

Building Modern APIs with Slim Framework

Code: <https://akrab.at/phpuk20>

Rob Allen, February 2020

Today's plan

- Learn a bit about Slim for context
- Learn the features of a good API
- Put them into practice within a Slim application
- Coffee & cake!

What do you know about APIs?



Rock-Paper-Scissors

An API to play Rock-Paper-Scissors

1. Create a new game
2. Make first player's move
3. Make second player's move
4. Get result

Create a game

```
$ curl -H "Content-Type: application/json" \  
http://localhost:8888/games \  
-d '{"player1": "Rob", "player2": "Jon"}'
```



Create a game

```
$ curl -H "Content-Type: application/json" \  
http://localhost:8888/games \  
-d '{"player1": "Rob", "player2": "Jon"}'  
  
{  
  "_links": {  
    "makeNextMove": {  
      "href": "/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves",  
      "description": "Make a player's move"  
    }  
  }  
}
```



Make a move

```
$ curl -H "Content-Type: application/json" \  
http://localhost:8888/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves  
-d '{"player": "Rob", "move": "rock"}'
```



Make a move

```
$ curl -H "Content-Type: application/json" \
http://localhost:8888/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves
-d '{"player": "Rob", "move": "rock"}'

{
  "_links": {
    "makeNextMove": {
      "href": "/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves",
      "description": "Make player 2's move"
    }
  }
}
```

Make other move

```
$ curl -H "Content-Type: application/json" \  
http://localhost:8888/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves  
-d '{"player": "Jon", "move": "rock"}'
```

Make other move

```
$ curl -H "Content-Type: application/json" \
http://localhost:8888/games/2ab83e2a-98d0-4110-b3ae-90d6fbf5/moves
-d '{"player": "Jon", "move": "rock"}'
```

```
{
  "result": "Draw. Both players chose rock",
  "_links": {
    "newGame": {
      "href": "/games/",
      "description": "Start a new game"
    }
  }
}
```





Slim The C in MVC

Slim Framework

- Created by Josh Lockhart (phptherightway.com)
- PSR-4 autoloading
- PSR-7 Request and Response objects
- PSR-15 Middleware and Request Handlers
- PSR-11 DI container support



Bring your own components

Slim provides the router and dispatcher:

You provide:

- PSR-7 component
- PSR-11 DI container
- View layer (templates)
- Model layer (Database/ORM)

Getting started

```
$ composer create-project akrabat/slim4-starter hello-api
```

Slim4-Starter provides:

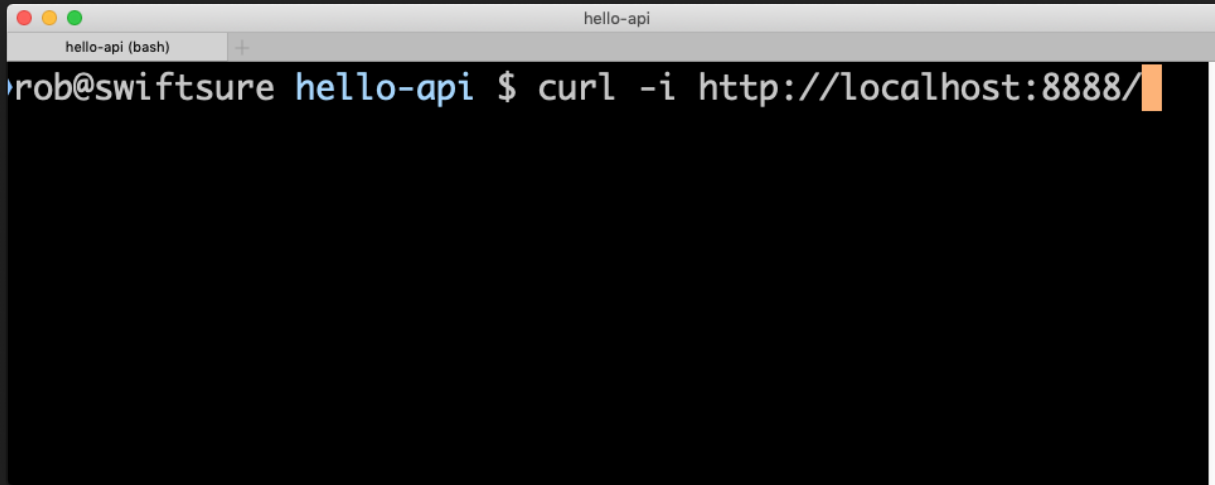
- Slim 4
- Slim-Psr7
- PHP-DI
- Monolog

Getting started

Start a web server

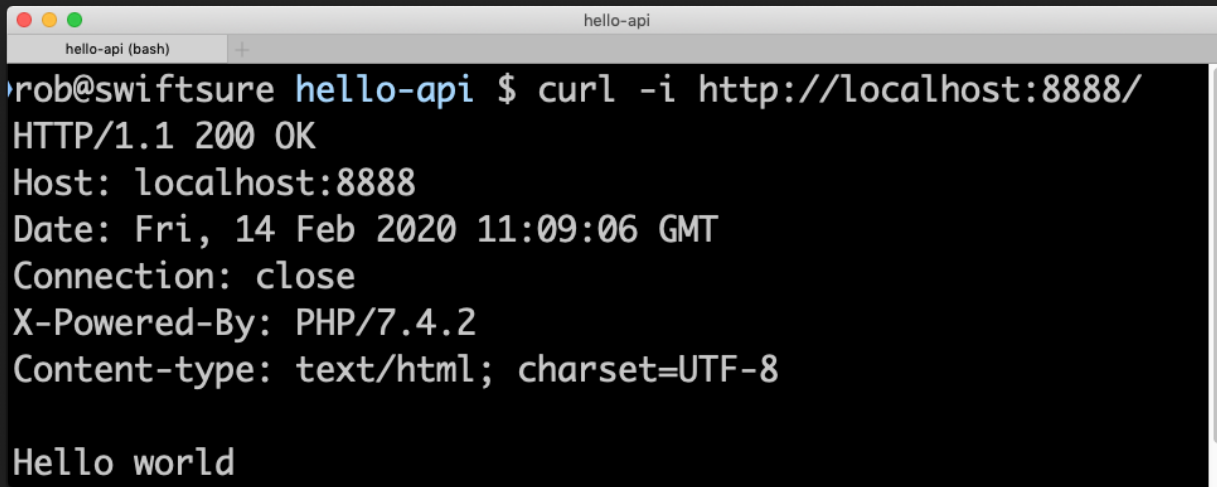
```
$ php -S 0.0.0.0:8888 -t public/  
PHP 7.4.2 Development Server (http://0.0.0.0:8888) started
```

Getting started

A terminal window titled 'hello-api' with a tab labeled 'hello-api (bash)'. The prompt is 'rob@swiftsure hello-api \$' and the command 'curl -i http://localhost:8888/' is entered, with an orange cursor at the end of the line.

```
rob@swiftsure hello-api $ curl -i http://localhost:8888/
```

Getting started

A terminal window titled 'hello-api' with a tab labeled 'hello-api (bash)'. The prompt is 'rob@swiftsure hello-api \$'. The command 'curl -i http://localhost:8888/' has been executed, resulting in an HTTP 200 OK response. The response headers are: 'Host: localhost:8888', 'Date: Fri, 14 Feb 2020 11:09:06 GMT', 'Connection: close', 'X-Powered-By: PHP/7.4.2', and 'Content-type: text/html; charset=UTF-8'. The body of the response is 'Hello world'.

```
rob@swiftsure hello-api $ curl -i http://localhost:8888/  
HTTP/1.1 200 OK  
Host: localhost:8888  
Date: Fri, 14 Feb 2020 11:09:06 GMT  
Connection: close  
X-Powered-By: PHP/7.4.2  
Content-type: text/html; charset=UTF-8  
  
Hello world
```

Directory structure

```
├── config/
│   ├── dependencies.php
│   ├── middleware.php
│   ├── routes.php
│   └── settings.php
├── public/
│   └── index.php
├── src/
│   └── Handler/
│       └── HomePageHandler.php
```

```
├── var/
│   ├── cache/
│   └── log/
├── vendor/
├── composer.json
└── composer.lock
```

How does Slim work?

1. Bootstrap application
2. Execute routing
3. Dispatch route handler
4. Return response



Routing

1. Inspect URL and matches segments

2. On match:

- Add `Route` object to Request attribute (`__route__`)
- Dispatch associated *route handler*

Route handlers

- Manage business logic operations
- Receive a PSR-7 `ServerRequest`
- Return a PSR-7 `Response`
- Implemented as PSR-15 `RequestHandler`

HomePageHandler

```
class HomePageHandler implements RequestHandlerInterface
{
    public function process(
        ServerRequestInterface $request,
    ) : ResponseInterface {
        $response = new Response();
        $response->getBody()->write("Hello $name");
    }
}
```

Coding time

Create a Slim app



HTTP is the foundation

It's all about HTTP

HTTP is a stateless request/response protocol that operates by exchanging messages.

RFC 7230

Client request

POST /v2.1/events/18/comments HTTP/1.1

Host: api.dev.joinind.in

User-Agent: curl/7.54.0

Accept: application/xml; q=0.8, application/json

Content-Type: application/json

Content-Length: 59

{"comment":"Great Talk. Nothing wrong with it!","rating":5}



Server response

```
HTTP/1.1 401 Unauthorized
Server: Apache/2.2.22 (Debian)
Status: 400
Content-Length: 32
Connection: close
Content-Type: application/json; charset=utf8

["Invalid Authorization Header"]
```



PSR-7

OO interfaces to model HTTP

- `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');  
$uri3 = $uri->withQuery('?filter=cfp');
```

`with()` methods return a new object with the new information

Key feature 1: Immutability

*// This *does not work**

```
$response = new Response();  
$response->withStatus(200);  
$response->withHeader('Content-type', 'text/xml');
```

Key feature 1: Immutability

```
// This *does not work*
```

```
$response = new Response();
```

```
$response->withStatus(200);
```

```
$response->withHeader('Content-type', 'text/xml');
```

```
// This works: Reassign returned object
```

```
$response = new Response();
```

```
$response = $response->withStatus(200);
```

```
$response = $response->withHeader('Content-type', 'text/xml');
```

Key feature 2: Streams

Message bodies are *streams*

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

API

Device $\leftrightarrow$ Server

/api/route

PUT title, waypoint[] $\rightarrow$ routeID

/api/route/id

Let's talk APIs

GET $\rightarrow$ Route, Waypoint[], Skin

/api/route/updates

POST $\rightarrow$ [" , " , "]

/api/route/browse*

GET $\rightarrow$ [Route, Skin]

*Optional geo data

Features of a good API

- Correctness
- Malleability
- Error handling
- Documentation

tl;dr

Ensure your API is maintainable and
developer-friendly

Get it right!

Embrace HTTP methods

Method	Used for	Idempotent?
GET	Retrieve data	Yes
PUT	Change data	Yes
DELETE	Delete data	Yes
POST	Change data	No
PATCH	Update data	No

(HTTP methods are also known as *verbs*)

HTTP method negotiation

If a given resource (URI) doesn't support the requested HTTP verb, then
return: 405 Method Not Allowed

HTTP method negotiation

If a given resource (URI) doesn't support the requested HTTP verb, then return: 405 Method Not Allowed

```
$ curl -i -X PUT -H "Accept: application/json" http://localhost:8888
```

```
HTTP/1.1 405 Method Not Allowed
```

```
Allow: GET
```

```
Content-type: application/json
```

```
{  
  "message": "405 Method Not Allowed",  
}
```

URLs matter

- Be consistent
- Prefer nouns for resource names
- Plural names work best for collections
- Unique URL for each resource (/users/123)
- Resources can be named after business processes too (e.g. customer-enrolment, money-transfer, merges)



Status codes

Send the right one for the right situation!

1xx	Informational	<i>(We're still working)</i>
2xx	Success	<i>(All done)</i>
3xx	Redirection	<i>(Go elsewhere)</i>
4xx	Client error	<i>(Your fault)</i>
5xx	Server error	<i>(Our fault)</i>



Rock-Paper-Scissors

Verb	URL	Purpose
GET	/games	List all games
POST	/games	Start a new game
GET	/games/123	Find out about game 123
POST	/games/123/moves	Play a move in game 123



Routing Uris in Slim

config/routes.php:

```
$app->get(  
    '/games',  
    App\Handler\ListGames::class  
)->setName('games');
```

Routes have a method

config/routes.php:

```
$app->get(                                     // Method
    '/games',
    App\Handler\ListGames::class
)->setName('games');
```


\$app method => HTTP verb

```
$app->get()  
$app->post()  
$app->put()  
$app->patch()  
$app->delete()
```

Multiple methods:

```
$app->any()  
$app->map(['GET', 'POST'], ...);
```

Routes have a pattern

config/routes.php:

```
$app->get(  
    '/games',  
    App\Handler\ListGames::class  
)->setName('games');
```

// Patten

Route pattern

// literal

```
$app->get('/games', $handler);
```

// placeholder within { & } (access as request attribute)

```
$app->get('/games/{id}', $handler);
```

// constrain using regex

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

// optional segments use [&]

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

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

Routes have a handler

config/routes.php:

```
$app->get(  
    '/games',  
    App\Handler\ListGames::class           // Handler  
)->setName('games');
```

Route handlers

- Manage business logic operations
- Receive a PSR-7 `ServerRequest`
- Return a PSR-7 `Response`
- Implemented as PSR-15 `RequestHandler`

Responses in Slim

Create a new Response object and use the `with` methods:

```
$response = new Response();
```

```
// status code
```

```
$response = $response->withStatus(404);
```

```
// headers
```

```
$response = $response->withHeader('Content-Type', 'application/json');
```

```
// body
```

```
$response->getBody()->write(json_encode(['foo' => 'bar']));
```

Coding time

Handlers



Incoming data

Content-Type handling

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

```
$ curl http://localhost:8888/games \  
  -H "Content-Type: application/json" \  
  -d '{"player1": "Rob", "player2": "Jon"}'
```

Read with getBody()

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

Output is a string:

```
'{"player1": "Rob", "player2": "Jon"}'
```

Read with getParsedBody()

Add Slim's body-parsing middleware to your app:

```
$app->addBodyParsingMiddleware();
```

Use in your handler:

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

Read with getParsedBody()

```
$ curl -H "Content-Type: application/json" \  
  -H "Content-Type: application/json" \  
  -d '{"player1": "Rob", "player2": "Jon"}'
```

Output is an array:

```
Array  
(  
  [player1] => Rob,  
  [player2] => Jon  
)
```

This also works with XML

```
$ curl "http://localhost:8888/games" \  
  -H "Content-Type: application/xml" \  
  -d "<game><player1>Rob</player1><player2>Jon</player2></game>"
```

Output:

Array

```
(  
  [player1] => Rob,  
  [player2] => Jon  
)
```

And form data

```
curl "http://localhost:8888/games" \  
  -H "Content-Type: application/x-www-form-urlencoded" \  
  -d "player1=Rob" -d "player2=Jon"
```

Output:

Array

```
(  
  [player1] => Rob,  
  [player2] => Jon  
)
```

addBodyParsingMiddleware() ?

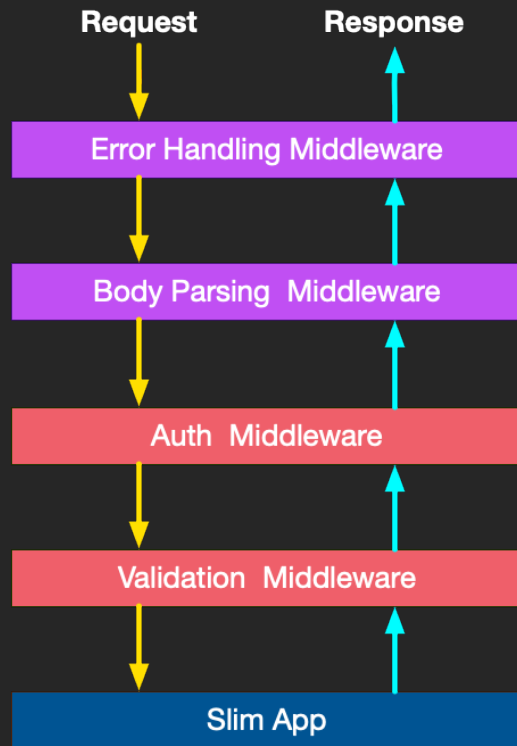
Middleware

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

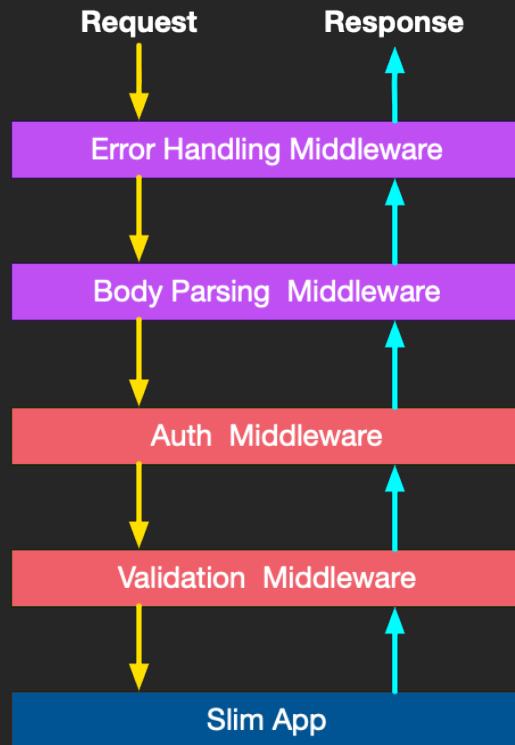
Take a request, return a response



Middleware

LIFO stack:

```
$app->add(ValidationMiddleware::class);  
$app->add(AuthMiddleware::class);  
$app->add(AuthMiddleware::class);  
$app->addBodyParsingMiddleware();  
$app->addErrorMiddleware(true, true, true);
```



PSR-15 MiddlewareInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

PSR-15 MiddlewareInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

PSR-15 MiddlewareInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

PSR-15 MiddlewareInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

TimerMiddleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;

        return $response->withHeader('Time-Taken', $taken);
    }
}
```

TimerMiddleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;

        return $response->withHeader('Time-Taken', $taken);
    }
}
```


TimerMiddleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;

        return $response->withHeader('Time-Taken', $taken);
    }
}
```

TimerMiddleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;

        return $response->withHeader('Time-Taken', $taken);
    }
}
```

TimerMiddleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;

        return $response->withHeader('Time-Taken', $taken);
    }
}
```

Route Handlers

The other half of PSR-15!



Route Handlers

Any callable!

```
$app->add('/', function ($request, $response) { ... });  
$app->add('/', ['RootController', 'aStaticFunction']);  
$app->add('/', [new RootController(), 'aFunction']);  
$app->add('/', RootController::class.':pingAction');  
$app->add('/', RootHandler::class);
```

Use the PSR-15 one

```
$app->add('/', RootHandler::class);
```

PSR-15 RouteHandlerInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface RequestHandlerInterface
{
    public function handle(
        ServerRequestInterface $request
    ): ResponseInterface;
}
```

PSR-15 RouteHandlerInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface RequestHandlerInterface
{
    public function handle(
        ServerRequestInterface $request
    ): ResponseInterface;
}
```


PSR-15 RouteHandlerInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface RequestHandlerInterface
{
    public function handle(
        ServerRequestInterface $request
    ): ResponseInterface;
}
```

PSR-15 RouteHandlerInterface

```
namespace Psr\Http\Server;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface RequestHandlerInterface
{
    public function handle(
        ServerRequestInterface $request
    ): ResponseInterface;
}
```

HomePageHandler

```
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Server\RequestHandlerInterface;
use Slim\Psr7\Response;

class HomePageHandler implements RequestHandlerInterface
{
    public function handle(Request $request): ResponseInterface
    {
        $response = new Response();
        $response->getBody()->write('Hello World');
        return $response;
    }
}
```

HomePageHandler

```
use Psr\Http\Message\ResponseInterface;  
use Psr\Http\Message\ServerRequestInterface as Request;  
use Psr\Http\Server\RequestHandlerInterface;  
use Slim\Psr7\Response;
```

```
class HomePageHandler implements RequestHandlerInterface  
{  
    public function handle(Request $request): ResponseInterface  
    {  
        $response = new Response();  
        $response->getBody()->write('Hello World');  
        return $response;  
    }  
}
```

HomePageHandler

```
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Server\RequestHandlerInterface;
use Slim\Psr7\Response;

class HomePageHandler implements RequestHandlerInterface
{
    public function handle(Request $request): ResponseInterface
    {
        $response = new Response();
        $response->getBody()->write('Hello World');
        return $response;
    }
}
```

HomePageHandler

```
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Server\RequestHandlerInterface;
use Slim\Psr7\Response;

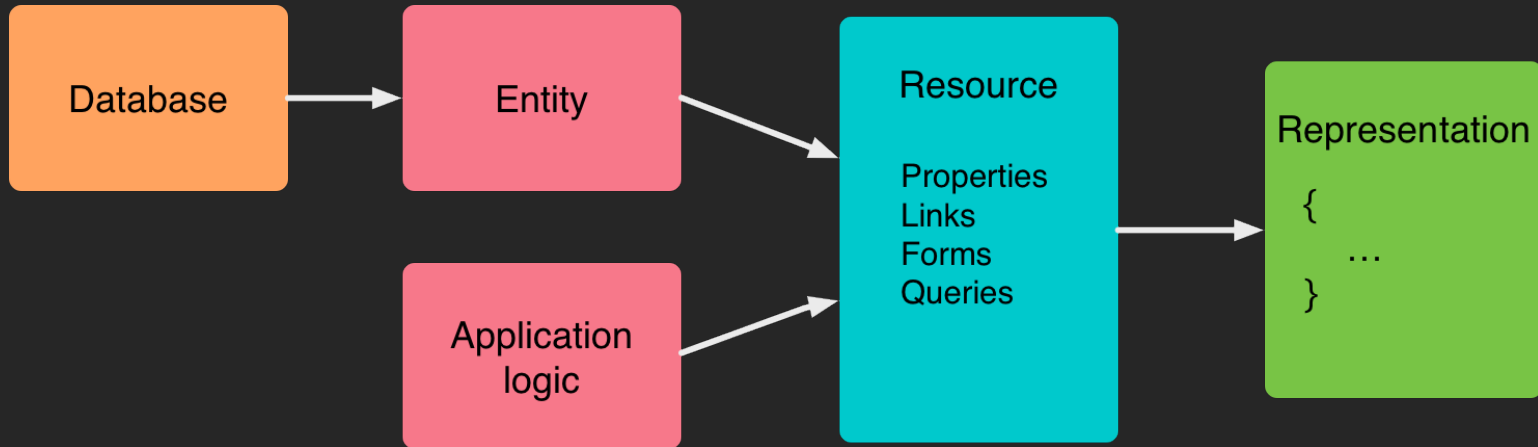
class HomePageHandler implements RequestHandlerInterface
{
    public function handle(Request $request): ResponseInterface
    {
        $response = new Response();
        $response->getBody()->write('Hello World');
        return $response;
    }
}
```

Coding time
Reading data

A close-up photograph of a potter's hands shaping a clay bowl on a pottery wheel. The hands are covered in white clay, and the wheel is spinning, creating a blurred green and blue background. The word "Malleability" is overlaid in green text.

Malleability

Decouple your representation



Hypermedia

- Media type used for a representation
- Links relationships between representations and states
 - Decouples client from server
 - Rename endpoints at will
 - Re-home endpoints on different domains

JSON and Hypermedia

JSON does not have a defined way of providing hypermedia links

Options:

- "Link" header (GitHub approach)
- application/collection+json
- application/hal+json

Hypermedia payloads

- Links should be fully qualified
- Always include `self` relation
- Add links to other related resources & collections

application/hal+json

http://stateless.co/hal_specification.html

```
{
  "_links": {
    "self": { "href": "https://example.com/orders/523" },
    "warehouse": { "href": "https://example.com/warehouse/56" },
    "invoice": { "href": "https://example.com/invoices/873" }
  },
  "currency": "GBP",
  "status": "shipped",
  "total": 123.45
}
```

Base URL in Slim

```
class BaseUrlMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $uri = $request->getUri();
        $scheme = $uri->getScheme();
        $authority = $uri->getAuthority();

        $rootUrl = ($scheme !== '' ? $scheme . ':' : '')
            . ($authority !== '' ? '//' . $authority : '');

        $request = $request->withAttribute('base_url', $rootUrl);
        return $handler->handle($request);
    }
}
```

Creating links

```
public function handle(ServerRequestInterface $request)
{
    $baseUrl = $request->getAttribute('base_url');

    $listGamesUrl = $baseUrl . '/games';
    $gameUrl = $baseUrl . '/games/' . $id;

    // ...
}
```



HAL payloads: collection

```
$hal = new Hal($baseUrl . '/games');  
foreach ($games as $game) {  
    $data['player1'] = $game->player1();  
    $data['player2'] = $game->player2();  
    ///...  
  
    $self = $baseUrl . '/games/' . $game->id;  
    $resource = new Hal($self, $data);  
    $hal->addResource('game', $resource);  
}  
  
$hal->setData(['count' => count($games)]);  
$json = $hal->asJson(true);
```


HAL output

```
$ curl http://localhost:8888/games/0a1846ab-df37-4f7b-8e42-c0ef3ec
```

```
{  
  "player1": "Rob",  
  "player2": "Jon",  
  "status": "Game complete",  
  "created": "2019-09-30 08:49:33",  
  "result": "Rob wins. rock beats scissors.",  
  "winner": "Rob",  
  "_links": {  
    ...  
  }  
}
```

HAL links

```
{  
  ...  
  "winner": "Rob",  
  "_links": {  
    "self": {  
      "href": "http://localhost:8888/games/0a1846ab-df37-4f7b-8e42...",  
    },  
    "newGame": {  
      "href": "http://localhost:8888/games/",  
      "description": "Start a new game"  
    }  
  }  
}
```

Pagination

Mobile devices don't *that* much memory!

- Link relations: `first`, `last`, `next` & `prev` relations
- Include total count of items too

Coding time

Hypermedia

Error handling



Error handling

- Internal logging of errors for ourselves
- Error representations are first class citizens
- Code for computers, messages for humans

Injecting a logger

More PSRs! 11 & 3

Configure PHP-DI

```
$containerBuilder = new ContainerBuilder();

$containerBuilder->addDefinitions([
    // Factory for a PSR-3 logger
    LoggerInterface::class => function (ContainerInterface $c) {
        $logger = new Logger($settings['name']);
        $logger->pushHandler(new ErrorLogHandler());
        return $logger;
    },
]);

AppFactory::setContainer($containerBuilder->build());
$app = AppFactory::create();
```


Configure PHP-DI

```
$containerBuilder = new ContainerBuilder();
```

```
$containerBuilder->addDefinitions([  
    // Factory for a PSR-3 logger  
    LoggerInterface::class => function (ContainerInterface $c) {  
        $logger = new Logger($settings['name']);  
        $logger->pushHandler(new ErrorLogHandler());  
        return $logger;  
    },  
]);
```

```
AppFactory::setContainer($containerBuilder->build());  
$app = AppFactory::create();
```

Configure PHP-DI

```
$containerBuilder = new ContainerBuilder();

$containerBuilder->addDefinitions([
    // Factory for a PSR-3 logger
    LoggerInterface::class => function (ContainerInterface $c) {
        $logger = new Logger($settings['name']);
        $logger->pushHandler(new ErrorLogHandler());
        return $logger;
    },
]);

AppFactory::setContainer($containerBuilder->build());
$app = AppFactory::create();
```

Configure PHP-DI

```
$containerBuilder = new ContainerBuilder();

$containerBuilder->addDefinitions([
    // Factory for a PSR-3 logger
    LoggerInterface::class => function (ContainerInterface $c) {
        $logger = new Logger($settings['name']);
        $logger->pushHandler(new ErrorLogHandler());
        return $logger;
    },
]);

AppFactory::setContainer($containerBuilder->build());
$app = AppFactory::create();
```



Configure PHP-DI

```
$containerBuilder = new ContainerBuilder();

$containerBuilder->addDefinitions([
    // Factory for a PSR-3 logger
    LoggerInterface::class => function (ContainerInterface $c) {
        $logger = new Logger($settings['name']);
        $logger->pushHandler(new ErrorLogHandler());
        return $logger;
    },
]);

AppFactory::setContainer($containerBuilder->build());
$app = AppFactory::create();
```

PHP-DI autowiring

Just type-hint your constructor!

```
class GetGameHandler implements RequestHandlerInterface
{
    private $logger;

    public function __construct(LoggerInterface $logger)
    {
        $this->logger = $logger;
    }
    ...
}
```

Logging

```
class GetGameHandler implements RequestHandlerInterface
{
    public function handle(Request $request): ResponseInterface
    {
        this->logger->info("Fetching game", ['id' => $id]);

        try {
            $games = $this->gameRepository->loadById($id);
        } catch (NotFoundException $e) {
            this->logger->info("No game found", ['id' => $id]);
            throw new HttpNotFoundException($request, 'No Game', $e);
        }
        ...
    }
}
```

Logging

```
class GetGameHandler implements RequestHandlerInterface
{
    public function handle(Request $request): ResponseInterface
    {
        this->logger->info("Fetching game", ['id' => $id]);

        try {
            $games = $this->gameRepository->loadById($id);
        } catch (NotFoundException $e) {
            this->logger->info("No game found", ['id' => $id]);
            throw new HttpNotFoundException($request, 'No Game', $e);
        }
        ...
    }
}
```

A problem has been detected and windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, please consult the hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup options, and then select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x954EBACE,0x00000002,0x00000000,0x954EBACE)

Rendering errors

Slim's error handling

Add Slim's error handling middleware to render exceptions

```
$displayDetails = true;  
$logErrors = true;  
$logDetails = true;
```

```
$app->addErrorMiddleware($displayDetails, $logErrors, $logDetails);
```



Error rendering

```
$ http -j DELETE http://localhost:8888/game
```

Error rendering

```
$ http -j DELETE http://localhost:8888/game
HTTP/1.1 405 Method Not Allowed
Allow: GET
Content-type: application/json
```

Error rendering

```
$ http -j DELETE http://localhost:8888/game
HTTP/1.1 405 Method Not Allowed
Allow: GET
Content-type: application/json
{
  "exception": [
    {
      "code": 405,
      "file": ".../Slim/Middleware/RoutingMiddleware.php",
      "message": "Method not allowed: Must be one of: GET",
      "type": "Slim\\Exception\\HttpMethodNotAllowedException"
    }
  ],
  "message": "Method not allowed: Must be one of: GET"
}
```

Convert errors to HTTP exceptions

```
public function handle(Request $request): ResponseInterface
{
    $id = $request->getAttribute('id');
    try {
        $game = $this->gameRepository->loadById($id);
    } catch (GameNotFoundException $e) {
        throw new HttpNotFoundException($request, 'No Game', $e);
    } catch (Exception $e) {
        throw new HttpInternalServerErrorException($request,
            'An unknown error occurred', $e);
    }
}
```



Not found error

```
public function handle(Request $request): ResponseInterface
{
    $id = $request->getAttribute('id');
    try {
        $game = $this->gameRepository->loadById($id);
    } catch (GameNotFoundException $e) {
        throw new HttpNotFoundException($request, 'No Game', $e);
    } catch (Exception $e) {
        throw new HttpInternalServerErrorException($request,
            'An unknown error occurred', $e);
    }
}
```



Not found error

```
$ http -j http://localhost:8888/games/1234
```

```
HTTP/1.1 404 Not Found
```

```
Content-type: application/json
```

```
{  
  "message": "No Game"  
}
```

```
(Production mode: $displayDetails = false)
```

Generic error

```
public function handle(Request $request): ResponseInterface
{
    $id = $request->getAttribute('id');
    try {
        $game = $this->gameRepository->loadById($id);
    } catch (GameNotFoundException $e) {
        throw new HttpNotFoundException($request, 'No Game', $e);
    } catch (Exception $e) {
        throw new HttpInternalServerErrorException($request,
            'An unknown error occurred', $e);
    }
}
```



Generic error

```
$ http -j http://localhost:8888/games/abcd
HTTP/1.1 500 Internal Server Error
Content-type: application/json
```

```
{
  "exception": [
    {
      "code": 500,
      "file": ".../src/Handler/GetGameHandler.php",
      "line": 43,
      "message": "An unknown error occurred",
      "type": "Slim\\Exception\\HttpInternalServerErrorException"
    },
  ],
}
```

```
{
    "code": 40,
    "file": ".../lib/Assert/Assertion.php",
    "line": 2752,
    "message": "Value \"abcd\" is not a valid integer.",
    "type": "Assert\\InvalidArgumentException"
},
"message": "An unknown error occurred"
}
```

(Development mode: `$displayDetails = true`)

Coding time

Error responses



Documentation

Documentation

- Tutorials
- Reference



OpenAPI Specification

- Spec-first API design
- Tooling: <https://openapi.tools>
 - Reference website: ReDoc
 - Linting/validation: Spectral
 - Mock server: Prism

OpenAPI Specification

openapi: "3.0.2"

info:

title: Rock-Paper-Scissors

version: "1.0.0"

description: An implementation of Rock-Paper-Scissors

contact:

name: Rob Allen

servers:

- **url:** <https://rock-paper-scissors.example.com>

OpenAPI Specification

paths:

/games:

post:

summary: Create a new game

description: Create a new game of Rock-Paper-Scissors

operationId: createGame

tags:

- Game

requestBody:

description: Game to add

content:

application/json:

schema:

\$ref: '#/components/schemas/NewGameRequest'

Test with prism

```
$ npm install -g @stoplight/prism-cli
```

```
$ prism mock rps-openapi.yaml
```

```
> [CLI] ... awaiting Starting Prism...  
> [HTTP SERVER] info  Server listening at http://127.0.0.1:4010  
> [CLI] info  POST   http://127.0.0.1:4010/games  
> [CLI] info  POST   http://127.0.0.1:4010/games/e3..1/moves
```

Test with prism

```
$ curl -i -X POST http://localhost:4010/games
```

```
HTTP/1.1 400 Bad Request
```

```
content-type: application/json
```

```
{"message": "Must provide both player1 and player2"}
```

Test with prism

```
$ curl -i -X POST http://localhost:4010/games
HTTP/1.1 400 Bad Request
content-type: application/json
```

```
{"message": "Must provide both player1 and player2"}
```

- › [HTTP SERVER] post /games info Request received
- › [NEGOTIATOR] info Request contains an accept header: */*
- › [VALIDATOR] warning Request did not pass the validation rules
- › [NEGOTIATOR] success Found response 400.
- › [NEGOTIATOR] success The response 400 has a schema.
- › [NEGOTIATOR] success Responding with status code 400
- › [VALIDATOR] error Violation: request Body parameter is required

Demo

The RPS OpenAPI Spec

A photograph of a space shuttle launching at dusk or dawn. The shuttle is ascending vertically, leaving a large, bright, billowing plume of white and orange smoke and fire. Three tall, slender service towers are visible in the background, flanking the launch pad. The sky is a deep, dark blue, and the horizon shows a faint glow of light. The overall scene is dramatic and captures the power of the launch.

To sum up

To sum up

- HTTP method negotiation
- Content-type negotiation
- Hypermedia
- Error handling
- Documentation

Resources

- <https://akrabat.com/category/api/>
- <https://akrabat.com/category/slim-framework/>
- <https://github.com/akrabat/slim4-rps-api>

- <https://apihandyman.io>
- <https://apievangelist.com>
- <http://restcookbook.com>

Thank you!

<https://joind.in/talk/ffd1d>

Rob Allen - Independent API developer

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

Photo credits

- The Fat Controller: HiT Entertainment
- Foundation: <https://www.flickr.com/photos/armchairbuilder/6196473431>
- APIs: <https://www.flickr.com/photos/ebothy/15723500675>
- Incoming Data: <https://www.flickr.com/photos/natspressoffice/13085089605>
- Computer code: <https://www.flickr.com/photos/n3wjack/3856456237/>
- Road sign: <https://www.flickr.com/photos/ell-r-brown/6804246004>
- Car crash: EuroNCAP
- Writing: <https://www.flickr.com/photos/froderik/9355085596/>
- Error screen: <https://www.flickr.com/photos/lrosa/2867091465/>
- Books: <https://www.flickr.com/photos/mdales/48981707361/>
- Pattery wheel: <https://www.flickr.com/photos/61091655@N00/6831352744/>
- Rocket launch: <https://www.flickr.com/photos/gsfrc/16495356966>
- Stars: <https://www.flickr.com/photos/gsfrc/19125041621>