UNIDAD5.ACTIVIDAD1

DESARROLLO

Francisco Javier Signes Costa 2º DAW online

En esta actividad nos vamos a familiarizar con algunas de las funcionalidades que nos proporciona Codeigniter 4. Deberás investigar las librerías o helpers que se indican abajo y crear un documento explicando la utilidad y cómo emplearlas. Ten en cuenta que las herramientas que vas a investigar ahora, las tendrás que utilizar en unidades posteriores, así que hacer un buen trabajo de investigación te beneficiará en el tiempo.

Uno de los objetivos principales de esta actividad es acercarnos y familiarizarnos con las documentaciones oficiales de los frameworks.

En esta actividad, deberemos de crear un documento de tipo Word, con una extensión que no sea mayor a 5 páginas, y que enviarás a tu profesor por email o mediante la plataforma educativa, detallando los siguientes puntos en los que analizaremos los siguientes puntos:

Debes investigar las siguientes librerías y helpers de Codeigniter 4 explicando para qué son útiles y cómo podemos utilizarlas.

- Library Reference – Pagination
- Library Reference – Working with Files
- Library Reference – File Collections
- Library Reference – Image Manipulation Class
- Helpers – Date Helper
- Helpers – XML Helper

Pagination

Pagination es una librería de Codeigniter que permite soportar múltiples paginaciones en una página individual.

En la mayoría de los casos vamos a usar la librería Pager para paginar los resultados provenientes de una consulta a una base de datos, de manera que podemos ordenar la salida de estos de una manera ordenada.

La forma más habitual de trabajar es usar la clase Model con el método paginate().

Tal y como nos indica la documentación de Codeigniter y, en el ejemplo de mostrar un listado de usuarios de una determinada aplicación, el controlador se vería así:

```php
<?php

namespace App\Controllers;

use App\Models\UserModel;

class UserController extends BaseController
{
    public function index()
    {
        $model = model(UserModel::class);

        $data = [
            'users' => $model->paginate(10),
            'pager' => $model->pager,
        ];

        return view('users/index', $data);
    }
}
```

Donde creamos una nueva instancia de nuestro UserModel. Después rellenamos los datos que enviaremos a la vista.

Estos datos los recogemos en un array $data donde paginamos 10 usuarios por página. Después debemos enviar la instancia Pager.

Para recoger estos datos lo haremos desde la vista con la variable $pager.

Podemos customizar a nuestra query añadiendo antes del método paginate():

- **Where.** Si deseamos añadir condiciones
- **Join**

Generación de la vista

Codeigniter nos permite centrarnos en la creación de la vista en sí, sólo los enlaces, es decir, el código de paginación propiamente dicho.

En la vista, podemos usar una serie de métodos que nos facilitan la edición de los links para las páginas.

Método *setSurroundCount()*

Especifica el número de links que queremos mostrar al lado de la página actual.

Métodos *getPrevious() & getNext()*

Devuelven la URL de la página anterior o la siguiente. Están basados en el parámetro pasado a *setSurroundCount(),*por lo que mostrará más o menos links a derecha e izquierda del actual.

Métodos *getFirst() & getLast()*

Igual que los métodos descritos anteriormente, pero indicando el primero y el último.

Estos son sólo una muestra del total que nos muestra la documentación de Codeigniter 4.

En la versión 4.6.0 se ha implementado el poder ver el total de ítems en la página. Tenemos estos nuevos métodos:

- *getTotal()*
- *getPerPage()*
- *getPerPageStart()*
- *getPerPageEnd()*

Working with Files

Codeigniter provee de una clase File que envuelve la clase **SplFileInfo** con nuevos métodos.

Creamos una nueva instancia de File pasando la dirección al archivo en el constructor.

$file = new \CodeIgniter\Files\Files($path);

Podemos pasar un argumento adicional para saber si el fichero existe o no.

Una vez creada la instancia ya podríamos usar todas las funcionalidades que nos da la clase SplFileInfo de PHP.

Nuevas funcionalidades

Adicionalmente podemos generar un nombre seguro con la función *getRandomName();* con la función *getSize()* podemos averiguar el tamaño en bytes del archivo. Con *getMimeType()* podemos saber qué tipo de archivo es y con *gessExtension()* nos devuelve la extensión del archivo.

Finalmente podemos mover archivos y renombrarlos con un segundo parámetro si así lo queremos.

Anexo

- SplFileInfo::__construct — Construct a new SplFileInfo object
- SplFileInfo::getATime — Gets last access time of the file
- SplFileInfo::getBasename — Gets the base name of the file
- SplFileInfo::getCTime — Gets the inode change time
- SplFileInfo::getExtension — Gets the file extension
- SplFileInfo::getFileInfo — Gets an SplFileInfo object for the file
- SplFileInfo::getFilename — Gets the filename
- SplFileInfo::getGroup — Gets the file group
- SplFileInfo::getInode — Gets the inode for the file
- SplFileInfo::getLinkTarget — Gets the target of a link
- SplFileInfo::getMTime — Gets the last modified time
- SplFileInfo::getOwner — Gets the owner of the file
- SplFileInfo::getPath — Gets the path without filename
- SplFileInfo::getPathInfo — Gets an SplFileInfo object for the path
- SplFileInfo::getPathname — Gets the path to the file
- SplFileInfo::getPerms — Gets file permissions
- SplFileInfo::getRealPath — Gets absolute path to file
- SplFileInfo::getSize — Gets file size
- SplFileInfo::getType — Gets file type
- SplFileInfo::isDir — Tells if the file is a directory
- SplFileInfo::isExecutable — Tells if the file is executable
- SplFileInfo::isFile — Tells if the object references a regular file
- SplFileInfo::isLink — Tells if the file is a link
- SplFileInfo::isReadable — Tells if file is readable
- SplFileInfo::isWritable — Tells if the entry is writable
- SplFileInfo::openFile — Gets an SplFileObject object for the file
- SplFileInfo::setFileClass — Sets the class used with SplFileInfo::openFile
- SplFileInfo::setInfoClass — Sets the class used with SplFileInfo::getFileInfo and SplFileInfo::getPathInfo
- SplFileInfo::__toString — Returns the path to the file as a string

Ilustración 1: Métodos de la clase SplFileInfo

File Collections

Con la clase FileCollection se facilita la labor de trabajar con grupos de archivos. Se facilita, en definitiva, la localización y el trabajo con grupos de archivos en todo el sistema de ficheros.

Básicamente, FileCollection es un array de archivos que construimos.

Para empezar debemos hacer uso de un constructor:

__construct(string[] $files = [])

El constructor acepta un array de rutas de archivos para iniciar la colección.

Con el método *define()* permitimos que las clases hijas definan sus propios archivos iniciales. Este método es llamado por el constructor y permite colecciones predeterminadas sin tener que usar sus métodos.

```php
<?php

use CodeIgniter\Files\FileCollection;

$files = new FileCollection([
    FCPATH . 'index.php',
    ROOTPATH . 'spark',
]);
$files->addDirectory(APPPATH . 'Filters');
```

Ilustración 2: Array FileCollection

Ingresando archivos

Disponemos de varios métodos para ingresar archivos

- *addFile(string $file) / addFiles(array $files)* -> para añadir las rutas de los archivos
- *removeFile(string $file) / removeFiles(array $files)*-> para eliminar archivos de la lista

- *addDirectory(string $directory, bool $recursive = false)* -> añade todos los archivos del directorio.

Filtrando archivos

Tenemos varios métodos para filtrar archivos

- *remove o retainPattern(string $pattern, string $scope = null)* -> filtra la lista actual del archivo conforme al patrón, quitando o dejando los archivos que indiquemos.

```php
<?php

$files->removeFile(APPPATH . 'Filters/DevelopToolbar.php');

$files->removePattern('#\.gitkeep#');
$files->retainPattern('*.php');
```

Ilustración 3:Ejemplo de eliminación de archivos

Recuperando archivos

Con *get(): string[]* devolvemos un array de todos los archivos cargados.

Image Manipulation Class

La clase de CodeIgniter 4 Manipulation permite:

- Ajuste de tamaño de imágenes
- Creación de Thumbnail
- Cropping de imágenes
- Rotación de imágenes
- Marcas de agua

Como en otras clases de CodeIgniter, la clase la inicializamos en el controlador llamando a la función global service()

```php
<?php

$image = service('image');
```

Ilustración 4: Iniciando la clase image

Procesando una imagen

Independientemente del proceso de manipulación que queramos hacer a la imagen (cambio de tamaño, recortes, rotación, etc.) vamos a proceder de la misma manera. Se establecen las preferencias y después llamamos a las funciones disponibles.

Como ejemplo, CodeIgniter nos muestra la creación de una imagen tipo Thumbnail:

```php
<?php

$image->withFile('/path/to/image/mypic.jpg')
    ->fit(100, 100, 'center')
    ->save('/path/to/image/mypic_thumb.jpg');
```

Ilustración 5; Thumbnail con CodeIgniter

Buscamos la imagen mypic.jpg en el path correspondiente, creamos una nueva y la guardamos como mypic_thumb.jpg. El método *fit()* intentará encontrar la mejor ratio para hacer cropping y resize.

Tenemos varios métodos de modificación de imagen:

- *$image->crop()*
- *$image->convert()*
- *$image->fit()*
- *$image->flatten()*
- *$image->flip()*
- *$image->resize()*
- *$image->rotate()*
- *$image->text()*

La documentación de CodeIgniter explica con detalle cada uno de ellos.

Date Helper

El archivo Date Helper contiene funciones que ayudan a trabajar con fechas.

Para cargar el helper:

```php
<?php

helper('date');
```

Ilustración 6: Cargando el helper Date

Funciones disponibles

Now([$timezone - null])

Se recomienda, no obstante, usar la clase Time para conseguir el timestamp de Unix:

Time::now()->getTimeStamp()

Si una zona horaria no está disponible devolverá el resultado de *time()*.

Si la zona horaria está disponible, devolverá un tiempo que está desplazado conforme a lo que ofrece la función *time()*. Si no vamos a referenciar nuestro sistema de modo que el usuario elija su franja horaria, no tiene sentido usar *timezone()* frente a *time()*.

XML Helper

Este helper contiene funciones que nos van a ayudar a trabajar con datos en formato XML.

Lo cargamos de la siguiente manera:

```php
<?php

helper('xml');
```

Ilustración 7: Cargando el helper para XML

Funciones disponibles

Xml_convert($str[,$protect_all=false])

$str -> es el string a convertir

$protect_all -> convertir todo el contenido susceptible de ser una entidad

Devuelve una cadena convertida a formato XML.