Introducción a PostgreSQL con PHP

Recientemente he tenido que utilizar PostgreSQL para realizar algunos proyectos en donde laboro, principalmente por las características que ofrece PostgreSQL sobre MySQL, así que me dí a la tarea de buscar información sobre PostgreSQL y PHP para empezar a desarrollar aplicaciones web utilizando ambas tecnologías. Este pequeño tutorial te enseñara como manipular una base de datos en PostgreSQL por medio de PHP a través de ejemplos muy sencillos.

Prerrequisitos

Necesitarás lo siguiente para seguir este tutorial:

- El servidor PostgreSQL instalado
- PHP compilado con soporte para PostgreSQL
- Una base de datos y un usuario para dicha base de datos

Eso es todo, si necesitas ayuda para instalar PostgreSQL, lee la documentación de **PostgreSQL.com**

Conexión a PostgreSQL desde PHP

Para empezar a usar PostgreSQL desde PHP, primero deberás conectarte a él. Esto es realizado con la función ***pg_connect()***. Esta función es muy directa y solo espera un argumento, la cadena de conexión. La cadena de conexión contiene toda la información necesaria para conectarse a la base de datos. Los argumentos válidos para la cadena de conexión son el equipo, puerto, tty, opciones, nombre de la base de datos, usuario y contraseña. Normalmente te vas a conectar a tu base de datos de la siguiente forma:

```
/* dbname es el nombre de la base de datos a la que te conectas
* user es el usuario de PostgreSQL que usas para conectarte
* password es la contraseña del usuario con el que te conectas
*/
pg_connect("dbname=base_de_datos user=usuario password=password") or die("No se pudo realizar la conexi&oacute;n: ".pg_last_error());
// pg_last_error() regresa el último error que haya ocurrido, de esta manera sabrás que ocurrió
```

Creación de una tabla con *pg_query*

Ahora creemos nuestra primer tabla. Haremos un script para ello, mostrando el uso de ***pg_query()***. Crearemos una tabla llamada "Contactos" con los campos 'nombre','apellidos' y 'correo'. Para hacerlo, usa la siguiente consulta:

```
/* Estamos usando la consulta
*    CREATE TABLE contactos
*    ( nombre varchar(50),
*    apellidos varchar(50),
*    email varchar(50)
*    )
*/
pg_connect("dbname=base_de_datos user=usuario password=password") or die("No se pudo realizar la conexi&oacute;n ".pg_last_error()); // Conexión a la base de datos
/* Uso de la consulta */
$query = "CREATE TABLE contactos
(
nombre varchar(50),
apellidos varchar(50),
email varchar(50)
)";
$query = pg_query($query); // Ejecuta la consulta
if($query)
  echo "Tabla Creada"; // Verifica si la consulta fue exitosa
else{
  echo "Ocurri&oacute; un error! ".pg_last_error();
}
```

Si ocurrió algún error al usar este script, entonces verifica si proporcionaste el usuario y contraseña correctos, y verifica que PostgreSQL este corriendo. También se cuidadoso, ejecuta este script una sola vez y después borralo. De otro modo obtendrás errores diciendo que ya existe una tabla llamada "contactos".

Nota: También puedes usar una herramienta de administración para ejecutar la consulta anterior. Para ello existen **PHPPgAdmin** y **WebMin**

Inserción de registros en tu base de datos

Ahora que ya tienes una tabla, es tiempo insertar algunos registros en ella, tu base de datos debe ser parecida a esta:

Database name: Contactos
nombre | apellidos | email

Es hora de usar la función pg_query de nuevo. En esta ocasión guardaremos información dentro de nuestra base de datos. Usaremos el comando SQL 'INSERT' para hacerlo, su sintaxis es la siguiente:
INSERT INTO nombre_tabla (columna1, columna2,...) VALUES (valor1, valor2,....)

nombre_tabla es el nombre de la tabla, en los paréntesis puedes especificar en que columnas quieres insertar datos. En los valores, pondrás lo que quieres guardar en la base de datos. de este modo INSERT INTO contactos VALUES ('Juan','Perez Perez','juanperez@dominio.com') insertaría Juan como nombre, Perez como apellidos, y juanperez@dominio.com como dirección de email. Ahora escribamos un script para hacer esto.

```
pg_connect("dbname=base_de_datos user=usuario password=username") or die("No se realiz&oacute la conexi&oacute;n".pg_last_error());
$query = "INSERT INTO contactos VALUES('Juan','Perez Perez','juanperez@dominio.com')";
$query = pg_query($query);
if($query)
  echo "Inserción completa!";
else{
  echo "Ocurri&oacute; un error! ".pg_last_error();
}
```

Esto debió haber hecho lo que queríamos, si fallo verfica tu consulta e intenta de nuevo. Si todo marchó bien, insertemos más registros. De hecho, ¿por que no hacemos un formulario para insertarlos?. Para eso debemos empezar con una formulario muy básico.

```
<form method="POST">
Nombre: <input type="text" name="nombre"><br />
Apellidos: <input type="text" name="apellidos"><br />
Email: <input type="text" name="email"><br />
<input type="submit">
</form>
```

Ahora debemos agregar los valores reales a nuestro formulario. Primero debemos verificar que todos los campos hayan sido llenados, y entonces insertarlos en la base de datos:

```
<?php
if($_REQUEST['nombre'] && $_REQUEST['apellidos'] && $_REQUEST['email']) // Verifica si todos los campos fueron llenados
{
  pg_connect("dbname=nombre_base_de_datos user=usuario password=password") or die("No se pudo realizar la conexi&oacute;n"); // Conexión a la base de datos
  $query = sprintf("INSERT INTO contactos VALUES('%s','%s','%s')",$_REQUEST['nombre'],$_REQUEST['apellidos'],$_REQUEST['email']); // Realiza la consulta SQL
/* Breve lección de Sprintf:
* Sprintf tomará una cadena y le dará forma. En este caso uso %s lo que significa que el siguiente parametro será una cadena,
*/
  $query = pg_query($query);
  if($query)
    echo "El registro ha sido agregado";
  else
```

```
   echo "Ocurri&oacute;o un error ! ".pg_last_error();
}
else{ // Si no tenemos todos los campos, mostramos el formulario
?>
<form METHOD="POST" ACTION="<?php echo $_SERVER['PHP_SELF']; ?>">
Nombre: <input type="text" name="nombre"><br />
Apellidos: <input type="text" name="apellidos"><br />
Email: <input type="text" name="email"><br />
<input type="submit">
</form>
<?php
   }
?>
```

Eso es todo. Ya tenemos un formulario para insertar registros dentro de nuestra base de datos, si lees los comentarios deberás entender completamente que hace cada parte del script.

Visualización de los registros en la base de datos

Ahora que ya podemos agregar registros a nuestra base de datos, necesitaremos ver todos los registros en la misma. Eso podemos hacerlo usando el comando SELECT de SQL. Ten en cuenta que aún seguimos usando la función pg_query(). Si quisieramos mostrar todos los campos de la base de datos, podríamos hacer un script que seleccionara todos (* en SQL) los registros.

```
pg_connect("dbname=base_de_datos user=usuario password=password") or die("No se
pudo realizar la conexión"); // Conexión a la base de datos
$query = "SELECT * FROM contactos";
$query = pg_query($query);
while($row = pg_fetch_array($query,NULL,PGSQL_ASSOC))
{
   // print_r($row);
   // Descomenta la línea anterior para ver el arreglo completo
   echo "Nombre: ".$row['nombre']."<br />";
   echo "Apellidos: ".$row['apellidos']."<br />";
   echo "E-Mail: ".$row['email']."<br /><br />";
}
```

No te preocupes si no entiendes cada parte del script. Tan solo use la función pg_fetch_array() para traer los resultados de la consulta dentro de un arreglo asociativo. Las llaves del arreglo son nombradas después de los nombres de columna de la tabla. De este modo $row['nombre'] contendrá lo que haya en la columna "nombre" de nuestra tabla, simple ¿no?. La variable $query es la consulta que quieres traer. NULL es el número de fila, cuando especificas NULL, la función simplemente se saltará ese parámetro. El último parametro, el que dice PGSQL_ASSOC, elige que tipo de arreglo regresará. PGSQL_ASSOC tendrá arreglos con los nombres de columna como llaves. PGSQL_NUM regresará un arreglo número, y PGSQL_BOTH regresará ambos. Como nota final, puedes reemplazar cada uno de los echo's por un print_r() para ver el arreglo completo.

Actualización de registros

Lo único que nos falta para que sepas lo básico, es el comando update, que es bastante sencillo. Su sintaxis es: UPDATE nombre_tabla SET nombre_columns = nuevo_valor WHERE nombre_columna = algun_valor. Así "UPDATE contactos SET nombre = 'Juanito' WHERE apellidos = 'Perez Perez'" pondría el valor 'Juanito' en los registros donde 'Perez Perez' sean los apellidos.

Conclusión

Ok, si me seguiste hasta aquí, probablemente estes pensando que PostgreSQL no es muy diferente de MySQL, es verdad. Como lo dije al principio de este tutorial, este es un tutorial BÁSICO. Si llegaste hasta aquí sin ningún problema, te recomiendo que leas la documentación de PostgreSQL y veas algunas de sus características. Hay un mundo entero más allá de MySQL.