PHP - Symfony

Avant-propos

Avant de commencer, il est important de savoir que Symfony propose régulièrement différentes façons de coder votre projet, selon les goûts, la pertinence et/ou vos besoins. Nous verrons ici une façon de faire mais garder toujours à l'esprit qu'il existe bien souvent une alternative à cette façon de faire.

Pour plus de précision, n'hésitez pas à consulter la documentation de Symfony.

De manière générale quand on travaille avec un framework, on utilise systématiquement ?

...

LA DOCUMENTATION !

Qu'est-ce que c'est ?

Symfony est un framework PHP libre, regroupant un ensemble de composant (bundles) et suivant une architecture MVC. Il fournit des fonctionnalités modulables et adaptable qui permettent de faciliter et d'accélérer le développement d'un site web.

La p'tite histoire

L'agence web française SensioLabs est à l'origine du framework Sensio Framework. À force de toujours créer les mêmes fonctionnalités de gestion d'utilisateur, gestion d'ORM, etc., elle a développé ce framework pour ces propres besoins.

Comme ces problématiques étaient souvent les mêmes pour d'autres développeurs, le code a été partagé avec la communauté des développeurs PHP. Le projet est alors devenu Symfony (conformément à la volonté du créateur de conserver les initiales S et F de Sensio Framework), puis symfony à partir de la version 2. Cette dernière casse la compatibilité avec la branche 1.x.

Actuellement, la version LTS (Long Term Support) est la version 4.4.7, et la dernière version stable est la 5.0.7.

Comment ça marche ?

Installation de Composer

Pour initialiser un projet Symfony, nous devons installer dans un premier temps PHP (ou un serveur local type Wampserver) et un gestionnaire de paquets, **Composer**.

Suivez les démarches dans ce tuto.

Installation de Symfony

Une fois notre serveur local et Composer installés, nous pouvons enfin initialiser notre projet Symfony. Pour cela, nous utiliserons les lignes de commandes (soit gitbash, soit cmd).

Suivez les différentes étapes de ce tuto pour initialiser votre premier projet Symfony

Une fois votre projet Symfony installé, nous allons enfin pouvoir créer une première vue !!

Créer une vue avec Symfony

Créer une page nécessite 3 étapes :

- Créer un contrôleur : un contrôleur est la fonction PHP écrite par le développeur en vue d'élaborer la page ; elle exploite les informations de la Request entrante et les utilise pour créer un objet Symfony de type Response pouvant contenir du HTML, une chaîne JSON ou même un fichier binaire, tel qu'une image ou un PDF.
- Créer une route : une route associe l'URL (par exemple /apropos) d'une page et un contrôleur ;
- Créer un template : le template est la vue que le contrôleur affichera à l'utilisateur

Le contrôleur

Les Contrôleurs ont pour rôle de gérer chaque requête adressée à l'application Symfony, et dans la plupart des cas, de rendre un template (vue) qui va élaborer le contenu de la réponse au client.

Un contrôleur est une fonction PHP que vous créez, qui lit les informations de l'objet Request (requêtes du client envoyées vers le site), crée et retourne un objet Response. La réponse peut être une page HTML, JSON, XML, un téléchargement de fichier, une redirection, une erreur 404 , ... Le contrôleur exécute le traitement métier dont l'application a besoin pour restituer le contenu d'une page.

Le routage

Une route est une map (correspondance) entre URL et une fonction. Cette route peut être configurée de plusieurs manières :

- avec un fichier XML
- avec une classe PHP
- avec un fichier de configuration .yaml
- avec des annotations

Tout dépend comment vous comptez gérer votre projet, mais de manière générale, les annotations restent plus pratiques comme solution : la route est affichée directement au-dessus de la méthode, ce qui permet une meilleure visibilité des informations dont vous pourriez avoir besoin.

Commençons par créer notre contrôleur en suivant cette ressource

Le template

Pour afficher des informations à l'utilisateur, Symfony utilise un moteur de template appelé Twig. Il a la particularité de récupérer facilement les données transmises par le contrôleur, et intègre la notion d'héritage sur les templates.

Sa structure est très semblable au HTML, les fichiers templates auront une extension .html.twig.

Pour créer notre première vue avec Symfony, suivez ce tuto.

En suivant cette ressource, vous verrez comment faire des liens entre les différentes vues et contrôleurs de votre application Symfony, ainsi que le passage de données du contrôleur vers une vue.

Maintenant que nous savons afficher du contenu sur une vue, et naviguer d'une page vers une autre, voyons la connexion à une basse de données.

Base de données et Doctrine

Pour accéder et manipuler une base de données avec Symfony, Nous allons utiliser un ORM (Object-relational mapping), Doctrine.

Un ORM est un type de programme qui se place en interface entre une application (notre site par exemple) et une base de données relationnelle pour simuler une base de données orientés objet. Ce programme définit les correspondances entre les schémas de la base de données et les classes du programme applicatif. L'association objets-tables est appelée `mapping`. Les classes sont associées à une table et chaque propriété de la classe est associé à un champ de la table. Avec `Doctrine`, cette opération est réalisée grâce à des annotations.

Voyons maintenant la connexion à la base de données. (méthode 1).

Une autre façon de se connecter à la base de données avec Doctrine à suivre par là.

Les formulaires et leurs contrôles sous Symfony

Lors de la génération du CRUD dans la partie suivante, nous avons vu qu'un fichier a été créé dans le dossier `Form`. Par convention, tous les formulaires se trouveront dans ce dossier. Ils ont également leur nom se terminant par `Type`

Suivez cette ressource pour apprendre plus sur les formulaires avec Symfony.

Pour les formulaires et base de données, c'est par ici.

L'authentification

La mise en place d'une authentification avec Symfony demande un de paramétrage, mais se fait assez facilement, et encore une fois pas mal de choses sont automatisées.

Pour mettre en place une interface de connexion sur votre projet, suivez cette ressource.

Sécurité avec Symfony : les voters

Si on veut faire en sorte que certains utilisateurs n'aient pas accès à certaines fonctionnalités d'un site / appli, il faudrait développer un code assez lourd et redondant pour faire les différentes vérifications nécessaires, et ce sur toutes les méthodes où il faudrait restreindre l'accès.

Encore une fois, Symfony nous facilite le travail en utilisant le système de voters.

Le voter est un composant de Symfony, dans lequel nous allons définir les différentes les autorisations selon les actions faîtes et les rôles des utilisateurs.

Suivez cette ressource pour mettre en place un voter sur votre site.

Les API avec API-Platform

Une API est une application qui permet de faire communiquer plusieurs systèmes avec une même base de données. On peut également utiliser une API pour faciliter la communication entre plusieurs technos (ici pour l'exemple Symfony avec Vue.js), et ainsi gagner en fluidité et performance sur les actions à effectuer.

Encore une fois, pour la gestion des API, Symfony va nous faciliter le travail.

Suivez cette ressource pour établir une API avec Symfony.

Attention : pour suivre cette partie, il est nécessaire d'avoir fait un crud sur la table `products`, et d'avoir suivi la ressource sur Vue.js.

Les tests unitaires

Pour mettre en place des tests unitaires sur vos applications / sites, cliquez ici

En complément

- Documentation de Symfony
- Tutos de Grafikart (Symfony 4)
- Chaine Youtube de Lior Chamla
- SymfonyCasts