UT6 - Utilización de técnicas de acceso a datos

Sumario

Curriculum

- Establecimiento de conexiones.
- Recuperación y edición de información.
- Utilización de conjuntos de resultados.
- Ejecución de sentencias SQL.
- Transacciones.
- Utilización de otros orígenes de datos.

Contenidos

En PHP podremos acceder a múltiples tipos de base de datos (MySQL, MariaDB, Oracle, PostgresSQL, etc). Cada base de datos tendrá su correspondiente controlador/conector que nos proporcionará las funciones para acceder a la misma.

Contenidos generales

- *Véase la presentación "ut6 - Acceso a BBDD - mysql"*

- *Véase el apartado "Bases de datos en PHP" del curso "Principios básicos para la programación en PHP" – Nota: Aquí se mezcla la presentación y el código, algo que deberemos evitar más adelante*
- *Véase los temas "Trabajando con Bases de Datos MySQL" y "Trabajando con MySQL desde PHP" del libro "Programador PHP Tomo 1 - Eugenia Bahit.pdf"*

Herramientas

Las herramientas siguientes nos permitirán interactuar con el sistema gestor de base de datos, con el objeto de crear esquemas, usuarios, tablas, realizar consultas, etc.

- *PhpMyAdmin*: Aplicación en PHP que permite realizar operaciones sobre bases de datos en MySQL.
  http://www.phpmyadmin.net/
- *MySQLWorkBrench*: Aplicación de MySQL que permite administrar una base de datos.
  https://www.mysql.com/products/workbench/
  A partir de la versión 8.0.19 de Workbench puede que requiera una conexión segura a través de SSL algo que no viene configurado por defecto en XAMPP, puede que resulte interesante instalar esta versión, la cual podremos obtener en
  https://downloads.mysql.com/archives/workbench/

Existen otros productos en el mercado que permiten realizar igualmente estas operaciones. Se han seleccionado los siguientes por ser el primero el que se incluye en la mayoría de los hosting de internet, y el segundo ser la aplicación diseñada por el propio fabricante que incorpora además opciones de modelado.

Por motivos didácticos, y para tener conocimiento de aplicaciones antiguas, aunque ya obsoletas se explicará el acceso a la base de datos utilizando la API MySQL original "*mysql_*" que actualmente está obsoleta y no se recomienda su utilización.

Fácilmente se puede utilizar la nueva extensión MySQL mejorada (desde versión MySQL 4.1) añadiendo el prefijo "mysqli_" a las mismas funciones que obsoletas.

La nueva extensión aporta mejoras que debemos también considerar.

Nota: Debéis observar que en las nuevas funciones siempre es preciso pasar como parámetro el enlace "$link" a la base de datos con la que estáis trabajando y que obtendréis en la conexión.

```
mixed mysqli_query(mysqli $link,string $query[,int
        $resultmode= MYSQLI_STORE_RESULT] )
```

Puede consultar la lista de funciones disponibles en:
http://www.w3schools.com/php/php_ref_mysqli.asp

Accediendo a MySQL (MaríaDB) de forma nativa

Véase PHP.NET: Extensión MySQL mejorada

- Interfaz dual: procedimental y orientada a objetos
- Conexiones

- Ejecutar sentencias
- Sentencias Preparadas

También hay ejemplos en W3Schools.com - PHP MySQL Database

Ejercicios propuestos

Bloque Consultas

1. Crea la base de datos “bdprovincias” en la que incluiréis los datos contenidos en el fichero “provincias.sql”.
2. Realiza una página en PHP que muestre las provincias que hay en España leyendo los datos de la base de datos.
3. Realiza una página en PHP que nos diga cuantas provincias tiene cada CCAA.
4. Realiza una página en PHP que muestre las provincias que tiene cada comunidad autónoma, con el siguiente formato:

   CCAA1 : prov1, prov2, …

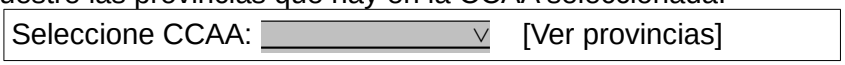
   CCAA2: …..

5. Realiza una página en PHP que muestre las provincias que tiene cada comunidad autónoma, en una tabla con el formato:

CCAA	Provincias
Andalucía	Almería
	Cadiz
	...
	Sevilla

> Cuando resolvemos problemas siempre es conveniente separar la lógica de negocio (obtener los datos de la base de datos, en este caso) de la presentación (mostrar la página web). Por este motivo es conveniente que por un lado obtengáis los datos y los almacenéis en un array, y por otro generéis el código HTML que muestre los datos.

6. Realiza una página que permita seleccionar la comunidad autónoma en un formulario y nos muestre las provincias que hay en la CCAA seleccionada.

   Después de enviar el formulario se mostrará el nombre de la comunidad autónoma seleccionada y las provincias que la conforman.

Bloque Modificaciones

> Habitualmente nos estamos conectando desde múltiples páginas a la misma base de datos en una aplicación, por lo que estaremos incluyendo redundantemente los datos de conexión en cada una de las páginas en las que nos conectamos. Por este motivo puede ser conveniente la creación de un fichero en php que incluya toda la información sobre la conexión
>
> **-- bbdd_conex.inc.php –**
> <?php

```
    $host='localhost';
    $user='root';
    $passwd='';
    $conex=mysql_connect($host, $user, $passwd) or die('Error BBDD');
-- otro_fichero.php –
<?php
    include ('bbdd_conex.inc.php');
    // resto del código
```

Más adelante se abordará este problema desde un enfoque de POO, lo que nos creará soluciones más elegantes.

7. Realiza una página que nos permita añadir nuevas provincias a nuestra base de datos. Las provincias creadas estarán ubicadas en una nueva comunidad autónoma que se llamará "Nuevas provincias".

   La entrada será filtrada, de forma que no se permitirá que introduzcan provincias repetidas o que se introduzcan cadenas en blanco. Igualmente el nombre de la provincia no deberá contener ningún dígito. Véase el artículo "Expresiones regulares en PHP" o el o el minimanual Explicaciones y ejemplos para el manejo de expresiones regulares.
   El programa tendrá un formulario similar al siguiente:

   Provincia: _____ [Añadir] Ver todas las provincias

   Después de cambiar, filtrando que los datos sean correctos, se seguirá preguntando para seguir cambiando el nombre.

   Donde desde el enlace "Ver todas las provincias" mostraremos todas las provincias y comunidades autónomas que tenemos almacenadas.

8. Realiza una página que nos permita modificar el nombre de las provincias.
   El programa tendrá un formulario similar al siguiente:

   Provincia: _____ Nuevo nombre:_____[Cambiar]
   Ver todas las provincias

   Donde desde el enlace "Ver todas las provincias" mostraremos todas las provincias.

Para la creación de los campos *select* resulta interesante crear una función en php que cree el campo en html utilizando los datos de un array. La función podría tener el siguiente formato :
*function creaSelect($nombreCampo, $valores /*array*/, $seleccionado='') {…}*

Luego la usaríais

```
<html>
…
<?php echo creaSelect('numeros', array('1'=>'uno', '2'=>'dos',...), $_POST['numeros']); ?>
```

9. Cuestión teórica: ¿Qué pasa si en el nuevo nombre de provincia incluís comillas simples o dobles en la cadena? Haz la prueba.
   Busca información sobre "inyección de código en SQL" (Wikipedia), para obtener una información más precisa sobre el problema.

Lee información sobre las funciones interesantes para el manejo de cadenas en SQL del que se incluye información mas adelante.

10. Realizar una página que nos permita borrar una provincia de la tabla. Cuando borramos es conveniente realizar una confirmación del borrado, por lo tanto el proceso se hará de la siguiente manera:

Formulario 1	Provincia: [▽] [Borrar] Ver todas las provincias
Formulario 2	¿Desea borrar la provincia XXXXXX? [Sí] [No]
Si pulsan [No] volvemos a Formulario 1, Sino eliminamos y vamos a Formulario 3	
Formulario 3	Se ha procedido a borrar la provincia XXXXXXX Borrar otra provincia - Ver todas las provincias

Vais a necesitar campos ocultos para solucionar el problema.

11. Ahora deseamos realizar el ejercicio anterior pero con otra presentación. El objetivo es poder borrar o modificar una provincia pero sin tener que seleccionarla en ningún campo del formulario. El procedimiento de trabajo será el siguiente:

    Se mostrarán todas las provincias y se dará la opción de borrarla o modificarla

LISTADO DE PROVINCIAS - Provincia 1 Modificar / Borrar - Provincia 2 Modificar / Borrar ...

    Para realizar esto deberéis utilizar parámetros de tipo GET en los que notificaréis la provincia con la que deseáis trabajar. En la operación de borrado se precisa confirmación igual que en el ejercicio anterior.

Seguridad en las consultas – Inyección de SQL

Cuando realizamos aplicaciones que trabajan con base de datos, debemos contemplar la posibilidad de que los datos que vengan del cliente no tengan el formato que esperamos. Debiendo ser muy cuidadosos sobre como trasladamos dichos datos a nuestra consulta SQL. No olvidéis que a fin de cuentas una consulta SQL no deja de ser una cadena de texto.

Véase:

- Inyección SQL - Wikipedia
- PHP.net Inyección de SQL
- Un ataque de inyección de SQL
- Inyecciones SQL – Otro ejemplo
- *Véase el ejemplo "bbdd_sql_inyeccion_de_codigo.php" e intente que se muestren todas las provincias incluyendo texto en el cuadro de búsqueda*

Las siguientes funciones resultan interesantes para evitar inyección de código

- *Addslashes*: Añade barras invertidas a una cadena
- *mysqli_escape_string*: Escapa una cadena para incluirla en una sentencia SQL

Por eficiencia y seguridad es mejor utilizar sentencias preparadas.

Sentencias preparadas

Las bases de datos MySQL soportan sentencias preparadas. Una sentencia preparada o una sentencia parametrizada se usa para ejecutar la misma sentencia repetidamente con gran eficiencia.

Flujo de trabajo básico

La ejecución de sentencias preparadas consiste en dos etapas: la preparación y la ejecución. En la etapa de preparación se envía una plantilla de sentencia al servidor de bases de datos. El servidor realiza una comprobación de sintaxis e inicializa los recursos internos del servidor para su uso posterior.

El servidor de MySQL soporta el uso de parámetros de sustitución posicionales anónimos con ?.

Léase los siguientes enlaces

- "Sentencias preparadas en PHP"
- PHP Prepared Statements
- PHP.NET Sentencias Preparadas

Véase

- Sentencias Preparadas en PHP
- Sentencias preparadas de MySQL en PHP (ejemplos orientados a objetos)
  Consultas preparadas y escapado
  http://programandolo.blogspot.com.es/2013/06/extension-mysqli-parte-3-consultas.html

Ejercicios propuestos – Bloque sentencias preparadas

12. Modifica varios de los ejercicios que has realizado anteriormente de forma que se utilicen consultas preparadas. Realiza un ejemplo de consulta, otro de inserción y otro de actualización.

Limitar los resultados de una consulta SQL

Véase el artículo: SQL-Limitando registros recuperados en una Consulta

Paginación de resultados de una base de datos.

La paginación de resultados nos permite mostrar grandes cantidades de datos en una aplicación web. El objetivo mostrar los datos que devuelve una operación, normalmente una consulta sobre una base de datos, de forma organizada, tratando en todo momento de evitar que se realicen cálculos o movimientos de datos innecesarios.

Se tendrán las siguientes variables:

- Nº de resultados por página: Puede ser un valor fijo en programación o configurable de diferentes formas.
- Nº de página a mostrar

Enlaces

- Véase el artículo: "Paginación de resultados" en el que se incluyen tres artículos sobre creación de páginación.

- Paginación de resultados con PHP y MySQL:
  En este tutorial **se calcula improductivamente el número de elementos** que hay que mostrar, deberíamos utilizar una consulta "*select count(*)...*" en lugar de traer todos los registros para saber cuantos son.
  http://www.desarrolloweb.com/articulos/1035.php

> No es conveniente utilizar la función *mysqli_num_rows(..)* o el atributo del mismo nombre si estámos utilizando la versión O.O. debido a que es improductiva y puede devolver resultados imprecisos.
> Para saber el número de resultados devueltos por una consulta deberiamos realizar un:
> `select count(*) where -condición-`

- Buscar en Google más artículos...

Cuando estamos paginando resultados al mostrar una tabla de una base de datos es muy poco productivo, casi podría considerarse un error, traer de la base de datos todos los registros para luego descartarlos

```
$ssql = "select * from pais " . $criterio;
$rs = mysqli_query($conn,$ssql);
$num_total_registros = mysql_num_rows($rs);
```

Luego ...

```
$ssql="select * from pais ".$criterio." limit ".$ini.",".TAM_PAG;
$rs = mysqli_query($conn, $ssql);
while ($fila = mysqli_fetch_object($rs)){
      echo $fila->nombre_pais . "<br>";
}
```

Cada vez que invocamos a `mysqli_query()` traemos todos los registros, pero la primera vez los ignoramos, pues solo nos importa el n.º de registros. Es mucho mejor, y menos costoso realizar

```
$ssql = "select count(*) as total from pais " . $criterio;
$rs = mysqli_query($conn,$ssql);
$reg= mysqli_fech_assoc($rs);
$num_total_registros = $reg['total'];
```

Ejercicios propuestos – Bloque paginación

13. Muestra la lista de países existentes en la base de datos países, de tal forma que solo se muestren 20 países de una vez. Crea en tu página un sistema de paginación que te permitirá avanzar o retroceder para mostrar los siguientes países.

Abstracción de la base de datos utilizando objetos

Cuando accedemos a la base de datos nos interesa abstraer y abordar con una metodología orientada a objetos. Esto puede ser realizado de forma simple, con una clase que nos abstraiga de un proveedor determinado.

Léanse los documentos ubicados en la carpeta *artículos*

- *BD-1-Crear una clase para conectar a base de datos con php (En el documento se ha modificado el original)*
- *BD-1-Crear una capa de conexión abstracta a base de datos con PHP*
- *BD-3-Creando_una_capa_de_abstracción_con_PHP_y_mysqli-Eugenia_Bahit*

Estos mecanismos los podréis encontrar en muchas aplicaciones antiguas. En las modernas la extensión PDO, de la cual hablaremos más adelante, y que es estándar en PHP consigue los mismos resultados.

Patrones de diseño

Patrón de diseño Singleton

Wikipedia: El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Este patrón será muy útil en nuestro propósito de crear una única conexión a la base de datos.

- El patrón singleton con PHP
- El patrón de diseño Singleton (implementado en PHP)

Ejercicios propuestos – Patrón singletón

14. Modifica varios de los ejercicios que has realizado anteriormente de forma que utilicen la nueva clase de acceso a datos. Esta clase estará definida en el fichero “db.php”. Se añadirá a la clase nueva funcionalidad si se precisa..

    En alguno de los artículos o ejemplos posteriores ya está codificada la clase “db”

Patrón de diseño DAO

Normalmente como hemos visto en el ejemplo anterior nos interesa desacoplar el acceso a la base de datos de la lógica de nuestro programa. De esta forma ante posibles cambios de sistema gestor de base de datos, de MySQL a PostgresSQL por ejemplo, tendremos localizado el código que hay que modificar, lo que facilitará la migración.

Otro de los patrones que nos pueden ayudar en esta labor es el patrón de diseño DAO (Data Access Object – Wikipedia)

Léase el artículo “Patrón de diseño DAO“ el cual contiene enlaces las siguientes páginas:

- Descripción conceptual:
  - DAO
  - ABSTRACCIÓN DE DATOS (I): EL PATRÓN DAO
- En PHP
  - DAO (Data Access Object) – PHP
  - Patrón DAO en PHP (Data Access Object in PHP): Observad que en PHP no es preciso crear la clase VO (Value Object) o TO (Transport object), pues el lenguaje crear dichas estructuras al vuelo.

Otro patrón de diseño para acceder a los datos es el Active Record del cual podréis ver una comparativa con el DAO en este artículo: Patrones de Diseño (Active Record vs DAO)

Videotutoriales:

- Implementación de Patrón de diseño DAO con PHP
- El patrón DAO (Data Access Object). Manteniendo la persistencia de datos | | UPV
-

PDO - Objetos de Datos de PHP

Dentro de PHP existen múltiples librerías que nos permiten abstraer la conexión a la base de datos, permitiendo que nuestra aplicación pueda cambiar de base de datos sin que haya que apenas realizar modificaciones en nuestro código.

Una de las formas es utilizando la librería PDO que viene incluida en PHP desde la versión 5.

Eche un vistazo al manual de referencia - Objetos de datos de PHP (PHP.net)- o véase el documento "PDO - Acceso a base de datos" que incluye varios artículos que tratan el acceso de es forma.

- Tutorial de PDO
- PHP Data Objects (PDO) – Ejemplo de conexión a varias bases de datos diferentes
- Consultas a bases de datos desde PHP con PDO (ejemplo simple)
- Acceso a bases de datos con PDO – Muy completo
- PDO PHP ejemplos con clases – Aquí podéis ver un ejemplo que utiliza clases para acceder a la BBDD

Manejo de errores - Mecanismo de excepciones

PDO utiliza las excepciones para indicar cualquier tipo de error. El procesamiento de las excepciones en PHP es similar a otro tipo de lenguajes como Java, C#, etc.

- Fundamentos de programación/Manejo de errores
- php.net – Excepciones
- MANEJO DE EXCEPCIONES EN PHP
- Excepciones en PHP

Transacciones

Cuando tenemos un conjunto de consultas de inserción o borrado, si no podemos realizar alguna de ellas, que las anteriores que sí se hayan podido realizar no tengan efecto o no se registren en la base de datos. En otras palabras, un conjunto de sentencias se comportan como una sola unidad, o se ejecutan todas o ninguna. Esto es más o menos la definición de transacción.

Este comportamiento sobre acciones en la base de datos lo podemos conseguir mediante el uso de transacciones, las cuales tendrán tresoperaciones básicas en cualquier SGBD:

- Comienzo de la transacción (Begin Trannsaction)
- Fin de la transacción correcto (Commit Trannsaction) Todas las operaciones quedan en firme.

- Fin de la transacción incorrecto (Rollback Trannsaction) Se deshacen todas las modificaciones que se hayan realizado.

MySQL y otras bases de datos nos permite realizar una gestión manual de transacciones.

- TRANSACCIONES MYSQL CON PHP
- Transacciones en MySQL en PHP
- Soporte de la API para transacciones MySQLi (PHP.net)
- Transacciones PDO (PHP.net)

Ultimo identificador insertado

Es una práctica recomendable crear un campo llamado “id” de tipo entero y con el modificador autoincremento para que sea la clave principal de nuestra tabla. Independientemente de que tengamos otros campos candidatos a ser clave principal.

El motivo que justifica esta decisión es

```
CREATE TABLE Persons (
      ID int NOT NULL AUTO_INCREMENT,
      LastName varchar(255) NOT NULL,
      FirstName varchar(255),
      Age int,
      PRIMARY KEY (ID)
);
```

El motivo que justifica esta decisión de diseño es que dicho campo, al no ser un atributo del dominio que estamos modelando nunca cambiará y nos permitirá identificar los registros de las tablas.

Pero, que pasa cuando insertamos un nuevo registros

```
INSERT INTO Persons (FirstName,LastName) VALUES ('Lars','Monsen');
```

¿Que valor toma el campo id?

En MySQL existe una sentencia ***LAST_INSERT_ID()*** que nos devuelve el último identificador insertado en la tabla relacionada. La librería MySQLI nos proporciona tanto una función como un atributo de la clase MySQLI que nos permiten obtener dicho identificador. La sintaxis es la siguiente:

Estilo por procedimientos:

```
mixto mysqli_insert_id (mysqli $identificador_de_conexion)
```

Ejemplo de uso:

```
$fk_pedidos = mysqli_insert_id($db);
```

Estilo orientado a objetos (atributo):

Ejemplo anterior:

```
$fk_pedidos = $db->insert_id;
```

Más información en el manual de PHP: http://docs.php.net/manual/es/mysqli.insert-id.php

Esta funcionalidad estará disponible también para otras bases de datos con una función similar. PDO (PDO::lastInsertId) incluye también esta funcionalidad

En las tablas que definamos en nuestros problemas/trabajos es conveniente poner siempre como clave principal un identificador (id) de tipo autonumérico.

Aunque conceptualmente fuese aconsejable utilizar otro campo clave, según la teoria del diseño de BBDD, por cuestiones de simplicidad en la creación de la aplicación es preferible usar como clave principal un campo que no tenga ningún significado en el dominio del problema.

Las claves principales “conceptuales” les asociaremos un indice único para garantizar su unicidad.