Building a RESTful Web Service API in YiiFramework

Truong-An Thai @bithai

01-23-2013

http://www.meetup.com/Austin-Yii/

RESTful Web Services Overview

- RESTful web API is a web service implemented using HTTP and principles of REST.
- Collection of resources with base URI
- Uses HTTP methods explicitly (e.g. GET, PUT, POST or DELETE)
- Typically supports data formats like JSON, XML for request and response
- http://en.wikipedia.org/wiki/Representational_state_transfer

API Resource Design

http method API Uri Base Version Resource

GET http://api.meetup.com/2/events

GET http://api.meetup.com/2/events/:id

POST http://api.meetup.com/2/events

PUT http://api.meetup.com/2/events/:id

DELETE http://api.meetup.com/2/events/:id

HTTP - CRUD

POST	add data	**C**REATE
GET	retrieve / search data	**R**EAD
PUT	update data	**U**PDATE
DELETE	delete data	**D**ELETE

HTTP Error Codes

200 - OK
201 - Created
204 - No Content
400 - Bad Request
401 - Unauthorized
403 - Forbidden
404 - Not Found
405 - Method Not Allowed
500 - Internal Server Error

Sending Requests

- Method Resource URI
  POST /events

- Body (Headers - Content-Type: application/json)

```
{
  “title”:  ”Building RESTFul API in Yii”,
  “event_date”:  ”2013-01-23”,
  “created_by”:  “jefftulsa”
}
```

Responses

- Request
  GET /events/123
- `Json Response`

```
{
    "id":"123",
    "title":"Building RESTFul API in Yii",
    "created_by":"jefftulsa",
    "created_at":"2012-11-03",
    "attending":"4"
}
```

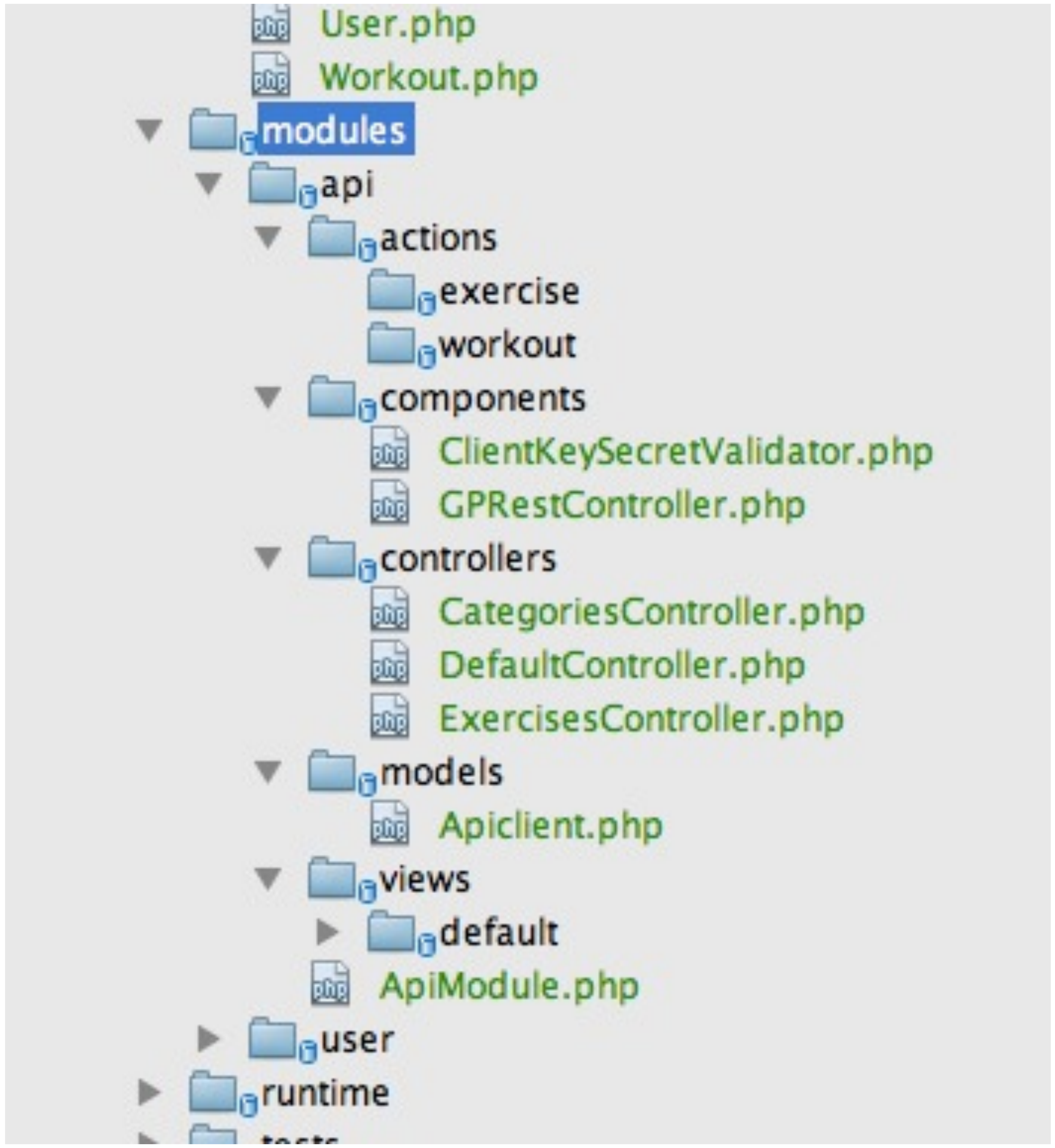
Why build RESTful API using Yii?

- Cuz doing things the Yii way rocks
- Advanced URL Routings
- API Module
- One Controller per Resource
- Controller maps URI to CAction classes
- Model for each resource representation
- CDbCriteria Compare (Search / Filter)
- CModel Behaviors to add API specific code

Url Resources Routing

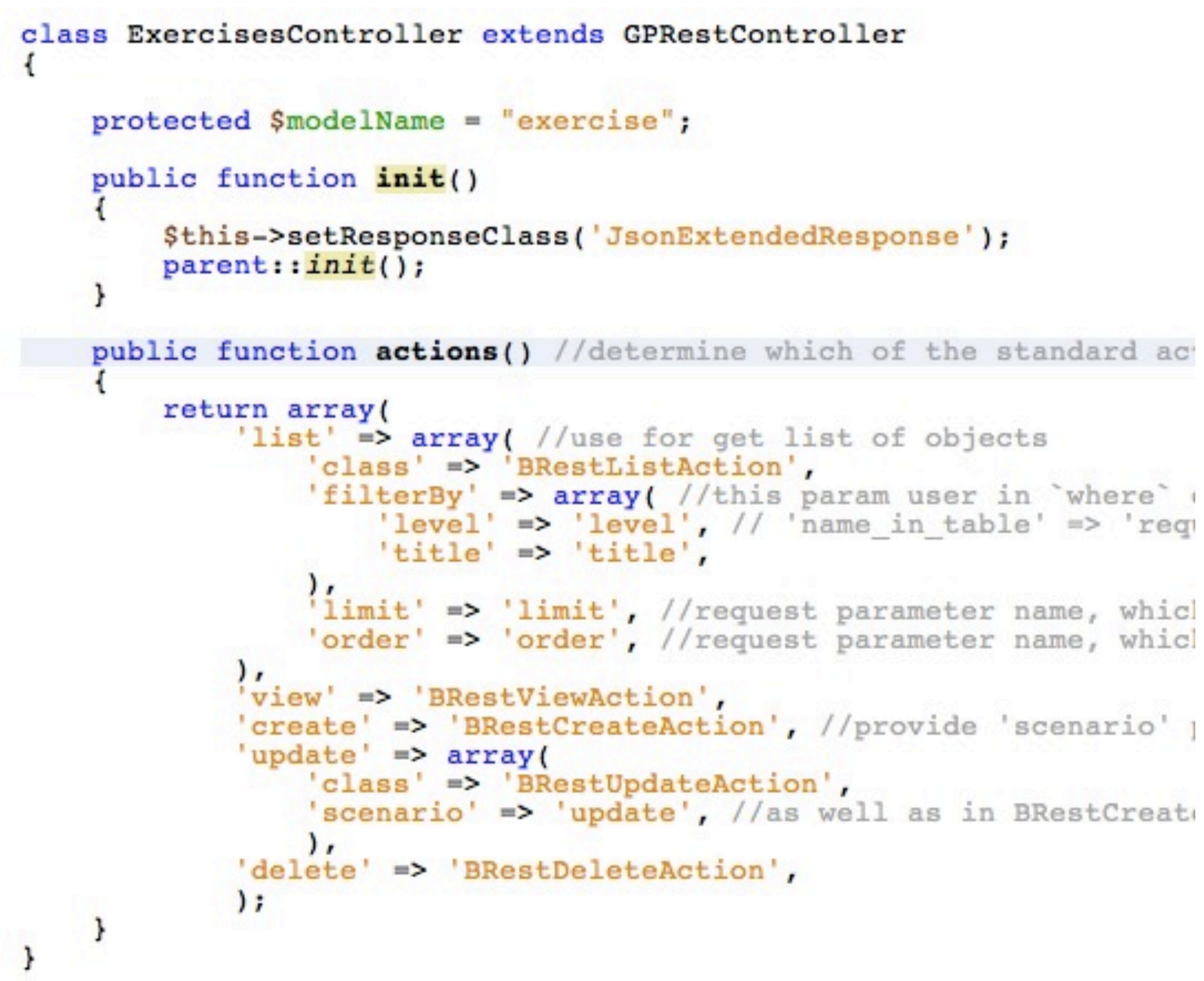
```
'urlManager' => array(
        'urlFormat' => 'path',
        'caseSensitive' => false,
        'showScriptName' => false,
        'rules' => array( ........
                //rest url patterns
                array('api/<resource>/list',
                        'pattern'=>'api/<version>/<resource:\w+>', 'verb'=>'GET'),

                array('api/<resource>/view',
                        'pattern'=>'api/<version>/<resource:\w+>/<id:\d+>', 'verb'=>'GET'),

                array('api/<resource>/create',
                       'pattern'=>'api/<version>/<resource:\w+>', 'verb'=>'POST'),

                array('api/<resource>/delete',
                       'pattern'=>'api/<version>/<resource:\w+>/<id:\d+>', 'verb'=>'DELETE'),

                array('api/<resource>/update',
                       'pattern'=>'api/<version>/<resource:\w+>/<id:\d+>', 'verb'=>'PUT'),
```

API Module



```
        User.php
        Workout.php
▼ modules
    ▼ api
        ▼ actions
                exercise
                workout
        ▼ components
                ClientKeySecretValidator.php
                GPRestController.php
        ▼ controllers
                CategoriesController.php
                DefaultController.php
                ExercisesController.php
        ▼ models
                Apiclient.php
        ▼ views
            ▶ default
            ApiModule.php
    ▶ user
▶ runtime
```

Resource Controller



```
class ExercisesController extends GPRestController
{

    protected $modelName = "exercise";

    public function init()
    {
        $this->setResponseClass('JsonExtendedResponse');
        parent::init();
    }

    public function actions() //determine which of the standard ac
    {
        return array(
            'list' => array( //use for get list of objects
                'class' => 'BRestListAction',
                'filterBy' => array( //this param user in `where`
                    'level' => 'level', // 'name_in_table' => 'req
                    'title' => 'title',
                ),
                'limit' => 'limit', //request parameter name, whic
                'order' => 'order', //request parameter name, whic
            ),
            'view' => 'BRestViewAction',
            'create' => 'BRestCreateAction', //provide 'scenario'
            'update' => array(
                'class' => 'BRestUpdateAction',
                'scenario' => 'update', //as well as in BRestCreat
                ),
            'delete' => 'BRestDeleteAction',
            );
    }
}
```

Yii CAction Component

```
public function actions()
{
        return array(
            'list' => array(
                'class' => 'BRestListAction',
            ),
            'view' => 'BRestViewAction',
            'create' => 'BRestCreateAction',
            'update' => array(
                'class' => 'BRestUpdateAction',
                'scenario' => 'update',
            ),
            'delete' => 'BRestDeleteAction',
            );
    }
```

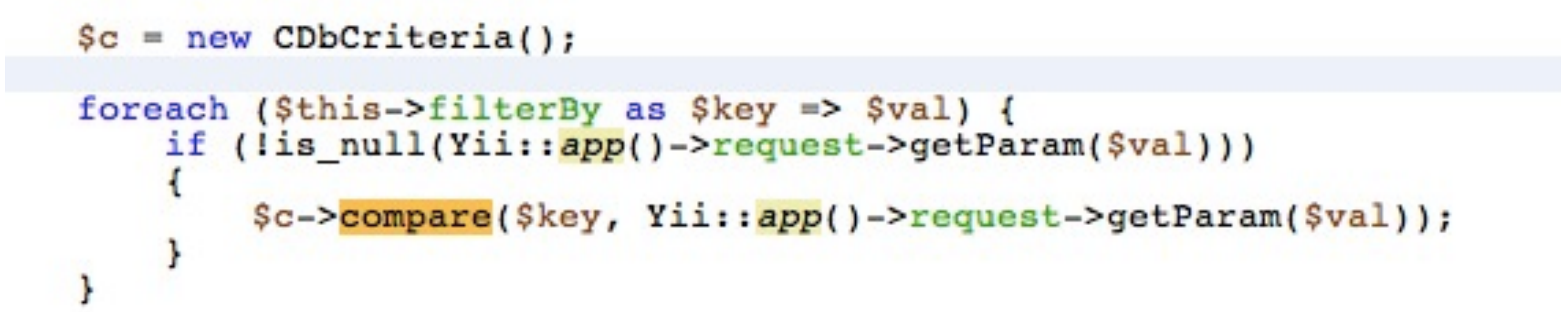
http://www.yiiframework.com/wiki/170/actions-code-reuse-with-caction/

Search / Filter on fields

CDbCriteria comapare()

The comparison operator is intelligently determined based on the first few characters in the given value. In particular, it recognizes the following operators if they appear as the leading characters in the given value:

- <: the column must be less than the given value.
- >: the column must be greater than the given value.
- <=: the column must be less than or equal to the given value.
- >=: the column must be greater than or equal to the given value.
- <>: the column must not be the same as the given value. Note that when $partialMatch is true, this would mean the value must not be a substring of the column.
- =: the column must be equal to the given value.
- none of the above: the column must be equal to the given value. Note that when $partialMatch is true, this would mean the value must be the same as the given value or be a substring of it.



```
$c = new CDbCriteria();

foreach ($this->filterBy as $key => $val) {
    if (!is_null(Yii::app()->request->getParam($val)))
    {
        $c->compare($key, Yii::app()->request->getParam($val));
    }
}
```

http://www.yiiframework.com/doc/api/1.1/CDbCriteria

Yii CModel Behaviors

```
class BRestModelBehavior extends CActiveRecordBehavior {

    protected $ownerModelAPIResponseMethod = 'getAPIResponseData';

    public function getAttributesForResponse()
    {
        $owner = $this->getOwner();
        $params = Yii::app()->getController()->restRequest->getParams();
        if(method_exists($owner, $this->ownerModelAPIResponseMethod)) {
            return $owner->getAPIResponseData($params);
        }
        else {
            return $owner->getAttributes();
        }
    }
}
```

Using BRestModelBehavior

```
// implement behaviors() function in the Model class to leverage behavior
public function behaviors() {
     return array(
          'BRestModelBehavior' => 'ext.brestapi.BRestModelBehavior',
     );
 }
```

```
public function getAPIResponseData($params=array())
{
        $attributeNames = array('id', 'first_name', 'last_name', 'about');

        // get the attribute name and values from this model
        $attributes = $this->getAttributes($attributeNames);

        // modify/add additional attributes here (optional)
        $attributes['about'] = trim($this->about);
        $attributes['full_name'] = $this->first_name.' '.$this->last_name;
        $attributes['images'] = $this->getThumbUrlArray();

        return $attributes;

    }
```

Demo

- Requests
- Response
- BRestAPI code walk thru

Links

- https://github.com/bithai/brestapi
  (lacking documentation, will have update within a week)
- http://www.yiiframework.com/wiki/175/how-to-create-a-rest-api/
  (works but not elegant / modular / extensible)

LEARN MORE ABOUT RESTful API:

- http://en.wikipedia.org/wiki/Representational_state_transfer
- http://blog.apigee.com/detail/slides_for_restful_api_design_second_edition_webinar/
- http://blog.apigee.com/detail/rest_api_design_for_sql_programmers/
- http://www.meetup.com/meetup_api/