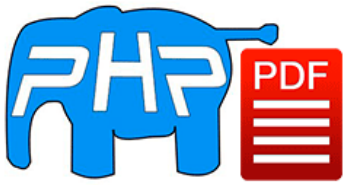
Editar formularios PDF con PHP (y III)



Ahora sí. Ya tenemos el conocimiento básico acerca de qué queremos hacer, qué herramientas vamos a usar, y cómo está constituido, internamente, un formulario PDF con campos editables. Ha llegado el momento de poner los dedos sobre el teclado, y empezar a programar. En este artículo vamos a ver como actúa todo el proceso PHP necesario.

EL ARCHIVO index.php

El archivo index.php se inicia con lo básico. Esto es: cargar la clase que se va a ocupar de hacer el trabajo, y crear un objeto de dicha clase, como ves a continuación:

```
include ('includes/pdf_data_injection.class.php');
$objPDF = new PDFDataInjection();
```

De la clase nos ocuparemos en su momento. Por ahora, baste saber que con crear el objeto se inicializan las variables necesarias, y se identifica el sistema operativo sobre el que se está trabajando.

Lo siguiente que hacemos es establecer las tres rutas principales: aquella en la que está el PDF original que se va a modificar, la ruta temporal donde se guardará el archivo FDF para modificar los valores de los campos que deseemos, y la ruta de destino, donde se guardará la copia del formulario PDF, una vez modificado. De esto se ocupan las siguientes líneas:

```
$objPDF->setSourcePath('./origen/');
$objPDF->setTempPath('./temp/');
$objPDF->setDestinationPath('./dest/');
```

Estas rutas deben poder ser modificables, como lo hacemos aquí, porque, evidentemente, tú puedes necesitar otras distintas en cada caso.

Lo siguiente es indicar cual es el nombre del archivo con el formulario PDF original, como se ve a continuación:

```
$objPDF->setPDF('formulario.pdf');
```

Tenemos que crear una matriz con los campos a los que queramos asignarles un valor por defecto. Esta matriz tendrá un elemento por campo. La clave del elemento es el nombre del campo, y el elemento es el valor que queremos asignarle al campo, así:

```
$datos = [
    'campo_edad'=>'2',
    'campo_comunitario'=>'comunitario',
    'campo_ciudad'=>'Madrid',
];
```

Tienes que tener en cuenta que el formulario podrá tener más campos de los que hay en esta matriz. Los que no incluyas en la matriz no serán afectados, permaneciendo en ellos el valor que tenían, o ningún valor, si no lo tenía previamente.

Una vez que hemos definido la matriz de datos que queremos para el formulario, debemos pasársela al objeto que representa al mismo. Lo hacemos así:

```
$objPDF->setFormData($datos);
```

El siguiente paso es crear el fichero FDF (del que hablamos en el artículo anterior), y almacenarlo, de modo provisional, en el directorio que definimos como temporal (en el ejemplo, /temp/), así:

```
$objPDF->createFDF();
```

Ahora tenemos que coger los datos que han entrado en el objeto a través del método setFormData() y sustituir, en el FDF, los valores que pudiera haber en esos datos por los que hemos incluido en la matriz, así:

```
$objPDF->insertData();
```

En este punto, ya está modificado el fichero FDF. Sin embargo, este no es el que vamos a obtener. Nosotros necesitamos un PDF, con los datos cumplimentados en los campos del formulario afectados. Por lo tanto, hace falta sacar una copia del PDF original en la que se inyecte el contenido modificado del FDF, y almacenarla en el directorio que hayamos establecido como destino. Después, es necesario eliminar el FDF, ya que ya es totalmente innecesario, y puede estar ocupando un espacio en disco (además de una entrada en el índice del mismo), que no sirve para nada. Lo hacemos así:

```
$objPDF->injectDataInPDF();
```

CONSIDERACIONES FINALES

Con esto tenemos creado el formulario PDF con los campos cumplimentados, en el directorio que hayamos establecido. Sin embargo, hay que tener en cuenta una cosa. Dado que puede haber múltiples concurrencias simultáneas, de distintos usuarios, cada PDF final debe tener un nombre único, para evitar que se sobrescriban unos a otros, y que se le envíe a un usuario un PDF con los datos introducidos para otro usuario. Por lo tanto, tenemos que recuperar el nombre del PDF generado por un objeto concreto en un ciclo de ejecución específico. Esto lo hacemos así:

```
$FinalPDFName = $objPDF->getFinalPDFName();
```

En cuanto al código de la clase PDFDataInjection, en principio podríamos decir que, como cualquier clase de PHP, el funcionamiento interno no es relevante. Lo que nos importa, que es lo que hemos descrito en este artículo, es como funciona un objeto creado a partir de la clase. Sin embargo, por si tienes curiosidad, te la he dejado en el ejemplo de código de este artículo, con una gran profusión de comentarios, para que puedas experimentar con ella. Tienes todo para descargar en GitHub, en https://github.com/eldesvandejose/PDFDataInjection.