# WEB PROGRAMLAMA II

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

# Ders İzlencesi

| Hafta | Modüller/İçerik/Konular |
|---|---|
| 1. Hafta | Oturum yönetimi |
| 2. Hafta | Cookies kullanımı ve oturum yönetimi |
| 3. Hafta | Session kullanımı ve oturum yönetimi |
| 4. Hafta | Sayfalama ve Arama İşlemleri |
| 5. Hafta | JavaScript, JQuery ve PHP |
| 6. Hafta | AJAX ve PHP |
| 7. Hafta | AJAX ve PHP |
| 8. Hafta | ARA SINAV |
| 9. Hafta | PHP'de Nesne Yönelimli Programlamaya Giriş(347) |
| 10. Hafta | PHP de PDO |
| 11. Hafta | |
| 12. Hafta | |
| 13. Hafta | |
| 14. Hafta | |

# Bu Ünitede Ele Alınan Konular

- PDO Nedir?
- Neden PDO?
- PHP Data Objects
- PDO ile Veritabanı Bağlantısı
- PDO Fonksiyonları
- PDO Temel Veritabanı İşlemleri

# PDO PHP Data Objects

PHP veri nesneleri (PDO), PHP'de veri tabanı işlemlerini daha kolay ve esnek bir halde uygulamak için geliştirilmiştir. Kullanım yapısı daha önce gördüğümüz MYSQLI sınıf yapısına benzemektedir. MYSQLI ye göre daha esnek ve daha kullanışlı olduğu görülmektedir. Ayrıca en önemli farklarından biri de birden fazla veritabanına entegre olabilme imkanı sunmasıdır. Yani sadece DSN parametresinin değerini ayarlayarak tüm kodlarınızı bir anda farklı bir veritabanına uygun hale getirebiliyorsunuz.

Bu gerçekten güzel ve esnek bir özellikle PostgreSQL, MySQL, ORACLE, SQLite gibi veritabanlarına kolayca entegre edile bilmektedir.

# PDO PHP Data Objects

**PDO**(PHP Data Objects / PHP Veri Objeleri) özetle; hafif ve tutarlı bir şekilde veritabanına erişimi sağlayan bir arayüz. Adından da anlayacağınız üzerie “Object Oriented Programming” arayüzüne sahip, onlarca veritabanı sürücüsü destekliyor;

PDO nun desteklediği diğer veritabanları ;

- Cubrid
- FreeTDS / Microsoft SQL Server / Sybase
- Firebird/Interbase 6
- IBM DB2
- IBM Informix Dynamic Server
- MySQL 3.x/4.x/5.x
- Oracle Call Interface
- ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
- PostgreSQL
- SQLite 3 and SQLite 2
- Microsoft SQL Server / SQL Azure

Bunların haricinde PDO 5.1‘den itibaren geliyor, yani çalışabilmesi için güncel versiyonlara ihtiyacınız olacak.

# PDO PHP Data Objects

PDO yu kullanabilmek için öncelikle PHP sunucunuzdaki PDO eklentilerini aktif etmeniz gerekmektedir. Bunun için sunucunuzdaki php.ini dosyasından extension=php_pdo.dll gibi PDO eklentilerinin önündeki ;(noktalı virgül) leri silmelisiniz. Daha sonra sunucunuzu yeniden başlatarak PDO fonksiyonlarını kullanmaya başlayabilirsiniz.

Sisteminizde açık ve kullanılabilir PDO veritabanı sürücülerini görmek için aşağıdaki fonksiyonu kullanabilirsiniz. Böylece sunucunuzda PDO tarafından desteklenen veritabanlarını öğrenebilirsiniz.

```
<?php
        print_r(PDO:: getAvailableDrivers());
?>
```

# PDO PHP Data Objects

PDO ile veritabanı bağlantısını gerçekleştirebilmek için öncelikle PDO sınıfından bir nesne üretmeniz gerekmektedir. PDO sınıfının kurucu metodu bizden veritabanı türü, headerlar, veritabanı kullanıcı bilgileri ve adı gibi bazı parametreleri istemektedir. Sınıfın kullanım şekli aşağıdaki gibidir.

```
PDO( string $dsn [, string $kullanıcı [, string $parola [, array $seçenekler ]]] )
```

**dsn** değiştirgesi, bir veritabanı bağlantısı oluşturmak için kullanılır ve veritabanına göre yazım dizgesi değişmektedir. Üç farklı parametreyi ister bunlar :

```
$dsn = 'mysql:host=localhost;port=3307;dbname=ders';
```

Diğer parametreler ise kullanıcı adı ve şifresidir. Aşağıda örnek bir MYSQL bağlantısı vardır.

```
<?php
$vt = new PDO('mysql:host=localhost;port=3307;dbname=ders','root','usbw');
?>
```

Bu bağlantıda eğer bir hata olursa direkt ekrana yazar hata kontrolü yapmak istiyorsak TRY -CATCH bloklarını kullanmak zorundayız.

# PDO PHP Data Objects

```php
<?php
try{
  $vt = @new PDO('mysql:host=localhost;port=3307;dbname=ders','root','usbw');
    }catch (PDOException  $e){
          echo "PDO Bağlantı Hatası ".$e->getMessage();
         }
?>
```

Bu şekilde meydana gelen hatalar kontrollü bir şekilde ekrana bastırılmış olmaktadır.

# PDO PHP Data Objects

PDO ile veritabanı bağlantısında sadece ilk parametre olan **mysql**:host=localhost;port=3307;dbname=ders parametresi değişmektedir. Örnek olarak yazım şekli olarak farklı veritabanları için;

```
 # MS SQL Server and Sybase with PDO_DBLIB
  $DBH = new PDO("mssql:host=$host;dbname=$dbname, $user, $pass");
  $DBH = new PDO("sybase:host=$host;dbname=$dbname, $user, $pass");

  # MySQL with PDO_MYSQL
  $DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);

  # SQLite Database
  $DBH = new PDO("sqlite:my/database/path/database.db");
```

# PDO PHP Data Objects

Daha düzenli bir PDO ile veritabanı bağlantısı için PDO içerisindeki parametreleri başlangıçta değişken olarak tanımlamak daha mantıklıdır. Bu formata uygun şekilde aşağıdaki gibi olmalıdır.

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders';
    $user = 'root';
    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
        }catch (PDOException  $e){
                echo "PDO Bağlantı Hatası ".$e->getMessage();
               }
?>
```

# PDO PHP Data Objects

PDO ile veritabanı bağlantısını açtıktan sonra işimiz bittiğinde veritabanı bağlantısını kapatmak için sadece nesne değerine NULL yüklememiz yeterli olacaktır..

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
    $user = 'root';
    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
        }catch (PDOException  $e){
                echo "PDO Bağlantı Hatası ".$e->getMessage();
               }
    $vt= NULL;
?>
```

# PDO PHP Data Objects

PDO da SQL sorgularını çalıştırmak için exec() metodunu kullanmaktayız.

Veritabanına her hangi bir veriyi eklemek için bir sonraki sayfada bulunan kodu kullanmalıyız.

Kodu inceleyecek olursak veritabnına bağlandıktan sonra SQL sorgusunu exec() metodu ile çalıştırdığımız görülmektedir. Bu metod ile bir SQL sorgusu çalıştırdığımızda bize eğer sorgu başarısız ise FALSE değeri döner başarılı ise işlemden etkilenen toplam satır sayısını döndürür.

# PDO PHP Data Objects

PDO ile veri ekleme;

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
    $user = 'root';    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
    echo "PDO Bağlantısı Başarılı";
    $sonuc=$vt->exec("INSERT INTO `bolum` (`adi`,`adres`) VALUES('Büro Yönetimi','Elmadağ')");
            if($sonuc===false)
                        echo "<br>PDO Sorgu Hatası";
            else
                        echo "<br>Sorgu Başarılı Eklenen Satır Sayısı =".$sonuc;
        }catch (PDOException  $e){
                echo "PDO Bağlantı Hatası ".$e->getMessage();
               }
    $vt= NULL;
?>
```

# PDO PHP Data Objects

PDO ile veri ekleme sonuç;

Array ( [0] => mysql [1] => pgsql [2] => sqlite )
Bağlandı
Sorgu Başarılı Eklenen Satır Sayısı =1

# PDO PHP Data Objects

PDO ile veri ekleme sonucunu incelediğimizde eklediğimiz satırda ğ gibi Türkçe karakterlerin bozuk olduğunu görebiliriz. Bunu önleyebilmek için Karakter kümesini UTF8 olarak ayarlamalıyız. Bunu yine Exec(); metodu ile yapacağız;

```
Exec('SET NAMES UTF8');
```

Bu şekilde Türkçe karakter sorununu da çözebiliriz.

# PDO PHP Data Objects

PDO ile veri ekleme; Karakter sorununu çözdük.

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
    $user = 'root';    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
            $vt->exec("SET NAMES UTF8");
    echo "PDO Bağlantısı Başarılı";
    $sonuc=$vt->exec("INSERT INTO `bolum` (`adi`,`adres`) VALUES('Büro Yönetimi','Elmadağ')");
            if($sonuc===false)
                        echo "<br>PDO Sorgu Hatası";
            else
                        echo "<br>Sorgu Başarılı Eklenen Satır Sayısı =".$sonuc;
       }catch (PDOException  $e){
              echo "PDO Bağlantı Hatası ".$e->getMessage();
             }
    $vt= NULL;
?>
```

# PDO PHP Data Objects

PDO ile veri ekleme sonuç; Türkçe Karakter Sorunu Yok

Array ( [0] => mysql [1] => pgsql [2] => sqlite )
Bağlandı
Sorgu Başarılı Eklenen Satır Sayısı =1

+ Seçenekler

| | bno | adi | adres |
|---|---|---|---|
| Düzenle Kopyala Sil | 1 | Bilgisayar | Emyo |
| Düzenle Kopyala Sil | 2 | Elektronik | Elmadağ |
| Düzenle Kopyala Sil | 3 | Muhasebe | *NULL* |
| Düzenle Kopyala Sil | 4 | Büro Yönetimi | Elmadağ |
| Düzenle Kopyala Sil | 7 | Büro Yönetimi | Elmadağ |

# PDO PHP Data Objects

PDO ile veri eklerken enson eklediğiniz satıra ait verinin id nosunu alabilirsiniz bunun için,

```
$vt->lastInsertedId();
```

kodu kullanılır.

# PDO PHP Data Objects

PDO ile veri çekmek ve listelemek için iki farklı yol vardır birincisi query() metodu ikincisi MYSQLIde de gördüğümüz PREPARE() ve bindParam() metodlarıdır.

QUERY() metodu iki parametre alır, birincisi sorgunun kendisi ikincisi ise dönen sonuçların diziye nasıl yükleneceğidir. Diziye yüklerken FETCH() metoduna göre modunu belirtiriz.

PDO::FETCH_ASSOC,

PDO::FETCH_NUM,

PDO::FETCH_BOTH,

PDO::FETCH_OBJ

gibi değerler almaktadır gelen sonuçlar bu değerlere göre diziye yüklenir.

# PDO PHP Data Objects

PDO ile veri ekleme; Karakter sorununu çözdük.

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
    $user = 'root';    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
            $vt->exec("SET NAMES UTF8");
    echo "PDO Bağlantısı Başarılı";
    $sonuc=$vt->query("SELECT * FROM `bolum`",PDO::FETCH_ASSOC);
            if($sonuc===false)
                        echo "<br>PDO Sorgu Hatası";
            else
                        foreach( $sonuc as $veri)
                              echo"<br>".$veri['adi'];
       }catch (PDOException  $e){
              echo "PDO Bağlantı Hatası ".$e->getMessage();
             }
    $vt= NULL;
?>
```

# PDO PHP Data Objects

PDO ile veri ekleme; Karakter sorununu çözdük.

```
<?php
    $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
    $user = 'root';    $sifre = 'usbw';
    try{
      $vt = @new PDO($dsn , $user , $sifre);
            $vt->exec("SET NAMES UTF8");
    echo "PDO Bağlantısı Başarılı";
    $sonuc=$vt->query("SELECT * FROM `bolum`",PDO::FETCH_NUM);
            if($sonuc===false)
                        echo "<br>PDO Sorgu Hatası";
            else
                        foreach( $sonuc as $veri)
                              echo"<br>".$veri[1];
       }catch (PDOException  $e){
              echo "PDO Bağlantı Hatası ".$e->getMessage();
             }
    $vt= NULL;
?>
```

# PDO PHP Data Objects

QUERY() metodu dışarıdan gelen verilere karşı savunmasızdır yani SQLINJECTION gibi saldırılara duyarlı değildir bunun için dışarıdan gelen verileri kontrol etmemiz gerekir.

Dışarıdan gelen verileri **QUOTE()** metodu ile kontrol etmeliyiz. Kullanım şekli **$veri = $vt-> QUOTE($veri);** şeklindedir. GET veya POST ile gelen bir veriyi **$_GET['adi'] = $vt->QUOTE($_GET['adi'] );** Şeklinde kontrol edebiliriz.

Tabi bu şekilde tek tek tüm verileri kontrol etmek zor olduğu için istersek daha önce öğrendiğimiz array_map() metodu ile bir kerede kontrol sağlayabiliriz.

```
$_POST = array_map(array($vt,'quote'),$_POST);
```

# PDO PHP Data Objects

QUERY() metodu dışarıdan gelen verilere karşı savunmasız olduğu için daha çok prepare ve bindParam metodları kullanılmaktadır. Prepare metodu sorguyu bir kez inceler ve hataları zararlı kodları düzenler ve birden çok kez çalıştırmaya olanak sağalar

Bu nedenle daha hızlı çalışan bir koddur. Kullanım şekli ise daha önce gördüğümüz MYSQLI ile aynıdır.

# PDO PHP Data Objects

PDO ile veri ekleme; Karakter sorununu çözdük.

```
<?php
      $dsn = 'mysql:host=localhost;port=3307;dbname=ders‘;
      $user = 'root';    $sifre = 'usbw';
      try{
        $vt = @new PDO($dsn , $user , $sifre);
                  $vt->exec("SET NAMES UTF8");
      echo "PDO Bağlantısı Başarılı";
      $ad=$_GET['ad'];
                  $sonuç = $vt->prepare("SELECT * FROM `ogrenci` WHERE `adi`=?");
                  $sonuc->bindParam(1,$ad,PDO::PARAM_STR);
                  $sonuc->execute();
                  if($sonuc===false)
                                    echo "<br>PDO Sorgu Hatası";
                  else
                                   while( $veri=$sonuc->fetch(PDO::FETCH_ASSOC) )
                                    echo"<br>".$veri['adi'];

                     }catch (PDOException  $e){
                   echo "PDO Bağlantı Hatası ".$e->getMessage();
                  }
      $vt= NULL;
?>
```

# PDO PHP Data Objects

bindParam metodu ile birden fazla veri bağlanacaksa tek tek yazmalıyız.

```
$sonuc=$vt->prepare("SELECT * FROM `ogrenci` WHERE `adi`=? AND `no`=?");
$sonuc->bindParam(1,$ad,PDO::PARAM_STR);
$sonuc->bindParam(2,$no,PDO::PARAM_INT);
```

# PDO PHP Data Objects

bindParam metodu ile değişkenleri bağlarken istersen ? Yerine değişken adı da kullanabiliriz.

```
$sonuc=$vt->prepare("SELECT * FROM `ogrenci` WHERE `adi`=:ad AND `no`=:no");
$sonuc->bindParam(':ad',$ad,PDO::PARAM_STR);
$sonuc->bindParam(':no',$no,PDO::PARAM_INT);
```

# PDO PHP Data Objects

Verileri sorguya bağlama işlemini bindParam metodu ile yapabildiğimiz gibi istersek sadece execute() metodu ile de yapabiliriz. Kullanım şekli aynı bindParam da olduğu gibidir. Sadece bu kez parametreleri bindParam içine değil execute içerisine yazıyoruz. Ama tekbir farkla değerleri dizi halinde gönderiyoruz.

```
$sonuc=$vt->prepare("SELECT * FROM `ogrenci` WHERE `adi`= :ad AND  `no`=:no");
$sonuc->execute(array(':ad'=>$ad, ':no'=>$no));
```

Soru işareti olduğunda ise ;

```
$sonuc=$vt->prepare("SELECT * FROM `ogrenci` WHERE `adi`= ?  AND  `no`=?");
$sonuc->execute(array($ad, $no));
```

# PDO PHP Data Objects

PDO ile Güncelleme işlemleri daha önce ekleme işlemlerinde olduğu gibi EXEC() metodu ile yapılabilir ancak daha önce de söylediğimiz gibi bu metod dışarıdan gelen SQL saldırılarına açıktır. Bu sebeple prepare ve bindParam ile kullanılması daha mantıklıdır. Yada EXECUTE ile yapılır.

Bir sonraki sayfada execute metodu ile yapılmış bir güncelleme örneği bulunmaktadır.

Aynı örneği iki sonraki sayfada prepare ve bindParam ile de yaptık.

# PDO PHP Data Objects

```php
<?php
try{
   $vt=new PDO('mysql:host=localhost;port=3307;dbname=UEOkul','root','usbw');
            $vt->exec("SET NAMES UTF8");

            $ad=$_GET['ad'];        $bl=$_GET['bol'];        $cns=$_GET['cns'];        $tc=$_GET['tc'];
            $sonuc=$vt->prepare("UPDATE `ogrenci` SET `adi`=?, `bolum`=? ,`cins`=? WHERE `tcNo`=?");
            $sonuc->execute(array($ad,$bl,$cns,$tc));

            if($sonuc===false)
                        echo"<br>PDO Sorgu Hatası";
            else{
                        echo $sonuc->rowCount()." Satır veri güncellendi";
            }
}catch(PDOException $e)
{
            echo $e->getMessage();
}
?>
```

# PDO PHP Data Objects

```php
<?php
try{
   $vt=new PDO('mysql:host=localhost;port=3307;dbname=UEOkul','root','usbw');
                $vt->exec("SET NAMES UTF8");

                $ad=$_GET['ad'];          $bl=$_GET['bol'];          $cns=$_GET['cns'];          $tc=$_GET['tc'];
                $sonuc=$vt->prepare("UPDATE `ogrenci` SET `adi`=:adi, `bolum`=:bo ,`cins`=:cn WHERE `tcNo`=:tcm");
                $sonuc->bindParam(':adi',$ad,PDO::PARAM_STR);
                $sonuc->bindParam(':bo',$bl,PDO::PARAM_STR);
                $sonuc->bindParam(':cn',$cns,PDO::PARAM_STR);
                $sonuc->bindParam(':tcm',$tc,PDO::PARAM_INT);
                $sonuc->execute();

                if($sonuc===false)
                                echo"<br>PDO Sorgu Hatası";
                else{
                                echo $sonuc->rowCount()." Satır veri güncellendi";
                }
}catch(PDOException $e)
{
                echo $e->getMessage();
}
?>
```

# PDO PHP Data Objects

PDO ile veri Silme işlemleri de aynı güncelleme işlemlerinde olduğu gibi EXEC() metodu ile yapılabilir ancak daha önce de söylediğimiz gibi bu metod dışarıdan gelen SQL saldırılarına açıktır. Bu sebeple prepare ve bindParam ile kullanılması daha mantıklıdır. Yada EXECUTE ile yapılır.

Bir sonraki sayfada execute metodu ile yapılmış bir veri silme örneği bulunmaktadır.

Aynı örneği iki sonraki sayfada prepare ve bindParam ile de yaptık.

# PDO PHP Data Objects

```php
<?php
try{
   $vt=new PDO('mysql:host=localhost;port=3307;dbname=UEOkul','root','usbw');
            $vt->exec("SET NAMES UTF8");

              $tc=$_GET['tc'];
            $sonuc=$vt->prepare("DELETE FROM `ogrenci`  WHERE `tcNo`=?");
            $sonuc->execute(array($tc));

            if($sonuc===false)
                        echo"<br>PDO Sorgu Hatası";
            else{
                        echo $sonuc->rowCount()." Satır veri Silindi";
            }
}catch(PDOException $e)
{
            echo $e->getMessage();
}
?>
```

# PDO PHP Data Objects

```php
<?php
try{
   $vt=new PDO('mysql:host=localhost;port=3307;dbname=UEOkul','root','usbw');
            $vt->exec("SET NAMES UTF8");

            $tc=$_GET['tc'];
                $sonuc=$vt->prepare("DELETE FROM `ogrenci`  WHERE `tcNo`=:tcm");
            $sonuc->bindParam(':tcm',$tc,PDO::PARAM_INT);
            $sonuc->execute();

            if($sonuc===false)
                        echo"<br>PDO Sorgu Hatası";
            else{
                        echo $sonuc->rowCount()." Satır veri silindi";
            }
}catch(PDOException $e)
{
            echo $e->getMessage();
}
?>
```

# PDO PHP Data Objects

PDO ile veri Silme işlemlerini daha güvenli şekilde yapabiliyoruz. Örneğin bir silme işlemi belirli veriler silindikten sonra yarıda hata meydana gelirse geri alınabiliyor. Bu şekilde veri kayıplarının önüne geçilerek daha güvenli bir işlem meydana getiriliyor.

İşlemlerin geri alına bilinmesi için sınıfın **beginTransaction()** metodu kullanılmaktadır. Bu metod verilerin işleneceği bir güvenli başlangıç noktası oluşturur. Böylece işlemler geri alınmak istendiğinde bu başlangıç noktasından sonraki yapılan işlemler geri alınır.

Verileri geri alma işlemini bu sınıftaki **rollBack()** metodu yapar, commit() metodu ise sorunsuz çalışan işlemler silsilesinden sonra işlemleri kalıcı hale getirir. **Commit()** metodundan sonra işlemler geri alınamaz

# PDO PHP Data Objects

```php
<?php
try{
   $vt=new PDO('mysql:host=localhost;port=3307;dbname=UEOkul','root','usbw');
                $vt->exec("SET NAMES UTF8");
                $tc=$_GET['tc'];    $tc1=$_GET['tc1'];
                $vt->beginTransaction(); //İşlemi geri almak için Transaction başlatalım
                $sonuc=$vt->prepare("DELETE FROM `ogrenci`  WHERE `tcNo`=:tcm");
                $sonuc->bindParam(':tcm',$tc,PDO::PARAM_INT);
                $sonuc->execute();

                $sonuc=$vt->prepare("DELETE FROM `ogrenci`  WHERE `tcNo`=:tcm");
                $sonuc->bindParam(':tcm',$tc1,PDO::PARAM_INT);
                $sonuc->execute();
                if($sonuc==false){
                                echo"<br>PDO Sorgu Hatası işlemi geri alalım";
                   $vt->rollBack();//işlemi geri alalım
                }else{
                                echo $sonuc->rowCount()." Satır veri güncellendi";
                                if($sonuc->rowCount()<=0){
                                                $vt->rollBack();//işlemi geri alalım
                                                 echo"<br>işlemi geri alalm";
                                }else
                                                $vt->commit();//işlemi tamamla
}catch(PDOException $e){
                echo $e->getMessage();
}
?>
```

# KAYNAKLAR

- İnternet ortamı
- PHP ve AJAX  Haydar TUNA
- A'dan Z'ye PHP Rıza  ÇELİK