SOA
Restful Web services
PHP Slim

Framework Slim

* « Micro » framework specialised to routing, Rest APIs and web apps.
  * HTTP router, supports parameters, pattern matching
  * PSR-7 HTTP message implementation -> may inspect and manipulate HTTP message method, status, URI, headers, cookies, and body.

Framework Slim

* First example

```
<?php
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Slim\Factory\AppFactory;
require __DIR__ . '/../vendor/autoload.php';

$app = AppFactory::create();
$app->addRoutingMiddleware();
```

Framework Slim

* First example

```
/** * Add Error Middleware
* @param bool $displayErrorDetails -> Should be set to false in production
* @param bool $logErrors -> Parameter is passed to the default ErrorHandler
* @param bool $logErrorDetails -> Display error details in error log
* @param LoggerInterface|null $logger -> Optional PSR-3 Logger

* * Note: This middleware should be added last. It will not handle any exceptions/errors
* for middleware added after it.
*/
$errorMiddleware = $app->addErrorMiddleware(true, true, true);
```

Framework Slim

* First example

```
// Define app routes
$app->get('/hello/{name}', function (Request $request, Response
$response, $args) {
$name = $args['name'];
$response->getBody()->write("Hello, $name");
return $response; });

// Run app
$app->run();
```

Framework Slim

* Route definition (*$app->get in the exemple*)

* With:
* get(), post(), put(), delete(),
* patch() (modification partielles)
* head() (entètes retournées si Get utilisé)
* and options() (option de com. Test existance ressource)

Framework Slim

Slight digression : Composer

* Dependency manager in PHP
* Allows the declaration of dependent libs

see

* https://getcomposer.org/doc/00-intro.md
  * get the PHP binary archive composer.phar

Framework Slim

Slight digression : Composer

* Use composer.json to write dependencies:
  { “require“:
  {“slim/slim“: “4. “}*
  }
* or php *composer.phar require slim/slim:"4.* » (doc)*
* *php composer.phar require slim/psr7*

Installed libs available into the *vendor* directory

Framework Slim

Slight digression : Composer

* Composer integrates its own PRS-4 class autoloader

```
<?php
    require 'vendor/autoload.php';
```

Framework Slim

PSR7 -> Resquest and Response

* Request and Response objects are *immutable value objects*. They are cloned when methods are called

```
$app = AppFactory::create();
$app->get('/foo', function (Request $request, Response $response, array $args) {
$payload = json_encode(['hello' => 'world'], JSON_PRETTY_PRINT);
$response->getBody()->write($payload);
return $response->withHeader('Content-Type', 'application/json'); });
$app->run();
```

Framework Slim

Request and Response

PSR-7 interface provides methods to transform Request and Response objects.
[Doc https://www.php-fig.org/psr/psr-7/](https://www.php-fig.org/psr/psr-7/)

Ex:

- *withHeader($name, $value)*
- *withBody(StreamInterface $body)*

Response

- *withStatus($code, $reasonPhrase = '')*

Framework Slim

Request management

PSR-7 interface provides methods to transform Request objects:

- *withMethod($method)*
- *withUri(UriInterface $uri, $preserveHost = false)*
- *withCookieParams(array $cookies)*
- *withQueryParams(array $query)*
- *withUploadedFiles(array $uploadedFiles)*
- *withParsedBody($data)*
- *withAttribute($name, $value)*
- *withoutAttribute($name)*

Framework Slim

Request with Query params

Example: (url/?var=value&var2=value2)

How to get parameters ?

$params = $request->getQueryParams();
var_dump($params);

$params is an array of the form

'var' => value
'var2' => value2

Framework Slim

Request management

Serialisation JSON or XML data
Can be done with Middleware:

```
$app->addBodyParsingMiddleware();
$app->addRoutingMiddleware();
$app->addErrorMiddleware(true, true, true);

$app->post('/', function (Request $request, Response $response, $args):
Response {

$data = (array)$request->getParsedBody();
```

Framework Slim

Request management

parameters in URL

Get them with routes :

$app->get('/books/{id}', function ($request, $response, array $args) {

// Show book identified by $args['id'] });

Framework Slim

Request Management

(optional) parameters in URL

```
$app->any('/books/[{id}]', function ($request, $response, array $args)
{
// Apply changes to books or book identified by $args['id'] if specified.
// /[{id}] optional -> all books or book id

// To check which method is used: $request->getMethod(); });
```

Framework Slim

Request Management

Route names : useful for Hateos

```
$app->get('/hello/{name}', function ($request, $response, array $args) {
$response->getBody()->write("Hello, " . $args['name']); return
$response; })->setName('hello');

$routeParser = $app->getRouteCollector()->getRouteParser();
echo $routeParser->urlFor('hello', ['name' => 'Josh'], ['example' =>
'name2']);
// Outputs "/hello/Josh?example=name2"
```

Framework Slim

Response Management

* Returning plain text

```
$body = $response->getBody();
$body->write('Hello');
Return $response;
```

Framework Slim

Response Management

* Returning JSON

```
$data = array('name' => 'Rob', 'age' => 40);
$payload = json_encode($data);
$response->getBody()->write($payload);
return $response
        –>withHeader('Content-Type', 'application/json')
        ->withStatus(201); // 201 created
```

Framework Slim

Error management
With middleware

/** * Add Error Middleware *
*@param bool $displayErrorDetails -> Should be set to false in production
* @param bool $logErrors -> Parameter is passed to the default ErrorHandler
* @param bool $logErrorDetails -> Display error details in error log
* @param LoggerInterface|null $logger -> Optional PSR-3 Logger *
* Note: This middleware should be added last. It will not handle any exceptions/errors * for middleware added after it. */

$errorMiddleware = $app->addErrorMiddleware(true, true, true);

Framework Slim

Error management

Use these Exceptions

- HttpBadRequestException
- HttpForbiddenException
- HttpInternalServerErrorException
- HttpMethodNotAllowedException
- HttpNotFoundException
- HttpNotImplementedException
- HttpUnauthorizedException

Framework Slim

Error management

Make your own exception

```
class HttpForbiddenException extends
            HttpSpecializedException {
protected $code = 504;
protected $message = 'Gateway Timeout.';
protected $title = '504 Gateway Timeout'; protected $description =
'Timed out before receiving response from the upstream server.'; }
```

Framework Slim

Log management

Monolog see doc.

https://github.com/Seldaek/monolog/

PHP client side

PHP Client side

* Guzzle https://docs.guzzlephp.org/en/stable/
* HTTP PHP Client framework

PHP Client side

* Guzzle sync request

```
use GuzzleHttp\Psr7\Request;
$request = new Request('PUT', 'http://httpbin.org/put');
$response = $client->send($request, ['timeout' => 2]);
```

Or

```
$client = new GuzzleHttp\Client(['base_uri' => 'https://foo.com/api/']);
// Send a request to https://foo.com/api/test
$response = $client->request('GET', 'test');

$response = $client->request('GET', 'http://httpbin.org?foo=bar');
```

PHP Client side

* Guzzle sync request with header

```
use GuzzleHttp\Psr7\Request;
$request = new Request('GET', 'http://httpbin.org/get');

// You can provide other optional constructor arguments.

$headers = ['X-Foo' => 'Bar'];

$body = 'hello!';
$request = new Request('PUT', 'http://httpbin.org/put', $headers, $body);
```

PHP Client side

* Guzzle sync request

```
$client = new GuzzleHttp\Client();
$res = $client->request('GET', 'https://api.github.com/user', [ 'auth' => ['user', 'pass'] ]);
echo $res->getStatusCode();
echo $res->getHeader('content-type')[0];
echo $res->getBody();
```

* Guzzle async request

```
// Send an asynchronous request.
$request = new \GuzzleHttp\Psr7\Request('GET', 'http://httpbin.org');
$promise = $client->sendAsync($request)->then(function ($response) { echo 'I completed! ' .
$response->getBody(); });
$promise->wait();
```

PHP Client side

* Manipulating Response objects in the client :

```
$response->getStatusCode();

// Get all of the response headers.
foreach ($response->getHeaders() as $name => $values) { echo
$name . ': ' . implode(', ', $values) . "\r\n"; }

$body = $response->getBody();
```

PHP Client side

* Error management in Guzzle : Exception

```
try { $client->request('GET’,
        'https://github.com/_abc_123_404'); }
catch (ClientException $e) {…}
```

Testing

Integration, functional testing

* Selenium
* https://rest-assured.io/
* https://github.com/intuit/karate
* https://citrusframework.org/ (integration testing, BD)

Manual testing

* Postman

Performance

* http://jmeter.apache.org/index.html
* https://gettaurus.org/

Mock testing

* https://github.com/mock-server/mockserver
* https://hoverfly.io/