Capítulo 32. Zend_Rest

Traduzido por Flávio Gomes da Silva Lisboa (Zend Framework versão 1.5.2)

Sumário

32.1. Introdução

REST Web Services usam formatos XML específicos para serviços. Esses padrões ad-hoc significam que a maneira para acessar um REST web service é diferente para cada serviço. REST web services tipicamente usam parâmetros de URL (método GET) ou informação de caminho para requisitar dados e método POST para enviar dados.

Zend Framework provê tanto as capacidades de Cliente quanto de Servidor, o que, quando usado junto permite uma experiência de interface muito mais "local" via acesso a propriedades de objetos virtuais. O componente servidor apresenta exposição automática de funções e classes usando um formato XML simples e significativo. Quando acessar esses serviços usando o Cliente, é possível recuperar facilmente os dados devolvidos da chamada remota. Se você deve desejar usar o cliente com um serviço não baseado em um Zend_Rest_Server, ainda assim ele proverá acesso mais fácil aos dados.

32.2. Zend_Rest_Client

32.2.1. Introdução

Usar o `Zend_Rest_Client` é muito similar a usar objetos `SoapClient` (extensão SOAP web service). Você simplesmente chama os prodecimentos do REST como métodos `Zend_Rest_Client`. Especifique o endereço completo dos serviços no construtor de `Zend_Rest_Client`.

Exemplo 32.1. Uma requisição REST básica

```
<?php
/**
 * Conecta-se ao servidor framework.zend.com e recupera um cumprimento
 */
require_once 'Zend/Rest/Client.php';
```

```php
$client = new Zend_Rest_Client('http://framework.zend.com/rest');
echo $client->sayHello('Davey', 'Day')->get(); // "Hello Davey, Good Day"
```

Diferenças na chamada

`Zend_Rest_Client` tenta fazer métodos remotos parecerem mais com métodos nativos quanto possível, a única diferença é que você deve seguir a chamada de método com um dos métodos `get()`, `post()`, `put()` ou `delete()`. Essa chamada pode ser feita via encadeamento de método ou em chamadas de método separadas:

```php
<?php
$client->sayHello('Davey', 'Day');
echo $client->get();
```

32.2.2. Respostas

Todas as requisições feitas usando `Zend_Rest_Client` retornam um objeto `Zend_Rest_Client_Response`. Esse objeto tem muitas propriedades que tornam mais fácil o acesso aos resultados.

Quando o serviço é baseado em `Zend_Rest_Server`, `Zend_Rest_Client` pode fazer várias assunções sobre a resposta, incluindo o status da resposta (sucesso ou falha) e o tipo de retorno.

Exemplo 32.2. Status da Resposta

```php
<?php
$result = $client->sayHello('Davey', 'Day')->get();
if ($result->isSuccess()) {
    echo $result; // "Hello Davey, Good Day"
}
```

No exemplo acima, você pode ver que nós usamos o resultado da requisição como um objeto, para chamar `isSuccess()`, e então por causa de `__toString()`, nós simplesmente usamos `echo` sobre o objeto para obter o resultado. `Zend_Rest_Client_Response` permitirá que você exiba qualquer valor escalar. Para tipos complexos, você pode usar a notação de matriz ou de objeto.

Se entretanto, você desejar consultar um serviço sem usar `Zend_Rest_Server` o objeto `Zend_Rest_Client_Response` comportar-se-á mais como um `SimpleXMLElement`. Contudo, para tornar as coisas mais fáceis, ele irá automaticamente consultar o XML usando XPath se a propriedade não é um descendente direto do elemento raiz do documento. Adicionalmente, se você acessar uma propriedade como um método, irá receber o valor PHP para o objeto, ou uma matriz de resultados de valores PHP.

Exemplo 32.3. Usando Technorati's Rest Service

```php
<?php
require_once 'Zend/Rest/Client.php';
```

```php
$technorati = new Zend_Rest_Client('http://api.technorati.com/bloginfo');
$technorati->key($key);
$technorati->url('http://pixelated-dreams.com');
$result = $technorati->get();
echo $result->firstname() .' '. $result->lastname();
```

Exemplo 32.4. Exemplo de Resposta Technorati

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- generator="Technorati API version 1.0 /bloginfo" -->
<!DOCTYPE tapi PUBLIC "-//Technorati, Inc.//DTD TAPI 0.02//EN" "http://api.techn
orati.com/dtd/tapi-002.xml">
<tapi version="1.0">
    <document>
        <result>
            <url>http://pixelated-dreams.com</url>
            <weblog>
                <name>Pixelated Dreams</name>
                <url>http://pixelated-dreams.com</url>
                <author>
                    <username>DShafik</username>
                    <firstname>Davey</firstname>
                    <lastname>Shafik</lastname>
                </author>
                <rssurl>http://pixelated-dreams.com/feeds/index.rss2</rssurl>
                <atomurl>http://pixelated-dreams.com/feeds/atom.xml</atomurl>
                <inboundblogs>44</inboundblogs>
                <inboundlinks>218</inboundlinks>
                <lastupdate>2006-04-26 04:36:36 GMT</lastupdate>
                <rank>60635</rank>
            </weblog>
            <inboundblogs>44</inboundblogs>
            <inboundlinks>218</inboundlinks>
        </result>
    </document>
</tapi>
```

Aqui nós estamos acessando as propriedades `firstname` e `lastname`. Mesmo quando esses não são elementos de alto nível, eles são automaticamente retornados quando acessados pelo nome.

Múltiplos itens

Se múltiplos itens são encontrados quando se acessa um valor por nome, uma matriz de SimpleXMLElements será retornada; acessar via notação de método irá retornar uma matriz de valores PHP.

32.2.3. Argumentos de Requisição

A menos que você esteja fazendo uma requisição para um serviço baseado em um `Zend_Rest_Server`, ocorrerá que você irá precisar enviar múltiplos argumentos com sua requisição. Isso é feito pela chamada a um método com o nome do argumento, passando o valor como primeiro argumento (e único). Cada uma dessas chamadas de método retorna o próprio objeto, permitindo o encadeamento, ou uso “fluente”. A primeira chamada, ou o primeiro

argumento se você passar mais de um, é sempre assumido como sendo o método quando chamar um serviço `Zend_Rest_Server`.

Exemplo 32.5. Configurando Argumentos de Requisição

```
<?php
$client = new Zend_Rest_Client('http://example.org/rest');
$client->arg('value1');
$client->arg2('value2');
$client->get();
// ou
$client->arg('value1')->arg2('value2')->get();
```

Ambos os métodos no exemplo acima, irão resultar nos seguintes argumentos GET:

```
?method=arg&arg1=value1&arg=value1&arg2=value2
```

Você notará que a primeira chamada de `$client->arg('value1');` resultou em ambos `method=arg&arg1=value1` e `arg=value1`; isso é assim pois `Zend_Rest_Server` pode compreender a requisição apropriadamente, melhor do que requerendo conhecimento pré-existente do serviço.

Exatidão de Zend_Rest_Client

Qualquer REST service que é estrito sobre os argumento que recebe irá provavelmente falhar usando `Zend_Rest_Client`, por causa do comportamento descrito acima. Isso não é uma prática comum e não deve causar problemas.

32.3. Zend_Rest_Server

32.3.1. Introdução

Zend_Rest_Server é pretendido como um servidor REST completamente caracterizado.

32.3.2. Uso de REST Server

Exemplo 32.6. Uso Básico de Zend_Rest_Server - Classes

```
<?php
require_once 'Zend/Rest/Server.php';
require_once 'My/Service/Class.php';
$server = new Zend_Rest_Server();
$server->setClass('My_Service_Class');
$server->handle();
```

Exemplo 32.7. Uso Básico de Zend_Rest_Server - Funções

```
<?php
```

```
require_once 'Zend/Rest/Server.php';
/**
 * Say Hello
 *
 * @param string $who
 * @param string $when
 * @return string
 */
function sayHello($who, $when)
{
    return "Hello $who, Good $when";
}
$server = new Zend_Rest_Server();
$server->addFunction('sayHello');
$server->handle();
```

32.3.3. Chamando um Serviço Zend_Rest_Server

Para chamar um serviço `Zend_Rest_Server` você deve suprir um argumento GET/POST `method` com um valor que seja o método que você deseja chamar. Você pode então prosseguir com qualquer número de argumentos usando ou o nome do argumento (por exemplo, “who”) ou usando `arg` seguido pela posição numérica do argumento (por exemplo "arg1").

Índice Numérico

Argumentos numéricos usam um índice baseado em 1.

Para chamar `sayHello` do exemplo acima, você pode usar:

```
?method=sayHello&who=Davey&when=Day
```

ou:

```
?method=sayHello&arg1=Davey&arg2=Day
```

32.3.4. Enviando um Status Customizado

Quando retornar valores, para retornar um status customizado, você pode retornar uma matriz com uma chave `status`.

Exemplo 32.8. Retornando Status Customizados

```
<?php
require_once 'Zend/Rest/Server.php';
/**
 * Say Hello
 *
 * @param string $who
 * @param string $when
 * @return array
 */
function sayHello($who, $when)
{
    return array('msg' => "An Error Occurred", 'status' => false);
```

```
}
$server = new Zend_Rest_Server();
$server->addFunction('sayHello');
$server->handle();
```

32.3.5. Retornando Respostas XML Customizadas

Se você desejar retornar XML customizada, simplesmente retorne um objeto `DOMDocument`, `DOMElement` ou `SimpleXMLElement`.

Exemplo 32.9. Retorna XML Customizado

```
<?php
require_once 'Zend/Rest/Server.php';
/**
 * Say Hello
 *
 * @param string $who
 * @param string $when
 * @return SimpleXMLElement
 */
function sayHello($who, $when)
{
    $xml ='<?xml version="1.0" encoding="ISO-8859-1"?>
<mysite>
    <value>Hey $who! Hope you're having a good $when</value>
    <code>200</code>
</mysite>';
    $xml = simplexml_load_string($xml);
    return $xml;
}
$server = new Zend_Rest_Server();
$server->addFunction('sayHello);
$server->handle();
```

A resposta do serviço será retornado sem modificação para o cliente.