Alexandre Bacco
Préface de Fabien Potencier

DÉVELOPPEZ VOTRE SITE WEB AVEC LE FRAMEWORK

SYMFONY3

OPENCLASSROOMS

EYROLLES

DÉVELOPPEZ VOTRE SITE WEB AVEC LE FRAMEWORK

SYMFONY3

Vous développez des sites web régulièrement et vous en avez assez de réinventer la roue ? Vous aimeriez utiliser les bonnes pratiques de développement PHP pour concevoir des sites de qualité professionnelle ? Cet ouvrage vous permettra de prendre en main Symfony, le framework PHP de référence. Comment créer un nouveau projet avec Symfony, mettre en place les environnements de test et de production, concevoir les contrôleurs, les templates, gérer la traduction et communiquer avec une base de données via Doctrine ? Vous découvrirez comment ce puissant framework, supporté par une large communauté, va vous faire gagner en efficacité.

QU'ALLEZ-VOUS APPRENDRE ?

Vue d'ensemble de Symfony
- Symfony, un framework PHP
- Vous avez dit Symfony ?
- Utiliser la console pour créer un bundle

Les bases de Symfony
- Mon premier « Hello World ! » avec Symfony
- Le routeur de Symfony
- Les contrôleurs avec Symfony
- Le moteur de templates Twig
- Installer un bundle grâce à Composer
- Les services, théorie et création

Gérer la base de données avec Doctrine2
- La couche métier : les entités
- Manipuler ses entités avec Doctrine2
- Les relations entre entités avec Doctrine2
- Récupérer ses entités avec Doctrine2
- Les événements et extensions Doctrine
- TP : consolidation de notre code

Aller plus loin avec Symfony
- Créer des formulaires avec Symfony
- Valider ses données
- Sécurité et gestion des utilisateurs
- Les services, fonctions avancées
- Le gestionnaire d'événements de Symfony
- Traduire son site

Préparer la mise en ligne
- Convertir les paramètres de requêtes
- Personnaliser les pages d'erreur
- Utiliser Assetic pour gérer les codes CSS et JS
- Utiliser la console depuis le navigateur
- Déployer son site Symfony en production

À PROPOS DE L'AUTEUR

Passionné de développement web, Alexandre Bacco participe à la création de la version 3 d'OpenClassrooms durant ses études. Diplômé de l'École Centrale de Lyon, une école d'ingénieur généraliste, il tombe sous le charme du framework Symfony avant même sa sortie et décide de partager ses connaissances en rédigeant un cours sur OpenClassrooms et pour les éditions Eyrolles.

L'ESPRIT D'OPENCLASSROOMS

Des cours ouverts, riches et vivants, conçus pour tous les niveaux et accessibles à tous gratuitement sur notre plate-forme d'e-éducation : www.openclassrooms.com. Vous y vivrez une véritable expérience communautaire de l'apprentissage, permettant à chacun d'apprendre avec le soutien et l'aide des autres étudiants sur les forums. Vous profiterez des cours disponibles partout, tout le temps : sur le Web, en PDF, en eBook, en vidéo…

www.editions-eyrolles.com

DÉVELOPPEZ VOTRE SITE WEB AVEC LE FRAMEWORK

SYMFONY3

DANS LA MÊME COLLECTION

M. Chavelli. – **Découvrez le framework PHP Laravel.**
N°14398, 2016, 336 pages.

R. De Visscher. – **Découvrez le langage Swift.**
N°14397, 2016, 128 pages.

M. Lorant. – **Développez votre site web avec le framework Django.**
N°21626, 2015, 285 pages.

E. Lalitte. – **Apprenez le fonctionnement des réseaux TCP/IP.**
N°21623, 2015, 300 pages.

M. Nebra, M. Schaller. – **Programmez avec le langage C++.**
N°21622, 2015, 674 pages.

SUR LE MÊME THÈME

P. Martin, J. Pauli, C. Pierre de Geyer, É. Daspet. – **PHP 7 avancé.**
N°14357, 2016, 732 pages.

R. Goetter. – **CSS 3 Flexbox.**
N°14363, 2016, 152 pages.

W. McKinney. – **Analyse de données en Python.**
N°14109, 2015, 488 pages.

E. Biernat, M. Lutz. – **Data science : fondamentaux et études de cas.**
N°14243, 2015, 312 pages.

B. Philibert. – **Bootstrap 3 : le framework 100 % web design.**
N°14132, 2015, 318 pages.

C. Camin. – **Développer avec Symfony2.**
N°14131, 2015, 474 pages.

S. Pittion, B. Siebman. – **Applications mobiles avec Cordova et PhoneGap.**
N°14052, 2015, 184 pages.

C. Delannoy. – **Le guide complet du langage C.**
N°14012, 2014, 844 pages.

Alexandre Bacco
Préface de Fabien Potencier

DÉVELOPPEZ VOTRE SITE WEB AVEC LE FRAMEWORK

SYMFONY3

OPENCLASSROOMS

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

© Groupe Eyrolles, 2016. ISBN Eyrolles : 978-2-212-14403-1

Préface

Pendant longtemps, PHP a été décrié et critiqué par les « développeurs professionnels » pour son côté rustique et simpliste : un langage pour les « développeurs du dimanche ». Pourtant, en dépit de ces critiques et de cette image maintenant datée, PHP est un langage qui a su évoluer, se structurer, se professionnaliser. Tant et si bien que c'est aujourd'hui de loin le langage dominant du Web. À lui seul, PHP motorise près de 70 % des sites web dans le monde. De nombreux sites à très fortes audiences que vous consultez régulièrement sont motorisés par PHP.

En 2005, je dirigeais Sensio, une agence web parisienne créée sept ans plus tôt avec mon associé Grégory Pascal. Pour professionnaliser nos méthodes de travail et capitaliser sur notre savoir-faire, je décidais de créer un framework, d'abord réservé à nos usages internes. La version 5 de PHP venait de sortir, proposant les premiers outils PHP réellement destinés aux professionnels : Mojavi, Propel, PHPUnit... C'est donc sur PHP que nous avons concentré nos efforts.

Assez rapidement, je mettais à disposition de tous les développeurs intéressés notre travail en licence open source. Symfony était né.

En 2011, nous avons lancé Symfony2 et franchit une nouvelle étape. Le succès fut phénoménal et l'adoption dans le monde entier n'a fait que croître depuis : chaque mois, Symfony est téléchargé plus d'un million de fois sur le site symfony.com et nous estimons que près de 300 000 développeurs dans le monde utilisent cette technologie.

Pourquoi un tel succès ?

Tour d'abord, parce que tous ceux qui contribuent à Symfony sont animés par une forte culture open source où chacun met à disposition de tous le fruit de son travail. Le projet a d'abord attiré des dizaines puis des centaines de développeurs qui ont progressivement à faire de Symfony le framework de choix pour les développeurs professionnels.

Ensuite, parce que Symfony est un projet très dynamique qui évolue très régulièrement pour accompagner les évolutions du Web et les demandes croissantes des utilisateurs.

Enfin, parce que la structure originale de Symfony – un framework mais aussi des composants autonomes – a séduit de nombreux projets open source d'importance (Drupal, EZ Publish, PhpBB, etc.) et les a conduits à asseoir leur développement sur le projet Symfony. La version 8 de Drupal par exemple intègre plus de 10 composants essentiels de Symfony. Cette large adoption par d'autres projets open source, mais aussi par de nombreux projets commerciaux a permis de crédibiliser et de populariser plus encore le framework.

Et vous dans tout cela ?

En décidant d'acheter et de lire ce livre, vous faites probablement vos premiers pas dans une technologie mais aussi une communauté unique. Dans les mois à venir, peut-être utiliserez-vous Symfony pour développer des projets pour des clients, aurez-vous besoin de consulter de la documentation, d'échanger avec d'autres utilisateurs, vous retrouverez-vous lors d'événements annuels (les Symfony Live) pour échanger avec vos pairs ? Quels que soient vos besoins, le site symfony.com vous offrira les ressources nécessaires.

Et puis, avec la pratique et l'expérience, j'espère que vous rejoindrez un jour les contributeurs dévoués qui font chaque jour le succès de Symfony.

D'ici là, je vous souhaite une excellente lecture !

Fabien Potencier
Créateur de Symfony et président de SensioLabs

Table des matières

Introduction **1**

Première partie – Vue d'ensemble de Symfony 3

1 Symfony, un framework PHP **5**

Qu'est-ce qu'un framework ? 6
L'objectif d'un framework 6
Définition 6
Objectif d'un framework 6
Pesons le pour et le contre 7
Alors, convaincus ? 8

Qu'est-ce que Symfony ? 8
Un framework 8
Un framework populaire 8
Un framework populaire et français 9
Qu'est-il possible de faire avec Symfony ? 9

Télécharger Symfony 10
Vérifier l'installation de PHP en console 10
Obtenir Symfony 11
Droits d'accès 13

En résumé 14

2 Vous avez dit Symfony ? **15**

L'architecture des fichiers 15
Liste des répertoires 15
Le répertoire /app 16

Le répertoire /bin . . . 16
Le répertoire /src . . . 16
Le répertoire /tests . . . 16
Le répertoire /var . . . 16
Le répertoire /vendor . . . 16
Le répertoire /web . . . 17
À retenir . . . 17
Le contrôleur frontal . . . 17

L'architecture conceptuelle . . . 20
Architecture MVC . . . 20
Parcours d'une requête dans Symfony . . . 21

Symfony et ses bundles . . . 23
La découpe en bundles . . . 23
L'intérêt . . . 23
La bonne pratique . . . 24
Les bundles de la communauté . . . 24
La structure d'un bundle . . . 25

En résumé . . . 25

3 Utilisons la console pour créer un bundle 27

Utilisation de la console . . . 27
Sous Windows . . . 27
Sous Linux et Mac . . . 28
À quoi cela sert-il ? . . . 28
Comment cela marche-t-il ? . . . 29

Le fil rouge de notre cours : une plate-forme d'échange . . . 30

Créons notre bundle . . . 30
Tout est bundle . . . 30
Exécuter la bonne commande . . . 30
Que s'est-il passé ? . . . 33
Pour conclure . . . 36

En résumé . . . 36

Deuxième partie – Les bases de Symfony 37

4 Mon premier « Hello World ! » avec Symfony 39

Créer sa route . . . 39
Quel est le rôle du routeur ? . . . 39
Créer son fichier de routes . . . 40
Informer Symfony que nous avons des routes pour lui . . . 41

Créer son contrôleur . . . 41
Quel est le rôle du contrôleur ? . . . 41
Créer Notre contrôleur . . . 42

Créer son template Twig 43
Les templates avec Twig 43
Utiliser Twig avec Symfony 44
L'objectif : créer une plate-forme d'annonces 47
Un peu de nettoyage 48
Schéma de développement sous Symfony 48
Pour conclure 49
En résumé 50

5 Le routeur de Symfony 51

Le fonctionnement 51
Fonctionnement du routeur 52
Convention pour le nom du contrôleur 54
Les routes de base 54
Créer une route 54
Créer une route avec des paramètres 55
Les routes avancées 57
Créer une route avec des paramètres et leurs contraintes 57
Utiliser des paramètres facultatifs 58
Utiliser des « paramètres système » 59
Ajouter un préfixe lors de l'import de nos routes 60
Générer des URL 61
Pourquoi générer des URL ? 61
Comment générer des URL ? 61
Application : les routes de notre plate-forme 63
Page d'accueil 63
Page de visualisation d'une annonce 63
Ajout, modification et suppression 64
Récapitulatif 64
Pour conclure 65
En résumé 66

6 Les contrôleurs avec Symfony 67

Le rôle du contrôleur 67
Retourner une réponse 67
Manipuler l'objet Request 69
Les paramètres de la requête 69
Les autres méthodes de l'objet Request 72
Savoir si la requête est une requête Ajax 72
Manipuler l'objet Response 73
Décomposition de la construction d'un objet Response 73
Réponses et vues 73

Réponse et redirection . . . 75
Changer le Content-type de la réponse . . . 77
Manipuler la session . . . 78
Application : le contrôleur de notre plate-forme . . . 81
À retenir . . . 83
L'erreur 404 . . . 83
La définition des méthodes . . . 83
Tester les types d'erreurs . . . 84
Pour conclure . . . 85
En résumé . . . 85

7 Le moteur de templates Twig 87

Les templates Twig . . . 87
Intérêt . . . 87
Des pages web, mais aussi des e-mails et autres . . . 88
En pratique . . . 88
À savoir . . . 89
Afficher des variables . . . 89
Syntaxe élémentaire pour afficher des variables . . . 89
Précisions sur la syntaxe {{ objet.attribut }} . . . 90
Les filtres utiles . . . 90
Twig et la sécurité . . . 91
Les variables globales . . . 92
Structures de contrôle et expressions . . . 93
Les structures de contrôle . . . 93
Les tests utiles . . . 95
Hériter et inclure des templates . . . 96
L'héritage de template . . . 96
L'inclusion de templates . . . 100
L'inclusion de contrôleurs . . . 101
Application : les templates de notre plate-forme . . . 103
Layout général . . . 104
Layout du bundle . . . 106
Les templates finaux . . . 106
Pour conclure . . . 113
En résumé . . . 114

8 Installer un bundle grâce à Composer 115

Composer, qu'est-ce que c'est ? . . . 115
Un gestionnaire de dépendances . . . 115
Comment Composer sait-il où trouver les bibliothèques ? . . . 116
Un outil innovant… dans l'écosystème PHP . . . 116
Concrètement, comment fonctionne Composer ? . . . 116

Installer Composer et Git 116
Installer Composer 116
Installer Git 117

Installer un bundle grâce à Composer 118
Manipulons Composer 118
Mettons à jour Symfony 120
Installer un bundle avec Composer 121
Gérer manuellement l'autoload d'une bibliothèque 123

Pour conclure 124

En résumé 124

9 Les services, théorie et création 125

Pourquoi utiliser des services ? 125
Genèse 125
Qu'est-ce qu'un service ? 126
L'avantage de la programmation orientée services 126
Le conteneur de services 126
Comment définir les dépendances entre services ? 129
Le partage des services 129

Utiliser un service en pratique 130

Créer un service simple 131
Créer la classe du service 131
Configurer le service 132
Utiliser le service 134

Créer un service avec des arguments 135
Injecter des arguments dans nos services 135
Injecter des dépendances 137
Aperçu du code 137

Pour conclure 138

En résumé 139

Troisième partie – Gérer la base de données avec Doctrine2 141

10 La couche métier : les entités 143

Notions d'ORM : utiliser des objets à la place des requêtes 143

Créer une première entité avec Doctrine2 144
Une entité, c'est juste un objet 144
Une entité, c'est juste un objet… mais avec des commentaires ! 145
Créer une entité : le générateur à la rescousse ! 147
Affiner notre entité avec de la logique métier 149
À retenir 150

Tout sur le mapping ! 151

L'annotation Entity 151
L'annotation Table 151
L'annotation Column 152

Pour conclure 154

En résumé 155

11 Manipuler ses entités avec Doctrine2 **157**

Matérialiser les tables en base de données 157
Créer la table correspondante dans la base de données 157
Modifier une entité 159
À retenir 160

Utiliser le gestionnaire d'entités 161
Les services Doctrine2 161
Les repositories 162
Enregistrer ses entités en base de données 164
Doctrine utilise les transactions 166
Doctrine simplifie la vie 166
Les autres méthodes utiles du gestionnaire d'entités 167

Récupérer ses entités avec un repository 168

En résumé 170

12 Les relations entre entités avec Doctrine2 **171**

Notions de base sur les relations 171
Entité propriétaire et entité inverse 171
Relations unidirectionnelle et bidirectionnelle 172
Relations et requêtes 172

Relation One-To-One 173
Présentation 173
Définir la relation dans les entités 174
Exemple d'utilisation 178

Relation Many-To-One 180
Présentation 180
Définir la relation dans les entités 183
Exemple d'utilisation 185

Relation Many-To-Many 187
Présentation 187
Définir la relation dans les entités 188
Remplir la base de données avec les fixtures 191
Exemples d'utilisation 192

Relation Many-To-Many avec attributs 197
Présentation 197
Définir la relation dans les entités 198
Remplir la base de données 201
Exemple d'utilisation 202

Les relations bidirectionnelles 206
Présentation 206
Définir la relation dans les entités 206

Pour conclure 211

En résumé 211

13 Récupérer ses entités avec Doctrine2 213

Le rôle des repositories 213
Définition 213
Construire ses requêtes pour récupérer des entités 214

Les méthodes de récupération de base 215
Définition 215
Les méthodes classiques 215
Les méthodes magiques 217

Les méthodes personnelles de récupération 218
La théorie 218
Le QueryBuilder 218
La Query 223
Utiliser le Doctrine Query Language (DQL) 226

Utiliser les jointures dans les requêtes 228
Pourquoi utiliser les jointures ? 228
Comment faire des jointures avec le QueryBuilder ? 228
Comment utiliser les jointures ? 230

Application : les repositories de notre plate-forme d'annonces 231
Plan d'attaque 231
À vous de jouer ! 231
La correction 232

En résumé 233

14 Les événements et extensions Doctrine 235

Les événements Doctrine 235
L'intérêt des événements Doctrine 235
Définir des callbacks de cycle de vie 236
Liste des événements de cycle de vie 238
Un autre exemple d'utilisation 239
Utiliser des services pour écouter les événements Doctrine 241
Essayons nos événements 244

Les extensions Doctrine 246
L'intérêt des extensions Doctrine 246
Installer le StofDoctrineExtensionBundle 246
Utiliser une extension : l'exemple de Sluggable 247
Liste des extensions Doctrine 249

Pour conclure 250

En résumé 250

15 TP : consolidation de notre code 251

Synthèse des entités . . . 251
Entité Advert . . . 251
Entité Image . . . 257
Entité Application . . . 258
Entité Category . . . 260
Entités Skill et AdvertSkill . . . 261
Et bien sûr... . . . 263

Adaptation du contrôleur . . . 263
Théorie . . . 263
Pratique . . . 264

Utiliser des jointures . . . 267

Paginer des annonces sur la page d'accueil . . . 269

Pour conclure . . . 272

En résumé . . . 273

Quatrième partie – Aller plus loin avec Symfony 275

16 Créer des formulaires avec Symfony 277

Gérer des formulaires . . . 277
L'enjeu des formulaires . . . 277
Qu'est-ce qu'un formulaire Symfony ? . . . 278
Gérer simplement un formulaire . . . 279
Ajouter des champs . . . 282
Gérer de la soumission d'un formulaire . . . 284
Gérer les valeurs par défaut du formulaire . . . 287
Personnaliser l'affichage d'un formulaire . . . 288
Créer des types de champs personnalisés . . . 291

Externaliser la définition de ses formulaires . . . 291
Définir le formulaire dans AdvertType . . . 291
Le contrôleur épuré . . . 292

Les formulaires imbriqués . . . 294
Intérêt de l'imbrication . . . 294
Un formulaire est un champ . . . 294
Relation simple : imbriquer un seul formulaire . . . 295
Relation multiple : imbriquer un même formulaire plusieurs fois . . . 297
Un type de champ très utile : entity . . . 303
L'option query_builder . . . 305

Aller plus loin avec les formulaires . . . 307
L'héritage de formulaire . . . 307
À retenir . . . 308
Varier la méthode de construction d'un formulaire . . . 308

Envoyer des fichiers avec le type de champ File . . . 311

Le type de champ File 311
Préparer l'objet sous-jacent 311
Adapter le formulaire 312
Manipuler le fichier envoyé 313
Automatiser le traitement grâce aux événements 315

Application : les formulaires de notre site 320
Théorie 320
Pratique 320

Pour conclure 326

En résumé 326

17 Valider ses données **327**

Pourquoi valider des données ? 327
Toujours se méfier des données de l'utilisateur 327
L'intérêt de la validation 327
La théorie de la validation 328

Définir les règles de validation 328
Les différents formats de règles 328

Déclencher la validation 334
Le service validator 334
La validation automatique sur les formulaires 335

Encore plus de règles de validation 336
Valider depuis un accesseur 336
Valider intelligemment un attribut objet 337
Valider depuis un Callback 338
Valider un champ unique 339

Valider selon nos propres contraintes 340
Créer la contrainte 341
Créer le validateur 342
Transformer son validateur en service 344
Définition du service 344
Modifier la contrainte 345
Modifier du validateur 345

Pour conclure 347

En résumé 347

18 Sécurité et gestion des utilisateurs **349**

Authentification et autorisation 349
L'authentification 349
L'autorisation 350
Exemples 350
Processus général 353

Première approche de la sécurité 354

Le fichier de configuration de la sécurité 354
Mettre en place un pare-feu 357
Les erreurs courantes 363
Depuis le contrôleur ou un service 364
Depuis une vue Twig 365

Gérer des autorisations avec les rôles 365
Définition des rôles 366
Tester les rôles de l'utilisateur 367
Pour conclure sur les méthodes de sécurisation 370

Gérer des utilisateurs avec la base de données 370
Qui sont les utilisateurs ? 370
Créons notre classe d'utilisateurs 371
Créer des utilisateurs de test 372
Définir l'encodeur pour la nouvelle classe d'utilisateurs 373
Définir le fournisseur d'utilisateurs 374
Demander au pare-feu d'utiliser le nouveau fournisseur 375
Manipuler les utilisateurs 375

Utiliser FOSUserBundle 376
Installer FOSUserBundle 376
Hériter FOSUserBundle depuis le OCUserBundle 377
Modifier notre entité User 378
Configurer le bundle 379
Mettre à jour la table User 380
Configurer la sécurité pour utiliser le bundle 380
Configurer le fonctionnement de FOSUserBundle 382
Manipuler les utilisateurs avec FOSUserBundle 386

Pour conclure 386

En résumé 387

19 Les services : fonctions avancées 389

Les tags sur les services 389
Les tags 389
Comprendre les tags à travers Twig 389
Appliquer un tag à un service 390
Une classe qui implémente une interface 391
Écrire le code qui sera exécuté 392
Méthodologie 393
Les principaux tags 394
Les événements du cœur 394
Les types de champs de formulaire 394

Dépendances optionnelles : les appels de méthodes (calls) 396
Les dépendances optionnelles 396
Les appels de méthodes (calls) 397
L'utilité des appels de méthodes 397

Les services courants de Symfony 398

En résumé 400

20 Le gestionnaire d'événements de Symfony **401**

Des événements ? Pour quoi faire ? 401
Qu'est-ce qu'un événement ? 401
Qu'est-ce que le gestionnaire d'événements ? 402

Écouter les événements 403
Notre exemple 403
Créer un service et son écouteur 403
Écouter un événement 405
Créer la méthode à exécuter de l'écouteur 408
Méthodologie 411

Les événements Symfony… et les nôtres ! 412
Les événements Symfony 412
Créer ses propres événements 417

Allons un peu plus loin 423
Les souscripteurs d'événements 423
L'ordre d'exécution des écouteurs 425
La propagation des événements 426

En résumé 427

21 Traduire son site **429**

Introduction à la traduction 429
Le principe 429
Traduire avec Symfony 430
Prérequis 431
Configuration 431
Mettre en place une page de test 432

Bonjour le monde 433
Le filtre Twig {{ 'string'\|trans }} 433
La balise de bloc Twig {% trans %} 433
Le service translator 434
Notre vue 435

Le catalogue 435
Les formats de catalogue 436
La mise en cache du catalogue 437
Notre traduction 438
Ajouter un nouveau message à traduire 438
Extraire les chaînes sources d'un site existant 438
Traduire dans une nouvelle langue 440

Récupérer la locale de l'utilisateur 440
Déterminer la locale 440
Routing et locale 441

Organiser vos catalogues 443
Utiliser des mots-clés plutôt que du texte comme chaînes sources 444
Nicher les traductions 445
Permettre le retour à la ligne au milieu des chaînes cibles 446

Utiliser des listes 447
Utiliser les domaines 448
Domaines et bundles 449
Un domaine spécial : validators 449

Traductions dépendant de variables 450
Les placeholders 450
Les placeholders dans le domaine validators 451
Gérer les pluriels 452
Afficher des dates au format local 453

Pour conclure 456

En résumé 457

Cinquième partie – Préparer la mise en ligne 459

22 Convertir les paramètres de requêtes 461

Théorie : pourquoi convertir des paramètres ? 461
Récupérer des entités Doctrine avant même le contrôleur 461
Les convertisseurs de paramètres 462
Un convertisseur utile : DoctrineParamConverter 462
Un peu de théorie sur les convertisseurs 462

Pratique : utiliser les convertisseurs existants 463
Utiliser le convertisseur Doctrine 463
Utiliser le convertisseur Datetime 467

Aller plus loin : créer ses propres convertisseurs 468
Comment sont exécutés les convertisseurs ? 468
Comment Symfony trouve-t-il tous les convertisseurs ? 468
Créer un convertisseur 469
L'exemple de notre JsonParamConverter 470

En résumé 472

23 Personnaliser les pages d'erreur 473

Théorie : remplacer les vues d'un bundle 473
Constater les pages d'erreur 473
Localiser les vues concernées 474
Remplacer les vues d'un bundle 474
Comportement de Twig 475
Pourquoi tous ces formats error.XXX.twig dans le répertoire Exception ? 475

Pratique : remplacer les templates Exception de TwigBundle 476
Créer la nouvelle vue 476
Le contenu d'une page d'erreur 476

En résumé 477

24 Utiliser Assetic pour gérer les codes CSS et JS de votre site 479

Théorie : entre vitesse et lisibilité, pourquoi choisir ? 479
À propos du nombre de requêtes HTTP d'une page web 479
Comment optimiser le front-end ? 480
Améliorer le temps de chargement ! 480
En action ! 480
Conclusion 481

Pratique : Assetic à la rescousse ! 481
Installer Assetic et les bibliothèques de compression 481
Servir des ressources 482
Modifier les ressources servies 484
Gérer le mode prod 486
Comprendre Assetic 486
Exporter ses fichiers CSS et JS 487
Et bien plus encore… 488

En résumé 488

25 Utiliser la console depuis le navigateur 489

Théorie : le composant Console de Symfony 489
Les commandes sont en PHP 489
Exemple d'une commande 490

Pratique : utiliser un ConsoleBundle 491
ConsoleBundle ? 491
Télécharger CoreSphereConsoleBundle 492
Enregistrer le bundle dans le kernel 493
Enregistrer les routes 493
Publier les assets 494
Utiliser la console dans son navigateur 494
Prêts pour l'hébergement mutualisé 494

En résumé 494

26 Déployer son site Symfony en production 495

Préparer son application en local 495
Vider le cache, tout le cache 495
Tester l'environnement de production 496
Soigner ses pages d'erreur 496
Installer une console sur navigateur 497
Vérifier la qualité de votre code 497
Vérifier la sécurité de vos dépendances 498

Vérifier et préparer le serveur de production 499
Vérifier la compatibilité du serveur 499

Déployer votre application 501
Méthode 1 : envoyer les fichiers sur le serveur par FTP 501
Méthode 2 : utiliser l'outil Capifony pour envoyer votre application 502

Les derniers préparatifs 502
S'autoriser l'environnement de développement 503
Mettre en place la base de données 503
S'assurer que tout fonctionne 504
Avoir de belles URL 504
Et profitez ! 505
Les mises à jour de la base de données 506
Une checklist pour vos déploiements 506

En résumé 507

Index **509**

Introduction

Vous développez des sites web régulièrement et vous en avez assez de réinventer la roue ? Vous aimeriez utiliser les bonnes pratiques de développement PHP pour concevoir des sites web de qualité professionnelle ?

Ce cours vous permettra de prendre en main Symfony, le framework PHP de référence. Pourquoi utiliser un framework ? Comment créer un nouveau projet de site web avec Symfony, mettre en place les environnements de test et de production, concevoir les contrôleurs, les templates, gérer la traduction et communiquer avec une base de données via Doctrine ?

Je vous montrerai tout au long de ce cours comment ce puissant framework, adopté par une large communauté, va vous faire gagner en efficacité. Fabien Potencier, créateur de Symfony, introduira chacun des chapitres par une vidéo explicative des principaux points abordés. Les vidéos peuvent être visionnées sur le site web associé au livre (*www.editions-eyrolles.com/dl/0014403*).

Première partie

Vue d'ensemble de Symfony

Commençons par le commencement ! Si vous n'avez aucune expérience dans les frameworks ni dans l'architecture MVC, cette partie sera très riche en nouvelles notions. Avançons doucement mais sûrement, vous êtes là pour apprendre !

1 Symfony, un framework PHP

Dans ce chapitre, nous allons découvrir pourquoi Symfony est un bon choix pour votre application web. Une boîte à outils faite en PHP qui a pour but de vous simplifier la vie, c'est toujours sympa, non ? Allons-y !

Vous savez déjà faire des sites Internet ? Vous maîtrisez votre code, mais n'êtes pas totalement satisfait ? Vous avez trop souvent l'impression de réinventer la roue ?

Alors ce cours est fait pour vous !

Symfony est un puissant framework qui va vous permettre de réaliser des sites complexes rapidement, mais de façon structurée et avec un code clair et maintenable. En un mot : le paradis du développeur !

Ce cours est destiné aux débutants de Symfony. Vous n'avez besoin d'aucune notion sur les frameworks pour l'aborder, car nous allons les découvrir ensemble, pas à pas. Cependant, il est fortement conseillé :

- d'avoir déjà une bonne expérience de PHP (consultez le cours *Concevez votre site web avec PHP et MySQL*, par Mathieu Nebra : https://openclassrooms.com/informatique/cours/concevez-votre-site-web-avec-php-et-mysql) ;
- de maîtriser les notions de base de la POO (consultez le cours *La programmation orientée objet*, par Mathieu Nebra : https://openclassrooms.com/informatique/cours/concevez-votre-site-web-avec-php-et-mysql/la-programmation-orientee-objet-6) ;
- d'avoir éventuellement des notions sur les espaces de noms, ou *namespaces* en anglais (consultez le cours *Les espaces de nom*, par Victor Thuillier : https://openclassrooms.com/informatique/cours/les-espaces-de-noms-en-php).

Si vous ne maîtrisez pas ces trois points, je vous invite vraiment à les apprendre avant de commencer la lecture de ce cours. Symfony requiert ces bases et, si vous ne les avez pas, vous risquez de mettre plus de temps pour assimiler ce cours. C'est comme acheter un A380 sans savoir piloter : c'est joli mais vous n'irez pas bien loin.

Alors, vous avez décidé de vous lancer dans Symfony ? Parfait, vous ne le regretterez pas ! Tout au long de ce cours, nous apprendrons à utiliser ce framework et vous comprendrez petit à petit la puissance de cet outil. Commençons tout d'abord par les bases et voyons précisément quels sont les objectifs et les limites d'un framework tel que Symfony.

Qu'est-ce qu'un framework ?

L'objectif d'un framework

L'objectif de ce chapitre n'est pas de vous fournir toutes les clés pour concevoir un framework, mais suffisamment pour pouvoir en utiliser un. On exposera rapidement l'intérêt, les avantages et les inconvénients de l'utilisation d'un tel outil.

Définition

Le mot *framework* provient de l'anglais *frame*, qui veut dire « cadre » en français, et *work*, qui signifie « travail ». Littéralement, c'est donc un cadre de travail. Concrètement, c'est un ensemble de composants qui sert à créer les fondations, l'architecture et les grandes lignes d'un logiciel. Il existe des centaines de frameworks couvrant la plupart des langages de programmation. Ils sont destinés au développement de sites web ou bien à la conception de logiciels.

Un framework est une boîte à outils conçue par au moins un développeur à destination d'autres développeurs. Contrairement à certains scripts tels que WordPress, Dotclear ou autres, un framework n'est pas utilisable tel quel. Il n'est pas conçu pour les utilisateurs finaux. Le développeur qui se sert d'un framework a encore du travail à fournir, d'où ce cours !

Objectif d'un framework

L'objectif premier d'un framework est d'améliorer la productivité des développeurs qui l'utilisent. Plutôt sympa, non ? Souvent organisé en différents composants, un framework offre la possibilité au développeur final d'utiliser tel ou tel composant pour lui faciliter le développement et ainsi de se concentrer sur le plus important.

Prenons un exemple concret. Il existe dans Symfony un composant qui gère les formulaires HTML : leur affichage, leur validation, etc. Le développeur qui l'utilise se concentre sur l'essentiel dans son application : chaque formulaire effectue une action et c'est cette action qui est importante, pas les formulaires. Étendez ce principe à toute une application ou tout un site Internet et vous comprenez l'intérêt d'un framework ! Autrement dit, le framework s'occupe de la forme et permet au développeur de se concentrer sur le fond.

Pesons le pour et le contre

Comme tout bon développeur, lorsqu'on veut utiliser un nouvel outil, on doit en peser le pour et le contre pour être sûr de faire le bon choix !

Le pour

L'avantage premier est donc, on vient de le voir, le gain en productivité. Mais il en existe bien d'autres ! On peut les classer en plusieurs catégories : le code, le travail et la communauté.

Tout d'abord, un framework vous aide à réaliser un « **bon code** », c'est-à-dire qu'il vous incite, de par sa propre architecture, à bien organiser votre code. Et un code bien organisé est évolutif et facile à maintenir ! De plus, un framework offre des briques prêtes à l'emploi (le composant `Form` de Symfony par exemple), ce qui vous évite de réinventer la roue, et surtout qui vous permet d'utiliser des briques puissantes et éprouvées. En effet, ces dernières sont développées par des équipes de développeurs chevronnés ; elles sont donc très flexibles et très robustes. Vous économisez ainsi des heures de développement !

Ensuite, un framework améliore **la façon dont vous travaillez**. En effet, dans le cas d'un site Internet, vous travaillez souvent avec d'autres développeurs PHP et un designer. Un framework vous aide doublement dans ce travail en équipe. D'une part, un framework utilise presque toujours l'architecture MVC ; on en reparlera, mais sachez pour le moment que c'est une façon d'organiser son code en séparant le PHP du HTML. Ainsi, votre designer peut travailler sur des fichiers différents des vôtres ; finis les problèmes d'édition simultanée d'un même fichier ! Par ailleurs, un framework a une structure et des conventions de code connues. Ainsi, vous pouvez facilement recruter un autre développeur : s'il connaît déjà le framework en question, il s'intégrera très rapidement au projet.

Enfin, le dernier avantage est la **communauté** soutenant chaque framework. C'est elle qui fournit les tutoriels ou les cours (comme celui que vous lisez !), de l'aide sur les forums et, bien sûr, les mises à jour du framework. Ces dernières sont très importantes : imaginez que vous codiez vous-mêmes tout ce qui est connexion utilisateur, session, moteur de templates, etc. Comme il est impossible de coder sans bogues, vous devriez logiquement corriger chaque erreur déclarée sur votre code. Maintenant, imaginez que toutes les briques de votre site, qui ne sont pas forcément votre tasse de thé, soient fournies par le framework. À chaque fois que vous ou les milliers d'autres utilisateurs du framework trouverez une bogue, les développeurs et la communauté s'occuperont de le corriger et vous n'aurez plus qu'à suivre les mises à jour. Un vrai paradis !

Il existe plein d'autres avantages que je ne vais pas vous détailler, mais un framework, c'est aussi :

- une communauté active qui utilise le framework et qui contribue en retour ;
- une documentation de qualité et régulièrement mise à jour ;
- un code source maintenu par des développeurs attitrés ;

- un code qui respecte les standards de programmation ;
- un support à long terme garanti et des mises à jour qui ne cassent pas la compatibilité ;
- etc.

Le contre

Vous vous en doutez, avec autant d'avantages il y a forcément des inconvénients. Et bien, figurez-vous qu'il n'y en a pas tant que ça !

S'il ne fallait en citer qu'un, cela serait évidemment la courbe d'apprentissage qui est plus élevée. En effet, pour maîtriser un framework, il faut un temps d'apprentissage non négligeable. Chaque brique qui compose un framework a sa complexité propre qu'il vous faudra appréhender.

Notez également que pour les frameworks les plus récents, tels que Symfony justement, il faut être au courant des dernières nouveautés de PHP. Connaître certaines bonnes pratiques telles que l'architecture MVC est un plus.

Toutefois, rien de tout cela ne doit vous effrayer ! Voyez l'apprentissage d'un framework comme un investissement : il y a un certain effort à fournir au début, mais les résultats se récoltent ensuite sur le long terme !

Alors, convaincus ?

J'espère vous avoir convaincus que le pour l'emporte largement sur le contre. Si vous êtes prêts à relever le défi aujourd'hui pour être plus productifs demain, alors ce cours est fait pour vous !

Qu'est-ce que Symfony ?

Un framework

Symfony est donc un framework PHP. Bien sûr, il en existe d'autres : Zend Framework (*http://framework.zend.com/*), CodeIgniter (*http://codeigniter.com/*), CakePHP (*http://cakephp.org/*), etc. Le choix d'un framework est assez personnel et doit être adapté au projet engagé. Sans vouloir prêcher pour ma paroisse, Symfony est l'un des plus flexibles et des plus puissants.

Un framework populaire

Symfony est très populaire. C'est un des frameworks les plus utilisés dans le monde, notamment dans les entreprises. Citons Dailymotion par exemple ! La première version de Symfony est sortie en 2005 et est aujourd'hui toujours très répandue. Cela lui apporte un retour d'expérience et une notoriété exceptionnels. Aujourd'hui, beaucoup d'entreprises dans le domaine d'Internet (dont OpenClassrooms !) recrutent des développeurs capables de travailler sous ce framework. Ces développeurs pourront ainsi

se greffer aux projets de l'entreprise très rapidement, car ils en connaîtront déjà les grandes lignes. C'est un atout si vous souhaitez travailler dans ce domaine.

La deuxième version est sortie en août 2011. Son développement a été fulgurant grâce à une communauté de développeurs dévoués. Bien que différente dans sa conception, cette deuxième version est plus rapide et plus souple que la première. Très rapidement après sa sortie, de nombreuses entreprises s'arrachaient déjà les compétences des développeurs Symfony2.

Enfin la troisième version, que nous étudierons dans ce cours, est la maturation de la version 2. Elle s'inscrit dans la continuité de la précédente et vient en supprimer tous les points dépréciés qui freinaient son développement. La version 3 est donc une version 2 améliorée, qui fait table rase des quelques erreurs de jeunesse et ouvre la voie à encore plus d'évolution à l'avenir ! Contrairement au passage entre les deux premières moutures, le passage entre les versions 2 et 3 se fait relativement facilement ; vous n'avez pas à réécrire votre code pour mettre à jour !

Comme vous pouvez le voir, Symfony se développe à vive allure et aujourd'hui il est presque incontournable en entreprise. Faites partie de la communauté !

Un framework populaire et français

Et, oui, Symfony, l'un des meilleurs frameworks PHP au monde, est français ! Il est édité par la société SensioLabs (*http://sensiolabs.com/*), dont le créateur est Fabien Potencier. Cependant, Symfony étant open source, il a également été écrit par toute la communauté : beaucoup de Français, mais aussi des développeurs de tous horizons : Europe, États-Unis, etc. C'est grâce au talent de Fabien et à la générosité de la communauté que Symfony a vu le jour.

Qu'est-il possible de faire avec Symfony ?

Avec Symfony, comme avec beaucoup de frameworks PHP, vous n'êtes limités que par votre imagination ! En effet, il est possible de tout faire : ce n'est pas le framework qui vous posera des limites, il ne met en place qu'un cadre de travail. Libre à vous d'utiliser ce cadre comme bon vous semble ! Je vous ai parlé de Dailymotion (*http://www.dailymotion.com/fr*), un site de partage de vidéos, mais vous pouvez également créer un site e-commerce, comme je l'ai fait avec Caissin (*https://www.caissin.fr/*) ou encore un site plus complexe tel qu'OpenClassrooms (*https://openclassrooms.com/*), qui tourne également sur Symfony.

C'est l'une des forces de Symfony : il vous permet de créer le site Internet de vos rêves en vous fournissant tous les outils nécessaires pour y arriver avec succès.

Télécharger Symfony

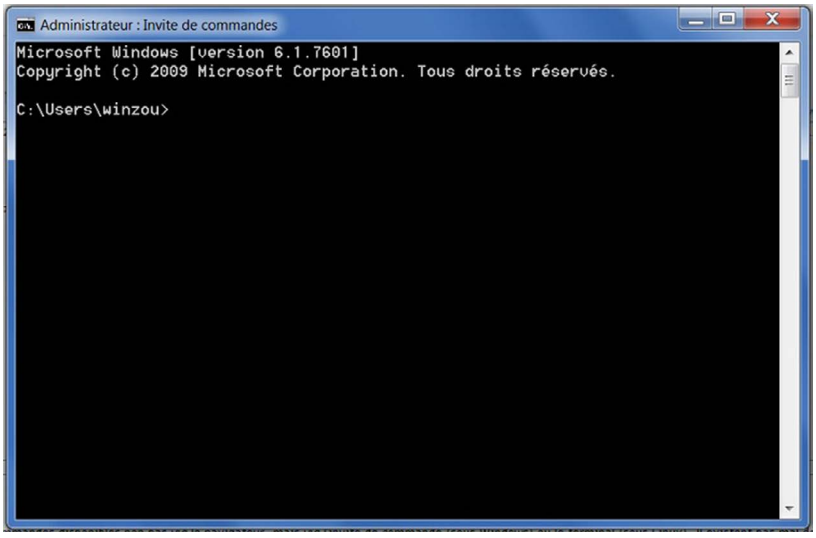
Vérifier l'installation de PHP en console

Nous aurons parfois besoin d'exécuter des commandes PHP via la console pour générer du code ou gérer la base de données. Ce sont des commandes qui vont nous faire gagner du temps (toujours le même objectif !). Vérifions donc que PHP est bien disponible en console. Rassurez-vous, je vous indiquerai toujours pas à pas comment les utiliser.

Si vous êtes sous Linux ou Mac, vous ne devriez pas avoir de souci ; PHP est bien disponible en console. Si vous êtes sous Windows, rien n'est sûr. Dans tous les cas, vérifiez-le en ouvrant l'invite de commandes pour Windows, ou le terminal pour Linux.

Sous Windows

Lancez l'invite de commandes : ***Menu Démarrer>Programmes>Accessoires>Invite de commandes***. Une fenêtre semblable à la figure suivante devrait apparaître.



La console Windows

Puis exécutez la commande suivante :

```
C:\Users\winzou> php -v
PHP 5.5.12 (cli) (built: Apr 30 2014 11:20:55)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
with Zend OPcache v7.0.4-dev, Copyright (c) 1999-2014, by Zend Technologies
```

Sous Linux et Mac

Ouvrez le terminal et exécutez la même commande :

```
winzou@laptop:~$ php -v
```

Si tout va bien

Si cette commande vous retourne bien la version de PHP et d'autres informations, tout est bon. Profitez-en pour vérifier votre version de PHP ; nous aurons besoin ici de la version 5.5 au minimum. Si vous avez PHP 5.4 ou inférieur, vous devez d'abord le mettre à jour.

En cas d'erreur

Si vous êtes sous Windows et si la commande affiche une erreur, votre PHP est sûrement bien installé, mais Windows ne sait pas où le trouver ; il faut juste lui montrer le chemin. Voici la démarche à suivre pour régler ce problème.

1. Allez dans les paramètres système avancés (***Démarrer>Panneau de configuration>Système et sécurité>Système>Paramètres système avancés***).
2. Cliquez sur le bouton ***Variables d'environnement...***
3. Regardez dans le panneau ***Variables système***.
4. Trouvez l'entrée ***Path*** (vous devriez avoir à faire descendre l'ascenseur pour la trouver) et double-cliquez dessus.
5. Entrez votre répertoire PHP à la fin, sans oublier le point-virgule en début de ligne. C'est le répertoire dans lequel se trouve le fichier `php.exe` ; par exemple, `C:\wamp\bin\php\php5.5.12`.
6. Confirmez en cliquant sur ***OK***. Vous devez ensuite redémarrer l'invite de commandes pour prendre en compte les changements.

Si vous êtes sous Linux, vérifiez votre installation de PHP. Vous devez notamment avoir le paquet `php5-cli`, qui est la version console de PHP.

Dans les deux cas, vérifiez après vos manipulations que le problème est bien résolu. Pour cela, exécutez à nouveau la commande `php -v`. Elle devrait alors vous afficher la version de PHP.

Et voilà, votre poste de travail est maintenant opérationnel pour développer avec Symfony !

Obtenir Symfony

Ce cours a été écrit pour la version 3.0 de Symfony (sortie fin novembre 2015).
Symfony 3.0 étant totalement compatible avec la version 2.8, vous pouvez suivre le cours même si vous êtes sur la branche 2.x en version 2.8. En revanche, certains points pourront être incompatibles avec les versions inférieures à 2.8. La transition se fait facilement, alors pensez à vous mettre à jour !

Il existe de nombreux moyens d'obtenir Symfony. Nous allons voir ici la méthode recommandée : le Symfony Installer. Il s'agit d'un petit fichier PHP (un package `PHAR` en réalité) à télécharger puis exécuter sur votre PC.

Rendez-vous à l'adresse suivante : *http://symfony.com/installer*. Cela va télécharger un fichier `symfony.phar`, que vous devez déplacer dans votre répertoire `/web` habituel, par exemple `C:\wamp\www` pour Windows ou `/var/www` pour Linux.

Ce fichier permet d'exécuter plusieurs commandes, mais la seule qui nous intéresse pour l'instant est `new`, qui crée un nouveau projet Symfony en partant de zéro.

Puis allez dans le répertoire où vous avez placé le fichier `symfony.phar`, en utilisant la commande `cd` (je vous laisse adapter la commande si vous êtes sous Linux ou Mac) :

```
Microsoft Windows [version 10.0.10586]
(c) 20015 Microsoft Corporation. Tous droits réservés.

C:\Users\winzou> cd ../../wamp/www
C:\wamp\www> _
```

Sous Windows, vous avez également la possibilité de vous rendre dans votre répertoire `/web` via l'explorateur de fichiers et de cliquer-droit en appuyant en même temps sur la touche ***Maj*** de votre clavier. Dans le menu contextuel, choisissez ***Ouvrir une fenêtre de commandes ici***.

Maintenant, exécutons la commande suivante pour créer un nouveau projet dans le répertoire `Symfony` :

```
C:\wamp\www> php symfony.phar new Symfony
Downloading Symfony...
4.97 B/4.97 MB ==================================> 100%
Preparing project...
OK Symfony 3.0.0 was successfully installed. Now you can:
   * Change your current directory to D:\www\Symfony
   * Configure your application in app/config/parameters.yml file.
   * Run your application:
     1. Execute the php bin/console server:run command.
     2. Browse to the http://localhost:8000 URL.
   * Read the documentation at http://symfony.com/doc
C:\wamp\www> _
```

Et voilà ! Vous venez de télécharger tout le nécessaire pour faire tourner un projet Symfony dans le répertoire `C:\wamp\www\Symfony` (ou `/var/www/Symfony` sur Linux).

Pour la suite du cours, je considérerai que les fichiers sont accessibles à l'URL *http://localhost/Symfony*. Je vous recommande d'avoir la même adresse, car je ferai ce genre de liens tout au long du cours.

Droits d'accès

Je fais un petit aparté pour les lecteurs travaillant sous Linux (sous Windows pas de souci, vous pouvez passer votre chemin). Symfony a besoin d'écrire dans le répertoire `var`, il faut donc bien régler les droits dessus. Pour cela, placez-vous dans le répertoire `Symfony` et videz d'abord `var` :

```
rm -rf var/*
```

Pour ceux qui sont encore en version 2.8, les répertoires dans lesquels Symfony2 écrit sont `app/cache` et `app/logs`. Vous devez donc adapter les commandes à ces répertoires.

Ensuite, si votre distribution supporte le `chmod +a`, exécutez ces commandes pour définir les bons droits :

```
HTTPDUSER=`ps aux | grep -E '[a]pache|[h]ttpd|[_]www|[w]ww-data|[n]ginx'
| grep -v root | head -1 | cut -d\ -f1`
sudo chmod +a "$HTTPDUSER allow delete,write,append,file_inherit,directory_
inherit" var
sudo chmod +a "`whoami` allow delete,write,append,file_inherit,directory_
inherit" var
```

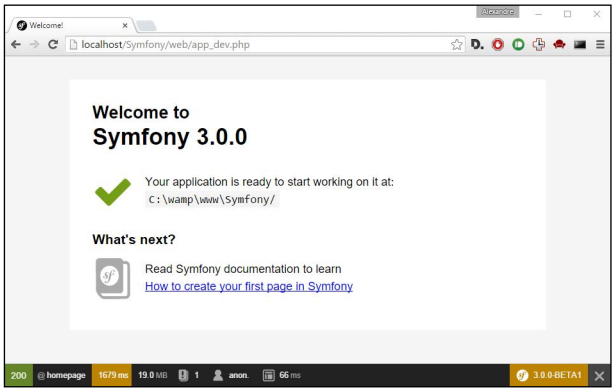
Si vous rencontrez une erreur avec ces commandes (le `chmod +a` n'est pas disponible partout), exécutez les commandes suivantes :

```
HTTPDUSER=`ps aux | grep -E '[a]pache|[h]ttpd|[_]www|[w]ww-data|[n]ginx'
| grep -v root | head -1 | cut -d\ -f1`
sudo setfacl -R -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX var
sudo setfacl -dR -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX var
```

Enfin, si vous ne pouvez pas utiliser les ACL (utilisés dans les commandes précédentes), définissez simplement les droits comme suit :

```
chmod 777 -R var
```

Voilà, vous pouvez dès à présent exécuter Symfony, félicitations ! Rendez-vous sur la page *http://localhost/Symfony/web/app_dev.php/*. Vous devriez avoir quelque chose ressemblant à la figure suivante.



La page d'accueil de Symfony

En résumé

- Le mot *framework* signifie « cadre de travail » en français.
- L'objectif principal d'un framework est d'améliorer la productivité des développeurs qui l'utilisent.
- Contrairement aux CMS, un framework est destiné à des développeurs et non à des novices en informatique.
- L'apprentissage d'un framework est un investissement : il y a un certain effort à fournir au début, mais les résultats se récoltent ensuite sur le long terme !
- Symfony est un framework PHP très populaire, français et très utilisé dans le milieu des entreprises.

Sur le site OpenClassrooms, vous trouverez la vidéo d'un entretien avec le créateur de Symfony.
Pour plus d'informations, n'hésitez pas à consulter les sites suivants :

- Symfony *http://symfony.com*
- SensioLabs Connect *https://connect.sensiolabs.com*

À l'époque où nous avons réalisé cette interview, Symfony n'était encore qu'en version 2 ; c'est donc celle-là qui est évoquée dans la vidéo. Mais le discours reste d'actualité !