Créer une App pour Azure (partie 3)

Cloud, CSS, Docker, HTML, PHP

Langage Compétences : 5

La transition digitale a poussé les éditeurs de logiciels et les entreprises à migrer leurs applications sur le web. Les services du cloud Azure ont offert la plateforme idéale, les services adéquats, ainsi que les outils, pour réaliser un développement vers le tout numérique en gardant le contrôle total sur les données. Cette publication propose la création d'une application simple en découvrant certains de ces services et outils.

Publié mercredi 22 juin 2022, 17h43
Modifié lundi 26 août 2024, 10h04

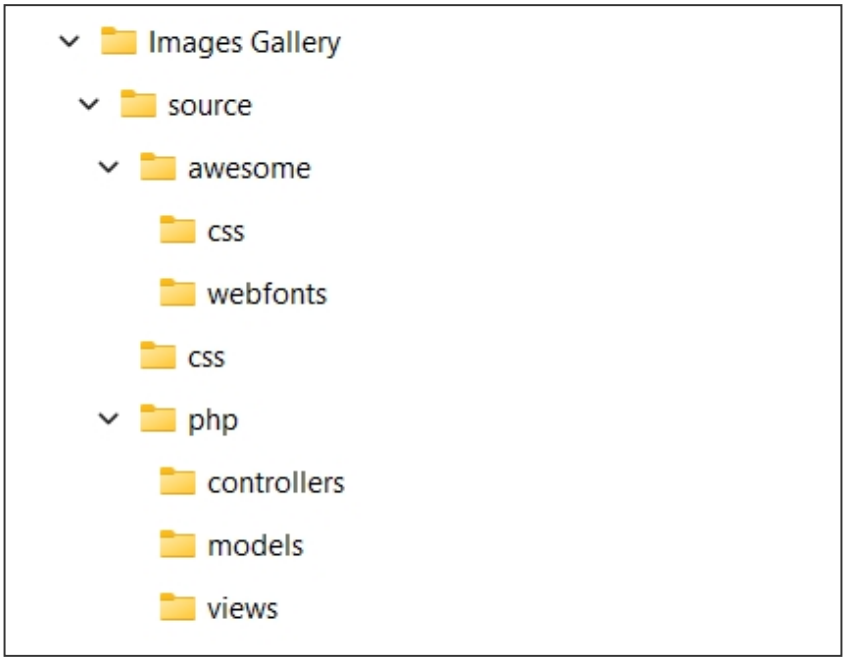
By Olivier Paudex

Le projet Images Gallery

> Cette publication est composée de plusieurs partie. Ceci étant la 3ème partie.

Maintenant que l'outil de développement est en place, il est temps de se lancer dans cette belle aventure.

- L'application va utiliser la structure de dossiers ci-dessous.
- Cliquez sur le lien **"Pièce jointe"**, en haut de cette publication pour télécharger le fichier **"ZIP"** joint à cette publication.
- Décompressez-le sur votre bureau.



La structure des dossiers de l'application

Le dossier Images Gallery

C'est le dossier racine du projet.

- Il contient le fichier **"dockerfile"** qui va nous permettre de créer le container Docker.
- Il contient le fichier **"composer.json"** qui va installer les librairies **SDK de Microsoft** et d'accès au Keyvault.

Le dossier source

C'est le dossier contenant tout le code du projet

- Il contient un fichier **"index.php"** et d'autres sous-dossiers. **"Index.php"** est le fichier de démarrage. Il a pour charge de regrouper toutes les fonctionnalités de l'application, ainsi que de gérer les éventuelles erreurs.

Le dossier awesome

- Ils permettent d'afficher des icônes sous la forme d'une police d'écriture.

Le dossier css

Le dossier **"css"** contient la feuille de style **"styles1.css"**.

- C'est le fichier qui prend en charge l'affichage des différents éléments.

Le dossier controllers

Le dossier contient un fichier **"controller.php"**.

- C'est l'aiguillage. Il a pour but d'appeler les différentes méthodes et fonctions.
- Pour chaque enregistrement de la base de donnée, une image stockée est associée.

Le dossier models

Le dossier **"models"** comprend quatre fichiers qui contiennent toutes les fonctions nécessaires à l'application.

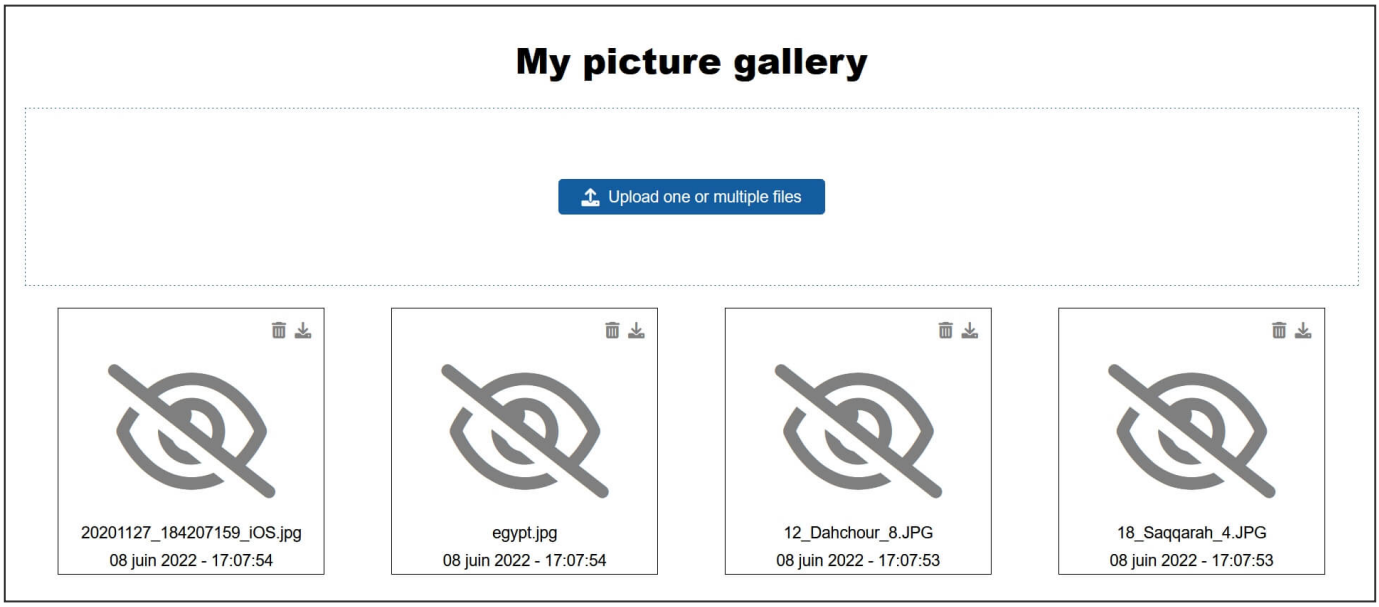
- Le fichier "**BlobManager.php**" permet de gérer le stockage sur Azure.
- Le fichier **"FileManager.php"** va permettre de manipuler les fichiers sélectionnées depuis la variable globale **"$_FILES"**.
- Le fichier **"ImageManager.php"** regroupe toutes les requêtes SQL.
- Le fichier **"Manager.php"** va permettre de se connecter à la base de données SQL, au stockage, ainsi qu'au KeyVault d'Azure, sorte de coffre-fort qui va permettre de gérer les éléments confidentiels.

> Le fichier **Manager.php** contient le nom du compte de stockage qu'il va falloir modifier car celui-ci doit être **UNIQUE** au monde.

Le dossier views

Le dossier **"views"** comprend trois fichiers qui vont gérer l'affichage des informations à l'écran.

- Le fichier **"ImagesView.php"** va s'occuper d'afficher l'écran principale de l'application.
- Pour chaque enregistrement de la base de données, l'application va afficher l'image associée sous forme de vignette.
- Si une erreur survient, une icône sous forme d'un **"oeil barré"** va s'afficher indiquant qu'il y a un problème de lecture de l'image ou de connexion avec le stockage.



Quand il y a une erreur !

- Le fichier **"UploadView.php"** va afficher un statut du téléversement des images avant affichage.
- Le fichier **"ErrorView.php"** va afficher les éventuelles erreurs.

La création du container Docker

La création d'une application contenant un seul container se fait à l'aide d'un fichier**"dockerfile"**, sans extension.

- Ouvrez le fichier **"Manager.php"**.
- Modifiez le nom du compte de stockage. Dans l'exemple ci-dessous, il s'agit de **"stimagesgallerywesteu001"**. Remplacez-le par un nom à votre goût.
- Sauvegardez le fichier php.

```
class StorageManager {
    // Connexion to Azure storage account
    public function storageConnect() {

        // Azure String connection
        $accountName = 'stimagesgallerywesteu001';
        $accountKey = getKeyVaultSecret('key-imagesgallery-storage');
        $connectionString = "DefaultEndpointsProtocol=https;AccountName=" . $accountName . ";Ac
        // Create blob client.
        $blobClient = BlobRestProxy::createBlobService($connectionString);

        return $blobClient;
    }  }
```

- Ouvrez **VSCode** et cliquez sur l'icône **Docker** dans la marge de gauche.
- Ouvrez un Terminal **Powershell** depuis le menu de **VSCode**.
- Placez-vous dans le dossier **"Image Gallery"**.
- Exécutez la commande ci-dessous. (N'oubliez pas le point qui représente le dossier actuel).

```
docker build -t images-gallery .
```

Voici les tâches effectuées par **Docker**.

- Le nom de l'image sera **"images-gallery"**.
- Le système d'exploitation est Debian version 11.
- Le PHP est en version 7.4.
- Docker va installer les packages **"git"**, **"zip"**, **"unzip"** et **"gnupg2"**.
- Docker va installer les drivers ODBC et les outils pour MSSQL Server.
- Docker va créer un fichier **"php.ini"**.
- Docker va copier les fichiers source de l'application.
- Docker va installer le **"SDK PHP pour Azure"** à l'aide de **"Composer"**.
- Docker va installer la librairie **"Az-KeyVault-PHP de Wapacro"** à l'aide de **"Composer"**.

Conclusion

La partie sur l'infrastructure de l'application et des fichiers le composant est maintenant terminée. Il a permis de se familiariser avec la strcture des dossiers et leurs contenus.

Cette partie a également couvert la création d'un container avec Docker.

Le prochain chapitre va être axé sur la création de l'infrastructure sur le cloud Azure, y compris l'utilisation du registre pour le container.