Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

PRÁCTICAS DE ARQUITECTURAS DISTRIBUIDAS

Práctica 4: DESARROLLO DE APLICACIONES WEB MEDIANTE PHP, BASES DE DATOS Y XML

Profesores:

Javier Vales Alonso
Esteban Egea López

1. Objetivos

- Practicar el uso del lenguaje PHP para procesar información.
- Introducir el manejo de BBDD relaciones y diseño de BBDD.
- Practicar el uso de sentencias SQL sencillas
- Comprender la utilidad de las *cookies* en aplicaciones reales.
- Introducir el procesado de XML y su aplicación para facilitar la interoperabilidad de aplicaciones y sistemas.

2. Introducción

El objetivo de esta práctica es el desarrollo de una aplicación simplificada de comercio electrónico. Durante el desarrollo de esta práctica, aplicará conceptos estudiados en teoría y trabajará con algunas de las tecnologías expuestas. Usará PHP como lenguaje de programación de las aplicaciones de procesado de información.

Comenzará practicando conceptos básicos de PHP. A continuación desarrollará gradualmente la aplicación de comercio electrónico, que llamaremos "Mi Tienda en Internet (MTI)":

- Esta aplicación imita de manera simplificada el funcionamiento de una tienda online.
- MTI dispone de un catálogo de productos clasificados en dos categorías "Electrónica" y "Literatura". Cada producto del catálogo tiene asociado un precio, una descripción y un número de unidades disponibles.
- La web de MTI lista al usuario los productos por categorías. Una vez listado, el usuario puede seleccionar productos que se añaden a un "carrito de la compra".
- Cuando el usuario ha terminado de seleccionar productos, puede efectuar la compra, pinchando el botón comprar. En ese momento se generará un pedido.
- El catálogo de MTI es exportable e importable mediante ficheros de texto.

Esta aplicación la desarrollará en varios pasos, como se detalla en el siguiente apartado.

3. Desarrollo de la práctica

3.1 Introducción a PHP

Realice los siguientes ejercicios. Para el desarrollo de los ejercicios consulte las transparencias de PHP así como la documentación de PHP en www.php.net

1. Realice un "Hola Mundo" en PHP. El navegador debe recibir código HTML, no solo texto.
2. Utilice un formulario HTML para que el usuario introduzca su nombre y lo envíe al servidor y cree un script PHP que dé la bienvenida al usuario. Utilice el método GET primero y luego el método POST.
3. Cree un script en el que en un array se almacene un conjunto de usuarios y sus contraseñas. Muestre todos los usuarios y contraseñas almacenadas en forma de tabla HTML.
4. Utilice un formulario HTML para que un usuario introduzca su usuario y contraseña y compruebe mediante PHP si la contraseña de dicho usuario es correcta.
5. Muéstrele a un usuario un formulario con una lista de selección, botones de selección única y casillas de selección múltiple y cree un script PHP que muestre las selecciones del usuario en cada caso.

3.2 Página principal y "carrito de la compra"

Para el desarrollo de la aplicación, no utilizará la BBDD al principio.

Comenzará desarrollando la página principal MTI y el carrito de la compra que almacena la selección de productos del usuario. Para ello, cree los .php que considere necesarios. Estos archivos los irá refinando a medida que avance en la práctica para dotarlos de la funcionalidad necesaria. El objetivo es que implemente un "carrito de la compra" mediante *cookies*. Puede seguir las siguientes recomendaciones:

1. Comience creando una página de bienvenida en la que el usuario puede seleccionar mediante una lista de selección la categoría de productos que desea listar.
2. Cuando el usuario elige la categoría de productos que desea examinar, recibirá un listado de productos en forma de tabla con la información asociada a cada producto. Además, el HTML incluirá en la parte superior de la página una cesta de la compra, es decir, una lista de los productos que el usuario ha seleccionado. En la parte inferior de la página aparecerá hiperenlaces con las distintas categorías y el catálogo de cada categoría en forma de tabla.
3. Cada elemento del catálogo incluye una casilla de selección exclusiva ("radio"), que permite al usuario seleccionar un producto. Además se incluye un botón para enviar la selección. Cuando el usuario selecciona un producto se actualiza la cesta de la compra. Para implementar esta funcionalidad realice las siguientes acciones:
    a. Cree un archivo PHP que se encargará de realizar todo el procesado.
    b. Declare e implemente tres funciones:
        i. *mostrarCatalogo($cat)*. Muestra el catálogo correspondiente a la categoría que se le pasa como parámetro.
        ii. *actualizarCarrito($id)*. Actualiza el carrito de la compra con el producto que se le pasa como parámetro.
        iii. *mostrarCarrito()*. Muestra los elementos en la cesta de la compra.
    c. Adicionalmente puede implementar las funciones *imprimirCabecera()* e *imprimirCierre()* que imprimen el HTML de inicio y cierre de documento.

Como puede comprobar, esta implementación de un carrito de la compra está bastante limitada. Por ejemplo, ¿qué ocurre si un usuario selecciona dos productos iguales? ¿se le ocurre alguna forma de implementar esta posibilidad utilizando sólo *cookies*?
Además, el código PHP probablemente no quede muy "limpio". Estos inconvenientes se resuelven de manera sencilla utilizando BBDD como verá en el siguiente apartado.

3.2 Desarrollo con la Base de Datos

Lo más adecuado es gestionar el catálogo de productos desde una BBDD relacional. En este punto, creará las tablas necesarias en la BBDD y accederá a ellas mediante PHP. Para ello, disponemos en el servidor del laboratorio una BBDD MySQL (www.mysql.com). La gestión de MySQL se puede realizar de manera muy sencilla e intuitiva mediante la interfaz web que proporciona phpMyAdmin (www.phpmyadmin.net). Dispone de una cuenta en la BBDD, tal y como le indicará el profesor.

1. Diseñe la implementación del catálogo en la BBDD: tablas, campos de las tablas, tipos de datos del campo, atributos del campo (NULL, autoincremental, etc.). Una vez disponga del diseño, cree las tablas necesarias en la BBDD mediante phpMyAdmin (labit601.upct.es/phpmyadmin). Introduzca varios

productos de prueba en la BBDD. Además del catálogo, la BBDD incluirá una tabla con usuarios y otra con pedidos, que relaciona los usuarios con los pedidos que han realizado.

2. Modifique los PHP del apartado anterior para que el catálogo que se muestra se lea de la BBDD. Para ello necesitará lo siguiente:
    a. Examine las funciones de trabajo con MySQL que proporciona PHP. En particular, los ejemplos de conexión y realización de consultas. http://www.php.net/manual/en/ref.mysql.php.
    b. Decida mediante los ejemplos proporcionados en el siguiente tutorial, las sentencias SQL que necesitará para mostrar el catálogo: http://dev.mysql.com/doc/refman/5.0/es/tutorial.html
    c. Modifique los PHP anteriores para que se muestre el catálogo a partir de los datos recibidos de la BBDD. De nuevo, los productos pueden ser seleccionados para el carrito de la compra pinchando sobre ellos.

3. Cuando el usuario pinche el botón de comprar, se le mostrará un formulario para que introduzca su DNI, nombre y apellidos y un botón para que acepte la compra. Cuando el usuario piche este botón, se añadirá a la tabla de usuarios y se crearán las distintas entradas en la tabla de pedidos, que relacionan al usuario con el pedido que acaba de hacer.

3.3 Procesado XML

Para finalizar la aplicación desarrollará un *parser* PHP que le permitirá exportar/importar el catálogo a la BBDD en forma de documento XML.

En terminología XML, se denomina *parser* a un programa que, partiendo de un documento XML es capaz de analizarlo, con el fin de realizar cualquier procesado útil de la información. Existen dos tipos básicos de *parsers* XML:

1. Construcción del árbol: Estos *parsers* generan en memoria, a partir del documentoXML original, un árbol de nodos, donde cada nodo es un elemento del documento. Después, mediante funciones de búsqueda permiten al programador, acceder a cualquier nodo del árbol, y procesarlo.
2. Basados en eventos: Los *parsers* del tipo anterior resultan cómodos para el programador, pero consumen una gran cantidad de memoria. Si el XML a procesar es muy grande, es probable que no existan suficientes recursos para construir el árbol de nodos.

Para esta práctica utilizará un *parser* tipo árbol. En particular, utilizará la implementación PHP de la especificación DOM (http://www.php.net/manual/en/ref.dom.php). Esta especificación define una API estándar para el procesado de documentos XML.

1. Realice un *script* PHP que extrae el catálogo de la BBDD y lo muestra como documento XML. Especifique un DTD adecuado previamente.

ANEXO

Ejemplo de procesado XML con DOM y PHP

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');

$element = $dom->createElement('test', 'This is the root element!');

// We insert the new element as root (child of the document)
$dom->appendChild($element);

echo $dom->saveXML();
?>
```

Ejemplo de acceso estándar a BBDD con PHP

```
<?php

$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Not connected : ' . mysql_error());
}

// make foo the current db
$db_selected = mysql_select_db('foo', $link);
if (!$db_selected) {
    die ('Can\'t use foo : ' . mysql_error());
}

$query = sprintf("SELECT firstname, lastname, address, age FROM
friends WHERE firstname='%s' AND lastname='%s'",
    mysql_real_escape_string($firstname),
    mysql_real_escape_string($lastname));

// Perform Query
$result = mysql_query($query,$link);

// Check result
// This shows the actual query sent to MySQL, and the error. Useful
for debugging.
if (!$result) {
    $message  = 'Invalid query: ' . mysql_error() . "\n";
    $message .= 'Whole query: ' . $query;
    die($message);
}

// Use result
// Attempting to print $result won't allow access to information in
the resource
// One of the mysql result functions must be used
// See also mysql_result(), mysql_fetch_array(), mysql_fetch_row(),
etc.
```

```
while ($row = mysql_fetch_assoc($result)) {
    echo $row['firstname'];
    echo $row['lastname'];
    echo $row['address'];
    echo $row['age'];
}

// Free the resources associated with the result set
// This is done automatically at the end of the script
mysql_free_result($result);
mysql_close($link);
?>
```