Déployer son site Symfony en production

Table des matières

I. Contexte

Durée : 1 h

Environnement de travail : PC ou Mac avec 8 Go de ram minimum

Ide : VSCode ou autre

Prérequis : notion de PHP et de POO

Contexte

Symfony est un framework PHP largement utilisé pour développer des applications web robustes et évolutives. Cependant, la mise en production d'un site Symfony nécessite une préparation minutieuse en amont pour garantir une expérience utilisateur optimale. En effet, la mise en production peut être une étape complexe et critique pour le bon fonctionnement de l'application, notamment en termes de sécurité, de performance et de disponibilité.

Avant de déployer votre site Symfony en production, vous devez prendre en compte plusieurs éléments clés, tels que la configuration du serveur, l'optimisation des performances, la sécurité, la gestion des erreurs et des sauvegardes, et la mise en place d'un plan de reprise d'activité en cas de dysfonctionnement. En planifiant ces éléments à l'avance, vous pouvez minimiser les risques de perturbation de votre application et garantir une expérience utilisateur optimale.

II. La préparation avant le déploiement en production

A. La préparation avant le déploiement en production

Optimiser les performances

Vous devez optimiser les performances de votre application Symfony pour assurer une expérience utilisateur rapide et fluide. Pour ce faire, vous pouvez utiliser des outils tels que le profiler Symfony pour identifier les problèmes de performance.

Méthode **Caching**

Le caching consiste à stocker en mémoire les résultats de requêtes ou de calculs afin de les réutiliser ultérieurement plutôt que de les recalculer à chaque fois. Dans Symfony, vous pouvez utiliser le composant Cache pour mettre en cache des objets, des requêtes de base de données, des fragments de page, etc.

Exemple

Voici un exemple de code qui utilise le composant Cache de Symfony pour stocker en cache les résultats d'une requête de base de données pendant 10 minutes :

```php
use Symfony\Component\Cache\Adapter\FilesystemAdapter;
use Symfony\Component\HttpFoundation\Response;

// ...

public function myAction()
{
    // Créer un cache adapter avec le temps de vie des entrées de 10 minutes
    $cache = new FilesystemAdapter('', 600);

    // Générer une clé unique pour cette requête de base de données
    $cacheKey = 'my_database_query_cache_key';
```

```php

    // Vérifier si la valeur est déjà en cache
    $cachedResult = $cache->getItem($cacheKey);

    if ($cachedResult->isHit()) {
        // Si la valeur est en cache, retourner la réponse avec le contenu mis en cache
        return new Response($cachedResult->get());
    }

    // Si la valeur n'est pas en cache, effectuer la requête de base de données
    $databaseResult = $this->getDoctrine()->getRepository(MyEntity::class)->findAll();

    // Convertir le résultat de la requête en une chaîne de caractères
    $resultAsString = serialize($databaseResult);

    // Stocker la chaîne de caractères en cache pendant 10 minutes
    $cachedResult->set($resultAsString);
    $cache->save($cachedResult);

    // Retourner la réponse avec le contenu de la requête de base de données
    return new Response($resultAsString);
}
```

Dans cet exemple, nous créons un cache `adapter` (`new FilesystemAdapter`) avec une durée de vie de 10 minutes, puis nous vérifions si les résultats sont déjà en cache, en utilisant une clé unique pour cette requête de base de données. Si les résultats sont en cache, nous les retournons directement à l'utilisateur. Sinon, nous exécutons la requête de base de données, convertissons les résultats en une chaîne de caractères, stockons la chaîne de caractères en cache et retournons-la à l'utilisateur.

Remarque

Notez qu'utiliser le composant cache pour optimiser les performances n'enlève pas le fait de vider la mémoire cache avant la mise en production. Cela peut se faire avec la ligne de commande ci-dessous :

```
$ php bin/console cache:clear
```

Méthode **Activer le mode production**

Pendant le développement de votre application, vous étiez en mode développement ou `dev`. Le mode de production doit être activé pour votre application Symfony avant le déploiement en production, afin de vous mettre avant le déploiement sur le serveur final. Pour activer le mode de production, vous devez définir la variable d'environnement `APP_ENV` sur prod.

Vous pouvez également supprimer tous les composants que vous avez installés qui ne sont utilisables qu'en mode `dev`, comme le profiler par exemple. Vous retrouverez tous ces composants dans composer.json et dans "require-dev". Pour les supprimer tous en une seule fois avec la commande :

```
$ composer install -no-dev
ou
$ composer --no-dev update
```

Cela permettra d'assurer que votre application Symfony en production utilise uniquement les composants nécessaires, optimisant ainsi ses performances et sa sécurité.

Git

Git est un système de gestion de version. Cela signifie qu'il permet de garder une trace de toutes les modifications apportées à un fichier ou à un ensemble de fichiers au fil du temps. Les développeurs l'utilisent souvent pour collaborer sur des projets, pour garder une trace des modifications apportées au code, pour faciliter les tests et pour sauvegarder le code en cas de problème.

Voici quelques-unes des commandes Git de base que vous pouvez utiliser :

Commandes	Objectifs
git init	Cette commande initialise un nouveau dépôt Git dans un répertoire existant ou vide.
git add	Cette commande ajoute des fichiers à la zone de staging, qui est une zone temporaire où vous préparez les fichiers avant de les versionner.
git commit	Cette commande enregistre les modifications apportées aux fichiers de la zone de staging dans l'historique des versions de Git. Vous pouvez également ajouter un message de commit pour décrire les modifications.
git status	Cette commande vous permet de vérifier l'état de votre dépôt Git. Vous pouvez voir quels fichiers ont été modifiés, quels fichiers sont dans la zone de staging, etc.
git log	Cette commande vous permet de voir l'historique des commits dans votre dépôt Git. Vous pouvez voir qui a effectué chaque commit, quand il a été effectué et quel message de commit a été ajouté.
git branch	Cette commande vous permet de créer une nouvelle branche dans votre dépôt Git. Les branches sont des copies du code qui vous permettent d'expérimenter et de travailler sur de nouvelles fonctionnalités sans affecter le code principal.
git merge	Cette commande vous permet de fusionner une branche dans la branche principale de votre dépôt Git.
git pull	Cette commande vous permet de récupérer les modifications apportées à un dépôt distant et de les intégrer à votre dépôt local.
git push	Cette commande vous permet de pousser les modifications apportées à votre dépôt local vers un dépôt distant, comme GitHub.

En utilisant ces commandes de base, vous pouvez commencer à utiliser Git pour gérer vos projets de développement de logiciels de manière efficace et organisée.

Pour donner encore plus de puissance à Git aujourd'hui, la plupart des développeurs utilisent Github ou Gitlab pour ne parler que des plus importantes plateformes de développement collaboratives de logiciels, basées sur Git. Avec ses hébergeurs de dépôts Git en ligne, il est possible de travailler avec d'autres développeurs sur le même code en temps réel. De plus, déployer directement une application web à partir d'un dépôt Git en ligne est tout à fait envisageable.

[cf. userSymfony1-main.zip]

Étape théorique pour qu'une application soit bien installée en production

- *Upload your code to the production server:*

  Transférer tous les fichiers de code nécessaire de votre environnement de développement vers le serveur de production.

- *Install your vendor dependencies (typically done via Composer and may be done before uploading):*

  L'installation se fait généralement via Composer et peut être faite avant ou après le téléchargement sur le serveur de production.

- *Running database migrations or similar tasks to update any changed data structures:*

  Étape nécessaire si vous avez apporté des modifications à la structure de votre base de données. Vous pouvez utiliser la commande Symfony "php bin/console doctrine:migrations:migrate" pour exécuter les migrations.

- *Clearing (and optionally, warming up) your cache:*

  Le cache doit être vidé pour s'assurer que votre application utilise les dernières modifications apportées au code. Vous pouvez utiliser la commande Symfony "*php bin/console cache:clear*" pour vider le cache.

- *A deployment may also include other tasks, such as:*

  Le déploiement peut également inclure d'autres tâches, telles que :

  - *Tagging a particular version of your code as a release in your source control repository*.

    C'est-à-dire taguer une version particulière de votre code en tant que version de sortie dans votre référentiel de contrôle de source.

  - *Creating a temporary staging area to build your updated setup "offline".*

    C'est-à-dire créer une zone temporaire appelée "*staging*" pour créer votre installation à jour hors ligne.

  - *Running any tests available to ensure code and/or server stability.*

    C'est-à-dire exécuter tous les tests disponibles pour garantir la stabilité du code et/ou du serveur.

  - *Removal of any unnecessary files from the public/ directory to keep your production environment clean.*

    Supprimer tous les fichiers inutiles du dossier public/ pour garder votre environnement de production propre.

- *Clearing of external cache systems (like Memcached or Redis).*

  Vider les systèmes de cache externes (comme Memcached ou Redis).

B. Exercice : Quiz

[solution n°1 p.13]

Question 1

Pourquoi la mise en production d'un site Symfony nécessite-t-elle une préparation minutieuse en amont ?

- ○ Pour garantir une expérience utilisateur optimale
- ○ Pour garantir une compatibilité avec tous les navigateurs web
- ○ Pour éviter les conflits entre les développeurs

Question 2

Quels sont les éléments clés à prendre en compte avant de déployer un site Symfony en production ?

- ○ La configuration du serveur, l'optimisation des performances, la sécurité, la gestion des erreurs et des sauvegardes
- ○ Le choix du nom de domaine, le design du site, la rédaction du contenu et les images
- ○ La publicité en ligne, le marketing digital et le référencement naturel

Question 3

Pourquoi est-il important de vider la mémoire cache avant la mise en production ?

- Pour libérer de l'espace disque sur le serveur
- Pour éviter les conflits avec les données déjà stockées en cache
- Pour éviter les problèmes de sécurité

Question 4

Qu'est-ce que le caching dans Symfony ?

- Une technique pour éviter les erreurs de syntaxe dans le code PHP
- Une technique pour stocker en mémoire les résultats de requêtes ou de calculs afin de les réutiliser ultérieurement plutôt que de les recalculer à chaque fois
- Une technique pour compresser les fichiers JavaScript et CSS

Question 5

Comment activer le mode production pour une application Symfony avant le déploiement en production ?

- En définissant la variable d'environnement APP_ENV sur prod
- En utilisant la commande php bin/console server:run --env=prod
- En modifiant le fichier .htaccess sur le serveur web

III. La mise en production de votre application Symfony

A. La mise en production de votre application Symfony

Une fois que vous avez préparé votre application Symfony pour la mise en production en suivant les étapes précédentes, vous pouvez commencer à déployer votre application sur le serveur final.

Choisissez un fournisseur d'hébergement fiable et sécurisé qui répond à vos besoins en termes de performances, de disponibilité et de support technique. Assurez-vous que le serveur est configuré pour exécuter les dernières versions de PHP et de Symfony.

Choisir un hébergeur

Choisir un bon hébergeur est une étape importante dans la création de votre site web. Voici quelques éléments à prendre en compte pour faire un choix éclairé :

- **La fiabilité** : assurez-vous que l'hébergeur offre une disponibilité élevée (idéalement proche de 100 %) et qu'il dispose d'une infrastructure fiable pour garantir la continuité du service.
- **Les performances** : vérifiez que l'hébergeur offre des temps de réponse rapides et des temps de chargement de pages optimisés, car cela peut avoir un impact significatif sur l'expérience utilisateur de vos visiteurs.
- **Le support client** : il est important de choisir un hébergeur qui offre un support client réactif et compétent, notamment pour résoudre rapidement les problèmes techniques.
- **Les fonctionnalités** : vérifiez que l'hébergeur propose les fonctionnalités nécessaires à votre projet, telles que les bases de données, les certificats SSL, les outils de sauvegarde, etc.
- **Le prix** : comparez les offres des différents hébergeurs en prenant en compte le coût et la qualité du service proposé.
- **Les avis des utilisateurs** : n'hésitez pas à consulter les avis et les retours d'expérience d'autres utilisateurs pour vous faire une idée de la qualité de service de l'hébergeur.

En prenant en compte ces critères, vous devriez être en mesure de choisir un hébergeur fiable et adapté à vos besoins.

Méthode **Déployer une application Symfony**

Afin de donner un exemple concret, nous allons déployer une application sur Heroku.

Heroku est une plateforme en tant que service, basée sur le cloud (Pass). Cette solution aide les développeurs et les entreprises à déployer entre autres leurs applications. Il est possible de déployer une petite application gratuitement à partir de Git.

Voici comment s'y prendre avec une application Symfony à partir de Git :

Tout d'abord, il vous faudra créer un compte sur Cloud Application Platform | Heroku[1]. Puis il vous faudra installer Heroku sur votre ordinateur. La méthode diverge si dans votre environnement vous travaillez avec Mac, Windows ou Linux. Vous trouverez toutes les informations sur The Heroku CLI | Heroku Dev Center[2] pour installer convenablement Heroku.

Une fois installé, il va falloir vous connecter. Pour cela, utilisez la ligne de commande à partir du répertoire de votre projet :

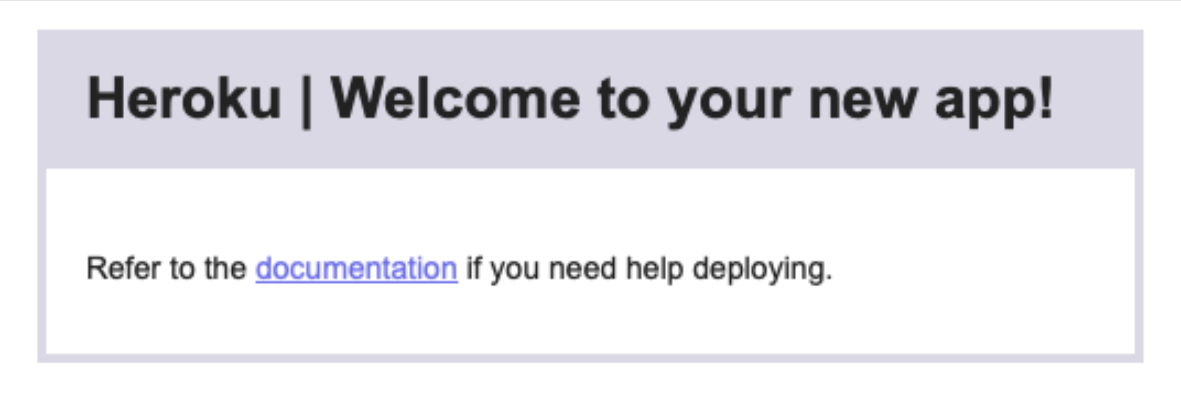
```
$ heroku login
```

Une fenêtre de votre navigateur par défaut va s'ouvrir sur le site d'Heroku, afin que vous vous y connectiez.

Retourner dans le terminal (vérifiez bien que vous êtes dans le bon répertoire), puis saisissez la ligne de commande :

```
$ heroku create
```

Le terminal va vous indiquer que l'app a bien été créée et vous propose un lien qui vous permet d'ouvrir immédiatement votre nouvelle application. Bien sûr, à ce stade, il n'y a pas grand-chose. Voici ce que votre navigateur affichera si vous cliquez sur le lien.



Page de l'application créer avec Heroku

[cf. usersymfony2-main.zip]

Méthode

Maintenant, il va falloir configurer l'application avant qu'elle ne soit envoyée à Heroku à la place de cette page par défaut qui ne nous sert pas à grand-chose pour l'instant, à part à constater que l'application fonctionne.

Commençons par créer un fichier Procfile afin d'indiquer le type de serveur que nous voulons utiliser et à partir de quel fichier l'application doit démarrer.

1 https://www.heroku.com/
2 https://devcenter.heroku.com/articles/heroku-cli

Pour cela, saisissez cette ligne de commande :

```
$ echo 'web: heroku-php-apache2 public/' >Procfile
```

Si vous regardez le code de votre application, vous devriez voir un nouveau fichier à la racine du projet qui se nomme Procfile avec dedans : `web: heroku-php-apache2 public/` qui indique que nous voulons utiliser un serveur apache et qu'il faut démarrer l'application à partir du dossier public. Ce qui est le cas pour toutes les applications Symfony.

Pour qu'apache soit convenablement configuré, il faut créer un fichier .htaccess. Vous pouvez le faire avec cette ligne de commande :

```
$ composer require symfony/apache-pack
```

Validez l'installation des recettes avec `'y'`.

Méthode

Nous allons maintenant ajouter une base de données. Cela se fait également par lignes de commande :

```
$ heroku addons:create cleardb:ignite
$ heroku config | grep CLEARDB_DATABASE_URL
$ heroku config:set DATABASE_URL='mysql://resultat_d'affichage_ligne_d'avant'
```

Avec la première ligne de commande, nous avons créé une base de données mysql sur le serveur d'Heroku. `'ignite'` définit le niveau de base de données que l'on souhaite, et `'ignite'` est la version gratuite.

Avec la deuxième ligne de commande, nous récupérons l'URL de la base de données qui a été créée dans CLEARDB_DATABASE_URL.

Puis avec la troisième ligne de commande, nous avons configuré la variable d'environnement DATABASE_URL avec l'URL récupérée.

La base de données est créée, mais sans les tables. Pour indiquer à Heroku d'effectuer les migrations, il va falloir dans le fichier composer.json ajouter dans les `"scripts"` ce morceau de code :

```
"scripts": {
            "compile": [
          "php bin/console doctrine:migrations:migrate"
      ]
  },
```

Bien évidemment, vous avez d'autres scripts dans `"scripts"` du fichier composer.json, vous devez les laisser et simplement ajouter `"compile"` comme ci-dessus.

Si votre variable d'environnement APP_ENV est encore en mode dev. Vous pouvez la modifier ainsi avec le terminal :

```
$ heroku config:set APP_ENV=prod
```

Méthode

Nous avons préparé le déploiement de notre application. Il nous faut versionner le projet pour que toutes ses modifications soient prises en compte :

```
$ git add .
$ git commit -m "heroku config"
$ git push origin main
```

Pour déployer l'application web, nous allons utiliser Git. En effet, si vous faites un "git remote", vous constaterez que vous avez 2 dépôts distants : celui de github ou de gitlab sous origin et celui de Heroku. Vous l'aurez peut-être compris, il ne nous reste plus qu'à pusher sur le dépôt d'Heroku pour déployer le projet. Voici la commande à saisir :

```
$ git push heroku main
```

Voilà ! Vous avez déployé votre première application web. Il ne vous reste plus qu'à la tester, soit en rafraichissant la page que vous aviez ouverte, soit en ouvrant la page que le terminal vous rappelle qui commence par https et qui finit par herokuapp.com

Afin de pouvoir être pragmatiques, nous avons fait un pas-à-pas d'un déploiement sur une plateforme comme Heroku. Mais tous les déploiements ne s'effectuent pas de la même façon, voici quelques points clés à prendre en compte pour une configuration optimale :

1. Configuration de php.ini : il est recommandé de suivre les recommandations de SensioLabs pour configurer php.ini afin d'améliorer les performances et la sécurité de votre application. Certains paramètres importants incluent la configuration du temps d'exécution max_execution_time et la configuration de la mémoire maximale memory_limit.
2. .htaccess : le fichier .htaccess est utilisé pour définir les règles de redirection et d'accès à votre application. Il est important de s'assurer que le fichier .htaccess est correctement configuré pour sécuriser votre application et pour rediriger toutes les requêtes vers le front controller de Symfony.
3. Configuration d'Apache : la configuration d'Apache est importante pour rediriger toutes les requêtes vers le front controller de Symfony (par exemple, public/index.php). Il est également important de configurer la gestion des erreurs pour renvoyer les erreurs et les exceptions à la page d'erreur personnalisée de Symfony.
4. Redirection vers le front controller : il est important de s'assurer que toutes les requêtes sont redirigées vers le front controller de Symfony, qui est responsable de la gestion des requêtes et des réponses de l'application. Le front controller par défaut pour Symfony est public/index.php.

B. Exercice : Quiz

[solution n°2 p.14]

Question 1

Quels éléments doivent être pris en compte lors du choix d'un hébergeur pour votre application Symfony ?

- ☐ La fiabilité
- ☐ Les performances
- ☐ Le support client
- ☐ Les fonctionnalités

Question 2

Qu'est-ce que Heroku ?

- ○ Un framework pour Symfony
- ○ Un hébergeur de site web
- ○ Un outil de développement pour Symfony

Question 3

Quelle est l'étape à suivre après avoir préparé une application Symfony pour la mise en production ?

- ○ Rédiger un manuel d'utilisation de l'application
- ○ Tester l'application
- ○ Déployer l'application sur le serveur final

Question 4

Comment créer un compte sur Heroku ?

- Directement sur le site web de Heroku
- En utilisant la ligne de commande à partir du répertoire de votre projet
- En installant Heroku sur votre ordinateur

Question 5

Quel est le fichier à créer pour indiquer le type de serveur à utiliser et à partir de quel fichier l'application doit démarrer ?

- .htaccess
- Procfile
- config.yml

IV. Essentiel

Symfony est un framework PHP couramment utilisé pour développer des applications web robustes et évolutives. Cependant, la mise en production d'un site Symfony nécessite une préparation minutieuse en amont pour garantir une expérience utilisateur optimale. Pour assurer une expérience utilisateur rapide et fluide, il est important d'optimiser les performances de votre application Symfony et d'identifier les problèmes de performance et de caching pour stocker en mémoire les résultats de requêtes ou de calculs.

La mise en production peut être une étape complexe et critique pour le bon fonctionnement de l'application, notamment en termes de sécurité, de performance et de disponibilité. Avant de déployer votre site Symfony en production, vous devez prendre en compte plusieurs éléments clés, tels que la configuration du serveur, la gestion des erreurs et des sauvegardes, la sécurité, et la mise en place d'un plan de reprise d'activité en cas de dysfonctionnement.

La préparation en amont vous permet de mieux comprendre les besoins de votre application et de mieux anticiper les problèmes potentiels. En y consacrant le temps et les ressources nécessaires, vous pouvez garantir la stabilité, la sécurité et les performances de votre application, ainsi que la satisfaction de vos utilisateurs finaux. Il est également important de noter que Git, un système de gestion de version, est souvent utilisé pour collaborer sur des projets et pour garder une trace de toutes les modifications apportées à un fichier ou à un ensemble de fichiers au fil du temps. Il vous permettra souvent d'être une base avant le déploiement d'une application.

Alors ne vous précipitez pas avant de choisir un hébergement, suivez bien les instructions de l'hébergeur !

V. Auto-évaluation

A. Exercice

Vous êtes en charge de la mise en production d'un site Symfony et vous avez pour tâche d'optimiser les performances de l'application, pour assurer une expérience utilisateur rapide et fluide.

Question 1

[solution n°3 p.15]

Utilisez le composant Cache de Symfony pour mettre en cache les résultats d'une requête de base de données pendant 5 minutes.

Question 2

[solution n°4 p.15]

On vous demande de créer un mémo pour se rappeler de toutes les lignes de commandes nécessaires, pour déployer une application web à partir de Git sur la plateforme Heroku avec un serveur apache et une base de données mysql.

B. Test

Exercice 1 : Quiz

[solution n°5 p.15]

Question 1

Il est recommandé de supprimer tous les fichiers inutiles du dossier public avant la mise en production.

- ○ Vrai
- ○ Faux

Question 2

Qu'est-ce que Git ?

- ○ Git est un déployeur d'application
- ○ Git est un package Symfony
- ○ Git est un système de gestion de version

Question 3

Quelle est la commande pour créer un fichier Procfile ?

- ○ $ heroku create
- ○ $ echo 'web: heroku-php-apache2 public/' >Procfile
- ○ $ git push heroku master

Question 4

Quel est l'intérêt de créer un fichier .htaccess ?

- ○ Configurer le serveur Apache
- ○ Établir une connexion sécurisée
- ○ Optimiser les performances de l'application

Question 5

Quelle est la commande pour déployer une application Symfony sur Heroku à partir de Git ?

- ○ $ heroku create
- ○ $ git push heroku master
- ○ $ heroku login

Solutions des exercices

Exercice p. 6 **Solution n°1**

Question 1

Pourquoi la mise en production d'un site Symfony nécessite-t-elle une préparation minutieuse en amont ?

- [x] Pour garantir une expérience utilisateur optimale
- [ ] Pour garantir une compatibilité avec tous les navigateurs web
- [ ] Pour éviter les conflits entre les développeurs

L'informatique réserve toujours des surprises, le fait de mettre son application en production implique des modifications de son application. Il convient donc de se préparer au mieux.

Question 2

Quels sont les éléments clés à prendre en compte avant de déployer un site Symfony en production ?

- [x] La configuration du serveur, l'optimisation des performances, la sécurité, la gestion des erreurs et des sauvegardes
- [ ] Le choix du nom de domaine, le design du site, la rédaction du contenu et les images
- [ ] La publicité en ligne, le marketing digital et le référencement naturel

Tous ces points sont essentiels à une application en ligne, car chacun des éléments clés mentionnés permet d'assurer le bon fonctionnement d'un site Symfony en production et d'offrir une expérience utilisateur de qualité.

Question 3

Pourquoi est-il important de vider la mémoire cache avant la mise en production ?

- [ ] Pour libérer de l'espace disque sur le serveur
- [x] Pour éviter les conflits avec les données déjà stockées en cache
- [ ] Pour éviter les problèmes de sécurité

Vider la mémoire cache avant la mise en production permet de s'assurer que le site en production utilisera la dernière version des fichiers de configuration et des templates.

Question 4

Qu'est-ce que le caching dans Symfony ?

- [ ] Une technique pour éviter les erreurs de syntaxe dans le code PHP
- [x] Une technique pour stocker en mémoire les résultats de requêtes ou de calculs afin de les réutiliser ultérieurement plutôt que de les recalculer à chaque fois
- [ ] Une technique pour compresser les fichiers JavaScript et CSS

Le caching (mise en cache) dans Symfony est une technique qui consiste à stocker temporairement des données en mémoire ou sur disque, afin d'accélérer l'exécution des requêtes ultérieures.

Question 5

Comment activer le mode production pour une application Symfony avant le déploiement en production ?

- ◉ En définissant la variable d'environnement APP_ENV sur prod
- ○ En utilisant la commande php bin/console server:run --env=prod
- ○ En modifiant le fichier .htaccess sur le serveur web

Q Symfony se sert de cette variable d'environnement pour définir le mode de travail.

Exercice p. 10 **Solution n°2**

Question 1

Quels éléments doivent être pris en compte lors du choix d'un hébergeur pour votre application Symfony ?

- ☑ La fiabilité
- ☑ Les performances
- ☑ Le support client
- ☑ Les fonctionnalités

Q Ces éléments doivent être pris en compte pour choisir un hébergeur fiable et sécurisé.

Question 2

Qu'est-ce que Heroku ?

- ○ Un framework pour Symfony
- ◉ Un hébergeur de site web
- ○ Un outil de développement pour Symfony

Q Heroku est une plate-forme en tant que service, basée sur le cloud (PaaS) qui permet aux développeurs de déployer leurs applications sur un serveur distant.

Question 3

Quelle est l'étape à suivre après avoir préparé une application Symfony pour la mise en production ?

- ○ Rédiger un manuel d'utilisation de l'application
- ○ Tester l'application
- ◉ Déployer l'application sur le serveur final

Q Après avoir préparé une application Symfony pour la mise en production, l'étape suivante consiste à déployer l'application sur le serveur final pour qu'elle soit disponible pour les utilisateurs.

Question 4

Comment créer un compte sur Heroku ?

- ◉ Directement sur le site web de Heroku
- ○ En utilisant la ligne de commande à partir du répertoire de votre projet
- ○ En installant Heroku sur votre ordinateur

Au préalable, il faut créer un compte directement sur le site d'Heroku, après il sera possible de se connecter à partir d'une ligne de commande.

Question 5

Quel est le fichier à créer pour indiquer le type de serveur à utiliser et à partir de quel fichier l'application doit démarrer ?

- ○ .htaccess
- ◉ Procfile
- ○ config.yml

Le fichier Procfile est utilisé pour indiquer le type de serveur à utiliser et à partir de quel fichier l'application doit démarrer lorsqu'elle est déployée sur Heroku.

p. 11 Solution n°3

Voici un exemple de code qui utilise le composant Cache de Symfony pour stocker en cache les résultats d'une requête de base de données pendant 5 minutes :

```php
use Symfony\Component\Cache\Adapter\FilesystemAdapter;
use Symfony\Contracts\Cache\ItemInterface;

$cache = new FilesystemAdapter();
$resultats = $cache->get('maRequete', function (ItemInterface $item) {
    $item->expiresAfter(300); // Durée de vie du cache en secondes (5 minutes)
    // Effectuer la requête ici et retourner les résultats
    return $resultats;
});
```

p. 11 Solution n°4

```
$ heroku login
$ echo ‘web: heroku-php-apache2 public/’ >Procfile
$ composer require symfony/apache-pack
$ heroku addons:create cleardb:ignite
$ heroku config | grep CLEARDB_DATABASE_URL
$ heroku config:set APP_ENV=prod
$ git add .
$ git commit -m “nom du commit”
$ git push origin main
$ git push heroku main
```

Il faut également penser à modifier dans le fichier composer.json “scripts” en ajoutant :

```
“compile”: [
 “php bin/console doctrine:migrations:migrate”
    ]
```

Ainsi, la migration s'exécute lors du déploiement.

Exercice p. 12 Solution n°5

Question 1

Il est recommandé de supprimer tous les fichiers inutiles du dossier public avant la mise en production.

- [x] Vrai
- [ ] Faux

Cela permettra de garder l'environnement de production net.

Question 2

Qu'est-ce que Git ?

- [ ] Git est un déployeur d'application
- [ ] Git est un package Symfony
- [x] Git est un système de gestion de version

Git sert à versionner un projet et permet de gérer les changements apportés à un code source au fil du temps.

Question 3

Quelle est la commande pour créer un fichier Procfile ?

- [ ] $ heroku create
- [x] $ echo 'web: heroku-php-apache2 public/' >Procfile
- [ ] $ git push heroku master

Dans cette commande, on précise quel serveur on souhaite utiliser et à partir de quel dossier l'application doit démarrer.

Question 4

Quel est l'intérêt de créer un fichier .htaccess ?

- [x] Configurer le serveur Apache
- [ ] Établir une connexion sécurisée
- [ ] Optimiser les performances de l'application

Le fichier .htaccess est utilisé pour configurer le serveur Apache, notamment pour définir des règles de redirection, de sécurité et de cache.

Question 5

Quelle est la commande pour déployer une application Symfony sur Heroku à partir de Git ?

- [ ] $ heroku create
- [x] $ git push heroku master
- [ ] $ heroku login

Déployer une application Symfony avec Heroku, à partir de Git, revient à déployer une application sur github ou gitlab.