Desarrollo de aplicaciones con Symfony 4

Formularios

Instalar el componente

En la aplicaciones que dispongan de symfony flex y que no hayan sido instaladas con **symfony/website-skeleton,** donde este componente viene por defecto, ejecutaremos:

```
$ composer require symfony/form
```

Creando un formulario simple

Aunque no lo vamos a hacer nunca, se puede crear el formulario dentro de la acción de nuestro controlador:

```
...
$form = $this->createFormBuilder($task)
        ->add('title', TextType::class)
        ->add('description', TextareaType::class)
        ->add('save', SubmitType::class, ['label' => 'CrearTask'])
        ->getForm();
```

y para pasárselo a la vista:

```
    'form' => $form->createView(),
```

Renderizando el formulario en la vista

Una vez creado el formulario, el siguiente paso es renderizarlo en la vista.

Esto se hace pasando un objeto de "vista de formulario" al tamplate ($ form-> createView () de nuestro controlador) y usando un conjunto de funciones de twig:

{{ form(form) }}

Manejo de envío de formularios

Veamos cómo gestionaríamos el envío de un formulario:

```
$task = new Task();
$form = $this->createFormBuilder($task)
      ->add(title, TextType::class)
…
$form->handleRequest($request);
iif ($form->isSubmitted() && $form->isValid()) {
     $task = $form->getData();
```

Manejo de envío de formularios

```
$task = new Task();
$this->createFormBuilder($task)
    ...
```

Aquí creamos el objeto Task y creamos un formulario asociándolo al nuevo objeto.

Manejo de envío de formularios

```
$form->handleRequest($request);
```

En la carga inicial cuando no ha habido submit:

Al cargar inicialmente la página en un navegador, el formulario se crea y se representa. handleRequest () reconoce que el formulario no se envió y no hace nada. isSubmitted () devuelve false si el formulario no fue enviado.

Manejo de envío de formularios

```
$form->handleRequest($request);
```

Cuando el usuario envía el formulario:

handleRequest () lo reconoce e inmediatamente escribe los datos enviados por post en objeto Task.

isSubmitted () devuelve true. y el objeto es validado. Si no es válido, isValid () devuelve false y el formulario se vuelve a mostrar, pero ahora con errores de validación;

Manejo de envío de formularios

Si queremos obtener los datos enviados en el submit, podemos hacerlo mediante:

```
$task = $form->getData();
```

El objeto Type

Como ya os dije al principio, el formulario se puede crear en el controller, pero no es una práctica recomendada.

Como ya vimos en el CRUD creado con el maker deberemos crear un objeto Type que extienda de AbstractType y que deberá ir en la carpeta src/Form/ de nuestro proyecto.

El objeto Type

Como ya explicamos en la introducción, Symfony nos provee de un componente que automatiza los procesos repetitivos como puede ser crear un controlador. Así que simplemente ejecutaremos el siguiente comando para generar la entidad

```
$ php bin/console make:form
```

El objeto Type

<< Ver la clase TaskType en phpStorm >>

Definir los campos del formulario

```
$builder
        ->add('title')
        ->add('description')
```

De esta forma, tomará todas las diferentes opciones de la entity que hemos indicado en:

```
public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'data_class' => Task::class,
    ]);
}
```

Definir los campos del formulario

```
$builder
        ->add('title')
        ->add('description')
```

De esta forma, tomará todas las diferentes opciones de la entity que hemos indicado en:

```
public function configureOptions(OptionsResolver $resolver)
{
    $resolver->setDefaults([
        'data_class' => Task::class,
    ]);
}
```

Definir los campos del formulario

¿Que pasa si queremos definir un tipo de input diferente o añadirle opciones? Muy sencillo, si vemos el metodo add que estamos utilizando para añadir campos al formulario:

...

```
public function add($child, $type = null, array $options = []);
```

...

Podemos ver los diferentes campos y las opciones aceptadas en:

https://symfony.com/doc/current/forms.html#built-in-field-types

Definir los campos del formulario

Veamos un ejemplo:

```
$builder
        ->add(
           'title',
           TextType::class,
           array(
                'label' => 'Título de la tarea',
                'required' => false,
                )
           );
```

Renderizando el formulario en la vista

Ya hemos visto anteriormente que con {{ form(form) }} redenrizábamos todo el formulario, pero tenemos otras opciones:

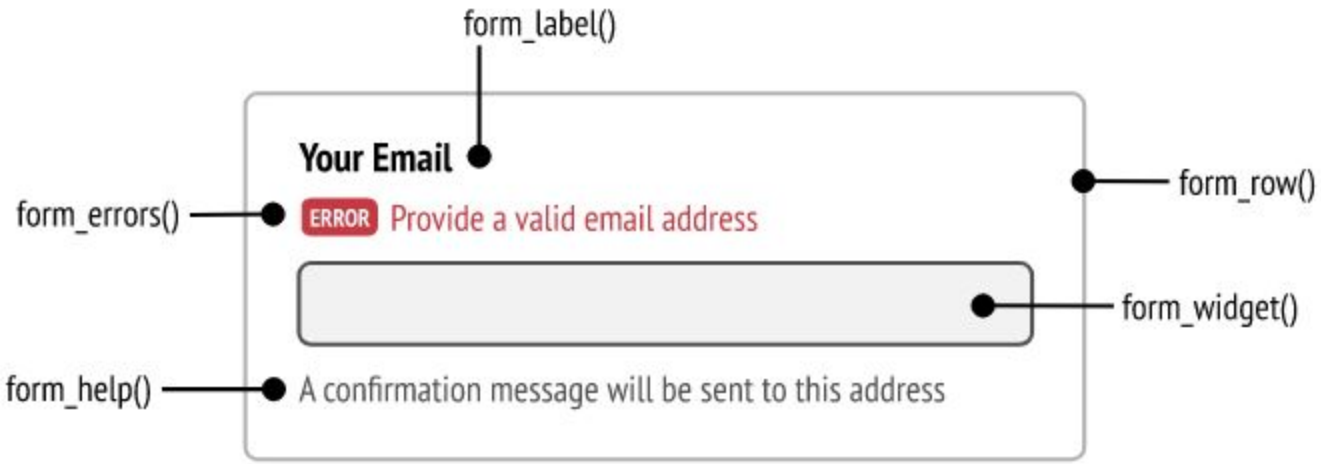
```
{{ form_start(form) }}
    {{ form_row(form.title) }}
    {{ form_row(form.descriptcion) }}
{{ form_end(form) }}
```

Renderizando el formulario en la vista

Ya hemos visto anteriormente que con {{ form(form) }} redenrizábamos todo el formulario, pero tenemos otras opciones:

```
{{ form_start(form) }}
    {{ form_errors(form.title) }}
    {{ form_label(form.title) }}
    {{ form_widget(form.title) }}
    ...
{{ form_end(form) }}
```

Renderizando el formulario en la vista



AvaiBook BOOKING

Symfony

Renderizando el formulario en la vista

También podemos añadir atributos a los tags que se van a rederizar:

```
{{ form_start(form, {'action': path('target_route'), 'method': 'GET'}) }}
```

```
{{ form_widget(form.author, { attr: { class: 'form-control' } }
```

Para que un campo del Type no se pinte en el twig hay que indicárselo:

```
{% do form.description.setRendered %}
```

Form themes

Symfony nos provee de una serie de temas predefinidos que nos ayudan a mejorar la imagen de nuestros formularios de una forma sencilla.

¿Cómo se usan?

Por defecto en Symfony trabaja con el tema form_div_layout.html.twig

{% form_theme form 'bootstrap_4_layout.html.twig' %}

o definiéndolo global para toda la aplicación

```
# config/packages/twig.yaml
twig:
    form_themes: ['bootstrap_4_horizontal_layout.html.twig']
    # ...
```

Form themes

Symfony nos provee de una serie de temas predefinidos que nos ayudan a mejorar la imagen de nuestros formularios de una forma sencilla.

¿Cómo se usan?

Por defecto en Symfony trabaja con el tema form_div_layout.html.twig

{% form_theme form 'bootstrap_4_layout.html.twig' %}

o definiéndolo global para toda la aplicación

```
# config/packages/twig.yaml
twig:
    form_themes: ['bootstrap_4_horizontal_layout.html.twig']
    # ...
```

Form themes

Symfony nos provee de una serie de temas predefinidos que nos ayudan a mejorar la imagen de nuestros formularios de una forma sencilla.

¿Cómo se usan?

Por defecto en Symfony trabaja con el tema form_div_layout.html.twig

{% form_theme form 'bootstrap_4_layout.html.twig' %}

o definiéndolo global para toda la aplicación

```
# config/packages/twig.yaml
twig:
    form_themes: ['bootstrap_4_horizontal_layout.html.twig']
    # ...
```

Más información sobre formularios

https://symfony.com/doc/current/components/form.html#learn-more

¿Preguntas?