

Serviços e Interoperabilidade de Sistemas

Web Services RESTful (PHP/Yii2) Módulos e Autenticação

Web Services RESTful (PHP/Yii2)

»» Módulos

Serviços Web RESTful

Módulos

Unidades de software que podem conter controllers, views, models, etc.

Utilidade

Reutilização mais facilitada...

Exemplo: módulo para gestão de utilizadores

Para separar a parte Web da parte REST e até diferentes versões da API REST

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

http://localhost:8888/gii **(Pretty Urls!)**

Create Module

Module class: app\modules\v1\Module

Module ID: **v1**

Preview

Desligar Views

Generate

O gii indica-nos o código a colocar no web.php para registar novo module:

```
'modules' => [  
    'v1' => [  
        'class' => 'app\modules\v1\Module',  
    ],  
],
```

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

```
<?php

$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';

$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm' => '@vendor/npm-asset',
    ],

    'modules' => [
        'v1' => [
            'class' => 'app\modules\v1\Module',
        ],
    ],

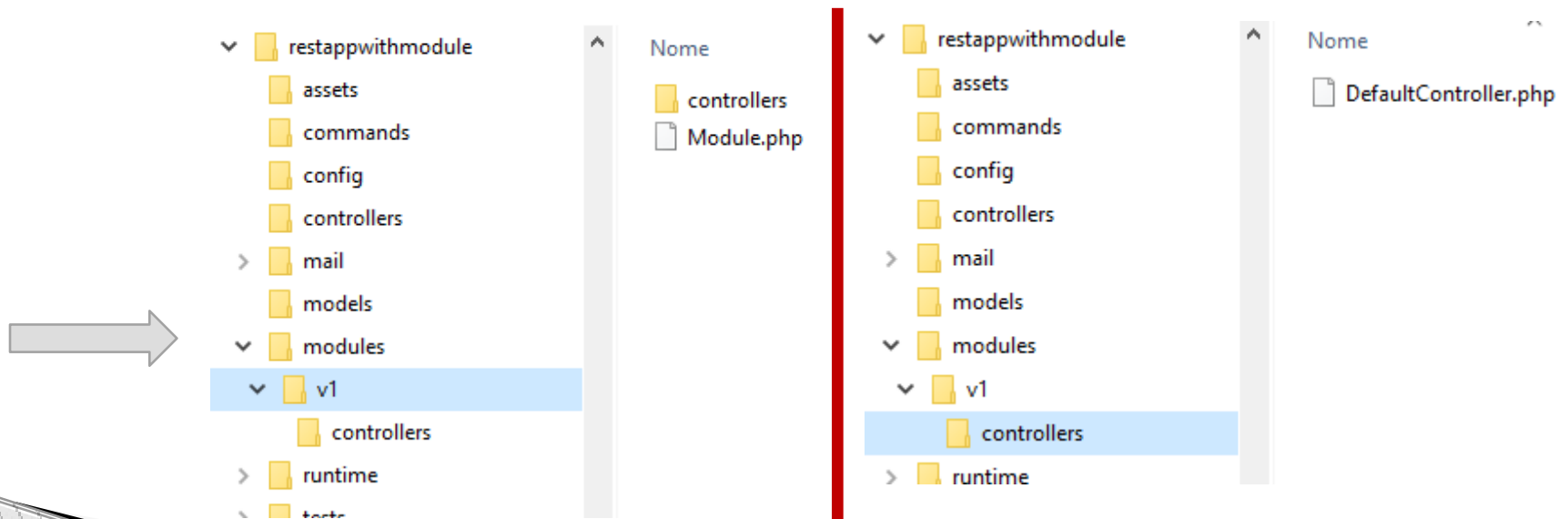
    'components' => [
        'request' => [
```

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

É criada a seguinte estrutura:



Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

Vamos criar model:

Table Name: clientes

Model Class: Clientes

(Namepsace: app\models ou dentro do módulo)

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

Modificar controlador criado (é do tipo web):

```
namespace app\modules\v1\controllers;

class DefaultController extends \yii\rest\ActiveController
{
    public $modelClass = 'app\models\Clientes';
}
```

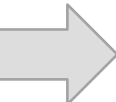

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

Dar a conhecer rotas do novo controlador RESTful

```
'urlManager' => [  
    'enablePrettyUrl' => true,  
    'showScriptName' => false,  
    'rules' => [  
        [  
            'class' => 'yii\rest\UrlRule',  
            'controller' => 'v1/default',  
            'pluralize' => false,  
        ],  
    ],  
],
```



Podíamos ter alterado o nome do controlador para Clientes ou criado outro

Serviços Web RESTful

Módulos

Exemplo: Vamos criar módulo v1 (versão 1.0 da REST API)

Teste:

http://localhost:8888/v1/default GET

Web Services RESTful (PHP/Yii2)

» Autenticação

Serviços Web RESTful

Autenticação

Aplicações Web: Sessions, Cookies, etc...

RESTful API: **stateless!**

Cada pedido terá que conter informação de autenticação

Normalmente cada pedido transporta um **token secreto de acesso** para autenticar o cliente, mas:

ataque de segurança: man-in-the-middle !

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth (username + password)
2. Query parameter (token apenas)
3. OAuth 2 (fora do âmbito desta UC)

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

a) Vamos desligar as sessions para este o modulo (stateless!):

```
// modules/v1/Module.php
public function init()
{
    parent::init();
    \Yii::$app->user->enableSession = false;
}
```

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

b) Vamos configurar autenticador HTTP Basic Auth

// /modules/v1/controllers/DefaultController.php

use yii\filters\auth\HttpBasicAuth;

public function behaviors()

{

\$behaviors = parent::behaviors();

\$behaviors['authenticator'] = [

'class' => **HttpBasicAuth::className()**,

];

return \$behaviors;

}



Automaticamente vai chamar o **findIdentityByAccessToken()** do model User Identity, para autenticar.

Mas a função apenas recebe um argumento! Apenas o nome chega à função. Deveria chegar também a password! A função testa o Argumento recebido com o campo AccessToken do user. Não poderemos depender deste comportamento automático! (Se passarmos o accessToken:password) funciona mas só testa mesmo o primeiro Campo.

```
'auth' => [$this, 'auth']  
'auth' => [$this, 'auth']
```

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

Logo, vamos fornecer a **nossa** função de autenticação:

```
public function behaviors()  
{  
    $behaviors = parent::behaviors();  
    $behaviors['authenticator'] = [  
        'class' => HttpBasicAuth::className(),  
        'auth' => [$this, 'auth']  
    ];  
    return $behaviors;  
}
```


Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

Logo, vamos fornecer a nossa função de autenticação:

```
public function auth($username, $password)  
{  
    $user = \app\models\User::findByUsername($username);  
    if ($user && $user->validatePassword($password))  
    {  
        return $user;  
    }  
}
```

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

A partir daqui, o controlador espera *header* específico no pedido HTTP:

Authorization 'Basic '.base64(\$username.':'.\$password);

Serviços Web RESTful

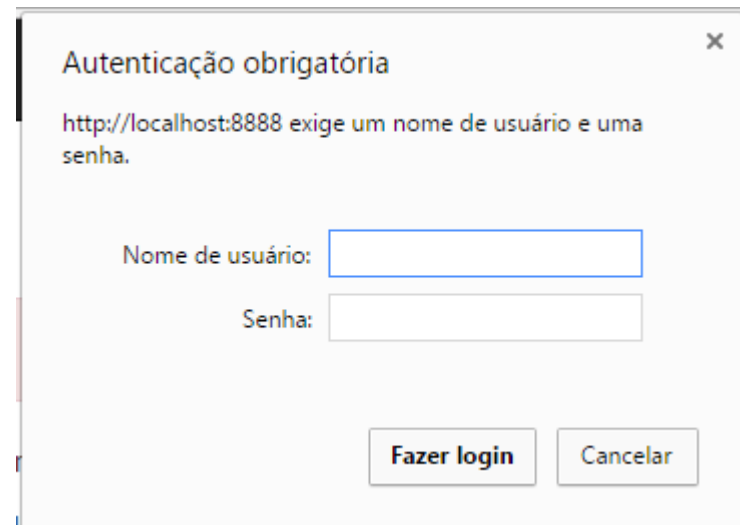
Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

Teste no browser:

`http://localhost:8888/v1/default`

Aparece



A screenshot of a browser authentication dialog box. The title bar says "Autenticação obrigatória" with a close button (X) in the top right corner. The main text inside the dialog says "http://localhost:8888 exige um nome de usuário e uma senha." Below this text are two input fields: "Nome de usuário:" followed by a text box, and "Senha:" followed by a password box. At the bottom right of the dialog are two buttons: "Fazer login" and "Cancelar".

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

Teste no Advanced Rest Client (chrome):

GET `http://localhost:8888/v1/default`

Headers

Accept `application/json`

Content-Type `application/json`

Authorization `basic YWRtaW46YWRtaW4=`

(Add header, Authorization, e lapis. Depois fornecer **login** e **password**. O encoding para base 64 é feito automaticamente resultando:

`basic YWRtaW46YWRtaW4=`

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

1. HTTP Basic Auth

Teste no Advanced Rest Client:

The screenshot shows the Advanced Rest Client interface. At the top, the Method is set to GET and the Request URL is http://localhost:8888/v1/default. Below this, the Parameters section is collapsed. The Headers section is expanded, showing a table with three headers: Accept (application/json), Content-Type (application/json), and Authorization (Basic YWRtaW46YWRtaW4=). Below the headers table, there is a red button labeled 'ADD HEADER'. At the bottom, a green checkmark icon indicates a successful request, followed by a green box containing '200 OK' and the response time '158.11 ms'.

Header name	Header value
Accept	application/json
Content-Type	application/json
Authorization	Basic YWRtaW46YWRtaW4=

ADD HEADER

200 OK 158.11 ms

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

2. Autenticação por Query Parameter

Assume-se que o cliente tem acesso a um token único. Esse token será enviado para o serviço via query parameter

`https://server.com/users?access-token=xxxxxxx`

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

2. Autenticação por Query Parameter

```
// /modules/v1/controllers/DefaultController.php
use yii\filters\auth\QueryParamAuth;
public function behaviors()
{
    $behaviors = parent::behaviors();
    $behaviors['authenticator'] = [
        'class' => QueryParamAuth::className(),
    ];
    return $behaviors;
}
```

Chama, naturalmente, a função **findIdentityByAccessToken** do model **User Identity** (implements **IdentityInterface**)

Serviços Web RESTful

Autenticação: 3 formas de enviar o token (disponíveis)

2. Autenticação por Query Parameter

Teste no browser:

GET <http://localhost:8888/v1/default>

```
<response>
<name>Unauthorized</name>
<message>Your request was made with invalid
credentials.</message>
<code>0</code>
<status>401</status>
<type>yii\web\UnauthorizedHttpException</type>
</response>
```


Serviços Web RESTful


Autenticação: 3 formas de enviar o token (disponíveis)

2. Autenticação por Query Parameter

Campo accessToken
da tabela Users

Teste no browser:

<http://localhost:8888/v1/default?access-token=100-token>



The screenshot shows a web browser window with the address bar containing the URL `localhost:8888/v1/default?access-token=100-token`. Below the address bar, a message states: "This XML file does not appear to have any style information associated with it". The main content area displays an XML response structure. The root element is `<response>`, which contains three `<item>` elements. Each `<item>` element has three child elements: `<id>`, `<nome>`, and `<morada>`. The first item has `<id>19</id>`, `<nome>ZZZ</nome>`, and `<morada>fdsfs</morada>`. The second item has `<id>20</id>`, `<nome>ZZZ</nome>`, and `<morada>fdsfs</morada>`. The third item has `<id>21</id>` and `<nome>ZZZ</nome>`, but the `<morada>` element is not visible in the screenshot.

```
<?xml version="1.0"?>
<response>
  <item>
    <id>19</id>
    <nome>ZZZ</nome>
    <morada>fdsfs</morada>
  </item>
  <item>
    <id>20</id>
    <nome>ZZZ</nome>
    <morada>fdsfs</morada>
  </item>
  <item>
    <id>21</id>
    <nome>ZZZ</nome>
```

Web Services RESTful (PHP/Yii2)

»» Autorização

Serviços Web RESTful

Autorização

Após autenticado, poderá ser necessário verificar se o utilizador tem **permissões** para chamar determinado método para determinado recurso

Re-escrever método **checkAccess()** do **controlador**.
Este método é chamado na altura de executar as
ações RESTful



Serviços Web RESTful

Autorização

Ex: imagine que apenas autores registados podem fazer post ou delete

```
public function checkAccess($action, $model = null, $params = [])  
{  
    if ($action === 'post' or $action === 'delete')  
        if (\Yii::$app->user->isGuest)  
            throw new \yii\web\ForbiddenHttpException('Apenas poderá  
                '.$action.' utilizadores registados...');  
}  
}
```