Table des matières

Documentation Symfony

1-Origine

2-Configuration (pré-requis)

3-Création de page

4-Routage

5- Contrôleurs

6-Modèles

7- Configuration(format)

1-Origine

Symfony est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web.

L'agence web française SensioLabs est à l'origine du framework *Sensio Framework*. À force de toujours recréer les mêmes fonctionnalités de gestion d'utilisateurs, gestion ORM, etc., elle a développé ce framework pour ses propres besoins. Comme ces problématiques étaient souvent les mêmes pour d'autres développeurs, le code a été par la suite 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 *Symfony2* à partir de la version 24. La version 2 de *Symfony* casse la compatibilité avec la branche 1.x.

Le 5 septembre 2017, Symfony passe la barre du milliard de téléchargements

2-Configuration

Exigences techniques

Avant de créer votre première application Symfony, vous devez:

- Installez PHP 7.2.5 ou supérieur et ces extensions PHP (qui sont installées et activées par défaut dans la plupart des installations PHP 7): Ctype , iconv , JSON , PCRE , Session , SimpleXML et Tokenizer ;
- Installez Composer , qui est utilisé pour installer les packages PHP.

En option, vous pouvez également installer Symfony CLI . Cela crée un binaire appelé `symfony` qui fournit tous les outils dont vous avez besoin pour développer et exécuter votre application Symfony localement.

Le `symfony` binaire fournit également un outil pour vérifier si votre ordinateur répond à toutes les exigences. Ouvrez votre terminal de console et exécutez cette commande:

3-Création de page

La création d'une nouvelle page - qu'il s'agisse d'une page HTML ou d'un point de terminaison JSON - est un processus en deux étapes:

1. Créer une route : Une route est l'URL (par exemple `/about`) de votre page et pointe vers un contrôleur;
2. Créer un contrôleur : Un contrôleur est la fonction PHP que vous écrivez qui construit la page. Vous prenez les informations de la demande entrante et vous les utilisez pour créer un `Response Object` Symfony , qui peut contenir du contenu HTML, une chaîne JSON ou même un fichier binaire comme une image ou un PDF.

4-Routage

Lorsque votre application reçoit une demande, elle appelle une action de contrôleur pour générer la réponse. La configuration de routage définit l'action à exécuter pour chaque URL entrante. Il fournit également d'autres fonctionnalités utiles, telles que la génération d'URL conviviales pour le référencement (par exemple `/read/intro-to-symfony`au lieu de `index.php?article_id=57`).

Création de routes

Les itinéraires peuvent être configurés en YAML, XML, PHP ou en utilisant des attributs ou des annotations. Tous les formats offrent les mêmes fonctionnalités et performances, alors choisissez votre favori. Symfony recommande des attributs car il est pratique de placer la route et le contrôleur au même endroit.

Création de routes comme attributs ou annotations

Sur PHP 8, vous pouvez utiliser des attributs natifs pour configurer les routes tout de suite. Sur PHP 7, où les attributs ne sont pas disponibles, vous pouvez utiliser des annotations à la place, fournies par la bibliothèque Doctrine Annotations.

Si vous souhaitez utiliser des annotations au lieu d'attributs, exécutez cette commande une fois dans votre application pour les activer: exemple

```
composer require doctrine/annotations
```

Cette commande crée également le fichier de configuration suivant:

5- Contrôleurs

Lorsque votre application reçoit une demande, elle appelle une action de contrôleur pour générer la réponse. La configuration de routage définit l'action à exécuter pour chaque URL entrante. Il fournit également d'autres fonctionnalités utiles, telles que la génération d'URL conviviales pour le référencement (par exemple /read/intro-to-symfonyau lieu de index.php?article_id=57).

Création de routes

Les itinéraires peuvent être configurés en YAML, XML, PHP ou en utilisant des attributs ou des annotations. Tous les formats offrent les mêmes fonctionnalités et performances, alors choisissez votre favori. Symfony recommande des attributs car il est pratique de placer la route et le contrôleur au même endroit.

Création de routes comme attributs ou annotations

Sur PHP 8, vous pouvez utiliser des attributs natifs pour configurer les routes tout de suite. Sur PHP 7, où les attributs ne sont pas disponibles, vous pouvez utiliser des annotations à la place, fournies par la bibliothèque Doctrine Annotations.

6-Modèles

Configuration de Twig

Twig dispose de plusieurs options de configuration pour définir des éléments tels que le format utilisé pour afficher les nombres et les dates, la mise en cache du modèle, etc. Lisez la référence de configuration Twig pour en savoir plus.

Un modèle est le meilleur moyen d'organiser et de rendre du HTML à partir de votre application, que vous ayez besoin de rendre du HTML à partir d'un contrôleur ou de générer le contenu d'un e-mail . Les modèles dans Symfony sont créés avec Twig: un moteur de modèles flexible, rapide et sécurisé.

Twig Templating Language

Le langage de création de modèles Twig vous permet d'écrire des modèles concis et lisibles qui sont plus conviviaux pour les concepteurs Web et, à plusieurs égards, plus puissants que les modèles PHP. Jetez un œil à l'exemple du modèle Twig suivant. Même si c'est la première fois que vous voyez Twig, vous en comprenez probablement l'essentiel:

```twig
<!DOCTYPE html>
<html>
    <head>
        <title>Welcome to Symfony!</title>
    </head>
    <body>
        <h1>{{ page_title }}</h1>

        {% if user.isLoggedIn %}
            Hello {{ user.name }}!
        {% endif %}

        {# ... #}
    </body>
</html>
```

La syntaxe Twig est basée sur ces trois constructions:

- {{ ... }}, utilisé pour afficher le contenu d'une variable ou le résultat de l'évaluation d'une expression;
- {% ... %}, utilisé pour exécuter une logique, telle qu'un conditionnel ou une boucle;
- {# ... #}, utilisé pour ajouter des commentaires au modèle (contrairement aux commentaires HTML, ces commentaires ne sont pas inclus dans la page rendue).

7- Configuration(format)

Contrairement à d'autres frameworks, Symfony ne vous impose pas de format spécifique pour configurer vos applications. Symfony vous permet de choisir entre YAML, XML et PHP et tout au long de la documentation Symfony, tous les exemples de configuration seront affichés dans ces trois formats.

Il n'y a aucune différence pratique entre les formats. En fait, Symfony les transforme et les met en cache tous en PHP avant d'exécuter l'application, il n'y a donc même pas de différence de performances entre eux.

YAML est utilisé par défaut lors de l'installation des packages car il est concis et très lisible. Voici les principaux avantages et inconvénients de chaque format:

- **YAML** : simple, propre et lisible, mais tous les IDE ne prennent pas en charge l'autocomplétion et la validation. Apprenez la syntaxe YAML ;
- **XML** : auto complété / validé par la plupart des IDE et analysé nativement par PHP, mais il génère parfois une configuration jugée trop verbeuse. Apprenez la syntaxe XML ;
- **PHP** : très puissant et il permet de créer une configuration dynamique, mais la configuration résultante est moins lisible que les autres formats.

Source supplémentaire

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