0**

The HTTP *response* might be

HTTP/1.0 200 OK

Connection: close

Content-Type: text/json; charset=UTF-8

Content-Length: 187

Date: Mon, 21 Apr 2008 12:42:15 GMT

```
[  
  { "guest": "laser", "comment_body": "super cool!" },  
  { "guest": "carrieh", "comment_body": "hello?hello?" },  
  { "guest": "agentzh", "comment_body": "Thanks for commenting!" }  
]
```

☆ Feeds: Your *RSS/Atom* for views.

(10:38:28 AM)agentzh: at present, **feed** objects are just
XML wrappers for **views**.

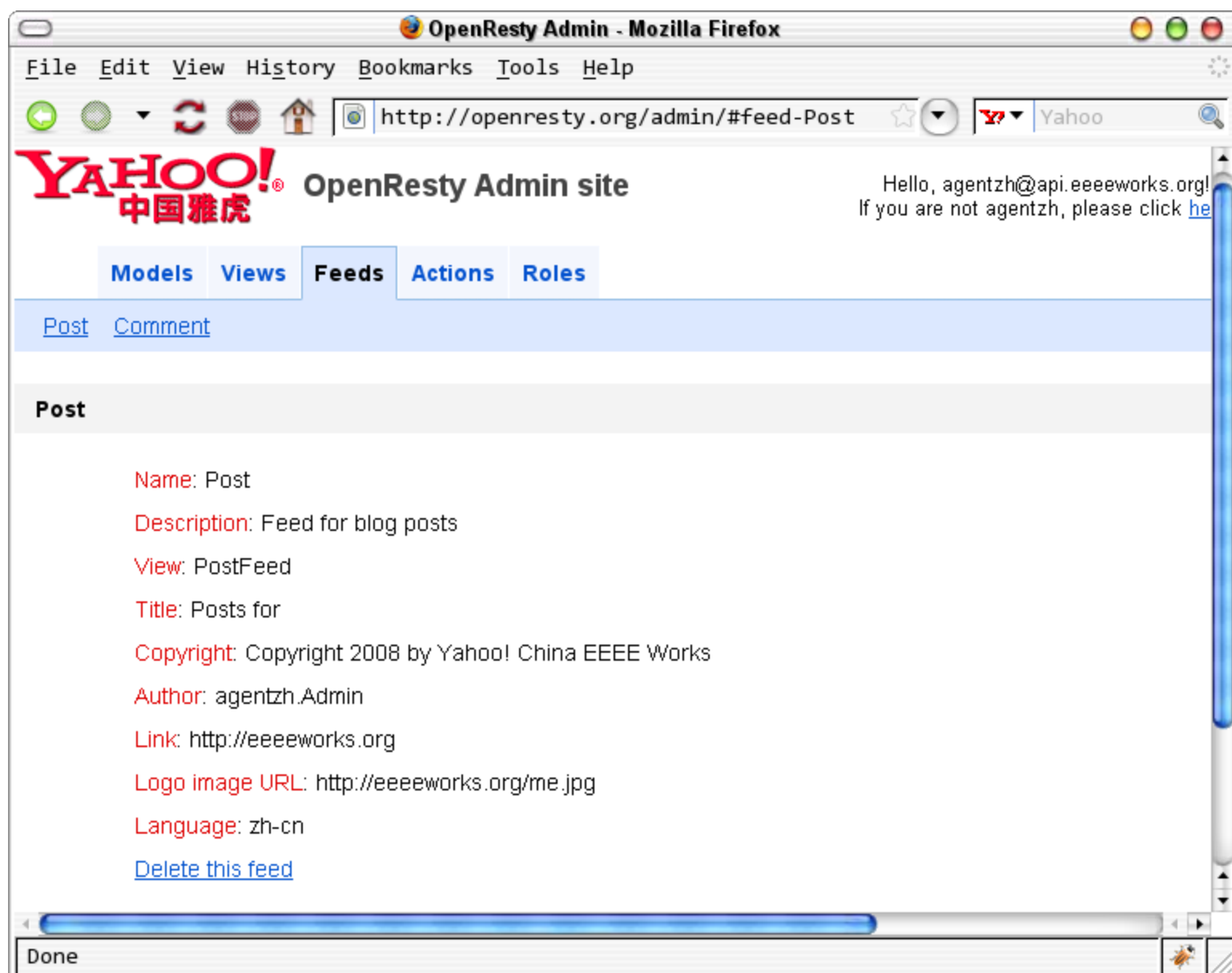
(10:38:31 AM)agentzh: heh

(10:38:37 AM)agentzh: i like this idea.

(10:38:28 AM) agentzh: 目前，**Feed** 对象就是 **View** 对象的 XML 包裹

(10:38:31 AM) agentzh: 呵呵

(10:38:37 AM) agentzh: 这个概念我挺喜欢的



We first define a *view* to serve as the data source

POST **/=/view/PostFeedView** **HTTP/1.0**

Content-Type: text/json

Content-Length: 312

```
{  
  "description": "View for post feed",  
  "definition":  
    "  
    select author, title,  
      content, created as published  
    from Post  
    order by created desc  
    limit $count | 20  
    "  
}
```

Then we create a *feed object* atop this view:

POST **/=/feed/Post** HTTP/1.0

Content-Type: text/json

Content-Length: 124

```
{  
  "description": "Feed for blog posts",  
  "author": "agentzh",  
  "copyright": "Copyright 2008 by Yahoo! China EEEE Works",  
  "language": "zh-cn",  
  "title": "Posts for Human & Machine",  
  "link": "http://blog.agentzh.org",  
  "logo": "http://blog.agentzh.org/me.jpg",  
  "view": "PostFeedView"  
}
```

The *feed* is ready for subscribers!

GET /=/feed/Post/~/~ **HTTP/1.0**

Or it can take *arguments* just as the underlying **view**:

GET **/=/feed/Post/count/20** **HTTP/1.0**

☆ **Actions:** *Compose* services the way you like

(03:17:11 PM)agentzh: **action** support is the next milestone

(03:17:28 PM)agentzh: so what is an action?

(03:17:47 PM)agentzh: an openresty action is a *bunch* of **restyscript commands** with a *name* attached to it.

Compose several **view invocations** into one action:

POST **/=/action/Init** **HTTP/1.0**

Content-Type: text/json

Content-Length: 224

```
{  
  "description": "Initialization for my site",  
  "parameters": [],  
  "definition": "  
    GET '/=/view/RecentComments/~/~';  
    GET '/=/view/RecentPosts/~/~';  
    GET '/=/model/Post/~/~?_offset=0&_limit=10';  
  "  
}
```

The server returns the *result sets* for all the **view calling**:

```
[  
  [ ... ], // the results for the first view  
  [ ... ], // the results for the second view  
  [ ... ]  // the results for the third view  
]
```

♥ You can also *mix*
SQL, HTTP commands, and JSON in one action!

POST /=/action/~ HTTP/1.0

Content-Type: text/json

Content-Length: 189

```
{  
  "name": "PostComment",  
  "description": "Action for posting a comment",  
  "parameters": [  
    { "name": "sender", "type": "literal" },  
    { "name": "body", "type": "literal" },  
    { "name": "post_id", "type": "literal" },  
  ],  
}
```

"definition":

"

update Post

set comments = comments + 1

where id = \$post_id;

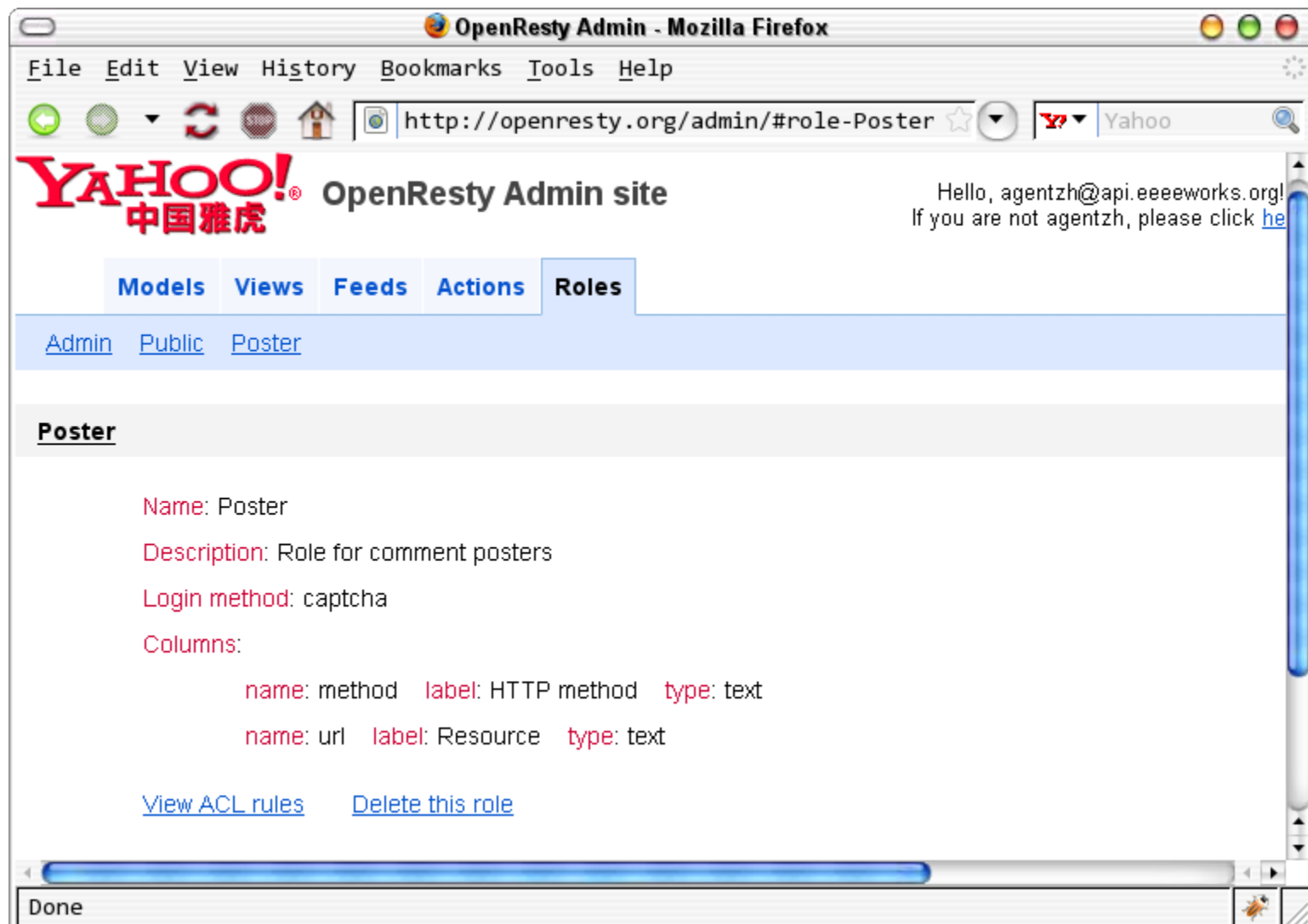
POST '/=/model/Comment/~/~'

{ "sender": \$sender, "body": \$body, "post": \$post_id }

"

}

☆ **Roles:** Your custom *access control*
(at service level)



Create a *read-only* role

POST **/=/role/Reader** HTTP/1.0

Content-Type: text/json

Content-Length: 120

```
{  
  "description": "My Read-only role",  
  "login": "anonymous", // or password or captcha  
  "password": null  
}
```

The **Reader** role *cannot* access any resources by default,
let's *add* some **ACL rules** to it.

POST **/=/role/Reader/~/~** HTTP/1.0

Content-Type: text/json

Content-Length: 82

[

// allow reading object lists:

{ **"method": "GET", "url": "/=/~" },**

// allow reading objects:

{ **"method": "GET", "url": "/=/~/~" },**

// allow reading object columns/parameters:

{ **"method": "GET", "url": "/=/~/~/~" },**

// allow reading object rows/rules:

{ **"method": "GET", "url": "/=/~/~/~/~" }**

]

Reader roles in action (allowed):

- ✓ **GET** /=/model/Post/id/3?_user=agentzh.Reader
- ✓ **GET** /=/view?_user=agentzh.Reader
- ✓ **GET** /=/role/Reader/method/GET?_user=agentzh.Reader
- ✓ **GET** /=/role/Reader?_user=agentzh.Reader

Reader roles in action (denied):

- × **DELETE** /=/model/Post?_user=agentzh.Reader
- × **PUT** /=/model/Post/author?_user=agentzh.Reader
- × **POST** /=/role/Reader/~/?_user=agentzh.Reader

☆ Captchas: Make your services *human-only*

(10:34:50 AM)Carrie: 恩。。。如果我想在发留言的时候

(10:34:56 AM)Carrie: 增加个水印标志

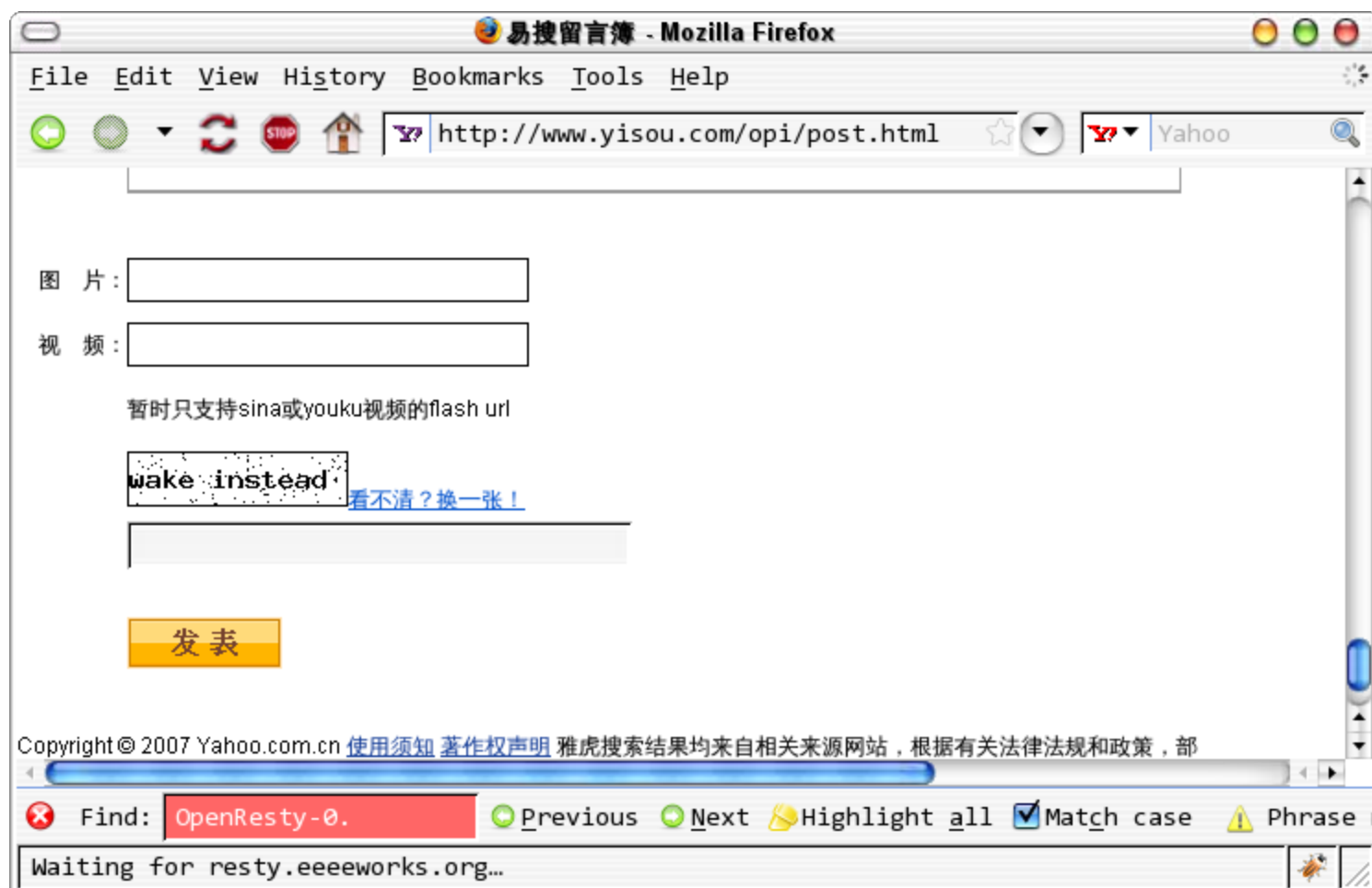
(10:36:44 AM)Carrie: 我想咱们可能得先实现这个水印的东西。。

。。。因为貌似很需要。。。。

(10:36:56 AM)agentzh: 验证码图片属于一种特殊的角色登录方式

(10:37:14 AM)agentzh: 该角色的"登录密码"将不放在应用的 JS 中

(10:37:21 AM)agentzh: 而是指向一个验证码



Create an ad-hoc **role** with the *captcha* login method

POST **/=/role/CommentPoster** **HTTP/1.0**

Content-Type: text/json

Content-Length: 64

```
{  
  "description": "Role for posting comments",  
  "login": "captcha"  
}
```

Assign *access* to the *operations* required solving captchas

POST **/=/role/CommentPoster/~/~** HTTP/1.0

Content-Type: text/json

Content-Length: 48

{ "method":"POST", **"url":**"/=/model/Comment/~/~" **}**

Then in the *client-side* JavaScript code...

GET /=/captcha/id **# get the captcha id**

GET /=/captcha/id/[**captcha_id**] **# get the captcha image**

POST /=/model/Comment/~/?_user=agentzh.**CommentPoster** \
 &_captcha=[**captcha_id**]:[**user_solution**] \
 HTTP/1.0

Content-Type: text/json

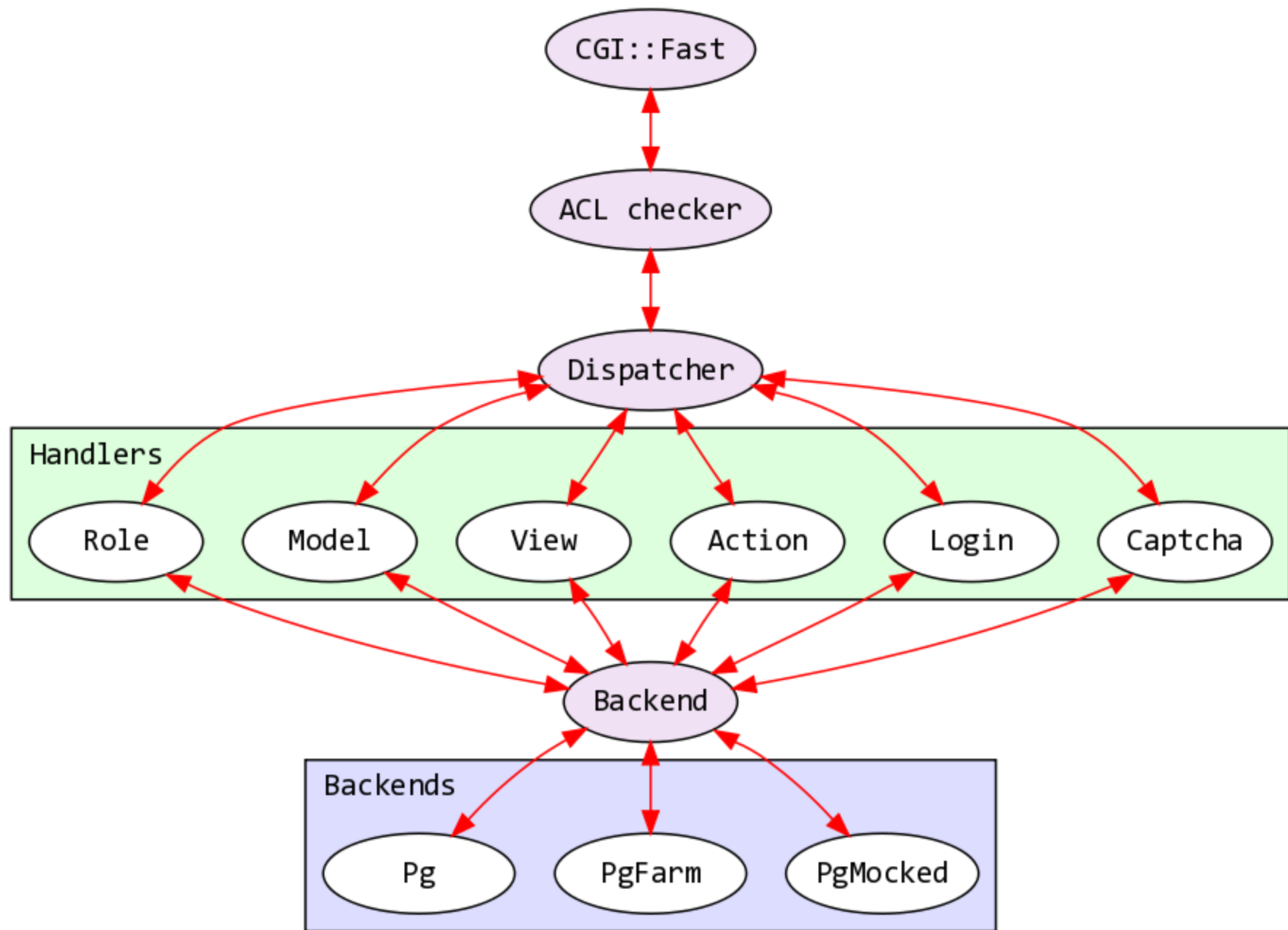
Content-Length: 52

{"sender":"agentzh","body":"Good post!","post":25}

OpenResty offers

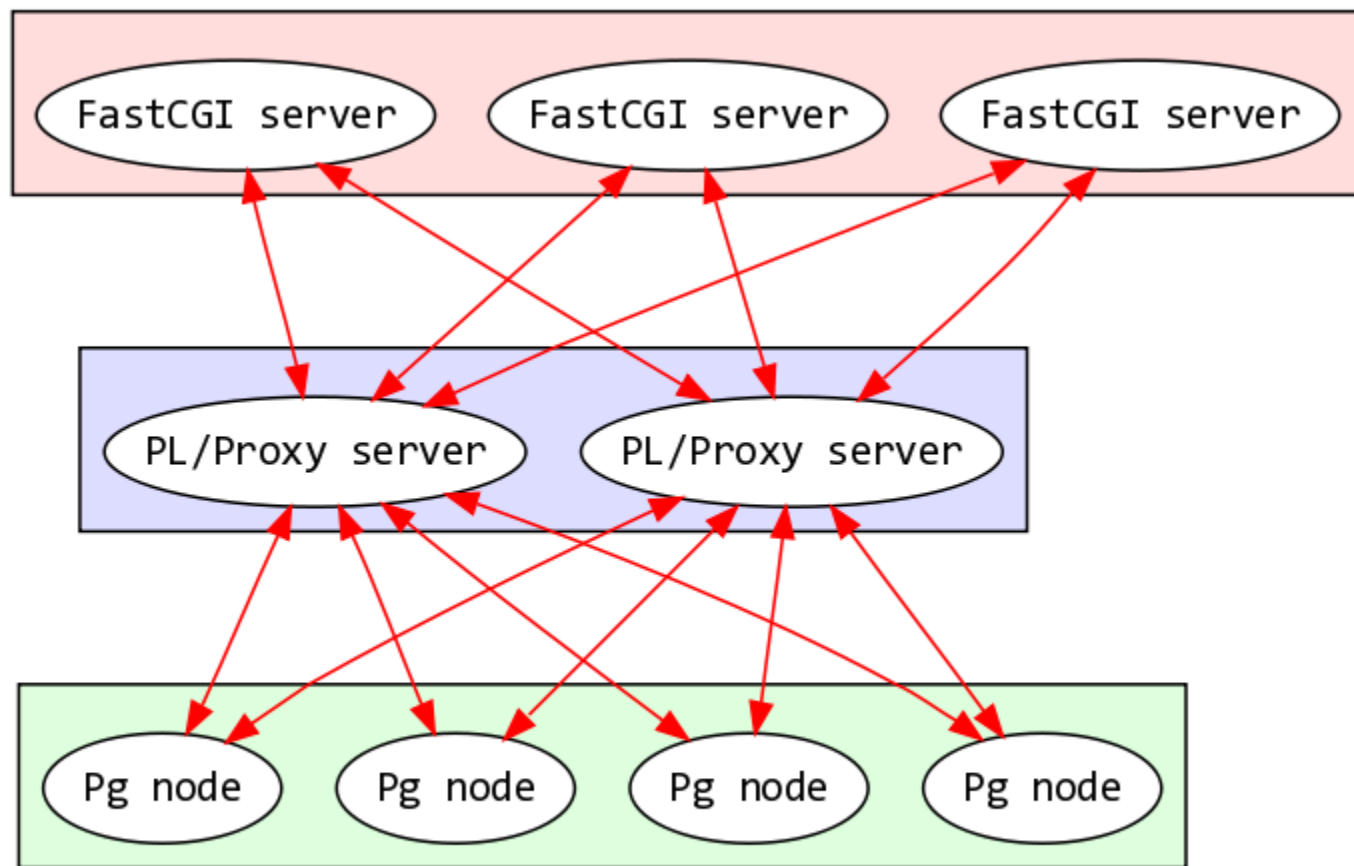
- ✓ (scalable) *relational* data **storage**
- ✓ *truely* **RESty** interface
- ✓ *JSON/YAML* data transfer format
- ✓ **SQL**-based reusable *views*
- ✓ a REST-oriented *role system* for **ACL**
- ✓ *view*-based **RSS feeds**
- ✓ user-defined *actions* in **RestyScript**
- ✓ native *captchas*
- ✓ *cross-site* **AJAX** support

♥ The OpenResty *FastCGI* server
is currently written in Perl 5.



The Internal Structure of the OpenResty Server

♥ The **PgFarm** backend of OpenResty
is designed to be *scalable*.



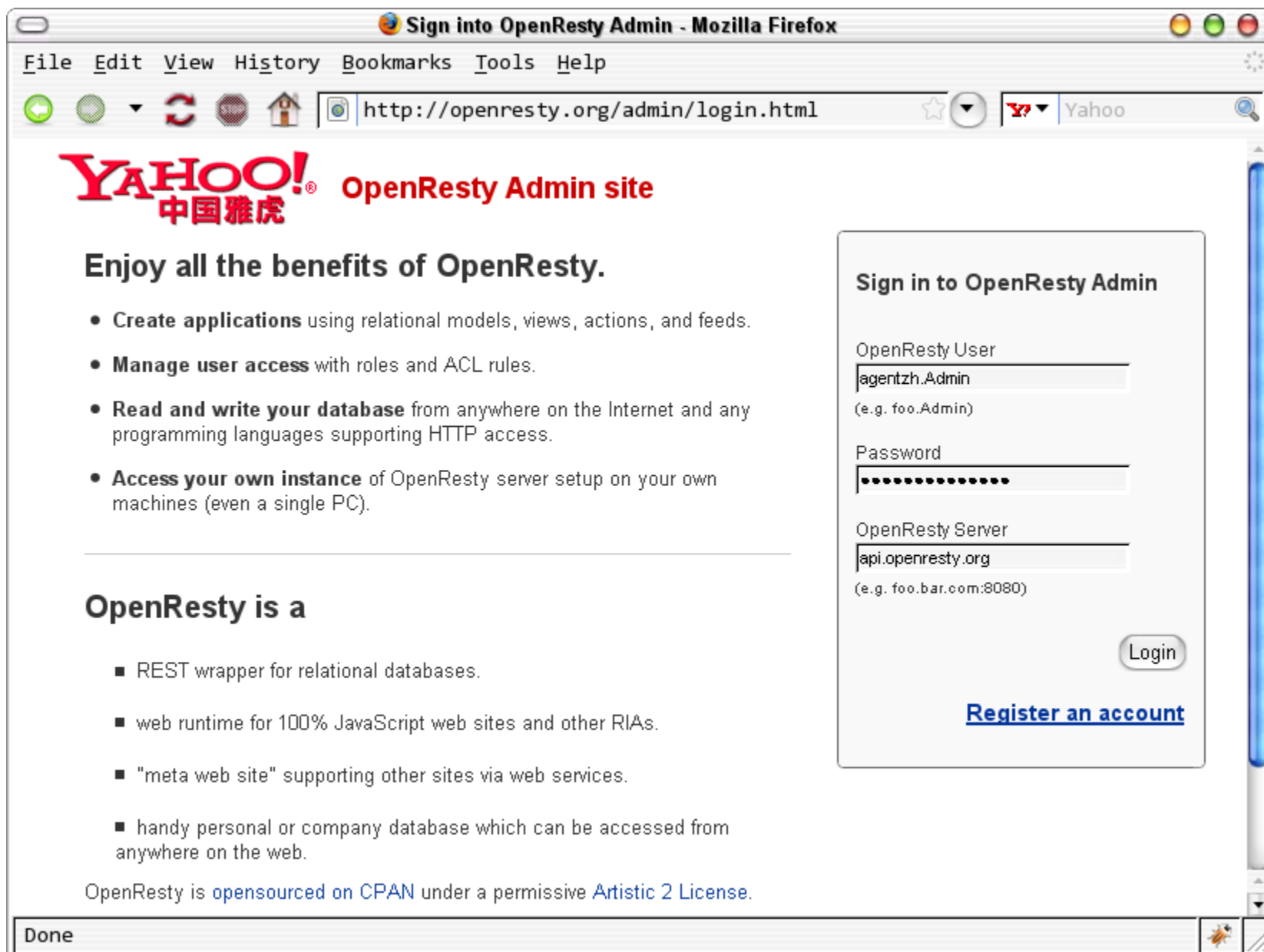
The Architecture Diagram for OpenResty PgFarm

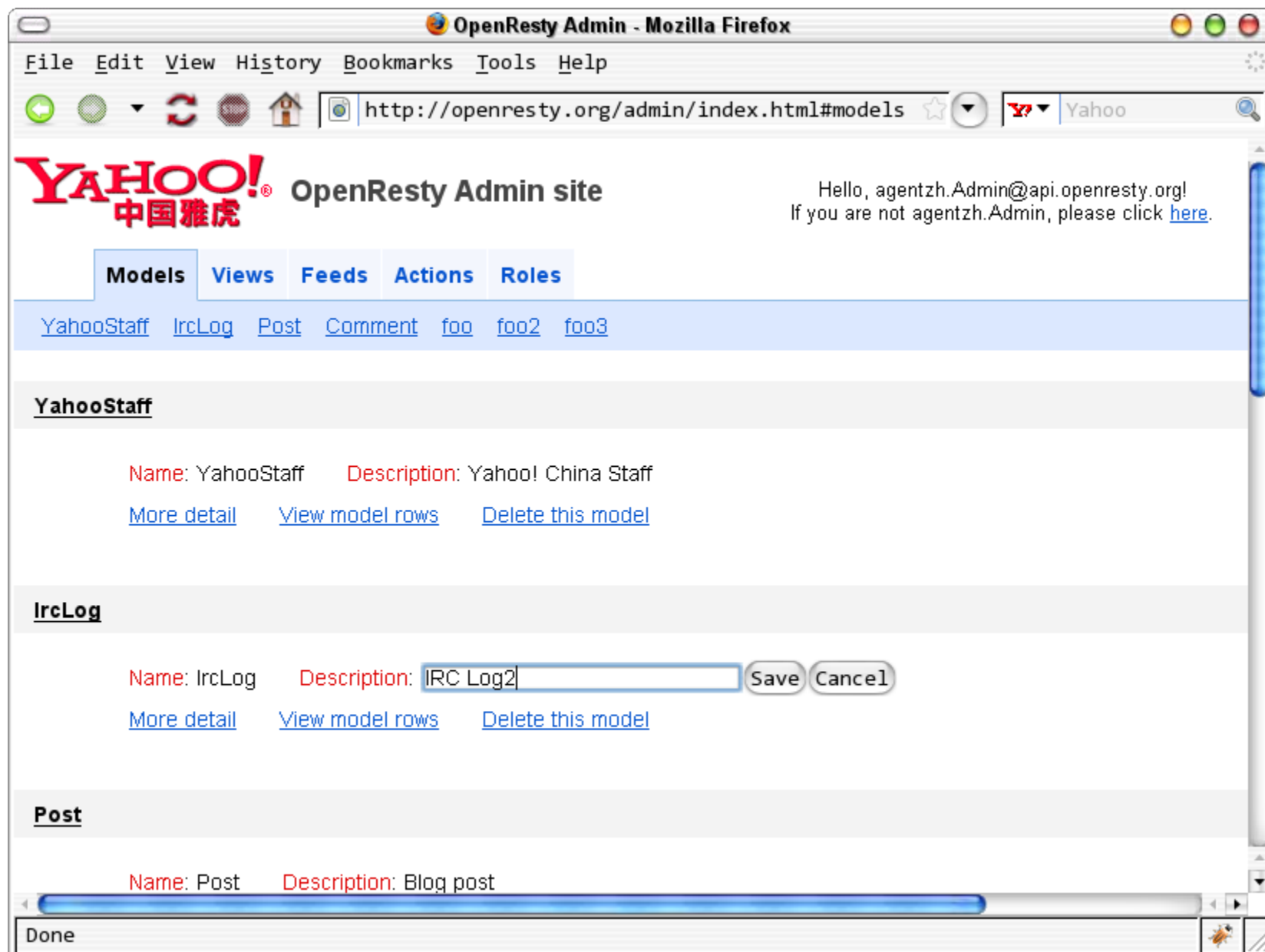
😊 OpenResty comes with a *pony*
(just like Jifty)



(04:49:43 PM)Carrie: 可是你不觉得你现在应该先写出一个**admin page**来么

(04:49:55 PM)Carrie: 要不大家以后用你的**api**就跟**背课文**似的。。。。





☺ It's written in *100%* JavaScript as well.

CPAN modules *born* from **OpenResty**

- ✓ **CGI::Cookie::XS**
- ✓ **Filter::QuasiQuote**

OpenResty *handlers* in the **future**:

♡ **/=/mail/...**

♡ **/=/attachment/...**

♡ **/=/template/...**

♡ **/=/git/...?**

♡ **/=/prophet/...?**

...*Any one?*

OpenResty *backends* in the **future**:

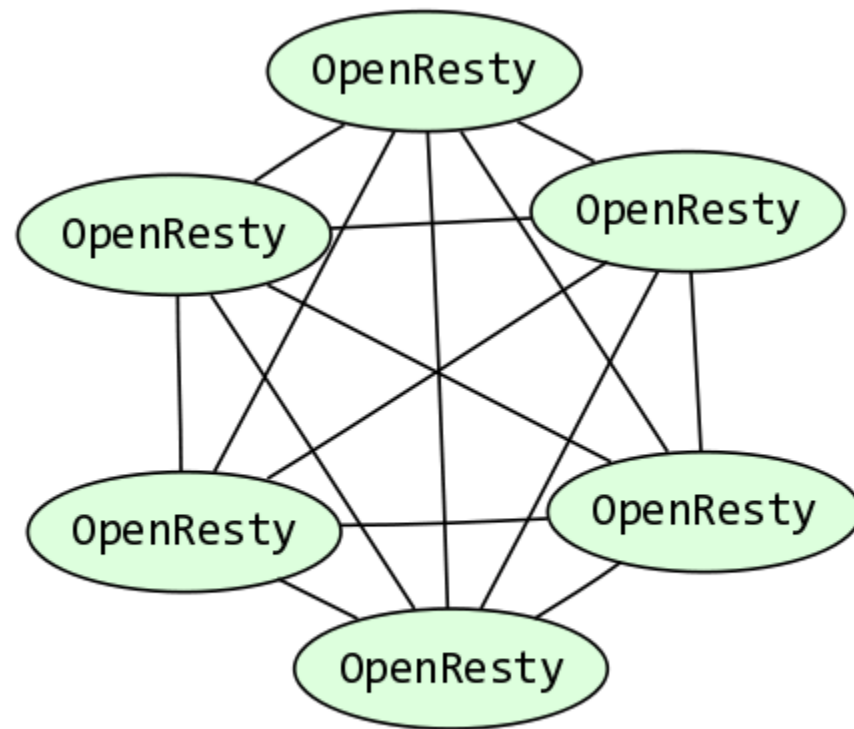
♡ **MySQL**

♡ **Oracle**

♡ **MS SQL Server**

♡ ***Put your favorite RDBMS here...***

Or *peer-to-peer* OpenResty **clusters** in the future?
A *cluster* of *clusters*?



Kademlia network for OpenResty clusters

😊 *Any questions?* 😊

