Server-side
Web Development

Unit 11. Controllers, routes, views. Guided example.

Fidel Oltra, Ricardo Sánchez

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2024-25

Index

1 Data model **2**

2 Application creation **3**

3 Controllers and routes **3**

4 Services **3**

5 Views **5**

5.1 Main page template and controller 5

5.2 Single contact template and controller 7

5.3 Contact list template and controller 11

In this example we're going to do a Contacts application. It's the first of a series of guided examples.

1 Data model

Our data model is a contact entity which can have none or various phones:

- Contact:
  - id: an integer that identifies the contact
  - title: string; can be "Mr.", "Mrs." or "Miss"
  - name: string
  - surname: string
  - birthdate: string containing a date in "yyyy-mm-dd" format.
  - email: string
- Phone:
  - idContact: integer, the id of the contact which belongs the phone number
  - number: string, the phone number.
  - type: string, can be "Mobile", "Landline" or "Work".

To simulate the storing of the sample data we will use 2 arrays:

```php
array $contacts = array(
    array("id"=> 11, "title"=> "Mr.", "name"=> "Mike", "surname"=> "Molina",
    "birthdate"=>"1975-10-21", "email"=>"molina@mail.com"),
    array("id"=> 12, "title"=> "Mrs.", "name"=> "Mary Jane", "surname"=>
    ↪  "Smith",
    "birthdate"=>"1986-05-22", "email"=>"mj@mail.com"),
    array("id"=> 13, "title"=> "Mr.", "name"=> "Arthur", "surname"=> "McFly",
    "birthdate"=>"1990-08-14", "email"=>"mcfly@mail.com"),
    array("id"=> 9, "title"=> "Miss", "name"=> "Lory", "surname"=> "Grimes",
    "birthdate"=>"1967-02-08", "email"=>"logrimes@mail.com")
);

array $phones = array(
    array("idContact"=>11, "number"=>"666557744", "type"=>"Mobile"),
    array("idContact"=>11, "number"=>"295667788", "type"=>"Landline"),
    array("idContact"=>13, "number"=>"667889900", "type"=>"Work"),
    array("idContact"=>9, "number"=>"664444666", "type"=>"Work"),
    array("idContact"=>9, "number"=>"667889888", "type"=>"Mobile")
);
```

2 Application creation

You can create your app using any way shown in the Unit 10:

```
symfony new contacts_symfony --version="7.1.*"
```

or

```
composer create-project symfony/skeleton:"7.1.*" contacts_symfony
```

3 Controllers and routes

In this step we will create 2 controllers:

- MainPageController, for the main page
- ContactsController, for showing information about one contact identified by its id.

Remember to install first the **maker bundle** and **Twig**:

```
composer require --dev symfony/maker-bundle

symfony composer require twig

symfony console make:controller MainPageController

symfony console make:controller ContactsController
```

You can check your routes starting the symfony server and going to the urls

4 Services

Before creating our first views, we need to retrieve the contact data, which we have stored in arrays.

To do this we will use a service and create the two arrays as properties of the ContactsData class, using get methods to retrieve them:

```php
<?php
//src/Service/ContactData.php
namespace App\Service;

class ContactData
{
    private static array $contacts = array(
        array("id"=> 11, "title"=> "Mr.", "name"=> "Mike", "surname"=>
        ↪   "Molina",
            "birthdate"=>"1975-10-21", "email"=>"molina@mail.com"),
        array("id"=> 12, "title"=> "Mrs.", "name"=> "Mary Jane", "surname"=>
        ↪   "Smith",
            "birthdate"=>"1986-05-22", "email"=>"mj@mail.com"),
        array("id"=> 13, "title"=> "Mr.", "name"=> "Arthur", "surname"=>
        ↪   "McFly",
            "birthdate"=>"1990-08-14", "email"=>"mcfly@mail.com"),
        array("id"=> 9, "title"=> "Miss", "name"=> "Lory", "surname"=>
        ↪   "Grimes",
            "birthdate"=>"1967-02-08", "email"=>"logrimes@mail.com")
    );
    private static array $phones =array(
        array("idContact"=>11, "number"=>"666557744", "type"=>"Mobile"),
        array("idContact"=>11, "number"=>"295667788", "type"=>"Land line"),
        array("idContact"=>13, "number"=>"667889900", "type"=>"Work"),
        array("idContact"=>9, "number"=>"664444666", "type"=>"Work"),
        array("idContact"=>9, "number"=>"667889888", "type"=>"Mobile")
    );

    public static function getContacts(): array
    {
        return self::$contacts;
    }

    public static function getPhones(): array
    {
        return self::$phones;
    }
}
```

5 Views

5.1 Main page template and controller

First of all, let's change our `base.html.twig` file to link to our css style, that's stored in `public/css/style.css`. We will use the `asset` function:

```
{% block stylesheets %}
    <link href="{{ asset('css/style.css') }}" rel="stylesheet" />
{% endblock %}
```

> **Note**: Before using the `asset` function, you probably will need to install it. Run:

```
composer require symfony/asset
```

Feel free to use the styles you prefer.

We also want to change the page title for all our views. To do that, change the line 5 of the `base.html.twig` file to:

```
<title>{% block title %}{{ page_title }}{% endblock %}</title>
```

Remember to remove the title block from the child views so that they use the base title block.

Now you can change or remove the icon and the script tag. The base file could be something like:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>{% block title %}{{ page_title }}{% endblock %}</title>

        {% block stylesheets %}
            <link href="{{ asset('css/style.css') }}" rel="stylesheet" />
        {% endblock %}

        {% block javascripts %}
        {% endblock %}
    </head>
    <body>
```

```
        {% block body %}{% endblock %}
    </body>
</html>
```

In the `MainPageController.php` controller we will create the page that the app will return when a user loads the page root (/).

```
<?php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class MainPageController extends AbstractController
{
    #[Route('/', name: 'app_main_page')]
    public function index(): Response
    {
        return $this->render('main_page/index.html.twig', [
            'page_title' => 'My Contacts App - Main page',
        ]);
    }
}
```

In this controller we are sending the `page_title` variable with the title to show in the browser bar.

Then, in our `index.html.twig` file, we want to modify the view, in a similar way that in the previous practice, but using twig:

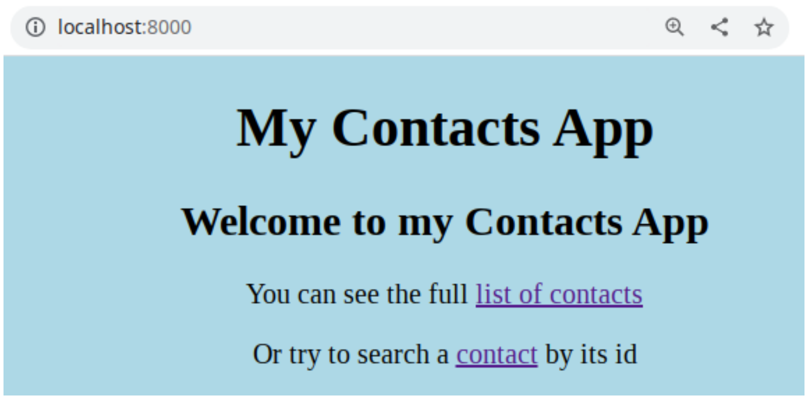
```
{% extends 'base.html.twig' %}
{% block body %}

<main>
    <h1>My Contacts App</h1>
    <h2>Welcome to my Contacts App</h2>
    <p>You can see the full  <a href ="">list of contacts</a></p>
    <p>Or try to search a <a href ="{{ path('app_contacts') }}">contact</a>
     ↪  by its id</p>
</main>
```

```
{% endblock %}
```

You can observe how we are using the `path` function to link to the route name `app_contacts` that leads to the related route in the `ContactsController`.

We'll return later to change the link to the contacts list page.

The result could be something like:



localhost:8000

My Contacts App

Welcome to my Contacts App

You can see the full list of contacts

Or try to search a contact by its id

Figure 1: Main page

5.2 Single contact template and controller

First of all we're going to change the controller so that it sends to the view the necessary data needed to show a single contact information:

- An associative array with the contacts data
- An array with one associative array per phone number of the contact
- The page title

In `ContactsController.php` change the names of the route and of the method, to add the contact id and to make them more meaningful:

```
#[Route('/contact/{id<\d+>}', name: 'single_contact')]
public function contact($id=''): Response
```

We need the data stored in the service ContactData, so we import it using its namespace:

```
use App\Service\ContactData;
```

Inside the `contact` method, the first thing is to search the contact in the array of contacts:

```
//Search the contact in the array
$contacts = ContactData::getContacts();
$contact = Array();
foreach ($contacts as $c){
  if($c['id'] == $id) {
      $contact = $c;
      break;
  }
}
```

Then, we check if we have an `id` and if we have a valid `id`, get the contact data in the `$contact` array. We also get the phones for the selected contact into an array:

We can construct the arrays like in the previous example:

```
$contact = array();

//Search the contact in the array
$contacts = ContactData::getContacts();
foreach ($contacts as $c){
    if($c['id'] == $id) {
        $contact = $c;
        break;
    }
}

//Check if we have an id and a contact
if(empty($id) ){
    $contact = null;
} elseif (empty($contact)) {
    $contact['id'] = $id;
    $contact['name'] = null; //Set the contact's name to null to check if
    ↪ the contact exists
    $contact['phones'] = array();
} else {
    $phones = ContactData::getPhones();
    $contactPhones = array();
    //We need only the phone of a single contact
    foreach ($phones as $phone) {
        if ($phone['idContact'] == $id) {
            $contactPhones[] = $phone;
        }
    }
    $contact['phones'] = $contactPhones;
}
```

Finally, call the render function to pass the data to the template. Note that we have changed the template's name to `contact.html.twig`:

```
return $this->render('contacts/contact.html.twig', [
    'contact' => $contact,
    'page_title' => 'My Contacts App - Contact',
]);
```

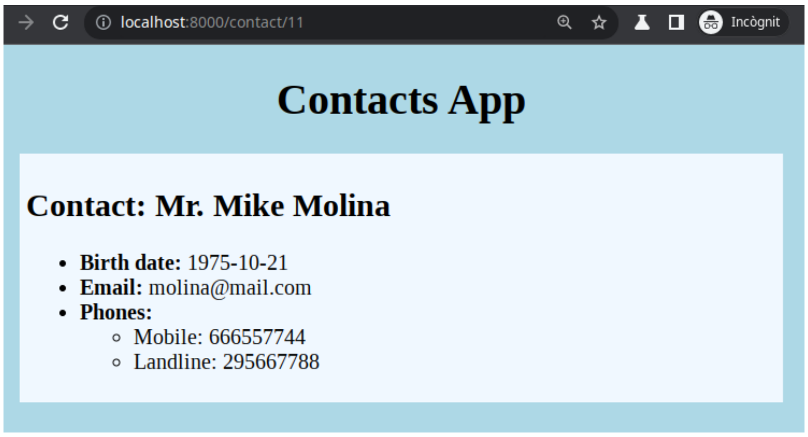
Then, go to the template (remember to rename/refactor it to `contact.html.twig`) and modify it:

```
{% extends 'base.html.twig' %}

{% block body %}

    <h1 class="centered">Contacts App</h1>

    {%  if contact == null %}
        <h2>Please, enter a contact Id</h2>
    {%  elseif contact.name == null %}
        <h2>No contact found with id {{ contact.id }} </h2>
    {%  else %}
        <main class="contact">
            <h2>Contact: {{ contact.title}} {{ contact.name }} {{
             ↪  contact.surname}} </h2>
            <ul>
                <li><strong>Birth date:</strong> {{ contact.birthdate }}</li>
                <li><strong>Email:</strong> {{ contact.email }} </li>
                <li><strong>Phones:</strong>
                    <ul>
                        {% for phone in contact.phones %}
```

```
                                        <li> {{ phone.type }}: {{ phone.number }} </li>
                                    {% endfor %}
                                </ul>
                            </li>
                        </ul>
                    </main>
    {% endif %}

{% endblock %}
```

Note how we are using the `if` and `for` sentences to generate the view dynamically based on the data, checking if the ID is null or if the ID is not found (in this case we pass the invalid ID, but a null value for the name).

The result view:



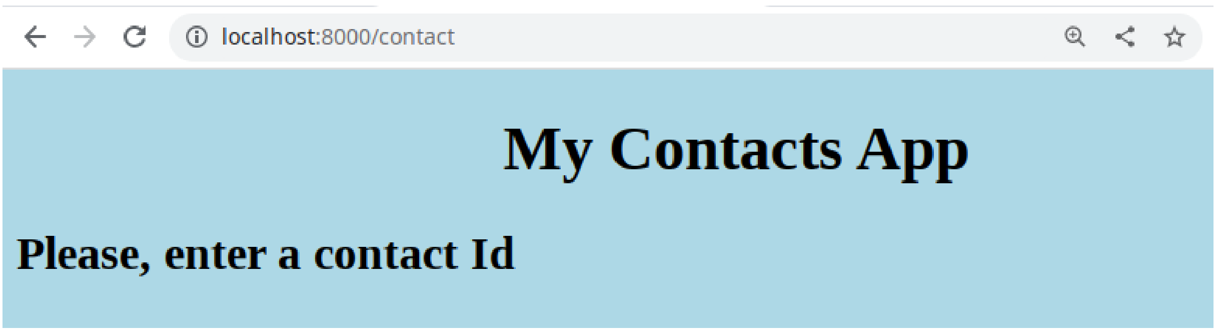
Contacts App

Contact: Mr. Mike Molina

- **Birth date:** 1975-10-21
- **Email:** molina@mail.com
- **Phones:**
  - Mobile: 666557744
  - Landline: 295667788

Figure 2: Single contact

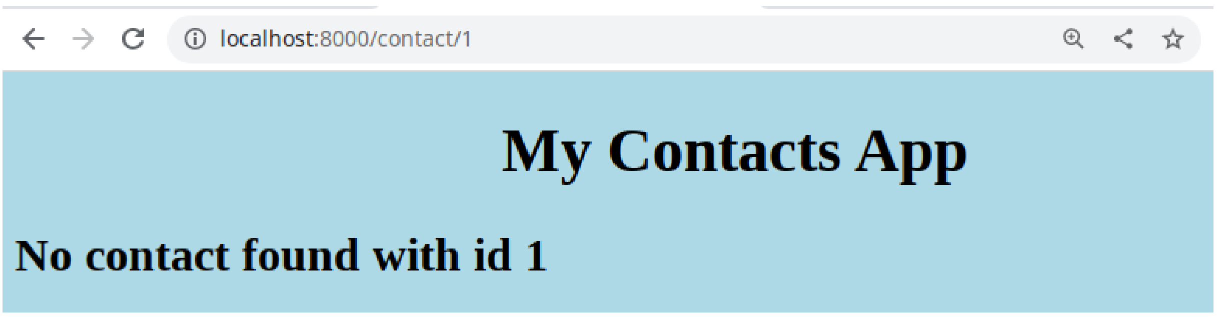
We can also enter no id or invalid ids:



My Contacts App

Please, enter a contact Id

Figure 3: No id


localhost:8000/contact/1

My Contacts App

No contact found with id 1

Figure 4: Invalid id

5.3 Contact list template and controller

To make the controller for the contact list page, we will use the same class for the previous one, `ContactsController`, but adding a new method to control the new route and to render a new template:

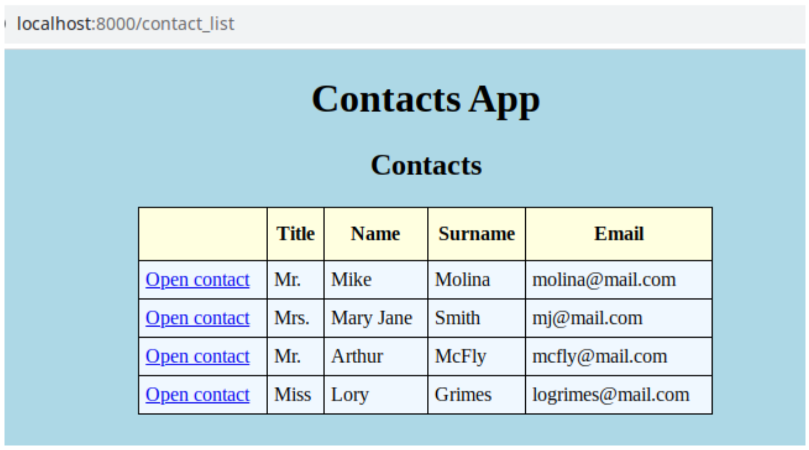
```
#[Route('/contact_list', name: 'contact_list')]
public function contactList(): Response
{
    return $this->render('contacts/list.html.twig', [
        'contacts' => ContactData::getContacts(),
        'page_title' => 'My Contacts App - Contact List'
    ]);
}
```

This method is pretty simple: it simply passes the list of contacts and the page title to the render.

We need a new template, placed in `templates/contacts`, named `list.html.twig`. In it we make a table using a `for` loop, creating a link in each row for opening the correspondent contact using the `path` function:

```
{% extends 'base.html.twig' %}
{% block body %}
    <h1 class="centered">Contacts App</h1>
    {%  if contacts == null %}
        <h2>No contacts found</h2>
    {%  else %}
        <main>
            <h2 class="centered">Contacts</h2>
            <table>
```

```
                <tr>
                    <th></th>
                    <th>Title</th>
                    <th>Name</th>
                    <th>Surname</th>
                    <th>Email</th>
                </tr>
                {% for contact in contacts %}
                <tr>
                    <td><a href="{{ path('single_contact', { id: contact.id
                     ↪   }) }}">Open contact</a></td>
                    <td>{{ contact.title }}</td>
                    <td>{{ contact.name }}</td>
                    <td>{{ contact.surname }}</td>
                    <td>{{ contact.email }}</td>
                </tr>
                {% endfor %}
            </table>
        </main>
    {% endif %}
{% endblock %}
```


localhost:8000/contact_list

Contacts App

Contacts

	Title	Name	Surname	Email
Open contact	Mr.	Mike	Molina	molina@mail.com
Open contact	Mrs.	Mary Jane	Smith	mj@mail.com
Open contact	Mr.	Arthur	McFly	mcfly@mail.com
Open contact	Miss	Lory	Grimes	logrimes@mail.com

Figure 5: Contact list

And, remember to change the link path in the `index.html.twig` to link to the list page:

```
<p>You can see the full  <a href ="{{ path('contact_list')}}">list of
 ↪  contacts</a></p>
```

You can get all the code from the GitHub repository.