

## Workshop : JSON

### Objectif

L'objectif de ce workshop est de manipuler JSON côté serveur dans une application symfony4

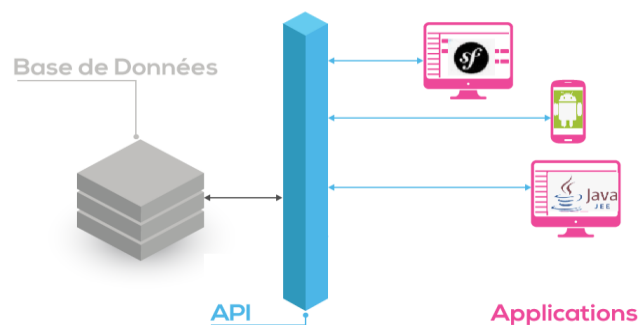


Figure 1: Consommation d'une API par les trois parties Java, Symfony et mobile

### I- Le principe de sérialisation

- JSON est un format de données léger, facile à lire et à écrire et compatible avec pas mal de langages de développement.
- Le framework PHP Symfony nous offre un composant **Serializer** pour sérialiser les objets en différents formats (json, xml ...).
- On peut utiliser la sérialisation dans une API, des microservices, services et pour la récupération d'objets depuis la base de données.

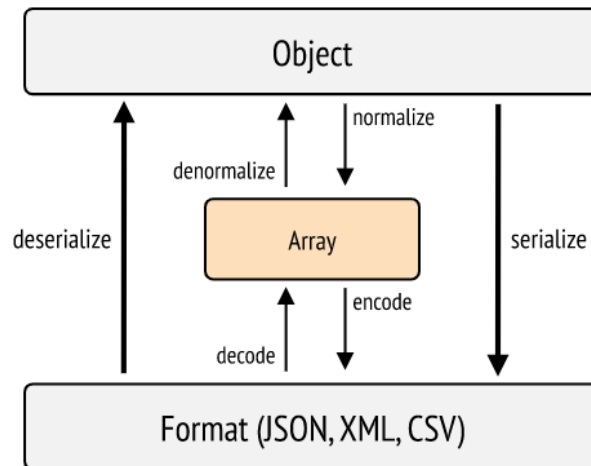


Figure 2: La sérialisation des objets

- D'après la figure ci-dessus, le « **serializer** » permet de :
  - ✓ Transformer des objets en un array à travers « **normalize** ».
  - ✓ Transformer ces tableaux en format JSON ou XML à travers « **encode** ».
- Le « **deserializer** » fait l'inverse, il décode le format en un tableau avant de le transformer en un objet.

## II- CRUD d'une entité avec le format JSON

### 1. Initialisation

- Soit le contrôleur « **StudentController** » et une entité « **Student** » qui contient :
  - ✓ id : clé primaire, integer
  - ✓ nsc : string
  - ✓ Email : string

### 2. Liste des étudiants

La fonction « AllStudents » permet d'afficher la liste des étudiant

```

//Définir le chemin.
/**
 * @Route("/AllStudents", name="AllStudents")
 */
//Dans cette fonction, nous utilisons le service NormalizerInterface en utilisant
//l'injection de dépendances
public function AllStudents(Request $request, NormalizerInterface $Normalizer )
{
    //Nous utilisons la Repository pour récupérer les objets que nous avons dans la base de données
    $repository = $this->getDoctrine()->getRepository(Student::class);
    $students = $repository->findAll();
    //Nous utilisons la fonction normalize qui transforme en format JSON nos donnée qui sont
    //en tableau d'objet Students
    $jsonContent = $Normalizer->normalize($students, 'json', ['groups'=>'post:read']);
    //Nous renvoyons une réponse TWIG qui prend en paramètre un tableau en format JSON
    return $this->render('student/allStudentJSON.html.twig', [
        'data'=> $jsonContent,
    ]);
}

```

- On peut aussi utiliser « **new Response** » comme suit :

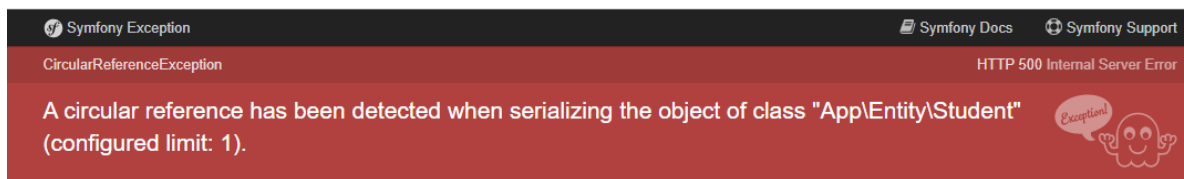
```
return new Response(json_encode($jsonContent));
```

- N'oubliez pas d'ajouter les use des classes utilisées

```
use Symfony\Component\Serializer\Normalizer\NormalizerInterface;
```

### Remarque :

- Pour éviter l'erreur "Reference Circulaire" (Figure2), Il faut indiquer/taguer les attributs qu'on va sérialiser en utilisant l'annotation **@Groups**. Dans notre cas, on a nommé **@Groups("post:read")** et on a l'utilisée comme paramètre dans la fonction **normalize()**.



**Figure 3: Erreur Reference Circulaire**

```
Student.php

/**
 * @ORM\Column(type="integer")
 * @Groups("post:read")
 */
private $id;

/**
 * @ORM\Column(type="string", length=50, nullable=true)
 * @Groups("post:read")
 */
private $nsc;

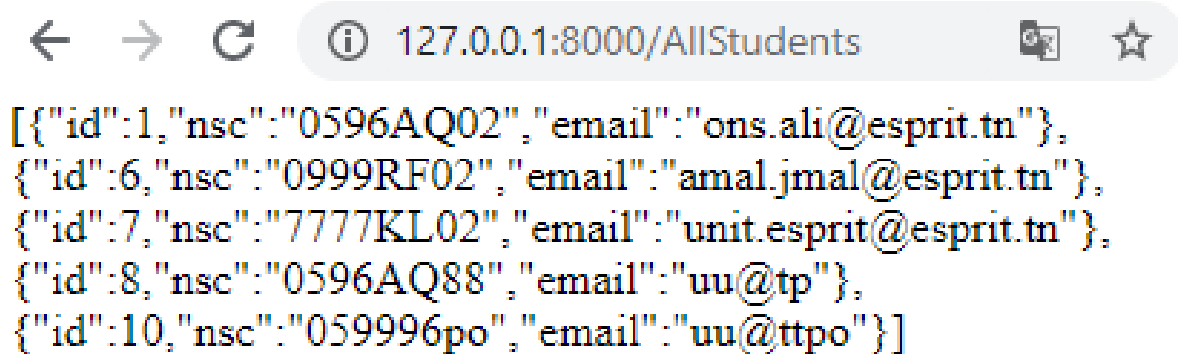
/**
 * @ORM\Column(type="string", length=255, nullable=true)
 * @Groups("post:read")
 */
private $email;

/**
 * @ORM\ManyToOne(targetEntity=Classroom::class, inversedBy="students")
 * @ORM\JoinColumn(nullable=false)
 */
private $Classroom;
```

- N'oubliez pas d'ajouter les uses des classes utilisées dans l'entité Student

```
use Symfony\Component\Serializer\Annotation\Groups;
```

En tapant le lien suivant <http://127.0.0.1:8000/AllStudents>, une liste des étudiants sera affichée



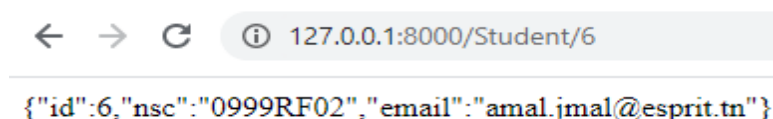
## 1. Récupération d'un étudiant selon l'id :

Le code ci-dessous permet de récupérer un seul étudiant

```
public function StudentId(Request $request,$id, NormalizerInterface $Normalizer )
{
    $em = $this->getDoctrine()->getManager();
    $student = $em->getRepository(Student::class)->find($id);
    $jsonContent = $Normalizer->normalize($student, 'json', ['groups'=>'post:read']);

    return new Response(json_encode($jsonContent));
}
```

- Ci-dessous le résultat de la fonction « StudentId » :



## 2. Ajout d'un étudiant :

Le code ci-dessous permet d'ajouter un étudiant

```

/**
 * @Route("/addStudentJSON/new", name="addStudentJSON")
 */
public function addStudentJSON(Request $request, NormalizerInterface $Normalizer)
{
    $em = $this->getDoctrine()->getManager();
    $student = new Student();
    $student->setNsc($request->get('nsc'));
    $student->SetEmail($request->get('email'));
    $em->persist($student);
    $em->flush();
    $jsonContent = $Normalizer->normalize($student, 'json', ['groups'=>'post:read']);
    return new Response(json_encode($jsonContent));
}

```

- Pour le test, appelez l'url suivant :  
«http://127.0.0.1:8000/addStudentJSON/new?nsc=655999&email=student.name@esprit.tn»

← → ↻ ⓘ 127.0.0.1:8000/addStudentJSON/new?nsc=655999&email=student.name@esprit.tn

```
{"id":24,"nsc":"655999","email":"student.name@esprit.tn"}
```

### 3. Modification d'un étudiant :

Le code ci-dessous permet d'ajouter un étudiant

```

/**
 * @Route("/updateStudentJSON/{id}", name="updateStudentJSON")
 */
public function updateStudentJSON(Request $request, NormalizerInterface $Normalizer, $id)
{
    $em = $this->getDoctrine()->getManager();
    $student = $em->getRepository(Student::class)->find($id);
    $student->setNsc($request->get('nsc'));
    $student->SetEmail($request->get('email'));
    $em->flush();
    $jsonContent = $Normalizer->normalize($student, 'json', ['groups'=>'post:read']);
    return new Response("Information updated successfully".json_encode($jsonContent));
}

```

### 4. Suppression d'un étudiant :

Le code ci-dessous permet de supprimer un étudiant

```

/**
 * @Route("/deleteStudentJSON/{id}", name="deleteStudentJSON")
 */
public function deleteStudentJSON(Request $request, NormalizerInterface $Normalizer, $id)
{
    $em = $this->getDoctrine()->getManager();
    $student = $em->getRepository(Student::class)->find($id);
    $em->remove($student);
    $em->flush();
    $jsonContent = $Normalizer->normalize($student, 'json', ['groups'=>'post:read']);
    return new Response("Student deleted successfully".json_encode($jsonContent));
}

```

- Pour le test, appelez l'url suivante :  
«http://127.0.0.1:8000/deleteStudentJSON/4»

