esprit Ecole Supérieure Privée d'Ingénierie et de Technologies	Année Universitaire : 2015-2016 **Workshops Framework Symfony2**

Workshop n°5
SwiftmailerBundle: configuration et utilisation

Objectif

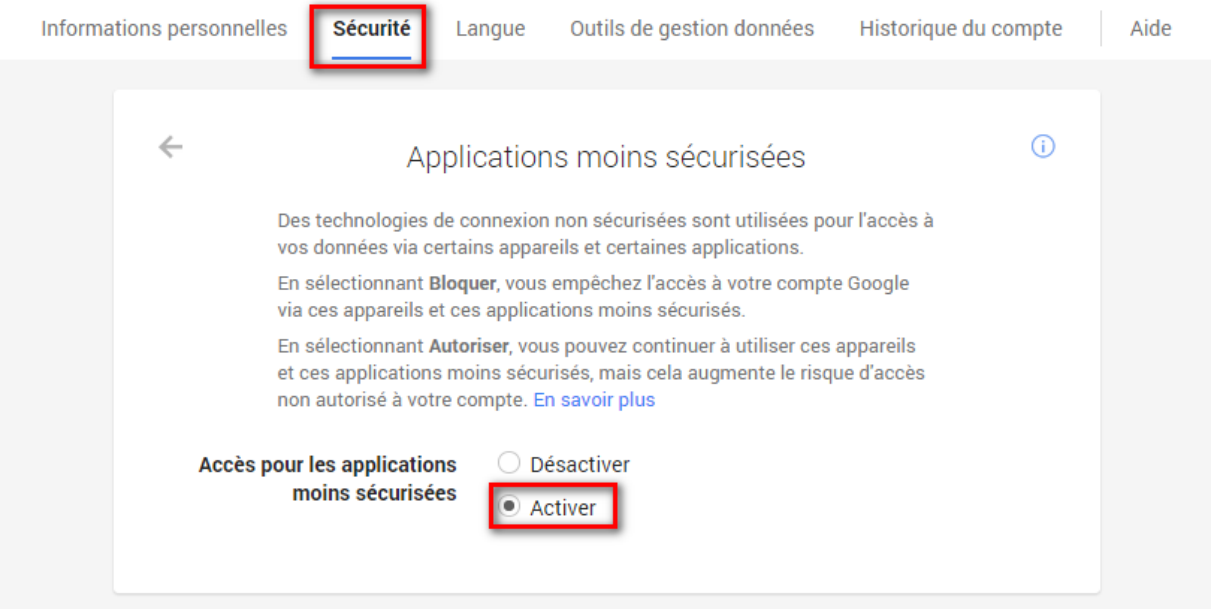
Le but de ce workshop est de configurer et d'utiliser le bundle Swiftmailer pour envoyer des e-mails

Création d'une boite mail

Avant de commencer notre Workshop, vous devez créer une boite gmail puis cliquer sur ce lien :

https://www.google.com/settings/security/lesssecureapps

Puis Configurer votre adresse mail afin d'être accéder par votre application et ceci en selctionnant « activer »



Configuration de SwiftmailerBundle

Tout d'abord nous allons créer un nouveau projet intitulé WorkshopSwiftMailerBundel puis nous créons un nouveau bundle intitulé MyAppMailBundle. Par la suite nous générons le contrôleur de ce bundle qui est MailControler.

Ps : N'oubliez pas d'activer le bundle dans votre Kernel avant de l'utiliser

```php
class AppKernel extends Kernel
{
    public function registerBundles()
    {
        $bundles = array(
            new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
            new Symfony\Bundle\TwigBundle\TwigBundle(),
            new Symfony\Bundle\MonologBundle\MonologBundle(),
            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
            new Symfony\Bundle\AsseticBundle\AsseticBundle(),
            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
            new Nomaya\SocialBundle\NomayaSocialBundle(),
            new MyApp\MailBundle\MyAppMailBundle(),
        );
```

Après la phase de création du bundle, nous allons maintenant le configurer afin d'utiliser le SwiftMailerBundle. Allons-nous vers : # app/config/config.yml puis, Copiez la configuration nécessaire :

```yaml
swiftmailer:
    transport:  smtp
    encryption: ssl
    auth_mode:  login
    port:       465
    host:       smtp.gmail.com
    username:   mail_crée_précedement
    password:   your_password
    delivery_address: mail_de_réception
    disable_delivery: false
```

PS : le host change selon votre serveur mail.

Création d'un formulaire de contact

Nous allons créer un formulaire de contact par lequel un visiteur peut nous contacter. Le résultat de validation du formulaire est un mail transmit vers l'administrateur du site.

Pour la phase de création du formulaire, nous irons créer :

- Une entité « Mail » dans le dossier Entity.
- Une classe « MailType » dans le dossier Form.
- Une vue « new.html.twig » dans le dossier ressources/views de notre bundle : cette vue permet l'affichage de formulaire.
- Une vue « mail.html.twig » » dans le dossier ressources/views de notre bundle : cette vue permet l'affichage d'un message de succès d'envoi.

1- Création de l'entité Mail

MyApp/MailBundle/Entity/Mail.php

```
namespace MyApp\MailBundle\Entity;
class Mail {
   private $nom;
    private $prenom;
    private $tel;
    private $from;
    private $text;
}
```

NB : N'oubliez pas de générer les getters et les setters.

2- Création du classe MailType

MyApp/MailBundle/Form/MailType.php

```
namespace MyApp\MailBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolverInterface;

class MailType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('nom', 'text')
            ->add('prenom', 'text')
            ->add('tel', 'integer')
            ->add('from', 'email')
            ->add('text', 'textarea')
         ->add('valider', 'submit') ;

    }
    public function getName()
    {
        return 'Mail';
    }
}
```

3- Ajout des actions dans notre contrôleur

Nous allons ajouter dans notre contrôleur quatre actions qui sont :

- index : qui permet de rediriger l'utilisateur vers la vue qui affiche le résultat de l'envoi.
- new : qui permet de vérifier si le formulaire est rempli ou non. Si oui, elle fait appel à la fonction sendMail. Sinon, elle fait appel à la fonction create.
- SendMail : c'est une fonction qui permet l'envoi d'email.

a- Ajout de la fonction index

```
public function indexAction() {
        return $this->render('MyAppMailBundle:Default:mail.html.twig', array());
    }return 'Mail';
}
```

b- Ajout de la fonction SendMail

```
public function sendMailAction() {
        $to = " ";
        $mail = new Mail();
        $form=  $this->createForm(new MailType(),  $mail);
        $request->get('request');
      $form->handleRequest($request) ;
        if ($form->isValid()) {
            $message = Swift_Message::newInstance()
                    ->setSubject($mail->getNom())
                    ->setFrom($mail-> getFrom())
                    ->setTo($to)
                    ->setBody($mail->getText());
            $this->get('mailer')->send($message);
           return $this->render('MyAppMailBundle:Default:mail.html.twig', array('to' => $to,
                            'from' => $mail-> getFrom()
                ));
        }
        return $this->redirect($this->generateUrl('my_app_mail_form'));}
    }
```

c- Ajout de la fonction new

```
public function newAction() {
        $mail = new Mail();
        $form=  $this-> createForm(new MailType(),  $mail);
        $request = $this->get('request');
        $form->handleRequest($request) ;

            if ($form->isValid()) {
                $this->sendMailAction('your_email@gmail.com', $mail-> getFrom(), $mail->getN
om(), $mail->getText());
            }
        return $this->render('MyAppMailBundle:Default:new.html.twig', array('form' => $form-
>createView())) ;  }
```

NB : N'oubliez pas d'ajouter « use Swift_Message; » en haut du contrôleur.

4- Création des vues

Dans cette partie, nous allons créer deux vues qui sont :

- new.html.twig : c'est la vue qui affiche notre formulaire.
- mail.html.twig : c'est la vue qui affiche le message de succès.

MyApp/MailBundle/ressources/views

a- Création de new.html.twig

```
<body>
    <h2> <strong> Formulaire De Contact</strong> </h2>
    <hr>
    <h3><p> Contacter nous </p></h3>
    <p>{{form_errors(form)}}</p>
    <hr>
    <form role="form" id="fr" method="POST" action='{{path('my_app_mail_sendpage')}}' >
        {{form_widget(form)}}
    </form>
</body>
```

a- Création de mail.html.twig

```
<body>
    Votre demande a été envoyée avec succès
</body>
```

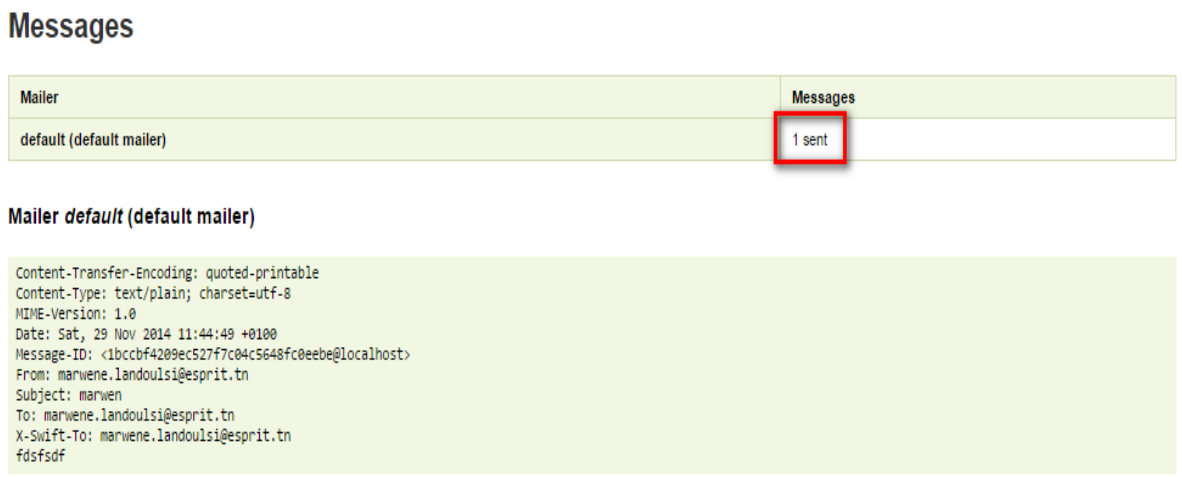
5- Configuration des routes

```
my_app_mail_succ:
    path:     /succ
    defaults: { _controller: MyAppMailBundle:Mail:index }

my_app_mail_form:
    path:     /mail
    defaults: { _controller: MyAppMailBundle:Mail:new }

my_app_mail_sendpage:
    path:  /sendmail
    defaults: { _controller: MyAppMailBundle:Mail:sendMail }
```

Vérification de l'envoi d'email

Vous pouvez vérifier si votre mail est envoyé ou pas, en consultant l'icône « enveloppe » dans la barre symfony.

En cliquant sur cette icône, vous trouvez tous les informations sur l'email envoyé.



Vous pouvez vérifier votre boite mail ! (boite mail de réception)