



<http://www.robertsallent.com> 
[@robertsallent](https://twitter.com/robertsallent) 

Tutorial RMF

Tutorial de creación de aplicaciones web
usando RMF (v1.0)



Robert Sallent
www.robertsallent.com

SOC Servei d'Ocupació
de Catalunya
CIFO Sabadell- Mancomunitat (Vallès)



**Generalitat
de Catalunya**



Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur



Índice

- Antes de nada
- Utilidades y características del *RMF*
- La aplicación
- Casos de uso
- Implementación
 - Cambiando el aspecto
 - Creando el modelo
 - Probando el modelo
 - Creando controladores
 - Las vistas
- Ejercicios



Antes de nada

- En las presentaciones anteriores se explicó cómo configurar y probar el *RobS Micro Framework (RMF)* que usamos en clase.
- También se describieron algunas de sus características y cómo usarlas en el desarrollo de aplicaciones web.
- **Esta presentación es un tutorial sobre cómo usar dicho *framework* para crear una pequeña aplicaciones web.**



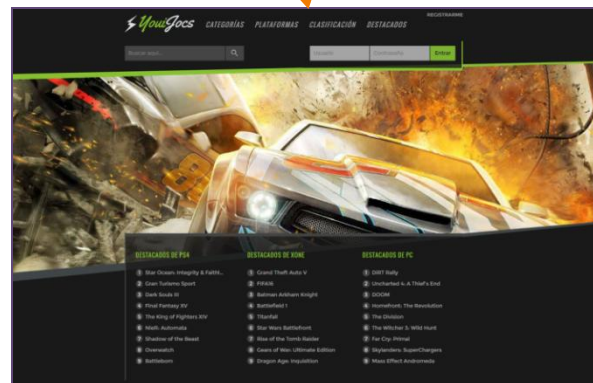
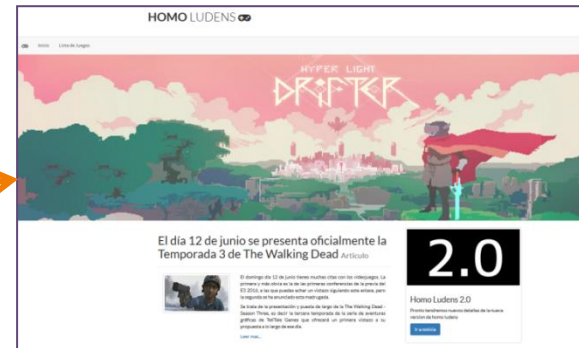
Antes de nada

- Esta herramienta está **pensada para uso docente**, puesto que sirve para comprender el funcionamiento y aplicación de la arquitectura Modelo Vista Controlador en sistemas web.
- Además, está inspirada en *CodeIgniter*, con lo que pasar a dicho *framework* resultaría casi natural.





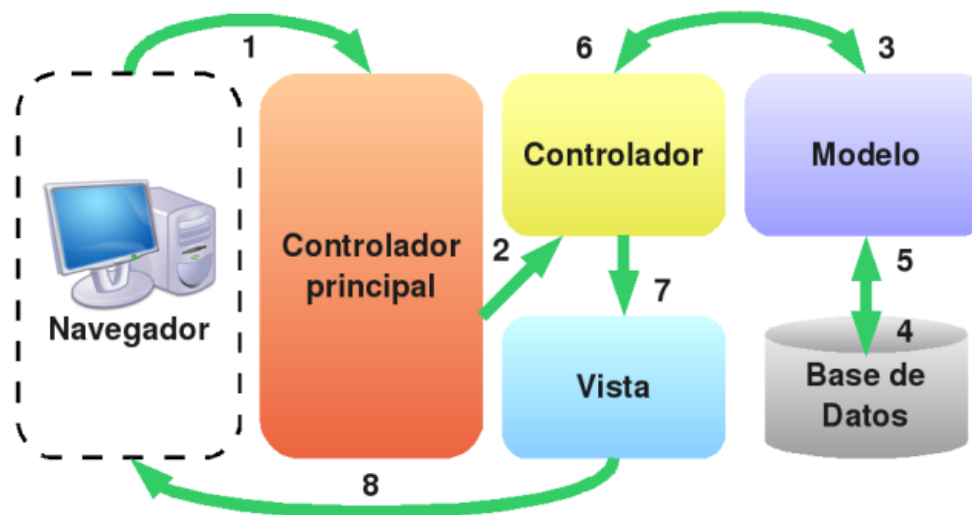
Ejemplos aplicaciones creadas con RMF





Antes de nada

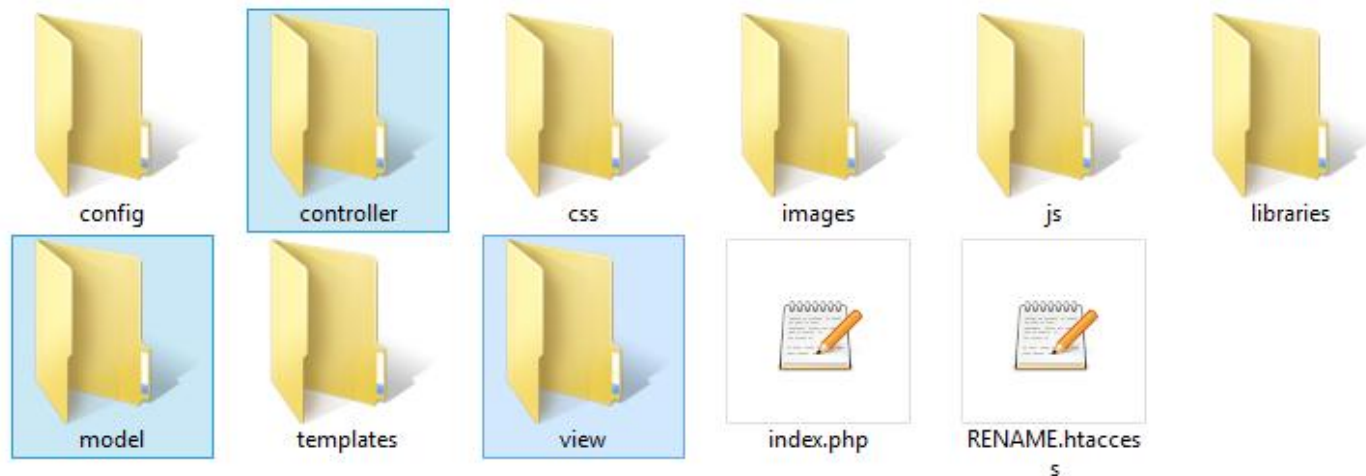
- *RMF* trabaja usando una arquitectura **modelo-vista-controlador** con **controlador frontal**.





Antes de nada

- Principalmente deberemos centrarnos en el desarrollo de las clases del **modelo**, **controladores** específicos y las **vistas** para la interacción con el usuario.





La aplicació

- Vamos a crear una aplicació sencilla, un blog de capítulos de *los Simpsons*.
- El análisis inicial ha determinado que habrá **tres perfiles de usuario** (roles): **usuario no registrado**, **usuario registrado** y **administrador**.





La aplicación

- Los usuarios no registrados pueden: **registrarse, ver el listado** de capítulos.
- Los usuarios registrados pueden hacer lo anterior y además: **identificarse, modificar sus datos, darse de baja y ver detalles** sobre los capítulos.
- El administrador puede hacer todo lo anterior y además: introducir datos de un **nuevo capítulo, modificar** un capítulo y **borrarlo**.



Diagrama de casos de uso

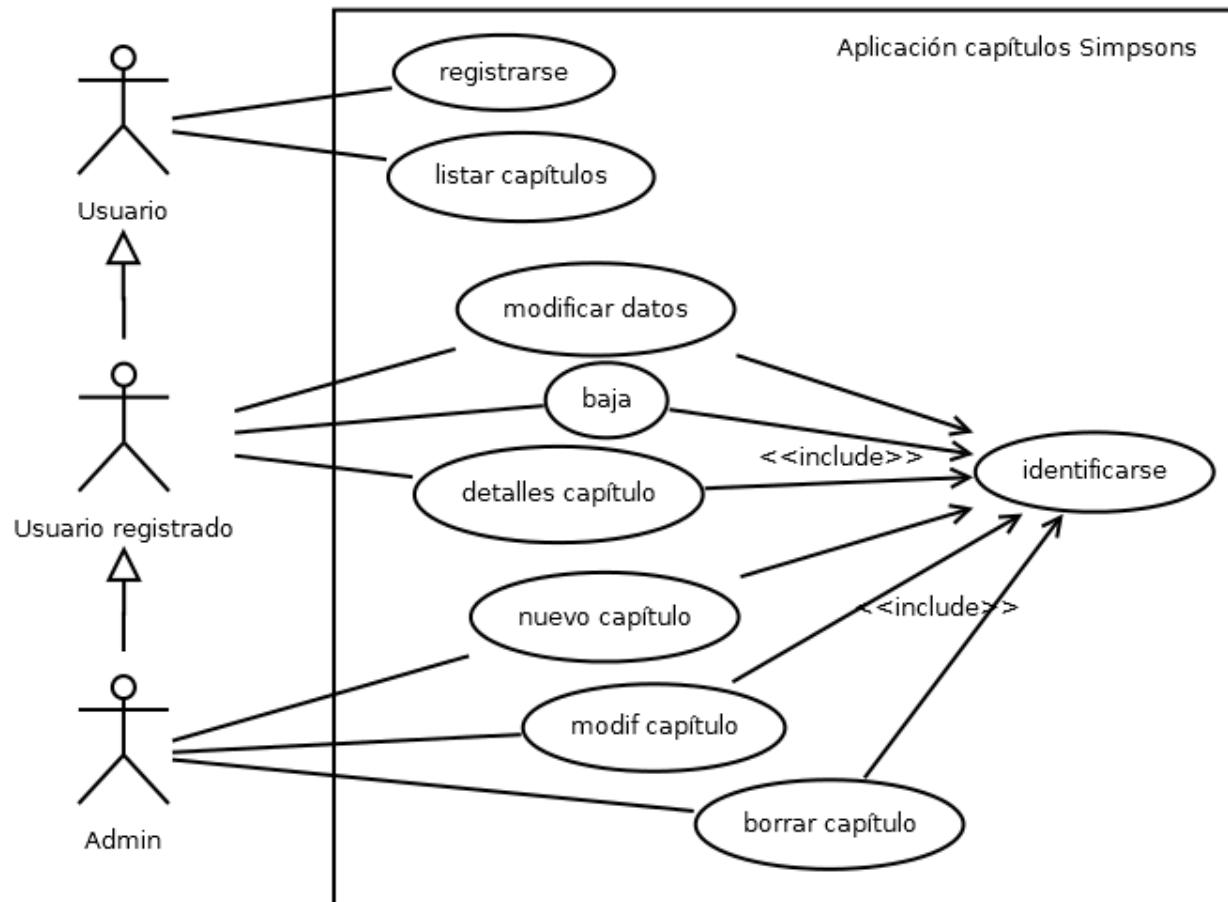
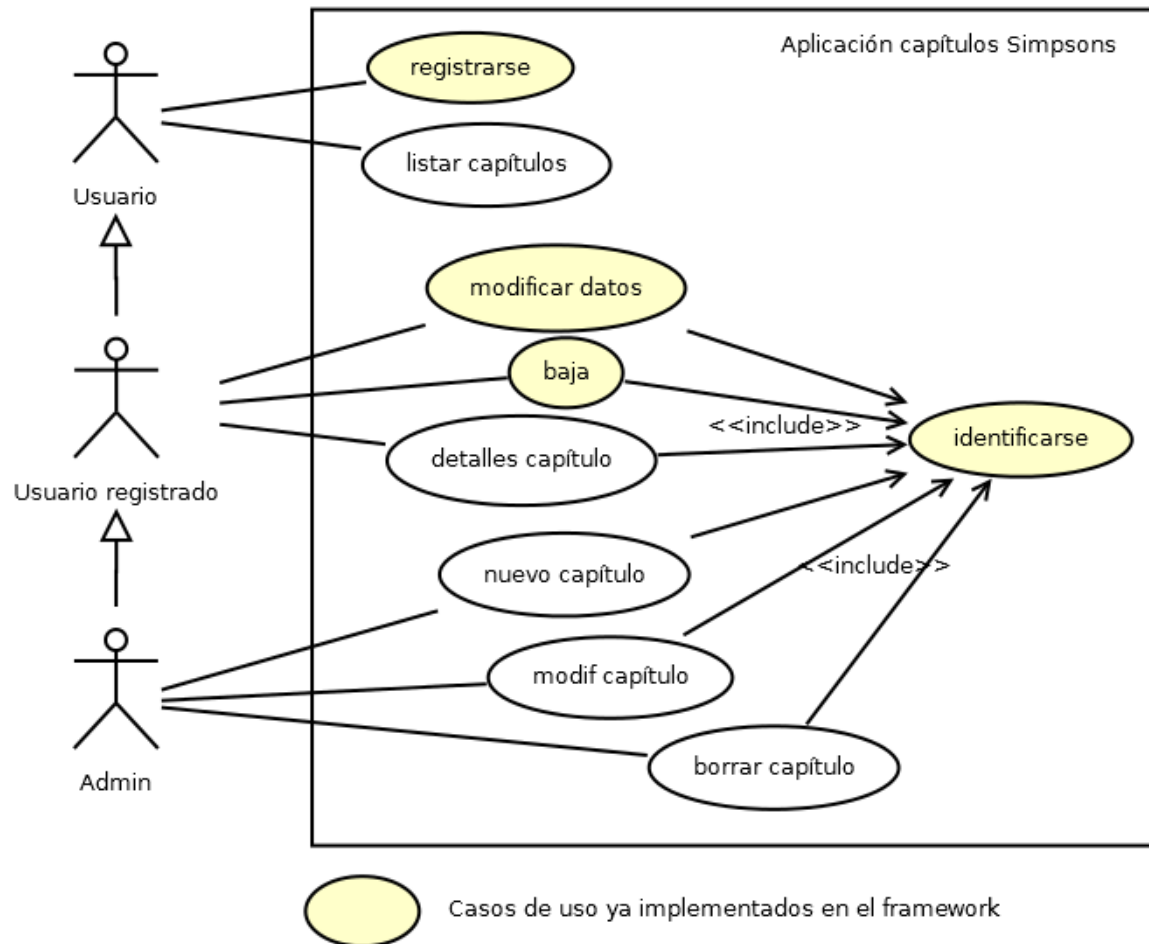




Diagrama de casos de uso





Casos de uso

- Los casos de uso anteriores se recogerán en el documento de especificación.
- A continuación se muestra un resumen de los mismos (solamente de los que no están ya implementados).

| CASO DE USO: REGISTRO DE USUARIO | | | |
|----------------------------------|--|-------|------------|
| Versión | 1.0 | Fecha | 02/09/2016 |
| Autores | Robert | | |
| Descripción | A continuación se describe el caso de uso: registro de usuarios. Mediante el cual un usuario se da de alta en la aplicación "simmons" y puede utilizar las herramientas disponibles para los usuarios registrados. | | |
| Actores | Usuario | | |
| Precondiciones | El usuario no debe estar registrado previamente. | | |
| Flujo principal | <ol style="list-style-type: none">1. El usuario accede a la aplicación "capítulos de los <u>simmons</u>" (<u>localhost/simmons</u>).2. El usuario accede a la opción "registro" que se encuentra disponible en cualquiera de las vistas.3. El sistema muestra un formulario de registro solicitando la siguiente información: nombre, apellidos...4. Tras rellenar los datos solicitados, el usuario presiona "aceptar".5. El sistema toma los datos, los procesa y los guarda en la base de datos. El usuario ha quedado registrado.6. El sistema informa al usuario que se ha completado exitosamente el proceso de registro. | | |
| Flujos alternativos | En caso de que no se pueda realizar el proceso de registro porque los datos no son correctos, se informará al usuario para los modifique antes de volver a enviar el formulario de registro. | | |
| Postcondiciones | Los datos del usuario quedan almacenados en la tabla de usuarios de la base de datos. | | |
| Requerimientos no funcionales | El proceso de registro debe funcionar desde dispositivos móviles. | | |
| Prioridad | Alta | | |
| Comentarios | Si el usuario no selecciona una foto para el perfil, se usará una imagen de usuario por defecto. | | |



Casos de uso (resumen)

- **Listar capítulos:** mediante la opción de menú “listar” se mostrará el listado de capítulos.
- **Detalles capítulo (registrados):** al hacer clic en “ver” cuando se está viendo el listado de capítulos, se mostrarán los detalles o una página de error solicitando identificación.



Casos de uso (resumen)

- **Nuevo capítulo** (admin): el administrador podrá acceder a la opción “nuevo capítulo” desde un menú de administrador. Se le presentará un formulario para que pueda introducir los datos del nuevo capítulo.
- **Modificar capítulo** (admin): desde el listado de capítulos, el administrador dispondrá de un botón para modificar el capítulo deseado. Se le presentará un formulario con los datos del capítulo seleccionado para que los pueda modificar.
- **Borrar capítulo** (admin): desde el listado de capítulos, el administrador dispondrá de un botón para borrar un capítulo. Se le pedirá confirmación de la operación.



La tecnología

- La tecnología que usaremos para el desarrollo es la que hemos estudiado en el curso **IDFC0210 – Desarrollo de aplicaciones con tecnologías web**:
 - *HTML, CSS, JavaScript* (si hace falta) para las vistas.
 - *PHP* para controladores y modelo.
 - *MySQL* para la base de datos.
- *RMF* como *framework* base para el desarrollo de la aplicación (arquitectura Modelo-Vista-Controlador).



La tecnologia





El modelo

- Queda bastante claro que la aplicación necesita manejar información de **usuarios** y **capítulos** (serán dos clases del modelo).
- No haremos cambios en la gestión de usuarios que hace el *framework*, así que solamente debemos determinar la información sobre los capítulos que será relevante para el sistema.
- En principio, de cada capítulo necesitamos saber: **identificador** (único), **número del capítulo**, **temporada**, **título**, **descripción**, **duración**, **fecha de emisión** e **imagen del capítulo**.



La base de datos

- La estructura de la base de datos `simpsons` será muy simple: la tabla `usuarios`, que necesitamos para gestionar los usuarios mediante el *RMF*, y la tabla `capítulos` para guardar información del show.

| simpsons.usuarios | |
|-------------------|------------------------|
| 🔑 | user : varchar(32) |
| 📄 | password : varchar(32) |
| 📄 | nombre : varchar(32) |
| # | privilegio : int(11) |
| # | admin : int(11) |
| 📄 | email : varchar(128) |
| 📄 | imagen : varchar(512) |

| simpsons.capitulos | |
|--------------------|-----------------------|
| 🔑 | id : int(11) |
| # | capitulo : int(11) |
| # | temporada : int(11) |
| 📄 | titulo : varchar(256) |
| 📄 | descripcion : text |
| # | duracion : int(11) |
| 📅 | fecha_emision : date |
| 📄 | imagen : varchar(256) |



Las vistas (GUI)





La lógica (controladores)

- De la misma forma que con el modelo, podemos determinar que necesitaremos dos controladores:
 - Uno para gestionar las **operaciones con los usuarios**.
 - Otro para gestionar las **operaciones con los capítulos**.
- Como el controlador de `Usuario` ya lo tenemos en el *framework*, solamente nos centraremos en la implementación del controlador de `Capítulos`.



Comenzando la implementación

- Una vez tenemos todo bien claro, sabemos lo que hay que hacer y las tecnologías que usaremos, **podemos comenzar con la implementación.**





Implementación la BDD

- Para comenzar, **debemos crear la base de datos** con el nombre “simpsons”.
- Cread en ella la tabla **usuarios** (podéis usar el fichero `tabla_usuarios.sql` incluido con el *framework*) y la tabla **capitulos**.



| 127.0.0.1 » simpsons | | |
|------------------------------------|-----------------|------------|
| Estructura | SQL | Buscar |
| Tabla | Acción | |
| | | |
| <input type="checkbox"/> capitulos | Examinar | Estructura |
| <input type="checkbox"/> usuarios | Examinar | Estructura |
| 2 tablas | Número de filas | |



Implementación la BDD

- La estructura de las tablas es la mostrada anteriormente:

| simpsons.usuarios | |
|-------------------|------------------------|
| 🔑 | user : varchar(32) |
| 📄 | password : varchar(32) |
| 📄 | nombre : varchar(32) |
| # | privilegio : int(11) |
| # | admin : int(11) |
| 📄 | email : varchar(128) |
| 📄 | imagen : varchar(512) |

| simpsons.capitulos | |
|--------------------|-----------------------|
| 🔑 | id : int(11) |
| # | capitulo : int(11) |
| # | temporada : int(11) |
| 📄 | titulo : varchar(256) |
| 📄 | descripcion : text |
| # | duracion : int(11) |
| 📅 | fecha_emision : date |
| 📄 | imagen : varchar(256) |






Implementación preparando el framework

- Ahora vamos a descargar y configurar el *framework*, tal y como se describió en la anterior presentación.
- Descargadlo desde:
http://recursos.robertsallent.com/mvc/robs_micro_fw_1.0.zip
- O si no funciona el enlace, mirad en:
<http://portfolio.robertsallent.com/php.php>

RECURSOS

 *RobS micro MVC Framework (RMF)*

 *Documentación de instalación y configuración del RMF:*



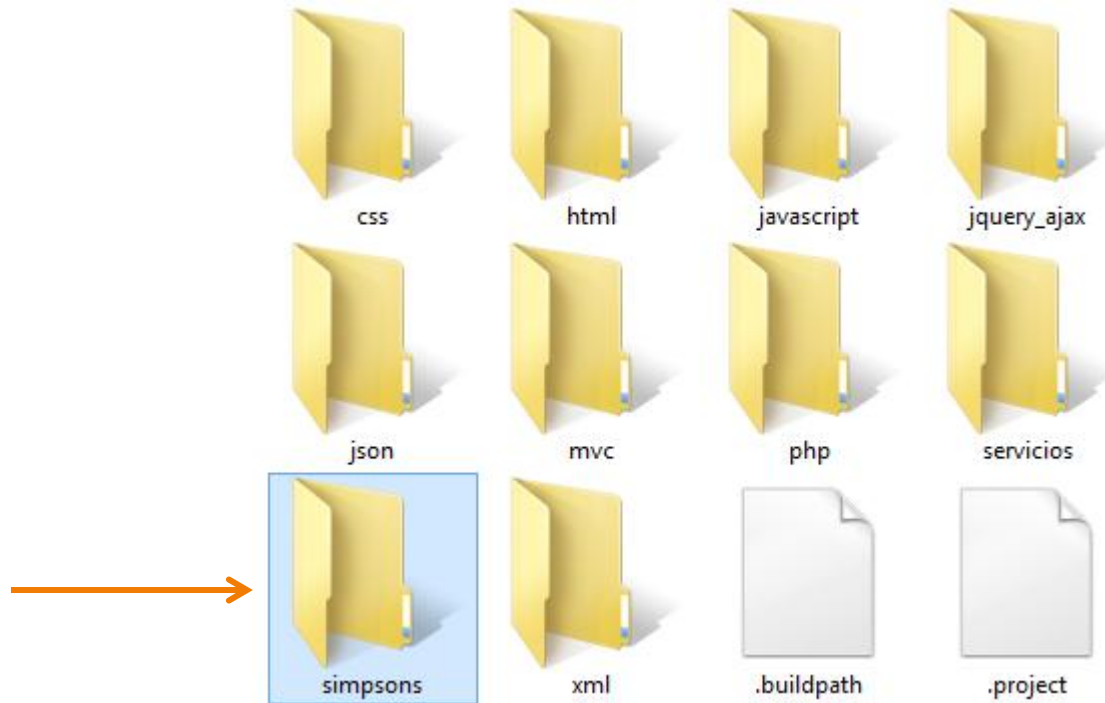


Comenzando la implementación

- Una vez descargado y descomprimido a nuestro *Workspace*:
 1. Renombramos el nombre de la carpeta `project` a `simpsons`, (que será el nombre de nuestro proyecto).
 2. Creamos el nuevo proyecto `simpsons` con *Eclipse*.
 3. Procedemos a editar el fichero de configuración en `config/Config.php`.
 4. Probamos que todo funcione.

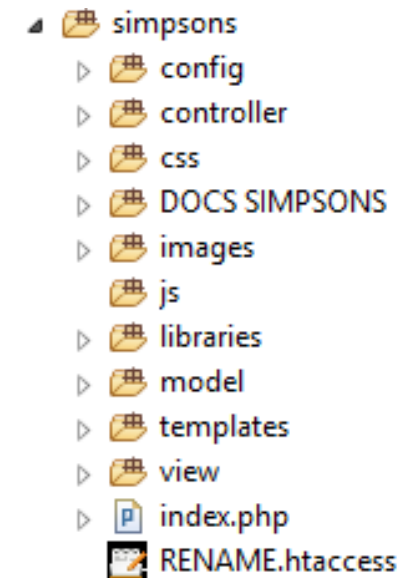
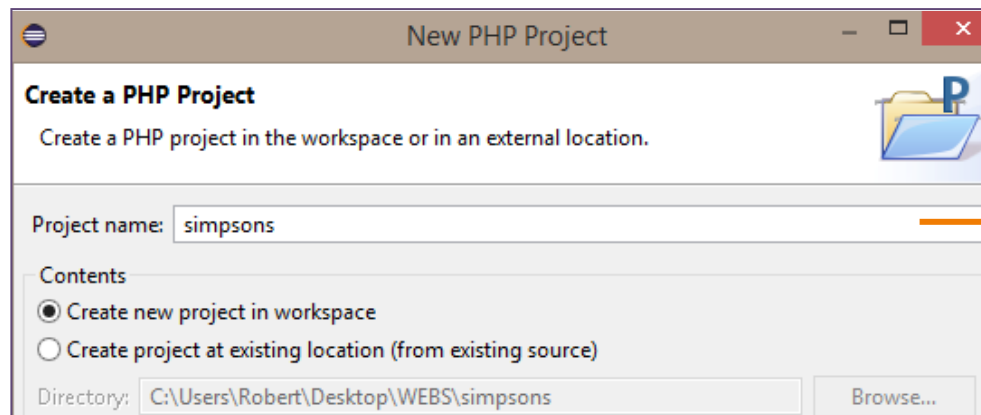


Ejemplo renombrar la carpeta





Ejemplo crear el proyecto en Eclipse





Ejemplo editar el fichero de configuración

```
//-----  
//EDITAR ESTOS PARAMETROS PARA CAMBIAR LA CONFIGURACION  
//URL BASE (ruta donde se encuentre el proyecto, desde el DOCUMENT_ROOT)  
private $url_base = '/simpsons/';  
  
//PARA LA BDD  
private $db_host = 'localhost'; //ubicación de la BDD  
private $db_user = 'root';      //usuario  
private $db_pass = '';        //password  
private $db_name = 'simpsons'; //nombre de la BDD  
private $db_charset = 'utf8';  //codificación a utilizar  
private $db_user_table = 'usuarios'; //nombre para la tabla de usuarios  
  
//CONTROLADOR Y OPERACION POR DEFECTO  
private $default_controller = 'Welcome'; //controlador por defecto  
private $default_method = 'index';      //método por defecto  
  
//ESTILO POR DEFECTO  
private $css = 'css/estilo.css'; //fichero CSS con el estilo por defecto  
  
//OPCIONES PARA LAS IMAGENES  
private $image_not_found = 'images/no_image.png'; //imagen no encontrada  
private $user_image_directory = 'images/users/';  //directorio para las imágenes de usuario  
private $default_user_image = 'images/users/user.png'; //imagen por defecto para usuarios  
private $user_image_max_size = 512000; //tamaño máx imágenes de usuario
```



Ejemplo probar que todo funcione

- Escribimos en la barra de direcciones del navegador: `localhost/simpsons`

Mi aplicación web

User: Password:

[Inicio](#) [Registro](#)

Notas

Este micro framework ha sido desarrollado con fines docentes para el CP de desarrollo de aplicaciones con tecnologías web (IFCD0210) que imparte Robert Sallent.

Es un ejemplo de arquitectura modelo-vista-controlador sencillo para entender los conceptos y poder trabajar con él.

A lo largo del curso se desarrollarán varios proyectos de ejemplo usando este pequeño framework, para ir entendiendo los conceptos básicos comunes a este tipo de herramientas de trabajo MVC existentes en PHP.

NO ES 100% SEGURO, así que no se debe usar para desarrollos en entornos de producción.

En el mismo curso, en el último módulo, utilizaremos otro CodeIgniter para desarrollos más complejos.

[RobS micro Framework](#) - solo para fines docentes, [Robert Sallent](#) - CIFO'16. Twitter: [@robertsallent](#)

- Probamos que funcione el usuario admin (password 1234), creamos algún usuario, modificamos sus datos...



Cambiando el aspecto

- Si queremos cambiar ya el aspecto de la aplicación, podemos editar directamente los siguientes ficheros:
 - El fichero `css/estilo.css` modifica el estilo de la aplicación.
 - El fichero `templates/Template.php` contiene patrones para crear de forma rápida las áreas comunes de todas las vistas (*header, footer, menú, login, logout...*).
 - Si modificamos las vistas directamente en la carpeta `views` sólo cambiaremos la vista concreta que editemos.

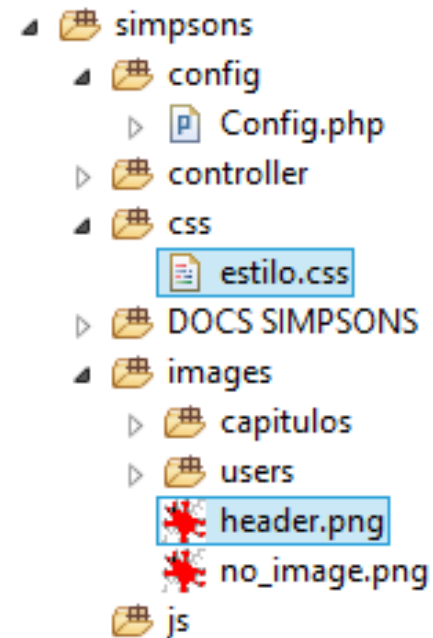


Ejemplo modificar el estilo CSS base

```
body{
  background-color: #ffc;
  font-family: arial, verdana;
  width: 80%;
  padding: 2%;
  margin: 10px auto 10px auto;
  box-shadow: 0px 0px 4px rgba(60,60,60,0.6);
}

header{
  background-image: url('../images/header.png');
  background-size: 40%;
  background-position: 75%;
  background-repeat: no-repeat;
  background-color: #ff5;
  padding: 50px;
  font-size: 30px;
  color: white;
  text-shadow: 2px 2px 2px rgba(60,60,60,0.6);
}

footer{
  padding: 20px;
  background-color: #ff5;
}
```





Ejemplo modificar el estilo CSS base

Mi aplicación web



User: Password:

[Inicio](#) [Registro](#)

Formulario de registro

User:

Password:

Nombre:

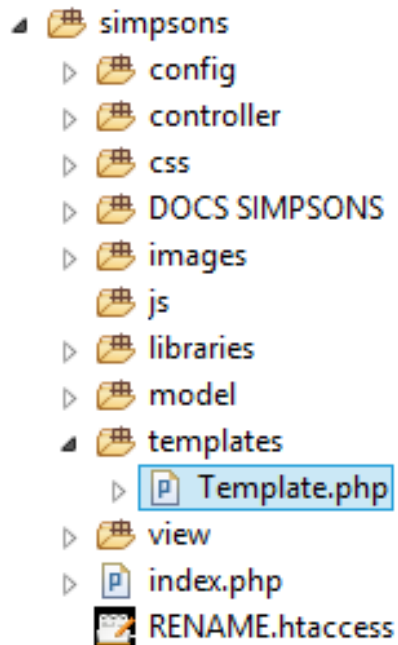
Email:

Imagen: No se ha seleccionado ningún archivo. max 500kb

RobS micro Framework - solo para fines docentes, Robert Sallent - CIFO'16. Twitter: @robertsallent



Ejemplo modificar el template



```
//PONE EL HEADER DE LA PAGINA
public static function header(){
    ?>
    <header>
        <h1>Simpsons Chapter List</h1>
        <h2>Listado completo de capítulos de los Simpsons</h2>
    </header>
    <?php
}
```





Ejemplo modificar las vistas

- La vista `view/welcome_message.php` se corresponde con la portada del sitio.

```
<section id="content">
  <h2>Bienvenidos</h2>
  <article class="flex-container">
    <div class="flex">
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
        incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
        exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
      <p>Duis aute irure
        dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
        Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit
        anim id est laborum.</p>
    </div>
    <figure class="portada flex">
      
      <figcaption>Los Simpsons</figcaption>
    </figure>
  </article>
</section>
```



Ejemplo modificar las vistas

- simpsons
 - config
 - controller
 - css
 - DOCS SIMPSONS
 - images
 - js
 - libraries
 - model
 - templates
 - view
 - capitulos
 - usuarios
 - error.php
 - exit.php
 - welcome_message.php
 - index.php
 - RENAME.htaccess





Implementando

- Bueno, vamos a comenzar con la parte interesante del desarrollo.
- Lo primero que haremos será **centrarnos en el modelo**. Debemos crear la clase `CapituloModel` en la carpeta `model`.
- Dicha clase permitirá crear y manipular objetos de clase `CapituloModel`. **Dispondrá también de los métodos que interactuarán con la *BDD* para realizar las operaciones *CRUD* (*Create, Read, Update, Delete*).**
- **Probaremos** cada uno de los métodos creados antes de seguir.



Creando el modelo

- Si miramos **los casos de uso**, descubrimos que necesitamos:
 - Una forma de **recuperar** todos los capítulos (listado).
 - Una forma de **recuperar** un capítulo concreto (detalles)
 - Una forma de **guardar** un capítulo (*admin*).
 - Una forma de **modificar** los datos de un capítulo (*admin*).
 - Una forma de **borrar** un capítulo (*admin*).
- Estos cinco métodos deberán estar implementados dentro de la clase `CapituloModel`.



Ejemplo CapituloModel (guardar)

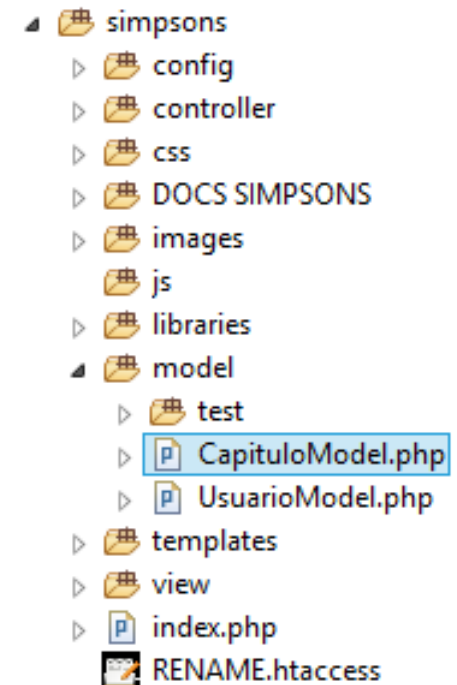
```
<?php
class CapituloModel{

    //PROPIEDADES que deben tener los capítulos
    public $id, $capitulo, $temporada;
    public $titulo, $descripcion;
    public $duracion, $fecha_emision;
    public $imagen;

    //METODOS (CRUD: Create Read Update Delete)
    //método que guarda un capítulo en la BDD
    public function guardar(){
        $consulta = "INSERT INTO capitulos
            VALUES(DEFAULT,
                $this->capitulo,
                $this->temporada,
                '$this->titulo',
                '$this->descripcion',
                $this->duracion,
                '$this->fecha_emision',
                '$this->imagen'
            );";

        return Database::get()->query($consulta);
    }
}
```

Hace uso de **database_library** que se encuentra en la carpeta **libraries**





Ejemplo CapituloModel (recuperar todos)

```
//Métodos que recupera todos los capítulos desde la BDD
//los retorna en un array de CapituloModel (array de capítulos)
public static function getCapitulos(){
    $consulta = "SELECT * FROM capitulos;"; //preparar la consulta

    //ejecutar la consulta (devuelve un mysqli_result)
    $resultado = Database::get()->query($consulta);

    $capitulos = array(); //prepara la lista de capítulos

    //vuelca los resultados a la lista de capítulos
    while($capitulo = $resultado->fetch_object('CapituloModel'))
        $capitulos[] = $capitulo;

    $resultado->free(); //libera el objeto de resultados

    return $capitulos; //retorna el array de CapituloModel
}
```

- simpsons
 - config
 - controller
 - css
 - DOCS SIMPSONS
 - images
 - js
 - libraries
 - model
 - test
 - CapituloModel.php
 - UsuarioModel.php
 - templates
 - view
 - index.php
 - RENAME.htaccess



Ejemplo CapituloModel (recuperar uno)

```
//Método que obtiene un capítulo concreto
//(devuelve null si no lo encuentra)
public static function getCapitulo($id=0){
    //preparar la consulta
    $consulta = "SELECT * FROM capitulos WHERE id=$id;";

    //ejecutar la consulta (devuelve un mysqli_result)
    $resultado = Database::get()->query($consulta);
    if(!$resultado) return null;

    $capitulo = $resultado->fetch_object('CapituloModel');
    $resultado->free();

    return $capitulo;
}
```

- ▲ 📁 simpsons
 - ▷ 📁 config
 - ▷ 📁 controller
 - ▷ 📁 css
 - ▷ 📁 DOCS SIMPSONS
 - ▷ 📁 images
 - ▷ 📁 js
 - ▷ 📁 libraries
 - ▲ 📁 model
 - ▷ 📁 test
 - ▷ 📄 CapituloModel.php
 - ▷ 📄 UsuarioModel.php
 - ▷ 📁 templates
 - ▷ 📁 view
 - ▷ 📄 index.php
 - 🔧 📄 RENAME.htaccess



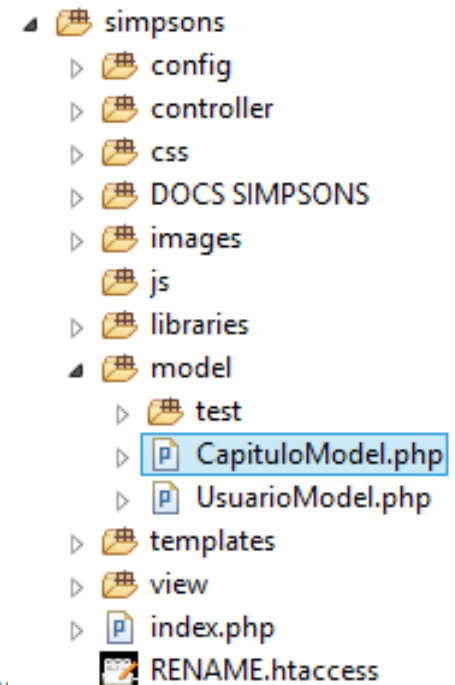
Ejemplo CapituloModel (borrar y modificar)

```
//Método que borra un capítulo
public function borrar(){
    $consulta = "DELETE FROM capitulos WHERE id=$this->id;";

    //ejecutar la consulta
    return Database::get()->query($consulta);
}

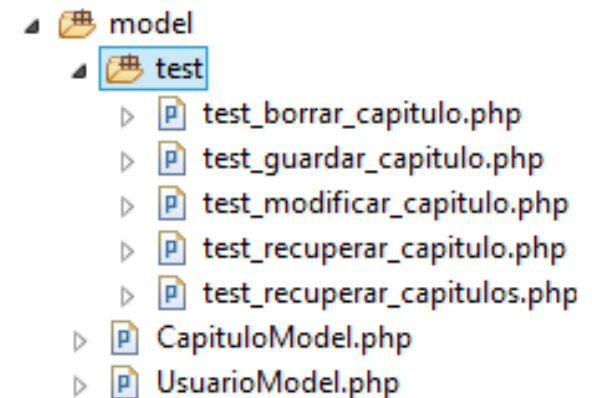
//Método que modifica un capítulo
public function modificar(){
    //preparar la consulta
    $consulta = "UPDATE capitulos SET
        capitulo=$this->capitulo,
        temporada=$this->temporada,
        titulo='$this->titulo',
        descripcion='$this->descripcion',
        duracion=$this->duracion,
        fecha_emision='$this->fecha_emision',
        imagen='$this->imagen'
        WHERE id=$this->id;";

    return Database::get()->query($consulta); //ejecutar la con:
}
?>
```



Probando el modelo

- Vamos a probar los métodos creados anteriormente.
- Para ello, creamos una carpeta `test` (que borraremos si subimos la aplicación a producción) con scripts *PHP* que prueben los métodos anteriores.
- Debemos asegurarnos que los resultados del test sean los deseados.





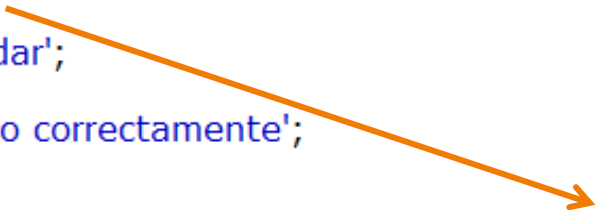
Ejemplo probando el modelo (guardar)

```
<?php
require '../config/Config.php';
require '../libraries/database_library.php';
require '../CapituloModel.php';

//PRUEBA DE GUARDADO
$capitulo = new CapituloModel();
$capitulo->capitulo = 1;
$capitulo->temporada = 2;
$capitulo->titulo = 'Marge se va con su profe de bolos';
$capitulo->descripcion = 'Marge conoce a un hombre y pone en duda su matrimonio';
$capitulo->duracion = 22;
$capitulo->fecha_emision = '1989-01-01';

if(!$capitulo->guardar())
    echo 'No se pudo guardar';
else
    echo 'Capítulo guardado correctamente';

?>
```



| | | | | | | | | |
|--------------------------|--|--|--|---|---|---|-----------------------------------|---|
| <input type="checkbox"/> |  Editar |  Copiar |  Borrar | 2 | 1 | 2 | Marge se va con su profe de bolos | Marge conoce a un hombre y pone en duda su matrimo... |
|--------------------------|--|--|--|---|---|---|-----------------------------------|---|




Ejemplo probando el modelo (recuperar todo)

```
<?php
require '../config/Config.php';
require '../libraries/database_library.php';
require '../CapituloModel.php';
```

```
$capitulos = CapituloModel::getCapitulos();
```

```
var_dump($capitulos);
```

```
?>
```



```
array (size=2)
  0 =>
    object(CapituloModel) [4]
      public 'id' => string '1' (length=1)
      public 'capitulo' => string '1' (length=1)
      public 'temporada' => string '1' (length=1)
      public 'titulo' => string 'Navidad sin paga' (length=16)
      public 'descripcion' => string 'Homer pierde la paga de navidad' (length=31)
      public 'duracion' => string '22' (length=2)
      public 'fecha_emision' => string '1988-12-25' (length=10)
      public 'imagen' => string 'images/no_image.png' (length=19)
  1 =>
    object(CapituloModel) [5]
      public 'id' => string '2' (length=1)
      public 'capitulo' => string '1' (length=1)
      public 'temporada' => string '2' (length=1)
      public 'titulo' => string 'Marge se va con su profe de bolos' (length=33)
      public 'descripcion' => string 'Marge conoce a un hombre y pone en duda su matrimonio' (length=53)
      public 'duracion' => string '22' (length=2)
      public 'fecha_emision' => string '1989-01-01' (length=10)
      public 'imagen' => string '' (length=0)
```



Ejemplo probando el modelo (recuperar todo)

```
<?php
require '../config/Config.php';
require '../libraries/database_library.php';
require '../CapituloModel.php';

$capitulo1 = CapituloModel::getCapitulo(1);
var_dump($capitulo1);

$capitulo2 = CapituloModel::getCapitulo();
var_dump($capitulo2);
```

?>



```
object(CapituloModel) [4]
  public 'id' => string '1' (length=1)
  public 'capitulo' => string '1' (length=1)
  public 'temporada' => string '1' (length=1)
  public 'titulo' => string 'Aventuras en la selva' (length=21)
  public 'descripcion' => string 'Homer pierde la paga de navidad'
  public 'duracion' => string '19' (length=2)
  public 'fecha_emision' => string '1988-12-25' (length=10)
  public 'imagen' => string 'images/no_image.png' (length=19)

null
```




Ejemplo probando el modelo (modificar)

```
<?php
require '../config/Config.php';
require '../libraries/database_library.php';
require '../CapituloModel.php';

$capitulo = CapituloModel::getCapitulo(1);

if(!$capitulo)
    echo 'No existe el capítulo';
else{
    $capitulo->titulo = 'Aventuras en la selva';
    $capitulo->duracion = 19;

    if($capitulo->modificar())
        echo 'Modificado correctamente';
    else
        echo 'No se pudo modificar';
}
?>
```



| id | capitulo | temporada | titulo | descripcion |
|----|----------|-----------|-----------------------------------|---|
| 1 | 1 | 1 | Aventuras en la selva | Homer pierde la paga de navidad |
| 2 | 1 | 2 | Marge se va con su profe de bolos | Marge conoce a un hombre y pone en duda su matrimo... |



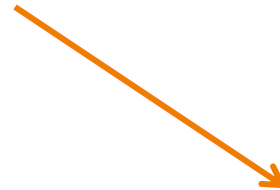
Ejemplo probando el modelo (borrar)

```
<?php
require '../config/Config.php';
require '../libraries/database_library.php';
require '../CapituloModel.php';

$capitulo1 = CapituloModel::getCapitulo(2);

if($capitulo1 && $capitulo1->borrar())
    echo "Borrado correctamente";
else
    echo "No se pudo borrar";

?>
```



| id | capitulo | temporada | titulo | descripcion |
|----|----------|-----------|-----------------------|---------------------------------|
| 1 | 1 | 1 | Aventuras en la selva | Homer pierde la paga de navidad |



Controladores y vistas

- A partir de este punto, comenzaremos a crear los métodos del controlador **Capítulo** y las vistas que necesitaremos para cada operación.
- **Primero veremos el controlador** y luego las vistas, aunque en este punto también voy a mencionar algunas más de las utilidades del *framework* que nos harán más sencilla la implementación..



El controlador Capitulo

- El controlador `Capitulo` es algo largo así que se encuentra dividido por sus métodos. Antes de cada método se describe el algoritmo implementando.
- Se implementará mediante la clase `Capitulo`, en el fichero `controller/Capitulo.php`, y contiene los métodos: `listar()`, `ver($id)`, `nuevo()`, `modificar($id)` y `borrar($id)`.
- Recordemos que los métodos se corresponderán con las operaciones que se pueden realizar, con capítulos de la serie los Simpsons, en la aplicación.



El controlador Capitulo (listar)

- La operación de **listar todos los capítulos** se implementa de la siguiente forma:
 - El controlador pide al modelo los capítulos (que éste recuperará de la BDD como vimos anteriormente).
 - El controlador carga **la vista de listado**, pasándole la información que debe mostrar: datos del usuario y lista de capítulos.
 - Si el usuario es admin, la vista mostrada será diferente, puesto que debe incluir las opciones de edición y borrado de capítulos.



Ejemplo controlador Capitulo (listar)

<?php

//CONTROLADOR CAPITULO: operaciones con los capítulos

class Capitulo **extends** Controller{

public function index(){ //método por defecto
 \$this->listar();
}

//para el listado de capitulos

public function listar(){
 //pedirle al modelo que recupere los capítulos
 \$this->load('model/CapituloModel.php');
 \$capitulos = CapituloModel::getCapitulos();

//pasarle los capítulos a la vista

\$datos = **array**();
\$datos['usuario'] = Login::getUsuario();
\$datos['capitulos'] = \$capitulos;

if(!Login::isAdmin()) //el admin verá una lista diferente
 \$this->load_view('view/capitulos/lista.php', \$datos);

else
 \$this->load_view('view/capitulos/admin/lista_admin.php', \$datos);
}

Hace uso de **login_library** que
se encuentra en la carpeta
libraries



El controlador Capitulo (ver)

- La operación de **ver un capítulo concreto** se implementa de la siguiente forma:
 - Se comprueba que el usuario esté identificado (como se describe en el caso de uso “ver detalles de un capítulo”).
 - El controlador pide al modelo los datos del capítulo concreto.
 - El controlador **carga la vista de detalles**, pasándole la información que debe mostrar: datos del usuario e información del capítulo.



Ejemplo controlador Capitulo (ver)

```
//método para ver un capítulo concreto
//solamente para los usuarios registrados
public function ver($id=0){
    //comprobar si el usuario está registrado e identificado
    if(!Login::getUsuario())
        throw new Exception('Solo para usuarios registrados');

    //pedirle al modelo que me pase los capítulos
    $this->load('model/CapituloModel.php');
    $capitulo = CapituloModel::getCapitulo($id);

    if(!$capitulo)
        throw new Exception('No se encuentra el capítulo');

    //pasarle el capítulo a la vista
    $datos = array();
    $datos['usuario'] = Login::getUsuario();
    $datos['capitulo'] = $capitulo;
    $this->load_view('view/capitulos/detalles.php', $datos);
}
```



El controlador Capitulo (nuevo)

- La operación de **guardar un capítulo nuevo** se implementa de la siguiente forma:
 - Se comprueba que el usuario esté identificado como administrador.
 - Si no llega la petición de “guardar”, se **muestra la vista con el formulario para la introducción de datos**.
 - Si llega la petición de guardar, se recuperan los datos que llegan por *POST*, se recupera la imagen y se le pide al modelo que guarde el capítulo en la BDD.
 - Finalmente el controlador **carga la vista de éxito**, pasándole la información que debe mostrar: datos del usuario y mensaje.



Ejemplo controlador Capitulo (nuevo 1 de 2)

```
//OPERACIONES DEL ADMINISTRADOR
//crear un nuevo capítulo
public function nuevo(){
    //comprobar que el usuario es ADMIN
    if(!Login::isAdmin())
        throw new Exception('Debes ser administrador');

    //si no me llegan los datos del nuevo capítulo
    if(empty($_POST['guardar'])){
        //cargamos la vista con el formulario
        $datos = array();
        $datos['usuario'] = Login::getUsuario();
        $this->load_view('view/capitulos/admin/nuevo.php', $datos);
    }else{
        //si llegan los datos del capítulo crearemos el nuevo capítulo
        $this->load('model/CapituloModel.php');
        $capitulo = new CapituloModel();

        //recuperamos los datos que llegan por POST
        $capitulo->capitulo = intval($_POST['capitulo']);
        $capitulo->temporada = intval($_POST['temporada']);
        $capitulo->titulo = Database::get()->real_escape_string($_POST['titulo']);
        $capitulo->descripcion = Database::get()->real_escape_string($_POST['descripcion']);
        $capitulo->duracion = intval($_POST['duracion']);
        $capitulo->fecha_emision = Database::get()->real_escape_string($_POST['fecha_emision']);
    }
}
```



Ejemplo controlador Capitulo (nuevo 2 de 2)

```
//PARA LA IMAGEN DEL CAPITULO!!!  
//pone la imagen por defecto  
$capitulo->imagen = Config::get()->image_not_found;  
  
//recuperar y guardar la imagen (solamente si ha sido enviada)  
if($_FILES['imagen']['error']!=4){  
    $upload = new Upload($_FILES['imagen'], 'images/capitulos/', 1000000);  
    $capitulo->imagen = $upload->upload_image();  
}  
  
//lo guardamos en la BDD  
if(!$capitulo->guardar())  
    throw new Exception('No se pudo guardar');  
  
//mostramos la vista de éxito  
$datos = array();  
$datos['usuario'] = Login::getUsuario();  
$datos['mensaje'] = 'Capítulo '.$capitulo->titulo.' guardado correctamente.';  
$this->load_view('view/exito.php', $datos);  
}  
}
```

Hace uso de **upload_library**
que se encuentra en la carpeta
libraries



El controlador Capitulo (modificar)

- La operación de **modificar un capítulo** se implementa de la siguiente forma:
 - Se comprueba que el usuario esté identificado como administrador.
 - Se le pide al modelo que recupere la información actual del capítulo a modificar de la BDD.
 - Si no llega la petición de “modificar”, se **muestra la vista con el formulario para la modificación de datos** (ya relleno con los datos del capítulo). A dicha vista hay que pasarle los datos del usuario actual y del capítulo que se desea modificar.



El controlador Capitulo (modificar)

- Si llega la petición de “modificar”, se recuperan los datos que llegan por *POST*, se recupera la nueva imagen (si se indicó) y se le pide al modelo que guarde el capítulo en la BDD.
- Se debe borrar la imagen antigua del sistema de ficheros del servidor.
- Finalmente el controlador **carga la vista de éxito**, pasándole la información que debe mostrar: datos del usuario y mensaje.
- A modo de ejemplo, mostraré también como hacer que cargue la vista de detalles o el listado en lugar de la vista de éxito (como comentarios en el código).



Ejemplo controlador Capitulo (modificar 1 de 3)

```
//modificar un capítulo existente
public function modificar($id=0){
    //comprobar si el usuario es admin
    if(!Login::isAdmin())
        throw new Exception('Debes ser administrador');

    //recuperar los datos del capítulo a modificar
    $this->load('model/CapituloModel.php');
    $capitulo = CapituloModel::getCapitulo($id);

    //comprobar que el capítulo se recuperó correctamente
    if(empty($capitulo))
        throw new Exception('No se encuentra el capítulo');

    //si no me mandan los nuevos datos
    if(empty($_POST['modificar'])){
        //poner el formulario
        $datos = array();
        $datos['usuario'] = Login::getUsuario();
        $datos['capitulo'] = $capitulo;
        $this->load_view('view/capitulos/admin/modificar.php', $datos);
    }
}
```



Ejemplo controlador Capitulo (modificar 2 de 3)

```
//si me mandan los datos
}else{
    //recuperar los nuevos datos que llegan por POST
    $capitulo->capitulo = intval($_POST['capitulo']);
    $capitulo->temporada = intval($_POST['temporada']);
    $capitulo->titulo = Database::get()->real_escape_string($_POST['titulo']);
    $capitulo->descripcion = Database::get()->real_escape_string($_POST['descripcion']);
    $capitulo->duracion = intval($_POST['duracion']);
    $capitulo->fecha_emision = Database::get()->real_escape_string($_POST['fecha_emision']);

    //recupera y guardar la imagen si se deseaba cambiar
    if($_FILES['imagen']['error']!=4){
        //prepara la carga de nueva imagen
        $upload = new Upload($_FILES['imagen'], 'images/capitulos/', 1000000);

        //guarda la imagen antigua en una var para borrarla después si todo ha funcionado
        $old_img = $capitulo->imagen;

        //sube la nueva imagen
        $capitulo->imagen = $upload->upload_image();
    }

    //actualizamos sobre la BDD
    if(!$capitulo->modificar())
        throw new Exception('No se pudo modificar el capítulo');
```



Ejemplo controlador Capitulo (modificar 3 de 3)

```
//borra la imagen anterior (solamente en caso que no sea imagen por defecto)
if(!empty($old_img) && $old_img!= Config::get()->image_not_found)
    @unlink($old_img);
```

```
//poner la vista de éxito (o la de detalles del capítulo
//con la de éxito
$datos = array();
$datos['usuario'] = Login::getUsuario();
$datos['mensaje'] = 'Capítulo '.$capitulo->titulo.' guardado correctamente.';
$this->load_view('view/exito.php', $datos);
```

```
//EJEMPLO: si queremos poner la vista de detalles en vez de "exito"
/*$datos = array();
$datos['usuario'] = Login::getUsuario();
$datos['capitulo'] = $capitulo;
$this->load_view('view/capitulos/detalles.php', $datos);*/
```

```
//EJEMPLO: si queremos ir al listado en vez de "exito"
//$this->listar();
```

```
}
}
```



El controlador Capitulo (borrar)

- La operación de **borrar un capítulo** se implementa de la siguiente forma:
 - Se comprueba que el usuario esté identificado como administrador.
 - Se le pide al modelo que recupere la información actual del capítulo a borrar de la BDD.
 - Si no llega la petición de “borrar”, se **muestra la vista de confirmación de borrado**. A dicha vista hay que pasarle los datos del usuario actual y del capítulo que se desea borrar.



El controlador Capitulo (borrar)

- Si llega la petición de “borrar”, se le pide al modelo que borre el capítulo de la BDD y se borra el fichero de imagen del directorio de imágenes en el servidor.
- Una vez completado el borrado, se **muestra la vista de éxito**. A dicha vista hay que pasarle los datos del usuario actual y un mensaje.



Ejemplo controlador Capitulo (borrar 1 de 2)

```
//borrar un capítulo
public function borrar($id=0){
    //comprobar si el usuario es admin
    if(!Login::isAdmin())
        throw new Exception('Debes ser administrador');

    //recuperar el capítulo a borrar de la BDD
    $this->load('model/CapituloModel.php');
    $capitulo = CapituloModel::getCapitulo($id);

    //comprobar que el capítulo se recuperó correctamente
    if(empty($capitulo))
        throw new Exception('No se encuentra el capítulo');

    //si no me están confirmando el borrado
    if(empty($_POST['borrar'])){
        //mostrar vista de confirmación de borrado
        $datos = array();
        $datos['usuario'] = Login::getUsuario();
        $datos['capitulo'] = $capitulo;
        $this->load_view('view/capitulos/admin/borrar.php', $datos);
    }
}
```




Ejemplo controlador Capitulo (borrar 2 de 2)

```
//si me están dando la confirmación
}else{
    //borrar el capítulo
    if(!$capitulo->borrar())
        throw new Exception('Se produjo un error al borrar');

    //borra la imagen (solamente en caso que no sea imagen por defecto)
    if($capitulo->imagen!=Config::get()->image_not_found)
        @unlink($capitulo->imagen);

    //mostrar la vista de éxito
    $datos = array();
    $datos['usuario'] = Login::getUsuario();
    $datos['mensaje'] = 'Capítulo '.$capitulo->titulo.' borrado correctamente.';
    $this->load_view('view/exito.php', $datos);
}
}
}
?>
```

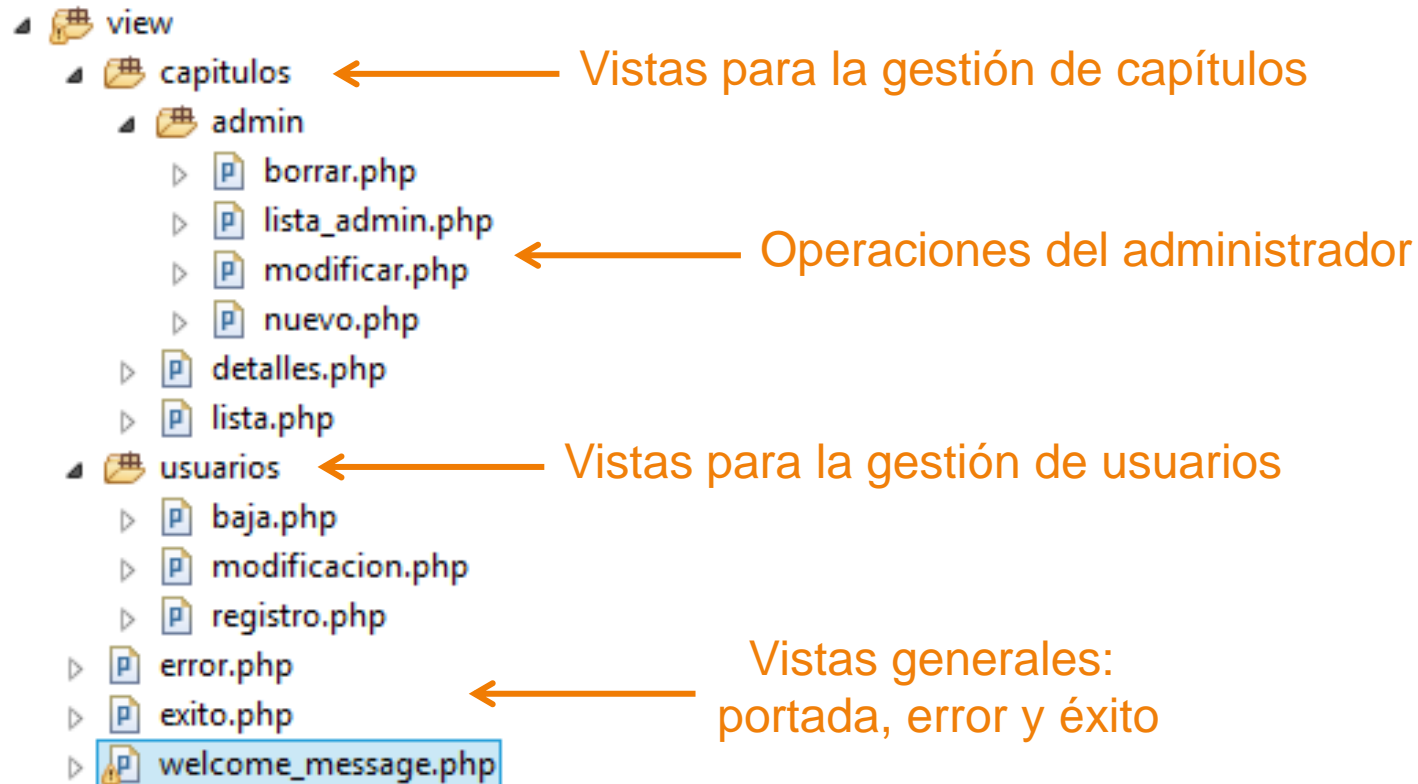


Creando vistas

- Otra de las cosas que debemos implementar son las **vistas**.
- Son el resultado de la maquetación de los diseños de la GUI, en los que debemos incorporar el código *PHP* adecuado para mostrar la información dinámicamente.
- Las vistas contienen el *HTML* y *CSS* visibles por el usuario final y conforman la interfaz gráfica de usuario.
- NOTA: en las vistas incluiremos llamadas a los métodos del *template* para ahorrar faena y simplificar los cambios.



Ejemplo carpetas para vistas





Sobre la navegación...

- Al crear las vistas, **necesitaremos poner enlaces** a cada una de las operaciones de que dispone nuestra aplicación.
- Si no usamos *URLs* amigables (más adelante), debemos tener en cuenta que **todas las operaciones pasan por el fichero `index.php`** (que invoca al controlador frontal).
- Las peticiones van en parámetros *HTTP GET*, de forma:
`index.php?controlador=capitulo&operacion=ver¶metro=3`
- En ese caso se mostrarían los datos del capítulo con código 3.



Sobre la navegación...

- Observad los enlaces del menú (en el fichero Template.php) a modo de ejemplo:

```
//PONE EL MENU DE LA PAGINA
public static function menu(){
    ?>
    <ul class="menu">
        <li><a href="index.php">Inicio</a></li>
        <li><a href="index.php?controlador=capitulo">Capítulos</a></li>
        <li><a href="index.php?controlador=Usuario&operacion=registro">Registro</a></li>
    </ul>
    <?php
}
```



Funcionamiento real

`index.php?controlador=capitulo&operacion=ver¶metro=3`

1. El fichero `index.php` carga el controlador frontal.
2. El controlador frontal llama al controlador `Capitulo`, ejecutando el método `ver()` con el parámetro 3.
3. El método `ver()` usa el modelo `CapituloModel` para recuperar los datos del capítulo 3 y los retorna al controlador `Capitulo`.
4. El controlador `Capitulo` carga la vista `detalles.php` con los datos.



Ejemplo vista para el listado de capítulos (1 de 3)

```
<!DOCTYPE html>
<html>
  <head>
    <base href="<?php echo Config::get()->url_base;?>" />
    <meta charset="UTF-8">
    <title>Listado de capítulos</title>
    <link rel="stylesheet" type="text/css" href="<?php echo Config::get()->css;?>" />
  </head>

  <body>
    <?php
      Template::header(); //muestra el encabezado

      if(!$usuario)
        Template::login(); //muestra el formulario de login
      else
        Template::logout($usuario); //muestra el formulario de logout

      Template::menu(); //muestra el menú principal

      if($usuario && $usuario->admin) //si el usuario es admin
        Template::menuadmin(); //pone el menú de administrador
    ?>
```



Ejemplo vista para el listado de capítulos (2 de 3)


```
<section id="content">
  <h2>Listado de capítulos</h2>
  <div class="titulos flex-container">
    <div class="flex">Cap.</div><div class="flex">Temp.</div>
    <div class="flex">Título</div><div class="flex">Duración</div>
    <div class="flex">Fecha</div><div class="flex">Operaciones</div>
  </div>
  <?php
    foreach($capitulos as $c){
      echo "<div class='contenido flex-container'>";
      echo "<div class='flex'>$c->capitulo</div>";
      echo "<div class='flex'>$c->temporada</div>";
      echo "<div class='flex'>$c->titulo</div>";
      echo "<div class='flex'>$c->duracion</div>";
      echo "<div class='flex'>$c->fecha_emision</div>";
      echo "<div class='flex'>";
      echo "<a href='index.php?controlador=capitulo&operacion=ver&parametro=$c->id'>";
      echo "<img class='boton' src='images/botones/ver.png'></a>";
      echo "</div>";
      echo "</div>";
    }
  <?>
  <a class="volver" href="index.php">Volver a inicio</a>
</section>
<?php Template::footer();?>
</body>
</html>
```




Ejemplo vista para el listado de capítulos (3 de 3)

Simpsons Chapter List




Listado completo de capítulos de los Simpsons



User: Password:

[Inicio](#) [Capítulos](#) [Registro](#)

Listado de capítulos

| Cap. | Temp. | Título | Duración | Fecha | Operaciones |
|------|-------|-------------|----------|------------|---|
| 1 | 1 | lorem ipsum | 99 | 2016-09-01 |  |
| 2 | 1 | lorem ipsum | 99 | 2016-09-01 |  |
| 3 | 1 | lorem ipsum | 99 | 2016-09-01 |  |
| 4 | 1 | lorem ipsum | 99 | 2016-09-01 |  |
| 5 | 1 | lorem ipsum | 99 | 2016-09-01 |  |

[Volver a inicio](#)

RobS micro Framework - solo para fines docentes, Robert Sallent - CIFO'16. Twitter: @robertsallent



Resultado final

- El resultado final de la aplicación desarrollada a lo largo de este tutorial lo podéis encontrar en:

<http://ejemplos.robertsallent.com/simpsons>



Resultado final



Ejercicios

Creando aplicaciones con el RMF





Ejercicios

1. Completa el ejemplo de los Simpsons, de forma que el administrador pueda gestionar la información de **personajes**.

Los usuarios (tanto los registrados como los que no), podrán ver un listado de personajes y ver sus detalles en otra vista diferente.





Ejercicios

The screenshots show a web application for 'The Simpsons' with the following pages:

- Home Page:** Features the title 'THE SIMPSONS', a subtitle 'LISTADO DE CAPITULOS DE LOS SIMPSONS', a navigation menu (Inicio, PERSONAJES, CAPITULOS, REGISTRO), and a 'BIENVENIDOS' section with a welcome message and a 'Logout' button.
- Character Creation Page:** Titled 'NUEVO PERSONAJE', it includes a form for 'Nombre', 'Descripción', 'Edad', and 'Imagen'. It also has a 'Validar al Estado' button and a 'Logout' button.
- Character Details Page:** Titled 'DETALLES DE BART SIMPSON', it displays the character's name, age, and a biography. It also includes a 'Volver al Estado' button and a 'Logout' button.

Arrows indicate the flow from the Home Page to the Character Creation Page, and from the Character Creation Page to the Character Details Page.



Ejercicios

2. Incluye la posibilidad de exportar el listado de capítulos a *XML* y a *JSON*. Para ello, incluiremos uno o dos métodos nuevos en el controlador `Capitulo`.

Deben recuperar el listado de capítulos y presentarlo en el formato especificado (no hace falta que llamen a una vista).

Ejemplo:

```
index.php?controlador=Capitulo&operacion=exportar&parametro=xml
```

O bien:

```
index.php?controlador=Capitulo&operacion=toxml
```

```
index.php?controlador=Capitulo&operacion=tojson
```