

# Building an API with Slim 3

Rob Allen ~ @akrabat  
December 2015

Let's start with Slim 3

# Hello world

```
<?php
require 'vendor/autoload.php';
$app = new \Slim\App();

$app->get('/ping', function ($request, $response) {
    $body = json_encode(['ack' => time()]);
    $response->write($body);
    $response = $response->withHeader(
        'Content-Type', 'application/json');
    return $response;
});

$app->run();
```

# Hello world

Method    Pattern    Action



```
$app->get('/ping', function ($request, $response) {  
    $body = json_encode(['ack' => time()]);  
    $response->write($body);  
    $response = $response->withHeader(  
        'Content-Type', 'application/json');  
    return $response;  
});
```

# Hello world

```
$ http --json http://localhost:8888/ping
```

```
HTTP/1.1 200 OK
```

```
Connection: close
```

```
Content-Length: 18
```

```
Content-Type: application/json
```

```
Host: localhost:8888
```

```
X-Powered-By: PHP/5.6.14
```

```
{
```

```
  "ack": 1445111794
```

```
}
```

PSR-7?

# It's all about HTTP

## Request:

```
{METHOD} {URI} HTTP/1.1  
Header: value1,value2  
Another-Header: value
```

Message body

## Response:

```
HTTP/1.1 {STATUS_CODE} {REASON_PHRASE}  
Header: value
```

Message body

# Current PHP

## Request:

- `$_SERVER`, `$_GET`, `$_POST`, `$_COOKIE`, `$_FILES`
- `apache_request_headers()`
- `php://input`

## Response:

- `header()`
- `echo` (& `ob_*` family)



# PSR 7: HTTP messaging

It's just some interfaces

- `RequestInterface` (& `ServerRequestInterface`)
- `ResponseInterface`
- `UriInterface`
- `UploadedFileInterface`

# Key feature 1: Immutability

Request, Response, Uri & UploadFile are *immutable*

```
$uri = new Uri('https://api.joind.in/v2.1/events');  
$uri2 = $uri->withQuery('?filter=upcoming');
```

```
$request = (new Request())  
    ->withMethod('GET')  
    ->withUri($uri2)  
    ->withHeader('Accept', 'application/json')  
    ->withHeader('Authorization', 'Bearer 0873418d');
```

# Key feature 2: Streams

Message bodies are *streams*

```
$body = new Stream();  
$body->write('<p>Hello');  
$body->write('World</p>');  
  
$response = (new Response())  
    ->withStatus(200, 'OK')  
    ->withHeader('Content-Type', 'application/header')  
    ->withBody($body);
```

Let's talk APIs

What makes a good API?

# A good API

- HTTP method negotiation
- Content-type handling
- Honour the *Accept* header
- Error handling
- Versions

# HTTP method negotiation

# HTTP method negotiation

```
$ http --json PUT http://localhost:8888/ping
HTTP/1.1 405 Method Not Allowed
Allow: GET
Connection: close
Content-Length: 53
Content-type: application/json
Host: localhost:8888
X-Powered-By: PHP/5.6.14
```

```
{
  "message": "Method not allowed. Must be one of: GET"
}
```



# HTTP method routing

```
$app->get('/author', function($req, $res) {});  
$app->post('/author', function($req, $res) {});
```

```
$app->get('/author/{id}', function($req, $res) {});  
$app->put('/author/{id}', function($req, $res) {});  
$app->patch('/author/{id}', function($req, $res) {});  
$app->delete('/author/{id}', function($req, $res) {});
```

```
$app->any('/author', function($req, $res) {});  
$app->map(['GET', 'POST'], '/author', /* ... */);
```

# Dynamic routes

```
$app->get('/author/{id}',  
  function($request, $response, $args) {  
    $id = $args['id'];  
    $author = $this->authors->loadById($id);  
  
    $body = json_encode(['author' => $author]);  
    $response->getBody()->write($body);  
  
    $response = $response->withHeader(  
      'Content-Type', 'application/json');  
  
    return $response;  
  });
```

# It's just Regex

```
// numbers only
```

```
$app->get('/author/{id:\d+}', $callable);
```

```
// optional segments
```

```
$app->get('/author[/]{id:\d+}]/', $callable);
```

```
$app->get('/news[/]{y:\d{4}}[/]{m:\d{2}}]/', $callable);
```

# Content-type handling

# Content-type handling

The *Content-type* header specifies the format of the incoming data

```
$ curl -X "POST" "http://localhost:8888/author" \  
-H "Content-Type: application/json" \  
-d '{ "name": "Terry Pratchett" }'
```

# Read with `getBody()`

```
$app->post('/author',  
    function ($request, $response, $args) {  
        $data = (string)$request->getBody();  
  
        return $response->write(print_r($data, true));  
    }  
);
```

Output:

```
{ "name": "Terry Pratchett" }
```

# Read with getParsedBody()

```
$app->post('/author',  
    function ($request, $response, $args) {  
        $data = (array)$request->getParsedBody();  
  
        return $response->write(print_r($data, true));  
    }  
);
```

Output:

```
Array  
(  
    [name] => Terry Pratchett  
)
```

# This also works with XML

```
curl -X "POST" "http://localhost:8888/author" \  
  -H "Content-Type: application/xml" \  
  -d "<author><name>Terry Pratchett</name></author>"
```

## Output:

```
Array  
(  
  [name] => Terry Pratchett  
)
```



# And form data

```
curl -X "POST" "http://localhost:8888/author" \  
  -H "Content-Type: application/x-www-form-urlencoded" \  
  --data-urlencode "name=Terry Pratchett"
```

## Output:

```
Array  
(  
  [name] => Terry Pratchett  
)
```

# Other formats? e.g. CSV

name, dob

Terry Pratchett,1948-04-28

Andy Weir,1972-06-17

as curl:

```
curl -X "POST" "http://localhost:8888/author" \  
  -H "Content-Type: text/csv" \  
  -d '$name, dob  
Terry Pratchett,1948-04-28  
Andy Weir,1972-06-17'
```

# Register media type

```
$request->registerMediaTypeParser(  
    'text/csv',  
    function ($input) {  
        $data = str_getcsv($input, "\n");  
        $keys = str_getcsv(array_shift($data));  
  
        foreach ($data as &$row) {  
            $row = str_getcsv($row);  
            $row = array_combine($keys, $row);  
        }  
  
        return $data;  
    }  
);
```

# Result

```
Array
(
  [0] => Array
    (
      [name] => Terry Pratchett
      [dob] => 1948-04-28
    )

  [1] => Array
    (
      [name] => Andy Weir
      [dob] => 1972-06-17
    )
)
```

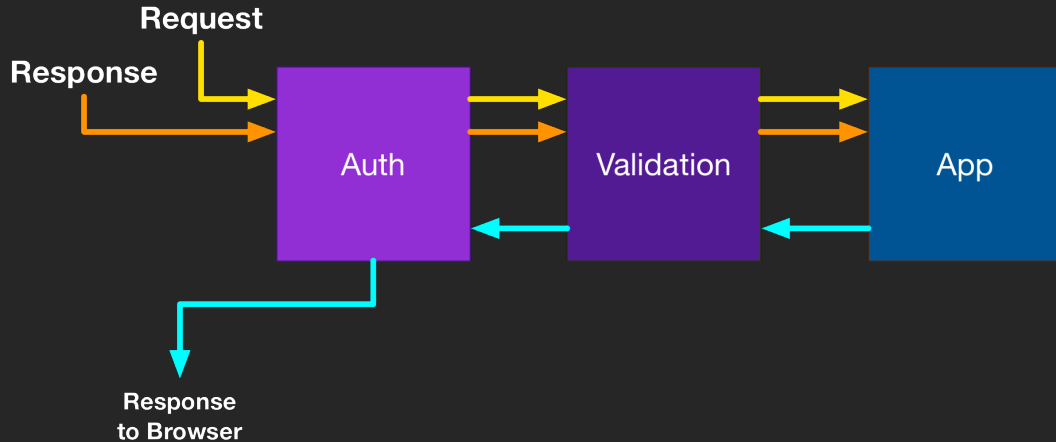
# Middleware

*Middleware is code that exists between the request and response, and which can take the incoming request, perform actions based on it, and either complete the response or pass delegation on to the next middleware in the queue.*

Matthew Weier O'Phinney

# Middleware

Manipulation of



# Middleware

```
function ($request, $response, callable $next = null)
{
    // do something before

    // call through to next middleware
    if ($next) {
        $response = $next($request, $response);
    }

    // do something with $response after

    return $response;
}
```

# Media type middleware

```
$app->add(function ($request, $response, $next) {  
  
    $request->registerMediaTypeParser('text/csv',  
        function ($input) {  
            $data = explode("\n", $input);  
            $keys = str_getcsv(array_shift($data));  
            foreach ($data as &$row) {  
                $row = str_getcsv($row);  
                $row = array_combine($keys, $row);  
            }  
            return $data;  
        }  
    );  
    return $next($request, $response);  
});
```



Honour the *Accept* header

# Honour the Accept header

Return data in the format the client expects

```
curl -X "POST" "http://localhost:8888/author" \  
-H "Accept: application/json" \  
-H "Content-Type: application/json" \  
-d '{ "name": "Terry Pratchett" }'
```

# Returning JSON

```
$app->post(  
    '/author',  
    function ($request, $response, $args) {  
        $author = new Author($request->getParsedBody());  
        $this->authors->save($author);  
  
        $response = $response->withJson($author->toArray());  
        $response = $response->withStatus(201);  
        return $response;  
    }  
);
```

# Returning JSON

```
HTTP/1.1 201 Created
Content-type: application/json
Content-Length: 106
```

```
{
  "id": "2ff815ad-491d-4db8-a025-363516e7c27e",
  "name": "Terry Pratchett",
  "biography": null
}
```

# Returning XML

```
curl -X "POST" "http://localhost:8888/author" \  
  -H "Accept: application/xml" \  
  -H "Content-Type: application/json" \  
  -d '{ "name": "Terry Pratchett" }'
```

# Determine media type

```
$ composer require willdurand/negotiation
// find preferred format from Accept header
function determineMediaType($acceptHeader)
{
    $negotiator = new \Negotiation\Negotiator();
    $known = ['application/json', 'application/xml'];

    $mediaType = $negotiator->getBest($acceptHeader, $known);
    if ($mediaType) {
        return $mediaType->getValue();
    }
    return 'application/json';
}
```

# Format output

```
$acceptHeader = $request->getHeaderLine('Accept')
$mediaType = determineMediaType($acceptHeader);

switch ($mediaType) {
    case 'application/xml':
        $response->getBody()->write(arrayToXml($data));
        break;

    case 'application/json':
        $response->getBody()->write(json_encode($data));
        break;
}

return $response->withHeader("Content-Type", $mediaType);
```

# XML output

HTTP/1.1 201 Created  
Content-type: application/xml  
Content-Length: 131

```
<?xml version="1.0"?>  
<root>  
  <id>98c22fa3-bf97-48c8-accd-025470c34b46</id>  
  <name>Terry Pratchett</name>  
  <biography/>  
</root>
```



# There's a component for this!

```
$ composer require akrabat/rka-content-type-renderer
```

To use:

```
$renderer = new RKA\ContentTypeRenderer\Renderer();  
$response = $renderer->render($request, $response, $data);  
return $response->withStatus(201);
```

# Error handling

# Error handling

- Method not allowed
- Not found
- Generic error

# Method not allowed

```
curl -X "PUT" "http://localhost:8888/ping"
```

```
HTTP/1.1 405 Method Not Allowed  
Content-type: text/html;charset=UTF-8  
Allow: GET
```

```
<html>  
  <body>  
    <h1>Method not allowed</h1>  
    <p>Method not allowed. Must be one of:  
      <strong>GET</strong></p>  
  </body>  
</html>
```

# Not found

```
curl -X "GET" "http://localhost:8888/foo" \  
-H "Accept: application/xml"
```

```
HTTP/1.1 404 Not Found  
Content-Type: application/xml  
Allow: GET
```

```
<root><message>Not found</message></root>
```

# Raise your own

```
$app->get(
    '/author/{id}',
    function ($request, $response, $args) {
        $author = $this->authors->loadById($args['id']);
        if (!$author) {

            return $this->notFoundHandler($request, $response);

        }

        // continue with $author
    }
);
```

# Generic error

```
$app->get('/error',  
  function ($request, $response, $args) {  
    throw new \Exception("Something has gone wrong!");  
  }  
);
```

```
curl -X "GET" "http://localhost:8888/error" \  
-H "Accept: application/json"
```

```
HTTP/1.1 500 Internal Server Error  
Content-type: application/json  
Content-Length: 43
```

```
{  
  "message": "Slim Application Error"  
}
```

# Exception information

```
$settings = [  
  'settings' => [  
    'displayErrorDetails' => true,  
  ],  
];  
  
$app = new Slim\App($settings);
```



# Exception information

HTTP/1.1 500 Internal Server Error  
Content-type: application/json

```
{
  "message": "Slim Application Error",
  "exception": [
    {
      "type": "Exception",
      "code": 0,
      "message": "Something has gone wrong!",
      "file": "/dev/an-api/app/routes.php",
      "line": 8,
      "trace": [
        "#0 [internal function]: Closure->{closure} ...
        "#2 /dev/an-api/vendor/slim/slim/Slim/Route.php(...
```

# Handle warnings

```
$app->get('/error',  
    function ($request, $response, $args) {  
        ini_get(); // will generate a warning  
    }  
);
```

```
// convert errors into exceptions  
set_error_handler(function ($level, $msg, $file, $ln) {  
    if (!(error_reporting() & $level)) { return; }  
    throw new \Exception($msg, $level);  
});
```

# Versions

# Versions

Two choices:

- Segment within URL: `http://api.example.com/v1/author`
- Media type: Accept: `application/vnd.rka.author.v1+json`

# URL segment

Use route groups:

```
$app->group('/v1', function () {  
  
    // http://api.example.com/v1/author  
    $this->get('/author',  
        function ($request, $response) { /*...*/ }  
    );  
  
    // http://api.example.com/v1/author/123  
    $this->get('/author/{id}',  
        function ($request, $response, $args) { /*...*/ }  
    );  
  
});
```

# Media type versioning

Firstly, let's look at Slim's dependency injection support

# The container

## Register services with the DIC

```
$settings = ['settings' => [  
    'dsn' => 'sqlite:data/bookshelf.db',  
]];  
$app = new Slim\App($settings);  
$container = app->getContainer();  
  
$container['pdo'] = function ($c) {  
    return new PDO($c['settings']['dsn']);  
};  
  
$container['authors'] = function ($c) {  
    return new Bibliotheque\AuthorMapper($c['pdo']);  
};
```

# Controller classes

Register your controller with the container

```
$container['AuthorController'] = function ($c) {  
    $renderer = $c->get('renderer');  
    $authors = $c->get('authors');  
    return new App\AuthorController($renderer, $authors);  
}
```

// Register with router:

```
$app->get('/author', 'AuthorController:listAll');
```



# Author controller

```
class AuthorController
{
    public function __construct($renderer, $authors)
    {
        $this->renderer = $renderer;
        $this->authors = $authors;
    }

    public function listAll($req, $res)
    {
        $authors = $this->authors->fetchAll();
        $data = ['authors' => $authors];
        return $this->renderer->render($req, $res, $data);
    }
}
```

# Media type versioning

Select controller based on Accept header

```
$container['AuthorController'] = function ($c) {  
  
    $request = $c->get('request');  
    $acceptHeader = $request->getHeaderLine('Accept');  
  
    if (strpos($acceptHeader,  
        'application/vnd.rka.author.v2') !== false) {  
        return new App\V2\AuthorController(/*...*/);  
    }  
  
    return new App\V1\AuthorController(/*...*/);  
};
```

To sum up

# Summary

A good API deals with:

- HTTP method negotiation
- Content-type handling
- Honour the *Accept* header
- Error handling
- Versions

# Resources

- <http://phptherightway.com>
- <http://slimframework.com/docs>
- <https://github.com/slimphp/Slim>
- <http://akrabat.com/category/slim-framework/>
- <http://ryanszrama.com/topics/slim>

# Questions?

Rob Allen - <http://akrabat.com> - @akrabat

Thank you!

Rob Allen - <http://akrabat.com> - @akrabat