PDO ile MYSQL’e bağlanmak

Genel olarak bağlantı olaylarını bir PDO sınıfını tanımlarken bir **DSN** yani “Data Source Name” ile belirtiyoruz. Hangi veritabanı sürücüsüne bağlanacağımızı ve bilgilerimizi DSN ile ifade ediyoruz. Diğer iki parametrede ise veritabanı kullanıcı adı ve şifremizi giriyoruz.

|  |
| --- |
| $db = new PDO('mysql:host=localhost;dbname=test', $user, $pass); |

Diğer sürücülerin DSN ifadeleri için [şuraya](http://www.php.net/manual/en/pdo.drivers.php) bakın.

|  |
| --- |
| $dsn = 'mysql:host=localhost;dbname=test';  $user = 'dbuser';  $password = 'mypassword';    try {      $db = new PDO($dsn, $user, $password);  } catch (PDOException $e) {      echo 'Connection failed: ' . $e->getMessage();  } |

Bu şekilde de bağlantı hatalarını yakalıyoruz.

Sorgu hatalarını yönetmek

Query ya da exec gibi sorgu gönderdiğimiz metodlar eğer sorguyu gerçekleştirdiklerinde bir hata ile karşılaşırlarsa **false** dönerler. Başarısız gerçekleşen bu sorguların hata mesajlarına ulaşmak için **errorInfo** metodunu kullanacağız.

Bu metod bize son yaptığımız sorgudaki hatanın kodunu ve mesajını içeren bir array döndürür. 3 adet elemanı bulunan bu dizide 0. ve 1. eleman hata kodlarını, 2. eleman ise hata mesajını verir.

|  |
| --- |
| if($users = $db->query('SELECT \* FROM users WHERE'))  {      // Sorgu başarıyla çalışırsa üyeleri listeleriz  }  else  {      echo 'Sorguda bir hata meydana geldi.';      $error = $db->errorInfo();      echo 'Hata mesajı: ' . $error[2];  } |

Örnekteki sorguda **WHERE** dedikten sonra herhangi bir koşul belirtmediğimiz için hata verecektir ve ekrana ilgili hatanın mesajı yazacaktır.

PDO ile sorgu göndermek

Eğer yapacağımız sorgudan bir sonuç almayı beklemiyorsak “**exec**“, bir sonuç isteniyorsa “**query**” metodlarını kullanmalıyız. Özetle; “DELETE/UPDATE/INSERT” gibi sorgularımız için “exec”, “SELECT” gibi sorgularımız için “query”.

|  |
| --- |
| try {      $db = new PDO('mysql:host=localhost;dbname=test', $user, $password);      $db->exec('SET NAMES `UTF-8`');      $count = $db->exec('DELETE FROM messages WHERE old = 1');      echo $count . ' messages deleted';  } catch (PDOException $e) {      echo 'Connection failed: ' . $e->getMessage();  } |

Exec metodu sonuç olarak etki ettiği satır sayısını döndürür.

Değişkenleri sorgulara dahil etmek

PDO’nun en önemli özelliklerinden birisi olan binding yöntemi sayesinde hazırladığımız sorgulara değişkenlerimizi güvenli ve düzgün bir şekilde yerleştirebiliyoruz.

Bunun için önce **prepare** metodu ile sorgumuzu hazırlayıp dışarıdan değerler vereceğimiz yerlere “?” (soru işareti) yerleştiriyoruz. Sonrasında hazırladığımızı sorguya **execute** metodu ile soru işareti olan yerlere gelecek değerlerimizi gönderiyoruz.

|  |
| --- |
| // Sorgumuzu hazırlıyoruz  $query = $db->prepare('INSERT INTO users (name, email) VALUES(?, ?)');    // Sorguda belirttiğimiz yerlere gelecek değerleri veriyoruz  $query->execute(array('Musa', 'email@email.com)); |

Bu ifadeyi çalıştırdığımızda çalıştırılacak sorgu aşağıdaki gibi olacaktır;

|  |
| --- |
| INSERT INTO users (name, email) VALUES('Musa', 'email@email.com') |

PDO kullanırken tüm dışarıdan aldığımız değişkenleri sorgularımıza bu yöntem ile dahil etmemiz uygulamamızın güvenliği ve düzeni açısından çok önemlidir. Bu yöntem sayesinde **SQL injection**açıklarından da arınmış oluyoruz.

PDO ile verileri listelemek

Bunun için yukarıda bahsettiğim **query** metodunu kullanacağız.

|  |
| --- |
| foreach($db->query('SELECT \* FROM users') as $row) {      echo $row['name'] . '<br/>';  } |

PDO ile bir satır veri çekmek

Bunun için query ile sorgumuzu çağırdıktan sonra fetch ile ilk sonucu alacağız.

|  |
| --- |
| $row = $db->query('SELECT \* FROM users WHERE id = 1')->fetch();  echo $row['name']; |

Yeni bir kayıt eklemek

Yeni bir kayıt eklemek için **exec** metodunu kullanacağız. Sonrasında eğer eklediğimiz satırın ID’sini almak istiyorsak **lastInsertId** metodunu çağıracağız. Eklerken bir sorun gerçekleşirse exec metodu false dönecektir. Bu nedenle öncelikle eklenip eklenmediği kontrol edip sonrasında ID’yi ekrana yazdırabiliriz.

|  |
| --- |
| if($db->exec('INSERT INTO users (name) VALUES ("Musa")'))  {      $id = $db->lastInsertId();      echo 'Yeni eklenen üyenin IDsi: ' . $id;  }  else  {      echo 'Yeni kayıt eklerken bir hata meydana geldi.';  } |

Sorguları geri alabilme

PDO’nun bir diğer önemli özelliği ise **transaction** denilen ifade edilen sorguları istenildiğinde geri alabilme ya da uygulayabilmesidir.

Yukarıdaki sorgu ifadelerini kullanmadan önce **beginTransaction** metodunu çağırarak geri alma işlemi yapabileceğimiz sorguları yazmaya başladığımızı ifade etmemiz gerekiyor.

BeginTransaction metodunu çalıştırdıktan sonra yazacağımız tüm ekleme, düzenleme ve silme sorguları normal kullandığımız şekilde çalışmaya devam eder. Eğer sorgular devam ederken tüm bu aralıktaki sorguları geri almak istersek **rollBack** metodunu çağırırız. Bu metodu çalıştırdığımızda beginTransaction’dan rollBack metoduna kadar yazdığımız tüm ekleme, düzenleme ve silme sorguları geri alınır.

Bu ifadeleri kullanırken bilmeniz gereken önemli bir konu; **DROP TABLE** ya da **CREATE TABLE**gibi tabloları komple silen ya da yeni tablolar ekleyen sorgular **geri alınamaz**. Fakat DELETE sorgusu ile sildiğiniz ya da UPDATE ile düzenlediğiniz her sorguyu geri alabilirsiniz.

|  |
| --- |
| // İşlemleri başlattığımızı ifade ediyoruz  $db->beginTransaction();    $db->exec('INSERT INTO users (name) VALUES ("Ahmet")');  $db->exec('UPDATE users SET name = "Ali"');  $db->exec('DELETE FROM users WHERE name = "Hasan"');    // Yaptığımız sorguları geri aldık  $db->rollBack(); |

Yaptığımız sorguları geri almak yerine başarıyla sonuçlandığında gerçekten uygulamak istiyorsak**commit** metodunu çağırırız.

PDO bağlantısını kapatmak

Eski MySQL bağlantısında da olduğu gibi açtığınız bağlantıyı kapatmasanız da PHP otomatik olarak işlem sonunda bağlantıyı keser. Fakat ille de ben kapatacağım diyorsanız “**null**” tanımlamanız yeterli.

|  |
| --- |
| // bağlantıyı açtık  $db = new PDO('mysql:host=localhost;dbname=test', $user, $pass);[    // bağlantıyı kapattık  $db = null; |

Dahası …

Ben bu yazıda PDO’nun temel ihtiyaç duyulan ifadelerinden bahsetmeye çalıştım. Fakat bununla yetinmeyip PDO hakkında daha fazla bilgi edinmek için [şuralardaki](http://www.php.net/manual/tr/book.pdo.php) örnekleri ve açıklamaları incelemeye devam edin.

*Bir sonraki serüvende görüşmek üzere, hoşçakalın.*