



CALLING PROCEDURES FROM PHP

In the first tutorial we looked at calling functions from PHP, if you have previous experience with using MySQL with PHP calling functions may have been familiar as in essence all we did was call a select statement which contained the procedure. If we wish to use procedures we can't use them within a select statement so we therefore have to call them from PHP differently.

There are two methods of retrieving data from procedures, calling a select statement from within the procedural code to return a result set or passing values in and out via parameters. In the first example we will look at using results sets as this will be similar in style to calling functions, then later we will look at passing parameters in and out of procedures using PHP.

RETURNING RESULT SETS

It may be a side effect or by design but MySQL stored procedures return result sets in a very different way to most other procedural languages. Simply by using a select statement in the procedure we can return result sets. For example the following procedure lists the name and id of all of the employees in the emps table.

```
drop procedure if exists select_emps//

create procedure select_emps()
begin
    select emp_id, emp_name from emps;
end
//
```

 [php_procedure1.myp](#)

So to get the results all we need to do is call the procedure.

```
call select_emps() //
+-----+-----+
| emp_id | emp_name |
+-----+-----+
|      1 | Roger    |
|      2 | John     |
|      3 | Alan     |
+-----+-----+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

Rather than this behaviour being a problem it makes calling the stored procedures from PHP similar to using tables.

We will assume that you have a working web server, PHP and MySQL installation, if you haven't then review the calling functions pages to find out what you need and how to verify that it is working correctly.

In this first example we will be calling the procedure above so if you haven't loaded this yet do this now. Next we need to create the PHP program, we will go through the code and then combine each stage into a PHP file which you can all at the end.

The first thing to do is create the connection to the MySQL server, this is done like so.

-- PHP --

```
<?php
$link = mysqli_connect("localhost","root","xxxxxx");

if (mysqli_connect_errno()) {
    echo "error";
    exit();
}
```

Make sure you change the parameters to the appropriate values for your setup. Next we need to select which database we will be using, in our case this is **pers**

-- PHP --

```
mysqli_select_db ($link,"pers");
```

Now that we have a connection and have selected the database we can call the procedure. This is done using the same method as calling a select statement but rather than using a select statement we use `call` as we would if calling the procedure from the MySQL command line. We need to specify a link to the connect and this is done using `$link` as the first parameter. In the actual page this will be enclosed in an IF statement to catch any errors but for now let's just look at the command.

-- PHP --

```
$result = mysqli_query($link,"call select_emps()")
```

The result of the call to `mysqli_query` is placed in an object called `$result`. We can then use the `mysqli_fetch_array` to return the results to an object called `$row` which will hold the results. This is done like so.

-- PHP --

```
$row = mysqli_fetch_array($result, MYSQLI_NUM);
```

In this example we are using **MYSQLI_NUM** this will return the results to be referenced by number, but we could use **MYSQLI_NAME** to reference by name or **MYSQL_BOTH** so that we can reference them by both name and number.

As we will be returning a number of rows from the procedure call we will want to loop through the result set. To do this we can enclose the `mysqli_fetch_array` in a while loop like so.

-- PHP --

```
while($row = mysqli_fetch_array($result, MYSQLI_NUM));
```

We can now display the output of the procedure call. To do this we use echo like so.

-- PHP --

```
echo "Emp ID : ".$row[0]." Name : ".$row[1]."<br> ";
```

In our example we are adding in some additional text, the important parts are the two `$row[]` statements. The first `$row[0]` displays the first value in the row that was returned by the stored procedure, in this case the employees ID number, `$row[1]` is the second, the employee's name.

The final command is a call to `mysqli_free_result` this clears the memory associated with the result set. All we need to do is pass in the result we want to clear from memory in our case `$result`.

-- PHP --

```
mysqli_free_result($result);
```

We can now combine all of the above sections to produce our PHP page.

```
<?php

$link = mysqli_connect("localhost","root","*****");

if (mysqli_connect_errno()) {
    echo "connection error";
    exit();
}

mysqli_select_db ($link,"pers");

if ($result = mysqli_query($link,"call select_emps()")) {

    while ($row = mysqli_fetch_array($result,MYSQLI_NUM)) {
        echo "Emp ID : ".$row[0]." Name : ".$row[1]."<br> ";
    }

    mysqli_free_result($result);
} else { echo "problem :( "; }

?>
```

 [php_selectemps.phx](#)

If you load the PHP file into your browser you should now see a list of employees. If you see the words "connection error" then you need to review the connection statement, or if you see the words "problem :(" then there was a problem calling the procedure.

It should be easy to covert this simple page to call other procedures you may have.

USING PARAMETERS

The second method of calling stored procedures is to pass in and return out parameters. This means we can produce procedures which don't return a result set but just a single value rather like a function, or infact any number of values.

Lets first look at how return a value from a procedure. We need to create a procedure which returns a value so lets do that now

```
drop procedure if exists php_helloworld //
```

```

create procedure php_helloworld(OUT p_out_param VARCHAR(30))
begin

    set p_out_param = 'Hello World';

end
//

call php_helloworld(@out_val) //
Query OK, 0 rows affected (0.00 sec)

select @out_val //
+-----+
| @out_val |
+-----+
| Hello World |
+-----+
1 row in set (0.00 sec)

```

php_procedure2.myp

We can now create a PHP page to call the procedure. We will be using the same connection and database selection methods as before so we can just take that straight from our previous PHP page.

-- PHP --

```

$link = mysqli_connect("localhost","root","*****");

if (mysqli_connect_errno()) {
    echo "connection error";
    exit();
}

mysqli_select_db ($link,"pers");

```

We can now add the code to call the procedure, the important part in this is how we deal with the parameter that is being returned. In this example we will be returning the value into a MySQL user variable, a user variable is a method of storing single values in MySQL for the duration of a session. A user variable is created using the following syntax.

SET @name = 'Roger';

We can now use the variable, in this case @name, in MySQL SQL statements or pass into procedures. For example we can simply select it to show it's value like so.

```

set @name = 'Roger' //
Query OK, 0 rows affected (0.00 sec)

select @name //
+-----+
| @name |
+-----+
| Roger |
+-----+
1 row in set (0.00 sec)

select emp_id from emps where emp_name = @name //
+-----+
| emp_id |
+-----+
|      1 |
+-----+
1 row in set (0.00 sec)

```

As user variables are stored in MySQL the method we will use in PHP is simply to call the procedure which

will populate the user variable and then call a further select statement to retrieve the value. In the above example we created the user variable using the **SET** command but we don't need to do this when calling the procedure as it will be automatically created and populated when the procedure is called.

The code to call the procedure is as follows.

-- PHP --

```
if ($result = mysqli_query($link,"call php_helloworld(@out_param)")) {  
} else { echo "problem :( outer"; }
```

This will call the procedure and assign the result to the @out_param user variable. You may have noticed the addition of **outer** to the else statment, this is because we will be calling a further SQL statement inside of this code so the outer will be useful when debugging.

The procedure has been called and @out_param holds the value returned we now need to retrieve that value to the PHP page, as shown above this is simply done by calling a select statement.

-- PHP --

```
if ($result = mysqli_query($link,"select @out_param")) {  
    $row = mysqli_fetch_array($result, MYSQLI_NUM);  
    printf($row[0]);  
    mysqli_free_result($result);  
} else { echo "problem :( inner "; }
```

This time the call to mysqli_query contains a simple single row, single column result set which we can fetch and use to display the result. So if we add all of the parts together we can display the php page.

-- PHP --

```
<?php  
  
$link = mysqli_connect("localhost","root","trustno1");  
  
if (mysqli_connect_errno()) {  
    echo "connection error";  
    exit();  
}  
  
mysqli_select_db ($link,"pers");  
  
if ($result = mysqli_query($link,"call php_helloworld(@out_param)")) {  
    if ($result = mysqli_query($link,"select @out_param")) {  
        $row = mysqli_fetch_array($result, MYSQLI_NUM);  
        printf($row[0]);  
        mysqli_free_result($result);  
    } else { echo "problem :( inner "; }  
} else { echo "problem :( outer"; }  
  
?>
```

 [php_proc_helloworld.phx](#)

IN PARAMETERS

There are two options when looking at when passing parameters into a procedure. We can create a user variable and then use that variable in the call or we can simply code the parameter into the call to the procedure. In this section we will use a modified version of the PHPHelloWorld procedure which can accept a parameter, concatenate this to the word hello and return this via the out parameter.

-- MYSQL --

```
drop procedure if exists php_helloworld //

create procedure php_helloworld(IN p_in_param VARCHAR(30),OUT p_out_param VARCHAR(30))
begin
    set p_out_param = concat('Hello ',p_in_param);
end
//

call php_helloworld('Dave',@out_val) //
Query OK, 0 rows affected (0.00 sec)

select @out_val //
+-----+
| @out_val |
+-----+
| Hello Dave |
+-----+
1 row in set (0.00 sec)
```

 [php_procedure3.myp](#)

Now lets look at our first option, using a user variable in MySQL. All we need to do is call the appropriate MySQL code to create a user variable.

SET @name = 'Dave';

This can be done in PHP like so.

-- PHP --

```
mysqli_query($link,"SET @name = 'Dave'")
```

All we need to do now is amend our call to the PHPHelloWorld procedure to accept the parameter. As we have set the user variable this is as simple as just adding the variable name like so.

-- PHP --

```
mysqli_query($link,"call php_helloworld(@name, @out_param)")
```

If we add this to the code used in previous examples we can see the parameter in action.

```
<?php

$link = mysqli_connect("localhost","root","trustno1");

if (mysqli_connect_errno()) {
    echo "connection error";
}
```

```

exit();
}

mysqli_select_db ($link,"pers");

if ($result = mysqli_query($link,"SET @name = 'Dave'")) {
    echo "user variable set <br><br>";
} else { echo "user variable problem :( <br><br>"; }

if ($result = mysqli_query($link,"call php_helloworld(@name, @out_param)")) {
    if ($result = mysqli_query($link,"select @out_param")) {

        $row = mysqli_fetch_array($result, MYSQLI_NUM);

        printf($row[0]);

        mysqli_free_result($result);

    } else { echo "problem :( inner "; }
} else { echo "problem :( outer"; }

?>

```

[php_proc_helloworld2.phx](#)

Coding the second method is even simpler than the first. We don't need to set a user variable so we only need to make an amendment to one of the mysqli calls. As the call to the procedure is a string we can simply add the parameter in as a literal like so.

-- PHP --

```

mysqli_query($link,"call php_helloworld('Dave', @out_param)")

```

Thats all we need to change. So we can make this change in the PHP page and see the results.

```

<?php

$link = mysqli_connect("localhost","root","trustno1");

if (mysqli_connect_errno()) {
    echo "connection error";
    exit();
}

mysqli_select_db ($link,"pers");

if ($result = mysqli_query($link,"call php_helloworld('Dave', @out_param)")) {
    if ($result = mysqli_query($link,"select @out_param")) {

        $row = mysqli_fetch_array($result, MYSQLI_NUM);

        printf($row[0]);

        mysqli_free_result($result);

    } else { echo "problem :( inner "; }
} else { echo "problem :( outer"; }

?>

```

[php_proc_helloworld3.phx](#)

< [Calling Functions From PHP](#)

