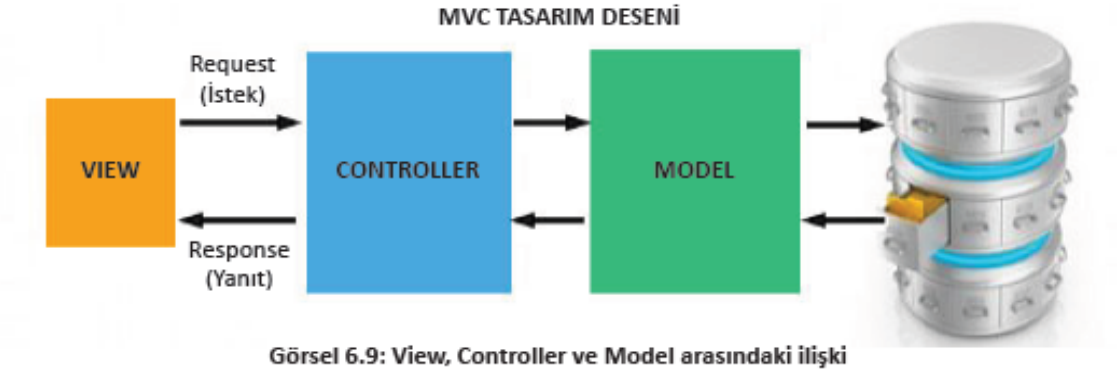
Geliştirilen uygulamalar genellikle verilere erişim için gerekli kodların, uygulama mantığını oluşturan kod parçalarının ve kullanıcı arayüzlerinin birleşiminden oluşur ve MVC (Model-View-Controller) tasarım deseni, uygulama geliştirme aşamasını bu nedenden dolayı üç parçaya ayırır.

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.

Model: Geliştirilen uygulamada kullanılacak verilerin özelliklerinin tutulduğu bölümdür. Örnek olarak kütüphane uygulamasında kitap bilgilerinin (adı, sayfa sayısı, vb.) özellikleri bu bölümde tutulur (Görsel 6.9).



Görsel 6.9: View, Controller ve Model arasındaki ilişki

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. Yeni bir model tanımlamak için **Models** klasörü seçili iken **New File** butonuna tıklanır ve modele bir isim verilir Model dosya uzantısının **.cs** olması zorunludur.

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.

Action Metotlar

Temel olarak controller sınıfında **public** olarak tanımlanan tüm metotlar **action metot** olarak isimlendirilir. Her bir controller sınıfı için **Index()** adlı metot varsayılan action metodudur. Yani action belirtilmediğinde çalışacak metot, **Index() 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.

View Katmanı

MVC tasarım deseninde view katmanı, kullanıcının göreceği tüm arayüzlerin oluşturulduğu katmandır. Bu katmanda yer alan dosyalar **.cshtml** uzantılı olmalıdır.
Bir önceki bölümde işlendiği üzere, bir controller bir veya birden fazla action metot bulundurabilir. Bir başka deyişle bir controller birden fazla view döndürebilir. 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.
Yeni bir view eklemek için **Views** klasörü altına, bu view dosyasıyla ilişkili controller sınıfı adıyla bir klasör oluşturulur. Oluşturulan klasör seçili iken **New File** butonuna tıklanır ve view dosyasına ilişkili action metodunun ismi verilir View dosya uzantısının **.cshtml** olması zorunludur.

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.

Standart Klasör ve Dosyalar

Views > Shared > _Layout.cshtml :

_Layout.cshtml dosyası, tüm sayfalar için ortak bir şablon yapısı tanımlamak için kullanılır. Bu dosya içerisindeki **@RenderBody()** yönergesi ile bu şablon dosyasını kullanan sayfaların içeriğinin görüntüleneceği kısım tanımlanır.

Program.cs
Her C# programında olduğu gibi programın başlangıç noktası public ve static erişim tiplerine sahip **Main()** metodudur. Main metodu **Program.cs** içerisinde yer alır ve web uygulamasını barındıran bir nesne oluşturur. Bu nesne, projelerin çalışmasını sağlar.

wwwroot
Proje içerisinde kullanılan her türlü statik (CSS, JS, resim, video, vb.) dosya, bu klasör altında bulunur. Bu klasöre eklenen dosyaları **https://localhost:5001/css/site.css** gibi bir link ile tarayıcıda görüntülemek mümkündür.

Properties
Bu klasör altında bulunan **launchSettings.json** dosyası aracılığıyla projenin çalışma zamanı ayarları değiştirilebilir. Örneğin, projenin 5001 No.lu portta değil de başka bir portta çalıştırılması istenirse bu dosyadaki ayarlar değiştirilmelidir.

bin
Proje derlendiğinde oluşturulan dosyalar **bin** klasörü altında bulunur.

Views > _ViewStart.cshtml
Her sayfanın gösterilmesinden önce çalıştırılması istenen kodlar bu dosya içerisinde bulunur.

Views > _ViewImports.cshtml
Projedeki tüm sayfalar için gerekli isim uzaylarını kullanmak gibi tüm sayfalar için ortak olan yönergeler bu dosya içerisinde tanımlanır.

Startup.cs
Startup.cs dosyası, .NET projelerindeki en önemli dosyalardan biridir. İçerisinde iki adet metot bulunur. Bu metotların önemli işlevleri şunlardır:
• **ConfigureServices**
Projede kullanılacak hizmetleri eklemek ve yapılandırmak için kullanılır. Proje ilk çalıştırıldığında öncelikle **ConfigureServices** metodu çağrılır.

services.AddControllersWithViews() ifadesi ile projede MVC tasarım deseni kullanılacağı bildirilmektedir. Ayrıca projede; veri tabanı, çerez (cookie), lokalizasyon (localization), oturum (session) gibi servisler kullanılması gerektiğinde bu servisler, **ConfigureServices** metodunda yapılandırılmalıdır.

Action Metottan View'e Veri Aktarımı

Bir action metottan view'e birkaç yöntemle veri aktarılabilir.
ViewBag Kullanarak: Controller içerisinde ViewBag ile üzerinden veriler dinamik (dynamic) olarak viewa aktarılabilir:

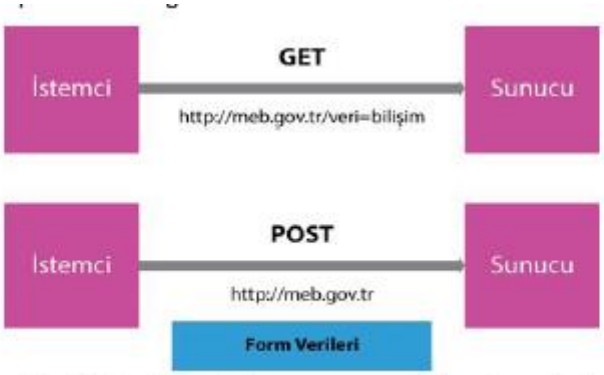
ViewData Kullanarak: ViewBag'e benzer şekilde veriler viewa aktarılır. Ancak ViewData'da anahtar / değer ikilileri kullanılır:

TempData Kullanarak: Kullanımı ViewData ile aynıdır. ViewData'dan tek farkı başka bir action metoda yönlendirme durumunda verilerin kullanılabilir olmasıdır.

Form GET Metodu Kullanımı
Form içerisinden girilen bilgiler GET metodu kullanılarak gönderilmek istendiğinde, form içindeki input elemanlarının name değerleri parametre; input içine girilen bilgiler ise parametrelerin değerleri olarak adres satırından query string olarak gönderilir. Form içinde bulunan buton tıklandığında veri gönderim işlemi başlatılır. Form içerisinden GET metodu ile gönderilen veriler **HttpGet** niteliğine sahip action tarafından alınarak işlenir.

Not

Controller içindeki actionlara herhangi bir nitelik verilmediği takdirde, ön tanımlı olarak HttpGet niteliğine sahip olurlar.



Görsel 6.29: GET ve POST metotları veri iletim yöntemleri

veriler GET metodu ile adres satırından gönderilirken POST metodunda ise HTTP istekleri içerisinde paketlenerek gönderilmektedir.

Form POST Metodu Kullanımı

POST metodunda formdan alınan bilgiler sunucuya tarayıcı tarafından Form Data **olarak gönderilir. GET metodunun aksine Form Data bilgileri adres çubuğunda görünmez. Gönderilen bu bilgiler; ASP.NET Core uygulamalarında veri tabanına kaydetme, e-posta olarak gönderme, işlem yapma vb. amaçları doğrultusunda kullanılır. Form içerisinden POST metodu ile gönderilen veriler** HttpPost **niteliğine sahip action tarafından alınarak işlenir.**
POST metodu ile Controller'a veri gönderme farklı yollardan yapılabilir. Bu yollar;

- **Parametre kullanma,**
- **IFormCollection sınıfı kullanma,**
- **Model Binder (Model Bağlama) kullanmadır.**

Model Binding; **formdaki input elemanların name değerleri aynı olan bir modelle eşleştirme veya bağlama işlemidir. Oluşturulan model sınıfının özellik isimleri ile input elemanlarının name değerleri aynı olmalıdır. Model sınıfından oluşturulan nesne ile formdan gönderilen verilere erişim sağlanarak arka uç tarafında işlenir.**