



Classes de Roteamento

Prof. Me. Hélio Esperidião

Roteamento

- **URI:** /Alunos
 - Get, post, put, delete
- O processo de quebrar a URI e um vetor, e fazer verificações por meio de ifs pode ser trabalhoso e tedioso.
- Existem algumas soluções de classes em php criadas por terceiros que tornam o processo de construção das rotas mais dinâmico e menos complexo.
 - <https://github.com/bramus/router>
 - Classe simples e leve que pode ser utilizada no processo de roteamento.

.htaccess – Ainda é preciso 😊

- RewriteEngine on
- RewriteCond %{REQUEST_FILENAME} !-f
- RewriteCond %{REQUEST_FILENAME} !-d
- RewriteRule ^(.*)\$ /index.php?path=\$1 [L]

```
<?php
```

```
//https://github.com/bramus/router
```

```
require_once "Router.php";
```

```
//Cria uma instância da classe Router
```

```
$router = new Router();
```

```
//mapeia a URI /
```

```
$router->get('/', function() {
```

```
    echo 'Ola mundo';
```

```
});
```

```
//mapeia a URI /rota1
```

```
$router->get('/rota1', function() {
```

```
    echo 'Eu sou a rota 1';
```

```
});
```

```
//mapeia a URI /rota1/rota2
```

```
$router->get('/rota1/rota2', function() {
```

```
    echo 'Eu sou a rota 2';
```

```
});
```

```
//inicializa o roteamento
```

```
$router->run();
```

```
?>
```

Index.php

Passagem de parâmetro via URI.

```
<?php
//https://github.com/bramus/router
require_once "Router.php";
```

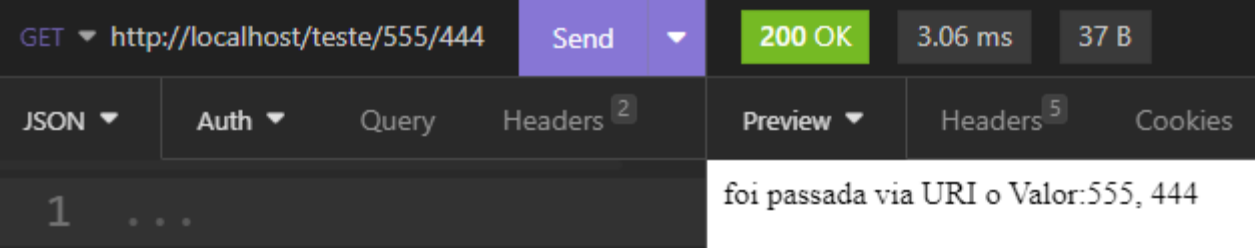
```
//Cria uma instância da classe Router
$router = new Router();
```

```
//mapeia a URI /teste/{int}
$router->get('/teste/(\d+)', function($v1) {
    echo "foi passada via URI o Valor:$v1 ";
});

//mapeia a URI /teste/{int}/{int}
$router->get('/teste/(\d+)/(\d+)', function($v1,$v2) {
    echo "foi passada via URI o Valor:$v1, $v2 ";
});
```

```
//inicializa o roteamento
$router->run();
?>
```

Padrão	Significado
(\d+)	Dígitos de 0-9



GET http://localhost/teste/555/444 Send 200 OK 3.06 ms 37 B

JSON Auth Query Headers 2 Preview Headers 5 Cookies

1 ... foi passada via URI o Valor:555, 444

```
<?php
//https://github.com/bramus/router
require_once "Router.php";
```

```
//Cria uma instância da classe Router
$router = new Router();
```

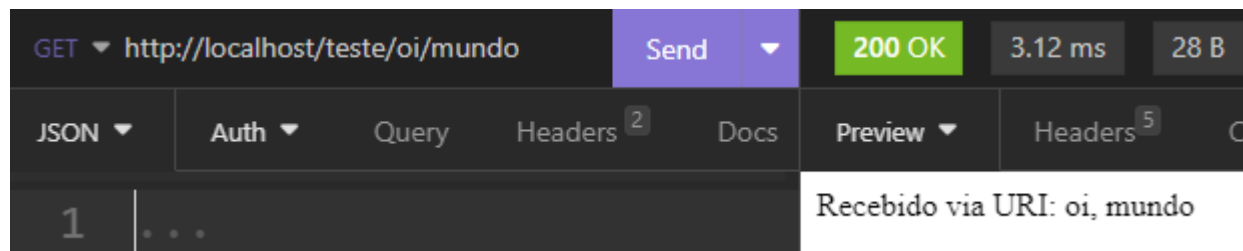
```
//mapeia a URI /teste/{string}
$router->get('/teste/(\w+)', function($v1) {
    echo "Recebido via URI: $v1 ";
});

//mapeia a URI /teste/{string}/{string}
$router->get('/teste/(\w+)/(\w+)', function($v1,$v2) {
    echo "Recebido via URI: $v1, $v2 ";
});
```

```
//inicializa o roteamento
$router->run();
?>
```

Passagem de paramento via URI.

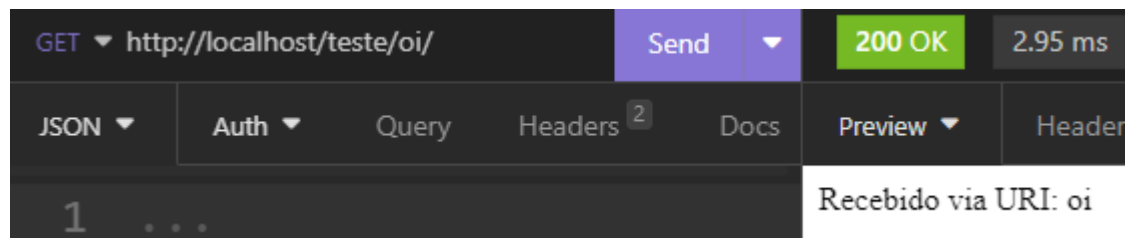
Padrão	Significado
(\w+)	Caracteres : a-z 0-9 _



GET http://localhost/teste/oi/mundo Send 200 OK 3.12 ms 28 B

JSON Auth Query Headers 2 Docs Preview Headers 5

1 ... Recebido via URI: oi, mundo



GET http://localhost/teste/oi/ Send 200 OK 2.95 ms

JSON Auth Query Headers 2 Docs Preview Header

1 ... Recebido via URI: oi

```
<?php
//https://github.com/bramus/router
require_once "Router.php";

//Cria uma instância da classe Router
$router = new Router();

//mapeia a URI /teste/{números e pontos}
$router->get('/teste/([0-9.]+)', function($v1) {
    echo "Recebido via URI: $v1 ";
});

//inicializa o roteamento
$router->run();
?>
```

Passagem de paramento via URI.

Padrão	Significado
(\w+)	Caracteres : a-z 0-9 _
([0-9.]+)	Float com “.”
([0-9,]+)	Float com “,”

GET http://localhost/teste/98.5 Send 200 OK 3.16 ms 23 B

JSON Auth Query Headers 2 Docs Preview Headers 5 Cookies Timeline

1 ... Recebido via URI: 98.5

```

<?php
//https://github.com/bramus/router
require_once "Router.php";

//Cria uma instância da classe Router
$router = new Router();

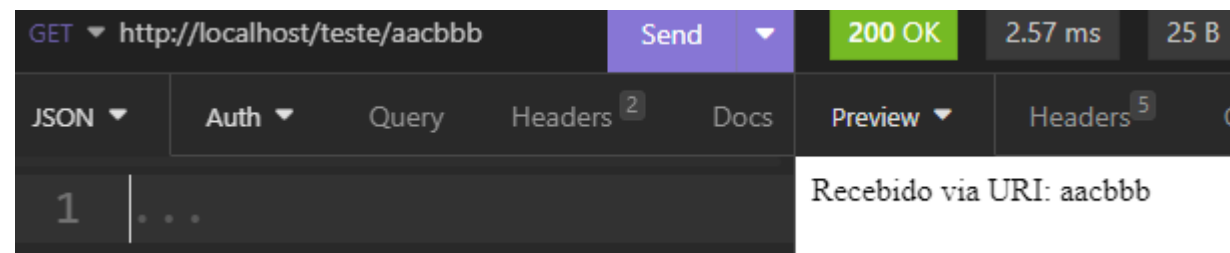
//mapeia a URI /teste/{números e virgula}
$router->get('/teste/([abc]+)', function($v1) {
    echo "Recebido via URI: $v1 ";
});

//inicializa o roteamento
$router->run();
?>

```

Passagem de parâmetro via URI.

Padrão	Significado
[abc]+	São aceitos apenas os caracteres ab c




```
<?php
//https://github.com/bramus/router
require_once "Router.php";
```

```
//Cria uma instância da classe Router
$router = new Router();
```

```
//mapeia a URI /
$router->post('/cliente/', function() {
    $jsonRecebido = file_get_contents('php://input');
    echo "post: $jsonRecebido";
});
```

```
$router->put('/cliente/(\d+)', function($id) {
    $jsonRecebido = file_get_contents('php://input');
    echo "put: $id - $jsonRecebido ";
});
```

```
$router->delete('/cliente/(\d+)', function($id) {
    echo "delete: $id";
});
```

```
//inicializa o roteamento
$router->run();
?>
```

POST, PUT, DELETE

A screenshot of a web client interface showing a successful POST request. The URL bar displays 'http://localhost/cliente/'. The status bar indicates a '200 OK' response with a time of 2.94 ms and a body size of 68 B. The 'JSON' tab is selected, showing the request body as a JSON object: `{ "nome": "helio", "email": "helioesperidiao@gmmmail.com" }`. The 'Preview' tab on the right shows the response body: `post: { "nome": "helio", "email": "helioesperidiao@gmmmail.com" }`.

A screenshot of a web client interface showing a successful PUT request. The URL bar displays 'http://localhost/cliente/25'. The status bar indicates a '200 OK' response with a time of 2.72 ms and a body size of 72 B. The 'JSON' tab is selected, showing the request body as a JSON object: `{ "nome": "helio", "email": "helioesperidiao@gmmmail.com" }`. The 'Preview' tab on the right shows the response body: `put: 25 - { "nome": "helio", "email": "helioesperidiao@gmmmail.com" }`.

A screenshot of a web client interface showing a successful DELETE request. The URL bar displays 'http://localhost/cliente/25'. The status bar indicates a '200 OK' response with a time of 2.55 ms and a body size of 10 B. The 'JSON' tab is selected, showing the request body as an empty object: `{ }`. The 'Preview' tab on the right shows the response body: `delete: 25`.

Exemplo Completo padrão MVC/POO

- Arquivos:
 - .htaccess
 - Reescrita de URLs/URIs
 - index.php
 - Cria as Regras de roteamento.
 - Retangulo.php
 - Classe para lidar com operações de um retângulo.
 - Router.php
 - Classe para facilitar o processo de roteamento.

.htaccess – Ainda é preciso 😊

- RewriteEngine on
- RewriteCond %{REQUEST_FILENAME} !-f
- RewriteCond %{REQUEST_FILENAME} !-d
- RewriteRule ^(.*)\$ /index.php?path=\$1 [L]

Retangulo.php.

Considerar que GETs e Sets foram programados

```
class Retangulo{  
    private $base;  
    private $altura;  
  
    public function calcularArea(){  
        return ($this->altura * $this->base);  
    }  
    public function calcularDiagonal(){  
        return sqrt(pow($this->altura, 2) + pow($this->base, 2));  
    }  
    public function calcularPerimetro(){  
        return ($this->altura * 2 + $this->base * 2);  
    }  
}
```

index.php -1/4

```
<?php
```

```
//https://github.com/bramus/router
```

```
//Importa os arquivos:
```

```
require_once "Router.php";
```

```
require_once "Retangulo.php";
```

```
//Cria uma instância da classe Router
```

```
$router = new Router();
```

index.php -2/4

```
$router->get('/retangulo/(\d+)/(\d+)/area', function($v1,$v2) {  
    $r1 = new Retangulo();  
    $r1->setBase($v1);  
    $r1->setAltura($v2);  
    $area= $r1->calcularArea();  
    $resposta['area']= $area;  
    echo json_encode($resposta);  
});
```

index.php -3/4

```
$router->get('/retangulo/(\d+)/(\d+)/perimetro', function($v1, $v2) {  
    $r1 = new Retangulo();  
    $r1->setBase($v1);  
    $r1->setAltura($v2);  
    $perimetro= $r1->calcularPerimetro();  
    $resposta['perimetro']= $perimetro;  
    echo json_encode($resposta);  
});
```

index.php -4/4

```
$router->get('/retangulo/(\d+)/(\d+)/diagonal', function($v1,$v2) {  
    $r1 = new Retangulo();  
    $r1->setBase($v1);  
    $r1->setAltura($v2);  
    $diagonal= $r1->calcularDiagonal();  
    $resposta['diagonal']= $diagonal;  
    echo json_encode($resposta);  
});
```

```
$router->run();  
?>
```


Router.php

- [//https://github.com/bramus/router](https://github.com/bramus/router)