# 1. Instalar composer

# 2. Instalar Symfony

DESCARGAR Ejecutable sympfnu

<http://symfony.com/installer>

BAJAR EL ARCHIVO PHAR. Y USAR PARA INSTALAR:

EJ

Php symfony.phar new new my\_project\_name 3.4

la recomendación en S3.4 y S4 es de **No usar Bundle**

**utilizar el App Bundle que se tiene para el proyecto.**

**Utilizar Routing por anotación**

**Templates plantillas demo con Twig**

**Usar Bootstrap**

**Imágenes free** [**https://unsplash.com/**](https://unsplash.com/)

**Rutas con parámetro**

**Ejemplo twig**

{{path('bares', {'ciudad': 'cal'})}}

**Controller:**

/\*\*

\* @Route("/bares/{ciudad}", name="bares")

\*/

Pasar un array al render de la vista

return $this->render('frontal/bares.html.twig', array("ciudad"=>$ciudad));

# LA BASE DE DATOS CON DOCTRINE

Se deben tener en cuenta los siguientes aspectos en mysql y en Sym

Configurar en parameters yml la base de datos

Mysql UTF 8 collatiion

Crear DB Con doctrine

php bin/console doctrine:database:create

Crear tablas

# Las Entity class

Para crear una entity se genera con el siguiente comando

php bin/console doctrine:generate:entity

1. Asignar nombre <bundle>:<entity>
2. Seleccionar metodo de formato de informacion adicional
3. El campo id se genera automaticamente\*
4. Se deberan introducir los campos, tipo de campo, tamaño
5. Al finalizar se crean dos archivos entity\tapa.php y repository\taaRepository.php

# Crear la DB usando doctrine

Para crear las tablas de la base de datos usando la consola con el siguiente comando:

php bin/console doctrine:schema:update --force

# Empezar a leer la DB con DOCTRINE dentro del controlador

Lo primero es incorporar la primer línea de datos: en el controlador

Esta línea se pone en el action

Llamar al recurso

use AppBundle\Entity\Tapa;

// Capturar el repositorio de la tabla con la DB ,Tapa es la entidad q trae los datos

$repository = $this->getDoctrine()->getRepository(Product::class);

Cambiar Producto por el nombre de la entidad (Tapa) para este curso

Hacer un select de todo con

*// finds \*all\* products*

$tapas = $repository->findAll();

# Generacion del formulario en symfony

Crear un objeto de la entity.

$tapa = new Tapa();

Crear el constructor del formulario

$formBuilder = $this->createFormBuilder($tapa)

Agregar las clases que utiliza el formulario de Symfony

$formBuilder = $this->createFormBuilder($tapa);

Se puede iniciar cargando cada uno de las clases de tipos de datos en el formulario de la siguiente forma:

$tapa = new Tapa();

$formBuilder = $this->createFormBuilder($tapa);

$formBuilder->add('nombre', TextType::class);

$formBuilder->add('descripcion', TextareaType::class);

$formBuilder->add('fechaCreacion', DateType::class);

// para crear el boton SUBMIT

**$formBuilder->add('guardar', SubmitType::class, array('label'=>'Crear plato'));**

**$formBuilder->getForm();**

\*Nota: Agregar las clases de datos usadas en el encabezado del controlador

**use Symfony\Component\Form\Extension\Core\Type\TextType;**

**use Symfony\Component\Form\Extension\Core\Type\TextareaType;**

**use Symfony\Component\Form\Extension\Core\Type\DateType;**

**use Symfony\Component\Form\Extension\Core\Type\SubmitType;**

Enviar a la vista el formulario renderizado

return $this->render('gestionPlatos/nuevoPlato.html.twig', array("form"=>$form->createView()));

Generar el formulario con el Framework

1CREAR carpeta de formulario en el bundle

AppBundle\Form

2 Crear archive TapaTYpe.php con los datos del formulario

namespace AppBundle\Form;

// Librerias para crear los formularios

use Symfony\Component\Form\AbstractType;

use Symfony\Component\Form\FormBuilderInterface;

//Clases para los datos del Form

use Symfony\Component\Form\Extension\Core\Type\TextType;

use Symfony\Component\Form\Extension\Core\Type\TextareaType;

use Symfony\Component\Form\Extension\Core\Type\DateType;

use Symfony\Component\Form\Extension\Core\Type\SubmitType;

//Se crea una clase que cinstruira el formulario según los datos

class TapaType extends AbstractType

{

public function buildForm(FormBuilderInterface $builder, array $options)

{

$builder

->add('nombre', TextType::class)

->add('descripcion', TextareaType::class)

->add('guardar', SubmitType::class, array('label'=>'Crear plato'))

;

}

}

3 Modificar el controlador incluyendo el archivo Form/TapaType.php

use AppBundle\Form\TapaType;

4 En el controlador crear el formulario de la siguiente forma

//Creacion del formulario

//Crear nuevo objeto de tipo tapa para construir el formulario

$tapa = new Tapa();

//Construccion del formulario

$form = $this->createForm(TapaType::class, $tapa);

# Editor para los ingredientes

## En cmd:

composer require egeloen/ckeditor-bundle

agregar en el AppKernel en el segmento bundles

new Ivory\CKEditorBundle\IvoryCKEditorBundle(),

## Ejecutar en el CMD

php bin/console ckeditor:install

## Instalar en los assets

Php bin/console assets:install web

## Incorporar el Editor en el formulario

use Ivory\CKEditorBundle\Form\Type\CKEditorType;

Modificar la clase del formulario

->add('descripcion', CKEditorType::class)

Usar una barra de edición básica …

## Abrirl config.yml y agregar

ivory\_ck\_editor:

configs:

my\_config:

toolbar: basic