Programming the Kernel of Web 2.0

Audrey Tang

Programming the Kernel of Web 2.0 Hype

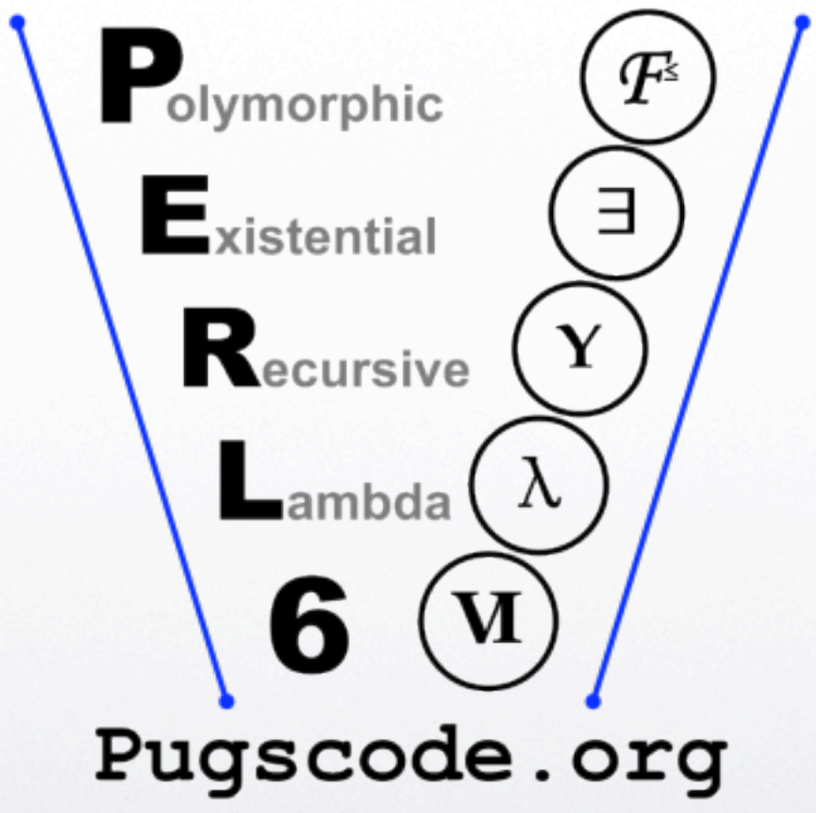
Jifty

RHOX

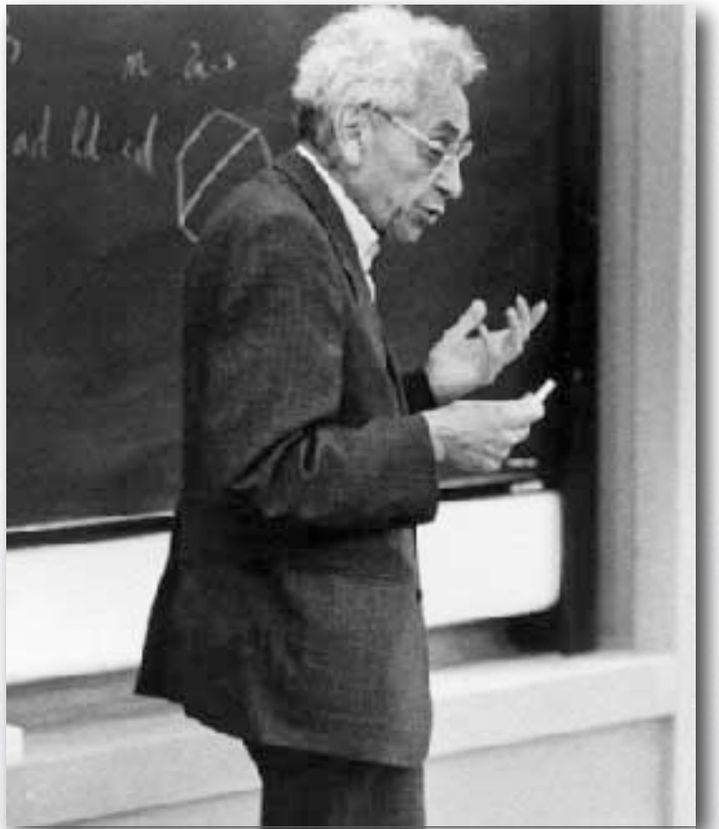
my $Job

my $Job

2005: Pugs



2006: Erdösing



2006: Erdösing

2006: Erdösing

$Job @ S-Team

$Job @ S-Team

- Consultant for banks

$Job @ S-Team

- Consultant for banks
- Huge existing codebase

$Job @ S-Team

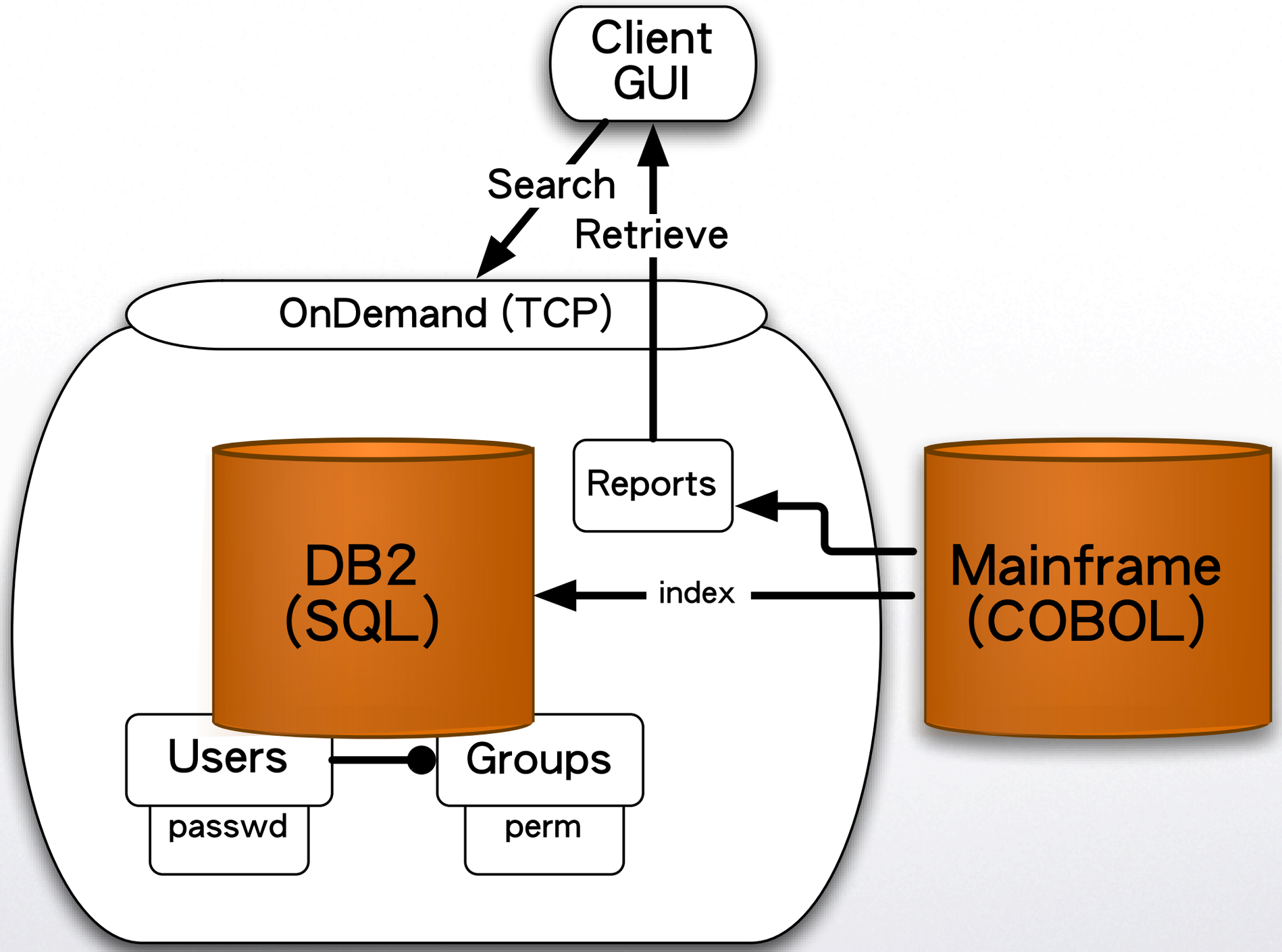
- Consultant for banks
- Huge existing codebase
- Weird complex systems

$Job @ S-Team

- Consultant for banks
- Huge existing codebase
- Weird complex systems
- Rapidly shifting spec

OnDemand Server

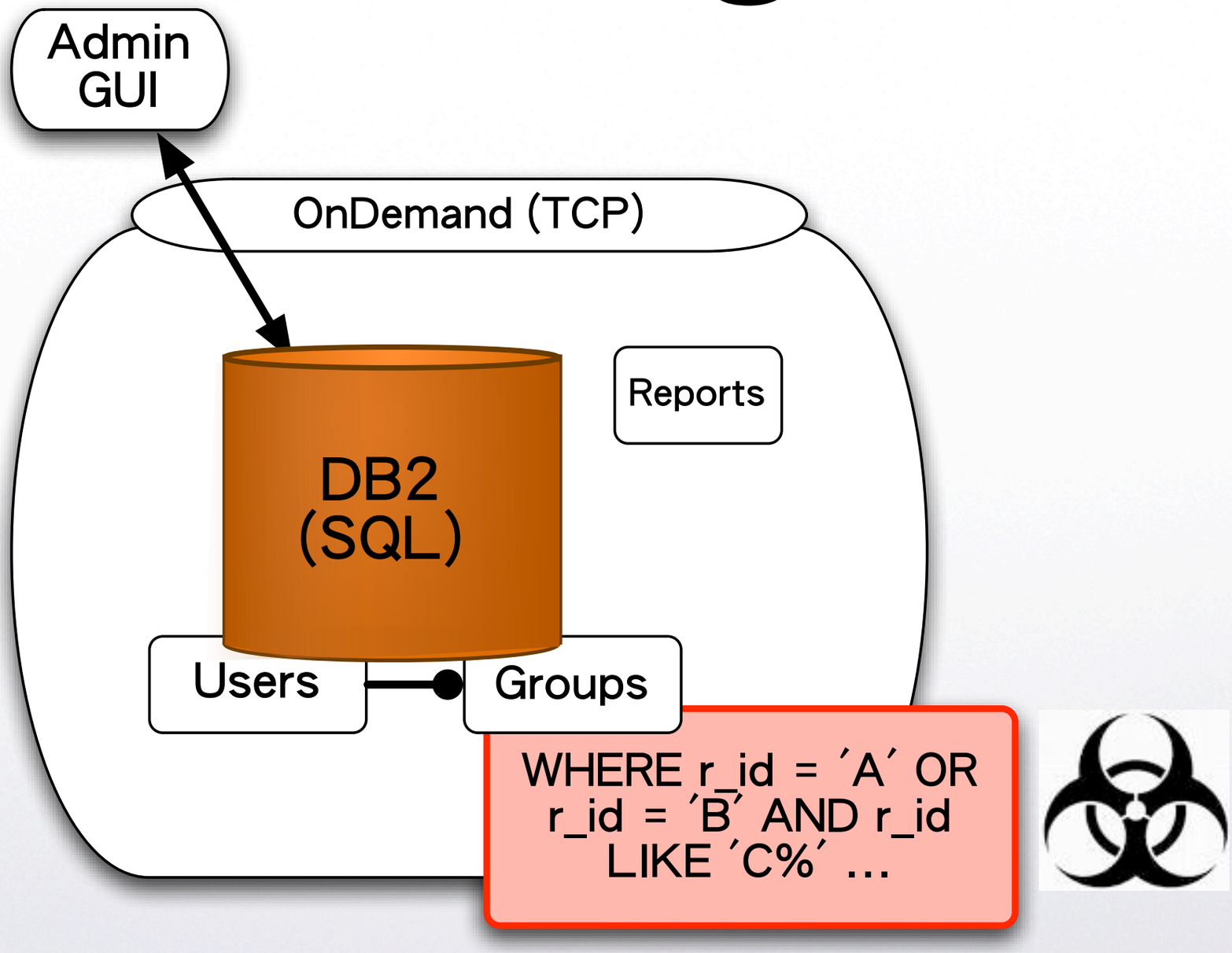
“Nobody gets fired.”


Client
GUI
Search
Retrieve
OnDemand (TCP)
DB2
(SQL)
Reports
index
Mainframe
(COBOL)
Users
Groups
passwd
perm

It all makes sense

if you *don't* think about it...

Sysadmin's Nightmare



Bad Black Box

Bad Black Box

- Hand-written WHEREs

Bad Black Box

- Hand-written WHEREs
- One group per user

Bad Black Box

- Hand-written WHEREs
- One group per user
- Windows GUI only

Bad Black Box

- Hand-written WHEREs
- One group per user
- Windows GUI only
- API? What's that?

Wish List

Wish List

- Single sign on

Wish List

- Single sign on
- Web-based ACL editor

Wish List

- Single sign on
- Web-based ACL editor
- Comprehensive logging

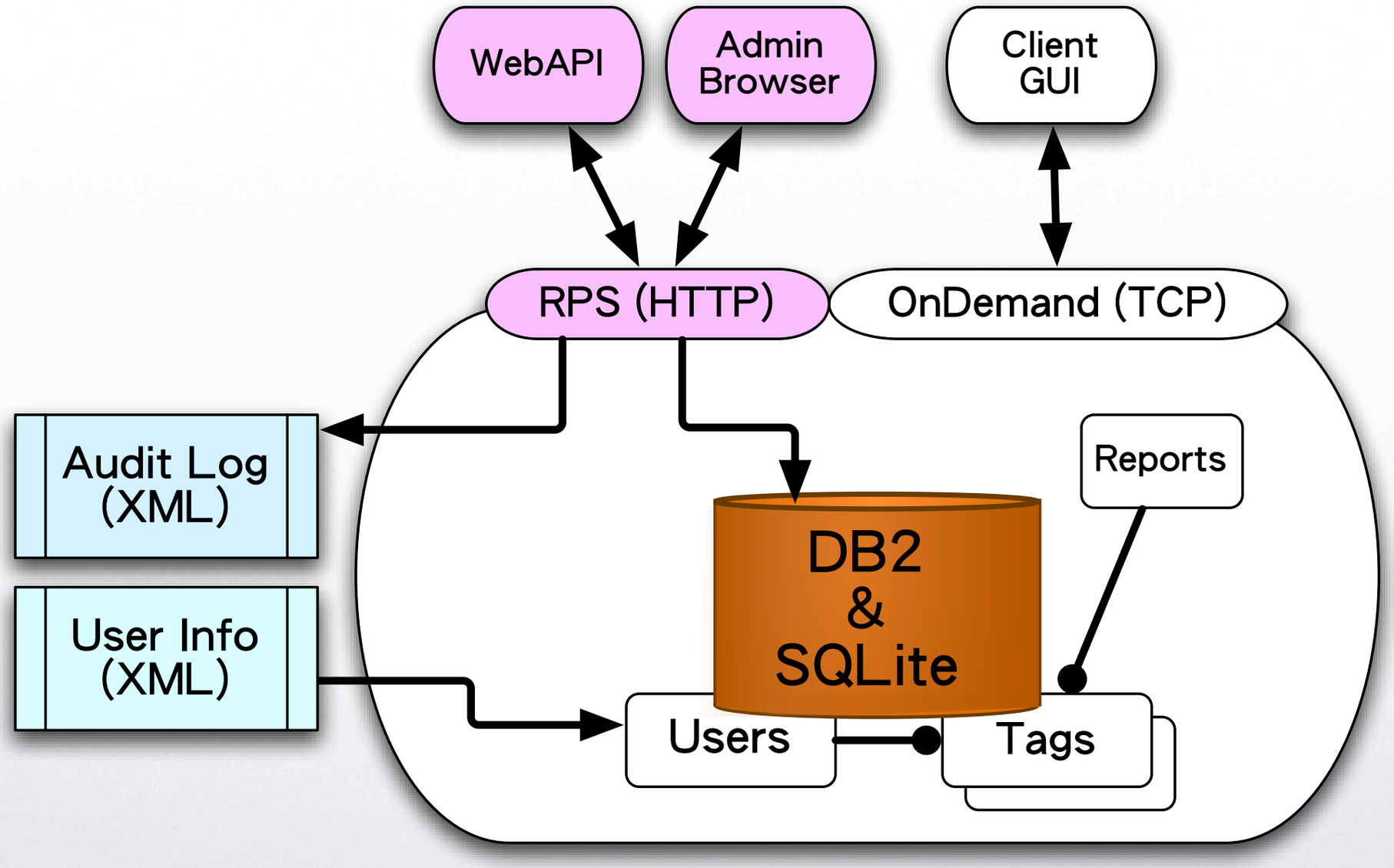
Wish List

- Single sign on
- Web-based ACL editor
- Comprehensive logging
- Bug-for-bug compatibility

Wish List

- Single sign on
- Web-based ACL editor
- Comprehensive logging
- Bug-for-bug compatibility
- …on 233MHZ AIX machines

Report Permission System



Design

Design

- *Tags* for users and reports

Design

- *Tags* for users and reports
- *Feeds* and *Forms* as API

Design

- *Tags* for users and reports
- *Feeds* and *Forms* as API
- *Sync* upstream user info

Design

- *Tags* for users and reports
- *Feeds* and *Forms* as API
- *Sync* upstream user info
- *Emit* SQL back to DB2

The RHOX Beast

The RHOX Beast

- Relational

The RHOX Beast

- Relational
- Hypertext

The RHOX Beast

- Relational
- Hypertext
- Object

The RHOX Beast

- Relational
- Hypertext
- Object
- XML

Relational

Relational

- *"Everything is a row."*

Relational

- *"Everything is a row."*

- DB2

Relational

- *"Everything is a row."*

- DB2
- SQLite

Hypertext

Hypertext

- *"Everything is a resource."*

Hypertext

- *"Everything is a resource."*

- HTTP

Hypertext

- *"Everything is a resource."*
- HTTP
- XmlHTTPRequest

Object

Object

- *"Everything is a graph."*

Object

- *"Everything is a graph."*

- **Perl**

Object

- *"Everything is a graph."*
- Perl
- YAML/JSON

XML

XML

- *"Everything is a tree."*

XML

- *"Everything is a tree."*

- XHTML

XML

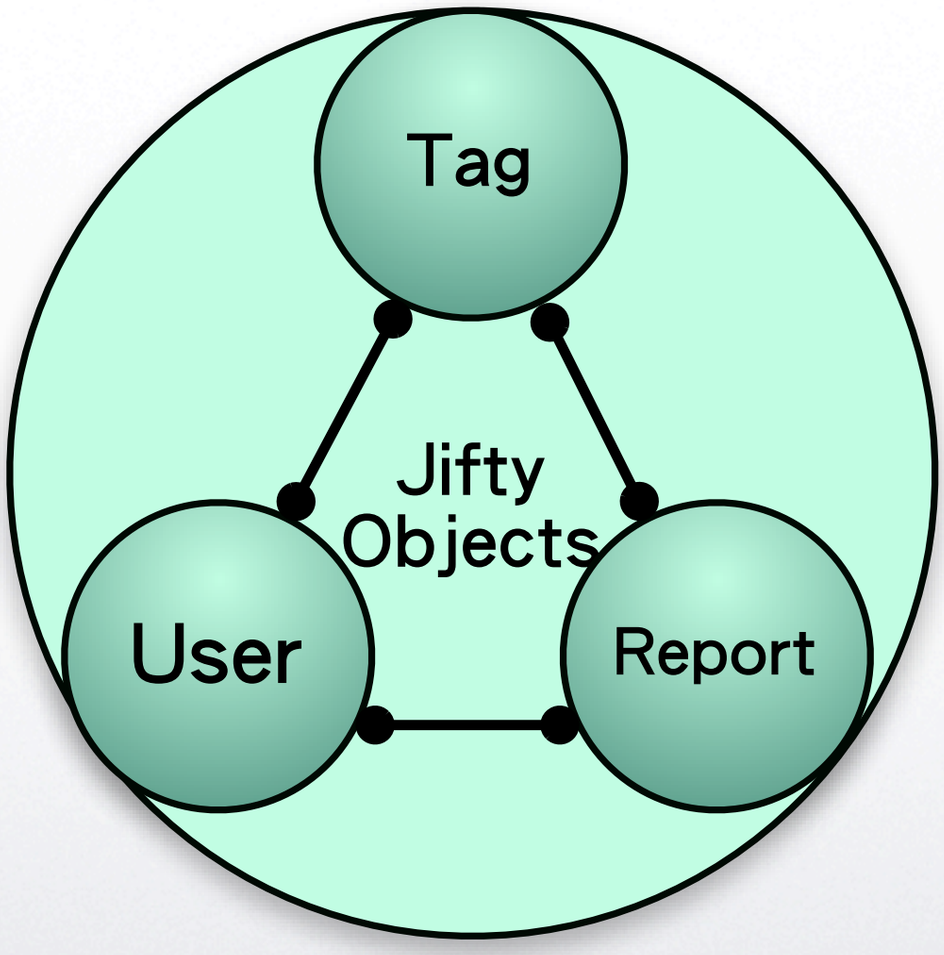
- *"Everything is a tree."*
- XHTML
- XPath/XSLT/XQuery

Admin UI
(HTTP + XHTML)

Web API
(HTTP + JSON)

Audit Log
(XML)

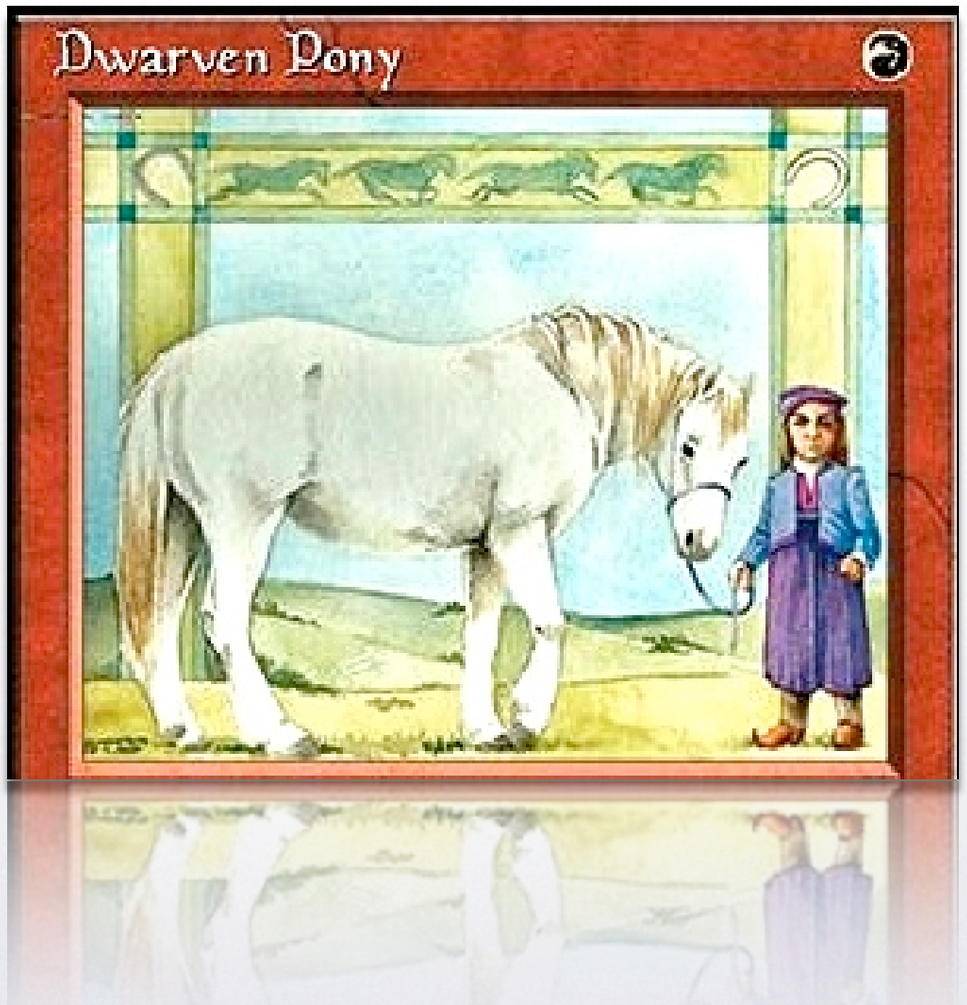
User Info
(XML)



DB2
(SQL)

Object Store
(SQLite)

Jifty



Everything is Perl

Everything is Perl

- CPAN is the language

Everything is Perl

- CPAN is the language
- Write once, run anywhere

Everything is Perl

- CPAN is the language
- Write once, run anywhere
- Fit syntax to the brain

Mini Languages

Mini Languages

- Pure Perl syntax

Mini Languages

- Pure Perl syntax
- *Not* source filtering

Mini Languages

- Pure Perl syntax
- *Not* source filtering
- Declare your schema

Mini Languages

- Pure Perl syntax
- *Not* source filtering
- Declare your schema
- Everything *becomes* an object

```
package RPS::Model::Report;
use Jifty::DBI::Schema;
use RPS::Record schema {

column name =>
    label is 'Name',
    is mandatory;

column owner =>
    refers to RPS::Model::Tag,
    label is 'Owner';

column last_updated =>
    type is 'date',
    render as 'Date',
    label is 'Last Updated';
    filters are Jifty::DBI::Filter::Date;

};
```

```
package RPS::Action::CreateReport;
use Jifty::Action::Schema;
use RPS::Action schema {

# Auto-inherits from the model's params
# Override and add new params here..
param owner =>
    render as 'Combobox',
    available are {
        display_from => 'description',
        value_from => 'id',
        collection => defer {
            Jifty->web->current_user->tags
        },
    };

};
```

Jifty RHOX!

RPS 1.0

Production

Database Callback

- Relational
- Hypertext
- Object
- XML

Audit Log

Audit Log

- Track all Tag/User/Report changes

Audit Log

- Track all Tag/User/Report changes
- Who made this change?

Audit Log

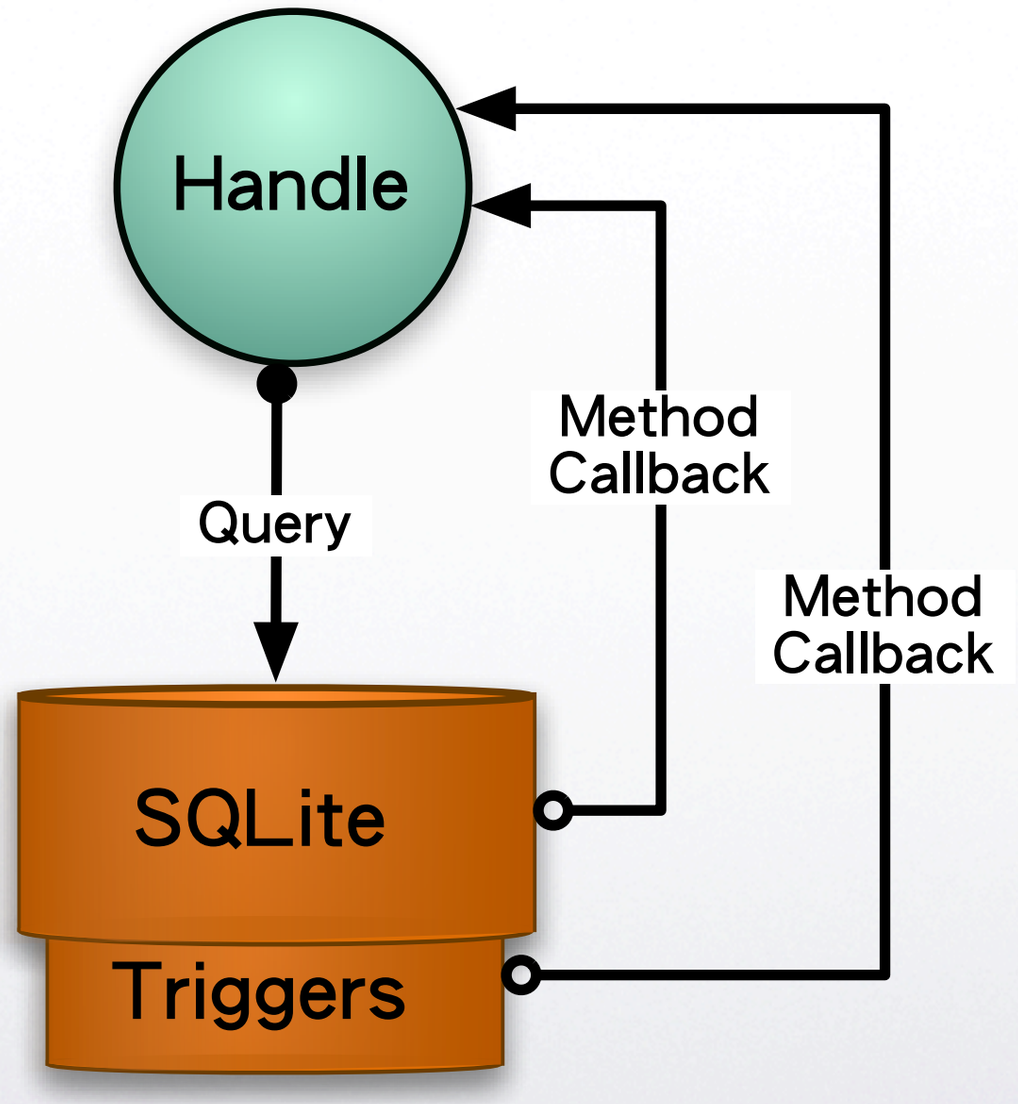
- Track all Tag/User/Report changes
- Who made this change?
- When did this change happen?

Audit Log

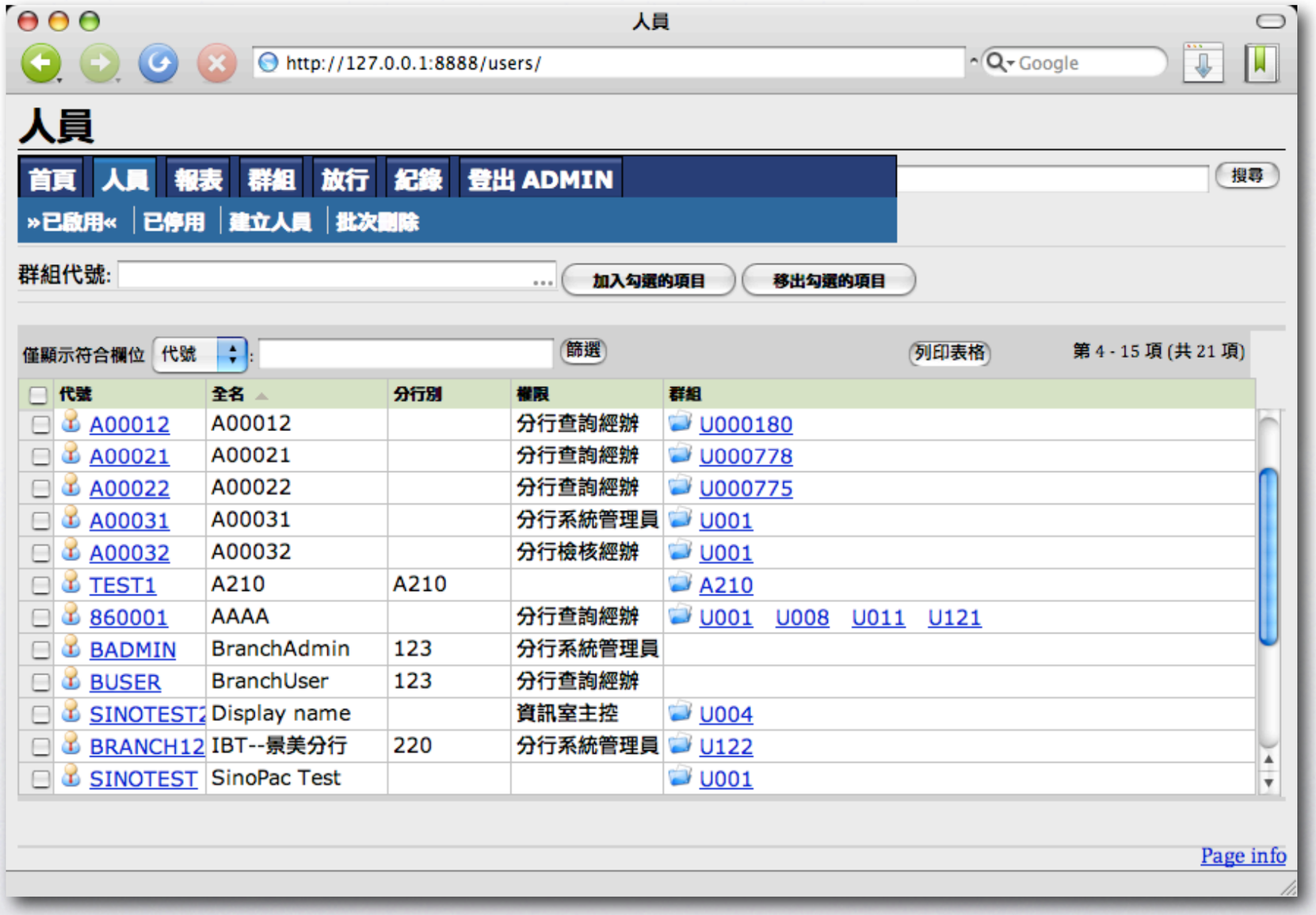
- Track all Tag/User/Report changes
- Who made this change?
- When did this change happen?
- Which users were affected?

```
-- Generated code: Do not edit!
CREATE TRIGGER LogTagReportsI
  AFTER INSERT ON tag_reports
  BEGIN INSERT INTO logs (
    request_id, authorizer,
    affected_users, date_time, row
  ) VALUES (
    REQUEST_ID(),          -- Jifty->handle->request_id
    USER_ID(),             -- Jifty->handle->user_id
    TAG_USERS(new.tag), -- Jifty->handle->tag_users
    DATETIME('now', 'localtime'),
    new.id,
  );
END;
```

Jifty::Handle::Callback

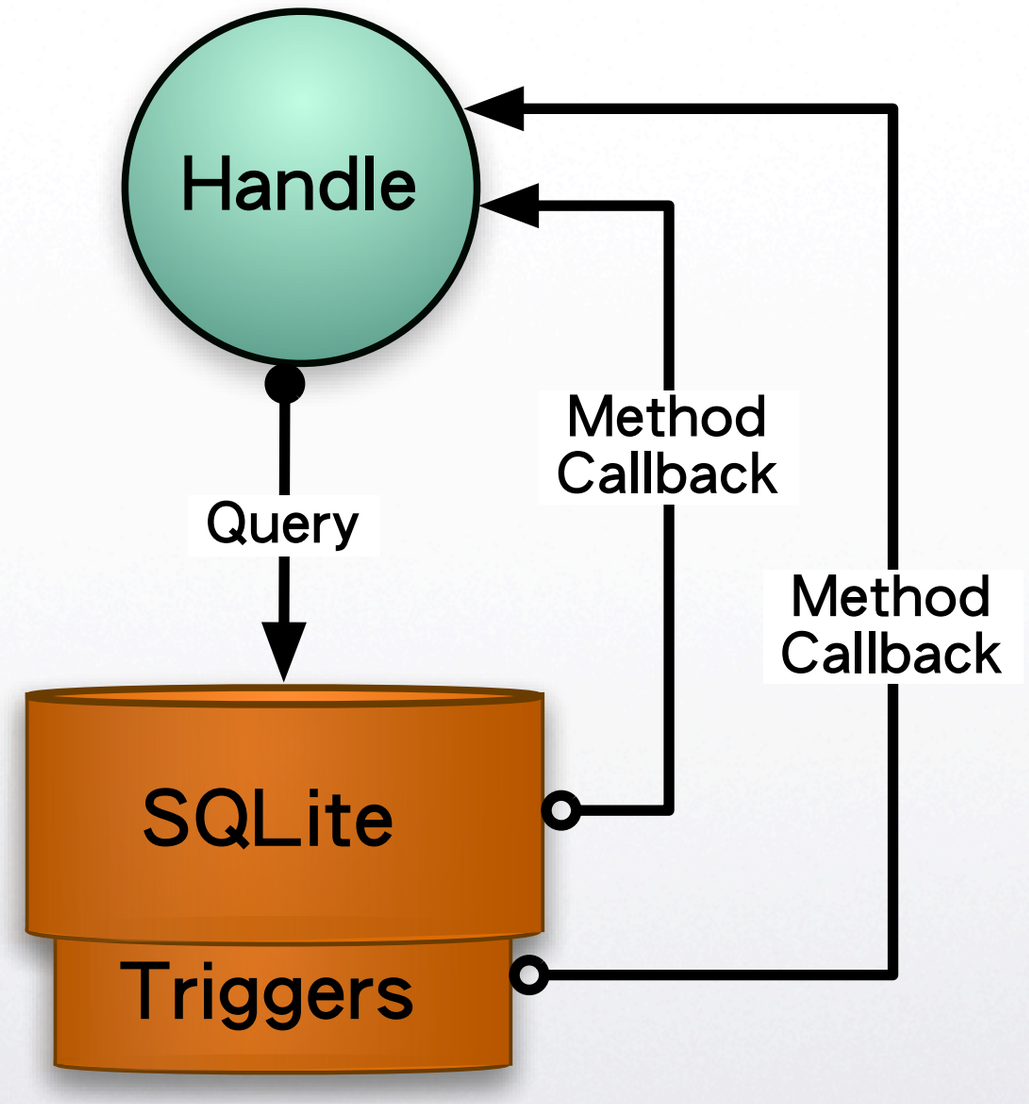


Custom Views



```
-- Generated code: Do not edit!
CREATE VIEW view_users AS
  SELECT DISTINCT users.id AS id,
         users.name AS Name,
         LOC_LEVEL(users.level) AS Level,
         (SELECT GROUP_CONCAT('tags', tags.name)
            FROM tags, tag_users
           WHERE tag_users.tag = tags.id
             AND tag_users.user = users.id
             AND IS_VISIBLE('tag’, tags.name)
        ORDER BY tags.name) AS Groups
    FROM users
GROUP BY users.id
```

How does it work?



```
package RPS::Handle;
use Moose;

# Class inheritance (“is”)
extends 'Jifty::Handle';

# Role mix-in (“does”)
with 'Jifty::Handle::Callback';

# All methods are exported to SQLite
sub user_id    { ... }
sub request_id { ... }
sub loc_level  { ... }
sub tag_users  { ... }
sub is_visible { ... }
```

```
package Jifty::Handle::Callback;
use Moose::Role;

after connect => sub {
    my $self = shift;
    my $dbh  = $self->dbh;

    # Find all methods with ->meta reflection
    for my $meth ($self->meta->get_method_list) {
        $dbh->func(
            $meth,                              # name
            -1,                                 # arg count
            sub { $self->$meth(@_) }, # callback
            'create_function',
        );
    }
};
```

REST API

Relational

Hypertext

Object

XML

Model Feeds

Model Feeds

- "Export as…" for all objects

Model Feeds

- "Export as…" for all objects
- Exploratory interface

Model Feeds

- "Export as…" for all objects
- Exploratory interface
- .html, .js, .yml, .csv, .perl

```
% GET http://user:pass@host/=/model/user
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};
```

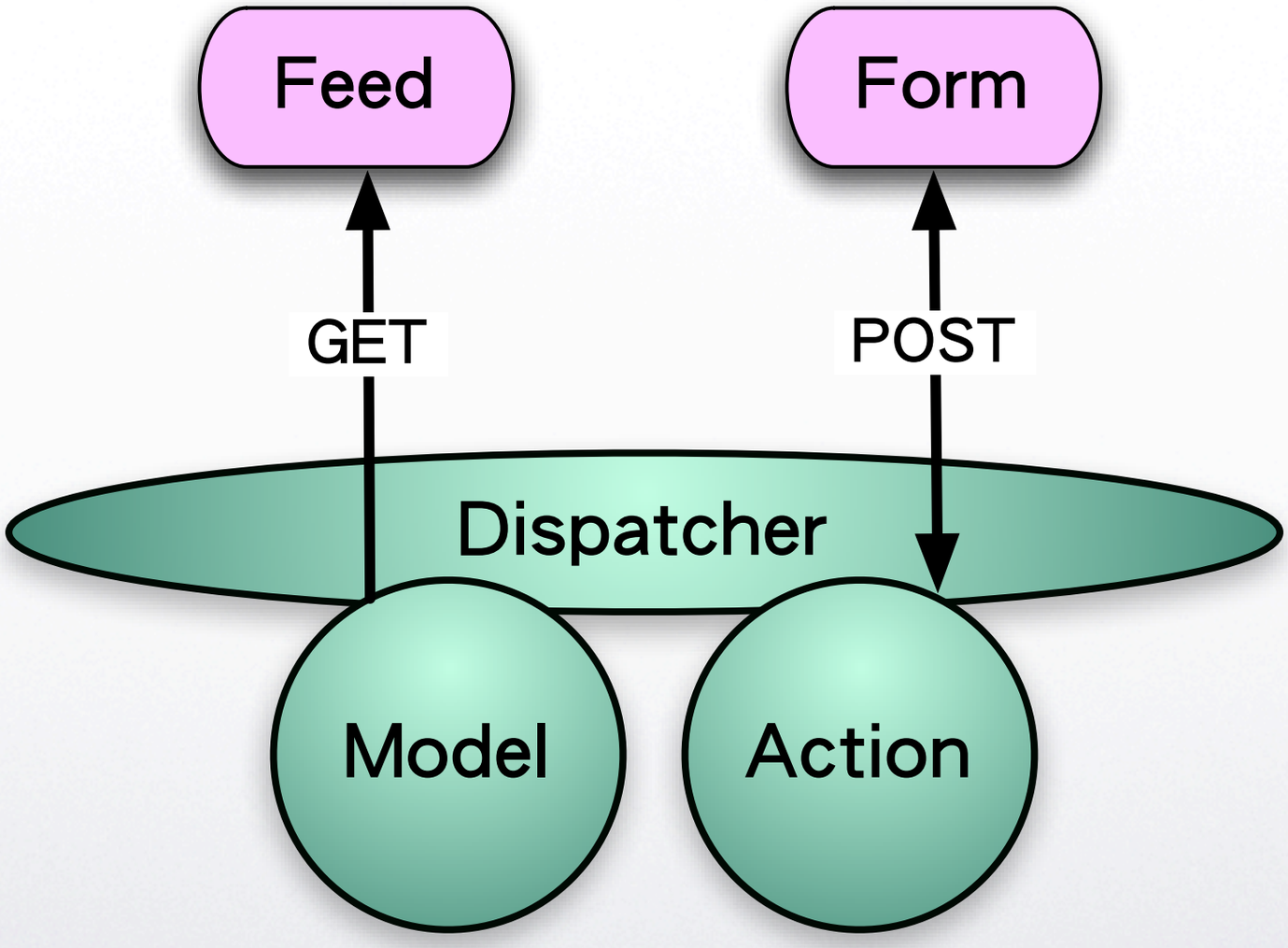
```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.js
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.yml
```

```
% GET http://user:pass@host/=/model/user
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Model::User</title></head><body><ul>
<li><a href="user/id">id</a></li>
<li><a href="user/name">name</a></li>
</ul></body></html>

% GET http://user:pass@host/=/model/user.js
var $_ = ['id', 'name'];

% GET http://user:pass@host/=/model/user/id.js
var $_ = [1, 2, 3, 4, 5];

% GET http://user:pass@host/=/model/user/id/1.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.js
var $_ = {'level':'99','name':'ADMIN','id':1,'description':'管理員'};

% GET http://user:pass@host/=/model/user/name/ADMIN.yml
---
description: "管理員"
id: 1
level: 99
name: ADMIN
```

Jifty::Plugin::REST



Action Forms

Action Forms

- Expose API via POST

Action Forms

- Expose API via POST
- Describe parameters via GET

Action Forms

- Expose API via POST
- Describe parameters via GET
- Also supports multiple formats

```
% GET http://user:pass@host/=/action/create_user.js
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
var $_ = ['Created'];
```

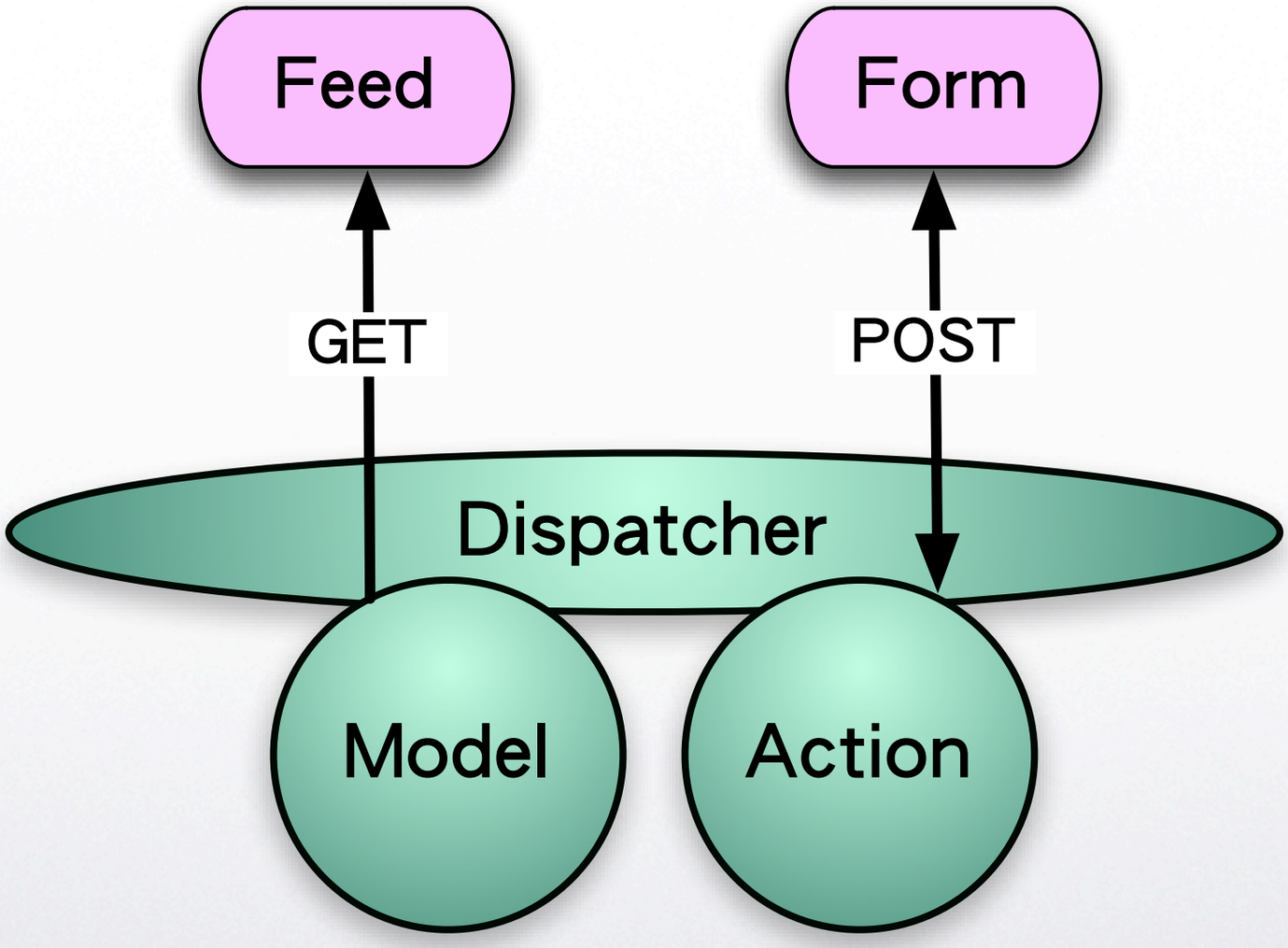
```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
var $_ = ['Created'];

% POST http://user:pass@host/=/action/create_user
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
var $_ = ['Created'];

% POST http://user:pass@host/=/action/create_user
Please enter content (application/x-www-form-urlencoded) to be POSTed:
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
var $_ = ['Created'];

% POST http://user:pass@host/=/action/create_user
Please enter content (application/x-www-form-urlencoded) to be POSTed:
name=guest;level=1;description=Guest
^D
```

```
% GET http://user:pass@host/=/action/create_user.js
var $_ = {
 'name':{'sort_order':'0','mandatory':'1','label':'Name'},
 'description':{'sort_order':'1','label':'Description'},
 'level':{'sort_order':'2','label':'Level',
   'default_value':'1','render_as':'Select','valid_values':
   [{'value':'99','display':'System Admin'},
    {'value':'20','display':'HQ User'},
    {'value':'10','display':'Branch User'},
    {'value':'-1','display':'Disabled'}]}};

% POST http://user:pass@host/=/action/create_user.js
Please enter content (application/x-www-form-urlencoded) to be POSTed:
{'name':'guest','level':1,'description':'Guest'}
^D
var $_ = ['Created'];

% POST http://user:pass@host/=/action/create_user
Please enter content (application/x-www-form-urlencoded) to be POSTed:
name=guest;level=1;description=Guest
^D
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>RPS::Action::CreateUser</title></head><body>OK<ul>
<li>Created</li>
</ul></body></html>
```

How does it work?



```
package Jifty::Plugin::REST::Dispatcher;
use Jifty::Dispatcher -base;

before '/=/**' => run {
    authenticate_user();
    tangent('/login') unless Jifty->web->current_user->id;
};

before qr{^ (/=/.*) \. (js|yml|perl|csv) $}x => run {
    header(Accept => mime_type($2));
    dispatch $1;
};

on GET    '/=/model'         => \&list_models;
on GET    '/=/model/*'       => \&list_model_keys;
on GET    '/=/model/*/*'     => \&list_model_items;
on GET    '/=/model/*/*/*'   => \&show_item;
on PUT    '/=/model/*/*/*'   => \&edit_item;
on DELETE '/=/model/*/*/*'   => \&delete_item;
on GET    '/=/action'        => \&list_actions
on GET    '/=/action/*'      => \&list_action_params;
on POST   '/=/action/*'      => \&run_action;
```

XML Expressions

Relational

Hypertext

Object

XML

Generating XML

Generating XML

- Each $client wants their own format

Generating XML

- Each $client wants their own format
- Ad-hoc schema changes every week

Generating XML

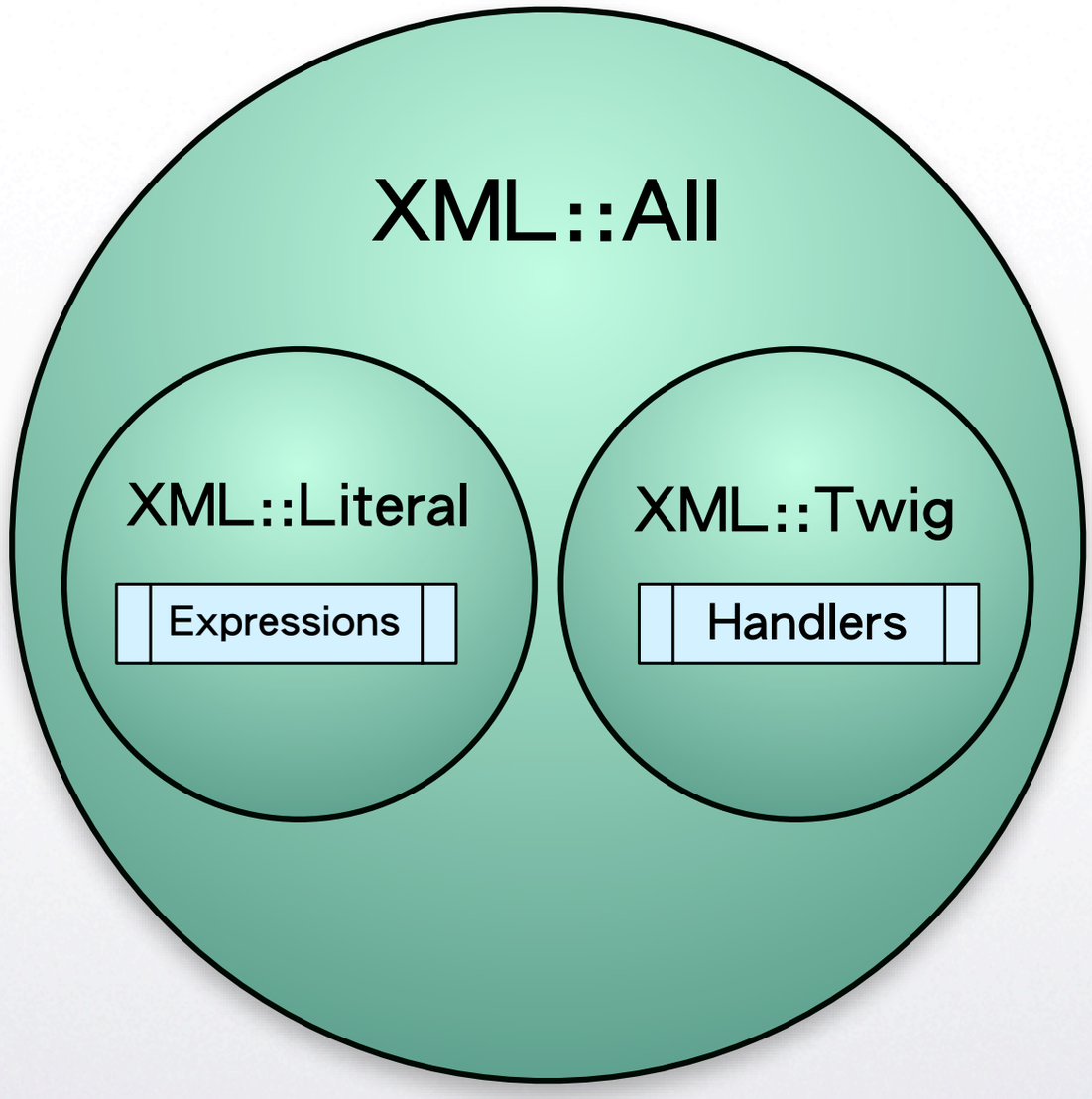
- Each $client wants their own format
- Ad-hoc schema changes every week
- Lots of reusable fragments

Generating XML

- Each $client wants their own format
- Ad-hoc schema changes every week
- Lots of reusable fragments
- So let's make the fragments Perl!

```perl
# This is not a source filter!
use XML::Twig;
use XML::Literal sub { XML::Twig->new->parse($_[0]) };

# Simple element
my $xml1 = <hr/>;

# With variable interpolation
my $type = 'submit';
my $xml2 = <input type='$type'/>;

# With an extra pair of angle brackets
my $xml3 = < <a href='/'> Some Text: $xml2 </a> >;
```

XML::All



Processing XML

Processing XML

- Parse User Info from $client

Processing XML

- Parse User Info from $client
- Lots of ad-hoc editing

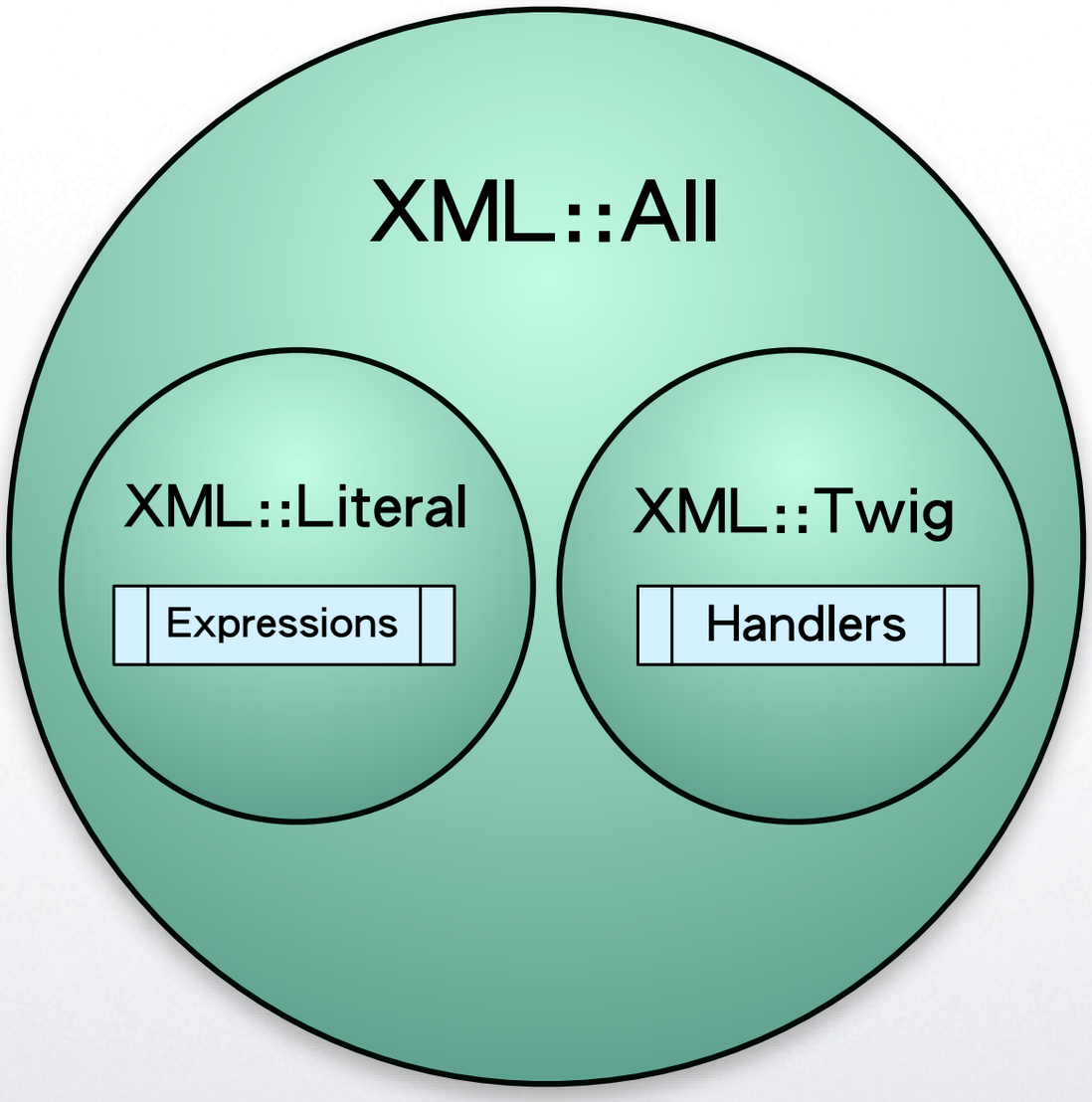
Processing XML

- Parse User Info from $client
- Lots of ad-hoc editing
- Need lazy streams for huge trees

```perl
use XML::All;

my $xml = < <a href='/'>1 <b>2</b> <em>3</em></a> >;

print $$xml;                # a
print join ",", @$xml; # 1, <b>2</b>, <em>3</em>
print join ",", %$xml; # href, '/'

print $xml->b();            # <b>2</b>
print $xml->b() * 10;   # <b>20</b>
print $xml->();             # 1

$$xml = 'link';
print $xml;                 # <link href='/'>...</link>

$xml->em() += <hr/>;
print $xml->em();           # <em>3<hr/></em>
```

```perl
use XML::All;

my $xml = xml 'input.html';

# In-place modification on the XML stream
$xml->(
    html => sub {
        $$_ = 'pdftemplate'; # Change tag
        %$_ = ();
    },
    center => sub {
        $_->{align} = 'center' for $_->p;
        undef $$_;              # Remove node
    },
    <a href/> => sub {
        $$_ = 'span';           # Change tag
        $links{ $_->{href} }++;
    },
    # ...
);
```

How does it work?



```
# in XML::Literal::import
*CORE::GLOBAL::glob = sub {
    if ($_[0] =~ m{^\s*<}) {
        # Looks like < <tag> >
        goto &$callback;
    }
    elsif ($_[0] =~ m{/\w*\s*$}) {
        # Looks like <tag/> or <tag\>...\</tag>
        @_ = "<$_[0]>";
        goto &$callback;
    }
    else {
        # Looks like file globbing
        goto &File::Glob::glob;
    }
};
```

RPS 2.0

Experimental

RPS 2.0

Experimental

Native Query

- Relational
- Hypertext
- Object
- XML

Jifty::DBI

Jifty::DBI

- Jifty's default O/R mapper

Jifty::DBI

- Jifty's default O/R mapper
- Write in Perl, generates SQL

Jifty::DBI

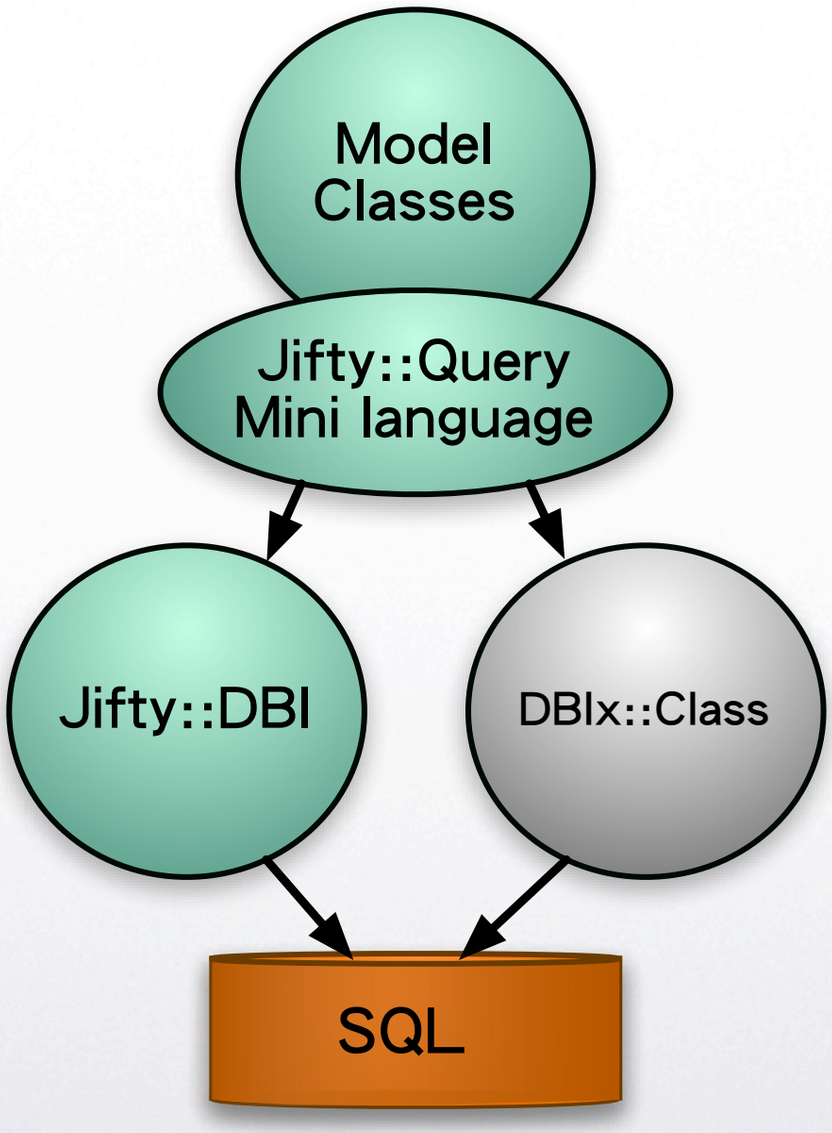
- Jifty's default O/R mapper
- Write in Perl, generates SQL
- But you still need to *think* in SQL

Jifty::DBI

- Jifty's default O/R mapper
- Write in Perl, generates SQL
- But you still need to *think* in SQL
- Awkward concept of *iterators*

```perl
my $users = RPS::Model::UserCollection->new;

$users->limit({
    column => 'name',
    operator => 'like',
    value => '%moose',
});

$users->order_by(
    { column => 'level' },
    { column => 'name' },
);

while (my $user = $users->next) {
    # ...
}
```

Jifty::Query


Model
Classes
Jifty::Query
Mini language
Jifty::DBI
DBIx::Class
SQL

Query Language

Query Language

- Lazily filled arrays

Query Language

- Lazily filled arrays
- Simple Perl statements

Query Language

- Lazily filled arrays
- Simple Perl statements
- Much more refactorable

```
my @users = fetch {
    model User =>
        name like '%moose',
        order by qw( level name );
};
```

```
for my $user (@users) {
    # ...
}
```

Joins and Filters

Joins and Filters

- Uniform dot syntax

Joins and Filters

- Uniform dot syntax
- Uses *refers_to* information

Joins and Filters

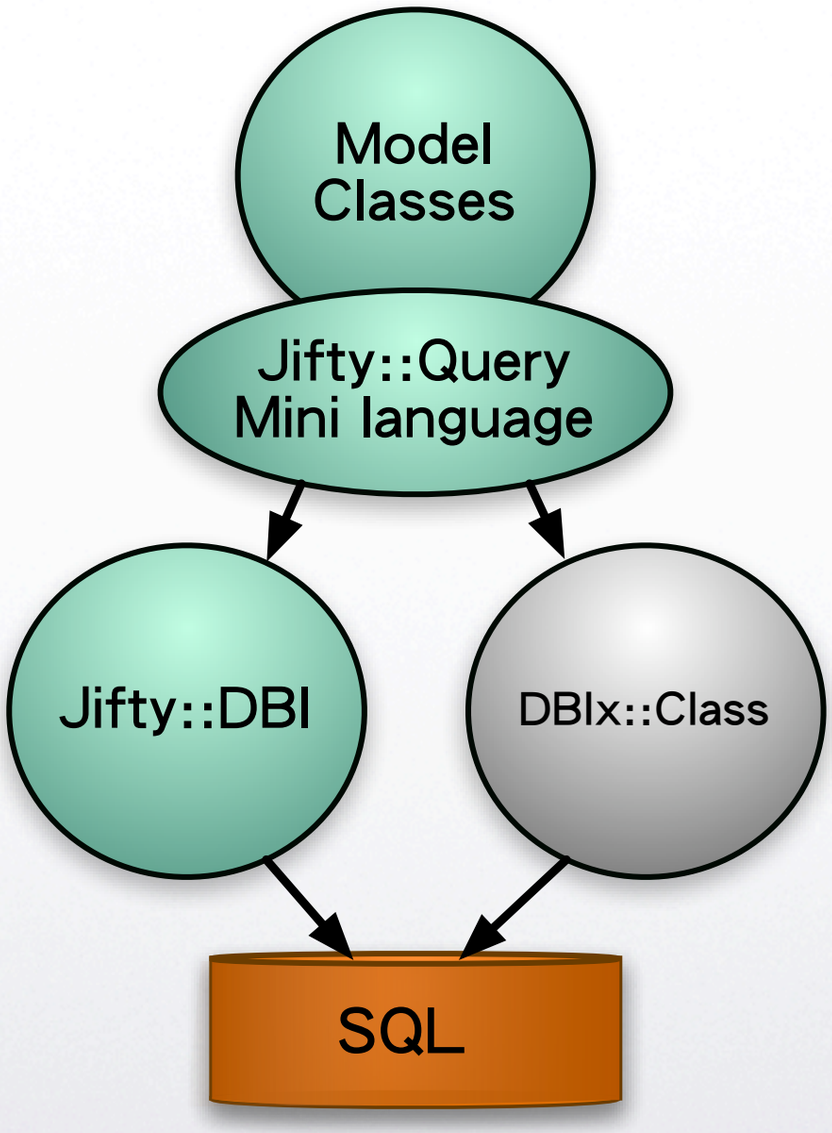
- Uniform dot syntax
- Uses *refers_to* information
- Callbacks to Perl code

Joins and Filters

- Uniform dot syntax
- Uses *refers_to* information
- Callbacks to Perl code
- Much better than *->join* calls

```
my @users = fetch {
    model User =>
        tag.name like 'web20'
        filter { -e '/authorized/'.name },
        order by id;
};

for my $user (@users) {
    # ...
}
```

How does it work?



```perl
use Object::Declare ['MyApp::Column', 'MyApp::Param'];

my %objects = declare {

param foo =>
   !is global,
    is immutable,
    valid_values are qw( more values );

column bar =>
    field1 is 'value',
    field2 is 'some_other_value';

};

print $objects{foo}; # a MyApp::Param object
print $objects{bar}; # a MyApp::Column object
```

```perl
use Object::Declare ['MyApp::Column', 'MyApp::Param'];

my %objects = declare {

param('foo',
   !'global'->is,          # UNIVERSAL::is
    'immutable'->is,
    'are'->valid_values('more', 'values')
);
column('bar',
    is->field1('value'), # is::AUTOLOAD
    is->field2('some_other_value');
);
};

print $objects{foo}; # a MyApp::Param object
print $objects{bar}; # a MyApp::Column object
```

Client-side Sessions

- Relational
- Hypertext
- Object
- XML

Server sessions

Server sessions

- Cookie stores *session key*

Server sessions

- Cookie stores *session key*
- Actual data stored on disk

Server sessions

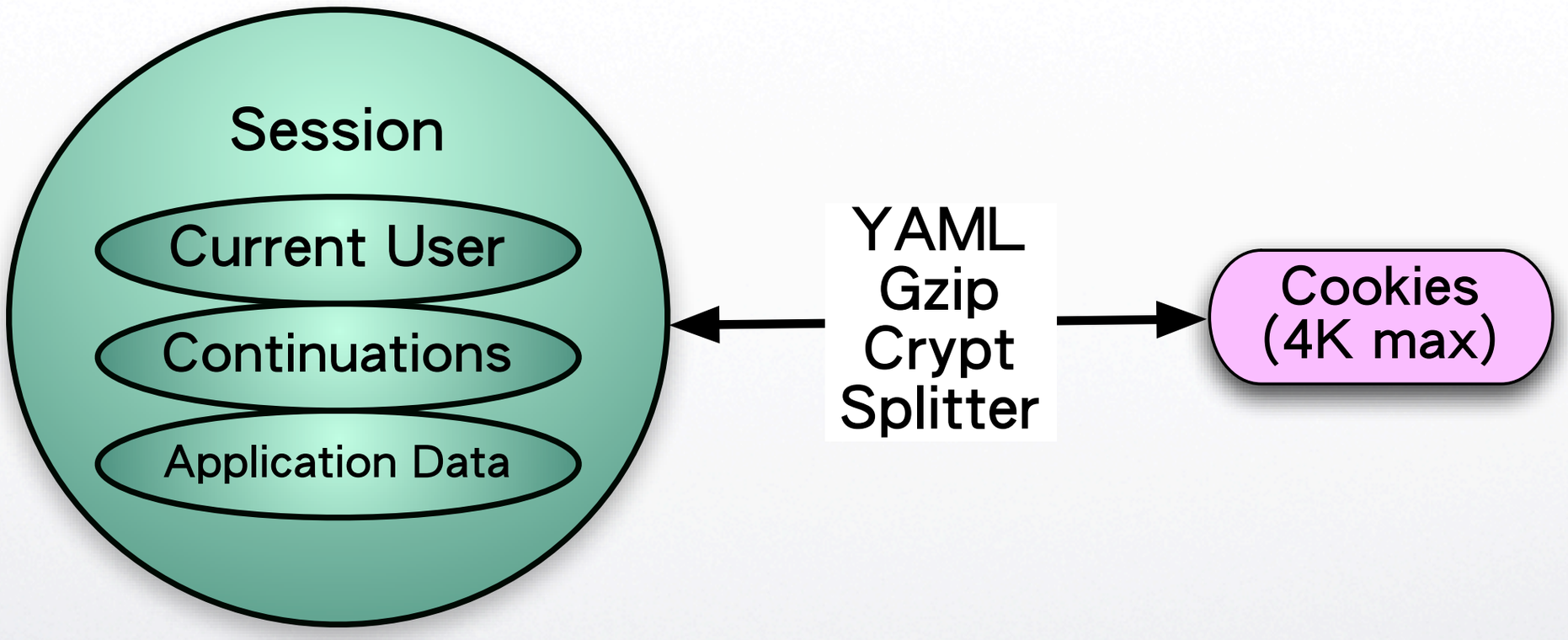
- Cookie stores *session key*
- Actual data stored on disk
- IO and expiry are expensive

Server sessions

- Cookie stores *session key*
- Actual data stored on disk
- IO and expiry are expensive
- …especially on 233MHZ/AIX

```
package Jifty::Model::Session;
use Jifty::DBI::Schema;
use Jifty::Record schema {

column session_id => type is 'varchar(32)';
column data_key   => type is 'text';
column value      => type is 'blob',
 filters are 'Jifty::DBI::Filter::Storable';
column created    => type is 'timestamp',
 filters are 'Jifty::DBI::Filter::DateTime';
column updated    => type is 'timestamp',
 filters are 'Jifty::DBI::Filter::DateTime';
column key_type   => type is 'varchar(32)';

};
```

Jifty::Web::Session::ClientSide



State in Cookies

State in Cookies

- Typical session is ~300 bytes

State in Cookies

- Typical session is ~300 bytes
- Encrypt with server secret

State in Cookies

- Typical session is ~300 bytes
- Encrypt with server secret
- Cut across 4K boundaries

State in Cookies

- Typical session is ~300 bytes
- Encrypt with server secret
- Cut across 4K boundaries
- Zero disk IO on the server

```yaml
# etc/config.yml
---
framework:
  Web:
    SessionClass: Jifty::Web::Session::ClientSide
    SessionSecret: fnord
```

Client Continuations

Client Continuations

- Per-session call stack

Client Continuations

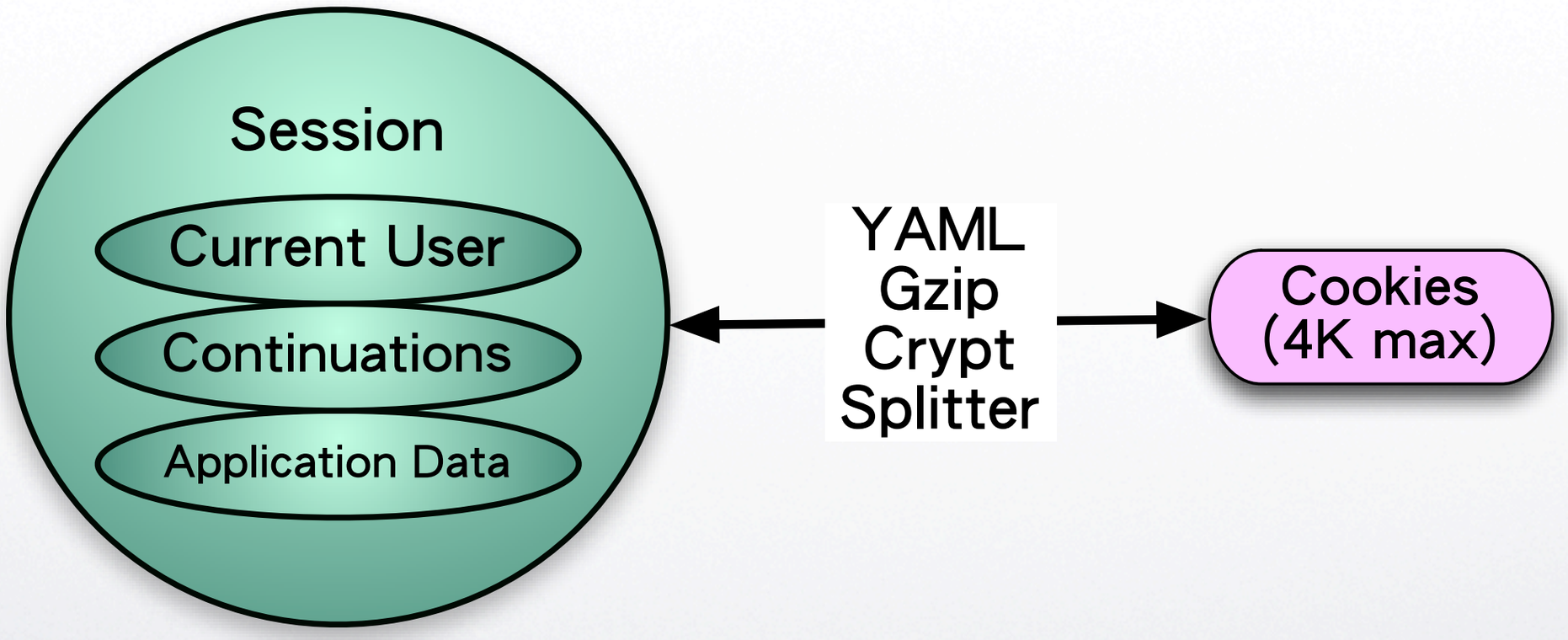
- Per-session call stack
- Perfect for server clusters

Client Continuations

- Per-session call stack
- Perfect for server clusters
- Even for *deferred* closures

```
package RPS::Dispatcher;
use Jifty::Dispatcher -base;

before '/restricted/**' => run {
    tangent('/login')
        unless Jifty->web->current_user->id;
}

on 'login' => run {
    authenticate_user();
    Jifty->web->return(to => '/');
        if Jifty->web->current_user->id;
};
```

How does it work?



```
package Jifty::Web::Session::ClientSide;
use base 'Jifty::Web::Session';

use YAML::Syck ();
use Crypt::CBC ();
use Compress::Zlib ();
use Crypt::Rijndael ();
use CGI::Cookie::Splitter ();
```

```
my $splitter = CGI::Cookie::Splitter->new;
my $secret = Jifty->config->framework('Web')
                        ->{SessionSecret};
my $cipher = Crypt::CBC->new(
    -key => $secret,
    -cipher => 'Rijndael',
);
```

```
sub load {
    my %cookies = CGI::Cookie->fetch();
    my $cookie_name = $self->cookie_name;
    my $session_id = $cookies{$cookie_name};

    my ($data) = grep {
        $_->name eq "JIFTY_DAT_$session_id"
    } $splitter->join(values %cookies);

    my $session = (
        YAML::Syck::Load(
            Compress::Zlib::uncompress(
                $cipher->decrypt(
                    $data->value
                )
            )
        )
    );
}
```

View Classes

- Relational
- Hypertext
- Object
- XML

Mason & TT2

Mason & TT2

- Abstract over XHTML

Mason & TT2

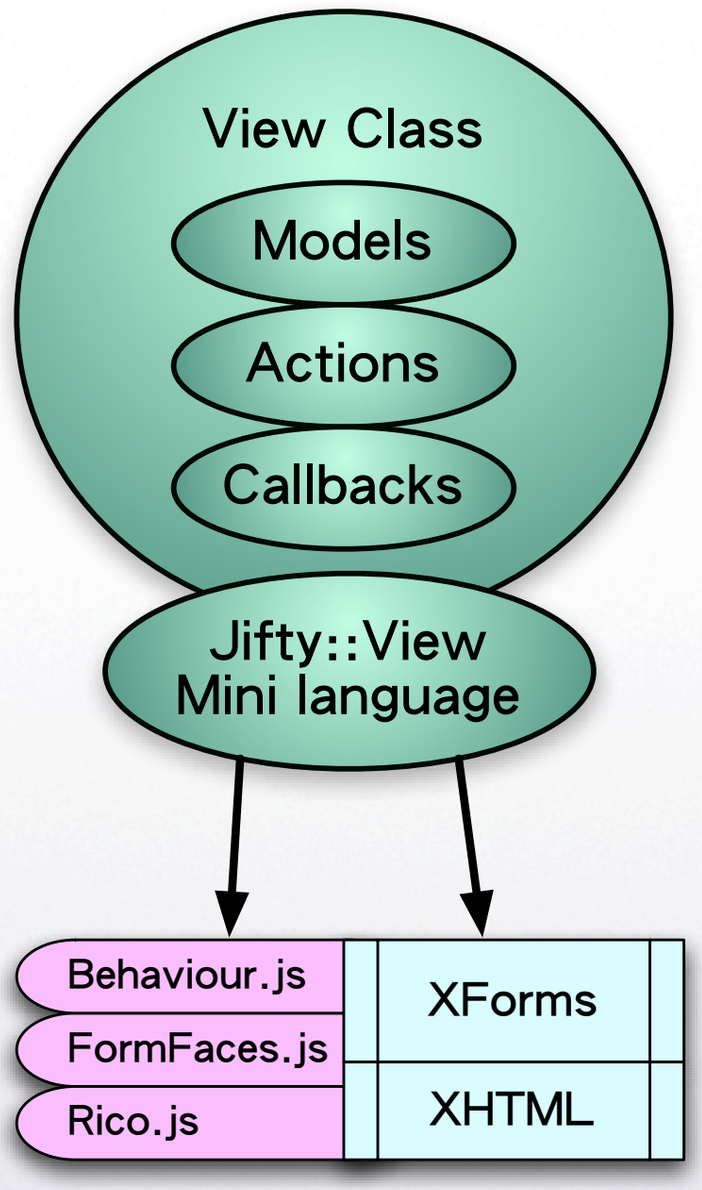
- Abstract over XHTML
- Pseudo Perl syntax

Mason & TT2

- Abstract over XHTML
- Pseudo Perl syntax
- *Non-interactive* output

```
<%ARGS>
$collection # From the dispatcher
</%ARGS>

<&|/_elements/wrapper&>
% while (my $item = $collection->next) {
  <h2><% $item->name %></h2>
  <div class="description">
    <% $item->description %>
  </div>
% }
</&>
```

Jifty::View::Class



Views with Class

Views with Class

- Inheritable methods

Views with Class

- Inheritable methods
- XForms via FormFaces

Views with Class

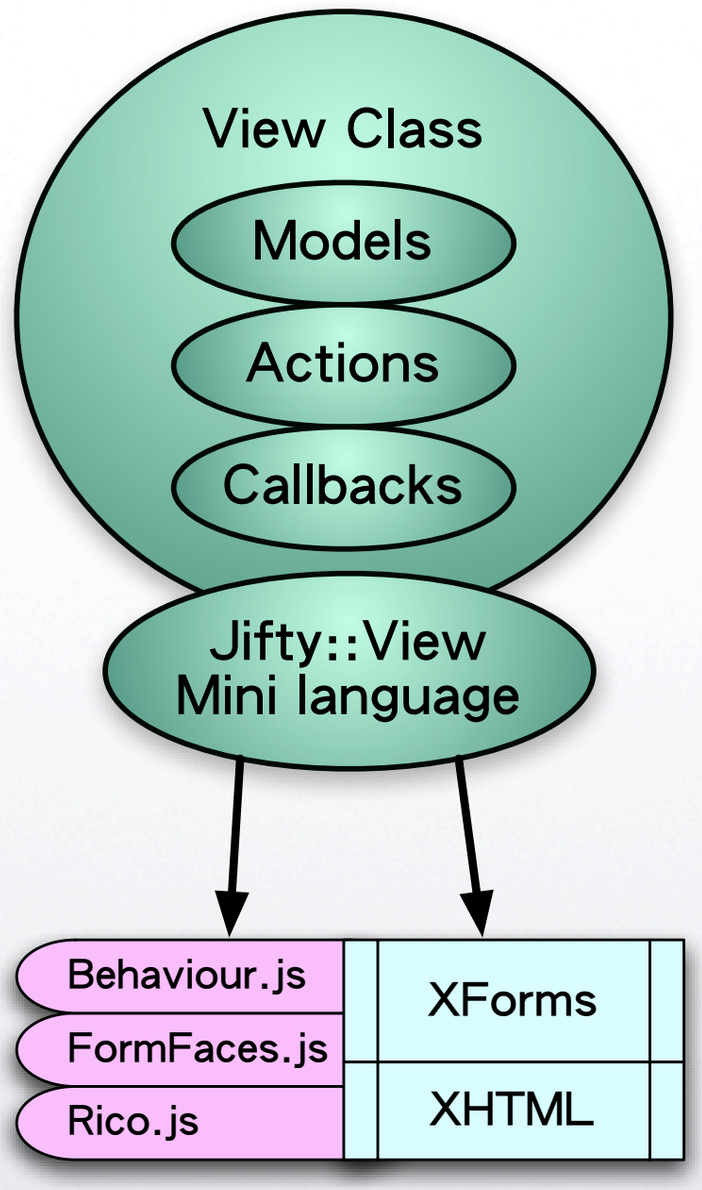
- Inheritable methods
- XForms via FormFaces
- "Native" JS widgets

Views with Class

- Inheritable methods
- XForms via FormFaces
- "Native" JS widgets
- Callbacks aka GWT

```
use RPS::View::Class schema {

button 'test' =>
    # defer { loc('Click me') }
    label is _('Click me'),

    # defer { run ... }
    on click run {
        Jifty->log->debug('User clicked me');
        Jifty->web->tangent(url => '/next');
    };

};
```

How does it work?



```perl
use Scalar::Defer; # This does not use tie()!

# a deferred value (not memoized)
my $x = 0;
my $dv = defer { ++$x };
print "$dv $dv $dv"; # 1 2 3

# a lazy value (memoized)
my $y = 0;
my $lv = lazy { ++$y };
print "$lv $lv $lv"; # 1 1 1

# force a normal value out of $dv
my $forced = force $dv;
print "$forced $forced $forced"; # 4 4 4
```

```perl
package Scalar::Defer;
use Class::InsideOut qw( private register id );

sub defer (&) {
    my $cv = shift;
    my $obj = register( bless \(my $s) );
    $_defer{ id $obj } = $cv;
    bless($obj => 0);
}

# Set up overload for the package "0"
overload::OVERLOAD(
    '0' => fallback => 1, map {
        $_ => sub { &{$_defer{ id $_[0] }} }
    } qw( bool "" 0+ ${} @{} %{} &{} *{} )
);

# Handle any ->method on the deferred value
*{"0::AUTOLOAD"} = sub { ... };
```

As seen on #perl6...

```
<TimToady> or maybe it's overloaded somehow...
<kolibrie> its overloaded
<TimToady> which overloading, I wonder...all of them?
<TimToady> seems like the only way it'd work.
<kolibrie> use overload fallback => 1, map { $_ => \&force }
           qw( bool "" 0+ ${} @{} %{} &{} *{} );
<TimToady> Perl is a scary language.
```

Conclusion