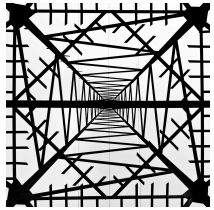
Les Formulaire avec Symfony

Première approche

Meetup Symfony Montpellier

mardi 11 octobre 2016 - Kaliop



Qui suis-je?

Julien Vinber

- Développeur Delphi / PHP depuis 15 ans
- Symfony depuis 2 - 3 ans
- Travail chez l'éditeur de logiciel Yooda depuis 8 ans.

YOODA
LIVE DATA SOLUTIONS

1.

L’art de rendre compliquer ce qui était simple.

Mon premier formulaire basique

Sans Symfony

- 1 fichier
- 25 lignes

Avec Symfony

- 4 fichiers
- 247 lignes

Exemple sans Symfony

```php
<?php

if (isset($_GET['valider'])){
    $sql = '
    INSERT INTO membre (nom, prenom, mail, telephone, date_naissance) VALUES (
        \''. $_GET['nom']           .'\',
        \''. $_GET['prenom']        .'\',
        \''. $_GET['mail']          .'\',
        \''. $_GET['telephone']     .'\',
        \''. $_GET['dateNaissance'] .'\')
    ';
    $mysqli = new mysqli("localhost", "root", "root", "formation");
}

?>
<h2>Ajouter un membre</ h2>
<form>
    <label for="nom">Nom :</ label><input id="nom" name="nom" type="text" /><br />
    <label for="prenom">Prénom :</ label><input id="prenom" name="prenom" type="text" /><br />
    <label for="mail">Mail :</ label><input id="mail" name="mail" type="email" /><br />
    <label for="telephone">Téléphone :</ label><input id="telephone" name="telephone" type="tel" /><br />
    <label for="dateNaissance">Naissance :</ label><input id="dateNaissance" name="dateNaissance" type="date" /><br />
    <button name="valider">Ajouter</ button>
</ form>
```

Extrait avec Symfony

```
…
…

public function addAction(Request $request)
{
    $membre = new Membre();

    $form = $this->createForm(MembreType::class, $membre);

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($membre);
        $em->flush();

        return $this->redirect($this->generateUrl('membre_add'));
    }

    return array(
        'form' => $form->createView()
    );
}

…
…
```

=> C’est plus complexe

2.

Comprendre le code

Un formulaire et la pour renseigner un objet

```php
/**
* @ORM\Entity
* @ORM\Table()
*/
class Membre
{
    /**
     * @var integer
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    protected $nom;


    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    protected $prenom;
```

```php
    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    protected $mail;

    /**
     * @var string
     * @ORM\Column(type="string", length=25)
     */
    protected $telephone;


    /**
     * @var \DateTime
     * @ORM\Column(type="date")
     */
    protected $dateNaissance;

…
…
…
```

On décrit un formulaire dans un objet

```php
class MembreType extends AbstractType
{

    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('nom')
            ->add('prenom')
            ->add('mail')
            ->add('telephone')
            ->add('dateNaissance', BirthdayType::class)
            ->add('cree', SubmitType::class, array('label' => 'Valider'))
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' => 'AppBundle\Entity\Membre'
        ));
    }
}
```

Une pincé html

```
{% extends 'base.html.twig' %}

{% block body %}
   <h2>Ajouter un membre</ h2>
   {{ form(form) }}
{% endblock %}
```

Enfin le coeur de notre programme, le contrôleur.

```php
class MembreController extends Controller
{
    /**
     * @Route("/membre/add", name="membre_add")
     * @Template()
     */
    public function addAction(Request $request)
    {
        $membre = new Membre();

        $form = $this->createForm(MembreType::class, $membre);

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $em = $this->getDoctrine()->getManager();
            $em->persist($membre);
            $em->flush();

            return $this->redirect($this->generateUrl('membre_add'));
        }

        return array(
            'form' => $form->createView()
        );
    }
}
```

3.

La force des formulaires Symfony

Gestion des scripte de modification

```php
/**
* @Route("/membre/{idMembre}", name="membre_add", requirements={"idMembre":"add|\d+"})
* @Template()
*/
public function addAction(Request $request, $idMembre)
{
    if ($idMembre == 'add'){
        $membre = new Membre();
    } else {
        $membre = $this->getDoctrine()->getRepository("AppBundle:Membre")->find($idMembre);
    }

    $form = $this->createForm(MembreType::class, $membre);

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        if ($idMembre == 'add') {
            $em->persist($membre);
        }
        $em->flush();

        return $this->redirect($this->generateUrl('membre_add', array("idMembre" => $membre->getId())));
    }

    return array(
        'form' => $form->createView()
    );
}
```

Gestion de la validation coter serveur

```php
/**
* @var string
* @ORM\Column(type="string", length=255)
* @Assert\NotBlank()
*/
protected $nom;

…

/**
* @var string
* @ORM\Column(type="string", length=255)
* @Assert\NotBlank()
* @Assert\Email(
*     message = "The email '{{ value }}' is not a valid email.",
*     checkMX = true
* )
*/
protected $mail;
```

Validation personaliser

```php
/**
* @Annotation
*/
class IsTelephone extends Constraint
{
    public $message = 'Cela ne ressemble pas à un numéro de téléphone.';
}
…
class IsTelephoneValidator extends ConstraintValidator
{
    public function validate($value, Constraint $constraint)
    {
        if (!preg_match('/^0[0-9]{9}$/', $value, $matches)) {
            $this->context->buildViolation($constraint->message)
                ->setParameter('%string%', $value)
                ->addViolation();
        }
    }
}
```

```php
/**
* @var string
* @ORM\Column(type="string", length=25)
* @Assert\NotBlank()
* @AppAssert\IsTelephone
*/
protected $telephone;
```

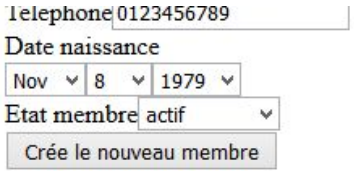
Champs complexe : choix dans une base

```php
class EtatMembre
{
    /**
     * @var integer
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     * @ORM\Column(type="string", length=255)
     */
    protected $nom;
…


/**
* @ORM\ManyToOne(targetEntity="EtatMembre",
inversedBy="membre")
* @ORM\JoinColumn(referencedColumnName="id")
*/
private $etatMembre;
```

```php
    $builder
        ->add('nom')
        ->add('prenom')
        ->add('mail')
        ->add('telephone')
        ->add('dateNaissance', BirthdayType::class)
        ->add('etatMembre')
        ->add('cree', SubmitType::class,
array('label' => 'Crée le nouveau membre'))
    ;
```



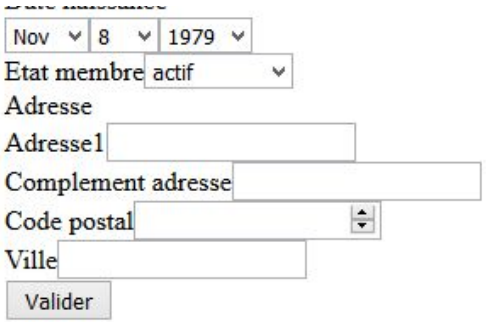
Champs complexes : sous formulaire, OneToOne

```php
/**
* @ORM\Entity
* @ORM\Table()
*/
class Adresse
{
...

class AdresseType extends AbstractType
{
    public function
buildForm(FormBuilderInterface $builder,
array $options)
    {
        $builder
            ->add('adresse1')
            ->add('complementAdresse')
            ->add('codePostal')
            ->add('ville')
        ;
    }

    public function
configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults(array(
            'data_class' =>
'AppBundle\Entity\Adresse'
        ));
    }
}
```

```php
    $builder
        ->add('nom')
        ->add('prenom')
        ->add('mail')
        ->add('telephone')
        ->add('dateNaissance', BirthdayType::class)
        ->add('etatMembre')
        ->add('adresse', AdresseType::class)
        ->add('cree', SubmitType::class, array('label'
=> 'Crée le nouveau membre'))
    ;
```

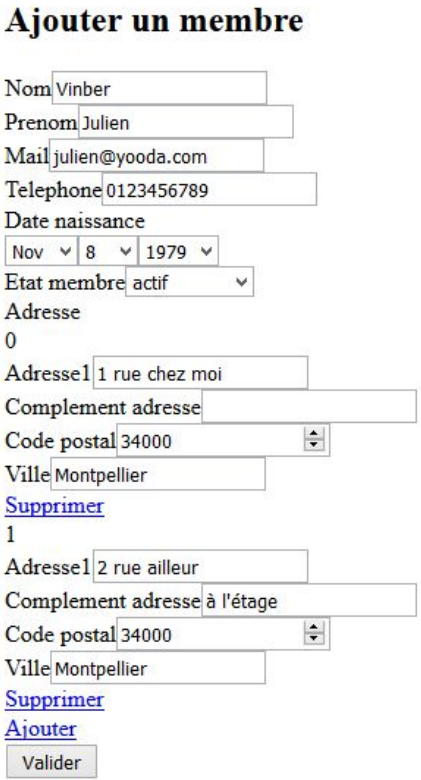


Champs complexes : sous formulaire multiple, OneToMany

Trop de code pour le montrer, mais cela permet d'avoir un résultat complexe relativement facilement. Pour cela il faut :

- Configurer nos entité normalement
- Crée le formulaire de notre entité
- Ajouter du code JS pour être capable d'ajouter ou supprimer des ligne
- Si le formulaire permet d'éditer un objet existant, alors il faut ajouter du code pour détecter et supprimer les sous objet supprimer.

Champs complexes : sous formulaire multiple, OneToMany, exemple



Est plus encore

- Protection CSRF (Cross-site request forgery)
- Champs personaliser
- Personalisation :
  - D’un champ
  - D’un formulaire
  - De tous les formulaires
- …

MERCI.

Des questions?

Documentation officiel :

http://symfony.com/doc/current/forms.html

http://symfony.com/doc/current/validation.html

Source de la présentation :

https://github.com/julienVinber/meetup20161011

Mail : julien@vinber.fr