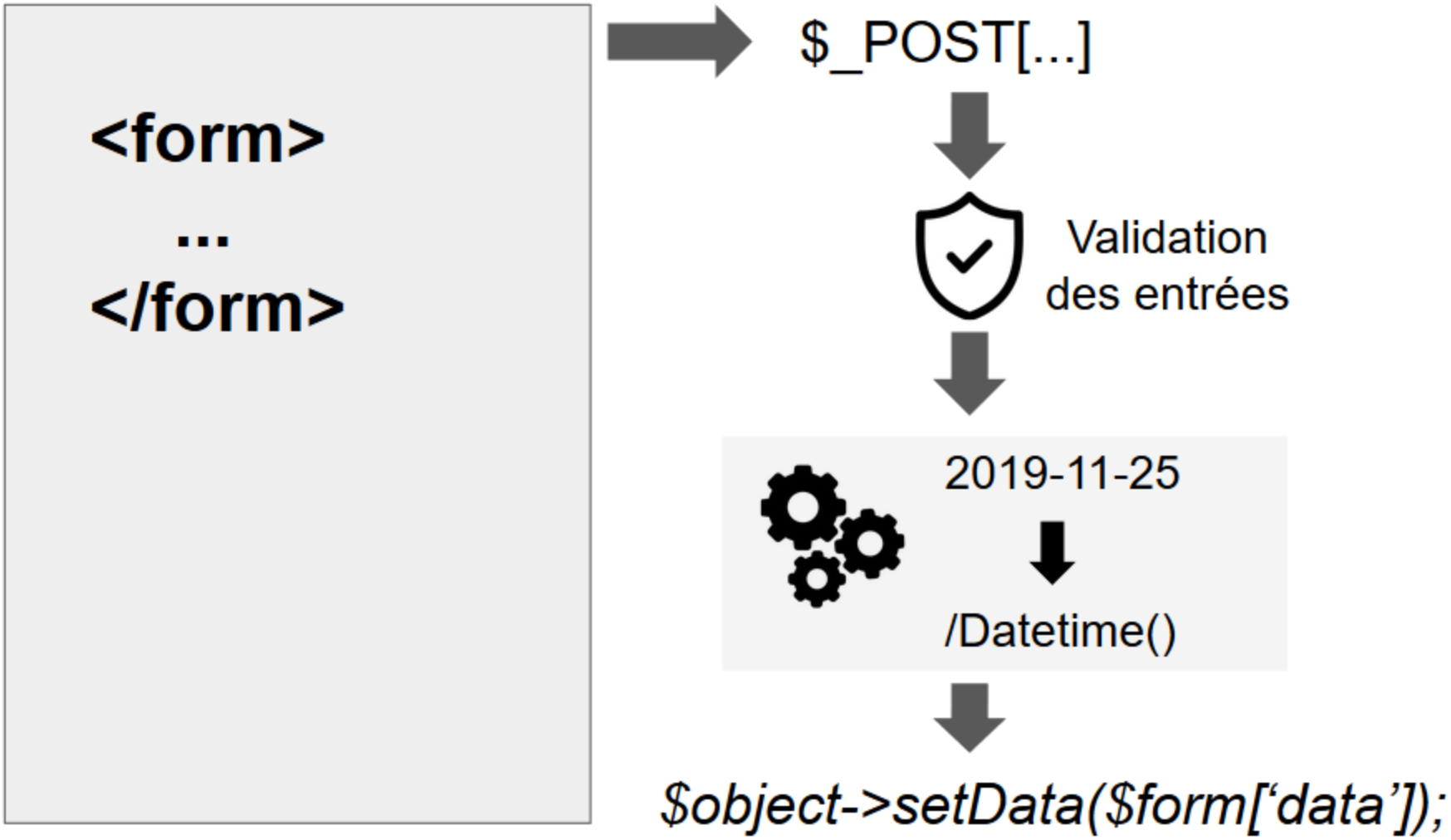
Programmation Web Avancé

Chapitre 4 : Symfony Avancé

Guillaume-Olivier LAFFITTE

guillaume.laffitte@stratow.com

Formulaires



Formulaires

```php
$formBuilder = $this->container->get('form.factory')->createBuilder();

$formBuilder
    ->add(
        "username",
        TextType::class,
        array(
            "label" => "Identifiant de l'utilisateur",
        )
    )
    ->add(
        "email",
        EmailType::class,
        array(
            "label" => "Adresse mail",
        )
    )
;

$formBuilder->getForm();
```

Formulaires

```php
public function createAction(Request $request)
{
    //build form

    $form = $formBuilder->getForm();

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid())
    {
        $data = $form->getData();

        //process

        return $this->redirectToRoute('create_success');
    }

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

Formulaires – Form Rendering

- Génération de la vue :

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

- Rendition complète :

```
{{ form_start(form) }}
{{ form_widget(form) }}
{{ form_end(form) }}
```

- Rendition d'un field avec BootStrap :

```
<div class="form-group {{ form.username.vars.valid==false ? "has-error" }}">
    {{ form_label(form.username, null, {'label_attr': {'class': 'col-sm-3 control-label'}}) }}
    <div class="col-sm-9">
        {{ form_widget(form.username) }}
        {{ form_errors(form.username) }}
    </div>
</div>
```

Formulaires - Contraintes

```php
$formBuilder
    ->add(
        "email",
        EmailType::class,
        array(
            "constraints" => array(
                new Constraints\NotBlank(),
                new Constraints\Email(),
                new Constraints\Length(array("max" => 255)),
                new Constraints\Callback(
                    array($this, "verifyExistingEmail")
                ),
            ),
            "label" => "mail"
        )
    )
;
```

Formulaires - Contraintes

- Constraints\Callback

```
public function verifyExistingEmail($value, ExecutionContextInterface $context)
{
    if($this->em->getRepository(User::class)->findOneByEmail($value])
    {
        $context->addViolation(
            "Erreur, cet email existe déjà"
        );
    }
}
```

Formulaires - Contraintes

- Pour implémenter une nouvelle contrainte il faut deux classes :
  - Une classe Constraint
  - Une classe Validator

```php
namespace Demo\DemoBundle\Form\Constraints;

use Symfony\Component\Validator\Constraint;

class UniqueEmail extends Constraint
{
    public $message = 'Erreur, cet email existe déjà';
}
```

- Par défaut le nom de la classe validator doit être :
  - *<<nom_de _la _classe_constraint>>*Validator

Formulaires - Contraintes

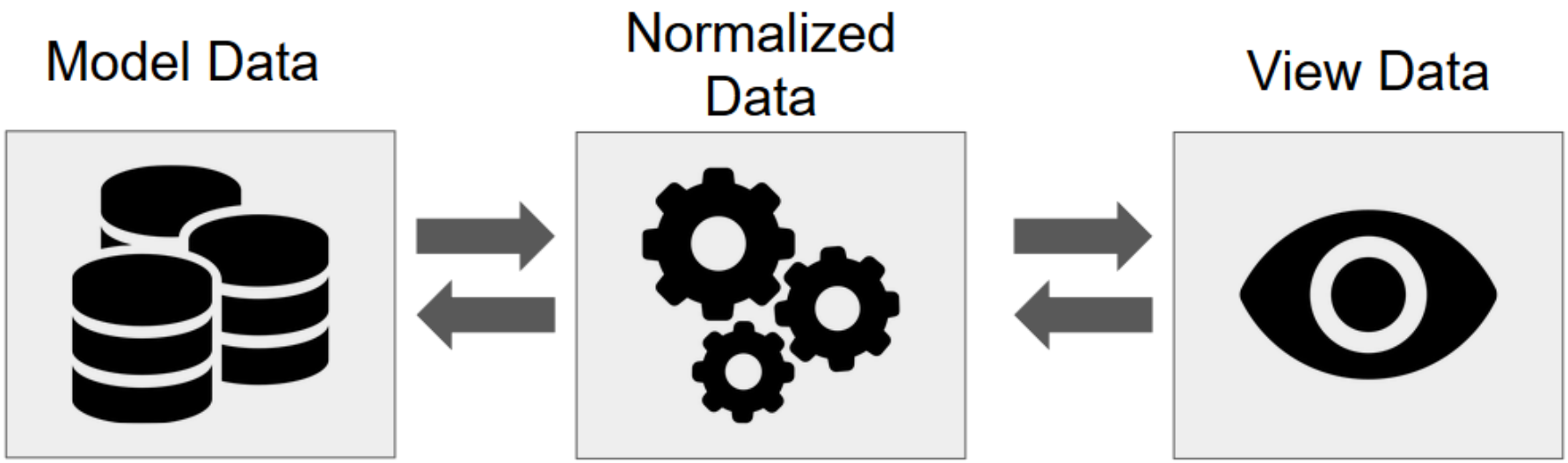
```php
namespace Demo\DemoBundle\Form\Constraints;

use Symfony\Component\Validator\Constraint;
use Symfony\Component\Validator\ConstraintValidator;

class UniqueEmailValidator extends ConstraintValidator
{
    protected $em;
    public function __construct($em) {
        $this->em = $em;
    }

    public function validate($value, Constraint $constraint)
    {
        if($this->em->getRepository(User::class)->findOneByEmail($value])
        {
            $this->context->addViolation(
                $constraint->message
            );
        }
    }
}
```

Formulaires - DataTransformer



Formulaires - DataTransformer

```php
namespace Demo\DemoBundle\Form\DataTransformer;

use Symfony\Component\Form\DataTransformerInterface;

class OctetGigaTransformer implements DataTransformerInterface
{
    // FrontEnd to BackEnd
    public function reverseTransform($sizeLimitNumberGiga)
    {

        $sizeLimitNumberOctet = $sizeLimitNumberGiga * 1000000000;

        return $sizeLimitNumberOctet;

    }

    // BackEnd to FrontEnd
    public function transform($sizeLimitNumberOctet)
    {
        $sizeLimitNumberGiga = $sizeLimitNumberOctet / 1000000000;
        return $sizeLimitNumberGiga;
    }
}
```

Security - firewalls

- Le firewall détermine comment un utilisateur va s'authentifier dans l'application.

```
security:
    firewalls:
        admin_protection:
            pattern: ^/admin/(.+)
            provider: in_memory
            http_basic: ~
```

- Ici, le firewall 'admin_protection' s'applique pour toutes les URL commençant par /admin/
- Les utilisateurs devront s'authentifier en http basic

Security - UserProvider

- Les UserProvider fournissent les informations relatives aux utilisateurs

```
security:
    providers:
        in_memory:
            memory:
                users:
                    admin1:
                        password: 123456
                        roles: 'ROLE_ADMIN'
                    admin2:
                        password: 123456
                        roles: 'ROLE_ADMIN'
        bd_provider:
            entity:
                class: DemoDemoBundle:User
                property: username
```

Security - Encoders

- Dans symfony les mots de passes sont cryptés. L'algorithme utilisé est spécifié grâce à l'encoder.

```
encoders:
    Symfony\Component\Security\Core\User\User:
        algorithm: bcrypt
        cost: 12
    Demo\DemoBundle\Entity\Users:
        algorithm: bcrypt
        cost: 12
```

Security – Access Control

- Les 'access_control' restreignent l'accès à une partie de l'application

```
security:
    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
```

- Un utilisateur à besoin du rôle admin pour accéder aux URL commençant par admin

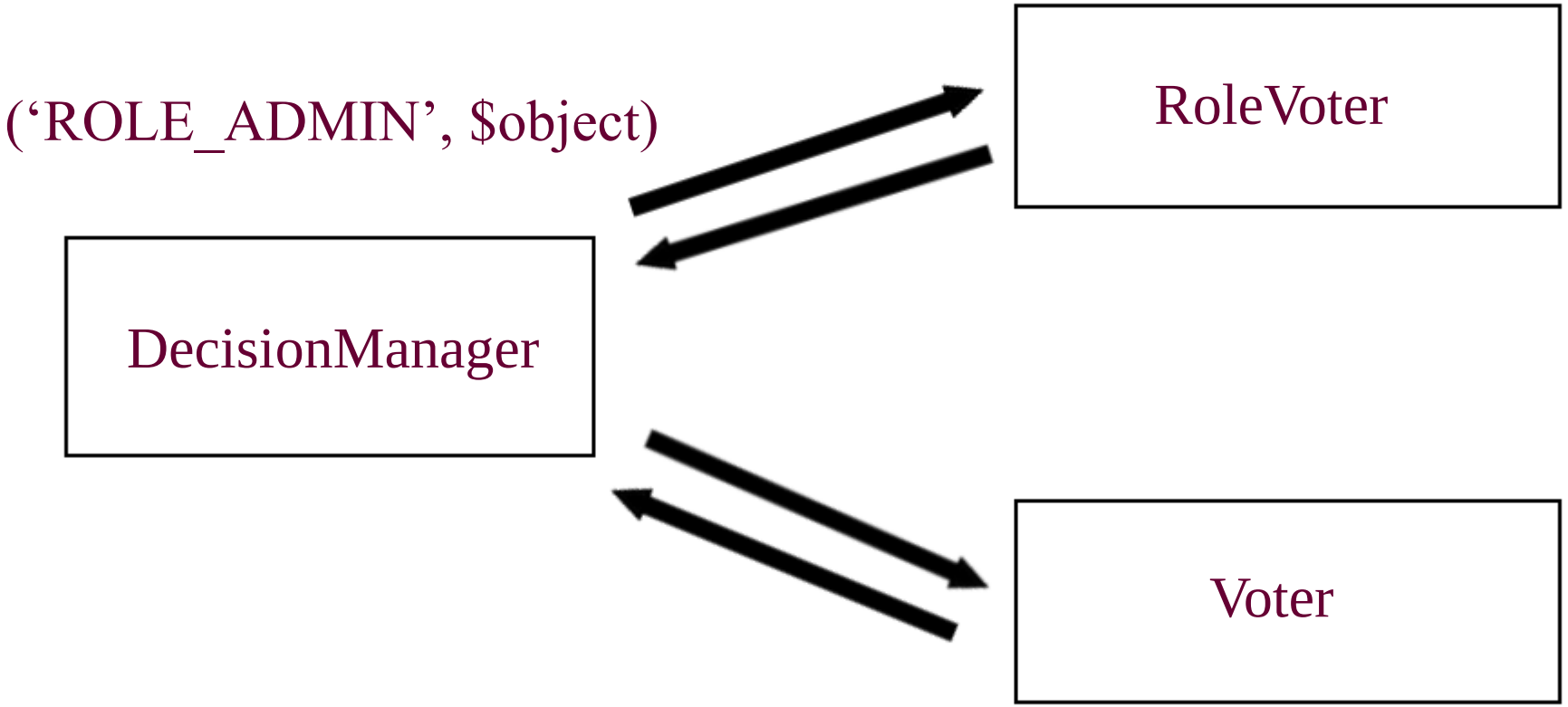
Security – Tester les droits dans un controller

```
$this->get('security.authorization_checker')->isGranted('ROLE_ADMIN'));//true or false
```

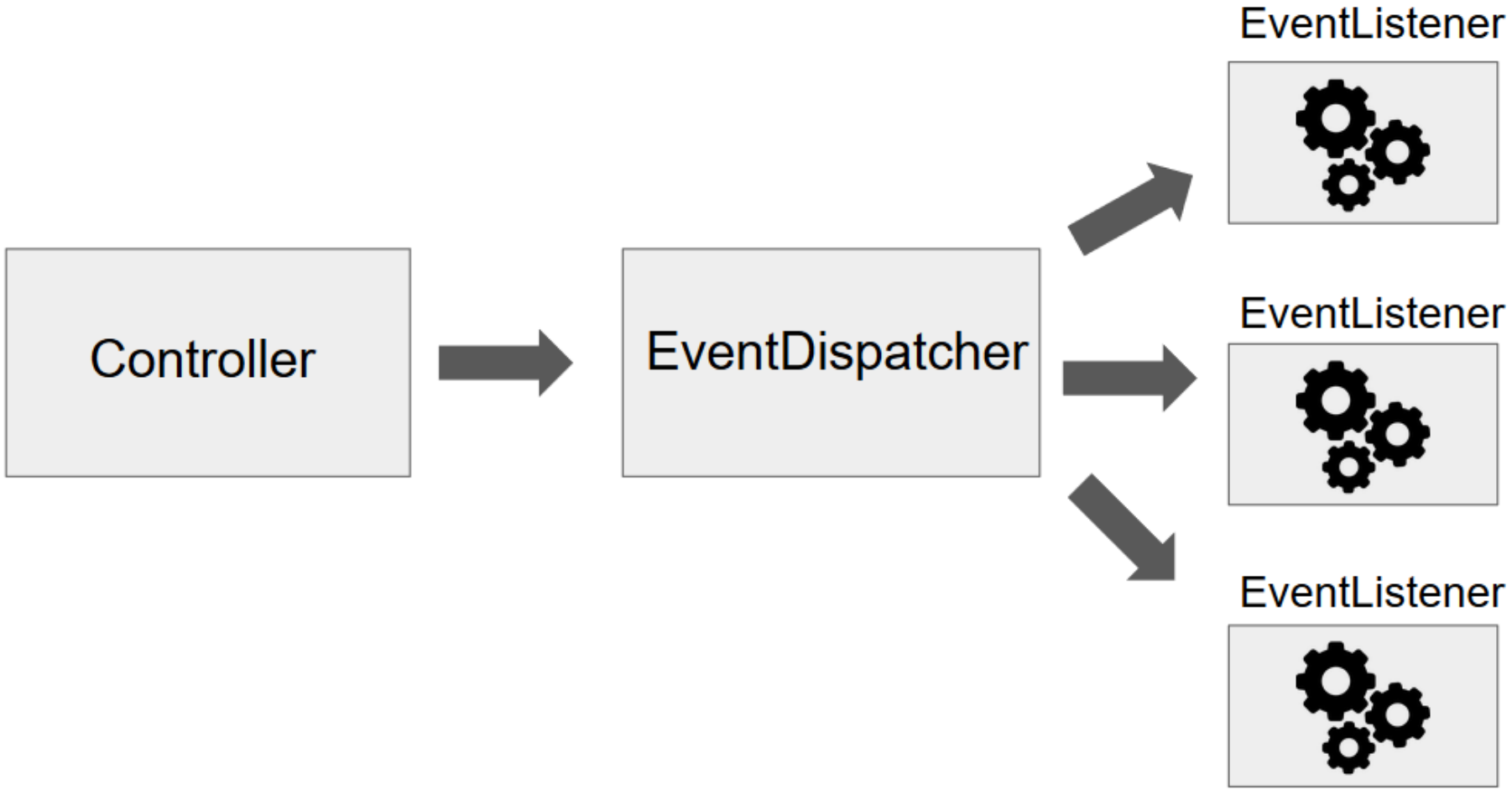
- La methode isGranted() est aussi disponible dans les templates Twig

```
{% if is_granted('ROLE_ADMIN') %}
    <p>hello Admin</p>
{% endif %}
```

DecisionManager



Gestion des Évènements



Gestion des Évènements – Exemple en JS

```
function sayHello() {
    alert('Hello World');
}
window.onload = function() {
    document.getElementById("my-button").addEventListener('click', sayHello);
}
```

```
//JQuery
$(document).ready(function () {
    $( "#my-button" ).click(function() {
        alert( "Hello World" );
    });
});
```

Gestion des Évènements – Event

- L'objet Event contient les informations utiles au traitement de l'événement

```php
<?php

namespace Demo\DemoBundle\Event;

use Symfony\Component\EventDispatcher\Event;

class UserEvent extends Event
{
    protected $user;
    protected $username;
    protected $password;
    protected $email;

    //getter et setter
}
```

Gestion des Évènements – Event

```php
$event = new UserEvent;
$event
    ->setPassword('123456')
    ->setUsername('administrator')
    ->setEmail('administrator@univ-paris13.fr')
;

$this->container->get('event_dispatcher')->dispatch('user.create', $event);
```

Gestion des Évènements – EventListener

- service.yml

```
test.user_create_event_listener:
    class: Demo\DemoBundle\EventListener\UserCreateEventListener
    arguments:
        - "@doctrine.orm.entity_manager"
    tags:
        - { name: kernel.event_listener, event: user.create, method: createUser, priority: 0 }
```

Gestion des Évènements – EventListener

```php
namespace Demo\DemoBundle\EventListener;

use Doctrine\ORM\EntityManager;
use Demo\DemoBundle\Event\UserEvent;
use Demo\DemoBundle\Entity\User;

class UserCreateEventListener
{
    private $em;

    public function __construct(EntityManager $em)
    {
        $this->em = $em;
    }

    public function createUser(UserEvent $event)
    {
        $user = new User();
        $user
            ->setUsername($event->getUsername())
            ->setPassword($passwordencoded)
            ->setEmail($event->getEmail())
        ;

        $this->em->persist($user);
        $this->em->flush();

        $event->setUser($user);
    }
}
```

Gestion des Évènements – EventListener

- service.yml

```
test.user_send_email_event_listener:
    class: Demo\DemoBundle\EventListener\UserEncryptPasswordEventListener
    tags:
        - { name: kernel.event_listener, event: user.create, method: encryptPassword, priority: 1 }

test.user_send_email_event_listener:
    class: Demo\DemoBundle\EventListener\UserSendEmailEventListener
    arguments:
        - "@doctrine.orm.entity_manager"
    tags:
        - { name: kernel.event_listener, event: user.create, method: sendEmailToUser, priority: -1 }
```

Gestion des Évènements – EventSubscriber

```php
namespace Demo\DemoBundle\EventSubscriber;

use Symfony\Component\EventDispatcher\EventSubscriberInterface;
use Demo\DemoBundle\Event\UserEvent;

class UserSubscriber implements EventSubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return array(
            'user.create' => array(
                array('encryptPassword', 1),
                array('sendEmailToUser', -1),
            )
        );
    }

    public function encryptPassword(UserEvent $event)
    {
        // ...
    }

    public function sendEmailToUser(UserEvent $event)
    {
        // ...
    }
}
```

Cache

```php
if(!$cache->has('stats'))
{
    $value = $this->complexCalculation();
    $cache->set('stats', $value);
    return $value;
}
else
{
    return $cache->get('stats');
}


//$cache->delete('stats');
//$cache->clear();
```

Commande

```php
namespace Demo\DemoBundle\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

class cronCommand extends Command
{
    protected function configure()
    {
        $this
            ->setName('text:cmd')
            ->setDescription('test')
            ->setHelp('test')
        ;
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        //script
    }
}
```