
# PHP Günleri 2013#1

## mysql_* Fonksiyonları Ömrünü Doldurmak Üzere. Peki Şimdi Ne Olacak?

Özgür Yazılım A.Ş.
www.ozguryazilim.com.tr

# Adil İlhan

## Yazılım Geliştirici – Özgür Yazılım A.Ş.


@adil_ilhan


www.adililhan.com


adil.ilhan@ozguryazilim.com.tr

# PHP ve MySQL
# Bitirim İkili

- PHP denilince ilk akla gelenlerden birisi MySQL.
- Yıllarca isimleri birlikte anıldı.
- Yıllarca mysql_* fonksiyonları kullanıldı ve hâlâ kullanılıyor.

# mysql_* dan kasıt nedir?

mysql_connect,

mysql_query,

mysql_fetch_assoc,

mysql_fetch_array vb.

# Ama artık mysql_* fonksiyonları ömrünü doldurmak üzere!

# Madem ömrü doldu neden hâlâ kullanılıyor?

- Alışkanlık!
  - PHP4'ten beri mysql_connect var.
- Yeniliklere ayak uyduramama
  - “Teknoloji gelişiyor” lafı PHP için de geçerli.
- Sunucularda eski PHP sürümleri var
  - Sunucudaki sürüm eski olunca, güncel fonksiyonlar kullanılamıyor.

# Alternatif nedir?

Şu an mysql_* fonksiyonları için iki iyi alternatif var:

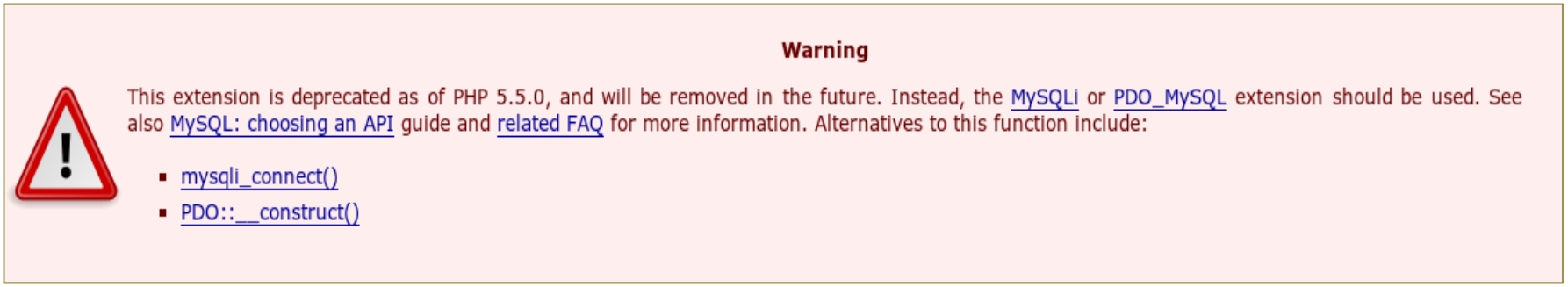
1) PDO (Php Data Objects)

2) mysqli (MySQL Improved Extension)

# Peki alternatifleri var ama, mysql_* fonksiyonları neden artık kullanılmamalı?

# Artık deprecated (önerilmemektedir)

- PHP geliştiricileri mysql_* fonksiyonlarını “resmi” olarak önermiyor.
- Geliştirilmesi durdu.
- 5.5.0'da deprecated olacak (önerilmemeye başlanacak).



**Warning**

This extension is deprecated as of PHP 5.5.0, and will be removed in the future. Instead, the MySQLi or PDO_MySQL extension should be used. See also MySQL: choosing an API guide and related FAQ for more information. Alternatives to this function include:

- mysqli_connect()
- PDO::__construct()

# Prepared statement (hazır deyim) özelliği yok!

Prepared stament ne ola ki?

# Prepared Statement'lar (Hazır Deyimler)

- Daha performanslı
  - Birçok kez çalışacak sorguyu önce derle, sonraki sorgularda daha performanslı çalışsın.
- Daha güvenli
  - mysql_real_escape_string ve türevlerinden kurtuluş
  - Tırnaklardan sorguyu korur.
- MySQL 4.1'den beri destekliyor.
- Yazımı daha kolay ve okunaklı (Bana göre:))

# Örnek yazıma geçmeden önce...

- MySQLi'de MySQL veritabanına bağlanmak:

```
$db = new mysqli('localhost', 'root', '123', 'seminer');
$db = mysqli_connect('localhost', 'root', '123', 'seminer');
```

- PDO'da MySQL veritabanına bağlanmak:

```
$db = new PDO('mysql:dbname=seminer;host=localhost;', 'root', '123');
```

# Örnek yazımı şöyle:

**$db->prepare('INSERT INTO seminer (ad, soyad) VALUES (:adi, :soyadi)');**

(Bu kullanım şu an sadece PDO'da var.)

## Bir diğeri:

**$db->prepare('INSERT INTO seminer (ad, soyad) VALUES (?, ?)');**

(Bu kullanımı PDO ve Mysqli destekliyor.)

# Parametre Bağlamak (Bind Param)

**İsimli parametreler PDO'da var.**

```
$q = $db->prepare('INSERT INTO seminer (ad, soyad, yas) VALUES (:adi, :soyadi, :yasi)');
$q->bindParam(':adi',$ad);
$q->bindParam(':soyadi',$soyad);
$q->bindParam(':yasi',$yas);
```

# Parametre Bağlamak (Bind Param)

**Soru işaretli değer atamalar hem PDO'da hem MySQLi'de var.**

Mysqli'de

```
$st = $db->prepare('INSERT INTO seminer (ad, soyad, yas) VALUES (?, ?, ?)');
$st->bind_param('ssi', $ad, $soyad, $yas);
```

PDO'da

```
$st->execute(array($ad, $soyad, $yas));
```

# Böylece ne elde ettik?

- Daha güvenli sorgu çalıştırma imkanı
  - mysql_real_escape_string vs. yeteri kadar iyi değiller. Ekstra kod yazmak gerekiyor.
  - Mysqli ve PDO'da bind param ile güvenlik işini PHP'ye bırakıyoruz.

# Böylece ne elde ettik?

- mysql_* fonksiyonlarından daha hızlı bir sorgu altyapısı.
  - MySQL, sorguyu ilk seferinde analiz edip, her defasında aynı işlemleri tekrarlamıyor. Parametrelere uygun olarak tampondan veriyi sunuyor.

# mysql_* fonksiyonlarının yeteneksizliği sadece bu kadar mı?

Tabii ki hayır :)

# Doğrudan Transaction Desteklenmiyor

mysql_* fonksiyonları doğrudan transaction yapmaya olanak sağlamıyor.

Ekstra fonksiyonlar yazılarak, veritabanına commit, rollback işlemleri yaptırılıyor.

# PDO ve Mysqli Varsayılan Olarak Transaction Destekliyor

- PDO'da ve Mysqli'de yazılımcının kolayca transaction yapabilmesi için gerekli metodlar sunuluyor.

mysql_* fonksiyonlarındaki gibi her rol (start, commit, rollback) için ekstra tanımlama zahmetine girilmiyor.

# mysql_ Fonksiyonları Tek Seferde Birden Fazla Sorgu (Multiple queries Yapamıyor

- ```
  $sql = "SELECT COUNT(id) FROM seminer;
          INSERT INTO seminer (ad, soyad) VALUES ('Adil', 'İlhan');
          SELECT COUNT(id) FROM seminer;";
  ```

- Bu sorguyu tek seferde mysql_ fonksiyonları işleyemiyor. Bu sorguyu ancak PDO veya Mysqli ile çalıştırabilirsiniz.
  ```

# mysql_* fonksiyonları OOP arayüzü desteklemiyor

- OOP mimarisine uygun kod yazdınız ama mysql_connect ile veritabanına bağlandınız.
  - Metodlara erişirken -> kullanıyorsunuz ama veritabanı işlemleri için alt çizgi (underscore) kullanmak zorundasınız.
  - Göz yorucu değil mi? :)

# Sürekli PDO ve Mysqli diyorsun da hangisini seçmek lazım?

# İkisi de iyi, ama aralarında ufak tefek farklar var. O farklarla çok iyi oldu çok da güzel oldu.

- PDO ile 12 farklı veritabanına bağlanabilirsiniz.
- PDO ile isimli parametre ataması da yapabilirsiniz.
- PDO'da sadece OOP kullanabilirsiniz.
- Mysqli'de hem OOP hem procedural kullanılabilir.

İyi güzel de bu fonksiyonlara göç etmek o kadar kolay değil ki...

Acı gerçek... :(

- Piyasada hâlen birçok projede mysql_* fonksiyonları kullanılıyor. Bunların bir şekilde göç etmesi gerekiyor.

- Göç yapılmadığı takdirde her yeni sürümde kodunuz güncel teknolojilerden daha da geri de kalacak.

- Ancak iş dünyasında bu tarz değişimler yapmak oldukça zor. Bunun başlıca sebeplerinden birisi de “vakit olmaması”. Vakit yetersiz olduğu için gerekli değişimler yapılamıyor. Yapılsa bile birçoğunun unit test'leri yapılmıyor.

mysql_* fonksiyonlarını avcumun içi gibi biliyorum. Yeni fonksiyonları öğrenmek bir sürü iş çıkaracak.

Evet, doğru bir öngörü.
Yeni fonksiyonlar iş çıkaracak.

Ancak yeni sürümler çıktıkça, kodunuzu güncellemediğiniz takdirde kodunuz geride kalacak.

Bu ileride daha çok zahmet oluşturacak.

# Peki nasıl göç edeceğiz?

- Eğer sorguları bir arayüz üzerinden yapıyorsanız işiniz nispeten daha kolay.
  - Arayüz?

```
$this->db->select('ad');
$this->db->from('seminer');
$this->db->where('id', 3);
$q = $this->db->fetch();
```

  - Ya arayüz yoksa?
    - O zaman işiniz zorlaşacaktır. :(

mysql_* fonksiyonlarını güncel fonksiyonlara göç ettiren bir betik olsa ne güzel olurdu...

# Sıkmayın canınızı o da var :)

- Oracle, bu göç problemi için bir betik yayınladı.
- Şu adreste bulabilirsiniz: http://bit.ly/ttklS8
- Betik sayesinde mysql fonksiyonlarını mysqli fonksiyonlarına çevirebiliyorsunuz.
- Betik oldukça başarılı.
- Örneğin; WordPress'i dönüştürünce birkaç warning dışında sistem eskisi gibi çalışıyor.

# Sorularınız?

Adil İlhan

Yazılım Geliştirici – Özgür Yazılım A.Ş.

www.adililhan.com

www.twitter.com/adil_ilhan

www.linkedin.com/in/adililhan