SIN6U5 : Développement web 2 (Prog. web coté serveur) La composante `Form` de Symfony

Luigi Santocanale
LIF, Aix-Marseille Université

22 mars 2018

Plan

Petit projet : un formulaire de contacte

Ajout au formulaire du téléversement de fichier

Validations coté client et/ou serveur

Ajout au formulaire d'un captcha

Documentation, principes

- Doc Symfony
- API de Symfony (3.4)
- API des formulaires de Symfony (3.4)

Principes :

- Un formulaire est un formulaire pour d'une entité . . .
- . . . ses champs existent pour recueillir les données d'une entité.
- Les processus
  - de construction des balises HTML « input »,
  - leur mise en forme,
  - leur validation

  sont automatisés.

Plan

Objectifs

- Envoyer un courriel au développeur d'un appli web.

Besoins

Besoins :

- 3 routes :
  1.
     - présentation du formulaire (avec vue) ;
     - validation du formulaire si soumis ;
     - traitement des données utilisateur (envoie du courriel) ;
     - redirection si/vers succès/échec.
  2. Page de confirmation si succès (avec vue) ;
  3. Page d'erreur si échec de l'envoie (avec vue).
- 3 vues.

Objectifs didactiques :

- Savoir utiliser les formulaires avec Symfony.
- Savoir envoyer des courriels (avec SwiftMailer).

Fichiers

```
.
|-- Controller
|   `--
 ↪   MessageController.php
|-- Entity
|   `-- Message.php
|-- Makefile
|-- tree.txt
`-- views
    |-- failure.html.twig
    |-- message.html.twig
    `-- success.html.twig

3 directories, 7 files
```

Controller/* $\mapsto$ src/AppBundle/Controller/

Entity/* $\mapsto$ src/AppBundle/Entity/

views/* $\mapsto$ app/Resources/views/

L'entité Message

```php
<?php

// src/AppBundle/Entity/Message.php

namespace AppBundle\Entity;

class Message {

    private $sender;
    private $from;
    private $text;
    private $subject;

    function getSender() {
        return $this->sender;
    }


    function setSender($sender) {
        $this->sender = $sender;
        return $this;
    }

    ...

}
```

Remarques

- Cette entité ne sera pas utilisée avec (persistée dans) une BD via l'ORM.
- On peut aussi créer des formulaires rattachés aux entités avec persistance.
  Exemple : les entités des CRUDs étudiés pendant le dernier cours.

Exercice :

- On stokera chaque message avec les cordonnées de l'expéditeur (nom, courriel, ip) dans une BD. Idée : si l'expéditeur est méchant, il sera banni du site (via le numéro d'IP).
- Pour ce faire, ajoutez à l'entité `Message` une propriété `ip` et des annotations à l'ORM.

Le contrôleur

```php
<?php

// src/AppBundle/Controller/MessageController.php

namespace AppBundle\Controller;

/**
 * @Route("message")
 */
class MessageController extends Controller {

    /**
     * @Route("/",name="message")
     */
    public function messageAction(Request $request) {

    }

    /**
     * @Route("/success",name="message_success")
     */
    public function success() {
        return $this->render('message/success.html.twig');
    }


    /**
     * @Route("/failure",name="message_failure")
     */
    public function fail() {
        return $this->render('message/success.html.twig');
    }

}
```

Remarques

Le suivant est un « `DocComment` » :

```
/**
* @Route("/",name="message")
*/
```

Il commence par `/**` et se temine par `*/`.

L'annotation suivante ne marchera pas :

```
/*
* @Route("/",name="message")
*/
```

Création du formulaire dans le contrôleur

```
10  use Symfony\Component\HttpFoundation\Request;
11  use Symfony\Component\Form\Extension\Core\Type\TextType;
12  use Symfony\Component\Form\Extension\Core\Type\TextareaType;
13  use Symfony\Component\Form\Extension\Core\Type\EmailType;
14  use Symfony\Component\Form\Extension\Core\Type\SubmitType;



21      /**
22       * @Route("/",name="message")
23       */
24      public function messageAction(Request $request) {
25          $message = new Message();
26          $message->setSender('Votre nom')
27                  ->setFrom('Votre courriel')
28                  ->setText('Votre message')
29                  ->setSubject('Objet');
30
31          $form = $this->createFormBuilder($message)
32                  ->add('sender', TextType::class, ['label' => 'Votre nom'])
33                  ->add('from', EmailType::class, ['label' => 'Votre courriel'])
34                  ->add('subject', TextType::class, ['label' => 'Objet'])
35                  ->add('text', TextareaType::class, ['label' => 'Votre message'])
36                  ->add('envoyer', SubmitType::class, ['label' => 'Envoyer'])
37                  ->getForm();
38
39          $form->handleRequest($request);

           ...
```

API

- `$controller->createFormBuilder($entity)`
- `$builder->add(string $field,string $inputType, array $options)`
- `FormInterface $builder->getForm()`
- `$form->handle(Request $request)`

- FormBuilderInterface
- FormInterface

Validation du formulaire

```
41          if ($form->isSubmitted() && $form->isValid()) {
42              $message = $form->getData();
43              // Traitement
44              $ret = $this->send($message);
45
46              if ($ret) {
47                  return $this->redirectToRoute('message_success');
48              } else {
49                  return $this->redirectToRoute('message_failure');
50              }
51          }
52          return $this->render('message/message.html.twig', array(
53                      'form' => $form->createView(),
54          ));
55      }
```

API

- `$form->isSubmitted()`
- `$form->isValid()`
- `$form->getData()`

- `FormInterface`

Traitement : utilisation de SwiftMailer

```
43          // Traitement
44          $ret = $this->send($message);


71      private function send(Message $message) {
72          $mailer = $this->container->get('swiftmailer.mailer.default');
73          $mail = new \Swift_Message($message->getSubject());
74          $mail->setBody(sprintf(
75                          "Message de la part de %s.\nTexte du message :\n%s", //
76                          $message->getSender(), //
77                          $message->getText() //
78          ));
79          $mail->setFrom([$message->getFrom() => $message->getSender()]);
80          $mail->setTo('luigi.santocanale@lis-lab.fr');
81          return $mailer->send($mail);
82      }
```

API

- `$mail->setFrom(...)`
- `$mail->sertTo(...)`
- `$mail->serBody(...)`
- `$mailer->send($mail)`

- Swift_Message
- Swift_Mailer

Paramétrage de SwiftMailer

Fichier `app/config/config.yml` :

```
65  # Swiftmailer Configuration
66  swiftmailer:
67      transport: gmail
68  #     host: '%mailer_host%'
69      username: '%mailer_user%'
70      password: '%mailer_password%'
```

Aperçu des vues

```
{# app/Resources/views/message/success.html.twig #}

{% extends "base.html.twig" %}

{% block body %}
    <h1>Message envoyé</h1>
{% endblock %}
```

```
{# app/Resources/views/message/success.html.twig #}

{% extends "base.html.twig" %}

{% block body %}
    <h1>Échec de l'envoie du message</h1>
{% endblock %}
```

Fichier app/Resources/views/base.html.twig

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}Welcome!{% endblock %}</title>
        {% block stylesheets %}{% endblock %}
        <link rel="icon" type="image/x-icon" href="{{ asset('favicon.ico') }}" />
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}{% endblock %}
    </body>
</html>
```

Mise en forme du formulaire

```twig
{# app/Resources/views/message/message.html.twig #}
{% extends "base.html.twig" %}

{# {% form_theme form 'form_table_layout.html.twig' %} #}
{% form_theme form 'bootstrap_4_layout.html.twig' %}

{% block stylesheets %}
    <link rel="stylesheet" href="{{ asset('css/bootstrap.css') }}" />
{% endblock %}

{% block body %}
    <div class="container">
        <h1>Envoyer un message au développeur</h1>
        {{ form_start(form) }}
        {{ form_widget(form) }}
        {{ form_end(form) }}
    </div>
{% endblock %}
```

Plan

Objectifs

Améliorations :

- Simplifier le code du contrôleur.
- Factoriser la définition du formulaire.
- Rendre le formulaire réutilisable.

Ajouts :

- Fichier en pièce jointe.
- Ajout d'un captcha.

Ajout d'un champ `file` à l'entité

```php
<?php

// src/AppBundle/Entity/Message.php

namespace AppBundle\Entity;

use Symfony\Component\Validator\Constraints as Assert;

class Message {

    protected $sender;
    protected $from;
    protected $text;
    protected $subject;
    protected $file = null;

    function getFile() {
        return $this->file;
    }

    function setFile($file) {
        $this->file = $file;
        return $this;
    }

       ...

}
```

Simplification du contrôleur

```php
<?php

// src/AppBundle/Controller/MessageController.php

namespace AppBundle\Controller;

use AppBundle\Entity\Message;
use AppBundle\Form\MessageType;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Component\HttpFoundation\Request;
 ...

    /**
     * @Route("/",name="message")
     */
    public function messageAction(Request $request) {
        $message = new Message();

        $form = $this->createForm(MessageType::class, $message);

        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
         ...

        }
        return $this->render('message/message.html.twig', array(
                    'form' => $form->createView(),
        ));
    }
```

Le formulaire comme classe : `MessageType`

Classes utilisés :

```php
<?php

// src/AppBundle/Form/MessageType.php

namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\FileType;
use Gregwar\CaptchaBundle\Type\CaptchaType;
use AppBundle\Entity\Message;

  ...
```

Ajout des widgets

```
  ...
class MessageType extends AbstractType {

    public function buildForm(FormBuilderInterface $builder, array $options) {
        $builder->add('sender', TextType::class, ['label' => 'Votre nom'])
                ->add('from', EmailType::class, ['label' => 'Votre courriel'])
                ->add('subject', TextType::class, ['label' => 'Objet'])
                ->add('text', TextareaType::class, [//
                    'label' => 'Votre message',
                    'attr' => ['rows' => 10]])
                ->add('file', FileType::class, [//
                    'label' => 'Pièce jointe',
                    'required' => false
        ]);


        /* Add submit */
        $builder->add('envoyer', SubmitType::class, ['label' => 'Envoyer']);
    }

    ...

}
```

Liaison entre le formulaire et l'entité

```
18  class MessageType extends AbstractType {
        ...
40
41      public function configureOptions(OptionsResolver $resolver) {
42          $resolver->setDefaults(array(
43              'data_class' => Message::class,
44          ));
45      }
46
47  }
```

Traitement avec SwiftMailer

```php
<?php


class MessageController extends Controller {

        ...

    private function send(Message $message) {
        $mailer = $this->container->get('swiftmailer.mailer.default');
        $mail = new \Swift_Message($message->getSubject());
        $mail->setBody(sprintf(
                        "Message de la part de %s.\nTexte du message :\n%s", //
                        $message->getSender(), //
                        $message->getText() //
        ));
        $mail->setFrom([$message->getFrom() => $message->getSender()]);
        $mail->setTo('luigi.santocanale@lis-lab.fr');

        /** @var Symfony\Component\HttpFoundation\File\UploadedFile $file */
        $file = $message->getFile();
        if ($file && $file->isValid()) {
            $mail->attach(

 ↪    \Swift_Attachment::fromPath($file->getPath())->setFilename('Attachment'));
        }

        return $mailer->send($mail);
    }

}
```

Plan

Ajout de contraintes de validation

```php
<?php
// src/AppBundle/Entity/Message.php

namespace AppBundle\Entity;

use Symfony\Component\Validator\Constraints as Assert;

class Message {

    /**
     * @Assert\Length(
     *      min = 5,
     *      max = 10,
     *      minMessage = "Au moins {{ limit }} caracteres",
     *      maxMessage = "Au moins {{ limit }} caracteres"
     * )
     */
    protected $sender;

    /**
     * @Assert\Email(
     *      message = "Le courriel '{{ value }}' n'est pas valide.",
     * )
     */
    protected $from;

    ...

}
```

Le formulaire sans validation client

```
1  {# app/Resources/views/message/message.html.twig #}
2  {% extends "base.html.twig" %}
3
4  {% form_theme form 'form_table_layout.html.twig' %}
5  {#{% form_theme form 'bootstrap_4_layout.html.twig' %} #}
6
7  {% block title %}Envoyer un message{% endblock %}
8
9  {% block body %}
10
11         <h1>Envoyer un message au développeur</h1>
12
13         {{ form_start(form,{'attr': {'novalidate':'novalidate'}} ) }}
14         {{ form_widget(form) }}
15         {{ form_end(form) }}
16
17 {% endblock %}
```

- Utile pour tester les validations coté serveur.

Le service validator

- On valide une entité, pas un formulaire.
- Par exemple : on peut valider une entité sans qu'elle aie été produite par un POST.
- Des validations possisbles :
  - NotBlank, Blank, NotNull, IsNull, IsTrue, IsFalse, Type,
  - Email, Length, Url, Regex, Ip,
  - Range
  - Date, DateTime, Time,
  - etc.
- Liste complète

Voir la doc.

Plan

Installation du bundle `gregwar/captcha-bundle`

- Le bundle sur packagist.

```
composer require gregwar/captcha-bundle
```

Enregistrement du bundle

Fichier `app/appKernel.php` :

```
1   <?php
7   class AppKernel extends Kernel
8   {
9       public function registerBundles()
10      {
11          $bundles = [
                ...
20              new Gregwar\CaptchaBundle\GregwarCaptchaBundle(),
21          ];
            ...
35      }
        ...
62  }
```

Paramétrage du bundle

Fichier `app/config/config.yml` :

```
gregwar_captcha:
    width: 150
    height: 50
    length: 6
```

Modification du formulaire

```
18  class MessageType extends AbstractType {
19
20      public function buildForm(FormBuilderInterface $builder, array $options) {
21          $builder->add('sender', TextType::class, ['label' => 'Votre nom'])
22                  ->add('from', EmailType::class, ['label' => 'Votre courriel'])
23                  ->add('subject', TextType::class, ['label' => 'Objet'])
24                  ->add('text', TextareaType::class, [//
25                      'label' => 'Votre message',
26                      'attr' => ['rows' => 10]])
27                  ->add('file', FileType::class, [//
28                      'label' => 'Pièce jointe',
29                      'required' => false
30          ]);
31
32          /* Add Captcha */
33          $builder->add('captcha', CaptchaType::class,
34                  ['label' => 'Recopiez ce nombre']
35                  );
36
37          /* Add submit */
38          $builder->add('envoyer', SubmitType::class, ['label' => 'Envoyer']);
39      }

        ...

47  }
```