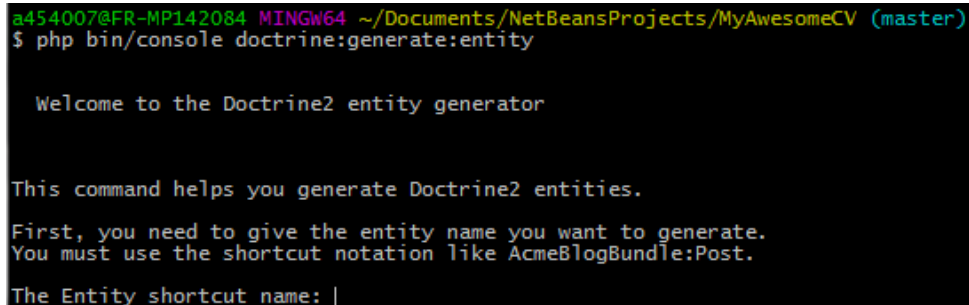


« Day 3 »

La semaine dernière, nous avons manipulé le modèle avec doctrine <http://www.doctrine-project.org/>. Nous l'avons fait manuellement, mais il est possible de le faire avec la commande `doctrine:generate:entity`.



```
a454007@FR-MP142084 MINGW64 ~/Documents/NetBeansProjects/MyAwesomeCV (master)
$ php bin/console doctrine:generate:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate.
You must use the shortcut notation like AcmeBlogBundle:Post.

The Entity shortcut name: |
```

1. Suivre la procédure pour générer de nouvelles entités : Passions, Loisirs, etc...
2. Dans votre action index de votre contrôleur par défaut, récupérez maintenant toutes vos données (Expériences, Formations, Loisirs) après les avoir créés, comme dans le « Day 2 » (on peut aussi imaginer les créer avec phpMyAdmin...)
 - a. La « magic query » `findAll` peut être utilisée de la façon suivante :
`$this->getDoctrine()->getRepository('AppBundle:Loisir')->findAll();`
 - b. Essayer de trouver avec `findBy` un moyen de récupérer que les loisirs qui commence par une lettre spécifique <http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/working-with-objects.html>
 - c. Transmettez ces données à votre vue twig
 - d. Remplacez dans votre page **index.html.twig** les éléments pouvant être généré dynamiquement comme pour le « Day 2 » (Etape 7 et 8)

Api-Platform (Question bonus Day 2)

Mettre à jour **composer** avec « `api-platform/core` », pour cela, taper « **php composer.phar require api-platform/core** »

Mettre à jour votre fichier `app/config/routing.yml` et `app/AppKernel.php` comme indiqué <https://api-platform.com/docs/core/getting-started#installing-api-platform-core>

Utiliser l'annotation `@ApiResponse` sur chaque entité de votre modèle pour générer une API de votre modèle.

Si la page `/api` ne s'affiche pas correctement : « `php bin/console assets:install` » permet d'installer les images et css de ce Bundle.

Les formulaires (créer/modifier) des données de vos pages

L'objectif de cette section est de vous permettre de créer ou modifier des données à l'aide de pages dédiées sur votre site. On parle alors de « formulaire ».

1. Créer un répertoire «Form » dans votre bundle AppBundle
2. Créer dans ce répertoire une classe du nom de votre entité suffixé de Type : `LoisirType.php`. Cette classe va servir à définir comment votre formulaire doit se comporter quand on souhaite créer un nouveau « Loisir »... Pour Loisir, nous avons

juste besoin d'un champ de type Text pour le nom du loisir que l'on va considérer obligatoire :

```
<?php

namespace AppBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\Form\Extension\Core\Type\TextType;

class LoisirType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder->add('name', TextType::class, array('required' => true));
    }

    public function getBlockPrefix()
    {
        return 'form_loisir';
    }
}
```

3. Ensuite il faut appeler votre FormType dans un contrôleur dédié (créer un nouveau contrôleur pour toutes les opérations sur votre entité Loisir) et l'associer à une entité loisir « vide »

```
<?php

namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use AppBundle\Form\LoisirType;
use AppBundle\Entity\Loisir;

/**
 * @Route("/loisirs")
 */
class LoisirController extends Controller
{
    /**
     * @Route("/create", name="create_loisir")
     * @Template()
     */
    public function createAction()
    {
        $loisir = new Loisir();
        $form = $this->createForm(LoisirType::class, $loisir);

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

4. Il ne vous reste plus qu'à créer une page dédiée pour votre nouvelle action dans views/Loisir/create.html.twig (rajouter la balise <html> et <body> pour avoir la toolbar de debug Symfony)

```
<h3>Création d'un nouveau loisir</h3>
```

```
{{ form(form) }}
```

5. Le formulaire s'affiche mais ne sauvegarde pas encore vos données !
6. Pour cela, rajouter dans votre fichier create.html.twig la ligne :
{{ form_start(form, {'action': path(validate_create_loisir) }) }} avant la balise
form(form)
7. Enfin faite l'action validate_create_loisir dans votre contrôleur des loisirs :

```
/**
 * @Route("/create_valid", name="validate_create_loisir")
 * @Method("POST")
 */
public function validateLoisirAction(Request $request)
{
    $loisir = new Loisir();
    $form = $this->createForm(LoisirType::class, $loisir);
    $form->handleRequest($request);

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

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

    return $this->redirectToRoute('create_loisir', array(
        'entity' => $loisir,
        'form' => $form->createView(),
    ));
}
```

A votre avis, que faut-il changer pour non plus créer de nouveaux loisirs mais les éditer ?

Un indice ? ...

```
/**
 * @Route("/edit/{id}", name="edit_loisir")
 * @Template()
 */
public function editAction($id)
{
    $entityManager = $this->getDoctrine()->getManager();
    $loisir = $entityManager->getRepository("AppBundle:Loisir")->findOneBy(["id" => $id]);
    $form = $this->createForm(LoisirType::class, $loisir);

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