İçindekiler

[1. .NET Core Teknolojisi 2](#_Toc164437993)

[1.1. .NET Kurulumu 2](#_Toc164437994)

[2. MVC Tasarım Deseni 2](#_Toc164437995)

[2.1. MVC Projesi Oluşturma 3](#_Toc164437996)

[2.2. Model Katmanı 3](#_Toc164437997)

[2.3. Controller Katmanı 3](#_Toc164437998)

[2.4. View Katmanı 4](#_Toc164437999)

[2.4.1 Razor View Motoru 4](#_Toc164438000)

[2.4.2. Action Metottan View'e Veri Aktarımı 4](#_Toc164438001)

[2.4.2.1 Veri Taşıma Kontrolleri İle Veri Gönderimi 4](#_Toc164438002)

[2.4.2.1.1 ViewBag İle Veri Gönderimi 4](#_Toc164438003)

[2.4.2.1.2 ViewData İle Veri Gönderimi 5](#_Toc164438004)

[2.4.2.1.3 TempData İle Veri Gönderimi 5](#_Toc164438005)

[2.4.2.1.3.1 TempData İle Başka Bir Action’a Veri Gönderimi 5](#_Toc164438006)

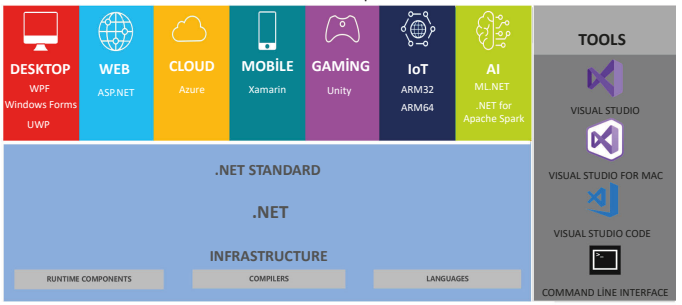
[2.4.2.2 Model Bazlı Veri gönderimi 6](#_Toc164438007)

[2. Yönlendirme (Routing) 6](#_Toc164438008)

# 1. .NET Core Teknolojisi

NET, platformdan bağımsız geliştirme yapılabilen (cross platform), yüksek performans sağlayan ve açık kaynak kodlu bir yazılım geliştirme ortamıdır.

.net platformu aşağıdaki gibidir.



**Desteklenen İşletim Sistemleri:** • Windows • macOS • Linux • Android • iOS • tvOS • watchOSÜ

**Desteklenen İşlemci Mimarileri:** • x64 • x86 • ARM32 • ARM64

**Desteklenen Programlama Dilleri:** • C# • Visual Basic • F#

## 1.1. .NET Kurulumu

NET’te uygulama geliştirme ve yazılan uygulamaları çalıştırabilmek için iki ayrı indirme seçeneği bulunur.

* **.NET SDK (Software Development Kit):** Uygulama geliştirme ve uygulamaları çalıştırmak için gerekli bileşenleri içerir.
* **.NET Runtime:** Uygulamaları çalıştırmak için gerekli bileşenleri içerir.

https://dotnet.microsoft.com/download adresinde, üzerinde çalışılmak istenen işletim sistemine göre güncel en son sürümler görüntülenebilir ve indirilebilir.

# 2. MVC Tasarım Deseni

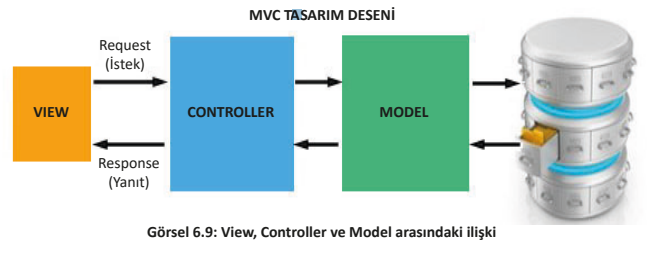
**Tasarım deseni, yazılımcıların karşılaştığı çeşitli sorunlar karşısında zaman içerisinde üretilen çözümleri ele alan standart çözüm yöntemleridir.** Tasarım desenleri; iyi bir tasarımın yapılması, yazılım esnekliğinin ve okunabilirliğinin arttırılması, sık karşılaşılan sorunlara çözüm sunması için deneyimlerin derlenmesiyle oluşturulmuştur.

Geliştirilen uygulamalar genellikle verilere erişim için gerekli kodların, uygulama mantığını oluşturan kod parçalarının ve kullanıcı arayüzlerinin birleşiminden oluşur ve **MVC (Model-View-Controller) tasarım deseni, uygulama geliştirme aşamasını bu nedenden dolayı üç parçaya ayırır.**

**Model:** Geliştirilen uygulamada kullanılacak verilerin özelliklerinin tutulduğu bölümdür. Örnek olarak kütüphane uygulamasında kitap bilgilerinin (adı, sayfa sayısı, vb.) özellikleri bu bölümde tutulur

**View (Görünüm):** Geliştirilen projede kullanıcının gördüğü arayüzlerin tasarlandığı bölümdür. Bu bölümde HTML, Javascript ve CSS kullanılır.

**Controller (Kontrolör):** Projedeki tüm hesaplamaların, veri tabanına erişimin ve diğer işlemlerin yapıldığı bölümdür. Uygulamadaki model ve view bölümleri arasındaki koordinasyonu sağlar.



## 2.1. MVC Projesi Oluşturma

**Visual studio code ile;**

Proje oluşturmak için vs code terminaline “**dotnet new MVC**” komutu yazılır.

* cd àKlasör değiştirme
  + cd .. à bir önceki dizine gider
  + cd klasörİsmi à klasör içinde verilen klasöre gider
* mkdir àKlasör oluşturma
  + mkdir MVC\_test àbulunan dizinde “MVC\_test” adında klasör oluşturur
* dotnet new MVC àbulunduğu dizine yeni bir dotnet projesi oluşturur
* dotnet run à MVC projesini çalıştırır
* “code .” ile oluşturulan proje visual studio içinde açılır
* C# extension ve Nuget package manager eklentilerinin eklenmesi önerilir

**Visual Studio ile;**

Asp .Net Core Web App ( Model – View – Controller ) seçilir

## 2.2. Model Katmanı

* MVC tasarım deseninde model katmanı, geliştirilen uygulamada kullanılacak olan verilerin özelliklerinin tanımlandığı katmandır
* Veri özelliklerini (Properties) tanımlarken her bir veri için ayrı ayrı sınıf (class) tanımlaması yapılmalıdır
* Model sınıfları projenin ana klasörü altındaki Models alt klasörü altında yer almalıdır
* Model katmanında yeni bir belge açılırken uzantısı “ .cs “ olmalıdır

## 2.3. Controller Katmanı

* MVC tasarım deseninde controller katmanı; içinde barındırdığı action metotları ile birlikte tarayıcıdan gelen sayfa isteklerini (URL) yöneten, view ve model katmanları arasındaki bağlantıyı sağlayan ve projenin iş sürecini kontrol eden katmandır.
* Sınıf isimlendirmesinde dikkat edilmesi gereken husus; sınıf adının “Controller” kelimesi ile bitmesi gerektiğidir

KitapController.cs adında yeni bir kontroller açıldığını varsayalım buradaki Index actionu Views altındaki Kitap klasörü içerisindeki “Index.cshtml“ adlı dosyayı döndürür. Aksi belirtilmediği sürece varsayılan view indeks adlı action metodudur.

Her bir action metodu;

• View (View katmanındaki ilgili dosya),

• Dosya (Resim, video, vb.),

• JSON (JavaScript Object Notation)

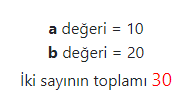
• string ve int gibi C# veri tipleri, gibi çeşitli tiplerde değer (ActionResult) döndürebilir ya da bir başka controller / actiona yönlendirme yapabilir.

## 2.4. View Katmanı

* MVC tasarım deseninde view katmanı, kullanıcının göreceği tüm arayüzlerin oluşturulduğu katmandır. Bu katmanda yer alan dosyalar .cshtml uzantılı olmalıdır.
* MVC tasarım deseninde viewlar, Views klasörü altında Controller adıyla aynı adlı bir alt klasör içerisinde yer almalıdır.
* Örneğin, HomeController sınıfı altındaki action metotları Views > Home klasörü altındaki View dosyalarını, KitapController sınıfı altındaki action metotları Views > Kitap klasörü altındaki View dosyalarını çağırır.

## 2.4.1 Razor View Motoru

* Razor View Motoru, view dosyaları içerisinde sunucu taraflı kodlar ile HTML kodlarının bir arada kullanımını sağlayan bir yapıdır. Razor view, @ karakteri ile başlayan ifadeleri sunucu taraflı çalıştırır ve HTML çıktısı oluşturur.



## 2.4.2. Action Metottan View'e Veri Aktarımı

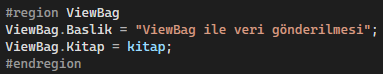
#### 2.4.2.1 Veri Taşıma Kontrolleri İle Veri Gönderimi

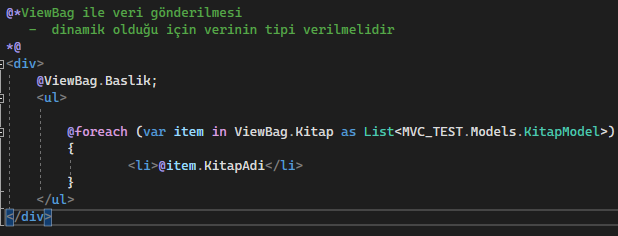
##### 2.4.2.1.1 ViewBag İle Veri Gönderimi

Controller içerisinde ViewBag ile üzerinden veriler dinamik (dynamic) olarak viewa aktarılabilir

Tanımlarken : ViewBag.[değişken] = değer

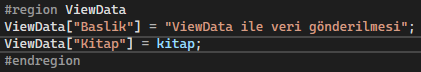
Kullanırken : @ViewBag.[değişken]

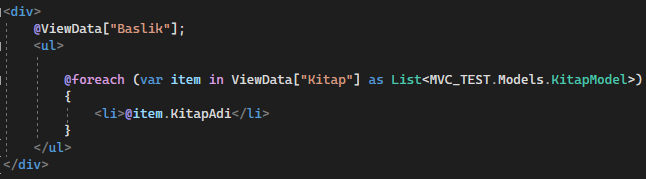




##### 2.4.2.1.2 ViewData İle Veri Gönderimi

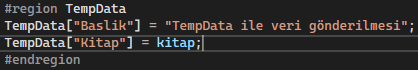
ViewBag’e benzer şekilde veriler viewa aktarılır. Ancak ViewData’da anahtar / değer ikilileri kullanılır:

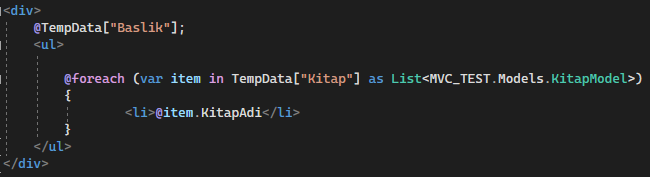




##### 2.4.2.1.3 TempData İle Veri Gönderimi

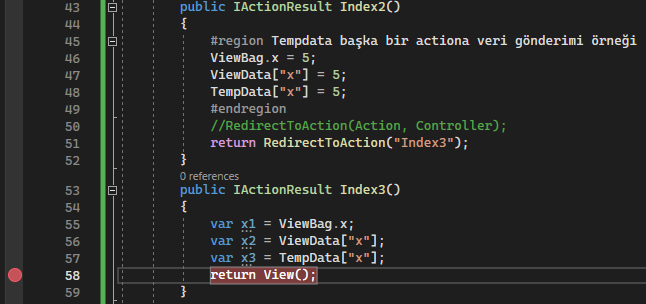
ViewData’ya benzer şekilde veriler viewa aktarılır. Ancak TempData’da başka bir actiona cookie ile veri gönderilebilir





###### 2.4.2.1.3.1 TempData İle Başka Bir Action’a Veri Gönderimi

Şekilde verilen görselde Index2 Çalıştırıldığında ilgili veriler oluşturulacak ve RedirectToAction ile Index3 çağrılacak. Index3 yüklendiğinde sadece TempData ile gönderilen veri geldiği diğer şekillerde “null” değeri döndüğü görülür.



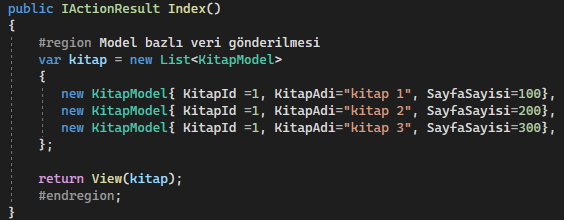
#### 2.4.2.2 Model Bazlı Veri gönderimi

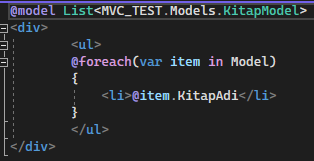
Var olan datayı viewe göndermek için hem controller hem view tarafında ayarlamalar yapmak gerekir.

Controller tarafında oluşturulan model bazlı veri view tarafına return ile gönderilmelidir --> return View(kitap);

View tarafında ise view’in model’ı tanıtılmalıdır. --> @model List<MVC\_TEST.Models.KitapModel>

View'e gelen data @Model ile kullanılır (M büyük önemli!!!)





# 2. Yönlendirme (Routing)

Yönlendirme, gelen isteklerin hangi controller ve hangi action metoda yönlendirileceğini belirleyen ASP.NET Core’daki önemli bileşenlerden birisidir

Web sitelerinde; gelen isteklerin URL yapılarının, arama motorlarının anlayabileceği, sayfa içeriğine uygun, anlamlı ve kullanıcılar tarafından okunabilir olması önemlidir. Yönlendirme konusu bu açıdan önemlidir.

Yeni bir MVC projesi oluşturulduğunda varsayılan olarak aşağıdaki şekilde bir yönlendirme oluşturulur:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Bu yönlendirme tanımında:**

• name=”default” ifadesi ile yönlendirme tanımına bir isim verilmiştir. İsteklerin eşleştirilmesinde hiçbir etkisi yoktur. Her yönlendirme tanımının ismi farklı olmalıdır.

• controller=Home ifadesi ile eğer controller ismi belirtilmezse yönlendirilecek varsayılan controller tanımlanır.

• action=Index ifadesi ile eğer action ismi belirtilmezse yönlendirilecek varsayılan action metot

tanımlanır.

• Sondaki id? ifadesi ise bu action metoda parametre gönderilmesini sağlar. “?” karakteri ile bu parametrenin seçimlik olacağı belirtilir.

metin, ekran görüntüsü, makbuz, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu