Asp.net Nedir ?

ASP.NET, Microsoft tarafından geliştirilmiş bir web uygulama gelişimi teknolojisidir. Dinamik web sayfaları, web uygulamaları ve XML tabanlı web hizmetleri geliştirilmesine olanak sağlar. Microsoft tarafından geliştirilen .NET Framework'un bir parçasıdır. Bir masaüstü uygulamasından bir web tarayıcı uygulamasına kadar her şey bu platform içinde düşünülmüştür ve desteklenmiştir.

Her ne kadar isim benzerliği olsa da ASP.NET, ASP'ye oranla çok ciddi bir değişim geçirmiştir. ASP.NET kodu ortak dil çalışma zamanı (ingilizce - common language runtime) altyapısına dayalı çalışır, diğer bir deyişle, yazılımcılar .Net çatısı tarafından desteklenen tüm dilleri ASP.NET uygulamaları geliştirmek için kullanabilirler. Yani, Java teknolojisinde olduğu gibi, yazılımcı tarafından yazılan kod, çalıştırılmadan önce sanal bir yazılım katmanı tarafından ortak bir dile çevirilmektedir.

Asp.net ile bir web uygulaması hazırlamak için .net dillerinden (C#, VB.net..) birisi kullanılabilir. .Net platformunun nesneleri Asp.net içerisinde de kullanılabilir. Asp.net ile bir web uygulaması geliştirmek için Visual Studio kullanılır.

Asp.net Web Form' un Temel Özellikleri

Asp.net code behind denilen tasarım öğeleri ile programlama öğelerini bir birinden ayrı sayfalarda bulunduran bir özelliği destekler. Bu özellik sayesinde bir proje üzerinde tasarımcı ve programcı aynı anda çalışabilir.

Sayfanın tasarım öğeleri (web forms için) .aspx uzantılı dosyada bulunurken program nesneleri.aspx.cs (Csharp için .cs, VB.net için .vb) uzantılı dosyalarda bulundurulur.

Asp.net' te Web Forms veya MVC programlama modellerini kullanarak uygulamalar geliştirebilirsiniz.

Microsoft .Net platformunun tüm özellikleri Asp.net uygulamaları için geçerlidir.

Visual Studio içerisindeki web kontrolleri (buton, textbox, dropdown list vs..) sürükle-bırak özelliği kullanarak hızlı ve basitçe uygulamaya dahil edilebilir.

Modern web uygulama arayüzleri oluşturabilmek için MasterPage ve Theme yapıları mevcuttur.

Asp.net' te gelişmiş oturum yönetimi desteği mevcuttur. Bu sayede üyelik işlemleri basit ve güvenli olarak yapılabilir.

ASP.NET Formlarının Dezavantajları

ViewState : ASP.NET formlarında ViewState güzel ve sempatik işlevleri olmasına karşın kötü bir tarafı var. Çok şişman. Şöyle: Geleneksel ASP.NET sayfalarında form, kullanıcı bilgileri ve durumları tutması ve postback anında kaybolmasını View State ile engelliyorduk. ViewState bu bilgileri base64 formatında şifreleyerek tutuyor. (ViewState kod oluşturma yapısı) Bu durum kullanıcı sayfayı ziyaret ettiğinde ve her bir istekte bulunuşunda onlarca hatta yüzlerce kilobyte'lık verinin oluşturulması ve HTML çıktısı sayfaya eklenmesi anlamına geliyor ve sayfayanın boyutunu artırıyor. Boyutu artan sayfa sunucuya gitmekle kalsa birde sunucudan tekrar bize geri dönüyor. (Postback) Her defasında hem sunucu tarafında ciddi bantgenişliği harcarken, kullanıcı tarafında ise sayfanın açılışı ciddi yavaşlamalara sebep oluyor.

Page Life Cycle (Sayfa Yaşam Döngüsü) : Bu yapı kullanıcı tarafı (client-side) ile server tarafı (server-side) arasında event'lerin ilişkilerini düzenleyen bir mekanizmadır. Request-response dediğimiz yapı yani kullanıcı tarafından gönderilen istek ve sunucu tarafından bu isteğe verilen cevap arasındaki ilişkileri düzenleyen bir yapıdır. Gelin görünki bu yapı oldukça karmaşıktır. Page life cycle'a müdahale edebilme şansımız var fakat bunu yaptığımızda kötü sürprizlerle karşılaşabilme ihtimalimiz oldukça yüksek. Öyle ki bazen ortaya çıkan hataların neden çıktığını bulamayabilirsiniz.

Serseri HTML Çıktıları : Sunucu taraflı (server-side) diller HTML çıktısı verirler bunu bilmeyenimiz yoktur. ASP.NET sayfalarında sunucuda yorumlandıktan sonra HTML çıktısı olarak bize gelir. Fakat burada hakimiyetimiz sınırlıdır yani istediğimiz gibi bir HTML sayfasını çıktı olarak alamayabiliriz, görüntü olarak istediğimiz HTML dosyasını alsak dahi kod temizliği ve W3 standartlarına uygun HTML çıktısı almak oldukça zor bir hal alıyor ve ortaya kendi halinde, bizim belirlediğimiz disiplinden uzak, serseri HTML sayfaları oluşabilir. Özellikle CSS ve javascript işin içine girince hayalkırıklığımız artabiliyor. ASP.NET 4.x sürümü ile bu sorun bir nebze çözülmüş olsada tamamen ortadan kalktığı söylenemez.

Aşırı Code-Behind Bağımlılığı : Katmansal bir mimariyle (N-Tier vb.) web uygulaması geliştireceksiniz diyelim. Sunum ve iş katmanlarını ayırdınız iyi güzelde ASP.NET formları aslında bu yapıya oldukça ters. Sebebi bir çok mantıksal işlemleri yapacak kodları sunum katmanındaki sayfaların code-behind tarafında yazmak zorunda kalmamız. Hani katmansal

mimarimiz? Ayrıca bu durum projenin daha sonra güncellemek veya eklentiler yapmak için ele alındığında işleri daha da zorlaştıracaktır çünkü dağınık bir kod yapısı bizi beklemektedir.

MVC Nedir?

Klişe bilgiler olacak ama kısaca MVC (Model-View-Controller), 1979 'da Trygve Reenskaug tarafından ortaya çıkarıldıktan sonra Xerox Parc'ta kurcalanmaya devam edilmiş ve yazılım mühendisliğinde kullanılan bir mimari desendir. İlk zamanlar adı Thing-Model-View-Controller olarak tanımlanmıştır İlk kullanım kılavuzu Applications Programming in Smalltalk-80: How to use Model-View-Controller olarak bilinir.

MVC Ne Değildir?

Şunu unutmayalım MVC (Model-View-Controller) bir yazılım dili değil, bir mimari desendir. Eğer MVC öğrenirken kendimizi bir yazılım dili öğrenecekmiş gibi hazırlarsak bu yapıyı anlamamız daha zor olacaktır.

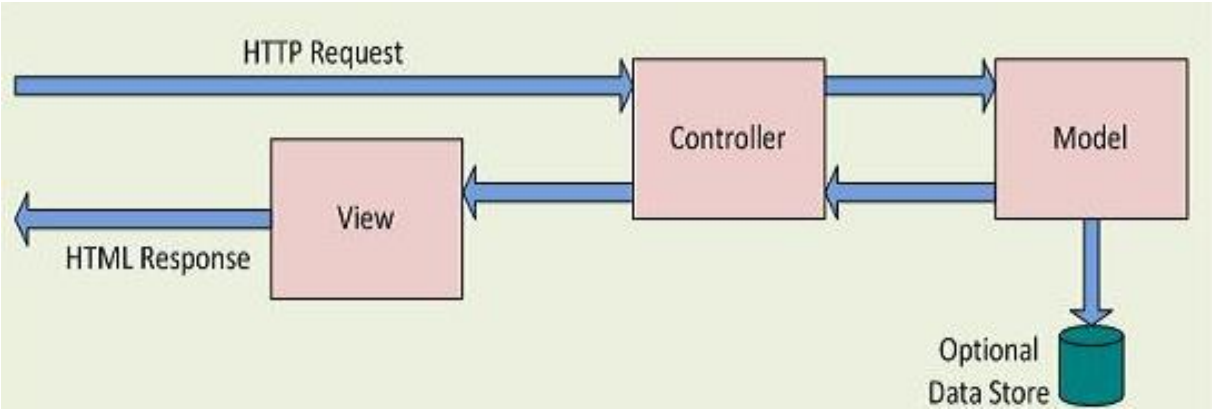
MVC Microsoft'a Mı Ait?

MVC denildiğinde akla bugünlerde Microsoft geliyor sebebi ASP.NET MVC 4 adında müthiş bir MVC framework çıkarmış olması fakat tek başına MVC kavramı Microsoft'a ait bir mimari/terim/ürün/desen değildir.

MVC Mimarisi

MVC mimarisinde uygulama üç farklı yapıya ayrılır. Model, View ve Controller.

Model : Veri kaynağının ki bu veritabanıdır genelde bulunduğu bölümdür. Veri kaynağının yanı sıra prosedürler ve işleyiş kurallarıda bu bölümde yer alır. Katmanlı mimariye müthiş uyumludur. Tek katmanlı olabileceği gibi fazlaca katmana da sahip olabilir.
View : Kaba tabirle, arayüze ait olup kullanıcının gördüğü şeyler bu bölümde yer alır. Örneğin MVC ile hazırlanmış bir web sitesi ya da uygulamasında arayüze dair ne varsa html, javascript, css… bu bölümdedir. View bölümünde iş akışına ait bir şeyler bulunmaz. Ayrıca View bölümü sayesinde uygulamanın arayüzü uygulamanın çekirdek kısmından ayrı tutulduğundan tasarımı ve tasarımın değiştirilmesi açısından bize avantaj sağlar.
Controller : İş akışının gerçekleştiği, arayüzden gelen kullanıcı etkileşimlerinin değerlendirildiği, işlendiği, gerekli metodların çalıştırıldığı, değişkenlerin ve nesnelerin oluşturulduğu, gerekirse modelle View bölümleri arasında iletişimin sağlandığı yer burasıdır.



Yukarıdaki şemada ASP.NET MVC mimarisinin nasıl çalıştığını gösteren aşamaları görüyoruz.

Özetle: Client (kullanıcı/ziyaretçi) tarafından başlatılan request (istek), controller tarafında işleme alınıyor ve neler yapılması gerekiyorsa, modelle irtibata geçilmesi gerekiyorsa modele giderek ilgili nesnelere ulaşıyor ve oradaki işlemlerini yaptıktan sonra tekrar controller'a geliniyor ve controller'da varsa son işlemden geçerek View'a gönderiliyor ve bizim reponse dediğimiz sunucu cevabı HTML çıktısı olarak bize geliyor.

Her View için bir Controller vardır fakat her Controller için View şartı yoktur.