Programación Avanzada
Grado en Ingeniería Informática en Sistemas de Información - Curso 2017/2018

EPD 5: Manejo de ficheros y procesamiento avanzado de formularios

La entrega del trabajo se hará a través de la tarea correspondiente en el Campus Virtual. Pasado el límite de entrega se aceptará el envío del trabajo, con una penalización de 2 puntos sobre 10 de la calificación por cada hora o fracción de retraso. La entrega consistirá en un único fichero comprimido en formato ZIP cuyo nombre deberá ser de la forma *equipoXX.zip*, donde *XX* serán dos cifras que indicará el número del equipo. Por ejemplo, equipo07.zip. Este fichero contendrá una serie de carpetas cuyo nombre deberá ser de la forma *ejY* o *pZ*, donde *Y* y *Z* representan, respectivamente, el número de cada ejercicio o problema del presente guión. Dentro de dichas carpetas se incluirán exclusivamente los archivos necesarios en la resolución del correspondiente ejercicio o problema. Las rutas de los ficheros empleados serán relativas, a fin de que las resoluciones a los ejercicios y problemas puedan ser examinadas en cualquier equipo. Cualquier entrega que no cumpla las reglas de nombrado, el formato de compresión del archivo o el contenido de los archivos del mismo, será penalizada con 2 puntos sobre 10 por cada incumplimiento.

Objetivos

- Construir aplicaciones de PHP con capacidad para la manipulación de ficheros.
- Habilitar códigos de PHP para el manejo de ficheros adjuntos como campos de formularios.
- Aplicar filtros a la entrada de páginas PHP para dotarlas de mayor robustez y seguridad.

Conceptos

1. Manejo de ficheros en PHP

PHP dispone de un nutrido conjunto de funciones para manipular ficheros. El manejo básico de ficheros es similar al que se practica en el lenguaje C, por lo que le resultará familiar.

Puede encontrar más información sobre el manejo básico de ficheros en el capítulo 9 de [1] y material de referencia sobre las funciones incluidas en PHP para el manejo de ficheros en el apéndice B de [1] y en [2].

2. Formularios adjuntando ficheros

PHP ofrece la posibilidad de manejar los ficheros adjuntos de una manera sencilla como campos en un formulario. Los ficheros se reciben automáticamente y sus propiedades (nombre, tamaño, etc.) pueden ser accedidas mediante el uso de la variable superglobal $_FILES.

Al recibir automáticamente estos ficheros, PHP guarda éstos en una localización temporal. Una vez terminada la ejecución de la página que los recibió, los ficheros recibidos desde un formulario serán borrados automáticamente. Por tanto, si se desea conservar un archivo adjunto, será necesario mover éste a otro directorio distinto al de la localización temporal. Para realizar esta acción se emplea la función *move_uploaded_file*.

Puede obtener más información sobre el manejo de ficheros adjuntos a formularios en el capítulo 10 de [1] y en [3].

3. Filtrado de campos de formularios

Filtrar las entradas a las páginas PHP desde formularios es una tarea muy importante para garantizar el correcto funcionamiento y seguridad de cualquier aplicación web. El proceso de filtrado incluye la comprobación de la validez de los datos suministrados como entrada a un formulario y/o el saneamiento de estos datos para evitar valores que puedan poner en riesgo la seguridad del sistema.

La tarea de filtrado se puede realizar empleando expresiones regulares como las vistas en las sesiones de Enseñanzas Básicas de la asignatura. Puede obtener en [4] más información sobre la definición y aplicación de expresiones regulares en PHP.

Finalmente, además del mecanismo anterior, PHP incluye un paquete que permite realizar de una forma sencilla el filtrado de las entradas: el paquete *Filter*. Este paquete engloba a un conjunto de funciones y filtros programados que permitirán realizar la tarea de filtrado de forma cómoda. Puede encontrar una introducción al uso del paquete *Filter* en [5] y una referencia sobre sus filtros y funciones en [6].

Bibliografía Básica

1. PHP 5: fast & easy web development. Julie Meloni. Thomson Course Technology, 2004. Parte II.
   http://0-site.ebrary.com.athenea.upo.es/lib/bupo/Doc?id=10058862
2. Referencia de funciones de manejo de ficheros.
   http://www.php.net/manual/en/ref.filesystem.php
3. Manejo de envío de archivos.
   http://www.php.net/manual/en/features.file-upload.php
4. Referencia de funciones de utilidad para expresiones regulares.
   http://www.php.net/manual/es/book.pcre.php
5. PHP Built-in Input filtering.
   http://devzone.zend.com/703/php-built-in-input-filtering/
6. Referencia de funciones de filtro.
   http://www.php.net/manual/en/book.filter.php

Experimentos

E1. (45 mins.) Cree, dentro de un nuevo proyecto PHP de NetBeans, un archivo de PHP y emplee el siguiente código:

```
<html>
  <body>
  <?php
          if(isset($_GET['envio'])){
                  if ($_FILES['archivo']['error'] > 0)
                          echo 'Error: ' . $_FILES['archivo']['error'] . '<br />';
                  else{
                    echo 'Nombre: ' . $_FILES['archivo']['name'] . '<br />';
                    echo 'Tipo: ' . $_FILES['archivo']['type'] . '<br />';
                    echo 'Tama&ntilde;o: ' . ($_FILES['archivo']['size'] / 1024) . ' Kb<br />';
                    echo 'Almacenado en: ' . $_FILES['archivo']['tmp_name'];
                    }
          }

          if(!isset($_GET['envio'])){
                  ?>
                          <h1> Env&iacute;a un archivo </h1>
                          <form method="get">
                                  <input type="file" name="archivo" /><br />
                                  <input type="submit" name="envio" value="Enviar" />
                                  <input type="reset" name="rest" value="Restaurar" />
                          </form>
                  <?php
          }
  ?>
  </body>
</html>
```

a) ¿Cuál es el propósito del código?

b) Ejecute la página y suministre cualquier archivo al formulario ¿La salida es la esperada? ¿Por qué? Solucione el problema.

c) Suministre en el formulario un fichero de gran tamaño ¿Qué ocurre? ¿Por qué?

Ejercicios

EJ1. (30 mins.) Amplíe la página desarrollada en el problema 2 del guión de la EPD anterior de tal forma que permita considerar textos dirigidos a un servicio de soporte técnico de redes. En concreto, el texto deberá contener una dirección IP origen, una dirección IP destino y dos números de puertos TCP en el rango 1000-1500. Para ello, verifique que en el texto aparecen al menos dos direcciones IP y dos números comprendidos en el intervalo [1000, 1500]. Use la función filter_var() de PHP, o bien expresiones regulares, para detectar este tipo de contenidos en el texto.

EJ2. (20 mins.) Realice una página en PHP que muestre en primer lugar un formulario para colaborar con una web dedicada a mostrar rutas de senderismo de Andalucía, donde el usuario introducirá en primer lugar sus datos personales, que incluirán: su nombre completo, su dirección de correo electrónico, su usuario de la aplicación Twitter, su teléfono fijo y su móvil. El formulario también incluirá una *lista desplegable,* que incluirá las 8 provincias de Andalucía y, por defecto, estará seleccionada la provincia "*Sevilla*". Por último, el formulario mostrará un área de texto donde se describa la ruta de senderismo que queremos dar a conocer. La misma página que contiene el formulario deberá procesarlo, comprobando que todos los campos están rellenos, si todo está correcto presentará una página a modo de resumen de los datos enviados, en caso incorrecto solicitará la introducción del campo que falte.

EJ3. (20 mins.) Partiendo del formulario del ejercicio 2 mejoraremos la capacidad del mismo. Al procesar el formulario se deberá, además de comprobar que todos los campos están rellenos, que su formato es el adecuado siguiendo estas condiciones: el nombre empieza por mayúscula, el correo electrónico y el usuario de la aplicación Twitter tiene el formato correcto, el teléfono fijo empieza por 9, y el móvil por 6. Se debe garantizar el saneamiento de los mismos. En caso de error deberá mostrar junto al campo erróneo el tipo de error cometido en color rojo (puede emplear CSS para esto), así como un ejemplo de campo válido.

Problemas

P1. (90 mins.) Cree un sistema, compuesto por varias páginas PHP, que permita gestionar vuelos y destinos de aerolíneas.

La primera página del sistema permitirá dar de alta una aerolínea y sus destinos. En ella se pedirá al usuario el nombre de la aerolínea y el número de destinos en los que opera. Al enviar esta información, se proporcionará al usuario un nuevo formulario en el que se podrán indicar, mediante listas desplegables (campos *select*), las ciudades de cada uno de los destinos. Considere para ello un vector con el siguiente repertorio de ciudades posibles: Roma, Paris, Londres, Dublin, Sevilla, Madrid, Barcelona y Amsterdam. Este repertorio de ciudades deberá ser leído de un archivo de texto que contendrá dichas ciudades separadas por comas (,). Recogidos todos estos datos, se escribirán en dos archivos. El primero almacenará dos campos: un código autonumérico y el nombre de cada aerolínea. El código autonumérico deberá ser generado a partir del siguiente número al máximo de los almacenados en el fichero. El segundo almacenará los nombres de los destinos de cada aerolínea (en forma de pares compuestos por el código numérico de la aerolínea y el nombre del destino). Estos ficheros emplearán el formato CSV para representar los datos, usando un punto y coma (;) como separador entre campos.

La segunda página del sistema permitirá introducir los vuelos. Se mostrará un listado con las diferentes aerolíneas registradas en el sistema. Justo debajo de cada nombre de aerolínea, en la misma página, se mostrará una lista de botones de radio indicando los diferentes destinos en los que opera la aerolínea, para que el usuario seleccione la ciudad de origen del vuelo. El usuario seleccionará uno y pulsará el botón enviar. A continuación, se le mostrará al usuario una lista displegable con aquellos destinos para los que no se tiene ningún vuelo que lo conecte desde la ciudad origen seleccionada en el paso anterior (estos datos los podremos obtener del fichero que se describe a continuación), así como un cuadro de texto de tipo *time* para especificar la duración del vuelo (en horas y minutos). En caso de que todos los destinos de la aerolínea hayan sido ya conectados con el destino seleccionado en el primer paso, se mostrará un mensaje de error indicando tal eventualidad. Recogidos los datos, se guardarán en un archivo CSV de vuelos, donde se indique el código numérico de la aerolínea, el nombre del destino (ciudad) origen del vuelo, el nombre del destino final, así como las horas y minutos de duración del vuelo.

Finalmente, una tercera página nos permitirá mostrar un informe de resumen de los destinos de una de las aerolíneas registradas en el sistema. Para ello, se ofrecerá una lista con botones de radio de cada una de las aerolíneas y, al enviar esta información, se ofrecerá el informe resumen indicando el nombre del destino, el número de vuelos (conexiones) que parten de él y el tiempo medio de vuelo hacia otros destinos. Esta tabla mostrará los destinos ordenados de forma descendente en función del número de conexiones.

Cree una página principal que permita el acceso a cada una de las páginas descritas anteriormente.

P2. (90 mins.) En este ejercicio creará un gestor de imágenes basadas en categorías. Para ello, vamos a utilizar el servicio que nos proporcionan en https://source.unsplash.com para obtener imágenes basadas en categorías (entre otros criterios de búsqueda). Puede obtener imágenes desde este servicio usando el envío de datos por el método *get*, de tal forma que se usa la URL https://source.unsplash.com/featured/?<categoria1>,<categoria2>,... obtendrá de vuelta una imagen cuyo contenido está relacionado con las categorías indicadas. Para hacer uso de este servicio, cree una función que reciba un vector de cadenas con las categorías y el nombre del fichero que contendrá la imagen recibida. Puede trasladar la imagen a un fichero local usando la función *copy*, donde el fichero de origen sea la URL descrita anteriormente (debe tener conexión directa a Internet para que esta opción funcione). Tenga en cuenta que si se desea soportar texto con espacios o caracteres especiales, deberá adjuntar a la URL la salida de la función *urlencode*.

Creada esa función, desarrolle una página, dispuesta en dos filas, compuesta por:

- Una fila superior con las imágenes obtenidas colocadas en una tabla HTML sin bordes. Debajo de cada imagen, a modo de leyenda, aparecerá la fechas de obtención y las categorías con las que está relacionada (separadas por coma).
- Una fila inferior con un formulario donde el usuario podrá obtener una nueva imagen. Para ello, podrá seleccionar las categorías deseadas mediante checkboxes a partir del siguiente conjunto disponible: *nature*, *lake*, *mountain*, *forest*, *dog*, *car,* girl, *agent*. También podrá subir un fichero de texto con las categorías deseadas (separadas por coma). En el caso de que se adjunte un fichero de texto, éste será compuesto por una única línea que contiene las categorías deseadas para una o varias imágenes a obtener. Si se especifican varias imágenes, se usará un delimitador punto y coma (;) para separar las categorías de una imagen de las de la siguiente.

Imponga una restricción sobre la longitud del texto con las categorías para que esté limitado a 200 caracteres, por lo que si la entrada es mediante el fichero, se ha de comprobar y mostrar un error en caso de no respetarse la limitación.

En función de lo descrito, deberá mantener un fichero de texto para saber qué imágenes tenemos y a qué categorías hacen referencia. Por ejemplo, si generamos dos imágenes desde un fichero llamado texto.txt cuyo contenido es *"nature,water;lake,mountain"* , deberemos tener un fichero con el siguiente contenido:

```
nature,water          2015-10-27_11-30-A.jpg
lake,mountain         2015-10-27_11-30-B.jpg
forest                2015-10-27_12-00.jpg
```

Observe los nombres dados a los archivos que contienen las imágenes. Éstos se componen de un sello de tiempo formateado según el ejemplo, junto con un contador alfabético (A,B,...) que permite numerar la imagen en el caso en el que en el fichero de texto suministrado se hayan especificado varias imágenes. Si es una sola imagen la especificada en el fichero o bien las categorías fueron indicadas en el formulario mediante los *checkboxes*, el nombre del fichero será sólo la marca de tiempo (tercer fichero del ejemplo).

P3. (90 mins.) Elabore un sistema de almacenamiento de artículos científicos. En la página inicial, se solicitará un nombre de usuario y una contraseña (cuyos caracteres permanecerán ocultos) de un usuario que ya exista en el sistema. Si no existe, se creará un nuevo usuario que recoja un nombre de usuario, una contraseña, su correo electrónico y la Universidad a la que pertenece, elegida de un campo *select* que muestre las 10 Universidades Públicas Andaluzas. La información obtenida se almacenará en un fichero de texto plano, que incluirá una fila por cada usuario, en la cual se incluirán los campos mencionados, separados por punto y coma. El sistema también comprobará la existencia previa o no del nombre de usuario, y si existiese, se mostraría un mensaje de error.

Al hacer la entrada correcta al sistema, se nos mostrará un formulario que permite subir archivos de tipo PDF, y una descripción del artículo que se dispone a subir. Existe, por una parte, una restricción de tamaño de fichero de 5Mb y por otra, una comprobación del tipo de fichero (no es válido un fichero de texto, o un documento DOCX u ODT). Si no se cumple alguna de las restricciones, se mostrará una advertencia de esta situación. En el mismo formulario, aparecerán los artículos subidos por ese usuario identificado, con lo que se podrá descargar cualquiera de esos artículos pulsando sobre ellos.

Cada vez que se suba un fichero el sistema comprobará que no existe un artículo exacto al almacenado en el sistema, para ello calcularemos el HASH MD5 del fichero haciendo uso de la función *md5_file*[1]. En el caso de coincidir el HASH MD5 del fichero subido con el de alguno de los artículos almacenados por el usuario, se procederá a informarle, no permitiendo la subida de dicho artículo al sistema.

1 http://php.net/manual/en/function.md5-file.php

La información de los ficheros almacenados en el sistema se mantendrá en un fichero CSV. El formato de este fichero será de tal forma que las columnas se corresponderán respectivamente con, el nombre del archivo, junto con la hora de subida al sistema , la descripción del artículo, el usuario propietario del mismo (nombre de usuario), la Universidad a la que pertenece el usuario y el HASH MD5 del fichero.

Todas las entradas del formulario de subida de archivos deberán ser validadas y corregidas.

Finalmente, la aplicación permitirá buscar artículos. Para ello, se ofrecerá un cuadro de búsqueda en el que se podrá escribir una palabra. El sistema, al recibirla, buscará aquellos artículos que contengan dicha palabra en su descripción y las mostrará de la forma en que habitual incluyendo en ese caso el nombre de usuario y la Universidad. También se podrá solicitar al buscador que muestre todos los artículos subidos por los usuarios de una misma Universidad.

Ampliación de Bibliografía

1. PHP 5 Power Programming. Andi Gutmans, Stig Bakken, Derick Rethans. Prentice Hall, 2004. Capítulos 5 y 9. http://www.informit.com/content/images/013147149X/downloads/013147149X_book.pdf
2. Beginning PHP5, Apache, and MySQL Web Development. Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec. Willey, 2005. Capítulo 8. http://site.ebrary.com/lib/bupo/docDetail.action?docID=10114243
3. PHP 6/MySQL Programming for the Absolute Beginner. Andrew B Harris. Course Technolgy, 2008. Capítulo6. http://site.ebrary.com/lib/bupo/docDetail.action?docID=10251031
4. PHP 5 for dummies. Janet Valade. Wiley Pub., 2004. Capítulos 11, 12 y 13. http://0-site.ebrary.com.athenea.upo.es/lib/bupo/Doc?id=10114230

Datos de la Práctica

Autor del documento: Carlos D. Barranco González (Diciembre 2007).

Revisiones:

1. Carlos D. Barranco González (Febrero 2009).
2. Carlos D. Barranco González (Noviembre 2009). Revisión de texto y enlaces.
3. Carlos D. Barranco González (Diciembre de 2010). Revisión del texto, actualización de enlaces a expresiones regulares PCRE y renovación de los ejercicios 1, 2 y 4 y problema 1.
4. Alejandro Gómez Morón (Noviembre de 2011). Renovación del ejercicio 3, ligero cambio en el ejercicio 5, adaptación del problema 1 para que tuviese concordancia con la APD 2 del curso 2011/2012, ligero cambio en el problema 2 y renovación del problema 3.
5. Carlos D. Barranco González (Noviembre 2011). Revisión y corrección del texto de la anteriores modificaciones.
6. Carlos D. Barranco González (Noviembre 2012): Adaptación del formato a Programación Avanzada. Corrección del enlace a la referencia 5 de la bibliografía básica. Adición de aclaraciones en el texto de la sección "Conceptos". Adición de aclaraciones al enunciado del Ej2. Modificación y corrección del enunciado del Ej3 , evitando que sea necesario hacer uso del material adicional incluido en la pasada edición. Corrección del enunciado del Ej4. Modificación de los enunciados de P2 y P3 para adaptarlos a los cambios hechos en el EPD2. Reemplazo de la referencia 3 de la bibliografía de ampliación dado que la anterior ya no está disponible en el catálogo de la biblioteca.
7. Miguel Ángel Montero (Noviembre 2013): Modificación del enunciado del ejercicio 4 y de los problemas P1, P2 y P3. Modificación de la temporización del ejercicio 3 y 4. Reemplazo de la referencia 2 de la bibliografía de ampliación puesto que la anterior ya no se encuentra disponible en el catálogo de la biblioteca.
8. Carlos D. Barranco (Noviembre 2013): Mejora de redacción de EJ4 y P3. Arreglos de formato en Ej2. Acortado enlace la referencia 3 de la ampliación de bibliografía.
9. Miguel Ángel Montero (Noviembre 2013): Mejora de redacción de EJ3 y EJ4.
10. Carlos D. Barranco González (Octubre 2014): Mejora de redacción de conceptos. Reorganización de ejercicios y problemas. Nuevas estimaciones de tiempos. Mejoras en la redacción de EJ3 y P2 (junto con sustitución del servicio de generación de códigos QR empleados). Actualización de la referencia de bibliografía básica 6.
11. Carlos D. Barranco González (Octubre 2015): Modificación de cabecera y objetivos. Renovación de Ej2 y, por extensión, Ej3, así como P2 y P3, junto con la eliminación de los antiguos problema 3 y 5. Redistribución de tiempos.
12. Jose A. .Gómez (Octubre 2016): Actualización de los Ejercicios 2 y 3. Modificación del Problema 3.
13. Carlos D. Barranco (Octubre 2016): Corrección de erratas en enunciados, así como su mejora en su redacción, de ejercicios y problemas.
14. Gualberto Asencio (Octubre 2017): Arreglado hipervínculo en la sección de ampliación de la bibliografía. Modificados ejercicio 1, problemas 1 y 2.
15. Carlos D. barranco (Octubre 2017): Mejora de formato en las referencias de la bibliografía básica. Correcciones de formato y erratas menores en los problemas 1 y 2.

Estimación temporal:

- Parte presencial: 120 minutos.
    - Explicación inicial: 5 minutos.
    - Experimentos: 45 minutos.
    - Ejercicios: 70 minutos.
- Parte no presencial: 270 minutos.
    - Lectura y estudio del guión y bibliografía básica: 0 minutos
    - Problemas: 270 minutos