# Optimum PHP Database Connectivity - the PDO Way by James Andy

Article published on June 25th 2012 | Software

Hypertext Processor or PHP, as it is more commonly known, has entranced the minds of developers worldwide. It is a general-purpose scripting language that can be embedded in HTML. One of the major advantages of PHP is its ability to talk with a variety of databases. This adds the much-needed interactive dimension to websites whereusers can fetch dynamic data and perform transactions. The heart of any web application is not its UI or its controller; itâ€™s the database that stores and provides data based on the queries that are made to it. PHP, being a widely popular programming language, is adopted across the world by dedicated PHP programmers handling large-scale projects for firms providing PHP Web Development Services. If you are one the countless offshore PHP developers who have handled database interactions, and still use the traditional PHP MySQL API for database connections, then this article is precisely targeted at you.

Did you know PHP offers 3 types of APIs for database connectivity? No? Well, hereâ€™s some information on them:

MySQL â€“ The most popular API which establishes database connectivity in the easiest of manners.

MySQLi â€“ An â€˜Improvedâ€™ version of the MySQL API.

PDO â€“ PHP Data Objects, an API that is database agnostic and supports prepared statements, thus negating the requirement for SQL Injection.

Looking at the above points, some might say feel that MySQL seems to be the easiest way out, then why should we try anything else! Well, let us give you three reasons. Firstly, some of the commonly used MySQL API database connectivity code snippets are all set to be deprecated as they do not adhere to the best practices policy are not exactly the right learning that greenhorn developers should pick up. Second, if you need to escape user input, you need to specifically do it from the code, which can be a complex endeavor, especially if you are new to database connectivity and donâ€™t have a clue about data sanitization. Third and the most important point; the MySQL API is inflexible which means you will come up a cropper if you decide to switch databases. Reason enough to consider PDO?

Remember the traditional way of connecting to a MySQL database:

```
# Connect

mysql_connect('localhost', 'username', 'password') or die('Could not connect: '. mysql_error());
```

Now consider the PDO technique where we create a new class instance, specify the driver, database name, username, and password:

```
$conn= newPDO('mysql:host=localhost;dbname=myDatabase', $username, $password);
```

The main advantage of this technique is that in the event of switching of databases, you can simply update the DSN. This reduces your dependency of MySQL and eliminates the absolute necessity of functions, such as mysql_connect.

But what about errors, you will ask! If the database connection is not established, how do we check the error? PDO offers 3 error mode options to deal with exceptions:

PDO::ERRMODE_SILENT

PDO::ERRMODE_WARNING

PDO::ERRMODE_EXCEPTION

The default error mode is PDO::ERRMODE_SILENT, which does not throw up any exceptions. You must remember to change the default mode to PDO::ERRMODE_EXCEPTION which throws up exceptions as soon as they occur and stops the execution of the script. You can write a simple try-catch code block to capture these exceptions. Failure to do so will require you to manually fetch these non-captured errors.

Now comes the most important part, the core function; retrieving information from the database. This can be done using two options, query and execute. The query method involves using the PDO::Quote method to manually escape data and is almost similar to the mysql_real_escape_string. A word of advice here; if you are dealing with data that a user fills in to a form, use prepared statements instead. For any other data that is not form data, the query method is ideal.

Prepared statements are used via the prepare method that use named parameters, eliminating the need for SQL injection. These statements are then executed by passing data into the named placeholders. Now we come to the part of receiving the data that is fetched by prepared statements.

The PDO API provides multiple options to interact with the result. These include:

PDO::FETCH_ASSOC â€“ default response, returns an array

PDO::FETCH_BOTH â€“ returns an array, indexed by both column-name, and 0-indexed

PDO::FETCH_BOUND â€“ returns TRUE and uses PHP variable for assignment of values in the columns

PDO::FETCH_CLASS â€“ returns a new instance of the class used

PDO::FETCH_OBJ â€“ returns an object that contains property names associated with columns

The PDO prepare extension is especially suited to the execution on the same query multiple times, each time with separate parameters. You can easily perform the common CRUD tasks that every PHP developer handles day in and day out. But the advantage that rises above all these features, is the usage of an object or class instance for mapping query results.  These inherent advantages should be proof enough for you to switch over from your old MySQL API to PDO. It will not only make your code more secure, but also streamline it to a great extent. And who can forget the coveted flexibility to switch over to any database! All of you, the PHP developer community including all offshore PHP developers, go PDO now, if you havenâ€™t already!

We are GoodCore, a leading Offshore Software Development Company in Malaysia. Having garnered expertise in every cutting-edge web development technology that emerges in the market, we apply our skills to the conceptualization, development and deployment of clutter-breaking solutions for clients across the world. We ensure that our team of developers, programmers and consultants are on top of every new development in the technology arena and are primed to lend that advantage to your product. Over the years, we have gained the trust and credibility of many elite clients across the globe. We wish to extend the same quality of services to you. You can hire dedicated PHP developers from us who will help develop cost-effective solutions that match your requirements to the T. If you wish to develop state-of-the-art PHP solutions that are not just path-breaking, but also ROI-intensive, donâ€™t hesitate to approach us. Our vast repertoire of talent is at your service!

Source: http://software-development-technologies.blogspot.com/

Article Source:
http://www.articleside.com/software-articles/optimum-php-database-connectivity-the-pdo-way.htm - Article Side

James Andy - About Author:
James Andy is technical writer and specialized in software development technologies. He has past five years experience of technical content writing, his strong area is Microsoft .net development technologies.

Article Keywords:
PHP development services

You can find more free articles on Article Side. Sign up today and share your knowledge to the community! It is completely FREE!