Radosław Zagórski
www.linkedin.com/in/radoslawzagorski

REST - Representational state transfer

/author/123

/author/123/posts

/author/123/posts/82

What we need?

symfony/framework-standard-edition

friendsofsymfony/rest-bundle

jms/serializer-bundle

nelmio/api-doc-bundle

CRUD

Create (POST)

Read (GET)

Update (PUT or PATCH)

Delete (DELETE)

Create - HTTP POST

Request:

POST /attendee HTTP/1.1

Host: 127.0.0.1:8000

Content-Type: application/json

```
{
    "name": “JCommerce User",
    "email": “rekrutacja@jcommerce.pl“
}
```

Response:

HTTP/1.1 201 Created

Content-Type: application/json

Location: /attendee/1

No Body

Create - HTTP POST

```php
/**
 * @Rest\Post("/attendee", name="_add_attendee", defaults={"_format"="json"})
 */
public function addAction(Request $request)
{
    $attendee = $this->deserialize(Attendee::class, $request);

    if ($attendee instanceof Attendee === false) {
        return View::create(array('errors' => $attendee), Response::HTTP_BAD_REQUEST);
    }

    $attendee = $this->get('attendee.service')->save($attendee);

    $url = $this->generateUrl(
        '_show_attendee',
        array('id' => $attendee->getId()),
        true
    );

    $response = new JsonResponse(['id' => $attendee->getId()]);
    $response->setStatusCode(Response::HTTP_CREATED);
    $response->headers->set('Location', $url);

    return $response;
}
```

Read - HTTP GET

Request:

GET /attendee/1 HTTP/1.1

Host: 127.0.0.1:8000

Response:

HTTP/1.1 200 OK

Content-Type: application/json


```
{
        "attendee": {
                "id": 1,
                "email": “rekrutacja@jcommerce.pl”,
                "name": “JCommerce User”,
                "created_at": "2017-11-29 22:10:18"
        }
}
```

Read - HTTP GET

```php
    /**
     * @Rest\Get("/attendee/{id}", name="_show_attendee", defaults={"_format"="json"})
     */
public function getAction(Attendee $attendee)
{
    $response = new JsonResponse(
        $this->get('jms_serializer')->serialize(
            ['attendee' => $attendee],
            'json'),
        Response::HTTP_OK, [], true
    );

    return $response;
}
```

Update - HTTP PUT

Request:

PUT /attendee/1 HTTP/1.1

Host: 127.0.0.1:8000

> Content-Type: application/json

{

"name": “John Doe”

}

Response:

HTTP/1.1 200 OK

Content-Type: application/json

{

"attendee": {

"id": 1,

"email": "",

“name": “John Doe,

"created_at": "2017-11-29 22:10:18"

}

}

Update - HTTP PUT

```php
/**
 * @Rest\Put("/attendee/{id}", name="_update_attendee", defaults={"_format"="json"})
 */
public function putAction(Attendee $attendee, Request $request)
{
    $newAttendee = $this->deserialize(Attendee::class, $request);

    if ($newAttendee instanceof Attendee === false) {
        return View::create(['errors' => $newAttendee], Response::HTTP_BAD_REQUEST);
    }

    $attendee = $this->get('attendee.service')->merge($attendee, $newAttendee);

    $response = new JsonResponse(
        $this->get('jms_serializer')->serialize(
            ['attendee' => $attendee],
            'json'),
        Response::HTTP_OK, [], true
    );

    return $response;
}
```

UPDATE

PUT or PATCH?

Delete - HTTP DELETE

Request:

DELETE /attendee/1 HTTP/1.1

Host: 127.0.0.1:8000

Content-Type: application/json

No Body

Response:

HTTP/1.1 204 No Content

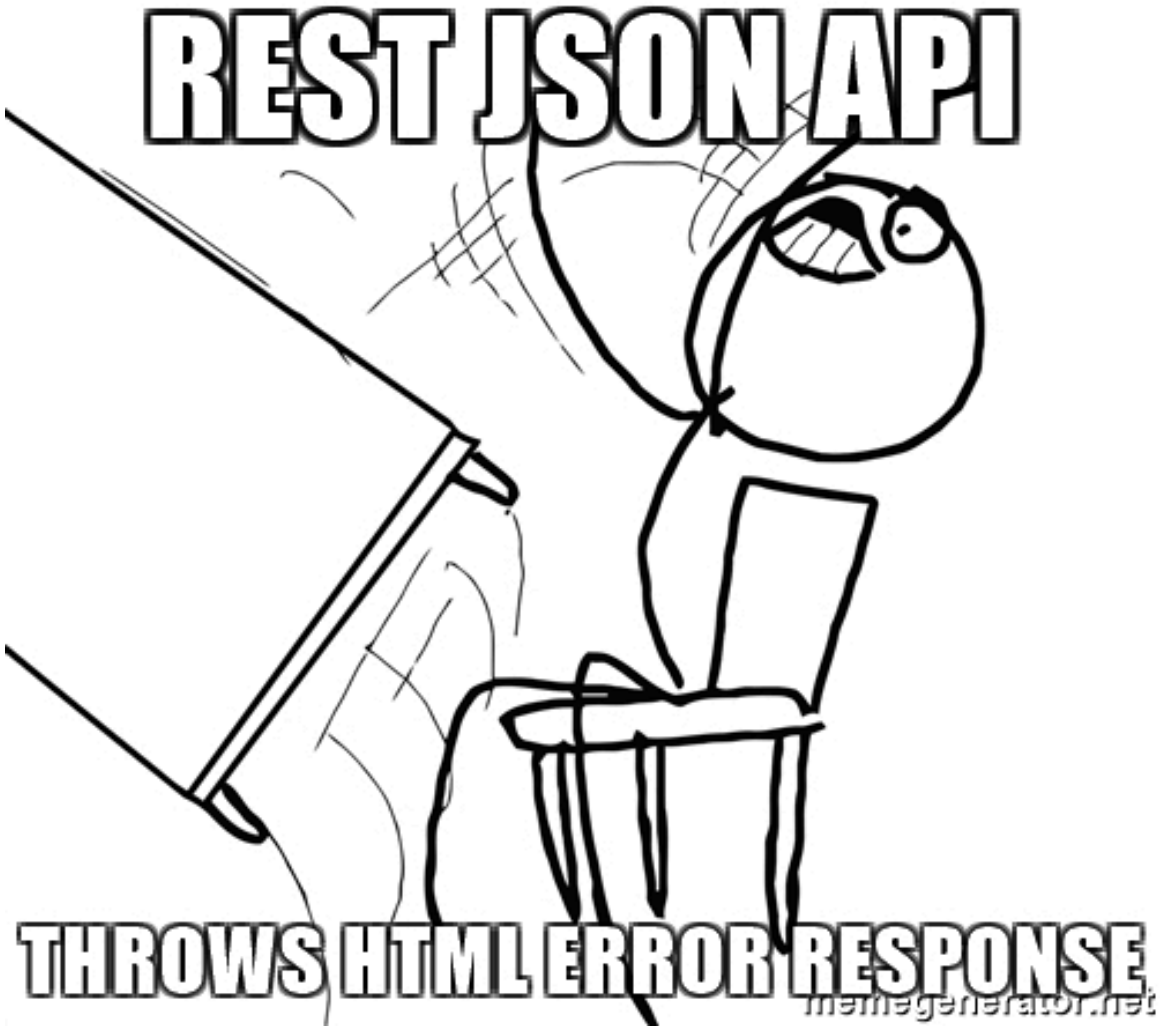
Content-Type: application/json

No Body

Delete - HTTP DELETE

```php
/**
 *
 * @Rest\Delete("/attendee/{id}", name="_delete_attendee", defaults={"_format"="json"})
 *
 * @Rest\View(statusCode=204)
 */
public function deleteAction(Attendee $attendee)
{
    $this->get('attendee.service')->remove($attendee);
}
```

Errors?



Errors?

- Machine doesn’t understand errors
- Handle every exception
- Always throw exception in business logic
- Test exceptions on DEV env and LIVE env
- Clear error message

How to protect?

- Functional tests (more is better)
- Create exceptions
- Kernel Listeners (onKernelException)
- Middleware (http://stackphp.com)

Errors? - example

```
{
  "status" : 400,
  "developerMessage" : "Verbose, plain language description of the problem. Provide developers
   suggestions about how to solve their problems here",
  "userMessage" : "This is a message that can be passed along to end-users, if needed.",
  "errorCode" : “444444", // internal code number
  "moreInfo" : “http://www.example.gov/developer/path/to/help/for/444444
}
```

Versions

- Never release an API without a version number.
- Versions should be integers, not decimal numbers, prefixed with ‘v’. For example:
  - Good: v1, v2, v3
  - Bad: v-1.1, v1.2, 1.3
- Maintain APIs at least one version back.

Versions - URL versioning

http://example.com/api/v1/user/1

http://v1.example.com/user/1

Versions - HTTP Header

GET /attendee/1 HTTP/1.1

Host: 127.0.0.1:8000

API-Version: 1

Versions - HATEOAS

GET /attendee/1 HTTP/1.1

Host: 127.0.0.1:8000

Accept: application/vnd.example.meetup-v1+json

Api Documentation

```
@ApiDoc(
    section="Attendees",
    resource=true,
    description="Get full date about Meetup Attendee",
    statusCodes={
        200="OK"
    }
)
```

- Nelmio Api Documentor (https://github.com/nelmio/NelmioApiDocBundle)
- Swagger (https://swagger.io)

Testing

Testing

- Unit Testing
- Functional Testing (Guzzle)
- Quick testing during development (Insomnia, PHPStorm Rest Client, Postman)
- Performance (JMeter, Gatling, Tsung, Flood)

We make IT JCommerce


JET
BRAINS

Thank you!