

Create an App for Azure (part 3)

Cloud, CSS, Docker, HTML, PHP

Language Skills : 5

The digital transition has pushed software companies and enterprises to migrate their applications to the web. Azure cloud services have provided the ideal platform, the right services, and the tools to go digital while keeping full control over the data. This publication proposes the creation of a simple application by discovering some of these services and tools.

Published Wednesday June 22nd 2022, 17:43
Modified Monday August 26th 2024, 10:04



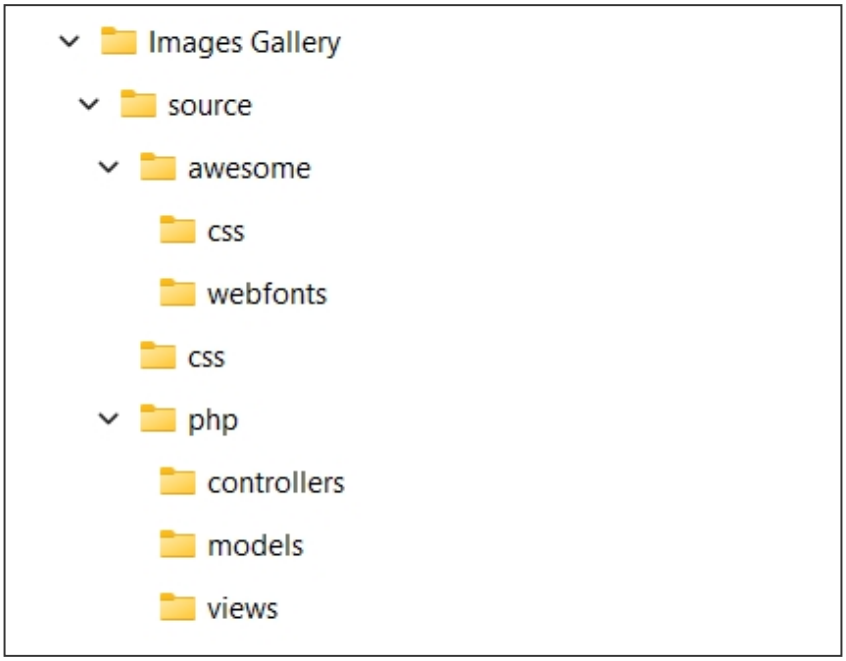
By Olivier Paudex

The Images Gallery project

> This publication is composed of several parts. This being the 3rd part.

Now that the development tool is in place, it's time to get started on this great adventure.

- The application will use the folder structure below.
- Click on the **"Attachment"** link, at the top of this post to download the **"ZIP"** file attached to this post.
- Unzip it to your desktop.



The application folder structure

The Images Gallery folder

This is the root folder of the project.

- It contains the **"dockerfile"** file that will allow us to create the Docker container.
- It contains the **"composer.json"** file that will install the Microsoft **SDK** and Keyvault access libraries.

The source folder

This is the folder containing all the project code

- It contains a **"index.php"** file and other subfolders. **"Index.php"** is the startup file. It is responsible for gathering all the features of the application, as well as handling any errors.

The awesome folder

- They allow icons to be displayed as a font.

The css folder

The **"css"** folder contains the **"styles1.css"** style sheet.

- This is the file that takes care of displaying the different items.

The controllers folder

The folder contains a **"controller.php"** file.

- This is the spacer. Its purpose is to call the various methods and functions.
- For each record in the database, a stored image is associated.

The models folder

The **"models"** folder includes four files that contain all the functions needed for the application.

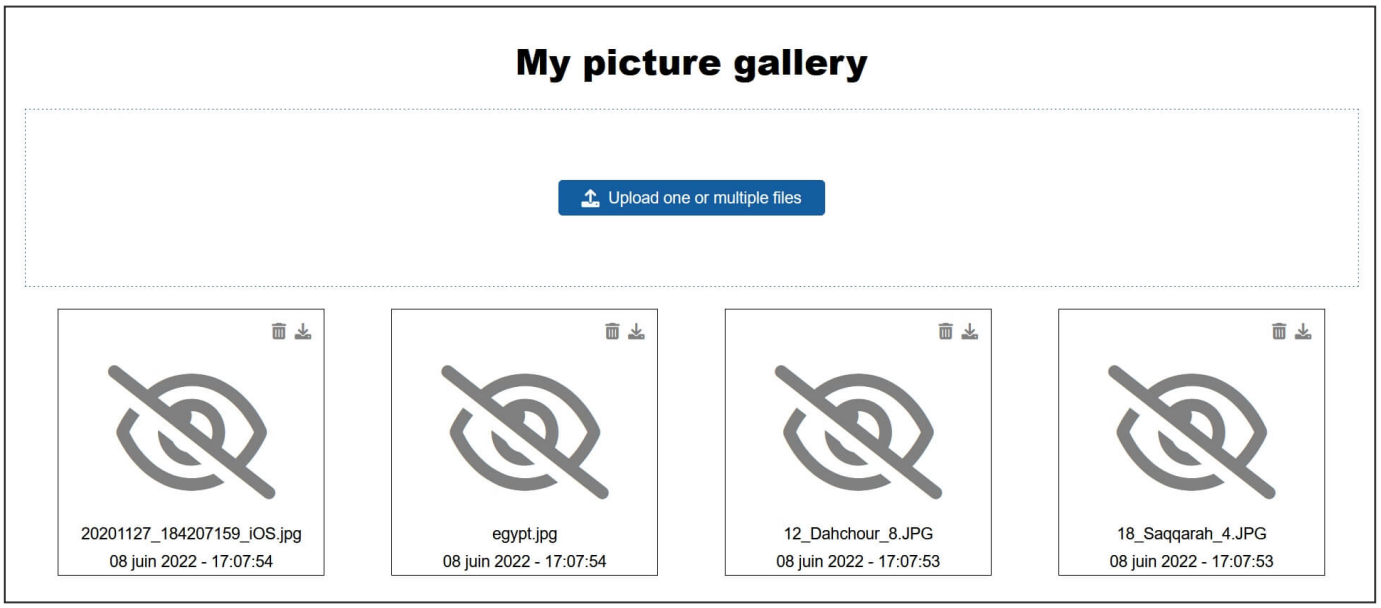
- The file "**BlobManager.php**" allows to manage the storage on Azure.
- The file **"FileManager.php"** will allow to manipulate the files selected from the global variable **"$_FILES"**.
- The file **"ImageManager. php"** gathers all the SQL queries.
- The **"Manager.php"** file will allow to connect to the SQL database, to the storage, as well as to the Azure KeyVault, a kind of vault that will allow to manage the confidential items.

> The **Manager.php** file contains the name of the storage account that we're going to have to change because it needs to be **UNIQUE** to the world.

The views folder

The **"views"** folder includes three files that will manage the display of information on the screen.

- The **"ImagesView.php"** file will take care of displaying the main screen of the application.
- For each record in the database, the application will display the associated image as a thumbnail.
- If an error occurs, an icon in the form of a **"crossed-out eye"** will be displayed indicating that there is a problem reading the image or connecting to storage.



When there is an error!"

- The **"UploadView.php"** file will display a status of the image upload before display.
- The **"ErrorView.php"** file will display any errors.

The creation of the Docker container

The creation of an application containing a single container is done using a **"dockerfile"**, without extension.

- Open the **"Manager.php"** file.
- Change the name of the storage account. In the example below, it is **"stimagesgallerywesteu001"**. Replace it with a name to your liking.
- Save the php file.

```
class StorageManager {
    // Connect 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;
    }  }
```

- Open **VSCode** and click on the **Docker** icon in the left margin.
- Open a Terminal **Powershell** from the **VSCode** menu.
- Locate yourself in the **"Image Gallery"** folder.
- Run the command below. (Remember the dot that represents the current folder).

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

Here are the tasks performed by **Docker**.

- The image name will be **"images-gallery"**.
- The operating system is Debian version 11.
- The PHP is version 7.4.
- Docker will install the **"git"**, **"zip"**, **"unzip"** and **"gnupg2"** packages.
- Docker will install ODBC drivers and tools for MSSQL Server.
- Docker will create a **"php. ini"**.
- Docker will copy the application source files.
- Docker will install the **"PHP SDK for Azure"**using **"Composer"**.
- Docker will install the **"Az-KeyVault-PHP by Wapacro"** library using **"Composer"**.

Conclusion

The part about the infrastructure of the application and the files composing it is now over. It has allowed us to become familiar with the structure of the files and their contents.

This part also covered creating a container with Docker.

The next chapter is going to focus on creating the infrastructure on the Azure cloud, including using the registry for the container.