FosRestBundle

1. **Installation :**

composer require friendsofsymfony/rest-bundle

Activer le bundle dans AppKernel.php

**new** FOS\RestBundle\FOSRestBundle(),

FosRestBundle besoin d’un serializer pour ce la on va installer jmsSerializer :

composer require jms/serializer-bundle

activer le bundle dans Appkernel.php

**new** JMS\SerializerBundle\JMSSerializerBundle(),

1. **Configuration :**

**fos\_rest:**

**body\_converter: #pour faire la serialisation et la deserialisation auto**

**enabled:** true

**view: #pour chosir le format de reponse**

**formats:** { **json:** true, **xml:** false, **rss:** false }

**view\_response\_listener:** true

**serializer: #pour serialiser un objet meme il est vide**

**serialize\_null:** true

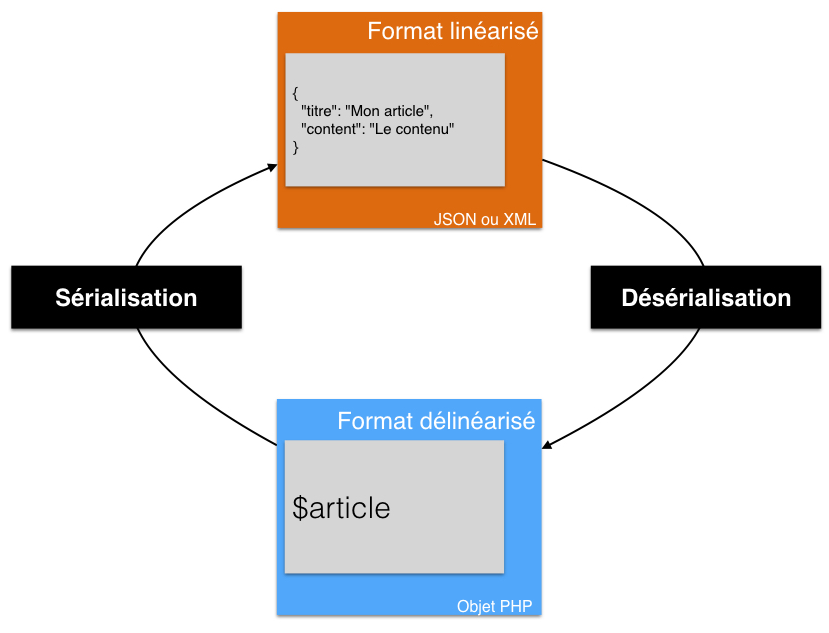
**format\_listener: #pour que les autres pages s’affiche**

**rules:**

- { **path:** '^/api', **priorities:** ['json'], **fallback\_format:** 'json' }

- { **path:** '^/', **priorities:** [ 'text/html', '\*/\*'], **fallback\_format:** html, **prefer\_extension:** true }

1. **La sérialisation :**

****

**use** FOS\RestBundle\Controller\Annotations as Rest;

**use** FOS\RestBundle\Controller\Annotations\View;

**use** FOS\RestBundle\Controller\FOSRestController;

**class** BrandController **extends** FOSRestController

{

*/\*\**

*\** ***@Rest\Get****(*

*\* path = "/api/brands",*

*\* name = "api\_brand\_list",*

*\**

*\* )*

*\** ***@View****(StatusCode = 200)*

*\*/*

**public function** indexAction()

{

$brandService=$this->get('core.service.brand');

$brands=$brandService->getActiveBrands(**false**);

**return** $brands;

}

*/\*\**

*\** ***@Rest\Get****(*

*\* path = "/api/brands/{id}",*

*\* name = "api\_brand\_list",*

*\* requirements = {"id"="\d+"}*

*\* )*

*\** ***@View****(StatusCode = 200)*

*\*/*

**public function show**Action($id)

{

$brandService=$this->get('core.service.brand');

$brands=$brandService->getBrand(**$id**);

**return** $brand;

}

}

1. **La désérialisation :**

@ParamConverter("customer", converter="fos\_rest.request\_body") permet defaire la désérialisation automatique

/\*\*

*\** ***@Post****(*

*\* path = "api/customers",*

*\* name = "api\_customer\_create",*

*\* )*

*\** ***@View****(StatusCode = 201)*

*\** ***@ParamConverter****("customer", converter="fos\_rest.request\_body")*

*\**

*\*/*

**public function** createAction(Customer $customer)

{

$em = $this->getDoctrine()->getManager();

$customer->setPwd(sha1($customer->getPwd()));

$em->persist($customer);

$address->setCustomer($customer);

$em->flush();}

1. **Valider des ressources :**

ajouter cette ligne de configuration :

**fos\_rest:**

**body\_converter:**

**enabled:** true

**validate:** true

**validation\_errors\_argument:** violations

validate: true : permet de faire que la validation de l'objet converti soit déclenchée juste avant qu'il soit passé à l'action ;

validation\_errors\_argument: violations : il s'agit du nom de l'argument contenant un objet de type ConstraintViolationList, contenant toutes les erreurs résultantes de la validation de l'objet.

<?php

**namespace** AppBundle\Controller;

// …

**use** FOS\RestBundle\Controller\FOSRestController;

**use** Symfony\Component\Validator\ConstraintViolationList;

**use** Symfony\Component\HttpFoundation\Response;

**class** ArticleController **extends** FOSRestController

{

*/\*\**

*\** ***@Rest\Post****("/articles")*

*\** ***@Rest\View****(StatusCode = 201)*

*\** ***@ParamConverter****("article", converter="fos\_rest.request\_body")*

*\*/*

**public function** createAction(Article $article, ConstraintViolationList $violations)

{

**if** (count($violations)) {

**return** $this->view($violations, Response::*HTTP\_BAD\_REQUEST*);

}

$em = $this->getDoctrine()->getManager();

$em->persist($article);

$em->flush();

**return** $article;

}

}

$violations et de méme nom que violations dans **validation\_errors\_argument:**

On doit créer des assertion pour la validation

**/\*\***

**\* @ORM\Column(type="text")**

**\* @Expose**

**\* @Assert\NotBlank**

**\*/**

**private $content;**

1. **Gestion des erreurs :**

Pour ce faire, plusieurs possibilités :

-**via la configuration**, il est possible de choisir le controller appelé à chaque fois qu'une exception est lancée ;

-**via un listener**, il est possible d'écouter l'évènement kernel.exception , évènement qui est lancé à chaque fois d'une exception est lancée ;

-**via un serializer handler**, un service qui se charge de formater le contenu de la réponse.

**Via la configuration :**

ajouter la ligne de configuration

**fos\_rest:**

**# …**

**exception:**

**enabled: true**

**codes:**

**{ AppBundle\Exception\ResourceValidationException: 400 }**

Il nous faut créer la classe d'exception ResourceValidationException. Créons donc un nouveau dossier Exception avec la classe suivante :

**<?php**

**namespace AppBundle\Exception;**

**class ResourceValidationException extends \Exception**

**{**

**}**

**utilisation depuis notre controller**

**class ArticleController extends FOSRestController**

**{**

**/\*\***

**\* @Rest\Post("/articles")**

**\* @Rest\View(StatusCode = 201)**

**\* @ParamConverter("article", converter="fos\_rest.request\_body")**

**\*/**

**public function createAction(Article $article, ConstraintViolationList $violations)**

**{**

**if (count($violations)) {**

**$message = 'The JSON sent contains invalid data. Here are the errors you need to correct: ';**

**foreach ($violations as $violation) {**

**$message .= sprintf("Field %s: %s ", $violation->getPropertyPath(), $violation->getMessage());**

**}**

**throw new ResourceValidationException($message);**

**}**

**// …**

**}**