

MVC

“Zeki çocuklar için” Serisi

Altan TANRIVERDİ

<http://javam.org>

Başlarken

MVC (Model-View-Controller) herhangi bir PHP yazılımcısının olmazsa-olmazları arasında bulunması gereken kodlama biçimidir. <http://en.wikipedia.org/wiki/Model-view-controller> adresi MVC'yi teorik olarak kavramak için iyi bir başlangıç olacaktır. Ancak biz işin pratiğine odaklanacağız ve kendimize ait oldukça basit bir framework yaratacağız. Burada söz konusu olacak kod ve terimleri anlamak için PHP5 ve OOP konusunda yeterli bilgiye sahip olmanız gerekmektedir.

Nedir bu M?, V? ve C?

M (Model)

M (Model), İş Mantığı (Business-Logic)¹ ve data işleme süreçlerini yürütür. C (Controller) tarafından gönderilen emirlere göre hareket eder. Bilgi işleme sürecinden sonra datayı C'ye, diğer modellere veya doğrudan V (View)'ye gönderir.

V (View)

V (View) son kullanıcıya gösterilecek olan datanın sunumu ile ilgilendir. V, bu bilgiyi C veya M'den alır, aynı zamanda son kullanıcıdan gelen talepleri C'ye iletir.

C (Controller)

C ise sistemin ana kısmıdır. Gelen talepleri kontrol eder ve sistemin diğer elemanlarının (M,V) bilgiyi uygun şekilde alıp, göndermelerini sağlar.

Kısacası, C beynimiz, M sinirlerimiz, V ise kas, burun, göz, kulak vb. kısacası dış dünya ile fiziksel ilişkisi olan herşeyimizdir.

Sevdiğimiz insanı gördüğümüzde gözümüz bu veriyi beynimize yollar (V -> C), beynimiz buna bir tepki vererek yüz sinirlerine emir gönderir (C -> M), sinirlerimiz beyinden gelen emri işleyerek kasları gerer ve gülümsememizi sağlarlar (M -> V). Tüm MVC süreci bununla özetlenebilir.

Şimdi sıra dünyanın en basit framework'ünü yazmaya geldi. Bu basit framework'te kullanıcıdan geldiğini varsaydığımız bir datayı güvenlik şemasından geçirerek ekrana basacağız. Böylece bir MVC framework'ünün nasıl işlediğine dair en basit ve temel bilgiyi edinmiş olacağız. İşe klasör yapımızdan başlayacağız ve belki serinin devam eden yazılarında bu framework yapısını birlikte geliştireceğiz.

¹ İş Mantığı bir veritabanı ve kullanıcı arayüzü arasında gerçekleşen bilgi değiş tokuşunu yöneten fonksiyonel algoritmaları temsil eder. (Kaynak: Wikipedia)

FrameWork

Adlandırma

Yazıya uygun şekilde klasörlerimizi de “m”, “v” ve “c” şeklinde adlandırmak en iyisi olacaktır. Bu klasörlerle birlikte index.php dosyamızı da sistemimize ekleyeceğiz:



index.php

Şimdi her birinin içerisine klasör adları ile ana PHP dosyalarını yükleyelim. “m” altına m.php, “v” altına v.php ve “c” altına c.php

ve index.php içerisine Model, View ve Controller ana dosyalarını yükleyelim:

```
<?php
```

```
require_once("m/m.php");  
require_once("v/v.php");  
require_once("c/c.php");
```

```
?>
```

Varsayalım kullanıcı id sini index.php dosyamızda bir form aracılığıyla gönderen kullanıcılarımız var ve biz bunu veritabanımızda işleterek oturum açıyoruz ve örneğimiz basit olsun diye bu datanın sadece numaralardan oluştuğunu ve form yazımını işe karıştırmamak için datanın post edilmiş olduğunu varsayacağız.

```
<?php
```

```
require_once("m/m.php");  
require_once("v/v.php");  
require_once("c/c.php");
```

```
$id = '123456789'; // Normal Kullanıcıdan gelen post değeri.  
$id = 'delete * from'; // Sisteme Saldıran Kullanıcıdan gelen post değeri.
```

```
?>
```

Bu dataları alt alta yazdım ancak ayrı ayrı varsayalım. Yani sistemi test ederken birinden birini kaldırın. index.php dosyamıza daha sonra dönüş yapacağız ama önce Controller dosyamızı oluşturmaya başlayalım.

c/c.php

Controller sınıf dosyamız, daha önce bahsettiğimiz gibi datayı işleyen değil Model'e datayı işlemlerini emreden dosyamız olacak. Bunun için yine örneğe uygun devam ederek “c” adlı bir sınıf yaratalım:

```
<?php  
class c {  
}
```

Böylece c sınıfını yaratmış olduk. Ek bilgi olarak sınıf dosyamızı kapatmadığımızı farketmişsinizdir. Bu PHP ve Zend Engine geliştiricileri tarafından dosya sonunda boş satırlar oluşmasını engellemek için include edilen dosyalar için önerilir.

Şimdi sınıfımıza M'ye emir verecek bir fonksiyon eklememiz gerekiyor:

```
<?php  
class c {  
    public function kullaniciya_guveniyormuyuz($id) {  
    }  
}
```

Şimdi de ilk emrimizi verelim. Bunu yaparken m.php içerisindeki sınıfa “m” adını vereceğimizi varsayalım:

```
<?php  
class c {  
    public function kullaniciya_guveniyormuyuz($id) {  
        $m = new m();  
        $cevap = $m->cevabi_soyle($id);  
    }  
}
```

Böylece M'ye emrimizi verdik. M değeri \$id üzerinde işlem yapacak, bir sonuç çıkaracak ve kullanıcıya güvenilir, güvenilmeyeceğini tespit edecektir. Daha sonra c.php dosyamıza döneceğiz. Şimdi m.php dosyamızı hazırlamaya başlayalım ve C'nin gönderdiği emri işleyelim.

m/m.php

```
<?php
```

```
class m {
    public function cevabi_soyle($id) {
        if (is_numeric($id)) {
            $cevap = "Evet kullanıcı güvenilir. Oturum açalım";
        } else {
            $cevap = "Sakın oturum açma, kullanıcı güvenilir değil";
        }
        return $cevap;
    }
}
```

Yukarıda C'nin emri üzerine kullanıcının girdiği id'nin numerik olup olmadığını kontrol ettik ve eğer numerik ise kullanıcıya güvenip, oturup açmayı, değilse işlemi yapmamayı söyledik. Tabi gerçek bir projede buralarda oturum, veritabanı işlemleri ve hata komutları verdiğimiz varsaymalısınız. Şimdi c.php dosyamıza geri dönelim ve M'nin bize verdiği cevabı değerlendirelim.

c/c.php

```
<?php
```

```
class c {
    public function kullaniciya_guveniyormuyuz($id) {
        $m = new m();
        $cevap = $m->cevabi_soyle($id);
        $v = new v();
        $v->cevabi_goster($cevap);
        $v->goster();
    }
}
```

Görüldüğü gibi şimdi de V'ye M'den gelen sonucu basma emri veriyoruz.

Son olarak V (View) dosyamızı yaratma zamanı geldi. Yukarıda v.php içerisinde oluşturacağımız gösterme sınıfının adını "v" varsaydık.

v/v.php

```
<?php

class v {
    private $id;
    public function goster() {
        echo $this->cevap;
    }
    public function cevabi_goster($cevap) {
        $this->cevap = $cevap;
    }
}
```

v adlı sınıfımızı oluşturduk ve M'den C'ye dönen cevabı bastık.

Geriye kalan tek şey index.php içerisinde Controller dosyasını çalıştırmak:

index.php

```
<?php

require_once("m/m.php");
require_once("v/v.php");
require_once("c/c.php");

$id = '123456789';           // Normal Kullanıcı.
$id = 'delete * from';     // Sisteme Saldıran Kullanıcı.

$c = new c();
$c->kullaniciya_guveniyormuyuz($id);

?>
```

Artık mini MVC framework'umuzu bitirdik, test edelim:

```
$id = '123456789';
```

için sonuç:

```
Evet kullanıcı güvenilir. Oturum açalım
```

```
$id = 'delete * from';
```

için sonuç:

```
Sakın oturum açma, kullanıcı güvenilir değil
```

Sanırım oldukça anlaşılır bir şekilde MVC yapısını inceledik. Fırsat oldukça MVC yapısını incelemeye devam edeceğiz. Eğer sorularınız olursa <http://javam.org> adresinden bize ulaşabilirsiniz.