Formation Symfony 2

Auteurs:

- **Socle commun:** Cédric Vanet (@cedvan)
- **Socle Fiducial:** Jonathan Vuillemin (jvuillemin@norsys.fr)

Norsys (@norsys)

Objectifs de la formation

- Découvrir les bases de Symfony 2
- Composer et la gestion de dépendances
- Créer un bundle
- Présenter le DIC
- Découvrir et utiliser Doctrine 2
- Présentation de Twig
- Découvrir les formulaires
- Découvrir la sécurité
- Présenter les bonnes pratiques
- Créer une architecture de projet Symfony 2
- Présenter l'écosystème Symfony 2
- Présentation du FiducialCore
- Mise en pratique avec le FiducialCore

Présentation du fil rouge

Afin de mettre en pratique la théorie, cette formation suivra le fil rouge d'un atelier concret

Mettre en place une application qui permette à un utilisateur **connecté** de visualiser la liste d'ordinateurs d'un parc informatique.

Pour chaque ordinateur, nous afficherons son identifiant, son IP, le nom et le prénom de son utilisateur.

Requirements atelier

Installer docker
https://docs.docker.com/installation

Ouvrir un nouveau terminal et lancer l'espace de l'atelier

```
$ docker run -it --rm -p 8080:80 \
    -v ~/formation-symfony/www:/var/www \
    -v ~/formation-symfony/vhosts:/etc/nginx/sites-enabled \
    norsystechteam/formation-symfony-2
```

Affichez la page http://localhost:8080

UV 1

Découvrir les bases de Symfony 2

Objectifs

- Présentation de Symfony 2
- Découvrir les bases de Symfony 2
- Présenter composer
- Créer un bundle
- Rappel du paradigme MVC
- Présenter Twig
- Utiliser Twig render

A propos de Symfony 2

- Framework sortie en 2005 maintenu par SensioLabs
- Reconnu à l'international
- Utiliser par Drupal, Ezpublish, Dailymotion, Yahoo, Blablacar et par bien d'autres
- Puissance et maintenabilité sont les maîtres mots
- Chiffres clés
  - 1 165 contributors
  - 21 558 commits
  - Communauté de plus de 300 000 developers
- Versions
  - Last : 2.7.1 (11 juin 2015)
  - LTS : 2.7 (30 mai 2015)
  - 2.6 (28 novembre 2014)
  - 2.5 (31 mai 2014)
  - 2.4 (3 décembre 2013)
  - 2.3 (3 juin 2013)

Les fondamentaux

- Très bonne documentation en ligne
- Environnement de débogage détaillé
- Commandes console
- Outils de cache
- Internationalisation
- Configuration flexible
- Gestion des performances

Installation de Symfony

Trois méthodes

- Binary Symfony
- Composer
- Manuelle

Installation Manuelle

Télécharger une archive symfony (LTS 2.7)

Décompression de l'archive

```
$ tar zxvf Symfony_Standard_Vendors_2.X.X.tgz
$ unzip Symfony_Standard_Vendors_2.X.X.zip
```

Installation via Composer

Récupération de composer
nécessite php

```
$ curl -s https://getcomposer.org/installer | php
```

Création d'une application Symfony

```
$ composer create-project symfony/framework-standard-edition project(=formation-symfony)
```

Installation via Binary

Installation de l'installeur Symfony

```
$ curl -LsS http://symfony.com/installer -o /usr/local/bin/symfony
$ chmod a+x /usr/local/bin/symfony
```

Création d'une application Symfony 2.7 formation-symfony

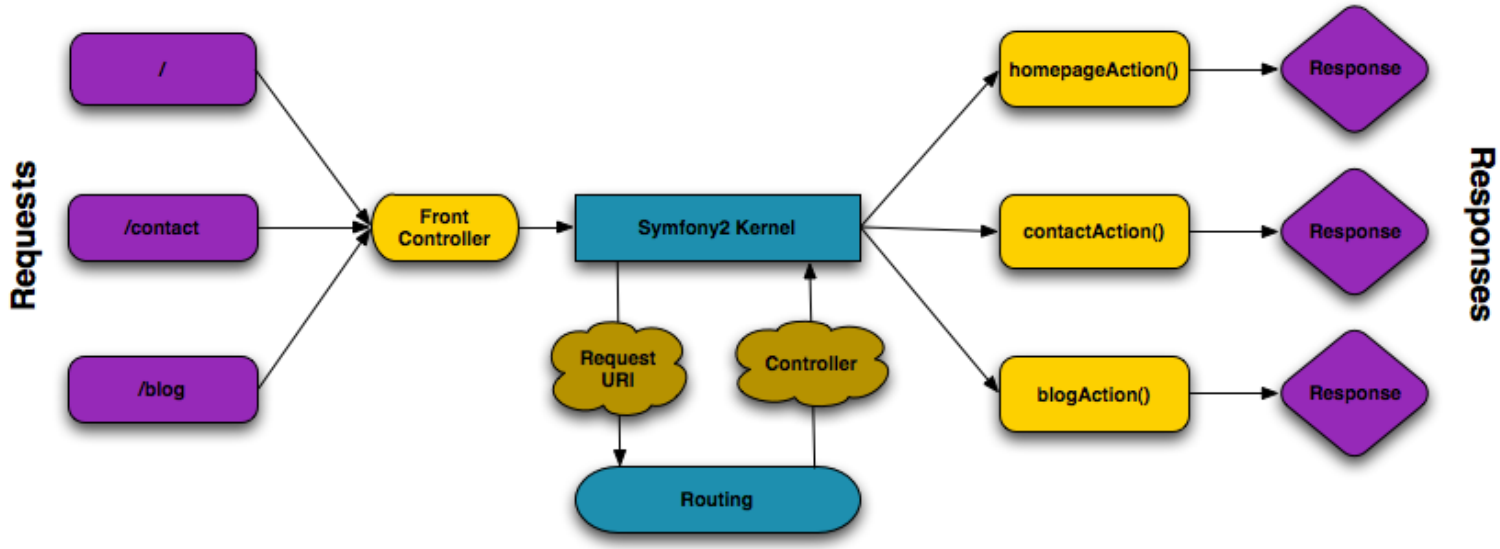
```
$ symfony new formation-symfony 2.7
$ cd formation-symfony && ll
```

Structure des dossiers

```
/app                # Configuration de l'application
    /Ressources     # Assets et Views globales
    /config         # Configuration environnements, sécurité, routing, ...
/var                # Fichiers temporaires, dossiers créé automatiquement par Symfony
    /cache
    /logs
    /session
/bin                # Binaries nécessaire à l'application
/src                # Sources de votre projet
/vendor             # Librairies PHP externes
/web                # Contenu exposé au public (contient les controlleurs frontaux)
```


Timeline des requètes HTTP sur le serveur

- Interception par Nginx / Apache2
- Redirection de Nginx / Apache2 vers un controlleur frontal (exemple web/app.php)



Gestion des environnements

Chargement des différents environnements par les biais des controlleurs frontaux situé à la racine du dossier web

Production *(app.php)*
Développement *(app_dev.php)*
Test *(app_test.php)*

```
# web/app.php

use Symfony\Component\ClassLoader\ApcClassLoader;
use Symfony\Component\HttpFoundation\Request;

$loader = require_once __DIR__.'/../var/bootstrap.php.cache';
$loader->unregister();

require_once __DIR__.'/../app/AppKernel.php';

$kernel = new AppKernel('prod', false);

$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
```

Activation de notre application Symfony

```
$ rm -f /etc/nginx/sites-enabled/default  # Suppression du vhost de nginx par défaut
$ cp -R /etc/nginx/sites-available/formation-symfony /etc/nginx/sites-enabled/
# Activation du vhost symfony préconfiguré de notre application
$ cat /etc/nginx/sites-enabled/formation-symfony
# Affichage du vhost symfony préconfiguré de notre application
$ service nginx reload  # rechargement de nginx
```

Rafraichisssez la page http://localhost:8080

You are not allowed to access this file. Check app_dev.php for more information.

Suppression de la sécurité du controlleur frontal de développement

Modification du controleur frontal app_dev.php

```
# web/app_dev.php

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Debug\Debug;

if (isset($_SERVER['HTTP_CLIENT_IP'])
|| isset($_SERVER['HTTP_X_FORWARDED_FOR'])
|| !(in_array(@$_SERVER['REMOTE_ADDR'], array('127.0.0.1', 'fe80::1', '::1')) || php_sapi
_name() === 'cli-server')
) {
header('HTTP/1.0 403 Forbidden');
exit('You are not allowed to access this file. Check '.basename(__FILE__).' for more info
rmation.');
}

...
```

Supprimer le *if* de sécurité

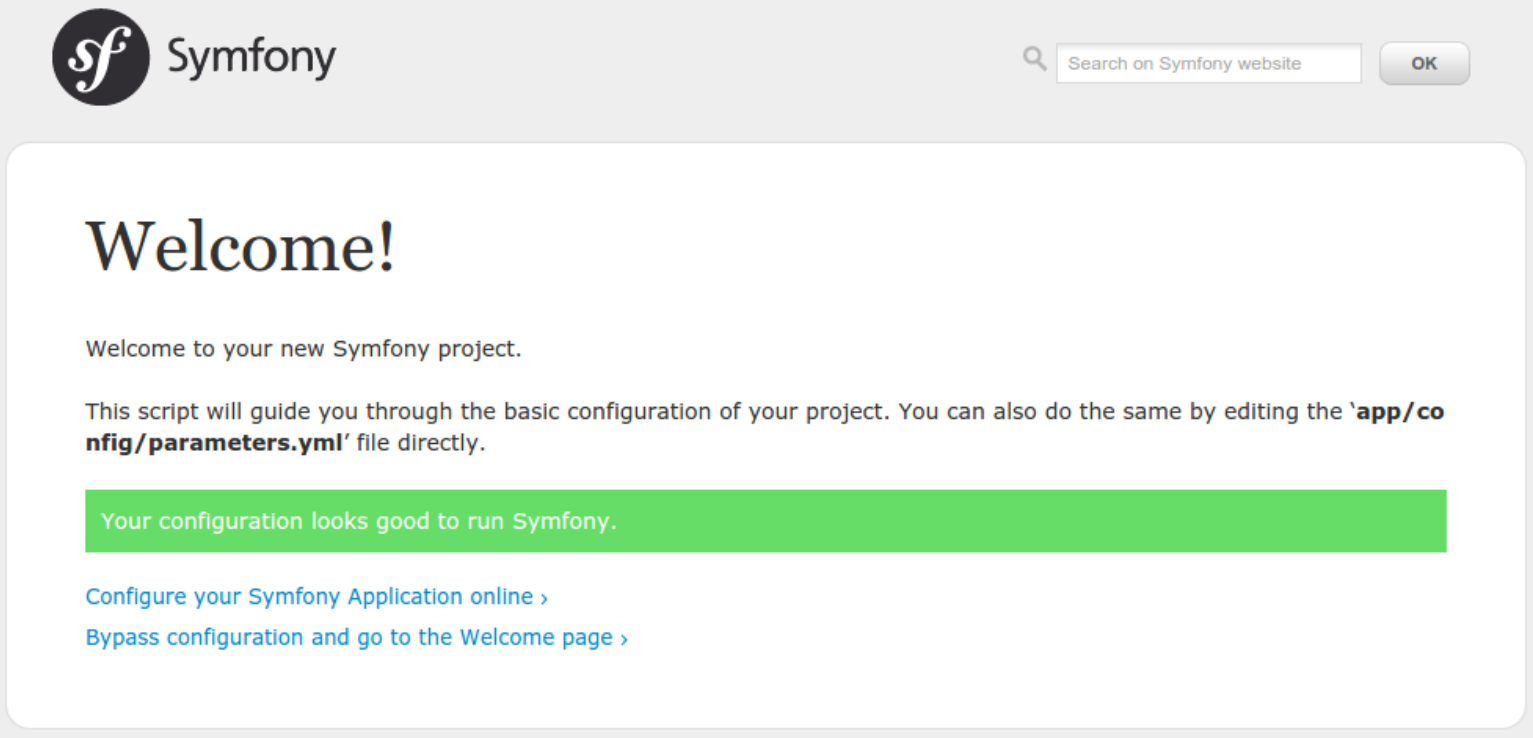
Correction du controlleur frontal de développement (suite)

Rafraichisssez la page http://localhost:8080

Contrôle de la configuration serveur pour Symfony

Affichez la page http://localhost:8080/config.php

Corriger le problème d'accès de la même manière que précédemment et rafraichisssez la page



Composer

Symfony requiert plusieurs ressources PHP pour fonctionner
Composer est un logiciel développé en php dans le but de simplifier la gestion de dépendances PHP

il se manipule en ligne de commande et se base sur deux fichiers :

- composer.json (Configuration des ressources PHP minimum nécessaire à l'application
- composer.lock (Liste des ressources PHP nécessaire à l'application

Le *.lock* est automatiquement généré à l'aide du *.json*

Composer stock ces fichiers téléchargés dans le dossier vendor et assure l'autoload des classes PHP de l'application à l'aide des Namespaces

Pour en savoir plus

Namespaces

Les espaces de noms représente une classe PHP ou un ensemble de classes PHP
De la même manière que les fichiers et les dossiers, une classe World ne peux existé deux fois dans le même namespace

les namespaces on résolus deux problèmes au sein de l'environnement PHP :

- Collisions de noms entre le code que vous créez, les classes, fonctions ou constantes internes de PHP, ou celle de bibliothèques tierces.
- La capacité de faire des alias ou de raccourcir des Noms_Extrememement_Long pour aider à la résolution du premier problème, et améliorer la lisibilité du code.

Voir la documention PHP pour en connaîte un peu plus
http://php.net/manual/fr/language.namespaces.rationale.php

l'AppKernel

L'Appkernel est le composant qui permet de charger dans son intégralité le kernel Symfony

```php
<?php

use Symfony\Component\HttpKernel\Kernel;
use Symfony\Component\Config\Loader\LoaderInterface;

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 AppBundle\AppBundle(),
        );

        if (in_array($this->getEnvironment(), array('dev', 'test'))) {
            $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
            $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
        }

        return $bundles;
    }

    public function registerContainerConfiguration(LoaderInterface $loader)
    {
        $loader->load($this->getRootDir().'/config/config_'.$this->getEnvironment().'.yml');
    }
}
```

Configuration de l'application

Tous les composants utilisés par le kernel Symfony nécessitent des configurations
Symfony 2 s'appuie sur l'utilisation du langage YAML *(YAML Ain't Markup Language)* pour répondre à ce besoin

Yaml est un Langage de représentation de données par sérialisation Unicode.
Stockage des données en string et tableaux de manière simples et efficaces

3 fichiers de configutation de base :

- app/config.yml (Configuration des bundles du kernel)
- app/routing.yml (Configuration des routes)
- app/security.yml (Configuration des firewalls)

Plus d'information sur ce langage et son utilisation dans

Configuration des bundles

```yaml
imports:
    - { resource: parameters.yml }
    - { resource: security.yml }
    - { resource: services.yml }

# Put parameters here that don't need to change on each machine where the app is deployed
# http://symfony.com/doc/current/best_practices/configuration.html#application-related-configuration
parameters:
    locale: en

framework:
    #esi:             ~
    #translator:      { fallbacks: ["%locale%"] }
    secret:          "%secret%"
    router:
        resource: "%kernel.root_dir%/config/routing.yml"
        strict_requirements: ~
    form:            ~
    csrf_protection: ~
    validation:      { enable_annotations: true }
    #serializer:      { enable_annotations: true }
    templating:
        engines: ['twig']
        #assets_version: SomeVersionScheme
    default_locale:  "%locale%"
    trusted_hosts:   ~
    trusted_proxies: ~
    session:
        # handler_id set to null will use default session handler from php.ini
        handler_id:  ~
    fragments:       ~
    http_method_override: true

# Twig Configuration
twig:
    debug:            "%kernel.debug%"
    strict_variables: "%kernel.debug%"
```

app/config/config.yml

Notion de Bundle

Un bundle représente une partie spécifique relative de votre application.

Voici quelques exemples:

- User
- Blog
- Planning
- Facturation

Concept devenu obsolète
Bonne pratique, limiter le nombre de bundles et tendre vers des applications mono bundle

Création de notre premier bundle

Nous allons créer le ParkBundle qui sera destiné à compartimenter tout notre code concernant la gestion du parc informatique

Pour se faire, Symfony nous fournit une console (détaillée dans la session 2) afin de nous faciliter les tâches répétitives

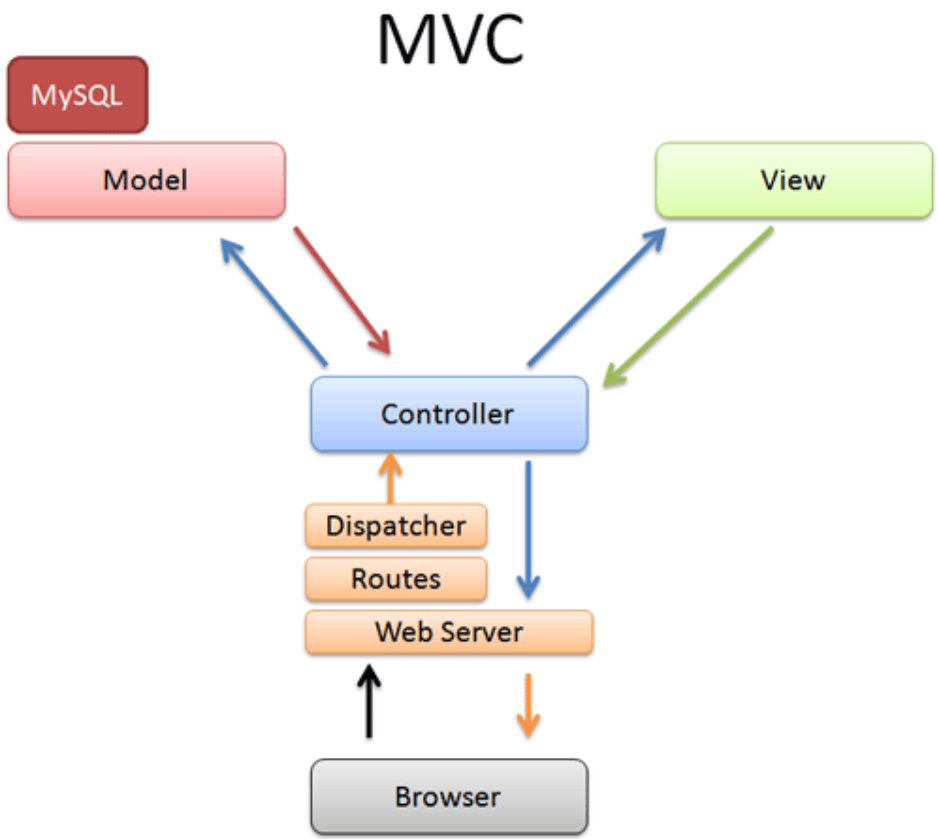
```
$ php app/console generate:bundle

Bundle namespace : ParkBundle
Keep ParkBundle as the bundle namespace : yes
Bundle name : ParkBundle
TargetDirectory : /var/www/formation-symfony/src
Configuration : yml
Generate whole directory structure : yes
confirm generation : yes
confirm update kernel : yes
confirm update router : yes

$ ll /var/www/formation-symfony/src/Formation/Bundle/ParkBundle
```

Structure d'un bundle

```
/Controller
/DependencyInjection  # Dépendances et injections
/Resources
    /config  # Configuration (Routing, Services, etc...)
    /doc  # Docmentation
    /public  # Ressources public destiné à être copié dans le dossier web (Css, Js, Img)
    /translations  # Traductions
    /views  # Vues Twig
/Tests  # Tests unitaires et fonctionnels
```

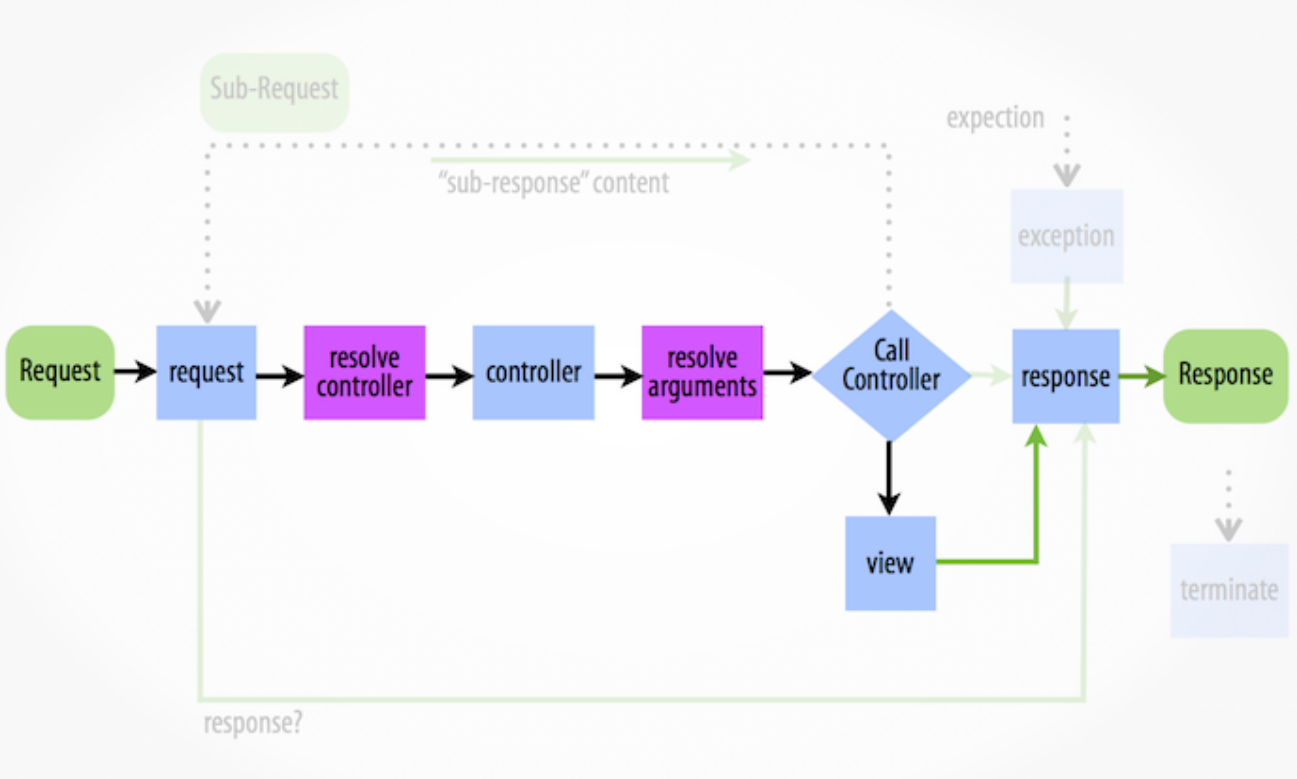
Paradigme MVC

Model Vue Controller



Timeline des requètes HTTP dans le kernel

Les requètes HTTP dans le kernel Symfony sont sous traitées au composant HttpRequest.



Pour plus de détail sur le fonctionnement de ce composant http://symfony.com/fr/doc/current/components/http_kernel/introduction.html

Router

Le travail du router consiste à jouer le rôle du resolver de controlleur

Il parse les URI des requètes HTTP reçu afin de déterminer grâce à sa liste de routes quelle méthode de controlleur éxecuter

Configuration YML

```
# app/config/routing.yml
blog_show:
    path:      /blog/{slug}
    defaults:  { _controller: AppBundle:Blog:show }
```

Router

Configuration par annotations

```
// src/AppBundle/Controller/BlogController.php
namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

class BlogController extends Controller
{
    /**
    * @Route("/blog/{slug}", name="blog_show")
    */
    public function showAction($slug)
    {
    // ...
    }
}
```

Router avec conditions

Le router permet d'affiner sa selection de controlleur en précisant notamment des conditions :

- Méthode HTTP de la requète
- Format de réponse attendu
- Paramètres envoyés
- ...

Configuration YML

```
# app/config/routing.yml
article_show:
    path:     /articles/{year}.{_format}
    defaults: { _controller: AppBundle:Article:show, _format: html }
    methods:  [GET]
    requirements:
        _format:  html|rss
        year:     \d+
```

Router avec conditions

Configuration par annotations

```
// src/AppBundle/Controller/ArticleController.php
class ArticleController extends Controller
{
    /**
    * @Route(
    *     "/articles/{year}.{_format}",
    *     defaults={"_format": "html"},
    *     requirements={
    *         "_format": "html|rss",
    *         "year": "\d+"
    *     }
    * )
    * @Method("GET")
    */
    public function showAction($year, $title)
    {
    }
}
```

Router avec conditions

Cas de **_format** avec les annotations

Lorsque vous utilisez ce paramètre, la valeur adaptée devient le "format de la demande" de l'objet de demande.

En fin de compte, le format de la demande est utilisé pour des choses telles que le réglage de la Content-Type de la réponse (par exemple, un format de demande de JSON se traduit par un Content-Type d'application / JSON)

Il peut également être utilisé dans le dispositif de commande pour rendre un modèle différent pour chaque valeur de _format. Le paramètre _format est un moyen très puissant pour rendre le même contenu dans des formats différents.

Routing avancé

Pour aller encore plus loin avec le routing ...

Configuration YML

```
# app/config/routing.yml
article_show:
    path:     /articles/{_locale}/{year}/{title}.{_format}
    defaults: { _controller: AppBundle:Article:show, _format: html }
    methods:  [GET]
    requirements:
        _locale:  en|fr
        _format:  html|rss
        year:     \d+
```

Routing avancé

Configuration par annotations

```
// src/AppBundle/Controller/ArticleController.php
class ArticleController extends Controller
{
    /**
    * @Route(
    *     "/articles/{_locale}/{year}/{title}.{_format}",
    *     defaults={"_format": "html"},
    *     requirements={
    *         "_locale": "en|fr",
    *         "_format": "html|rss",
    *         "year": "\d+"
    *     }
    * )
    * @Method("GET")
    */
    public function showAction($_locale, $year, $title)
    {
    }
}
```

Afficher la liste des routes existantes

```
$ php app/console debug:router
```

Je vous invite à consulter la documentation détaillée fournit par SensioLabs
http://symfony.com/fr/doc/current/book/routing.html

Création de la page liste des ordinateurs

La route

- Créez la route
- Déterminez une méthode de controlleur à executer
- Vérifiez son bon fonctionnement avec le router debug
- Tentez d'appeler cette route avec votre navigateur
- Faites en sorte que cette route renvoie une liste d'ordinateurs

Attributs Ordinateur: id, name, ip, enabled

Twig

Système de templating développé par Sensio Labs et utilisé dans Symfony

Sa mission est de s'occuper de séparer la couche View du MVC et ainsi d'afficher et mettre en pages les vues de l'application

Il améliore également la maintenabilité du code et apporte aux graphistes une solution simple leur permettant d'éviter l'apprentissage fastidieux du langage PHP.

Layout et héritage 1/2

```
{% extends 'MobiliteMainBundle:Mobile:layout.html.twig' %}

{% block title %}{{parent()}} - {{'mobile.clients'|trans|capitalize}} {% endblock %}

{% block header %}
    <div class="row">
        <div class="col-xs-4">
            {% include 'MobiliteMainBundle:Mobile:boutonHome.html.twig' %}
        </div>
        <div class="col-xs-4" style="margin-top: 4px;padding:0">
            <h3>
                {{'mobile.tiers.fiche'|trans|capitalize}} <br>
                <small>{{client.codeTiers}}</small>
            </h3>
        </div>
    </div>
{% endblock %}

{% block content %}

    <div id="ficheClient">
```

Layout et héritage 2/2

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />

        <title>{% block title %}Ges'Mouv{% endblock %}</title>

        {% block stylesheets %}
            {% stylesheets
                '@MobiliteMainBundle/Resources/public/css/bootstrap-glyphicons.css'
                '@MobiliteMainBundle/Resources/public/css/mobile.css'
            %}
                <link href="{{ asset_url }}" type="text/css" rel="stylesheet" />
            {% endstylesheets %}
        {% endblock %}
    </head>

    <body>

        <header>                {% block header %}  {% endblock %}  </header>

        <div id="container">    {% block content %} {% endblock %}  </div>

        <footer>                {% block footer %}  {% endblock %}  </footer>
```

La gestion des assets

Les images, css et js vont être définis dans leur bundle respectif

Pour les utiliser ils devront pourtant être dans le dossier web
Pour cela nous allons utiliser la commande assets:install qui va aller les chercher dans les bundles et les ranger dans le dossier web

Exemple :

Filtre Twig

Twig gère les variables et les conditions simples de programmation (if, for, ...).

Mais il propose surtout une bilbiothèque de fonctions (filtres) très puissantes pour la mise en page

```
{% for user in users %}
{{ user.name }}
{% else %}
No user has been found.
{% endfor %}

{{ "now"|date("d/m/Y") }}
```

Twig permet également l'inclusion et l'héritage de views, ainsi il est très facile de composer des pages à l'aide de plusieurs block

Liste de filtres twig

```
capitalize  # Mise en majuscule de la premiere lettre
url  # Formatage d'une URL
date  # Formatage d'un objet Datetime
merge  # Formatage d'une URL
number_format  # Formatage d'un nombre
sort  # Tri d'un tableau
replace  # Remplacement dans une chaine de caractère
raw (deprecated)  # Autorise l'affichage de code HTML et Javascript
...
```

pour en savoir plus
http://twig.sensiolabs.org/documentation

Création de la page liste des ordinateurs

La vue

- Créer un layout personnalisé
- Mettre boostrap et insérer le dans le layout
- Mettre JQuery et insérer le dans le layout
- Créer une view index.html.twig qui sera la liste des ordinateurs
- Pour les impatients: filtre twig personnalisé (ex: rendu du statut)

Formation Symfony 2

UV 2

Découvrir les bases de Symfony 2 (suite)

Objectifs

- Présentation Doctrine
- Présenter le DIC
- Utiliser la console
- Présenter les Formulaires

norsys

L'outil console

Symfony met à disposition un outil console de commande PHP CLI nous permettant d'éviter la répétition de tâches fastidieuses

La console Symfony se trouve dans le dossier app/ et sera déplacée dans le dossier bin/ à la version 3.0

Pour obtenir la liste des commandes disponibles pour les bundles installés:

```
php app/console
```

Commande console personnalisée

Il est possible de créer des commandes personnalisées pour nos bundles

Nous pouvons par exemple générer des commandes pour envoyer des mails automatiquement à l'aide d'un cron, ou encore désactiver des utilisateurs non connectés depuis un certains temps

Documentation sur la création de commandes:
http://symfony.com/fr/doc/current/cookbook/console/console_command.html

Doctrine - Object Relational Mapping (ORM)

Gère les interactions avec la base de données

- Abstraction de la technologie de la base de données
- Lien entre le modèle objet et la base de données
- Définition et création du schéma relationnel de données
- Stockage et récupération d'entités
- Création de requêtes SQL
- Gères les relations entre les entités

Doctrine - Les finders

Très pratiques, Doctrine fournit un set de fonctions pour récupérer les entités des repositories

- findAll() → récupère toutes les entités
- find($id) → récupère l'entité dont l'identifier est $id
- findByXXX($v) → récupère toutes les entités dont le champ XXX vaut $v
- findOneByXXX($v) → récupère l'entité dont le champ XXX vaut $v

Doctrine - EntityManager

Le service permettant d'exploiter doctrine est l'entityManager.
Il permet d'accéder aux repositories sur lesquelles s'appliquent les finders.

```
$em = $this->getDoctrine()->getManager();

$entities = $em->getRepository('NorsysAnnuaireBundle:Personne')->findAll();
```

C'est avec lui que l'on va pouvoir créer et exécuter des requêtes SQL

Doctrine - Persist / Flush

Création et modification

```
$a = new Article();
$a->setTitre('Bienvenue');
$a->setAuteur($user);

$em->persist($a);
$em->flush();
```

Le persist est seulement nécessaire lors d'une création d'entité

Suppression

```
$em->remove($entity);
$em->flush();
```

Doctrine - La documentation officielle

http://symfony.com/fr/doc/current/book/doctrine.html

Et pour aller plus loin avec doctrine.

http://doctrine-orm.readthedocs.org/en/latest/#

Mise en pratique de doctrine

- Configurer la base de données
- Créer les entités du projet (doctrine:generates:entities)
- Utiliser les relations OneToMany et ManyToOne
- Créer les tables en base de données (doctrine:schema:update)
- Remplacer le tableau PHP par l'utilisation de doctrine dans le controller
- Factoriser votre code

Mise en pratique de la création de commande

- Rajouter un attribut enabled à l'entity Computer
- Créer une commande pour désactiver tout les computers

Dependancy Injection Container (DIC)

Symfony Le DIC est sans doute un des outils les plus important de Symfony, c'est même ce qui fait la force de ce dernier

Il assure l'instanciation de nos classes PHP en tant que service et gère les arguments (dépendances)

Notion de service

Un service est une classe PHP destiné à une action spécifique. Symfony fonctionne sur une architecture orientée service. Ceci permet un code réutilisable et découplé.

Exemple :

- Manager User
- Repository User
- Extension twig de gestion des locales
- Listener logs

Le DIC, un service

Le DIC est lui même un service, il pourrait donc être tentant de l'injecter dans tous nos services afin que ceux-ci puissent communiquer entre eux très facilement.

Cependant cela imposerai des dépendances beaucoup trop forte à nos classes PHP. Hors, notre but est de rendre le plus indépendant nos classes afin qu'elle utilisent seulement ce dont elleS ont besoin.

N'injectez donc jamais le container lui même à une classe PHP, sauf si c'est nécessaire et obligatoire

Déclaration de services

Les services se déclarent dans le dossier Resources/config de nos Bundles

```
# AppBundle/Resources/config/services.yml

services:

    mailer:
        class:     Mailer
        arguments: ["%mailer.transport%"]

    newsletter_manager:
        class:     NewsletterManager
        calls:
            - [setMailer, ["@mailer"]]
```

Pour en savoir plus:
http://symfony.com/fr/doc/current/components/dependency_injection/introduction.html

DIC et les tags

Nous avons parfois besoin d'utiliser un service à un instant précis et d'une manière spécifique. Les tags ont été créé pour répondre à cette nécessité.

Par exemple, pour la création d'un listener qui écouterait les events du kernel Symfony, il faudrait lui ajouter le tag kernel.event_listener

Les tags sont très puissants et décuplent les possiblités des services dans Symfony

Voir la documentation officielle pour en connaître davantage
http://symfony.com/fr/doc/current/reference/dic_tags.html

Mise en pratique du DIC

- Création d'un service computerManager destiné à récupérer les entités Computer
- Refactorisation du computerControlleur pour utiliser le computerManager

Les formulaires

Le composant Form de Symfony est très puissant mais très complexe
Il est souvent sujet aux critiques de la communauté vis à vis de sa complexité éxagérée par rapport à son apport

Les formulaires symfony ont pour mission de générer les vues html des formulaires en fonction des entités auxquels ils sont reliés

Ils assurent également la validation des données reçues afin d'assurer la cohérence de la base de données

UV 3

Présenter les bases de la sécurité et de l'authentification

Objectifs

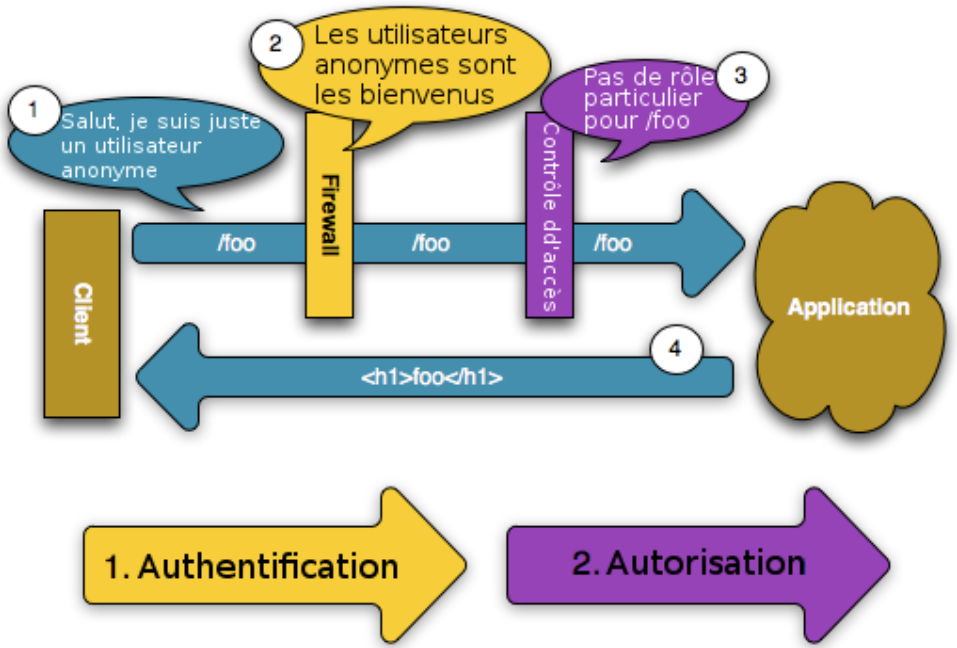
- Présenter les bases de la sécurité
- Présenter les bases de l'authentification

La base

La sécurité dans Symfony fonctionne en deux étapes

- L'authentification : le système identifie qui vous êtes
  - Formulaire d'authentification
  - Authentification HTTP
- L'authorisation : Puis il vérifie votre autorisation à accéder à la ressource demandée
  - Accès aux urls
  - Accès aux objets et aux méthodes
  - Liste de contrôle d'accès (ACL)

Fonctionnement



L'authentification

L'authentification est géré à l'aide d'un système de firewall

Ceux-ci sont déclarés dans le fichier app/config/security.yml

Fonctionnement

- Une requète HTTP arrive
- Les firewalls vérifie si l'url les concerne en fonction de leur pattern de protection
- Les firewalls concernés vérifie si l'url nécessite une authentification
- Si oui alors redirection du client vers le processus d'authentification du firewall (formulaire, ...)
- Si non le firewall laisse passer la requète

NB : une authentification est automatiquement requise si le mode anonymous dans la configuration du firewall est désactivé

L'authentification (suite)

L'authentification est complètement géré par le composant Security de Symfony

Votre seul travail consiste à :

- Configurer dans vos firewalls l'url de login
- Configurer dans vos firewalls l'url de login vérification
- De créer la page login et le formulaire de login

Le composant Sécurité interceptera tous les appels emis vers l'url de login verification afin de tenter d'authentifier l'utilisateur
Par défaut si l'authentification réussi, l'utilisateur est automatiquement redirigé verz l'URI demandée lors du déclenchement du processus d'authentification
Chaque firewall possède sa propre authentification, être authentifié sur un firewall ne signifie pas que vous l'êtes sur les autres

NB: la route de login d'un firewal doit obligatoirement être accessible en public pour le bon fonctionnement du système ;)

Exemple de firewall

```
# app/config/security.yml
security:
    firewalls:
        secured_area:
            pattern:    ^/
            anonymous: ~
            http_basic:
                realm: "Secured Demo Area"
    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
    providers:
        in_memory:
             memory:
                users:
                    ryan:  { password: ryanpass, roles: 'ROLE_USER' }
                    admin: { password: kitten, roles: 'ROLE_ADMIN' }
    encoders:
        Symfony\Component\Security\Core\User\User: plaintext
```

L'authorisation

L'authorisation s'appui sur un système de ROLE.
Chaque utilisateur possède un tableau de ROLES qui lui sont assignés

Ensuite il nous reste plus qu'à sécuriser nos routes, classes et méthodes en spécifiant un ROLE ou plusieurs ROLES requis

L'authorisation (suite)

Sécuriser une route (Le firewall test les règles de la première à la dernière et s'arrête dès le premier access_control qui match l'URI appelée)

```
# app/config/security.yml
security:
    access_control:
        - { path: ^/admin/users, roles: ROLE_SUPER_ADMIN }
        - { path: ^/admin, roles: ROLE_ADMIN }
```

Sécurisé une méthode de controlleur

```
# ParkBundle\Controleur\ComputerControlleur:newAction

if (false === $this->get('security.context')->isGranted('ROLE_ADMIN')) {
    throw new AccessDeniedException();
}
```

L'authorisation (annotations)

Sécuriser une route par l'utilisation de l'annotation **@Security**

```
// ...
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/**
* @Security("has_role('ROLE_ADMIN')")
*/
public function helloAction($name)
{
// ...
}
```

security.yml Providers

Les providers fournissent la liste des utilisateurs au composant sécurité de Symfony 2

```
providers:
    in_memory:
        memory:
            users:
                user:  { password: user, roles: [ 'ROLE_USER' ] }
                admin: { password: admin, roles: [ 'ROLE_ADMIN' ] }
                superadmin: { password: superadmin, roles: [ 'ROLE_SUPER_ADMIN' ] }
```

Voici un exemple de provider simple, il utilise une liste d'utilisateurs local et static directement écrite dans le security.yml

security.yml Encoders

Les encoders permettent de hasher les mot de passes

```
encoders:
    Symfony\Component\Security\Core\User\User: plaintext

encoders:
    Symfony\Component\Security\Core\User\User: sha512
```

Voici deux exemples encoders :

- Le premier traite le mot de passe en plaintext, donc en réalité il applique aucun hash
- Le second hash le mot de passe en sha512

security.yml Firewalls

Les firewalls sont la pièce centrale de la sécurité dans Symfony

Ils ont pour charge de gérer l'authentification, Symfony lis les firewalls l'un après l'autre et s'arrête dès qu'il en trouve un compatible avec l'URI de la request

```
firewalls:

    # Disabling the security for the web debug toolbar, the profiler and Assetic.
    dev:
        pattern:  ^/(_(profiler|wdt)|css|images|js)/
        security: false

    main:
        pattern: ^/
        anonymous: true
        provider:  in_memory
        form_login:
            login_path: security_login
            check_path: security_login_check
        logout:
            path:   security_logout
            target: /
```

security.yml
Firewalls (next)

security.yml
Role hierarchy

Symfony permet de définir des roles par utilisateurs et par groupe d'utilisateurs
Ceux-ci permettent de déclarer différents niveaux de privilèges

```
role_hierarchy:
    ROLE_ADMIN:       [ROLE_USER]
    ROLE_SUPER_ADMIN: [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

Ici trois rôles sont déclarer :

- ROLE_USER Utilisateur lambda
- ROLE_ADMIN Administrateur
- ROLE_SUPER_ADMIN Dieu (Développeur)
- ROLE_ALLOWED_TO_SWITCH Rôle automatique fournit pas Symfony pour autoriser le changement d'utilisateur authentifier

security.yml Access control

Les access control permettent d'autoriser ou non des URI en fonction des rôles de l'utilisateur

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

- La page login est autorisé aux utilisateus avec le rôle ROLE_ANONYMOUS (Tous)
- Les pages d'administration sont seulement autorisées aux utilisateus avec le rôle ROLE_ADMIN

Aller plus loin avec la sécurité

http://symfony.com/fr/doc/current/book/security.html
http://symfony.com/fr/doc/current/cookbook/security/securing_services.html
http://symfony.com/fr/doc/current/cookbook/security/form_login.html

UV 4

Bonnes pratiques avec Symfony 2

Objectifs

- Tester avec PHPUnit / Atoum
- PhpProjectAnalyser
- Découvrir les bonnes pratiques de développement d'un bundle

PHP Unit

- Tests unitaires automatisés
- Facilement utilisable avec Symfony2

https://phpunit.de/getting-started.html

PHPProjectAnalyser

- Bundle Symfony2
- Permet d'effectuer des tests de qualimétie directement depuis l'environnement de dev
- Développé par Jean Davis Labails, Norsys
  https://github.com/jdlabails/PhpProjectAnalyzer

ApplicationBundle

Ne pas créer de bundle si celui-ci n'est pas facilement réutilisable hors de l'application
Dans ce cas, stocker le code dans un ApplicationBundle

Symfony tends donc aujourd'hui vers une architecture mono bundle

Strcuture des dossiers

```
blog/
├─ app/
│   ├─ console
│   ├─ cache/
│   ├─ config/
│   ├─ logs/
│   └─ Resources/
├─ src/
│   └─ AppBundle/
├─ vendor/
└─ web/
```

En savoir plus

http://symfony.com/doc/current/best_practices/index.html
http://symfony.com/fr/doc/current/cookbook/bundles/best_practices.html

UV 5

Créer une architecture projet Symfony 2

Objectifs

- Créer une architecture de projet Symfony 2
- Présenter l'écosystème Symfony 2

Architecture type Symfony 2

```
app/
├─ cache/
├─ config/
├─ logs/
├─ console
├─ bootstrap.php.cache
└─ Resources/
bin/
src/
└─ AppBundle/
vendor/
web/
├─ app.php
└─ app_dev.php
```

Architecture mise en place à la sortie de Symfony 2. Toujous beaucoup utilisée, neanmoins elle est illogique sur quelques points.

Architecture type Symfony 3

```
app/
├─ config/
└─ Resources/
bin/
└─ console
src/
└─ AppBundle/
var/
├─ cache/
├─ logs/
└─ bootstrap.php.cache
vendor/
web/
├─ app.php
└─ app_dev.php
```

Symfony a revu ces points illogiques et a appliqué quelques correctifs à sa nouvelle architecture pour Symfony 3 afin d'adopter la même approche que le système linux.

3 approches d'architectures des sources différentes

- Le multi bundles
  - Création de plusieurs bundles
  - Tous le codes est centralisé dans les bundles
- Le mono bundle
  - Création d'un seul AppBundle
  - Tous le codes est centralisé dans ce bundle
- L'externalisation
  - Création de un ou plusieurs bundles
  - Externalisation du code PHP en dehors de bundles
  - Conserver le minimum de code dans les bundle pour la liaison avec Symfony 2

Le multi bundles

Architecture historique de Symfony 2, elle provient de la vision plugins présente en Symfony 1.4

```
src/
└─ AppBundle/
└─ BlogBundle/
└─ UserBundle/
└─ MailBundle/
└─ OrderBundle/
└─ ...
```

Deprecated depuis Symfony 2.6

Raisons de ce choix :

- Les bundles spécifiques à une application n'ont pas vocations à être autonomes et partagés
- Entraine une complexité de tri des classes non nécessaires
- Quand faut-il créer un nouveau bundle ?
- Où ranger une classe utiles à plusieurs bundles ?

Le mono bundle

Architecture mise en avant par une partie de la communauté de Symfony 2, né des raisons évoquées précédemment

De plus en plus d'architectures de ce type apparaissent à partir de la version 2.3

```
src/
└─ AppBundle/
```

Devenu une best practice depuis Symfony 2.6

L'externalisation

Architecture défendu par certains développeurs de la communauté de Symfony 2
Elle permet de sortir le code PHP pur et le code métier de Symfony afin de minimiser l'imbrication avec le Framework.
Ainsi il est plus simple d'intégrer de nouveaux outils extra framework
De plus le coût d'un changement de framework est réduit

```
src/
├─ Bundle/
│  └─ AppBundle/
│     ├─ Controller/
│     ├─ Resources/
│     └─ ...
├─ Components/
├─ Listener/
├─ Model/
└─ .../
```

Impose l'utilisation de fichiers de configuration tel que YAML afin de nettoyer le code des annotations Symfony 2
Encore peu utilisée, son usage croit tout de même de plus en

4 types de configuration possibles

Durant le développement, il est nécesaire de configurer le routing, les services, l'ORM, les assertions, etc...

Pour ce fait, les souplesse de Symfony 2 nous propose 4 possibilitées :

- Php
- Annotations
- Xlm
- Yaml

Configuration par PHP

```php
// app/config/routing.php
use Symfony\Component\Routing\RouteCollection;
use Symfony\Component\Routing\Route;

$collection = new RouteCollection();
$collection->add('blog_show', new Route('/blog/{slug}', array(
    '_controller' => 'AcmeBlogBundle:Blog:show',
)));

return $collection;
```

- Chaque fichier à un but precis, simple à maintenir
- Multiplications du nombre de fichiers de l'applications, légère perte de productivité en mode développement
- Permet l'externalisation des classes PHP en dehors du bundle
- Légèrement complexe à lire et à modifier

Configuration par annotations

```
/**
 * @Route("/{id}", requirements={"id" = "\d+"}, defaults={"foo" = "bar"})
 */
public function showAction($id)
{
}
```

- Les configurations sont proches du code, accessibilité accrut en mode développement
- Mélange du code et de la configuration, maintenance plus délicate sur un ancien projet
- Mélange de différentes actions dans un seul fichiers, complexité accrut de l'architecture

Configuration par Xml

```
AcmeBlogBundle:Blog:show
```

- Chaque fichier à un but precis, simple à maintenir
- Multiplications du nombre de fichiers de l'applications, légère perte de productivité en mode développement
- Permet l'externalisation du code PHP en dehors de bundles
- Complexe à lire et à modifier

Configuration par Yaml

```
# app/config/routing.yml
blog_show:
    path:    /blog/{slug}
    defaults:  { _controller: AcmeBlogBundle:Blog:show }
```

- Chaque fichier à un but precis, simple à maintenir
- Multiplications du nombre de fichiers de l'applications, légère perte de productivité en mode développement
- Non supporté par PHP, demande un pre traitement par l'application, le cache empêche la perte de performance
- Permet l'externalisation du code PHP en dehors de bundles
- Système choisit par Symfony 2 pour la configuration générale des bundles (cf config.yml)
- Très simple à lire et à modifier

Choisir son architecture

Ce choix se défini selon vos besoins et vos affinités

Pensez toutefois à rester proche des bests practices afin de faciliter l'intégration de nouveaux développeurs.

Cependant, si vous estimez une solution plus simple d'utilisation pour le developpement de votre application, n'hésitez pas !
Symfony 2 offre une souplesse importante afin que chacun puisse trouver chaussure à son pied.

Création d'un projet de type mono bundle

Dans cette formation, nous allons suivre les bests pratices.
Nous allons donc refactoriser notre projet en type mono bundle

- Création d'un AppBundle
- Refactorisation de notre ParkBunde dans l'AppBundle
- Mise en place d'une configuration Yaml pour le Routing
- Mise en place d'une configuration Yaml pour les services
- Mise en place d'une configuration Yaml pour l'ORM
- Mise en place d'une configuration Yaml pour les assertions

L'écosystème de Symfony 2

Symfony 2 est devenu un framework à échelle internationale.

Par conséquent ça communauté est importante.

Symfony soutient le partage de bundles.
On trouve sur le web via Github notemment, un grand nombre de projets destinés à être partagé avec les développeurs Symfony 2.

Faisons un tour dans cet écosystème remplit d'outils utiles lors de nos développement.

FOSUserBundle

Gestion automatique de la couche utilisateur

- Founit deux classes User et Groupe propres et fonctionnelles
- Authentification
- Authorisation
- Inscription avec confirmation par mail optionnelle
- Réinitialisation du mot de passe

- 2 375 commits
- 286 contributeurs
- 16 releases (dernière le 1 Juin 2015)

Documentation:
https://symfony.com/doc/master/bundles/FOSUserBundle/index.html
Github: https://github.com/FriendsOfSymfony/FOSUserBundle

FOSRestBundle

Ajout d'outils pour facilité le développement d'une application RESTful

- Gestion du format de sortie des vues par le biais des controlleurs
- Router avec des urls respectant les conventions
- Custom mime types dans les Headers
- Authorisation
- RESTful decoding du corp des requêtes HTTP
- Exceptions avec HTTP status codes approriées

- 1 969 commits
- 186 contributeurs
- 16 releases (dernière le 1 Juin 2015)

Documentation:
http://symfony.com/doc/master/bundles/FOSRestBundle/index.html
Github: https://github.com/FriendsOfSymfony/FOSRestBundle

DoctrineBehaviors

Ajout de features pour Doctrine

- Tree - Système d'arbre
- Translatable - Traductions des entitées
- Timestampable - Date de création et modification
- SoftDeletable - Suppression simuler avec un booléen
- Blameable - Dernier utilisateur editeur
- Loggable - Ajout de log
- Geocodable - Géolocalisation
- Filterable - Facilite les filtres
- Sluggable - Ajout d'un système de slug

    - 464 commits
    - 47 contributeurs
    - 6 releases (dernière le 14 July 2015)

Github: https://github.com/KnpLabs/DoctrineBehaviors

SonataAdminBundle

Génération automatique d'un système d'administration

- Création, ajout, modification et suppression des ressources
- Authentification
- Authorisation
- Exports
- Recherche
- Internationalisation
- etc...

- 3 926 commits
- 363 contributeurs
- 18 releases (dernière le 23 Mars 2015)

Demo: http://demo.sonata-project.org
Documentation: https://sonata-project.org/bundles/admin/2-3/doc/index.html
Github: https://github.com/sonata-project/SonataAdminBundle

Sylius

Moderne e-commerce pour PHP Symfony 2

- Gestion d'articles
- Gestion de prix
- Gestion de commandes
- Gestion de factures
- Gestion de stock
- etc ...

- 5 496 commits
- 221 contributeurs
- 11 releases (dernière le 20 Mai 2015)

Site: http://sylius.org/
Demo: http://demo.sylius.org/
Documentation: http://docs.sylius.org/en/latest/
Github: https://github.com/Sylius/Sylius

GenemuFormBundle

Ajout d'extras pour les formulaires Symfony 2

- Select2
- Captcha GD
- ReCaptcha
- Tinymce
- JQueryUi
- File
- Image
- Colorpicker
- Rating

- 447 commits
- 54 contributeurs
- 9 releases (dernière le 14 Décembre 2014)

Github: https://github.com/genemu/GenemuFormBundle

AliceBundle

Ajout du support de la librairie Alice pour générer des fixtures de manière simple et efficaces

- Génération de fixtures
- Utilisation de fichiers YAML pour déclarer les fixtures
- Multiple références entre entitées
- Fonctions de génération d'username, firstname, lastname, email, slug, date, datetime, range, etc...
- Création de fonctions customs (Génération numéro de facture ?)

  - 131 commits
  - 15 contributeurs
  - 8 releases (dernière le 15 Août 2015)

Github: https://github.com/hautelook/AliceBundle
nelmio/Alice: https://github.com/nelmio/alice

NelmioApiDocBundle

Ajout d'un système de documentation automatique à une API REST

- Documentation automatique
- Respect des conventions
- Système manuel de test fonctionnel des routes
- Utilise la PHPDoc
- Interface web

    - 834 commits
    - 120 contributeurs
    - 26 releases (dernière le 16 Mai 2015)

Github: https://github.com/nelmio/NelmioApiDocBundle

NelmioJsLoggerBundle

Log les erreurs Javascripts du front

- Ajout d'un block twig au template
- Configuration du type d'erreurs à logguer

  - 40 commits
  - 7 contributeurs
  - 6 releases (dernière le 19 Août 2015)

Github: https://github.com/nelmio/NelmioJsLoggerBundle

BazingaJsTranslationBundle

Rend disponible les traductions statiques de Symfony 2 dans Javascript

- Inclusion avec Assetic
- Configuration possible des domaines actifs et locales actives
- Support des domaines
- Support de la pluralisation
- Support des placeholders
- Support de tous les types de clés

- 274 commits
- 44 contributeurs
- 35 releases (dernière le 21 Mai 2015)

Github:
https://github.com/willdurand/BazingaJsTranslationBundle

WebProfilerExtraBundle

Ajout d'informations de déboages à la web debug bar de Symfony 2

- Informations supplémentaires sur les routes
- Informations supplémentaires sur le DIC
- Informations supplémentaires sur Assetic
- Informations supplémentaires sur Twig

  - 131 commits
  - 28 contributeurs
  - 7 releases (dernière le 22 Juin 2015)

Github: https://github.com/Elao/WebProfilerExtraBundle

XhprofBundle

XHProf hiérarchique profiler pour PHP

- Ajout de rapports sur l'applications
- Nombre d'appels à des fonctions
- Temps d'execution
- Consommation CPU
- Consommation de mémoire

- 136 commits
- 29 contributeurs
- 3 releases (dernière le 16 Juin 2014)

Documentation: http://php.net/manual/fr/book.xhprof.php
Github: https://github.com/jonaswouters/XhprofBundle

ZenstruckCacheBundle

Ajout d'une commande de création d'un cache warmup HTTP

- 36 commits
- 2 contributeurs
- 8 releases (dernière le 19 Mai 2015)

Github: https://github.com/kbond/ZenstruckCacheBundle

SpraedPDFGeneratorBundle

Génération d'HTML pour des documents PDF

Utilisation d'une librairie Java Flying Saucer project, donc ce bundle requière un installation de java fonctionnelle sur l'environnement d'execution

- 81 commits
- 6 contributeurs
- 3 releases (dernière le 10 Juillet 2014)

Github:
https://github.com/stedekay/SpraedPDFGeneratorBundle

Installation de SpraedPDFGeneratorBundle

- Installation du bundle SpraedPDFGeneratorBundle avec composer
- Créer une action de téléchargement de la liste des computers en pdf

Plus de Bundles ?

Knp Bundles: http://knpbundles.com/

Packagist: https://packagist.org/search/?type=bundle

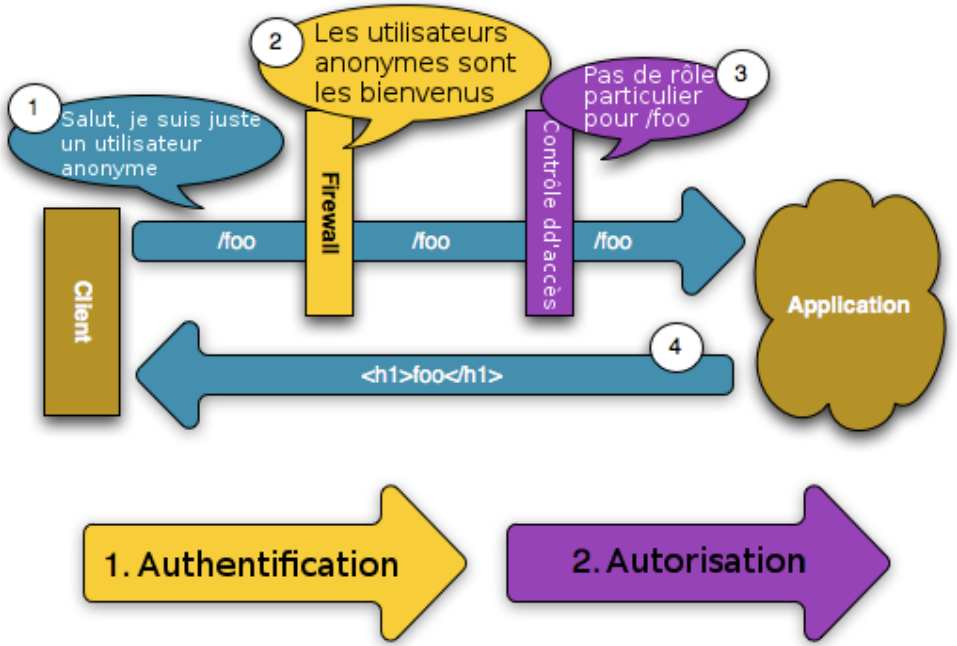
UV 6

Gestion de l'authentification

Objectifs

- Mettre en place une authentification
- Créer une distribution spécifique intégrant la gestion de l'authentification avec un bundle vendor

Reminder security



security.yml Providers

Les providers fournissent la liste des utilisateurs au composant sécurité de Symfony 2

```
providers:
    in_memory:
        memory:
            users:
                user:  { password: user, roles: [ 'ROLE_USER' ] }
                admin: { password: admin, roles: [ 'ROLE_ADMIN' ] }
                superadmin: { password: superadmin, roles: [ 'ROLE_SUPER_ADMIN' ] }
```

Voici un exemple de provider simple, il utilise une liste d'utilisateurs local et static directement écrite dans le security.yml

security.yml Encoders

Les encoders permettent de hasher les mot de passes

```
encoders:
    Symfony\Component\Security\Core\User\User: plaintext

encoders:
    Symfony\Component\Security\Core\User\User: sha512
```

Voici deux exemples encoders :

- Le premier traite le mot de passe en plaintext, donc en réalité il applique aucun hash
- Le second hash le mot de passe en sha512

security.yml Firewalls

Les firewalls sont la pièce centrale de la sécurité dans Symfony

Ils ont pour charge de gérer l'authentification, Symfony lis les firewalls l'un après l'autre et s'arrête dès qu'il en trouve un compatible avec l'URI de la request

```
firewalls:

    # Disabling the security for the web debug toolbar, the profiler and Assetic.
    dev:
        pattern:  ^/(_(profiler|wdt)|css|images|js)/
        security: false

    main:
        pattern: ^/
        anonymous: true
        provider:  in_memory
        form_login:
            login_path: security_login
            check_path: security_login_check
        logout:
            path:   security_logout
            target: /
```

security.yml
Firewalls (next)

security.yml
Role hierarchy

Symfony permet de définir des roles par utilisateurs et par groupe d'utilisateurs
Ceux-ci permettent de déclarer différents niveaux de privilèges

```
role_hierarchy:
    ROLE_ADMIN:       [ROLE_USER]
    ROLE_SUPER_ADMIN: [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

Ici trois rôles sont déclarer :

- ROLE_USER Utilisateur lambda
- ROLE_ADMIN Administrateur
- ROLE_SUPER_ADMIN Dieu (Développeur)
- ROLE_ALLOWED_TO_SWITCH Rôle automatique fournit pas Symfony pour autoriser le changement d'utilisateur authentifier

security.yml Access control

Les access control permettent d'autoriser ou non des URI en fonction des rôles de l'utilisateur

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

- La page login est autorisé aux utilisateus avec le rôle ROLE_ANONYMOUS (Tous)
- Les pages d'administration sont seulement autorisées aux utilisateus avec le rôle ROLE_ADMIN

Création d'une sécurité de base (authentification)

- Créer une configuration security.yml avec un provider in_memory
- Créer une page login
- Créer une page login_check
- Créer une page logout
- Activer une authentification à l'aide d'un SecurityController
- Ajout des liens de login / logout dans le menu

Création de rôles de base (authorisation)

- Créer des rôles : ROLE_USER, ROLE_MODERATOR, ROLE_ADMIN, ROLE_SUPER_ADMIN
- Protéger l'ajout d'un computer avec le ROLE_ADMIN
- Protéger la modification d'un computer avec le ROLE_MODERATOR
- Protéger la suppression d'un computer avec le ROLE_SUPER_ADMIN

Création d'une entité User avec ses fixtures

- Créer une entité User qui hérite de Symfony\Component\Security\Core\User\UserInterface
  - id
  - username
  - password
  - plainPassword
  - name
  - firstname
  - salt
  - roles
  - enabled
- Créer des fixtures pour l'entité User

Mise à jour de la security pour utiliser notre entité User

- Créer un provider entity
- Configurer un encoder sha512
- Créer une classe SecurityUser pour encodé les mots de passe
- Exécuter l'encodage du mot de passe sur l'évènement prePersist de Doctrine

Création d'un système de gestion utilisateurs

- Liste des utilisateurs
- Créer un utilisateur
- Modifier un utilisateur
- Supprimer un utilisateur

Maks3w/FR3DLdapBundle

Authentification LDAP pour PHP Symfony 2

- 159 commits
- 10 contributeurs
- 11 releases (dernière le 29 Avril 2015)

Documentation:
https://github.com/Maks3w/FR3DLdapBundle/blob/master/Resources/doc/index.md
Github: https://github.com/Maks3w/FR3DLdapBundle

Formation Symfony 2

UV 7
FiducialCore

Objectifs

- Présentation du FiducialCore
- Présentation de ses fonctionnalités
- Comment travailler avec FiducialCore

Présentation

FiducialCore est un ensemble de Bundles Symfony2, à ce jour:

- **CoreBundle**: bundle principal
- **UserBundle**: bundle gestion utilisateurs
- **ReferentialBundle**: bundle données référentiel

Basé sur l'utilisation de **bundles majeurs SF2**
(FosUser, FosRest, JMSSerializer, KnpMenu, etc.)

Conçu pour etre **utilisé et/ou surchargé**

Quelques fonctionnalités ...

- **BaseCode (MVP)**: contexte objet PHP (extends, implements, use)
- **IHM**: responsive, bootstrap3
- **Générateurs**: bundles, entités & cruds
- **Services**: services SF2 dédiés (surchargeables)
- **Cruds**: Cruds élaborés (formTypes, ajaxGrids)
- **REST**: Compatibilité REST des Cruds
- **Composants**: notifications, dashboard, etc.

Plus d'informations sur la documentation technique FiducialCore

... orientées applis de gestion

- **Sécurité**: authentification (FOSUser)
- **Gestion utilisateurs**: reservé admins
- **Dashboard**: ensemble de widgets pour homepage
- **Notifications**: widget dashboard, Crud d'administration
- **Grids**: grilles ajax complexes (filtres, row/mass actions, exports)
- **Menus**: gestion simple des menus de l'application
- **Search**: moteur de recherche embarqué

BaseCode: contexte objet PHP

Code capitalisé, accélérateur/cadrage de developpements

- **Base controlleurs**: Crud et CrudREST
- **Base models**: Modèles d'entités et repositories Doctrine prêts à l'emploi
- **Interfaces**: pour cadrage comportemental des entités dans le Core
- **Traits**: pour (ré)utilisation de code fonctionnel
- **Annotations**: pour utilisation simple de fonctionnalités du Core

```
use Fiducial\CoreBundle\Annotations\Entity as CoreEntity;
use Fiducial\CoreBundle\Entity\Traits\TimestampableTrait;
use Fiducial\CoreBundle\Entity\Interfaces\ListableEntityInterface;

/**
 * Article.
 * @CoreEntity\Searchable(fields={"code","label"})
 */
class Article implements ListableEntityInterface
{
    use TimestampableTrait;

    //...
```

Générateurs

```
php app/console fiducialcore:generate:bundle
```

- **Compatibles FiducialCore**: préparation des layouts compatibles
- **Menus**: pour entrées dans le menu de l'application

```
php app/console fiducialcore:generate:entity
```

- **TimeStampable**: pour entités avec dates création, modification
- **SoftDeletable**: pour entités avec suppression logique
- **Searchable**: pour entités éligibles au moteur de recherche

```
php app/console fiducialcore:generate:crud
```

- **REST**: pour exposition de routes REST du Crud
- **IHM**: grids & forms responsive (full bootstrap)
- **Spécifiques**: datepickers, grids softDeletable, massActions, etc.

Merci