Projet Lambda: Guide du développeur

Installation, utilisation, maintenance et développement

But du projet

Le but du projet est de mettre à disposition des utilisateurs une plate-forme de partage et d'échange d'objets divers. Il est souhaité de développer ce projet sous framework Symfony.

Installation

Pré-requis : Cette installation suppose que votre serveur soit correctement configuré, PHP y compris. Il n'est pas vraiment nécessaire que Php.exe fasse partie des variables d'environnement du système, toutefois ce sera plus confortable si c'est le cas. Symfony peut dans certains cas nécessiter que la section 'Locale' de Php.ini soit renseignée, et nécessitera un certificat spécifique. Vous pouvez trouver la démarche d'installation du certificat dans la documentation Symfony, ainsi que le certificat lui-même à cette adresse :

https://symfony.com/doc/current/setup.html

Afin de pouvoir développer le projet, et aussi de le tester, vous avez besoin d'un environnement de développement stable.

C'est pourquoi je recommande l'utilisation de l'IDE NetBeans. Contrairement aux autres IDE, NetBeans vous permet d'avoir accès au menu de commande de Symfony, et aussi au menu de commande de Composer. Le lancement et le développement de ce projet nécessitera ces deux menus.

En préambule au lancement de ce projet, il vous faudra importer la base de données du projet. Commencez donc par télécharger le fichier 'Lambdadef.sql' à la racine du repository GitHub du projet. Ensuite il vous faudra vous rendre dans l'interface de management des bases de données de votre serveur (par exemple tout simplement PHPMyAdmin), et créer une nouvelle base de données. Vous pouvez l'appeler comme vous le souhaitez, il faudra simplement se rappeler de ce nom afin de le configurer dans les paramètres de Symfony, qui aura besoin de s'y connecter. Il vous suffira ensuite de sélectionner votre nouvelle base de données, et d'importer le fichier SQL.

Il faut ensuite vous assurer que NetBeans est bien configuré pour utiliser Composer et Symfony. Il suffit pour cela de vérifier dans les options de Netbeans > section PHP > onglet 'Frameworks and tools' respectivement dans 'Composer' et 'Symfony 2/3', que NetBeans connait l'emplacement des fichiers .PHAR : un pour Composer, un pour Symfony.

Si ce n'est pas le cas vous pouvez les télécharger respectivement aux adresses suivantes :

https://getcomposer.org/download/1.4.1/composer.phar

https://symfony.com/installer

Importation du projet :

Sous NetBeans, faites simplement un nouveau projet PHP, sans Symfony, sans rien. Importez ensuite le projet depuis GitHub (clic droit > git) en entrant cette adresse :

https://github.com/wilfalken/Projet-lambda.git

Vous devriez avoir un ou deux conflits, avec les fichiers 'ProjectProperties.xml'. Nous n'en aurons pas réellement besoin, vous pouvez les écraser ou les ignorer.

Afin de récupérer le répertoire 'vendors' de Symfony, qui n'est jamais inclut dans les Git de projets Symfony, vous aurez besoin de Composer. Faites donc un clic-droit sur le projet, Composer > install-dev. Composer va aller chercher toutes les dépendances du projet (Symfony, l'ORM Doctrine, Twig, etc.).

Il reste une dernière chose à faire : se rendre dans le projet, dans le fichier app/config/parameters.yml et remplacer le nom de base de données par celui de la base que vous avez créée précédemment. >> Le projet est prêt à être exécuté !

Il suffit pour cela de se rendre dans les commandes Symfony (Clic-droit > Symfony > Run command), ou d'exécuter le petit fichier Batch se trouvant sur GitHub au nom : Server run.bat

Le modifier au besoin si PHP.EXE ne fait pas partie des variables d'environnement du système, ainsi que la racine d'exécution du projet.

Organisation du projet

Le projet Lambda a été développé sous framework Symfony.

Il est au modèle MVC.

Il est composé de trois Bundles :

- LambdaBundle (bundle principal)

- BaseBundle

- AdminBundle

Le Bundle LambdaBundle est le bundle principal. Il regroupe donc, contrairement aux deux autres : des vus spécifiques, notamment l'index; et regroupe également les entités (classes modèles) dans le répertoire 'Entity'.

Routage

Niveaux

Le routage du projet s'effectue en plusieurs niveaux.

- un niveau primaire, qui est le routage de l'application. Ce routage importe le routage des bundles, notamment celui de LambdaBundle.

- un niveau secondaire, situé dans les bundles eux-mêmes.

Dossiers / fichiers

Les fichiers de routage sont des fichiers au format d'écriture YAML dont le nom est toujours 'routing.yml'

Le routage de niveau primaire est situé dans le répertoire applicatif, soit :
/Source Files/app/config/routing.yml

le routage secondaire s'effectue dans chaque bundle, dans le dossier :
/Source Files/src/Lambda/Nomdubundle/Resources/config/routing.yml

Nomenclature

Le routage s'effectue en associant une route (adresse) à la fonction d'un contrôleur spécifique, suivant cette syntaxe :

```
login:
    path: /login
    defaults: { _controller: LambdaBundle:Security:login }
```

Dans cette exemple, l'adresse http://%serveur%/login pointe vers la fonction 'loginAction' du contrôleur SecurityController, du bundle LambdaBundle.

Peu importe qu'un bouton pointe sur cette route à l'aide de {{ path('login') }} ou bien qu'un utilisateur tape directement l'adresse /login dans son navigateur, cette fonction du contrôleur sera appellée.

Entités

Les entités sont donc stockées dans :

Source Files/src/Lambda/LambdaBundle/Entity

Exploitées par l'ORM Doctrine 2.0, le format de déclaration du mapping en base de données de ces entités qui a été choisi est 'annotations'.

Ce format sert à déclarer directement les informations de mapping dans les classes entités, qui deviennent moins lisibles, mais permettent de s'affranchir de chercher / basculer entre un nombre beaucoup plus important de fichiers.

//extrait de l'entité 'User'

```
/**
 * @var \Doctrine\Common\Collections\Collection
 * inverse side
 * @ORM\ManyToMany(targetEntity="Lambda\LambdaBundle\Entity\Adresse",
 *                         mappedBy="users")
 * @ORM\OrderBy({"principale" = "DESC"})
 */
private $adresses;
```

D'autre part, le format annotation permet également de déclarer les validations champ, utilisées lors de la validation de formulaires, directement dans les entités également :

```
        //extrait de l'entité 'evenement'
/**
 * @var string
 *
 * @ORM\Column(name="lienImage", type="string", length=150, nullable=false)
 *
 * @Assert\Image(
 *     minWidth = 600,
 *     maxWidth = 1000,
 *     minHeight = 400,
 *     maxHeight = 600
 * )
 */
private $lienimage;
```

Dans cet exemple, et dans le cas d'un champ upload d'un formulaire, ciblant l'attribut $lienimage de l'entité évènement, si l'image n'est pas au moins large de 600px et au plus 1000px, le champ ne sera pas validé, et donc le formulaire sera rejeté, avec un message d'erreur. De la même façon, ce champ est obligatoire ('nullable = false')

Contrôleurs

Les contrôleurs Symfony doivent tous étendre Controller, et leur nom doit toujours être suffixé de 'Controller' :

```
class SecurityController extends Controller { ... }
```

De la même façon, les noms de leurs fonctions, afin d'être reconnues en tant que route, doivent tous être suffixés de 'Action':

```
public function loginAction(Request $request) { ... }
```

Méthodes de récupération particulières :

Concernant certaines variables particulières, il n'est pas nécessaire de les envoyer au contrôleur volontairement ou au travers d'une requête.

Par exemple, pour récupérer l'utilisateur connecté à l'application directement dans les paramètres du contrôleur :

```
public function newAction(Request $request, UserInterface $user) { ... }
```

En utilisant l’objet UserInterface $user, il est donc possible d'utiliser directement cet utilisateur.

Formulaires

Les formulaires se trouvent dans tous les bundles, dans le dossier /form.

Ceux du Bundle LambdaBundle se trouvent donc dans Source Files/src/Lambda/LambdaBundle/form.

Les formulaires sont toujours appelés et gérés dans la même fonction contrôleur.

La fonction du contrôleur les appelle et gère les actions effectuées après leur validation.

Annexes

Voici ce que vous devriez pouvoir lire lorsque Composer installera les dépendances du projet :

```
"C:\wamp64\bin\php\php5.6.25\php.exe"
"C:\wamp64\bin\php\php5.6.25\composer.phar" "--ansi" "--no-interaction"
"install"
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 33 installs, 0 updates, 0 removals
  - Installing doctrine/lexer (v1.0.1): Loading from cache
  - Installing doctrine/annotations (v1.2.7): Loading from cache
  - Installing twig/twig (v1.31.0): Loading from cache
  - Installing symfony/polyfill-util (v1.3.0): Loading from cache
  - Installing paragonie/random_compat (v2.0.4): Loading from cache
  - Installing symfony/polyfill-php70 (v1.3.0): Loading from cache
  - Installing symfony/polyfill-php56 (v1.3.0): Loading from cache
  - Installing symfony/polyfill-mbstring (v1.3.0): Loading from cache
  - Installing symfony/symfony (v3.2.4): Loading from cache
  - Installing symfony/polyfill-intl-icu (v1.3.0): Loading from cache
  - Installing psr/log (1.0.2): Loading from cache
  - Installing psr/cache (1.0.1): Loading from cache
  - Installing doctrine/inflector (v1.1.0): Loading from cache
  - Installing doctrine/collections (v1.3.0): Loading from cache
  - Installing doctrine/cache (v1.6.1): Loading from cache
  - Installing doctrine/common (v2.6.2): Loading from cache
  - Installing jdorn/sql-formatter (v1.2.17): Loading from cache
  - Installing doctrine/doctrine-cache-bundle (1.3.0): Loading from cache
  - Installing doctrine/dbal (v2.5.12): Loading from cache
  - Installing doctrine/doctrine-bundle (1.6.7): Loading from cache
  - Installing doctrine/instantiator (1.0.5): Loading from cache
  - Installing doctrine/orm (v2.5.6): Loading from cache
  - Installing incenteev/composer-parameter-handler (v2.1.2): Loading from
cache
  - Installing sensiolabs/security-checker (v4.0.0): Loading from cache
  - Installing sensio/distribution-bundle (v5.0.18): Loading from cache
  - Installing sensio/framework-extra-bundle (v3.0.22): Loading from cache
  - Installing monolog/monolog (1.22.0): Loading from cache
  - Installing symfony/monolog-bundle (v3.0.3): Loading from cache
  - Installing symfony/polyfill-apcu (v1.3.0): Loading from cache
  - Installing swiftmailer/swiftmailer (v5.4.6): Loading from cache
  - Installing symfony/swiftmailer-bundle (v2.4.2): Loading from cache
  - Installing sensio/generator-bundle (v3.1.2): Loading from cache
  - Installing symfony/phpunit-bridge (v3.2.4): Loading from cache
paragonie/random_compat suggests installing ext-libsodium (Provides a
modern crypto API that can be used to generate random bytes.)
doctrine/doctrine-cache-bundle suggests installing symfony/security-acl
(For using this bundle to cache ACLs)
sensio/framework-extra-bundle suggests installing symfony/psr-http-message-
bridge (To use the PSR-7 converters)
monolog/monolog suggests installing aws/aws-sdk-php (Allow sending log
messages to AWS services like DynamoDB)
monolog/monolog suggests installing doctrine/couchdb (Allow sending log
messages to a CouchDB server)
monolog/monolog suggests installing ext-amqp (Allow sending log messages to
an AMQP server (1.0+ required))
monolog/monolog suggests installing ext-mongo (Allow sending log messages
to a MongoDB server)
monolog/monolog suggests installing graylog2/gelf-php (Allow sending log
messages to a GrayLog2 server)
```

```
monolog/monolog suggests installing mongodb/mongodb (Allow sending log
messages to a MongoDB server via PHP Driver)
monolog/monolog suggests installing php-amqplib/php-amqplib (Allow sending
log messages to an AMQP server using php-amqplib)
monolog/monolog suggests installing php-console/php-console (Allow sending
log messages to Google Chrome)
monolog/monolog suggests installing rollbar/rollbar (Allow sending log
messages to Rollbar)
monolog/monolog suggests installing ruflin/elastica (Allow sending log
messages to an Elastic Search server)
monolog/monolog suggests installing sentry/sentry (Allow sending log
messages to a Sentry server)
Generating autoload files
> Incenteev\ParameterHandler\ScriptHandler::buildParameters
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::buildBootstrap
Creating the "app/config/parameters.yml" file
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::clearCache

 // Clearing the cache for the dev environment with debug true


 [OK] Cache for the "dev" environment (debug=true) was successfully
cleared.


> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installAssets
>
Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installRequirement
sFile
>
Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::prepareDeploymentT
arget

 Trying to install assets as relative symbolic links.

 -- -------- ----------------
     Bundle   Method / Error
 -- -------- ----------------


 [OK] All assets were successfully installed.


Done.
```