Support de Cours : Les vnements en Symfony 7

1. Introduction aux vnements en Symfony

Symfony utilise un systme d'vnements bas sur le dispatcher d'vnements pour permettre une communication flexible
entre diffrents composants sans couplage direct.

Le systme repose sur trois lments principaux :
- Le Dispatcher : Gre les vnements et appelle les listeners/subscribers associs.
- Les vnements : Objets transportant des informations.
- Les Listeners et Subscribers : Fonctions ou classes qui ragissent des vnements.

Symfony fournit plusieurs vnements natifs et permet la cration d'vnements personnaliss.

2. vnements Cls en Symfony

Symfony gre des vnements tout au long du cycle de vie d'une requte HTTP. Voici quelques vnements importants :

- kernel.request : Avant que le routeur ne dtermine la route
- kernel.controller : Avant l'excution du contrleur
- kernel.response : Avant l'envoi de la rponse au client
- kernel.terminate : Aprs l'envoi de la rponse (utile pour des tches lourdes en arrire-plan)
- kernel.exception : Lorsqu'une exception est leve

3. Crer un Listener d'vnement

Un Listener est une classe qui coute un vnement prcis.

Exemple de Listener pour kernel.request :

```
namespace App\EventListener;

use Symfony\Component\EventDispatcher\Attribute\AsEventListener;
use Symfony\Component\HttpKernel\Event\RequestEvent;
use Psr\Log\LoggerInterface;

class RequestListener
{
```

```
    private LoggerInterface $logger;

    public function __construct(LoggerInterface $logger)
    {
        $this->logger = $logger;
    }

    #[AsEventListener(event: 'kernel.request')]
    public function onKernelRequest(RequestEvent $event)
    {
        $request = $event->getRequest();
        $this->logger->info('Nouvelle requte : ' . $request->getPathInfo());
    }
}
```

4. Crer un Subscriber d'vnements

Un Subscriber permet d'couter plusieurs vnements en une seule classe.

```
namespace App\EventSubscriber;

use Symfony\Component\EventDispatcher\EventSubscriberInterface;
use Symfony\Component\HttpKernel\Event\ResponseEvent;
use Symfony\Component\HttpKernel\Event\ExceptionEvent;
use Symfony\Component\HttpKernel\KernelEvents;
use Psr\Log\LoggerInterface;

class KernelSubscriber implements EventSubscriberInterface
{
    private LoggerInterface $logger;

    public function __construct(LoggerInterface $logger)
    {
        $this->logger = $logger;
    }

    public static function getSubscribedEvents(): array
    {
        return [
```

```
            KernelEvents::RESPONSE => 'onKernelResponse',
            KernelEvents::EXCEPTION => 'onKernelException',
        ];
    }

    public function onKernelResponse(ResponseEvent $event)
    {
        $this->logger->info('Une rponse a t envoye.');
    }

    public function onKernelException(ExceptionEvent $event)
    {
        $exception = $event->getThrowable();
        $this->logger->error('Une exception est survenue : ' . $exception->getMessage());
    }
}
```

5. Crer un vnement Personnalis

Dfinition d'un vnement UserRegisteredEvent :

```
namespace App\Event;

use Symfony\Contracts\EventDispatcher\Event;
use App\Entity\User;

class UserRegisteredEvent extends Event
{
    public function __construct(private User $user) {}

    public function getUser(): User
    {
        return $this->user;
    }
}
```

6. Conclusion

Les vnements en Symfony 7 permettent de dcoupler le code et de ragir des actions sans modifier le code source principal.

Questions ? Besoin d'un exercice pratique ?