1 Practice 11.1. Controllers and routes

> This practice consists of the first of a series of practices, consisting of a Library application. Each practice will be based on the previous one.
>
> Once done, do the submission via GitHub Classroom as detailed in the end of the practice.

1.1 Application overview

In this application we are going to make a simple **library**. In this library we will have a list of **books** with their corresponding **publishers**. Each book has a publisher and each publisher can have several books.

Later we will make a database for the library, but for now we will use 2 arrays to model the data.

1.2 Exercise 1. Creating the project

Clone the project repository task:

1. You need a GitHub account. Create one if you don't have any.
2. Click on the assignment invitation link. Enter your GitHub credentials and authorize GitHub Classroom to access your GiHub account.
3. Select your name from the "Join the Classroom" list.
4. Accept the assignment.
5. Go to your assignment repository. Copy the repository URL (http or ssh), that is in the green "Code" button.
6. In your computer, create the project:

```
symfony new library-app
```

7. Enter into your project folder and add the remote repository which URL have you copied in the point 5:

```
cd library-app
git remote add origin your_repository_name
git branch -M main
git push -u origin main
```

8. Verify that your code has been uploaded to your repo and start coding.

1.3 Exercise 2. Creating the Service

Create a new PHP class in `src/Service` named **`LibraryData.php`**. This will contain 2 static arrays, for the books and publishers:

```
private static $books = array (
    array("isbn" => "A111B3", "title" => "The Lord of the Rings", "author" =>
    ↪ "J.R.R. Tolkien",
        "pages" => 1536, "pub_date" => "2020-11-03", "publisher" => 1),
    array("isbn" => "C221B6", "title" => "Factotum", "author" => "Charles
    ↪ Bukowski",
        "pages" => 208, "pub_date" => "2002-03-31", "publisher" => 2),
    array("isbn" => "A546783", "title" => "A Wizard of Earthsea", "author" =>
    ↪ "Ursula K. Le Guin",
        "pages" => 210, "pub_date" => "2012-09-11", "publisher" => 1),
    array("isbn" => "F666764", "title" => "The Lathe Of Heaven", "author" =>
    ↪ "Ursula K. Le Guin",
        "pages" => 192, "pub_date" => "2008-04-15", "publisher" => 4),
    array("isbn" => "66788745", "title" => "Foundation", "author" => "Isaac
    ↪ Asimov",
        "pages" => 816, "pub_date" => "2022-07-07", "publisher" => null)
);

private static $publishers = array (
    array ("id"=> 1, "name"=>"Clarion Books", "email"=>"info@clarion.com"),
    array ("id"=> 4, "name"=>"Ecco", "email"=>"ecco_info@ecco.com"),
    array ("id"=> 2, "name"=>"Scribner", "email"=>"scribner@scr.com")
);
```

Create 2 static get methods to get each array.

1.4 Exercise 3. Creating the controllers

In this step you must create 3 controllers:

- MainPageController
- BooksController
- PublishersController

1.5 Exercise 4. Making the Main page view

In the **MainPageController** we will show a welcome message when a user enters the root of the app.

Change the route attribute to:

`#[Route('/', name: 'app_main_page')]`.

Pass the page title to the view as parameter.

In the **Twig template**:

- Use the `path` function to make the links.
- Use the CSS styles you want. You can also add a custom header and footer, preferably in your base template.

The result should be something like:

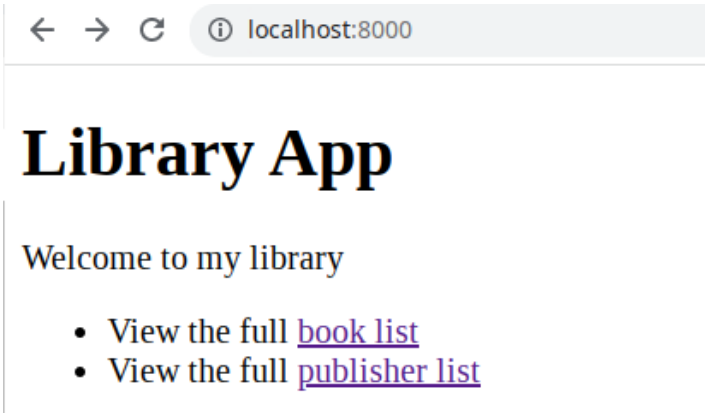

Figure 1: Main page (without styles)

1.6 Exercise 5. Making the single book page view

Change the route attribute to:

`#[Route('/book/{isbn}', name: 'single_book')]`.

Change the view's name related to `BooksController` to `single_book.html.twig`.

Pass the **page title** and the **book array (with the publisher)** as parameters to the view.

Show different messages when the ISBN hasn't been passed and when the ISBN hasn't been found.

The result should be something like:

Figure 2: Single book page

1.7 Exercise 6. Making the single publisher page view

Change the route attribute on `PublisherController` to:

`#[Route('/publisher/{id<\d+>}', name: 'single_publisher')]`.

Change the view's name to `single_publisher.html.twig`.

Pass the **page title** and the **publisher array** as parameters to the view.

Show different messages when the publisher's ID hasn't been passed and when the publisher's ID hasn't been found.

The result should be something like:

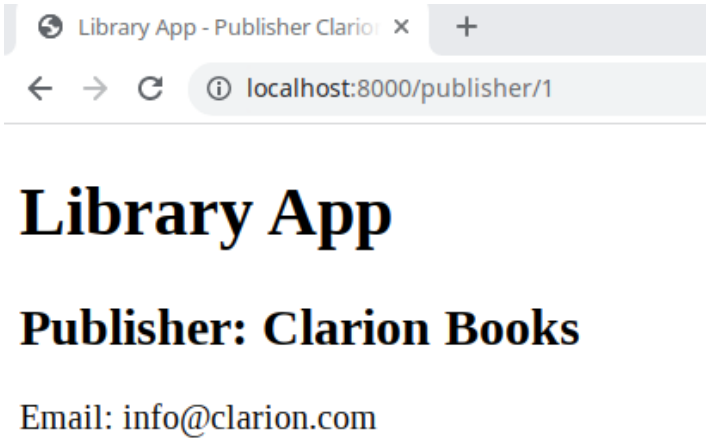

Figure 3: Single publisher page

1.8 Exercise 7. Making the book list page view

Modify the `BooksController` to add a new route:

```
#[Route('/book_list', name: 'book_list')]
```

Create the related view so that they use Twig to render the view. Create a new twig file, inside `templates/books` named `book_list.html.twig`.

For each book we want to show:

- A link for opening the book's page (use the `path` function)
- The book's title
- The author
- The publisher
- The publication data

Pass the page title and the necessary data to view the required data.

The result should be something like:

Library App

Book list

	Title	Author	Publisher	Publication date
Open book	THE LORD OF THE RINGS	J.R.R. Tolkien	Clarion Books	2020-11-03
Open book	FACTOTUM	Charles Bukowski	Scribner	2002-03-31
Open book	A WIZARD OF EARTHSEA	Ursula K. Le Guin	Clarion Books	2012-09-11
Open book	THE LATHE OF HEAVEN	Ursula K. Le Guin	Ecco	2008-04-15
Open book	FOUNDATION	Isaac Asimov		2022-07-07

Figure 4: Book list

1.9 Exercise 8. Creating the publisher list page view

Modify the `PublishersController` to add a new route:

```
#[Route('/publisher_list', name: 'publisher_list')]
```

Create a new twig file, inside `templates/publishers` named `publisher_list.html.twig`.

For each publisher we want to show:

- A link for opening the publishers's page (use the `path` function)
- The publishers's id
- The publishers's name
- The publishers's email

Pass the page title and the necessary data to view the required data.

The result should be something like:

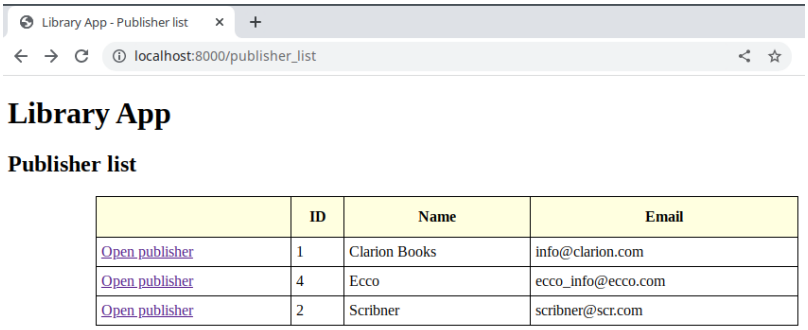

Figure 5: Publisher list

1.10 How to submit to GitHub Classroom

Once you finish the task, make a commit with the comment "SUBMISSION COMMIT" and push it to GitHub. It's recommended to tag your commit with the tag "Practice_11.1".

Before that, you can do the commits and push you want. If you change your code after your submission commit, make another commit and push with the same text in the message adding the corrections you've done.

If you have any doubt in your task, you can push your code and ask me by email what's your problem. It will make it easier for both the solutions of code issues.