CODEIGNITER 4

Instalación manual

- Desempaqueta proyecto en htdocs
- php.ini: Activar **intl** y **mbstring**
- Visual Studio: Instalar extensión "**My CodeIgniter**"

Tareas de configuración en **config/**

- **App.php** **base_url** = http://localhost/aplicacion/ (no olvidar barra final)
- **constants.php** Para definir constantes globales
- **database.php** Modificar el array $default con la BD por defecto para la app

Habilitar accesos a través de rutas

- **routing.php** public bool $autoRoute = **false**;
- Copiar **index.php** y **.htaccess** de la carpeta public a carpeta raíz del proyecto
- Modificar **raíz/index.php**: FCPATH . 'app/Config/Paths.php' (**quitarle '..**')
- **routes.php**: $routes->get('/', 'Home::index');

    ➔ Al abrir http://localhost/aplicacion/ se ejecutará método **index**() del controlador **Home**

MODELOS en carpeta **Models**

```php
namespace App\Models;
use CodeIgniter\Model;

class M_restaurante extends Model{

	protected $db;

	public function __construct()	{
		$this->db=db_connect();  //se conecta la BD de config/database.php
	}
}
```

CONTROLADORES en carpeta **Controllers**

```
namespace App\Controllers;

class C_restaurante extends BaseController {

        protected $modelo;

        public function __construct()      {
                $this->modelo=new M_restaurante();
        }

        public function cargarVistas(){
                 echo view(“v_header”);
                 echo view("v_mivista");
                echo view(“v_foot”);
        }

        public function cargarVistas_pasando_datos(){

                $datos[‘msg’]=”Mensaje para vista”;
                $datos['platos']=$this->modelo->platosConPedidos();
                echo view("vlistaplatos",$datos, false);
        }

        public function recibir_datos_formulario(){
                if ($this->request->getPost('submit')){
                        $idIng=$this->request->getPost('idingrediente');
                        ……..
                }
        }


        public function recibir_datos_enlace($param1,$param2){

        }
        public function modificar_la_sesion(){
                $valor=$this->session->get(‘var_sesion’);
                $valor++;
                $this->session->set(‘var_sesion’, $valor );
        }
        public function redireccionar(){
                return redirect()->to(“login/”.$id);


        }
}
```

La vista tendrá disponibles variables:

$msg y **$platos**

routes.php

$routes->get('/login/(:any)','C_login::loguear/$1');

(Cambia la petición actual por una nueva petición al controlador **C_login,** método **loguear**, pasándole un parámetro)

Vista con enlace

```
echo "<table>";
foreach ($platos as $plato){
   echo "<tr>";
   echo "<td>$plato->nombre</td>";
   echo "<td>$plato->precio €</td>";

   echo "<td>";
   echo anchor(site_url()."/verdetalle/".$plato->idplato, "VER DETALLE");

   echo "</td>";
   echo "</tr>";
}
echo "<table>";
```

routes.php

```
$routes->get('/verdetalle/(:any)','C_restaurante::showDetail/$1');
```

(Se dirige al método **showDetail** del controlador **C_restaurante** pasándole como parámetro el id del plato)

Vista con formulario (helper form)

```
echo form_open(site_url().'/aniadePlato');
echo form_label("NOMBRE");
echo form_input('nombre');
echo form_label("PRECIO");
echo form_input("precio","0",['type'=>'number', 'min'=>'10', 'max'=>'500']);
echo form_label("FECHA");
echo form_input(["name"=>"fecha", "value"=>strFechaHoy(), "type"=>"date"]);
echo form_submit("submitNuevoPlato","AÑADIR");
echo form_close();
```

routes.php

```
$routes->post('/aniadePlato', 'C_restaurante::newPlato');
```

Se dirige al método **newPlato** del controlador **C_restaurante**. Los datos del formulario van ocultos en la petición ($this->request)

HELPERS

- Son ficheros procedimentales con funciones de ayuda.

- Helpers del sistema: **url** (funciones para crear links), **form** (funciones para dibujar formularios), **file**, **cookie**, **text**,etc

- **Helper personalizados**
    1. Debes crearlos en la carpeta helpers: archivo con nombre "xxx_helper.php"
    2. Constan de funciones públicas
    3. Cargar helpers manualmente
        - Función helper("xxxx"): busca el archivo xxx_helper en **App\Helpers** y en **System\Helpers**
        - Si lo hacemos en el constructor del controlador, estará disponible para éste y las vistas que cargue.
    4. Autocargar helpers
        - En **config/autoload.php** se pueden indicar los helpers a autocargar (del sistema o nuestros)

```
public $helpers = ['form'];
```

LIBRERIAS

- Son **clases** para añadir funcionalidad a la aplicación

- Hay librerías del sistema: session, email, validation, security,…

- **Librerías personalizadas**
    1. Podemos crearlas en la carpeta libraries , con la estructura

```
namespace App\Libraries;

class MiLibreria {
        public function metodoLib(){
                ....
        }
}
```

    2. Y usarlas así en el controlador:

```
namespace App\Controllers;
use App\Libraries\MiLibreria

class MiControlador{
        public $lib;
        public function __construct(){
                $this->lib=new MiLibreria();
        }
        public function metodo(){
                $this->lib->metodoLib();
        }
}
```

BaseController

- Es el controlador base del que heredan el resto.
- En él podemos poner recursos que queramos que hereden todos los controladores: activar la sesión, activar helpers, …
- Es un buen lugar para activar la sesión

Sesión

- Para que la sesión esté disponible en todos los controladores, pondremos en el controlador BaseController

  protected $session;

  Y en su método initController:

  ```
  $this->session = \Config\Services::session();
  ```

  ($this->session estará disponible en todos los controladores)

- Otra opción es llamar a la función **session()** cada vez que se quiera acceder a la sesión

Layouts - Plantillas

1. Crear **vista plantilla**, indicando las secciones sustituibles con $this->renderSection('nombre');

   Ejemplo **v_plantilla.php**

```
<html>….<body> Contenido fijo…..

        <?= $this->renderSection('seccion1'); ?>        → Sección con la parte variable

Más contenido fijo

        <?= $this->renderSection('seccion2'); ?>        → Sección con la parte variable

Más contenido fijo……</body>/html>
```

2. Crear **vista que utiliza la plantilla**
   - Indicando que la extiende en su 1ª línea
   - Sustituyendo las secciones que se deseen en bloques $this->section('nombre') ……$this->endSection();

   Ejemplo **v_usa_plantilla.php**

```
        <?=  $this->extend('plantilla');  ?>

        <?=  $this->section('seccion1');  ?>
                Contenido de la sección 'seccion1'
        <?=  $this->endSection();         ?>

        <?=  $this->section('seccion2');  ?>
                Contenido de la sección 'seccion2'
        <?=  $this->endSection();         ?>
```

3. Nuestro controlador, justo haría:
   echo view('**v_usaplantilla,'**,$datos);