|  |
| --- |
| <?php |
|  | /\*\* |
|  | \* Copyright (C) Phppot |
|  | \* |
|  | \* Distributed under 'The MIT License (MIT)' |
|  | \* In essense, you can do commercial use, modify, distribute and private use. |
|  | \* Though not mandatory, you are requested to attribute Phppot URL in your code or website. |
|  | \*/ |
|  | namespace Phppot; |
|  |  |
|  | /\*\* |
|  | \* Generic datasource class for handling DB operations. |
|  | \* Uses MySqli and PreparedStatements. |
|  | \* |
|  | \* @version 2.7 - PDO connection option added |
|  | \*/ |
|  | class DataSource |
|  | { |
|  |  |
|  | // PHP 7.1.0 visibility modifiers are allowed for class constants. |
|  | // when using above 7.1.0, declare the below constants as private |
|  | // for better encapsulation |
|  | const HOST = 'localhost'; |
|  |  |
|  | const USERNAME = 'root'; |
|  |  |
|  | const PASSWORD = ''; |
|  |  |
|  | const DATABASENAME = 'signup'; |
|  |  |
|  | private $conn; |
|  |  |
|  | /\*\* |
|  | \* PHP implicitly takes care of cleanup for default connection types. |
|  | \* So no need to worry about closing the connection. |
|  | \* |
|  | \* Singletons not required in PHP as there is no |
|  | \* concept of shared memory. |
|  | \* Every object lives only for a request. |
|  | \* |
|  | \* Keeping things simple and that works! |
|  | \*/ |
|  | function \_\_construct() |
|  | { |
|  | $this->conn = $this->getConnection(); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* If connection object is needed use this method and get access to it. |
|  | \* Otherwise, use the below methods for insert / update / etc. |
|  | \* |
|  | \* @return \mysqli |
|  | \*/ |
|  | public function getConnection() |
|  | { |
|  | $conn = new \mysqli(self::HOST, self::USERNAME, self::PASSWORD, self::DATABASENAME); |
|  |  |
|  | if (mysqli\_connect\_errno()) { |
|  | trigger\_error("Problem with connecting to database."); |
|  | } |
|  |  |
|  | $conn->set\_charset("utf8"); |
|  | return $conn; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* If you wish to use PDO use this function to get a connection instance |
|  | \* |
|  | \* @return \PDO |
|  | \*/ |
|  | public function getPdoConnection() |
|  | { |
|  | $conn = FALSE; |
|  | try { |
|  | $dsn = 'mysql:host=' . self::HOST . ';dbname=' . self::DATABASENAME; |
|  | $conn = new \PDO($dsn, self::USERNAME, self::PASSWORD); |
|  | $conn->setAttribute(\PDO::ATTR\_ERRMODE, \PDO::ERRMODE\_EXCEPTION); |
|  | } catch (\Exception $e) { |
|  | exit("PDO Connect Error: " . $e->getMessage()); |
|  | } |
|  | return $conn; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* To get database results |
|  | \* |
|  | \* @param string $query |
|  | \* @param string $paramType |
|  | \* @param array $paramArray |
|  | \* @return array |
|  | \*/ |
|  | public function select($query, $paramType = "", $paramArray = array()) |
|  | { |
|  | $stmt = $this->conn->prepare($query); |
|  |  |
|  | if (! empty($paramType) && ! empty($paramArray)) { |
|  |  |
|  | $this->bindQueryParams($stmt, $paramType, $paramArray); |
|  | } |
|  | $stmt->execute(); |
|  | $result = $stmt->get\_result(); |
|  |  |
|  | if ($result->num\_rows > 0) { |
|  | while ($row = $result->fetch\_assoc()) { |
|  | $resultset[] = $row; |
|  | } |
|  | } |
|  |  |
|  | if (! empty($resultset)) { |
|  | return $resultset; |
|  | } |
|  | } |
|  |  |
|  | /\*\* |
|  | \* To insert |
|  | \* |
|  | \* @param string $query |
|  | \* @param string $paramType |
|  | \* @param array $paramArray |
|  | \* @return int |
|  | \*/ |
|  | public function insert($query, $paramType, $paramArray) |
|  | { |
|  | $stmt = $this->conn->prepare($query); |
|  | $this->bindQueryParams($stmt, $paramType, $paramArray); |
|  |  |
|  | $stmt->execute(); |
|  | $insertId = $stmt->insert\_id; |
|  | return $insertId; |
|  | } |
|  |  |
|  | /\*\* |
|  | \* To execute query |
|  | \* |
|  | \* @param string $query |
|  | \* @param string $paramType |
|  | \* @param array $paramArray |
|  | \*/ |
|  | public function execute($query, $paramType = "", $paramArray = array()) |
|  | { |
|  | $stmt = $this->conn->prepare($query); |
|  |  |
|  | if (! empty($paramType) && ! empty($paramArray)) { |
|  | $this->bindQueryParams($stmt, $paramType, $paramArray); |
|  | } |
|  | $stmt->execute(); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* 1. |
|  | \* Prepares parameter binding |
|  | \* 2. Bind prameters to the sql statement |
|  | \* |
|  | \* @param string $stmt |
|  | \* @param string $paramType |
|  | \* @param array $paramArray |
|  | \*/ |
|  | public function bindQueryParams($stmt, $paramType, $paramArray = array()) |
|  | { |
|  | $paramValueReference[] = & $paramType; |
|  | for ($i = 0; $i < count($paramArray); $i ++) { |
|  | $paramValueReference[] = & $paramArray[$i]; |
|  | } |
|  | call\_user\_func\_array(array( |
|  | $stmt, |
|  | 'bind\_param' |
|  | ), $paramValueReference); |
|  | } |
|  |  |
|  | /\*\* |
|  | \* To get database results |
|  | \* |
|  | \* @param string $query |
|  | \* @param string $paramType |
|  | \* @param array $paramArray |
|  | \* @return array |
|  | \*/ |
|  | public function getRecordCount($query, $paramType = "", $paramArray = array()) |
|  | { |
|  | $stmt = $this->conn->prepare($query); |
|  | if (! empty($paramType) && ! empty($paramArray)) { |
|  |  |
|  | $this->bindQueryParams($stmt, $paramType, $paramArray); |
|  | } |
|  | $stmt->execute(); |
|  | $stmt->store\_result(); |
|  | $recordCount = $stmt->num\_rows; |
|  |  |
|  | return $recordCount; |
|  | } |
|  | } |