WEB TASARIM I

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

Ders İzlencesi

Hafta	Modüller/İçerik/Konular
1. Hafta	PHP Tanımı ve Sunucu Kurulumları
2. Hafta	PHP Yazım Notasyonu ve Değişkenler
3. Hafta	PHP de Karar kontrol yapıları ve Döngüler
4. Hafta	Dizi ve Dizi işlemleri
5. Hafta	Fonksiyon, Sınıf ve Nesne Kavramları
6. Hafta	HTML ve PHP
7. Hafta	PHP ile Veritabanı İşlemleri
8. Hafta	MYSQL, MYSQLİ
9. Hafta	AJAX
10. Hafta	
11. Hafta	
12. Hafta	
13. Hafta	
14. Hafta	

Bu Ünitede Ele Alınan Konular

- HTML, CSS, JavaScript ve PHP
- MYSQLI (Improved)
- MYSQLI ile Veritabanı bağlantı işlemleri
- MYSQLI ile Veritabanı seçme işlemleri
- MYSQLI ile Veritabanı sorgu işlemleri
- MYSQLI ile Veritabanı kayıt listeleme işlemleri
- MYSQLI ile Veritabanı kayıt ekleme işlemleri
- MYSQLI ile Veritabanı kayıt güncelleme işlemleri
- MYSQLI ile Veritabanı kayıt silme işlemleri
- MYSQLI ile Veritabanı bellek boşaltma işlemleri
- MYSQLI ile Veritabanı bağlantı kapatma işlemleri

Ders Kazanımları

Bu bölümü Bitirdiğimizde,

- HTML, CSS, JavaScript ve PHP
- MYSQLI (Improved)
- MYSQLI ile Veritabanı bağlantı işlemleri
- MYSQLI ile Veritabanı seçme işlemleri
- MYSQLI ile Veritabanı sorgu işlemleri
- MYSQLI ile Veritabanı kayıt listeleme işlemleri
- MYSQLI ile Veritabanı kayıt ekleme işlemleri
- MYSQLI ile Veritabanı kayıt güncelleme işlemleri
- MYSQLI ile Veritabanı kayıt silme işlemleri
- MYSQLI ile Veritabanı bellek boşaltma işlemleri
- MYSQLI ile Veritabanı bağlantı kapatma işlemleri

öğrenmiş olacaksınız.

HTML, PHP ve MYSQLI

PHP de veritabanı işlemleri için veritabanı MYSQL fonksiyonlarını bir önceki sunumumuzda kullandık. MYSQL fonksiyonlarının SQL INJECTION gibi açıkları bulunduğuna da değindik. Ayrıca Türkçe karakter sorunu gibi bazı sorunları da vardı. Bu tür sorunlardan kurtulmak için sürekli aynı kodları yazmak yerine tek yapmamız gerekenin bu kodların bir arada kullanıldığı sınıf ve metodlarımızı oluşturmak ve böylece sürekli yazmaktan kurtulmak olduğu aşikardır. Yani diğer bir deyişle kendi kütüphanemizi yada framework ümüzü oluşturmamız gerekiyor.

Bu gereksinimin farkına varmış olacak ki MYSQL grubuda veritabanı 4.1 versiyonu ile birlikte MYSQLI (IMPROVED) yani geliştirilmiş MYSQL versiyonunu yayınladı.

HTML, PHP ve MYSQLI

MYSQLI ile birlikte bir çok aksaklık giderildi ve nesne yönelimli programlama (OOP) alt yapısına göre veritabanı fonksiyonları tekrar oluşturuldu. Böylece her işlem için ayrı ayrı acaba zararlı SQL kodu var mı? Acaba açık bir nokta bıraktık mı? Diye düşünmek zorunda kalmadan bu frameworkteki fonksiyonları kullanarak güvenli bir veritabanı bağlantısını daha hızlı bir şekilde oluşturma imkanı bulabiliyoruz.

Gerçi bana sorarsanız en güvenli kod bildiğiniz koddur ☺. Sonuç olarak adamların geliştirdiği hazır kodları kullanıyoruz ve içlerinde ne var bilmiyoruz. Yapabiliyorsanız, ki çokta zor değil, kendi kütüphanenizi oluşturmanız en mantıklısı.

HTML, PHP ve MYSQLI

MYSQLI aslında MYSQL fonksiyonlarının kullanıldığı sınıfın adıdır. Bu sebeple tüm MYSQLI veritabanı işlemleri için, bu sınıftan bir nesne oluşturarak kullanacağız.

Veritabanına bağlanmak için bu sınıftan bir nesne oluşturmak yeterlidir. Kurucu metoduna tanımlandığı için nesne oluşur oluşmaz bağlantı gerçekleşmektedir.

Kullanım şekli;

```
$veriTabani= new mysqli('adres', 'kullanıcı Adı', 'Şifre', 'veritabani adi');
```

Eğer bağlantı başarılı ise bağlantı nesnesi döner. Bağlantının ve veritabanı seçiminin başarılı olup olmadığını bu sınıfa ait connect_error değişkeni ile kontrol edebiliriz, başarılı ise true değilse hata mesajı döner.

HTML, PHP ve MYSQLI

```
$veriTabani=@new mysqli('localhost','root','usbw','ogrenci');

 if( $veriTabani->connect_error)
    echo $veriTabani->connect_error;
            else{
               echo "VT Bağlantısı ve seçimi başarılı";
               $veriTabani->close();
            }
```

$veriTabani Adında bir değişken oluşturduk ve mysqli nesnesini bu değişkene yükledik artık mysqli sınıfına ait tüm işlemlere bu değişkenle ulaşacağız.

Gerekli bilgileri parametre olarak gönderip mysqli nesnemizi oluşturduktan sonra bağlantının başarılı olup olmadığını $veriTabani->connect_error değişkeninin değerine bakarak kontrol ettik.

$veriTabani->close(); metodu mysql_close() fonksiyonunun görevini yaparak veritabanı bağlantısını kapatıyor.

Öğr. Gör. M. Mutlu YAPICI

HTML, PHP ve MYSQLI

Mysqli ile sorgu işlemlerini yine MYSQL de olduğu gibi query() fonksiyonu ile yapacağız. Gelen sonuçları da yine fetch() fonksiyonları ile dizilere yükleyeceğiz.

```
$veriTabani=@new mysqli('localhost','root','usbw','ogrenci');

 if( $veriTabani->connect_error)
   echo $veriTabani->connect_error;
            else{
                 echo "VT Bağlantısı ve seçimi başarılı";
                    $sql="SELECT * FROM `ogrencibilgileri`";
                    $sonuc= $veriTabani->query($sql);
                  if($sonuc){
                        $dizi= $sonuc->fetch_assoc();
                        print_r($dizi);
                   } else
                        echo "Sorgu Hatası";
            $veriTabani->close();
          }
```

$sql değişkeninde tanımlı SQL cümleciğini query() fonksiyonu ile sorguya gönderdik ve bize gelen cevap sorgu başarılı ise yeni bir nesnedir, başarılı değilse false değeridir.

Gelen cevaptaki nesnenin fetcth_assoc() fonksiyonu ile tüm verileri (verilerin ilk satırını) dizi değişkenine yükledik ve içeriğini yazdırdık. Fetch_assoc(), fetch_row() ve fetch_array() olmak üzere 3 farklı fonksiyon var. Bu fonksiyonların farklarını daha önceki sunumda anlatmıştık burada da aynı şekilde çalışıyorlar.

HTML, PHP ve MYSQLI

MYSQLI ile devam etmeden önce MYSQLIND sürücüsü kurulumu test etmeliyiz, eğer MySqLInd kurulu değilse bir çok Mysqli fonksiyonu çalışmayacaktır.

```
$veriTabani=@new mysqli('localhost','root','usbw','ogrenci');

 if( $veriTabani->connect_error)
   echo $veriTabani->connect_error;
          else{
               echo "VT Bağlantısı ve seçimi başarılı";

   if( function_exists('mysqli_stmt_get_result')===false)
       die ("Mysqli sürücü desteği yok. Bazı fonksiyonlar çalışmayacak");

               $veriTabani->close();
          }
```

function_exists('mysqli_stmt_get_result') fonksiyonu ile 'mysqli_stmt_get_result fonksiyonunun var olup olmadığına bakıyoruz. Eğer sürücü desteği yok ise bu sorgunun sonucu false olacaktır. Die fonksiyonu ise hata mesajı verip aşağıdaki kodların çalışmamasını sağlayacaktır.

HTML, PHP ve MYSQLI

Mysqli ile sorgu işlemlerini query() fonksiyonu ile yaptığımızda MYSQL de olduğu gibi yine SQL INJECTION saldırılarına karşı savunmasız oluyoruz. Bu sebeple yine önlem alıp her bir değişkeni tek tek real_escape_string() fonksiyonundan geçirmeliyiz.

```
$veriTabani=@new mysqli('localhost','root','usbw','ogrenci');

 if( $veriTabani->connect_error)
   echo $veriTabani->connect_error;
            else{
                 echo "VT Bağlantısı ve seçimi başarılı";

                    $kadi= $veriTabani-> real_escape_string($_POST['kadi']);
                    $sfr= $veriTabani-> real_escape_string($_POST['ksfr']);

$sql="SELECT * FROM `ogrencibilgileri` WHERE `kullaniciAdi`='$kadi' AND `sifre`='$sfr' ";

                    $sonuc= $veriTabani->query($sql);
                    if($sonuc->num_rows <=0)
                              echo "<h1> Giriş yetkiniz Yok</h1>";
                       else
                              echo "<h1> Yönetici Hoşgeldin</h1>";


                 $veriTabani->close();
            }
```

HTML, PHP ve MYSQLI

query() fonksiyonu ile yaptığımız sorgularda her değişkeni tek tek kontrol etmek gerektiği ve bunun her değişken için tek tek yapılmasının ne kadar zor olduğu aşikardır. Bu sebeple Mysqli de sorguları hazırladığımız ve sorgularda kullanacağımız dışarıdan form yoluyla gelen verileri kontrol ettiğimiz bir fonksiyon grubu mevcuttur. Bu fonksiyon grubunu kullanarak daha güvenli sorgular gerçekleştirebiliriz.

prepare() fonksiyonu göndereceğimiz sorgunun doğru olup olmadığını kontrol eder ve doğru ise true değilse false değeri döndürür. Bir anlamda sorgumuzun hazır olup olmadığını kontrol etmemizi sağlar. Bind_param() ve execute() fonksiyonları ile birlikte kullanılır.

Bind_param() fonksiyonu dışarıdan gelen değişkenleri alıp kontrolden geçirip, prepare() fonksiyonundaki sorguya bağlamak için kullanılır. Yani bu fonksiyondan geçirdiğimiz değişkenler zararlı karakterlerden temizlenmektedir. Böylece artık real_escape_string() gibi ekstra işlemlerle uğraşmak zorunda kalmıyoruz.

execute() fonksiyonu ise prepare() ve bind_param() fonksiyonları ile hazırladığımız fonksiyonları çalıştırmamızı sağlar. Çalıştırılan sorgunun get_result() fonksiyonu ile sonuçları alırız.

HTML, PHP ve MYSQLI

```
$veriTabani=@new mysqli('localhost','root','usbw','ogrenci');

 if( $veriTabani->connect_error)
   echo $veriTabani->connect_error;
              else{
                   echo "VT Bağlantısı ve seçimi başarılı";

                              $kadi= $_POST['kadi'];
                              $sfr= $_POST['ksfr'];
                               $sql="SELECT * FROM `ogrencibilgileri` WHERE `kullaniciAdi`=? AND `sifre`=? ";
                               $sorguHazirla = $veriTabani->prepare($sql);//sorguyu hazılıyoruz
                              if(!$sorguHazirla)
                                     die("Sorgu Hazırlanırken Hata Oluştu.".$veriTabani->error);
                              else{   /*Sorgu Başarıyla Hazırlandı Şimdi Dışardan Gelen Parametreleri Bağlayalım*/
                                       $sorguHazirla->bind_param('ss',$kadi,$sfr);
                                       /*Şimdi Sorguyu Çalıştıralım*/
                                       $sorguHazirla->execute();
                                       /*Sorgu Sonucunu Alalım*/
                                       $sonuc=$sorguHazirla->get_result();
                                       $sorguHazirla-> close(); /*Sorguyu bitirip bellekten silelim */

                                       if($sonuc->num_rows <=0)
                                                echo "<h1> Giriş yetkiniz Yok</h1>";
                                       else
                                                echo "<h1> Yönetici Hoşgeldin</h1>";
                                  }
                         $veriTabani->close();
                   }
```

HTML, PHP ve MYSQLI

Şimdi veritabanından SELECT sorgusu ile veri listeleme yapalım. Bundan sonra veritabanına bağlantı işlemlerini yazmayacağım onlar aynı zaten.

```
$sql="SELECT * FROM `ogrencibilgileri`";
$sorguHazirla = $veriTabani->prepare($sql);//sorguyu hazılıyoruz
if(!$sorguHazirla)
  die("Sorgu Hazırlanırken Hata Oluştu.".$veriTabani->error);
else{       $sorguHazirla->execute(); /*Şimdi Sorguyu Çalıştıralım*/
            $sonuc=$sorguHazirla->get_result(); /*Sorgu Sonucunu Alalım*/
        $sorguHazirla-> close(); /*Sorguyu bitirip bellekten silelim */
        if($sonuc)
        {
          echo"<h2>Veritabanından ". $sonuc->num_rows." kişi seçildi</h2>";
          $metin="<table border='5' cellpadding='10px'>";
          while($dizi= $sonuc-> fetch_array( MYSQL_ASSOC) ) {
                $metin .="<tr><td>";
                $metin .=implode("</td><td>",$dizi);
                $metin .="</td></tr>";
                                                    }
           echo $metin ."</table>";
          }
    }
```

Veritabanından 5 kişi seçildi

Mutlu YAPICI	Happy	12345	12345678909
ener G kt rk	S t	A k pek	12345678994
Mehmet	Mehmet0619	12345	12355678909
	S	A	13345678994
?eng l G kt rk	S ?t z m	A??k?pek	14445678994

HTML, PHP ve MYSQLI

Tablodaki Türkçe karakter sonunu çözmek için veritabanının karakter türünü utf-8 yapalım. Bunun için $veriTabani->set_charset('utf8') fonksiyonunu kullanacağız

```
$sql="SELECT * FROM `ogrencibilgileri`";
$veriTabani->set_charset('utf8');
$sorguHazirla = $veriTabani->prepare($sql);//sorguyu hazılıyoruz
if(!$sorguHazirla)
  die("Sorgu Hazırlanırken Hata Oluştu.".$veriTabani->error);
else{       $sorguHazirla->execute(); /*Şimdi Sorguyu Çalıştıralım*/
            $sonuc=$sorguHazirla->get_result(); /*Sorgu Sonucunu Alalım*/
        $sorguHazirla-> close(); /*Sorguyu bitirip bellekten silelim */
        if($sonuc)
        {   echo"<h2>Veritabanından ". $sonuc->num_rows." kişi seçildi</h2>";
            $metin="<table border='5' cellpadding='10px'>";
            while($dizi= $sonuc-> fetch_array( MYSQL_ASSOC) ) {
                $metin .="<tr><td>";
                $metin .=implode("</td><td>",$dizi);
                $metin .="</td></tr>";
                                                    }
             echo $metin ."</table>";
            }
    }
```

Veritabanından 5 kişi seçildi

Mutlu YAPICI	Happy	12345	12345678909
Þener Göktürk	Söðüt	AþýkÝpek	12345678994
Mehmet	Mehmet0619	12345	12355678909
	S	A	13345678994
Şengül Göktürk	Söğütözüm	AşıkİpeK	14445678994

HTML, PHP ve MYSQLI

Veritabanına gelen formdaki verileri ekleme işlemi.

```
if( !empty($_POST['adi']) && !empty($_POST[‘kadi']) && !empty($_POST[‘ksfr']) && !empty($_POST[‘tcno']) ) {
        extract($_POST);//gelen verileri değişkenlere çıkarttık
        //Önce SQL Sorgusunu Oluşturalım
        $sql="INSERT INTO `ogrencibilgileri` (`adi`, `kullaniciAdi`,`sifre`,`tcNo`) VALUES(?,?,?,?)";
        $veriTabani-> set_charset('utf8');//karakter türünü belirledik
        $sqlHazirla=$veriTabani->prepare($sql);//Sorguyu Hazırladık

      if(!$sqlHazirla){
              die("Veri Ekleme Sorgu Hazırlama Hatası".$veriTabani->error);
        }else{///sorgu hazırsa
              $sqlHazirla->bind_param('sssi', $adi, $kadi, $ksfr, $tcno);///parametreleri bağlayalım
              $sqlHazirla->execute();///Sorguyu Çalıştırdık

              if($sqlHazirla){
                    echo "<h2>".$sqlHazirla->affected_rows." tane kayıt eklendi</h2>";
                  }else {
                    die("Veri Ekleme Hatası".$veriTabani->error);
                        }
            }
}else
    echo "<h2> Veritabanına Eklemek İçin Tüm Verileri Doldurun</h2>";
```

Öğr. Gör. M. Mutlu YAPICI

HTML, PHP ve MYSQLI

Bir önceki sunumda mysqli fonksiyonlarını kullanarak veritabanına ekleme işlemini gerçekleştirdik. Veritbanınadan güncelleme ve silme işlemleri ekleme işlemiyle bire bir aynıdır tek farkı sadece SQL sorgusudur. SQL sorgusunu değiştirdiğimiz zaman aynı kodlarla sileme ve güncelleme de yapabiliriz. 6. sunumda mysql ile silme ve güncelleme işlemlerini yaptığımız için burada tekrar anlatmayacağım. Derste de ayrıca örneklerle zaten bu işlemleri de yaptık.

Veritabanında ARAMA ve veritabanı bilgilerini SAYFALAMA işlemlerini dersimizde örnek olarak yapacağız. Sunumlarda tüm kodları tek tek yazmak istemiyorum. Bu tür işlem kodları PHP den ziyade veritabanı ve algoritma bilgisi gerektirmektedir. Bu tür işlemleri önerdiğim kitap ve kaynaklarda ve de internette rahatça bulup inceleyebilirsiniz. Eğer veritabanını PHP ile etkin kullanmak istiyorsanız kesinlikle bu tür örnekler yapmanızı öneririm.

Şimdi biz konularımıza **veritabanında güvenlik** kısmı ile devam edelim. Veritabınında işlemlerini gerçekleştirilmesinin yanında en önemli konulardan birisi güvenliktir. Veritabınımızdaki bilgilerin kötü niyetli insanların eline geçmesini istemiyorsak her türlü kod açığını düşünüp kapatarak ilerlemeliyiz. Daha önceki sunumlarda da gösterdiğim gibi en basit veritabanı saldırılarından biri **SQL INJECTIONS** dediğimiz SQL cümleciklerinin kötü amaçlı olarak kodlarımızın arasına eklenmesidir. Eğer önlem almazsanız bu sorun **MYSQL** de olduğu gibi **MYSQLI** 'de de devam etmektedir.

HTML, PHP, MYSQLI ve Güvenlik

Veritabanına gelen formdaki verilerle üye girişi işlemi.

```
if( !empty($_POST['adi']) && !empty($_POST['sfr']) ) {
        extract($_POST);//gelen verileri değişkenlere çıkarttık
        //Önce SQL Sorgusunu Oluşturalım
        $sql="SELECT * FROM `ogrencibilgileri` WHERE `adi`='$adi' AND `sube`='$sfr'";
        $veriTabani-> set_charset('utf8');//karakter türünü belirledik
        $sorgu=$veriTabani->query($sql);//Sorguyu Hazırladık

      if(!$sorgu){
                die("Veri Ekleme Sorgu Hatası".$veriTabani->error);
        }else{///sorgu  doğruysa
                if(!$sorgu->num_rows>0)
                 echo"<h1>Yönetici Girişi Başarılı</h1>";
                else
                 echo"<h1>Giriş Başarısız</h1>";
            }
}
<form action="formdeneme.php" method="POST" name="frm">
        Kullanıcı Adınız: <input type="text" name="adi"/>
        Kullanıcı Şifreniz: <input type="password" name="sfr"/>
       <input type="submit" value="Giriş" name="btn"/>
 </form>
```

HTML, PHP, MYSQLI ve Güvenlik

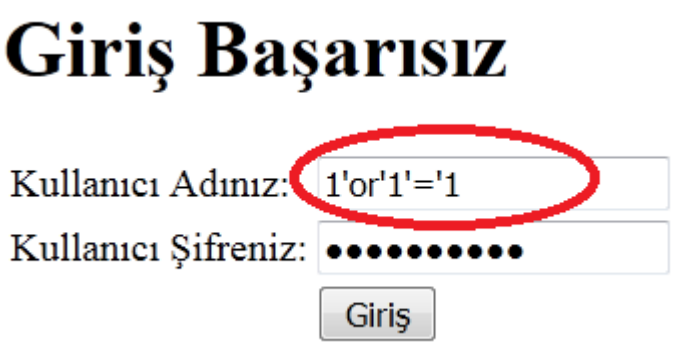
Bir önceki sunumda bulunan kodları kullandığımızda aşağıdaki gibi bir form gelecektir ilgili veritabanımızdaki ilgili tablodaki bilgilere göre kullanıcı adı ve şifre bilgilerini doğru girersek Yönetici Girişi Başarılı yazısını görürüz ancak yanlış girersek Giriş Başarısız yazısını görürüz. Daha önce de gösterdiğimiz SQL INJECTION saldırısını uygularsa yani kullanıcı adı ve şifresi yerine 1'or'1'='1 yazarsak yine yönetici girişi yapıldığı aşikardır. Bunu önlemek için yine mysqli de de bulunan real_escape_string fonksiyonunu kullanarak kontrolü sağlayabiliriz. Yada daha önceki sayfalarda kullandığımız prepare, bind ve execute fonksiyonlarını kullanmalıyız.

HTML, PHP, MYSQLI ve Güvenlik

MYSQLI de real_escape_string kullanımı;
$adi= $vt->real_escape_string($_GET['adi']); şeklinde her değişkeni tek tek kontrol edebiliriz.

İsterseniz diziler konusunda işlediğimiz ARRAY_MAP fonksiyonunu kullanarak daha kolay ve hızlı bir şekilde GET veya POST ile gelen tüm verileri real_escape_string fonksiyonundan geçirebiliriz.

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



HTML, PHP, MYSQLI ve Güvenlik

Veritabanında bulunan verilerin diğer bir güvenliği de şifreleme algoritmalarıyla sağlanmaktadır. Veri tabanına kaydettiğiniz verileri özellikle de şifre verileriniz çeşitli algoritmalarla şifreleyerek kaydederseniz, hackerlar tarafından bilgiler çalınsa dahi çözülmesi zor hatta imkansız olacaktır. Çeşitli veri şifreleme algoritmaları aşağıdaki gibidir.

Şifreleme Fonksiyonu	Açıklama
ENCODE(veri, anahtar)	Veriyi anahtara göre şifreler.
DECODE(veri, anahtar)	Şifreli veriyi anahtarı kullanarak çözer
PASSWORD(veri)	Veriyi geri dönüşü olmayacak şekilde şifreler
MD5(veri)	Veriyi geri dönüşü olmayacak şekilde şifreler
SHA1(veri)	Veriyi geri dönüşü olmayacak şekilde şifreler 40 haneli veri oluşur

ENCODE, DECODE ve PASSWORD, PHP nin üst versiyonlarında çalışmamaktadır.
Diğerlerinin çıktısı aşağıdaki gibidir.

```
$veri='Merhaba Dünya';
Echo MD5($veri);
echo SHA1($veri);
```

3c5cee1b19a1ca2a0ae50cb42c701a85

69f896ab6c48f74e6745acecaf7ca63ff4553db5

KAYNAKLAR

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