Improve your API with OpenAPI

Rob Allen

PHPSW, November 2021
The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service.

https://spec.openapis.org/oas/latest.html
It's about documentation
It's about design-first
It's about communicating changes
It's about development workflows
It's about

a contract
Anatomy of the specification
openapi: "3.1.0"  # or "3.0.3"

info:  # ...
servers:  # ...
paths:  # ...
webhooks:  # ...
components:  # ...
security:  # ...
tags:  # ...
externalDocs:  # ...

Rob Allen ~ @akrabat
openapi.yaml

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
    description: "RPS production API"
paths:
  post:
    summary: Create a new game
    description: Create a new game of Rock-Paper-Scissors.
    requestBody:
      # ...
    responses:
      # ...
openapi.yaml

requestBody:
  description: Game to add
  required: true
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/NewGameRequest'
Reuse of objects

$ref allows us to define one & use in many places

components:
schemas:
  GameId:
    type: string
    examples:
      - "2BC08389-885A-4322-80D0-EF0DE2D7CD37"
    format: "uuid"
  Player:
    type: string
    example: "Rob"
Build on other components

```json
schemas:
  NewGameRequest:
    properties:
      player1:
        $ref: '#/components/schemas/Player'
      player2:
        $ref: '#/components/schemas/Player'
    required:
      - player1
      - player2
  examples:
    - '{"player1":"Ian", "player2":"Dave"}'
```

Rob Allen ~ @akrabat
openapi.yaml

responses:
  "201":
    $ref: '#/components/responses/NewGameResponse'
  "400":
    $ref: '#/components/responses/NewGameError'
  "500":
    $ref: '#/components/responses/InternalServerError'
Writing your spec
Editing

- Editor with plugins: vim, VS Code, etc
- GUI: Stoplight, OpenAPI-GUI, Swagger Editor
Linting

CLI tools: Spectral, openapi-spec-validator, etc.

$ spectral lint rps-openapi.yaml

.../slim4-rps-api/doc/rps-openapi.yaml
3:6 warning info-contact  Info object must have "contact" object.  info

1 problem (0 errors, 1 warning, 0 infos, 0 hints)
Game

All the operations you need to play a game of Rock-Paper-Scissors.

Create a new game

Create a new game of Rock-Paper-Scissors

REQUEST BODY SCHEMA: application/json

Game to add

- player1 (required)
  - string (schema)
- player2 (required)
  - string (schema)

Responses

- 201: The game was successfully created.
- 401: Unable to create game due to a problem with the provided information.

Request samples

Payload

```
Content type
application/json

Copy
Expand all
Collapse all

{   "player1": "Rob Allen",
    "player2": "Mathieu Napoli"
}
```

Response samples

```
201
401
500
```

Content type
application/json
Numbers API

The Numbers API enables you to manage your existing numbers and buy new virtual numbers for use with the Vonage APIs. Further information is here: https://developer.nexmo.com/numbers/overview

Available Operations:
- `GET` List the numbers you own
- `GET` Search available numbers
- `POST` Buy a number
- `PUT` Cancel a number
- `PUT` Update a number

List the numbers you own

Retrieve all the inbound numbers associated with your Vonage account.

```
GET https://rest.nexmo.com/account/numbers
```

Example Responses

```
{
  "count": 1,
  "numbers": [
    {
      "country": "GB",
      "msisdn": "66770000000"
    }
  ]
}
```
Mock server

$ prism mock rps-openapi.yaml

$ curl http://127.0.0.1:4010/games -d '{}'

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

The schema section can be used to validate the request and response

- Validate early and return a 422
- Validate that we return what we say we will
- Put it in CI to prevent regressions

Rob Allen ~ @akrabat
But I already have validation!

Your code:

• isn't good enough!
• isn't reusable!
• doesn't match the docs!
But I already have validation!

Your code:
• isn't good enough!
• isn't reusable!
• doesn't match the docs!

However...

Business logic validation still needed!

Rob Allen ~ @akrabat
Validation in PHP

OpenAPI 3.0:
    league/openapi-psr7-validator

OpenAPI 3.1:
    opis/json-schema
Validation middleware

```php
public function process($request, $handler)
{
    // Determine OAS schema for this route into $schema

    // Validate request
    $builder = (new ValidatorBuilder())->fromSchema($schema);
    $validator = $builder->getServerRequestValidator();
    try {
        $match = $validator->validate($request);
    } catch (ValidationFailed $e) {
        throw new RequestValidationFailed($e);
    }
}
```
Validation middleware

```php
public function process($request, handler)
{
    // Determine OAS schema for this route into $schema

    // Validate request
    $builder = (new ValidatorBuilder())->fromSchema($schema);
    $validator = $builder->getServerRequestValidator();
    try {
        $match = $validator->validate($request);
    } catch (ValidationFailed $e) {
        throw new RequestValidationFailed($e);
    }
}
```
Validation middleware

```php
public function process($request, handler)
{
    // Determine OAS schema for this route into $schema

    // Validate request
    $builder = (new ValidatorBuilder())->fromSchema($schema);
    $validator = $builder->getServerRequestValidator();
    try {
        $match = $validator->validate($request);
    } catch (ValidationFailed $e) {
        throw new RequestValidationFailed($e);
    }
}
```
// Process request
$response = $handler->handle($request);

// Validate response
try {
    $builder->getResponseValidator()->validate($match, $response);
} catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
}
return $response;
Validation middleware (cont)

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

// Validate response
try {
    $builder->getResponseValidator()
        ->validate($match, $response);
} catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
}
return $response;
// Process request
$response = $handler->handle($request);

// Validate response
try {
    $builder->getResponseValidator()
        ->validate($match, $response);
} catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
}
return $response;
Validation middleware (cont)

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

// Validate response
try {
    $builder->getResponseValidator()->validate($match, $response);
} catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
}
return $response;

Rob Allen ~ @akrabat
Validation Middleware in Laravel

```php
// composer install softonic/laravel-psr15-bridge

// app/Http/Kernel.php
use \League\OpenAPIValidation\PSR15\ValidationMiddleware;
protected $routeMiddleware = [
    // ...
    'openapi-validation' => ValidationMiddleware::class,
];

// Routes file
Route::post('/game', 'GameController@create')
    ->middleware('openapi-validation');
```

Rob Allen ~ @akrabat
To sum up
Resources

- https://www.openapis.org
- https://openapi.tools
- https://github.com/akrabat/slim4-rps-api
- https://github.com/thephpleague/openapi-psr7-validator
- https://github.com/primitivesocial/openapi-initializer
Thank you!

https://joind.in/talk/08577