SERIALIZER COMPONENT

...

“The Serializer component is meant to be used to turn objects into a specific format (XML, JSON, YAML, ...) and the other way around.”

Historique

...

Composant datant de 2011 et ré-écrit pour le projet API Platform (Kévin Dunglas)

Projets qui utilise ce composant

...

Symfony
Drupal
API Platform
Thelia

...

Utilisation

...

Transformer des données pour des API web, de l'échange de données entre applications, hydrater des objets à partir de base de données ou d'API externe
⇒ Bien contrôler ses données en entrée et sortie!

https://symfony.com/doc/current/components/serializer.html

Fonctions PHP ?

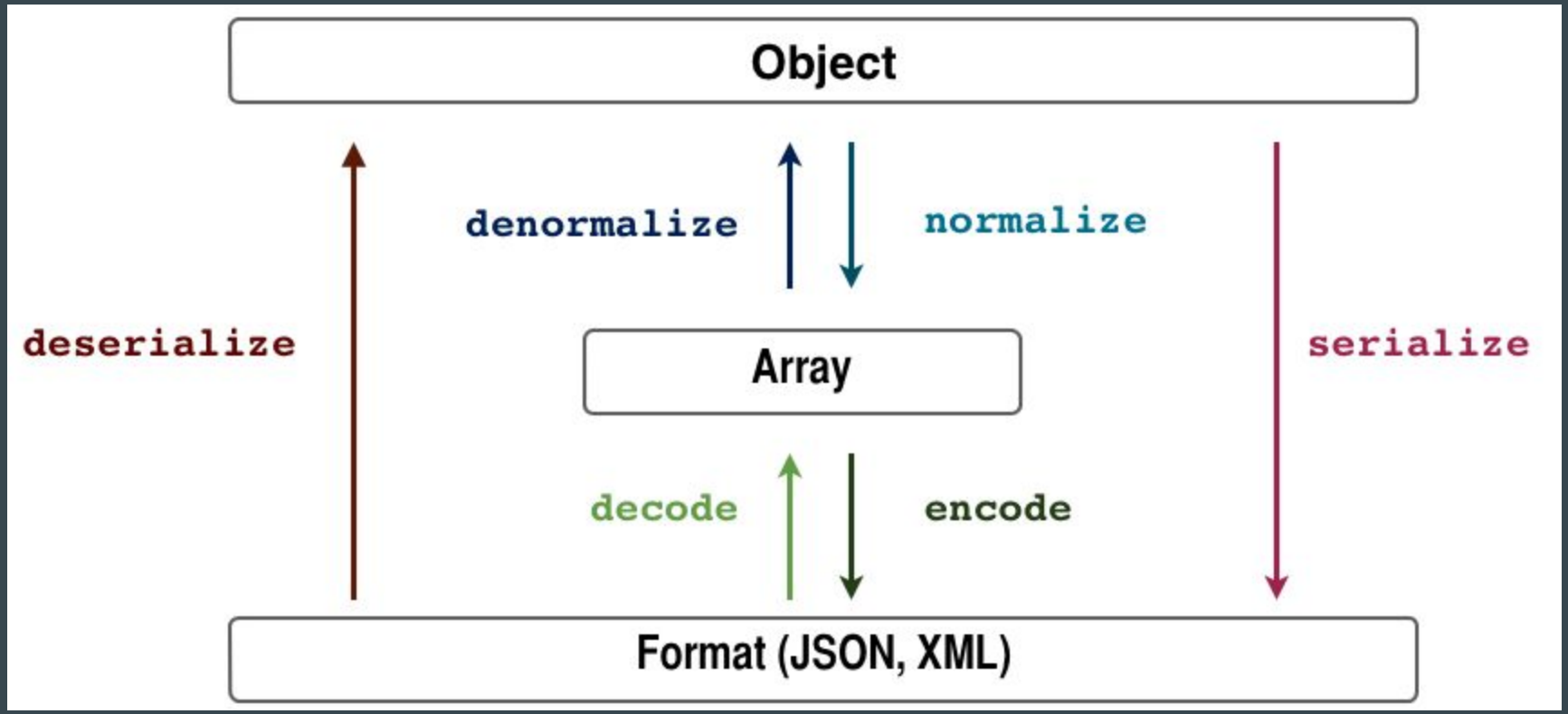
...

serialize() : 'O:16:"App\Model\Meetup":1:{s:4:"name";s:15:"PHP Meetup 2018";}'

unserialize() : object(App\Model\Meetup)[77]
public 'name' => string 'PHP Meetup 2018' (length=15)

json_encode() : '{"name":"PHP Meetup 2018"}'

json_decode() : object(stdClass)[77]
public 'name' => string 'PHP Meetup 2018' (length=15)


Object
denormalize
normalize
deserialize
serialize
Array
decode
encode
Format (JSON, XML)

1 . SERIALIZATION = NORMALIZATION + ENCODING

1 . Un normalizer (objet)

Le normalizer ⇒ Passage objet à un tableau associatif (ne récupère que les clefs publiques).

2 . Un encoder (objet)

L'encoder ⇒ D'un tableau associatif à une string.

On peut passer plusieurs normalizers, le premier normalizer prend la priorité.

On peut implémenter un normalizer custom.

Exemple

```php
/**
 * Serialize in json.
 */
public function serialize()
{
    $serializer = new Serializer( // Le Serialiser
        [new PropertyNormalizer()], // On passe un tableau de normalizer(s).
        [new JsonEncoder()] // On passe un tableau d'encoder(s).
    );

    $meetup = new Meetup();
    $meetup->name = 'PHP Meetup 2018';

    var_dump($serializer->serialize($meetup, format: 'json'));
    die();
    // '{"name":"PHP Meetup 2018"}'
}
```

2 . DESERIALIZATION = DECODING + DENORMALIZATION

L’inverse de la sérialisation.

Exemple

```php
/**
 * Deserialize in json.
 */
public function deserialize()
{
    $serializer = new Serializer( // Le Serialiser
        [new PropertyNormalizer()], // On passe un tableau de normalizer(s).
        [new JsonEncoder()] // On passe un tableau d'encoder(s).
    );

    var_dump($serializer->deserialize(
         data: '{"name":"PHP Meetup 2018"}',
         type: Meetup::class,
         format: 'json')
    );
    die();

    // object(App\Model\Meetup)[36]
    // public 'name' => string 'PHP Meetup 2018' (length=15)
}
```

Autre exemple

```php
/**
 * Serialize full in json.
 */
public function serializeFull()
{
    $serializer = new Serializer( // Le Serialiser
        [
            new DateTimeNormalizer(),
            new DataUriNormalizer(),
            new ObjectNormalizer(),
        ], // On passe un tableau de normalizer(s).
        [new JsonEncoder()] // On passe un tableau d'encoder(s).
    );

    $meetup = new MeetupFull();
    $meetup->id = 1;
    $meetup->name = 'PHP Meetup 2018';
    $meetup->date = new \DateTimeImmutable( time: '2018/07/11');
    $meetup->logo = new \SplFileInfo( file_name: 'php-meetup.jpg');

    var_dump($serializer->serialize($meetup,  format: 'json'));
    die();
    // '{"id":1,"name":"PHP Meetup 2018","date":"2018-07-11T00:00:00+02:00","logo":"data:image\/jpeg;base64,\/9j\/
}
```

Modifier un objet

```php
/**
 * Serialize full in json.
 */
public function update()
{
    $serializer = new Serializer( // Le Serialiser
        [
            new ObjectNormalizer(),
        ], // On passe un tableau de normalizer(s).
        [new JsonEncoder()] // On passe un tableau d'encoder(s).
    );

    $meetup = new MeetupFull();
    $meetup->id = 1;
    $meetup->name = 'PHP Meetup 2018';
    $meetup->date = new \DateTimeImmutable( time: '2018/07/11');

    var_dump($serializer->deserialize(
         data: '{"id":1,"name":"PHP THE BEST"}',
         type: MeetupFull::class,
         format: 'json',
        ['object_to_populate' => $meetup]));
    die();
    //object(App\Model\MeetupFull)[44]
    //public 'id' => int 1
    //public 'name' => string 'PHP THE BEST' (length=12)
    //public 'date' =>
    //object(DateTimeImmutable)[82]
    //  public 'date' => string '2018-07-11 00:00:00.000000' (length=26)
    //  public 'timezone_type' => int 3
    //  public 'timezone' => string 'Europe/Zurich' (length=13)
    //public 'logo' => null
}
```

Plusieurs NORMALIZERS

1 . PropertyNormalizer ⇒ Accède uniquement aux propriétés publiques

2 . GetSetMethodNormalizer => + get() / set() et is() / has()

3 . ObjectNormalizer => (utiliser par Symfony) accède à l'ensemble des propriétés de l'objet.

Et des spécifiques : DataUriNormalizer, DateIntervalNormalizer, ...

Pour aller plus loin...

On peut ajouter des attributs (phase de normalization) que l'on souhaite depuis le front et ainsi récupérer les datas que l'on souhaite.

Le Serializer lors de la dénormalization contrôle les types!!!

E.g : Si on pass un string à la place d'un int, le normalizeur va lever une exception.

Autre exemple aussi intéressant, on peut l'utiliser pour implémenter un arbre d'un menu par exemple.

Mon dernier test

```php
/**
 * Export CSV.
 */
public function exportCSV()
{
    $serializer = new Serializer( // Le Serialiser
        [
            new ObjectNormalizer(),
        ], // On passe un tableau de normalizer(s).
        [new CsvEncoder()] // On passe un tableau d'encoder(s).
    );

    $data = [
        'foo' => 'aaa',
        'bar' => [
            ['id' => 111, 1 => 'bbb'],
            ['lorem' => 'ipsum'],
        ]
    ];

    $csv = $serializer->encode($data, format: 'csv');

    $response = new Response($csv);
    $response->headers->set( key: 'Content-Type', values: 'text/csv; charset=utf-8; application/force-download');
    $response->headers->set( key: 'Content-Disposition', values: 'attachment; filename="data.csv"');

    return $response;
}
```

C’est fini!

Si vous voulez en savoir plus je vous invite à l’essayer et l’utiliser..