Configuración e Instalación del proyecto

1. Instalación de herramientas
   1. Instalar servidor de app web (XAMPP)
   2. Composer
   3. Versión de php 8.1
   4. yarn 1.22 (Para bootstrap)
2. Instalación de una web completa
   1. ***composer create-project symfony/skeleton:"6.4.\*" processar-web***
   2. ***cd processar-web***
   3. ***composer require webapp***
3. Instalar paquete para que symfony trabaje con servidor Apache (en el directorio del proyecto)
   1. ***composer require symfony/apache-pack***
4. Crear un Host virtual
   1. Editar ***C:\xampp\apache\conf\extra\httpd-vhosts.conf***
   2. Editar ***C:\Windows\System32\drivers\etc\hosts***
      1. Agregar 127.0.0.1 processar.com.ar.devel
   3. Reiniciar servicio Apache para que tome los cambios
5. Configurar Controladores y Twig
   1. ***composer require twig*** (Para el manejo de las vistas)
   2. Copiar el archivo ***.htaccess*** a la carpeta: ***/public*** (si no está)
6. Instalación de Bootstrap, (en el directorio del proyecto)
   1. ***yarn -v*** (Para ver que este instalado yarn, y cual es su versión)
   2. ***composer require symfony/webpack-encore-bundle***
   3. ***yarn install*** (para que actualice)
   4. En el archivo ***webpack.config.js*** descomentar la línea
      1. ***.enableSassLoader()***
   5. En el directorio:  ***//assets/styles/*** modifico los archivos para que tengan extensión **scss**
   6. ***yarn encore dev --watch*** (para ver que problemas o que cosas faltan instalar)
   7. Instalar ***yarn add sass-loader@^13.0.0 sass --dev***
   8. ***yarn dev --watch*** (compruebo que este todo bien)
   9. ***npm install bootstrap --save-dev***
   10. agrego ***@import "~bootstrap/scss/bootstrap";*** al app.scss
   11. ***npm install jquery @popperjs/core --save-dev***
   12. ***npm install -g sass***
   13. En el archivo: *//app.js* agrego abajo de ***import './styles/app.scss';***
       1. ***const $ = require('jquery');***

***// this "modifies" the jquery module: adding behavior to it***

***// the bootstrap module doesn't export/return anything***

***require('bootstrap');***

* 1. En el archivo ***base.html.twig*** agregar antes de los bloques:
     1. ***<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet" >***
     2. ***<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" ></script>***
     3. ***<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.min.js" integrity="sha384-Atwg2Pkwv9vp0ygtn1JAojH0nYbwNJLPhwyoVbhoPwBhjQPR5VtM2+xf0Uwh9KtT" crossorigin="anonymous"></script>***

1. Configurar ***.env*** con los datos de la base de datos
   1. ***DATABASE\_URL="mysql://root@127.0.0.1:3306/processar\_web"***
2. Configuración Doctrine:
   1. ***composer require symfony/orm-pack***
   2. ***composer require --dev symfony/maker-bundle***
3. Configurar Anotaciones
   1. Eliminar de ***config/packages/framework.yaml*** la línea ***annotations:false***
   2. Ejecutar: ***composer require doctrine/annotations***
   3. Revisar que en ***config/packages/doctrine.yaml*** no tenga declarado que usa atributos
4. Creo la base de datos
   1. ***php bin/console doctrine:database:create***
5. Creo las clases dentro de src/Entity
   1. Para el proyecto, use las entidades que había creado antes con las respectivas relaciones y después ejecute los comandos:
      1. Generar los Gets y Sets: ***php bin/console make:entity --regenerate***
      2. ***php bin/console doctrine:migrate:diff***
      3. ***php bin/console doctrine:migrate:migrate***
   2. Para generar nuevas entidades, las creo con:
      1. ***php bin/console make:entity <Nombre\_entidad>***
      2. Generar los Gets y Sets: ***php bin/console make:entity --regenerate***
      3. ***php bin/console doctrine:migrate:diff***
      4. ***php bin/console doctrine:migrate:migrate***
   3. Comparo con el diagrama de tablas que había hecho y corrijo las diferencias desde entity

Estructura del proyecto Symfony

1. assets
2. bin:
   1. console
   2. prueba unitaria
3. config
   1. packages: directorio que almacena la configuración de cada paquete instalado en la aplicación.
   2. ***routes.yaml***: el archivo define la [configuración de](https://symfony.com/doc/6.4/routing.html) rutas
   3. ***bundles.php***: permite activar/desactivar paquetes en la aplicación
   4. ***services.yaml***: define la configuración de servicios
4. migrations
5. public
   1. build
   2. ***index.php***
   3. htaccess
6. src
   1. ***kernel.php***
   2. command
   3. controllers
   4. datafixtures
   5. entity
   6. eventsubscriber
   7. form
   8. repository
   9. security
   10. twig
7. templates
8. test
9. translations
10. var
    1. cache
    2. log
11. vendors: Se instalan paquetes y dependencias
    1. ***composer.json***: se indican las versiones de los programas o funciones utilizadas
    2. ***.env***
    3. ***.gitignore***

Bases de Datos y Doctrine

Doctrine (ORM) <https://symfony.com/doc/current/doctrine.html> – Ejemplos en el proyecto del curso Aprendiendo-Symfony. Varios de estos puntos están en AnimalController

1. Configuración Doctrine:
   1. ***composer require symfony/orm-pack***
   2. ***composer require --dev symfony/maker-bundle***
2. Creación de base de datos.
   1. En el archivo ***.env*** indicar los datos de la base de datos
      1. Ej: ***DATABASE\_URL="mysql://root@127.0.0.1:3306/aprendiendo-sf4"***
   2. Desde la línea de comandos usar el siguiente comando que va a crear una base de datos con la configuración indicada en el archivo ***.env***
      1. ***php bin/console doctrine:database:create***
3. Generar entidades desde base de datos. En esta parte del curso explica cómo generar Entidades usando YAML.
   1. Actualizar el archivo ***composer.json*** para que permita usar **‘annotations’**
   2. En composer, en la parte de requerido agregar: ***"doctrine/annotations": "^1.0"***
   3. Ejecutar como ejemplo: php bin/console doctrine:mapping:convert –from-database yml ./src/Entity
   4. Ejecutar como ejemplo: ***php bin/console doctrine:mapping:convert –from-database annotation ./src/Entity***
   5. Para que genere los Gets y Sets: ejecutar ***php bin/console make:entity --regenerate App***
   6. ***php bin/console doctrine:mapping:import App\\Entity yml –path=src/Entity***
4. Generar Entidades desde la base de datos o Actualizar
   1. Generar entidad con el comando: ***php bin/console doctrine:mapping:import App\\Entity annotation –path=src/Entity***
   2. Generar métodos faltantes: Get y set: ***php bin/console make:entity --regenerate App***
   3. Actualiza diferencias: ***php bin/console doctrine:migrations:diff***
5. Generar Entidades y luego generar la tabla en la base de datos ( <https://symfony.com/doc/current/doctrine.html#creating-an-entity-class> )
   1. Crear entidades:
      1. ***php bin/console make:entity Usuario***
      2. Cargar los campos en la entidad generada
      3. ejecutar: ***php bin/console make:entity –regenerate***
   2. Crear tabla desde entidades
      1. Ejecutar: ***php bin/console doctrine:migrations:diff*** (Migra las diferencias)
      2. Crea un archivo con la migración en la carpeta migrations
      3. Para migrar: ***php bin/console doctrine:migrations:migrate***
6. Guardar datos en la base de datos con Doctrine.
   1. En el controlador:
      1. ***use Doctrine\Persistence\ManagerRegistry as PersistenceManagerRegistry;***

***public function save(PersistenceManagerRegistry $doctrine){***

* + 1. Cargo el Entity Manager: ***$entityManager = $doctrine→getManager();***
    2. Asigno valores :

***$animal = new Animal();***

***$animal->setTipo('Avestruz');***

***$animal->setColor('gris');***

***$animal->setRaza('Africana');***

* + 1. Guardo objeto doctrine, en el ejemplo animal : ***$entityManager→persist($animal);***
    2. Vuelco los datos en la base de datos: ***$entityManager→flush();***
  1. Comprobar si se guardaron los objetos: ***php bin/console doctrine:query:sql “SELECT \* FROM animales”***

1. Búsqueda:
   1. se puede usar:
      1. ***findAll()***: Permite buscar todos del listado
      2. ***findBy()***: Permite buscar los que cumplen con la condición indicada.
         1. Ejemplo: ***$animal = $animal\_repo→findBy([‘raza’ => ‘africana’],[‘id’=>’DESC’]);***
      3. ***FindOneBy()***: permite buscar el primero que cumple con la condición
2. QueryBuilder <https://www.doctrine-project.org/projects/doctrine-orm/en/2.16/reference/query-builder.html>
   1. Para realizar consultas
3. DQL (Doctrine Query Lengaje)
   1. Para realizar consulta en Doctrine similar a SQL
4. SQL
   1. Se puede hacer consultas directamente a la base de datos
5. Repositorios
   1. Clase en la cual va a tener métodos que interactúa con la base de datos
   2. El repositorio crearlo en App\Repository y asignarle en el constructor la clase Animal
   3. Asociar el modelo Animal, por ejemplo, al repositorio: ***@ORM\Entity(repositoryClass="App\Repository\AnimalRepository")***

Controladores, Rutas y Vistas

1. Controladores
   1. Van dentro de la carpeta src.
      1. Desde la consola de comandos: ***php bin/console make:controller***
2. Rutas
   1. Se pueden cargar de dos formas:
      1. En el controlador: ***#[Route('/home', name: 'app\_home')]***
      2. En el archivo ***routes.yaml*** (DEBEN TENER ESA ESTRUCTURA TABULADA). Ejemplo:

***index:***

***path: /inicio***

***controller: App\Controller\HomeController::index***

* + 1. Para indicar parámetros: van en la ruta con {} y con el ? Va opcional. También se asigna con default. Si tiene indicado un valor por defecto, no importa si se indica o no el ? Lo toma igual como opcional.Ejemplo:

***validar\_email:***

***path: /validar-email/{email}***

***controller: App\Controller\AnimalController::validarEmail***

1. Requerimientos:
   1. ***‘w+’*** Indica que es texto
   2. ***‘[a-zA-Z ]+’*** Indica que es texto
2. Redirecciones
   1. Ejemplo: ***return $this→redirectToRoute('index');***
   2. ***return $this->redirectToRoute('animales', [***

***'nombre'=>'Adrian',***

***'apellidos'=>'Magnago’]);***

* 1. ***return $this->redirect('https://www.google.com.ar/');***

1. Vistas y Twig ( <https://twig.symfony.com/doc/> )
   1. En las vistas no se usa php, se usa la nomenclatura ‘twig’
   2. Tipos de instrucciones:
      1. ***{%*** : Es para ejecutar algo
      2. ***{#*** : Es para comentar
      3. ***{{*** : Es para mostrar
   3. Operadores o comandos:
      1. ***~*** : sirve para concatenar, por ejemplo: aves.tipo ~ ' - ' ~ aves.raza
      2. ***|*** : sirve para aplicar filtros en operadores por ejemplo: if animales|length >= 0
   4. Extensiones: (Funciones o filtros creadas por el programador)
      1. ***php bin/console make:twig-extension <Nombre>***
      2. Agregar en ***config/services.yaml***

***App\Twig:***

***resource: '../src/Twig'***

***tags: ['twig.extension']***

Formularios

Crear los formularios automáticamente relacionarlos a las entidades con **formbuilder**

1. Validación
   1. Validar datos de formularios asociados a entidades
      1. En el controlador usar isValid
      2. En la clase usar Asserts
   2. Validar datos aislados <https://symfony.com/doc/current/reference/constraints.html>
      1. ***use Symfony\Component\Validator\Validation;***
      2. para mail por ejemplo: ***use Symfony\Component\Validator\Constraints\Email;***

Problemas - Soluciones

1. Problemas con el chache en las pruebas: borrar la carpeta cache ubicada en ***‘var/cache’***
2. Problemas con la anotaciones de Doctrine: Eliminar de ***config/packages/framework.yaml*** la línea ***annotations:false*** porque sino no usa anotaciones y usa atributos
3. Problemas al llamar a un controlador: “Error 404”: Copiar el archivo ***.htaccess*** a la carpeta: ***/public***
4. Problemas: Si la página no funciona y da error: Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6 Server at localhost Port 80.
   1. Reinstalar el apache-pack: Eliminando de ***package.json*** ***symfony/apache-pack***
   2. Ejecutando: ***composer require symfony/apache-pack***
5. Problemas si no carga Bootstrap: faltan importar:
   1. ***<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.min.css" rel="stylesheet" >***
   2. ***<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" ></script>***
   3. ***<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.min.js" integrity="sha384-Atwg2Pkwv9vp0ygtn1JAojH0nYbwNJLPhwyoVbhoPwBhjQPR5VtM2+xf0Uwh9KtT" crossorigin="anonymous"></script>***

Anexo

1. **Páginas útiles**
   1. [**https://symfonycasts.com/tracks/symfony**](https://symfonycasts.com/tracks/symfony)
2. **Comandos**
   1. Ayuda: ***php bin/console help***
   2. Listar comandos: ***php bin/console list***
3. **Publicar en Producción**
   1. <https://victorroblesweb.es/2017/08/15/curso-gratuito-symfony-3-en-produccion/>