Adriania

Donde los Adrianos se reúnen

Distintos modos de hacer queries SQL desde PHP

17 February, 2011 PHP, Web Dev databases, PHP

Este es un tema que montones de programadores novatos no conocen, o conocen de a poco y cada vez con menos entusiasmo. El hecho es que hay al menos una media docena de modos distintos de solicitar información desde una base de datos, y todos nos preguntamos -o deberíamos preguntarnos- qué es lo mejor que puede hacerse. Veamos las alternativas, pero antes, el mensaje esencial: **no deberían pasar nunca a la DB datos provenientes del usuario que no estén validados y sanitizados en el server**. **Validar solamente con javascript no sirve** (Mensaje extra: investiguen sobre el tema de Cross Site Request Forgery)

Habiendo pasado un tiempo desde este post, tengo una actualización importante que hacer. Actualmente, usar PDO correctamente (o algún framework/ORM sobre PDO) es el método standard y recomendado de conectarse a DB usando PHP. Aprendan y usen. Y la verdad, no hay casi ninguna buena razón para seguir usando en un nuevo desarrollo las funciones "directas" de PHP, como mysql_query. Nos complican la vida en un problema que ya está resuelto.

Usar las funciones "directas" de PHP

`mysql_query()`, `pg_query()`, etc. Esta es la opción más sencilla y más peligrosa: hay que recordar hacer un escaping correcto de los parámetros pasados a la query para evitar el problema de Bobby Tables, por ejemplo usando antes `mysql_real_escape_string()` o `pg_escape_string()`. Es recomendable también usar la función `sprintf()` y formatear los datos por tipo, o filtrar los resultados de acuerdo a sus tipos: si debo ingresar un número en el campo edad, es buena idea por ejemplo filtrarlo con `is_numeric()`. Un ejemplo:

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

Usar queries parametrizadas

PHP ofrece algunas funciones que nos permiten diseñar por un lado una query, y por el otro lado sus parámetros -es decir los valores variables a introducir en la query-, dejando el sistema mismo haga el escap-

ing de los parámetros. Las alternativas son `pg_query_params()` y usar la extensión orientada a objetos mysqli en lugar de las funciones *mysql_** a secas. Un ejemplo:

```
$result = pg_query_params($dbconn, 'SELECT * FROM shops WHERE name
= $1', array("Joe's Widgets"));
```

Usar PDO

PDO significa PHP Data Objects, es una librería orientada a objetos que ofrece una interface liviana y eficiente de consulta a bases de datos. El único inconveniente es que no está disponible por defecto en versiones antiguas de PHP. Tal vez sea una buena oportunidad de actualizar el sistema… Un ejemplo es:

```
$pdo_obj = new PDO( 'mysql:server=localhost; dbname=mydatabase',
$dbusername, $dbpassword );
$sql = 'SELECT column FROM table WHERE condition=:condition';
$params = array( ':condition' => 1 );
$statement = $pdo_obj->prepare( $sql, array( PDO::ATTR_CURSOR =>
PDO::CURSOR_FWDONLY ) );
```

Con lo cual están conectándose a db, creando la query con los parámetros aparte (en este caso, *:condition* en la query señala el punto en que se sustituirá el parámetro que se encuentra con ese nombre en *$params*, y por último, preparar la query, es decir, hacer la sustitución y crear el objeto que se pasará a la DB. Otro ejemplo más simple, de inserción de datos:

```
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:
name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

donde se usan los métodos `bindParam()` para lograr el mismo propósito: reemplazar en la query preparada los valores variables. Pueden ver más detalles en la página del manual de PDO.

Usar el sistema de querying de su framework

Casi todos los frameworks, y ciertamente todos los respetables (CakePHP, Lithium, Codeigniter, Symfony, sin menospreciar a otros que olvido o desconozco) usan sistemas más o menos complejos para abstraer las consultas a bases de datos. Personalmente, creo que a veces son útiles y otras, nocivos. Por un lado nos evitan bastante todo el tema de hacer escaping y ordenan los resultados en arrays de un modo muy práctico. Por otro lado, me ha pasado que la query que yo quería hacer simplemente no podía hacerse usando esa ab-

stracción, o que me forzaban a implementar código en la base de datos. Por ejemplo, los frameworks que conozco recomiendan calurosamente que cada tabla tenga un campo ID numérico autoincrementante como primary key. *Y si yo ya tengo una primary key (compuesta o no) que no es numérica?* Hay que decir que este no es un problema exclusivo de los frameworks en PHP, eso sí.

Usar librerías de abstracción de queries

Un ejemplo de esto es ADOdb, que nos ofrece también una librería que nos permite abstraer las queries incluso hasta el nivel de mapearlas a objetos PHP. Otro ejemplo importante es Doctrine. Con estas librerías lo que debemos hacer será primero un mapeo de las tablas a clases PHP (esto puede ser más o menos automágico), y luego una serie de métodos proporcionados por la librería nos permiten extraer la información básica o crear queries más complejas. Esto suena demasiado etéreo, ya sé, así que les dejo el enlace a varios ejemplos. En resumen:

- Creamos una tabla.
- Creamos una clase PHP que extiende un objeto de la librería (usualmente, la clase debe tener un nombre relacionado con la tabla, por ejemplo la tabla *personas* tendrá un objeto relacionado llamado *class persona.*
- Automáticamente, la nueva clase tiene atributos correspondientes a los campos de la tabla, y métodos que nos permiten automáticamente hacer queries simples.

Un ejemplo usando MDB2, una librería disponible mediante PEAR:

```
include('MDB2.php');

$db =& MDB2::factory('mysql://username:password@host/database');
// cómo devolverá los datos: array asociativo en este caso
$db->setFetchMode(MDB2_FETCHMODE_ASSOC);

// query de datos
$query = 'SELECT id,label FROM myTable';
$result = $db->queryAll($query);

// insertar datos
// Este es el prepare statement, noten los ? en la sección de
VALUES
$statement = $db->prepare('INSERT INTO mytable(id,label)
VALUES(?,?)');
// Seteamos las variables que vamos a pasar como VALUES:
$sqlData = array($id, $label);
// Ahora pasamos los datos y ejecutamos la query
$statement->execute($sqlData);
```

```
// Limpiamos todo.
$statement->free();
$db->disconnect();
```

Pros y contras de cada una

- Cuando estamos haciendo un script corto, rápidamente, a las dos de la mañana, usar las funciones *mysql_** es usualmente el método más rápido, y podemos caer en la tentación de decir "*ma sí, total quién va a tocar esto?*", especialmente luego de la tercera taza de café / vaso de Mountain Dew (TM). Este espíritu de cowboy coder es el que tantos problemas y mala fama le causan a los demás programadores que usan PHP. Obviamente, yo también he sucumbido a este pecado. Es más, por mucho tiempo, ni supe que habían alternativas. *Sanitizar las queries? Queloqué?* Recuerden: **una query que use parámetros que ingresa el usuario sin estar sanitizada es una invitación a que les abran el sitio en dos como un pescado**.
- Para usar las extensiones, debemos tenerlas instaladas y en el caso de PDO, debemos manejar un mínimo de programación orientada a objetos. Casi nada, pero lo suficiente para entender cómo crear y configurar un objeto PDO. *$pdo_obj =new PDO();? $pdo_obj->prepare()? Queloqué?*
- A veces, dependiendo del host de nuestro sitio las extensiones no están, o la versión de PHP directamente no las tiene. En cualquier caso, un pedido amable debería solucionarles las cosas. Por cierto, en pleno 2011, un host con PHP en versión menor a 5.1 por otra razón que no sea mantener aplicaciones legacy particulares realmente es incomprensible.
- La ventaja de usar librerías de abstracción es que nos permiten abstraernos de la base de datos usada. El ejemplo clásico es "Imaginate que en dos años tenés que cambiar de DB: tendrías que cambiar todas las queries". Ese ejemplo siempre me suena a mentira: si creamos una app compleja, de todos modos cambiarle la DB debajo de los pies no va a ser sencillo. Lo que sí es cierto es que, si ustedes planean desde un primer momento vender su aplicación a clientes (hosteándola en los servers de los clientes) van a tener que pensar desde el primer momento en soportar más de una base de datos, y una librería de abstracción les permite ahorrarse bastante trabajo en este caso.

Extras

Para completar la docena de panadero, les recuerdo que, si no tienen o no necesitan un motor de bases de datos relacional, pueden encontrar una alternativa simple en sqlite, una librería robusta y muy usada que nos ofrece una base de datos SQL transaccional, autocontenida, que no necesita de server ni de configuración, para propósitos relativamente livianos. Pruébenla. Como mínimo les puede servir para hacer prototipos rápidos.

5 thoughts on "Distintos modos de hacer queries SQL desde PHP"

Pingback: Distintos modos de hacer queries SQL desde PHP « DbRunas – Noticias y Recursos sobre Bases de Datos

Pingback: Lo mejor de mi RSS del 14 al 20 de febrero | Linux Hispano

Pingback: Lo mejor de mi RSS del 14 al 20 de febrero | Superlinux

20 February, 2013 at 15:04

Julio Ruiz

hola… antes que nada… no se si Adriana es tu nombre o es algun Acronimo.. pero me llamo la atencion por que pones “Donde los Adrianos se reunen…” no se que quiere decir… pero mi Hija de 7 años se llama Adriana y por eso me llamo la atencion tu slogan…..

bueno ya seguido…. gracias por compartir tu conocimiento yo soy un programador ya de algunos 10 años pero por motivos de trabajo me dedique toda la vida a programar en visua fox…. ya hora que ya no estoy en la empesa y trabajo por mi cuenta… quiero entrar al mundo del desarrollo web… he comenzado con un sistema pequeño para llevar el control de pacientes de un cliente que es medico… y compre un pequeño admin template con html5, javascript bueno… gran error .. querer entrar de primas a primeras con varios lenguajes que no conozco para nada…. en los cuales se desarrolla bien diferente a visual foxpro…. asi que tengo el problema que quiero hacer muchas cosas que he estado acostumbrado en visual fox que en programa web creo que no aplican.
para no quitarme mas tu tiempo… mi pregunta es la sliguiente…. leyento este blog me abrio los ojos a muchas cosas nuevas que no tenia ni la mas minima idea….
y quisiera saber basados en la experiencia de los mas conocedores cual seria el mejor metodo para poder manipular los datos.

yo he optado por tener una pagina con php para mostrar la informacion final del cliente, jquery para interactuar con los eventos del lado del cliente y este asu vez se encarga de hacer llamadas por post a otras paginas en php que se encargan de ejecutar un procedimiento en el servidor que me retorna los datos y posterior mento los convierto a json para que el mismo jquery se encarge de refrescar la data del lado del cliente.

no se si esta es una forma segura y corecta para trabajar los datos… y si el usar

procedmientos del lado del servidor para recuperar cualquier informacion es correcto o lo mejor es dejar las consultas en el codigo php. espero tus comentarios.

y gracias por tu tiempo…

23 February, 2013 at 22:32

★Adriano

Hola! gracias por el comentario, y te contesto por orden
1 – El sitio se llama Adriania, es simplemente porque yo me llamo Adriano, nada más. El slogan es 100% fantástico.
2 – Sobre la seguridad y corrección de manipular datos así o de otro modo, en realidad la respuesta es más compleja:
– Lo que vos planteás, PHP del lado server y jquery para ordenar y mostrar datos, es razonable. El punto más bien es qué hacés con PHP, si lo usás de modo ordenado y limpio, teniendo cuidado de la seguridad, o no. En realidad, para aprender, te recomiendo que leas algún tutorial de frameworks PHP actuales, como por ejemplo codeigniter, o symfony. Más allá del framework, te van a mostrar un cierto orden de trabajo, dónde va cada parte del código, etc.
Saludos!

Comments are closed.