

مرجع کامل

ASP.NET MVC 5.2

مهندس بهروز راد

انتشارات پندار پارس

سرشناسه : راد، بهروز، ۱۳۶۴ -
 عنوان و نام پدیدآور : مرجع کامل ASP.NET MVC 5.2/بهرروز راد.
 مشخصات نشر : تهران : پندار پارس، ۱۳۹۳.
 مشخصات ظاهری : ۷۲۰ ص.: مصور، جدول .
 شابک : ۹۷۸-۶۰۰-۶۵۲۹-۶۲-۲:۲ ریال:۴۲۰۰۰۰
 وضعیت فهرست نویسی : فیبا
 موضوع : صفحه‌های سرور فعال
 موضوع : ای.اس.پی. (پروتکل شبکه کامپیوتری)
 موضوع : پایگاه‌های اطلاعاتی -- مدیریت
 موضوع : مایکروسافت دات‌نت فریم‌ورک
 موضوع : شخصی‌سازی وب
 رده بندی کنگره : ASP.NET MVC 5.2/بهرروز راد
 رده بندی دیویی : ۳۷۴/۰۰۵
 شماره کتابشناسی ملی : ۳۵۵۴۶۹۴

انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ www.pendarepars.com
 تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۱۲۲۴۵۲۳۴۸
info@pendarepars.com



نام کتاب : مرجع کامل ASP.NET MVC 5.2
 ناشر : انتشارات پندار پارس
 ترجمه و تالیف : مهندس بهروز راد
 چاپ نخست : مرداد ۹۳
 شمارگان : ۱۰۰۰ نسخه
 لیتوگرافی، چاپ، صحافی : ترام‌سنج، فرشویه، خیام

قیمت : ۴۲۰۰۰ تومان شابک : ۹۷۸-۶۰۰-۶۵۲۹-۶۲-۲



*هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد *

تقدیم با بوسه بر دستان

پدرم که همچون کوه استوار است

و مادرم، که به زلالی چشمه است

و نیز همسرم، که به صمیمیت باران است

درباره‌ی نویسنده

بهروز راد با بیش از ۱۴ سال تجربه‌ی برنامه‌نویسی با تمرکز بر بستر وب، خالق کتاب‌های "مرجع کامل Regular Expressions"، "مرجع کامل CSS" و "مرجع کامل Entity Framework" است. او هم‌اینک به عنوان عضو تیم زیرساخت در شرکتی که دارای بالاترین رتبه در تمامی زمینه‌ها توسط شورای عالی انفورماتیک است، بر روی توسعه و نگهداری زیرساختی مبتنی بر ASP.NET MVC مشغول به کار است. وی از نسخه‌های ابتدایی ASP.NET MVC با آن آشنا بوده و کار می‌کرده است و علاقه‌ی بسیاری به تکنولوژی‌های مرتبط با توسعه‌ی وب دارد. او در اوقات فراغت خود، به توسعه‌ی پروژه‌های شخصی، تدریس و تماشای فیلم می‌پردازد.

پیش‌گفتار

این روزها نقش وب به عنوان بستری با اهمیت برای انجام فعالیت‌های آنلاین، مانند به اشتراک‌گذاری منابع، تجارت، حضور در شبکه‌های اجتماعی و بسیاری افعال دیگر بر هیچ‌کس پوشیده نیست. رشد و سرعت پیشرفت اینترنت به حدی است که باعث می‌شود روزانه ابزارهای مختلفی برای تسهیل در ایجاد بسترهای لازم برای توسعه‌ی برنامه‌های مبتنی بر وب خلق شوند. مایکروسافت به عنوان یکی از بزرگ‌ترین و مطرح‌ترین شرکت‌ها در تولید و ایجاد بسترهای مورد نیاز برای توسعه‌گران، با توجه مطلوب به نقش پُر رنگ وب، شرکتی پیشتاز در ارائه‌ی ابزارها و بسترهای لازم برای تولید برنامه‌های مبتنی بر وب است. ASP.NET MVC، جدیدترین فریمورکی است که با توجه کامل به امکان استفاده از جدیدترین استانداردهای وب، قابلیت سفارشی‌سازی، و مفاهیم واقعی پروتکل HTTP، توسط مایکروسافت عرضه شده است و هر روز بیش از پیش مورد توجه قرار می‌گیرد. در این کتاب سعی شده است تا خواننده با همه‌ی ابعاد این فریمورک آشنا شده و در پایان بتواند به عنوان یک متخصص ASP.NET MVC شناخته شود. تمام سعی من بر این بوده است که مفاهیم، چه آنها که ترجمه بوده‌اند و چه آنها که توسط اینجانب نوشته شده‌اند، در قالبی قابل درک و ملموس ارائه شوند و جای ابهامی برای شما خواننده‌ی گرامی باقی نماند. با این وجود، با توجه به حجم کار و از آنجا که هیچ نوشته‌ای بدون اشکال و بحث نیست، دوستان گرامی می‌توانند نظرات خود را مستقیماً با اینجانب از طریق ایمیل behrouz.rad@gmail.com در میان بگذارند. در پایان، از همه‌ی کسانی که به من یاد دادند چگونه یاد بگیرم و چگونه یاد بدهم سپاسگزاری می‌کنم.

با آرزوی بهترین‌ها...

بهروز راد

تابستان ۹۳

فهرست

۱	بخش نخست: معرفی ASP.NET MVC
۳	فصل ۱: ایده‌ی اصلی
۳	تاریخچه‌ی کوتاهی از توسعه‌ی برنامه‌های مبتنی بر وب
۵	ASP.NET Web Forms
۶	مشکلات ASP.NET Web Forms چیست؟
۷	جایگاه توسعه‌ی وب در زمان حال
۷	استانداردهای وب و REST
۷	Agile و توسعه‌ی تست محور
۹	Ruby on Rails
۹	Sinatra
۱۰	Node.js
۱۰	مزایای اصلی ASP.NET MVC
۱۱	معماری MVC
۱۱	توسعه‌پذیری
۱۲	کنترل کامل بر HTML و HTTP
۱۲	تست‌پذیری
۱۳	سیستم مسیریابی قدرمند
۱۳	ساخته شده بر مبنای بهترین قسمت‌های ASP.NET
۱۴	API پیشرفته
۱۴	ASP.NET MVC، متن باز است
۱۵	عدم وابستگی الزامی به فایل‌های فیزیکی موجود در سیستم
۱۵	امکان مدیریت بهتر قسمت‌های مختلف سایت در پوشه‌های جداگانه
۱۶	کنترل بهتر بر روی اعتبار سنجی اطلاعات دریافتی
۱۶	امکان استفاده از فرم‌های و View‌های Razor به جای موتور وب فرم‌ها
۱۶	امکان تعریف بیش از یک فرم در صفحه
۱۶	امکان Refactoring بهتر کدهای تکراری در ASP.NET MVC به کمک مفهوم فیلترها
۱۶	چه کسی باید از ASP.NET MVC استفاده کند؟
۱۷	مقایسه با ASP.NET Web Forms
۱۷	مهاجرت از ASP.NET Web Forms به ASP.NET MVC
۱۸	مقایسه با Ruby on Rails
۱۸	مقایسه با MonoRail
۱۸	نتیجه‌گیری
۱۹	فصل ۲: آمادگی برای شروع

۱۹	آماده‌سازی سیستم توسعه
۱۹	Visual Studio 2013 نصب
۲۰	ASP.NET MVC 5 نصب
۲۱	نصب ابزارهای اختیاری
۲۱	ASP.NET MVC کدهای
۲۲	IIS Express
۲۲	SQL Server Management Studio Express
۲۲	آماده‌سازی سرور
۲۳	IIS نصب
۲۴	نصب ابزارهای اضافه
۲۴	منابع بیشتر برای یادگیری
۲۵	نتیجه‌گیری
۲۷	فصل ۳: نخستین پروژه‌ی ASP.NET MVC
۲۷	ایجاد یک پروژه‌ی جدید ASP.NET MVC
۲۹	قالب Empty
۲۹	قالب Web Forms
۳۰	قالب MVC
۳۰	قالب Web API
۳۰	قالب Single Page Application
۳۱	قالب Facebook
۳۱	قالب Visual Studio 2012
۳۲	VS 2013 در پروژه‌های Bootstrap
۳۴	پشتیبانی از چند فریم‌ورک در پروژه
۳۵	روش‌های تصدیق هویت در قالب‌های پیشفرض
۳۵	گزینه‌ی No Authentication
۳۶	گزینه‌ی Individual User Accounts
۳۶	گزینه‌ی Organizational Accounts
۳۶	گزینه‌ی Windows Authentication
۳۸	اضافه کردن نخستین کنترلر
۴۰	آشنایی با Routeها
۴۱	پردازش در صفحات وب
۴۱	ایجاد یک View
۴۴	اضافه کردن خروجی پویا
۴۶	ایجاد یک پروژه‌ی ساده برای کار با داده‌ها
۴۶	آماده‌سازی

۴۷	طراحی مدل داده‌ها
۴۷	افزودن یک کلاس برای مدل
۴۸	ارتباط بین متدهای اکشن
۴۹	ایجاد یک متد اکشن
۵۰	ایجاد یک View نوع‌دار
۵۰	ایجاد ساختار View
۵۳	مدیریت فرم‌ها
۵۵	استفاده از Model Binding
۵۵	نمایش Viewهای دیگر
۵۷	افزودن تعیین اعتبار
۵۹	مشخص کردن کنترل‌های نامعتبر
۶۰	ایجاد ظاهر زیبا
۶۱	استفاده از NuGet برای نصب Bootstrap
۶۲	ایجاد ظاهر زیبا برای Index View
۶۳	ایجاد ظاهر زیبا برای RsvpForm View
۶۵	ایجاد ظاهر زیبا برای Thanks View
۶۶	تکمیل پروژه
۶۷	نتیجه‌گیری
۶۹	فصل ۴: معماری MVC
۶۹	تاریخچه‌ی MVC
۷۰	مفهوم الگوی MVC
۷۱	آشنایی با Domain Model
۷۱	پیاده‌سازی معماری MVC در ASP.NET
۷۲	مقایسه‌ی MVC با الگوهای دیگر
۷۳	آشنایی با الگوی Smart UI
۷۴	آشنایی با معماری Model-View
۷۴	آشنایی با معماری سه لایه‌ی کلاسیک
۷۵	آشنایی با اشکال مختلف الگوی MVC
۷۵	آشنایی با الگوی Model-View-Presenter
۷۶	آشنایی با الگوی Model-View-View Model
۷۷	متدولوژی Domain Driven Design
۷۷	مدل کردن یک Domain
۷۸	زبان یکپارچه
۷۸	Simplification و Aggregate
۸۰	ایجاد Repositoryها

۸۱	ایجاد بخش‌های تفکیک شده
۸۲	استفاده از تزریق وابستگی (Dependency Injection)
۸۴	مثالی از تزریق وابستگی در ASP.NET MVC
۸۵	استفاده از Dependency Injection Container
۸۶	آغازی برای آزمایش واحد خودکار
۸۷	آشنایی با Unit Testing
۸۹	استفاده از TDD و منطق Red-Green-Refactor
۹۵	به سوی آیین TDD بشتابید!
۹۵	آشنایی با Integration Testing
۹۶	نتیجه‌گیری
۹۷	فصل ۵: قابلیت‌های کلیدی زبان
۹۷	قابلیت‌های کلیدی C#
۹۷	استفاده از Automatic Properties
۹۹	استفاده از Collection_INITIALIZER و Object_INITIALIZER
۱۰۱	استفاده از Extension Methods
۱۰۲	اعمال Extension Method به یک Interface
۱۰۴	ایجاد Extension Methodها برای فیلتر کردن
۱۰۵	استفاده از عبارات‌های لامبدا
۱۰۶	شکل‌های دیگر عبارات‌های لامبدا
۱۰۷	استفاده از Type Inference
۱۰۸	استفاده از Anonymous Type
۱۰۸	استفاده از LINQ
۱۱۲	آشنایی با مفهوم کوئری‌های با تأخیر در LINQ
۱۱۴	استفاده‌ی دوباره از یک کوئری با تأخیر
۱۱۵	LINQ و اینترفیس IQueryable<T>
۱۱۶	استفاده از متدهای Async
۱۱۷	استفاده از کلیدواژه‌های await و async
۱۱۸	آشنایی با سینتکس موتور Razor
۱۱۹	ایجاد پروژه
۱۱۹	ایجاد Model
۱۱۹	ایجاد کنترلر
۱۲۰	ایجاد View
۱۲۰	تنظیم مسیر پیش فرض
۱۲۱	بررسی یک View ساده در Razor
۱۲۱	کار با شیء Model

۱۲۲ استفاده از کد در Razor
۱۲۴ تعریف یک بلاک کد در Razor
۱۲۵ انتقال مقادیر به View با استفاده از شیء ViewBag
۱۲۶ کار با قالب‌ها
۱۲۸ کار بدون قالب‌ها
۱۲۹ تفسیر خودکار عبارت "/"~" توسط موتور Razor
۱۲۹ صفت‌های شرطی HTML در موتور Razor
۱۳۰ نتیجه‌گیری
۱۳۱ فصل ۶: ابزارهای مهم برای ASP.NET MVC
۱۳۲ استفاده از Ninject
۱۳۳ ایجاد پروژه
۱۳۴ اضافه کردن Ninject
۱۳۴ شروع کار با Ninject
۱۳۶ ایجاد زنجیره‌ای از وابستگی‌ها
۱۳۷ تعیین مقادیر خصیصه‌ها و پارامترها
۱۳۸ استفاده از Self-Binding
۱۳۸ برگشت نوع مشتق شده
۱۴۰ استفاده از شرط در معرفی کلاس‌ها
۱۴۱ استفاده از Ninject در ASP.NET MVC
۱۴۲ آزمایش واحد با Visual Studio
۱۴۲ ایجاد پروژه
۱۴۴ ایجاد آزمایش‌های واحد
۱۴۹ اجرای آزمایش‌های واحد و مواجهه شدن با خطا
۱۴۹ پیاده‌سازی قابلیت
۱۵۰ استفاده از Moq
۱۵۱ افزودن Moq به پروژه
۱۵۱ ایجاد یک Mock با Moq!
۱۵۲ استفاده از قابلیت انتخاب متد توسط Moq
۱۵۲ تعیین مقدار برای پارامتر متدها توسط Moq
۱۵۳ برگشت یک نتیجه
۱۵۳ آزمایش واحد با Moq
۱۵۵ تأیید با Moq
۱۵۶ نتیجه‌گیری
۱۵۷ بخش دوم: بررسی کامل ASP.NET MVC
۱۵۹ فصل ۷: نمای کلی پروژه‌های ASP.NET MVC

۱۵۹ کار با پروژه‌های ASP.NET MVC
۱۶۳ آشنایی با مفهوم قراردادهای ASP.NET MVC
۱۶۳ قوانین نام‌گذاری کلاس‌های کنترلر
۱۶۴ قوانین نام‌گذاری View ها
۱۶۴ قوانین نام‌گذاری برای قالب‌ها
۱۶۵ دیباگ پروژه‌های ASP.NET MVC
۱۶۵ ایجاد پروژه
۱۶۵ اجرای دیباگر Visual Studio
۱۶۶ توقف دیباگر در Breakpoint
۱۶۷ استفاده از Breakpoint ها
۱۶۹ توقف دیباگر در زمان رخ دادن خطا
۱۷۰ استفاده از قابلیت Edit and Continue
۱۷۰ فعال‌سازی قابلیت Edit and Continue
۱۷۱ اصلاح پروژه
۱۷۲ Edit and Continue
۱۷۲ استفاده از قابلیت اتصال مرورگر
۱۷۴ دیباگ آزمایش‌های واحد
۱۷۵ استفاده از تزریق وابستگی در کل پروژه
۱۷۷ نتیجه‌گیری
۱۷۹ فصل ۸: آدرس‌ها، مسیریابی و AREA ها
۱۷۹ معرفی سیستم مسیریابی
۱۸۰ اسمبلی سیستم مسیریابی
۱۸۰ ایجاد پروژه‌ی مسیریابی
۱۸۱ آشنایی با URL Patterns
۱۸۳ ایجاد و معرفی یک Route ساده
۱۸۴ تعریف مقادیر پیش‌فرض
۱۸۶ ایجاد URL Pattern های ثابت
۱۸۶ حق تقدم Route ها
۱۸۸ تعریف متغیرهای Segment سفارشی
۱۸۹ استفاده از متغیرهای Segment سفارشی به عنوان پارامترهای یک متد اکشن
۱۹۰ تعریف متغیرهای Segment اختیاری
۱۹۱ تعریف مسیرهای با طول متغیر
۱۹۲ اولویت‌بندی کنترلرها به وسیله‌ی فضاها‌ی نام
۱۹۴ ایجاد قید برای Route ها
۱۹۴ ایجاد قید با استفاده از عبارتهای باقاعده

۱۹۵	ایجاد قید برای یک Route بر مبنای چند مقدار
۱۹۵	ایجاد قید برای یک Route با استفاده از متدهای HTTP
۱۹۶	استفاده از قیدهای نوع و مقدار
۱۹۸	تعریف یک قید سفارشی
۱۹۹	مسیریابی بر مبنای صفت
۱۹۹	مسیریابی قرارداد محور در مقایسه با مسیریابی بر مبنای صفت
۲۰۰	فعال سازی و استفاده از قابلیت مسیریابی بر مبنای صفت
۲۰۲	ایجاد Routeها با متغیرهای Segment
۲۰۴	ادغام قیدها
۲۰۴	استفاده از پیشوند برای مسیر
۲۰۵	رفتار Routeها با فایل های موجود بر روی فایل سیستم
۲۰۸	دور زدن سیستم مسیریابی
۲۰۹	ایجاد آدرس های خروجی
۲۰۹	چه کاری نباید انجام دهیم؟ تعریف دستی آدرسها!
۲۱۰	آماده سازی پروژه
۲۱۰	ایجاد آدرس های خروجی در Viewها
۲۱۱	آشنایی با نحوه ی تطبیق Route برای ایجاد آدرس خروجی
۲۱۲	تعیین کنترلر دلخواه در ایجاد آدرس خروجی
۲۱۲	پاس دادن مقادیر اضافه
۲۱۳	آشنایی با مفهوم «استفاده ی دوباره از متغیرهای Segment»
۲۱۴	کار با صفت های HTML
۲۱۵	ایجاد آدرس های کامل برای لینکها
۲۱۵	ایجاد آدرس ها (و نه لینکها)
۲۱۶	ایجاد لینکها و آدرسها با استفاده از اطلاعات Route
۲۱۶	ایجاد آدرس های خروجی در متدهای اکشن
۲۱۷	ایجاد آدرس از یک Route مشخص
۲۱۷	نقطه ضعف استفاده از نام Routeها
۲۱۸	سفارشی سازی سیستم مسیریابی
۲۱۸	ایجاد رفتار تطبیقی سفارشی برای Routeها
۲۱۹	سفارشی سازی سیستم مسیریابی برای آدرس های ورودی
۲۲۱	سفارشی سازی سیستم مسیریابی برای ایجاد آدرس های خروجی
۲۲۲	ایجاد یک مدیر Route سفارشی
۲۲۳	کار با Areaها
۲۲۴	ایجاد یک Area
۲۲۶	کار با یک Area

۲۲۷	حل مشکل تداخل نام کنترلرها.....
۲۲۸	ایجاد لینک برای متدهای اکشن در Areaها.....
۲۲۹	طراحی مناسب آدرسها.....
۲۲۹	آدرسهای خود را ساده و کاربرپسند طراحی کنید.....
۲۳۱	GET و POST: انتخاب صحیح.....
۲۳۱	نتیجه گیری.....
۲۳۳	فصل ۹: کنترلرها و اکشنها.....
۲۳۳	معرفی کنترلر.....
۲۳۳	آمادهسازی پروژه.....
۲۳۳	ایجاد یک کنترلر با استفاده از اینترفیس IController.....
۲۳۵	ایجاد یک کنترلر با ارث‌بری از کلاس Controller.....
۲۳۶	دریافت ورودی.....
۲۳۷	استخراج داده‌ها از اشیاء Context.....
۲۳۸	استفاده از پارامترها در متد اکشن.....
۲۳۹	آشنایی با نحوه‌ی پُرشدن پارامترهای متد اکشن.....
۲۳۹	آشنایی با پارامترهای اختیاری و اجباری.....
۲۴۰	تعیین مقدار پیش فرض برای پارامتر.....
۲۴۱	تولید خروجی.....
۲۴۲	آشنایی با نتایج اکشن.....
۲۴۵	برگشت نتیجه در قالب HTML با ارسال یک View.....
۲۴۷	آزمایش واحد: پردازش یک View.....
۲۴۸	تعیین یک View با استفاده از مسیر آن.....
۲۴۹	انتقال داده‌ها از یک متد اکشن به یک View.....
۲۴۹	آماده‌سازی یک مدل برای یک View.....
۲۵۰	آزمایش واحد: مدل برای View.....
۲۵۱	انتقال داده‌ها با ViewBag.....
۲۵۲	آزمایش واحد: ViewBag.....
۲۵۲	انتقال داده‌ها با ViewData.....
۲۵۳	آزمایش واحد: ViewData.....
۲۵۳	هدایت کاربر به آدرسی دیگر.....
۲۵۳	الگوی POST/REDIRECT/GET.....
۲۵۴	هدایت کاربر به یک آدرس واقعی.....
۲۵۴	آزمایش واحد: هدایت کاربر به یک آدرس واقعی.....
۲۵۵	هدایت کاربر به مسیری ثبت شده در سیستم مسیریابی.....
۲۵۵	آزمایش واحد: مسیره‌های ثبت شده در سیستم مسیریابی.....

۲۵۶	هدایت کاربر به یک متد اکشن.....
۲۵۶	نگهداری داده‌ها در هنگام هدایت کاربر.....
۲۵۸	برگشت داده‌های متنی.....
۲۵۹	آزمایش واحد: نتایج حاصل از فراخوانی متد Content.....
۲۵۹	برگشت داده‌ها با فرمت XML.....
۲۶۰	برگشت داده‌ها با فرمت JSON.....
۲۶۱	ارسال فایل‌ها و داده‌های باینری.....
۲۶۱	ارسال یک فایل.....
۲۶۳	ارسال آرایه‌ای از بایت‌ها.....
۲۶۳	ارسال داده‌ها در قالب Stream.....
۲۶۴	آزمایش واحد: نتایج حاصل از فراخوانی متد File.....
۲۶۴	برگشت خطاها و کدهای HTTP.....
۲۶۵	ارسال یک کد HTTP مشخص.....
۲۶۵	برگشت نتیجه‌ی ۴۰۴.....
۲۶۵	برگشت نتیجه‌ی ۴۰۱.....
۲۶۵	آزمایش واحد: کدهای HTTP.....
۲۶۶	ایجاد یک نتیجه‌ی اکشن سفارشی.....
۲۶۸	نتیجه‌گیری.....
۲۶۹	فصل ۱۰: فیلترها
۲۶۹	استفاده از فیلترها.....
۲۷۰	Attributeها در .NET: یادآوری.....
۲۷۱	معرفی چهار نوع فیلتر.....
۲۷۲	اعمال فیلترها به کنترلرها و متدهای اکشن.....
۲۷۲	فیلترهای مرتبط با اعطای مجوز دسترسی به منبع.....
۲۷۳	ایجاد یک فیلتر مرتبط با اعطای مجوز دسترسی به منبع.....
۲۷۵	استفاده از فیلتر موجود مجوز دسترسی به منبع.....
۲۷۶	ایجاد منطق سفارشی برای دسترسی به منابع.....
۲۷۷	ایجاد منطق سفارشی، هنگام نداشتن مجوز دسترسی به منبع.....
۲۷۸	فیلتر AllowAnonymous.....
۲۷۸	استفاده از فیلترهای مدیریت خطا.....
۲۷۸	ایجاد یک فیلتر مدیریت خطا.....
۲۸۰	استفاده از فیلتر موجود مدیریت خطا.....
۲۸۱	استفاده از فیلترهای Action و Result.....
۲۸۲	پیاده‌سازی متد OnActionExecuting.....
۲۸۳	پیاده‌سازی متد OnActionExecuted.....

۲۸۴	پیااده‌سازی یک فیلتر Result
۲۸۶	استفاده از فیلتر موجود Action و Result
۲۸۷	استفاده از قابلیت‌های دیگر فیلترها
۲۸۸	استفاده از فیلترها بدون Attribute
۲۸۹	استفاده از فیلترهای Global
۲۹۰	ترتیب اجرای فیلترها
۲۹۲	استفاده از فیلترهای موجود
۲۹۳	استفاده از فیلتر RequireHttps
۲۹۳	استفاده از فیلتر OutputCache
۲۹۷	نتیجه‌گیری
۲۹۹	فصل ۱۱: سفارشی‌سازی کنترلرها
۲۹۹	اجزای دخیل در روند اجرای درخواست
۲۹۹	ایجاد یک Controller Factory
۳۰۰	ایجاد یک Controller Factory سفارشی
۳۰۲	ثبت یک Controller Factory سفارشی
۳۰۲	کار با Controller Factory موجود
۳۰۳	ایجاد حق تقدم برای فضاها نام
۳۰۴	سفارشی‌سازی فرایند ایجاد کنترلر در DefaultControllerFactory
۳۰۴	استفاده از Dependency Resolver
۳۰۶	استفاده از یک Controller Activator
۳۰۷	Override کردن متدهای کلاس DefaultControllerFactory
۳۰۷	ایجاد یک Action Invoker سفارشی
۳۰۹	استفاده از Action Invoker موجود
۳۱۰	استفاده از یک نام متد اکشن سفارشی
۳۱۱	استفاده از Action Method Selection
۳۱۲	ایجاد یک Action Method Selector سفارشی
۳۱۳	مکانیزم کارکرد Action Method Selector
۳۱۴	مدیریت متدهای اکشنی که وجود ندارند
۳۱۴	استفاده از Action Method Selectorها برای پشتیبانی از سرویس‌های REST
۳۱۵	معرفی افعال HTTP به روشی دیگر
۳۱۶	معرفی افعال HTTP به روشی دیگر در یک فرم ASP.NET MVC
۳۱۷	افزایش کارایی با کنترلرهای خاص
۳۱۷	استفاده از کنترلرهای Sessionless
۳۱۷	مدیریت Sessionها از طریق IControllerFactory سفارشی
۳۱۸	مدیریت Sessionها با استفاده از DefaultControllerFactory

۳۱۹ استفاده از کنترلرهای نامتقارن
۳۲۰ ایجاد یک کنترلر نامتقارن
۳۲۲ عملیات پشت صحنه و Thread های بلاک شده
۳۲۲ ایجاد متدهای Async و Completed
۳۲۳ شروع عملیات نامتقارن
۳۲۴ پایان عملیات نامتقارن
۳۲۴ پاس دادن مقادیر از متد Async به متد Completed
۳۲۵ مدیریت حداکثر زمان مجاز برای اجرای عملیات نامتقارن
۳۲۶ توقف عملیات نامتقارن
۳۲۶ استفاده از الگوی برنامه‌نویسی نامتقارن NET
۳۲۸ چه هنگام باید از کنترلرهای نامتقارن استفاده نمود؟
۳۲۸ نتیجه‌گیری
۳۲۹ فصل ۱۲: VIEW ها
۳۲۹ ایجاد یک View Engine سفارشی
۳۳۱ ایجاد یک IView سفارشی
۳۳۲ ایجاد یک پیاده‌سازی از اینترفیس IViewEngine
۳۳۳ معرفی یک View Engine سفارشی
۳۳۵ بهبود سرعت نمایش صفحات با حذف View Engine های اضافی
۳۳۵ استفاده از View Engine های دیگر
۳۳۶ کار با موتور Razor
۳۳۶ آشنایی با نحوه‌ی پردازش View توسط موتور Razor
۳۳۸ استفاده از الگوی DI برای View های Razor
۳۴۰ پیکربندی مکان‌های جست‌وجوی View ها
۳۴۲ افزودن محتویات پویا به View های موتور Razor
۳۴۳ استفاده از Inline Code
۳۴۳ Inline Code و اصل جداسازی لایه‌ها
۳۴۳ معرفی فضای نام به یک View
۳۴۴ استفاده از دستور @using در یک View
۳۴۴ معرفی فضای نام در فایل Web.config
۳۴۵ آشنایی با مفهوم کدگذاری HTML در موتور Razor
۳۴۷ استفاده از View هایی با نوع dynamic به عنوان Model
۳۴۸ استفاده از HTML Helper ها
۳۴۸ ایجاد یک Inline HTML helper
۳۴۹ ایجاد یک External HTML helper
۳۵۱ استفاده از HTML Helper های موجود

۳۵۲	ایجاد تگ form
۳۵۲	ایجاد فرم‌هایی که به خودشان ارسال می‌شوند
۳۵۳	استفاده از Input Helper ها
۳۵۴	استفاده از متدهای HTML helper با نوع Strongly Typed
۳۵۵	اضافه کردن صفت‌ها به تگ HTML
۳۵۶	ایجاد تگ select
۳۵۷	ایجاد لینک‌ها و آدرس‌ها
۳۵۸	استفاده از WebGrid Helper
۳۶۳	استفاده از Chart Helper
۳۶۵	استفاده از Helper های دیگر موجود
۳۶۵	استفاده از Section ها
۳۶۸	بررسی وجود Section ها (روش نخست)
۳۶۸	بررسی وجود Section ها (روش دوم)
۳۶۹	استفاده از Partial View ها
۳۶۹	ایجاد یک Partial View
۳۷۰	استفاده از Partial View های Strongly Typed
۳۷۲	استفاده از Child Action ها
۳۷۲	ایجاد یک متد Child Action
۳۷۳	فراخوانی متد Child Action
۳۷۴	نمایش View ها بر اساس نوع دستگاه درخواست کننده
۳۷۴	انتخاب View در زمان اجرا با قابلیت DisplayModes
۳۷۴	استفاده از حالت Mobile قابلیت DisplayModes
۳۷۴	تست دستی قابلیت DisplayModes برای دستگاه‌های مختلف
۳۷۶	ایجاد DisplayMode های سفارشی
۳۷۸	خلاقیت با قابلیت DisplayModes
۳۷۸	دادن اختیار به کاربر برای تغییر DisplayMode ها
۳۸۱	نتیجه‌گیری
۳۸۳	فصل ۱۳: قالب‌های MODEL
۳۸۳	استفاده از Templated View Helpers
۳۸۷	استفاده از CSS در HTML تولیدی
۳۸۸	استفاده از Model Metadata
۳۸۹	استفاده از Metadata برای کنترل ویرایش و رؤیت
۳۹۱	مستثنا کردن یک خصیصه از تولید کد HTML برای آن
۳۹۱	استفاده از Metadata برای تگ <label>
۳۹۲	استفاده از Metadata برای مقادیر

۳۹۴	استفاده از Metadata برای انتخاب قالب نمایش
۳۹۷	اعمال Metadata به یک کلاس Buddy
۳۹۸	کار با خصیصه‌هایی از نوع Complex
۳۹۹	سفارشی‌سازی سیستم تولید کدهای HTML
۳۹۹	ایجاد یک قالب سفارشی برای حالت ویرایش
۴۰۲	آشنایی با ترتیب جست‌وجوی قالب
۴۰۳	ایجاد یک قالب سفارشی برای حالت نمایش
۴۰۳	ایجاد یک قالب عمومی
۴۰۴	جایگزینی قالب‌های موجود
۴۰۵	استفاده از خصیصه‌ی ViewData.TemplateInfo
۴۰۶	توضیحی در ارتباط با فرمت داده‌ها
۴۰۶	کار با پیشنهادهای تولیدی برای تگ‌های HTML
۴۰۷	فراهم کردن اطلاعات اضافی برای یک قالب
۴۰۸	آشنایی با سیستم Metadata Provider
۴۰۹	ایجاد یک Model Metadata Provider سفارشی
۴۱۲	سفارشی‌سازی Data Annotations Model Metadata Provider
۴۱۳	نتیجه‌گیری
۴۱۵	فصل ۱۴: MODEL BINDING
۴۱۵	آشنایی با Model Binding
۴۱۶	استفاده از Model Binder پیش‌فرض
۴۱۷	Model Binding برای انواع داده‌های ساده
۴۱۸	حساسیت Model Binding به قوانین زبان
۴۱۸	Model Binding برای نوع‌های Complex
۴۲۰	ایجاد کدهای HTML برای Model Binding آسان
۴۲۰	تعیین پیشنهادهای سفارشی
۴۲۲	انتخاب خصیصه‌های مورد نظر برای Binding
۴۲۳	Binding برای آرایه‌ها و مجموعه‌ها
۴۲۳	Binding برای مجموعه‌ای از انواع داده‌های سفارشی
۴۲۴	Binding برای مجموعه‌ها با اندیس‌های غیر ترتیبی
۴۲۵	Binding برای نوع داده‌ی Dictionary
۴۲۵	فراخوانی Model Binding با کدنویسی
۴۲۶	محدود کردن فرایند Model Binding به منبعی مشخص
۴۲۷	مدیریت خطاها در فرایند Model Binding
۴۲۸	استفاده از Model Binding برای دریافت فایل‌های آپلودی
۴۲۹	سفارشی‌سازی سیستم Model Binding

۴۲۹.....	ایجاد یک Value Provider سفارشی
۴۳۱.....	ایجاد یک Model Binder با الگوی DI
۴۳۲.....	ایجاد یک Model Binder سفارشی
۴۳۴.....	ایجاد Model Binder Providerها
۴۳۵.....	استفاده از صفت ModelBinder
۴۳۶.....	نتیجه‌گیری
۴۳۷.....	فصل ۱۵: MODEL VALIDATION
۴۳۷.....	ایجاد پروژه
۴۳۹.....	تعیین اعتبار یک مدل به شکل صریح
۴۴۱.....	ایجاد ظاهر مناسب برای Check Boxها
۴۴۲.....	نمایش پیغام‌های خطای مرتبط با تعیین اعتبار مقادیر
۴۴۵.....	نمایش پیغام‌های خطای در سطح خصیصه
۴۴۶.....	تکنیک‌های دیگر تعیین اعتبار مقادیر
۴۴۷.....	تعیین اعتبار مقادیر از طریق Model Binder
۴۵۰.....	تعیین قوانین تعیین اعتبار داده‌ها از طریق Metadataها
۴۵۳.....	ایجاد یک Attribute سفارشی برای تعیین اعتبار در سطح خصیصه
۴۵۴.....	ایجاد یک Attribute سفارشی برای تعیین اعتبار در سطح مدل
۴۵۵.....	ایجاد مدل‌هایی که خود را تعیین اعتبار می‌کنند
۴۵۶.....	ایجاد یک Validation Provider سفارشی
۴۶۰.....	معرفی یک Validation Provider سفارشی
۴۶۰.....	تعیین اعتبار در سمت کلاینت
۴۶۲.....	فعال‌سازی/غیر فعال‌سازی تعیین اعتبار در سمت کلاینت
۴۶۳.....	آشنایی با CDN
۴۶۴.....	استفاده از یک CDN برای کتابخانه‌های JavaScript
۴۶۵.....	استفاده از قابلیت تعیین اعتبار داده‌ها در سمت کلاینت
۴۶۸.....	آشنایی با نحوه‌ی کارکرد فرایند تعیین اعتبار در سمت کلاینت
۴۶۹.....	قابلیت درونی ASP.NET MVC برای تعیین اعتبار در سمت کلاینت در مقایسه با کتابخانه‌ی jQuery Validation
۴۶۹.....	سفرسازی فرایند تعیین اعتبار در سمت کلاینت
۴۷۰.....	ایجاد مستقیم صفت‌های مرتبط با تعیین اعتبار در کدهای HTML
۴۷۲.....	ایجاد صفت‌هایی که از تعیین اعتبار سمت کلاینت پشتیبانی می‌کنند
۴۷۴.....	ایجاد قوانین تعیین اعتبار سفارشی در سمت کلاینت
۴۷۶.....	آشنایی با Remote Validation
۴۷۹.....	نتیجه‌گیری
۴۸۱.....	فصل ۱۶: AJAX

۴۸۱	استفاده از قابلیت Unobtrusive AJAX در ASP.NET MVC
۴۸۱	ایجاد پروژه
۴۸۴	فعال‌سازی/غیرفعال‌سازی Unobtrusive AJAX
۴۸۴	استفاده از فرم‌های Unobtrusive AJAX
۴۸۶	آشنایی با نحوه‌ی کارکرد قابلیت Unobtrusive AJAX
۴۸۷	تنظیمات AJAX
۴۸۷	اصل «تنزل مطبوع» (Graceful Degradation)
۴۸۹	آگاه‌سازی کاربر در هنگام ایجاد یک درخواست AJAX
۴۹۰	تأیید کاربر، پیش از ارسال درخواست AJAX
۴۹۱	ایجاد لینک‌های آژاکسی
۴۹۳	اصل «تنزل مطبوع» برای لینک‌های AJAX بی
۴۹۴	کار با Callbackها در AJAX
۴۹۷	کار با JSON
۴۹۸	افزودن پشتیبانی از فرمت JSON به کنترلر
۵۰۰	پردازش داده‌های JSON در مرورگر
۵۰۱	تشخیص درخواست‌های AJAX در متد اکشن
۵۰۲	ارسال داده‌ها با فرمت JSON به سرور
۵۰۴	نتیجه‌گیری
۵۰۵	فصل ۱۷: JQUERY
۵۰۵	ایجاد پروژه
۵۰۷	ارجاع به jQuery
۵۰۸	مدیریت نسخه‌های مختلف jQuery
۵۰۹	نوشتن کدهای jQuery
۵۱۰	اجرای jQuery در یک محیط ایزوله
۵۱۰	استفاده از Firefox
۵۱۲	استفاده از Chrome
۵۱۳	مبانی jQuery
۵۱۴	آشنایی با Selectorهای jQuery
۵۱۵	نکته‌ای در ارتباط با Id هر تگ
۵۱۵	استفاده از چند Selector به طور هم‌زمان
۵۱۶	استفاده از Attribute Selectorها
۵۱۷	استفاده از فیلترها در jQuery
۵۱۸	استفاده از فیلترهای محتوا
۵۱۹	استفاده از فیلترهای فرم
۵۱۹	آشنایی با متدهای jQuery

۵۲۰	انتظار برای بارگذاری صفحه
۵۲۱	متدهای مرتبط با کار با CSS در JQuery
۵۲۴	کار با DOM
۵۲۸	استفاده از رویدادها در JQuery
۵۲۹	استفاده از جلوه‌های بصری در JQuery
۵۳۱	استفاده از JQuery UI
۵۳۲	ارجاع به کتابخانه‌ی JQuery UI
۵۳۳	استفاده از ابزار ThemeRoller
۵۳۳	ایجاد دکمه‌هایی با ظاهری زیباتر
۵۳۴	استفاده از کامپوننت Slider
۵۳۷	نتیجه‌گیری
۵۳۹	بخش سوم: ادامه‌ی توانایی‌های پروژه‌های ASP.NET MVC
۵۴۱	فصل ۱۸: امنیت و آسیب‌پذیری
۵۴۱	تمامی ورودی‌های برنامه می‌توانند جعل شوند!
۵۴۲	HTTP چگونه کار می‌کند؟
۵۴۲	یک درخواست GET ساده
۵۴۳	یک درخواست POST همراه با کوکی‌ها
۵۴۳	جعل درخواست‌های HTTP
۵۴۵	HTML Injection و Cross-Site Scripting
۵۴۵	آشنایی با حملات XSS
۵۴۷	کدگذاری HTML از طریق موتور Razor
۵۴۸	تعیین اعتبار درخواست
۵۴۹	غیر فعال‌سازی قابلیت تعیین اعتبار درخواست
۵۵۰	کدگذاری مقادیر رشته‌ای برای JavaScript
۵۵۲	Session Hijacking
۵۵۳	محافظت از طریق بررسی آدرس IP درخواست‌دهنده
۵۵۳	محافظت با تنظیم خصیصه‌ی HttpOnly کوکی‌ها
۵۵۴	حملات Cross Site Request Forgery
۵۵۴	حمله
۵۵۵	دفاع
۵۵۶	جلوگیری از حملات CSRF در ASP.NET MVC
۵۵۸	SQL Injection
۵۵۸	حمله
۵۵۹	دفاع با استفاده از کوئی‌های پارامتردار
۵۵۹	دفاع با استفاده از ORM‌ها

استفاده‌ی امن از ASP.NET MVC.....	۵۵۹
متدهای اکشن را سهوا در معرض دسترسی قرار ندهید.....	۵۵۹
اجازه ندهید Model Binding، مقادیر خصیصه‌های حساس را تغییر دهد.....	۵۶۰
نتیجه‌گیری.....	۵۶۱
فصل ۱۹: تصدیق هویت و مجوز دسترسی به منابع.....	۵۶۳
استفاده از تصدیق هویت Windows.....	۵۶۳
استفاده از روش تصدیق هویت بر اساس فرم‌ها.....	۵۶۶
تنظیمات Forms Authentication.....	۵۶۸
مدیریت لاگین.....	۵۷۰
استفاده از Forms Authentication، بدون کوکی.....	۵۷۰
استفاده از سیستم عضویت، نقش‌ها و پروفایل‌ها.....	۵۷۲
پیکربندی و استفاده از سیستم عضویت.....	۵۷۴
استفاده از SqlMembershipProvider با SQL Server نسخه‌ی Express.....	۵۷۴
آماده‌سازی دستی SQL Server.....	۵۷۴
مدیریت سیستم عضویت.....	۵۷۶
ایجاد یک Provider سفارشی برای سیستم عضویت.....	۵۷۶
پیکربندی و استفاده از نقش‌ها.....	۵۷۸
پیکربندی SqlRoleProvider.....	۵۷۹
مدیریت نقش‌ها.....	۵۷۹
ایجاد یک Provider سفارشی برای سیستم نقش‌ها.....	۵۸۰
پیکربندی و استفاده از پروفایل‌ها.....	۵۸۱
پیکربندی SqlProfileProvider.....	۵۸۱
پیکربندی، خواندن و نوشتن داده‌های پروفایل.....	۵۸۲
ایجاد پروفایل‌های Anonymous.....	۵۸۳
ایجاد یک Provider سفارشی برای سیستم پروفایل.....	۵۸۴
چرا نباید مجوزهای دسترسی به منبع را بر اساس آدرس آن تعیین کنیم؟.....	۵۸۶
محدودکردن دسترسی با استفاده از آدرس‌های IP و دامنه‌ها.....	۵۸۷
خطرهای محدودکردن دسترسی با استفاده از آدرس‌های IP و دامنه‌ها.....	۵۸۷
نتیجه‌گیری.....	۵۸۸
فصل ۲۰: قرار دادن پروژه بر روی سرور.....	۵۸۹
آماده‌سازی پروژه برای انتشار.....	۵۸۹
تشخیص خطا در Viewها، پیش از انتشار پروژه.....	۵۸۹
پیکربندی کامپایل پویا.....	۵۹۰
انتشار پروژه بر روی سرور با استراتژی Bin Deployment.....	۵۹۱
تغییر Web.config، با توجه به حالت کامپایل.....	۵۹۱

۵۹۳	آشنایی با ساختار تبدیل
۵۹۴	قرار دادن تگ‌های مربوط به پیکربندی
۵۹۶	حذف تگ‌های مربوط به پیکربندی
۵۹۷	مقداردهی و حذف صفت‌ها
۵۹۸	جایگزینی تگ‌ها
۵۹۸	استفاده از صفت Locator
۶۰۱	قرار دادن پایگاه داده‌ی پروژه بر روی سرور
۶۰۳	آشنایی با مبانی IIS
۶۰۳	آشنایی با مفهوم «وب سایت‌ها»
۶۰۳	آشنایی با مفهوم Virtual Directory
۶۰۴	آشنایی با مفهوم Application Pool
۶۰۴	تنظیمات Binding مختلف برای سایت‌ها
۶۰۵	آماده‌سازی سرور برای انتشار
۶۰۶	سایت خود را باید کجا مستقر کنم؟
۶۰۷	انتشار یک پروژه
۶۰۷	انتشار پروژه با کپی کردن فایل‌های آن
۶۰۸	انتشار پروژه با قابلیت Deployment Package
۶۰۸	ایجاد Deployment Package
۶۰۹	استفاده از Deployment Package
۶۱۱	انتشار پروژه با قابلیت One-Click Publishing
۶۱۲	نتیجه‌گیری
۶۱۳	بخش چهارم: قابلیت‌های پیشرفته
۶۱۵	فصل ۲۱: BUNDLE‌ها
۶۱۵	آماده‌سازی مثال نمونه
۶۱۵	اضافه کردن بسته‌های NuGet
۶۱۵	ایجاد مدل و کنترلر
۶۱۶	ایجاد Master Page و View
۶۱۹	رهگیری بارگذاری فایل‌های جاوا اسکریپت و CSS
۶۲۱	استفاده از Bundle‌های Style و Script
۶۲۱	تعریف Bundle‌ها
۶۲۴	اعمال Bundle‌ها
۶۲۶	بهینه‌سازی فایل‌های جاوا اسکریپت و CSS
۶۲۸	نتیجه‌گیری
۶۲۹	فصل ۲۲: ASP.NET WEB API
۶۲۹	Web API چیست؟

۶۲۹	چرا Web Api؟
۶۳۰	تفاوت Web API و WCF
۶۳۳	ایجاد یک پروژه‌ی Web API
۶۳۳	اضافه کردن مدل
۶۳۳	اضافه کردن کنترلر
۶۳۵	فراخوانی Web API از طریق مرورگر
۶۳۷	فراخوانی Web API با استفاده از کتابخانه‌ی jQuery
۶۳۷	بازیابی فهرستی از محصولات
۶۳۸	بازیابی یک محصول با استفاده از مشخصه‌ی آن
۶۳۸	اجرای پروژه
۶۳۹	آشنایی با مفهوم مسیریابی در Web API
۶۴۰	مشاهده‌ی درخواست ارسالی و پاسخ دریافتی
۶۴۱	مدیریت کدهای وضعیت در Web API
۶۴۲	بازیابی رکورد
۶۴۲	ایجاد رکورد
۶۴۳	آپدیت رکورد
۶۴۳	حذف یک رکورد
۶۴۴	نتیجه‌گیری
۶۴۵	فصل ۲۳: WEB API و برنامه‌های تک صفحه‌ای وب
۶۴۵	مفهوم برنامه‌های تک صفحه‌ای (Single-page Application)
۶۴۵	آماده‌سازی پروژه‌ی نمونه
۶۴۶	ایجاد مدل
۶۴۸	افزودن کتابخانه‌ها از طریق NuGet
۶۴۸	افزودن کنترلر
۶۴۹	افزودن Master Page و Viewها
۶۵۱	تعیین نقطه‌ی آغاز پروژه و تست برنامه
۶۵۲	استفاده از Web API
۶۵۲	ایجاد کنترلر Web API
۶۵۴	آزمایش صحت عملکرد کنترلر API
۶۵۵	آشنایی با نحوه‌ی عملکرد کنترلر API
۶۵۶	آشنایی با روش انتخاب متد اکشن مناسب توسط کنترلر API
۶۵۷	تناظر متدهای HTTP به متدهای اکشن
۶۵۸	استفاده از Knockout برای برنامه‌های تک صفحه‌ای
۶۵۹	افزودن کتابخانه‌های جاوا اسکریپت مورد نیاز به Master Page
۶۶۰	پیاده‌سازی بخش نمایش نوبت‌های رزرو شده

۶۶۲	تعریف توابع AJAX
۶۶۲	تعریف مدل
۶۶۳	تعریف انقیادها
۶۶۴	پردازش انقیادها
۶۶۴	تست انقیادها
۶۶۶	بهبود قابلیت حذف
۶۶۷	عادت به استفاده از سیتکس Knockout
۶۶۷	پیاپی سازی قابلیت ثبت نوبت جدید
۶۶۹	اصلاح مدل
۶۶۹	ایجاد المانهای تعاملی
۶۷۰	مدیریت رویدادها
۶۷۰	تست قابلیت ثبت نوبت جدید
۶۷۱	تکمیل پروژه
۶۷۱	ساده سازی کنترلر Home
۶۷۱	مدیریت رویت محتوا
۶۷۴	نتیجه گیری
۶۷۵	فصل ۲۴: ASP.NET MVC 5.1 & 5.2
۶۷۵	نصب ASP.NET MVC 5.1 & 5.2
۶۷۶	پشتیبانی از مقادیر شمارشی (Enums) در Viewها
۶۸۰	بهبود قابلیت مسیریابی بر مبنای صفت
۶۸۰	یادآوری تعریف قید برای مسیر
۶۸۰	قیدها در مسیریابی بر مبنای صفت
۶۸۱	ایجاد یک قید سفارشی برای مسیریابی
۶۸۴	پشتیبانی از کلاسهای فریمورک Bootstrap در قالبهای ویرایشی (Editor Templates)
	پشتیبانی از تعیین اعتبار نامحسوس (Unobtrusive) در سمت کلاینت برای صفتهای MinLength و MaxLength
۶۸۶	
۶۸۷	پشتیبانی از کلمه کلیدی this در Unobtrusive AJAX
۶۸۷	اینترفیس IDirectRouteProvider
۶۸۹	گروه بندی گزینه های کنترل های ListBox و DropDownList
۶۹۰	بهبود متدهای نمایش خطا
۶۹۱	امکان پاس دادن صفت های HTML به متد EditorFor با استفاده از شیء Dictionary
۶۹۱	نتیجه گیری

بخش نخست

معرفی ASP.NET MVC

ASP.NET MVC، مسیری کاملاً متفاوت برای توسعه‌گران وبی است که از بسترهایی که توسط مایکروسافت برای خلق صفحات مبتنی بر وب ایجاد شده است، استفاده می‌کنند. معماری منحصر به فرد، استفاده‌ی آسان از الگوهای طراحی (Design Patterns)، قابلیت تست برنامه و عدم پنهان‌سازی عملیاتی که در پشت صحنه برای پردازش و تولید صفحه‌ی وب انجام می‌پذیرد، همه و همه، ASP.NET MVC را به عنوان یک تکنولوژی جالب و جذاب در کانون توجه توسعه‌گران وب قرار داده است.

در بخش نخست این کتاب با مفهوم معماری ASP.NET MVC و قابلیت‌های برجسته‌ی آن آشنا می‌شوید.

فصل ۱

ایده‌ی اصلی

ASP.NET MVC، بستری برای ایجاد برنامه‌های مبتنی بر وب است که توسط میکروسافت ارائه شده و از کارایی و نظم موجود در معماری MVC (Model View Controller) و آخرین نظرات و تکنیک‌های موجود در تفکر Agile بهره می‌برد.

Agile مجموعه‌ای از ارزش‌ها و اصول، برای توسعه‌ی نرم‌افزارهای کارا توسط تیم‌های خود سازمانده می‌باشد. ارزش‌ها و اصول Agile در سال ۲۰۰۱ به‌وسیله‌ی ۱۷ نفر از استادان معتبر جهانی صنعت توسعه‌ی نرم‌افزار، در بیانیه‌ای با نام بیانیه‌ی «توسعه‌ی چابک» تنظیم و ارائه گردید. اساس و هدف این اصول و ارزش‌ها، ارائه‌ی نرم‌افزار و یا محصول کارا به مشتری می‌باشد.

ASP.NET MVC می‌تواند جایگزینی کامل برای ASP.NET Web Forms باشد و مزایای بسیاری را برای توسعه دهندگان وب به ارمغان آورده است. در این فصل در مورد هدف میکروسافت از تولید ASP.NET MVC و مقایسه‌ی آن با تکنولوژی‌های پیشین و جانشین آن خواهیم پرداخت.

تاریخچه‌ی کوتاهی از توسعه‌ی برنامه‌های مبتنی بر وب

برای درک قابلیت‌های ممتاز و اهداف طراحی ASP.NET MVC، ارزش خواهد داشت تا کمی در مورد تاریخچه‌ی توسعه‌ی برنامه‌های مبتنی بر وب صحبت کنیم. از گذشته تاکنون، تکنولوژی‌های توسعه‌ی وب میکروسافت، قدرتمندتر و متأسفانه پیچیده‌تر شده‌اند. در جدول ۱-۱ این تکنولوژی‌ها را مشاهده می‌کنید. هر یک از آنها برای رفع نواقصی که در تکنولوژی پیشین وجود داشت ایجاد شدند.

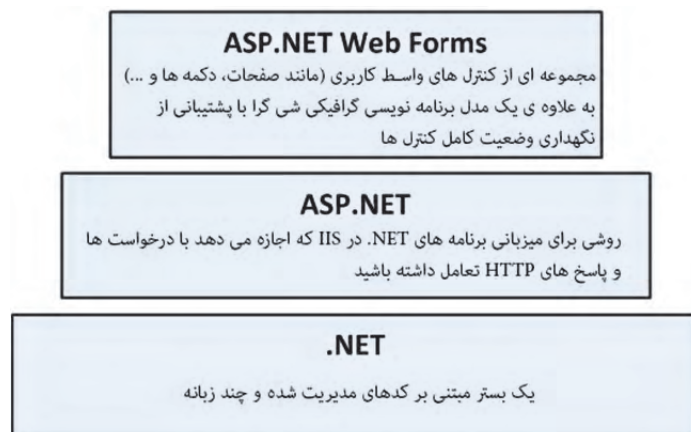
جدول ۱-۱: تکنولوژی‌های توسعه‌ی وب میکروسافت از ابتدا تاکنون

بازهی انتشار	نام تکنولوژی	مزایا	معایب
دوران دایناسورها!	Common Gateway Interface (CGI)	<ul style="list-style-type: none">آسانانعطاف‌پذیرتنها گزینه در زمان خودش	برنامه‌ای است که خارج از وب سرور اجرا می‌شود و به ازای هر درخواست، پروسس جداگانه‌ای را ایجاد می‌کند که باعث مصرف بیش از حد منابع سیستم عامل می‌شود.
زمانی که انسان با فلز آشنا شد!	Microsoft Internet Database Connector (IDC)	در وب سرور اجرا می‌شود	فقط واسطه‌ای است که اجازه‌ی انجام دستورات متداول SQL بر روی داده‌های پایگاه داده را می‌دهد. فرمت‌دهی داده‌ها به قالب HTML نیز با

معايب	مزایا	نام تکنولوژی	بازه انتشار
این تکنولوژی انجام می‌پذیرد			
<ul style="list-style-type: none"> • کامپایل و تفسیر دستورات در زمان اجرا • کدهای در هم آمیخته (اسپاگتی) 	چند منظوره	Active Server Pages (ASP)	1996
<ul style="list-style-type: none"> • مصرف زیاد پهنای باند • کدهای HTML غیر بهینه • سخت بودن تست کدها 	<ul style="list-style-type: none"> • کدهای کامپایل شده • کنترل‌هایی که وضعیت خود را نگه می‌دارند • امکانات زیاد و زیرساخت قدرتمند • دیدگاه جدیدی از برنامه‌نویسی شیء‌گرا 	ASP.NET Web Forms 1.0/1.1	2002-2003
---	---	ASP.NET Web Forms 2.0	2005
---	---	ASP.NET AJAX	2007
---	---	ASP.NET Web Forms 3.5	2008
---	---	ASP.NET MVC 1.0	2009
---	---	<ul style="list-style-type: none"> • ASP.NET MVC 2.0 • ASP.NET Web Forms 4.0 	2010
---	---	ASP.NET MVC 3.0	2011
---	---	<ul style="list-style-type: none"> • ASP.NET MVC 4.0 • ASP.NET Web Forms 4.5 	2012
---	---	<ul style="list-style-type: none"> • ASP.NET MVC 5.0 • ASP.NET Web Forms 4.5.1 	2013
---	---	<ul style="list-style-type: none"> • ASP.NET MVC 5.1 • ASP.NET MVC 5.2 	2014

ASP.NET Web Forms

زمانی که در سال ۲۰۰۲، ASP.NET Web Forms معرفی شد، دیدی که نسبت به توسعه‌ی برنامه‌های مبتنی بر وب وجود داشت کاملاً تغییر کرد و متحول شد. در شکل ۱-۱، دیدگاه مایکروسافت از ASP.NET Web Forms را مشاهده می‌کنید.



شکل ۱-۱: دیدگاه مایکروسافت از ASP.NET Web Forms

در ASP.NET Web Forms، تلاش مایکروسافت، مخفی‌سازی جزئیات پروتکل HTTP (که در ذات خود، وضعیت درخواست‌ها را نگهداری نمی‌کند) و HTML (که در آن زمان، توسعه‌گران، آشنایی زیادی با آن نداشتند) بود. این مخفی‌سازی، با تلاش برای ایجاد ظاهر صفحات وب از طریق ایجاد تعدادی کنترل با ساختاری سلسله‌مراتبی انجام شد. هر کنترل، وظیفه‌ی حفظ حالت خود را در میان ارسال درخواست به سمت سرور و دریافت پاسخ، با استفاده از قابلیت با عنوان View State بر عهده داشت. نحوه‌ی تولید کدهای HTML برای ایجاد ظاهر کنترل، از پیش تعریف شده بود و رویدادهای سمت کلاینت و سرور (همانند کلیک بر روی یک دکمه) نیز در موقع لزوم فراخوانی می‌شدند. در این حالت، ASP.NET Web Forms، تبدیل به پوششی می‌شود که سعی در ارائه‌ی محیطی همانند یک محیط کلاسیک تولید برنامه‌های Win App برای وب دارد.

هدف از تولید برنامه‌های ASP.NET Web Forms در آغاز این بود که توسعه‌گر، تجربه‌ای همانند تجربه‌ی تولید یک برنامه‌ی مبتنی بر ویندوز را داشته باشد، با جزئیات سطح پایین درخواست‌ها و پاسخ‌های پروتکل HTTP سر و کار نداشته باشد و محیطی با حفظ خودکار وضعیت کامل کنترل‌ها را تجربه کند. با ASP.NET Web Forms نیازی نیست تا از فاقد وضعیت بودن ذاتی صفحات وب و کنترل‌های آنها نگرانی داشته باشید. تنها کافی است تا کنترل‌ها را از جعبه‌ابزار بکشید و بر روی محیط طراحی رها سازید. بسیاری از کارها در سمت سرور به طور خودکار انجام می‌شود.

مشکلات ASP.NET Web Forms چیست؟

ایده‌ی آغارین ایجاد ASP.NET Web Forms خوب بود اما در حقیقت باعث پیچیدگی‌های بیشتری می‌شد. استفاده از ASP.NET Web Forms در پروژه‌های واقعی، رفته رفته برخی مشکلات آن را نمایان کرد:

- **حجم View State:** مکانیزم واقعی برای نگهداری وضعیت صفحه در حین رد و بدل شدن درخواست و پاسخ بین کلاینت و سرور، یک فیلد مخفی در صفحه است که به آن View State گفته می‌شود. حجم این فیلد می‌تواند گاهی به چندین کیلوبایت برسد. این حجم، موجب هدر رفتن پهنای باند و کاهش سرعت بارگذاری صفحات می‌گردد.
- **چرخه‌ی حیات صفحه:** روال‌های بسیاری در تولید صفحه‌ی وب از زمان ارسال درخواست تا هنگام دریافت پاسخ اجرا می‌شوند. این روال‌ها باعث کاهش سرعت اجرای صفحات شده و کدنویسی آنها نیز در صورتی که نیاز به استفاده از این روال‌ها باشد مشکل است.
- **برداشت اشتباه از جداسازی لایه‌ها:** مدل Code Behind که در ASP.NET برای جداسازی کدهای برنامه از کدهای HTML استفاده می‌شود، نخست ممکن است به نظر برسد که منطق برنامه را از لایه‌ی نمایش جدا می‌کند؛ اما به‌راستی باعث ادغام کدهای لایه‌ی نمایش (سر و کار داشتن با کنترل‌های صفحه در Code Behind) با منطق برنامه (بازیابی داده‌ها و انجام عملی بر روی آنها) می‌شود. نتیجه‌ی پایانی می‌تواند برنامه‌ای شکننده و پیچیده باشد.
- **کنترل محدود بر خروجی HTML:** خروجی کنترل‌های سمت سرور، کدهای HTML است؛ اما این کدها لزوماً آن HTMLی که مد نظر شما است نخواهند بود. پیش از ASP.NET Web Forms 4.0، خروجی کنترل‌های سمت سرور، با استانداردهای وب، فاصله‌ی زیادی داشت و استفاده از CSS برای فرمت‌دهی به آنها مشکل بود. این کنترل‌ها، idهای غیر قابل پیش‌بینی و نامأنوسی تولید می‌کردند که موجب می‌شد ارجاع به آنها از طریق JavaScript سخت باشد. این مشکلات در ASP.NET Web Forms 4.0 کاهش پیدا کرد؛ اما همچنان به‌دست آوردن خروجی HTML دلخواه به‌طور کامل میسر نیست.
- **مخفی‌سازی بسیار:** بسیاری از جزئیات پروتکل HTTP و HTML خروجی، پنهان شده است و در برخی موارد نیاز به انجام مهندسی معکوس برای آگاهی از جزئیات پشت صحنه و انجام رفتار دلخواه است. این حجم پنهان‌سازی برای توسعه‌گران وب حرفه‌ای، ملال‌آور است.
- **قابلیت تست‌پذیری پایین:** سازندگان ASP.NET Web Forms، پیش‌بینی نکرده بودند که تست برنامه‌ها که از آن با عنوان (Test Driven Development) یا توسعه‌ی تست محور) یاد می‌شود، به بخش مهمی از صنعت تولید نرم‌افزار تبدیل می‌شود. پس تعجیبی ندارد که معماری به هم پیچیده‌ای که توسط آنها تولید شده است، برای انجام آزمون‌های واحد (Unit Testing) مناسب نیست.

ASP.NET Web Forms به حرکت خود در طی سالیان ادامه داده است. ASP.NET Web Forms 2.0، قابلیت‌های بسیاری نسبت به نسخه‌ی پیشین خود معرفی کرد که تا ۷۰٪ باعث کاهش کدنویسی می‌شد. در سال ۲۰۰۷ که تب AJAX و Web 2.0 فراگیر شده بود، مایکروسافت، AJAX و AJAX Control Toolkit را معرفی کرد که حاوی ۴۰ کنترل برای غنی‌سازی صفحات وب و استفاده از امکانات AJAX در سمت کلاینت بود. جدیدترین نسخه‌ی ASP.NET Web Forms که در زمان نوشتن این کتاب، نسخه‌ی ۴.۵.۱ آن موجود است، خروجی استانداردتر و سازگارتر با قوانین HTML را ارائه داده است اما همچنان برخی محدودیت‌ها در این بین وجود دارد.

جایگاه توسعه‌ی وب در زمان حال

هم‌اکنون، وب از جهات مختلفی، رشد و جهش بسیار خوبی داشته است. سوای از AJAX، تکنولوژی‌های دیگری که در این عرصه مهم هستند را در ادامه، بررسی می‌کنیم.

استانداردهای وب و REST

در سال‌های اخیر، تمایل برای پذیرش استانداردهای وب افزایش پیدا کرده است. با توجه به ورود مرورگرهای جدید به بازار مرورگرها، رشد شبکه‌های اجتماعی و فراگیر شدن بیش از پیش اینترنت و دستگاه‌های ارتباطی، لزوم ایجاد خروجی مناسب که بر روی تمامی دستگاه‌ها (کامپیوترهای شخصی، تبلت‌ها، گوشی‌های تلفن همراه و ...) به‌درستی نمایش داده شود به عنوان عامل برتری یک سایت و افزایش سهم او در بازار مورد رقابت و از خواسته‌های کارفرمایان و از دغدغه‌های اصلی توسعه‌گران وب امروزی است. ارائه‌دهندگان بسترهای نرم‌افزاری تحت وب، به خوبی با این موارد آشنا هستند و سعی می‌کنند با ارائه‌ی بسترهای لازم و با توجه به استانداردها و نیازهای امروزی، ابزارهای مورد نیاز را برای تولید یک پروژه‌ی قدرتمند تحت وب توسط توسعه‌گران ارائه کنند.

REST (Representational State Transfer)، امروزه به عنوان معماری قابل توجه برای برقراری ارتباط میان برنامه‌های مختلف، در مقابل SOA (Service Oriented Architecture یا معماری سرویس‌گرا) مطرح است. در REST، منابع یک برنامه از طریق آدرس‌های وب، و اعمالی که باید بر روی آنها انجام شود از طریق متدهای HTTP، توصیف و عرضه می‌شود. برای نمونه، با استفاده از متد PUT می‌توان محصولی جدید را به آدرسی همچون `http://www.example.com/Products` ارسال، یا همگی مشتری‌هایی با نام Behrouz را با استفاده از متد DELETE از طریق آدرس `http://www.example.com/Customers/Behrouz` حذف کرد. برنامه‌های تحت وب امروزی، داده‌ها را تنها در قالب HTML ارسال نمی‌کنند؛ بلکه از فرمت‌های دیگری همچون JSON و XML نیز در مواقع لزوم برای ارتباط با تکنولوژی‌های دیگری همچون AJAX و Silverlight استفاده می‌کنند. این برقراری ارتباط، ذاتاً می‌تواند با استفاده از REST انجام شود اما نیاز به مکانیزمی برای پیاده‌سازی آن از طریق HTTP و آدرس‌های اینترنتی است که این کار به آسانی با ASP.NET Web Forms قابل انجام نیست.

Agile و توسعه‌ی تست محور

این تنها وب نیست که در دهه‌ی جاری حرکتی رو به جلو داشته است، بلکه توسعه‌ی نرم‌افزار نیز به سمت روش Agile پیش رفته است. روش Agile، از یک سری ارزش‌ها و اصول به منظور تولید نرم‌افزارهای با کیفیت استفاده می‌کند و بدین منظور از تعدادی نرم‌افزار نیز که معمولاً متن‌باز هستند بهره می‌برد.

TDD یا توسعه‌ی آزمایش محور و آخرین شکل آن که با نام Behavior Driven Development (BDD) یا توسعه‌ی رفتار محور) شناخته می‌شود، دو نمونه از روش‌هایی هستند که در تفکر Agile استفاده می‌شوند. ایده‌ای که در پشت TDD و BDD نهفته است، ابتدا طراحی نرم‌افزار با توصیف رفتارهای مورد نظر آن است. به این توصیف‌ها، "آزمایش" می‌گویند. در هر زمان می‌توان پایداری و صحیح بودن رفتار قسمت‌های مختلف برنامه را با اجرای آزمایش‌ها بررسی کرد. ابزارهای زیادی برای پیاده‌سازی TDD در .NET وجود دارند اما این ابزارها به خوبی با ASP.NET Web Forms سازگار نیستند:

- ابزارهای آزمایش واحد (Unit testing tools) اجازه می‌دهند تا درستی رفتار کلاس‌ها و متدها را به صورت مجزا بررسی کنید. اما این بررسی تنها در حالتی مؤثر خواهد بود که اصل جداسازی لایه‌ها به خوبی در برنامه رعایت شده باشد؛ بدین ترتیب، هر آزمایش می‌تواند به صورت مجزا و مستقل و بدون نیاز به وابستگی بیهوده به اجزای دیگر کلاس اجرا شود. متأسفانه تعداد کمی از برنامه‌هایی که با ASP.NET Web Forms ایجاد می‌شوند می‌توانند بدین طریق آزمایش شوند. دلیل این عدم آزمایش‌پذیری به خاطر معماری خاص ASP.NET Web Forms است که بسیاری از برنامه‌نویس‌ها را مجاب کرده است تا منطق برنامه را در رویدادهای فرم یا کنترل‌ها پیاده‌سازی کنند یا حتی از کنترل‌هایی استفاده کنند که مستقیماً کوئری را بر روی پایگاه داده اجرا می‌کنند. این به هم پیوستگی، مرگ TDD را رقم خواهد زد!
- ابزارهای خودکارسازی محیط کاربری (UI automation tools)، اجازه‌ی شبیه‌سازی رفتارهای کاربر را در هنگام کار با برنامه می‌دهند. در تئوری، این ابزارها می‌توانند با ASP.NET Web Forms استفاده شوند اما در صورتی که کوچک‌ترین تغییری در ساختار صفحه صورت گرفت، این ابزارها نمی‌توانند وظیفه‌ی خود را انجام دهند. در صورتی که دقت نکنید، Id کنترل‌ها و HTML تولیدی آنها در ASP.NET Web Forms با هر تغییری که در ساختار کنترل‌های صفحه می‌دهید ممکن است تغییر کنند و موجب بیهوده بودن تست‌هایی شوند که با ابزارهای خودکارسازی محیط کاربری انجام می‌پذیرند.

افرادی که به فرهنگ نرم‌افزاری متن‌باز علاقه دارند یا افراد و گروه‌های مستقل تولید نرم‌افزار که کیفیت محصولات نرم‌افزاری برای آنها با اهمیت است، ابزارهای متفاوتی را در بستر .NET. به منظور افزایش کیفیت و قابلیت اطمینان نرم‌افزارها تولید کرده‌اند. ابزارهایی همانند برنامه‌های آزمایش واحد مثل NUnit و xUnit، فریمورک‌های mock مثل Moq و Rhino Mocks، فریمورک‌های واگذاری مسئولیت (Inversion of Control) مانند Ninject و AutoFac، ابزارهای کنترل یکپارچه‌ی کیفیت نرم‌افزار که با عنوان Continuous Integration شناخته می‌شوند مانند Cruise Control و TeamCity، ابزارهای ORM مانند NHibernate، Subsonic و Entity Framework و همانند آنها از این قبیل هستند. آقای David Larabee، در ماه آوریل سال ۲۰۰۷، چنین ابزارهایی را، که در جامعه‌ی برنامه‌نویسان .NET. به منظور افزایش کیفیت و ابداع روش‌های جدید برای افزایش بهره‌وری کار تولید می‌شوند (و دیگر تفکرات این چنینی را)، ALT.NET نامید. هر سال، کنفرانسی نیز با همین نام برگزار می‌شود. در آدرس زیر، مقاله‌ی کاملی در مورد ALT.NET وجود دارد:

<http://msdn.microsoft.com/en-us/magazine/cc337902.aspx>

ASP.NET Web Forms، به خاطر معماری یکپارچه‌ی خود نمی‌تواند به خوبی با چنین ابزارها و تکنیک‌هایی تعامل پیدا کند؛ بنابراین از دید افرادی که قصد استفاده از چنین ابزارهایی را در پروژه‌های خود دارند، ASP.NET Web Forms، تکنولوژی مناسبی نیست.

نوشتن آزمایش واحد برای کلاس‌هایی که با یک سری از الگوریتم‌ها، مسائل ریاضی و امثال آن سر و کار دارند، ساده است. عموماً این نوع کلاس‌ها، وابستگی خارجی آنچنانی ندارند؛ اما در عمل، کلاس‌های ما ممکن است وابستگی‌های خارجی بسیاری پیدا کنند؛ برای نمونه، کار با پایگاه داده، اتصال به یک وب سرویس، دریافت فایل از اینترنت، خواندن اطلاعات از انواع فایل‌ها و غیره.

مطابق اصول آزمایش‌های واحد، یک آزمون واحد خوب باید ایزوله باشد. نباید به مرزهای سیستم‌های دیگر وارد شده و کارکرد سیستم‌های خارج از کلاس را بررسی کند.

این مثال ساده را در نظر بگیرید:

فرض کنید برنامه‌ی شما قرار است از یک وب سرویس، فهرستی از آدرس‌های IP یک کشور خاص را دریافت کند و در یک پایگاه داده‌ی محلی آنها را ذخیره نماید. به صورت متداول، این کلاس باید اتصالی را به وب سرویس گشوده و اطلاعات را دریافت کند و همچنین آنها را خارج از مرز کلاس در یک پایگاه داده ثبت کند. نوشتن آزمون واحد برای این کلاس، مطابق اصول مربوط، ممکن نیست. اگر کلاس آزمون واحد آن را تهیه نمایید، این آزمون، integration test نام خواهد گرفت زیرا باید از مرزهای سیستم عبور نماید.

همچنین یک آزمون واحد باید تا حد ممکن سریع باشد تا برنامه‌نویس از انجام آن بر روی یک پروژه‌ی بزرگ منصرف نگردد و ایجاد این اتصالات در خارج از سیستم، بیشتر سبب کندی کار خواهد شد. ابزارهایی که انجام چنین تست‌هایی را آسان می‌سازند، mock frameworks (چهارچوب‌های تقلید) نام دارند.

Ruby on Rails

در سال ۲۰۰۴، Ruby on Rails که پروژه‌ای متن باز بود، مورد توجه بسیاری از توسعه‌گران وب قرار گرفت و قواعد تولید برنامه‌های مبتنی بر وب را تغییر داد. Ruby on Rails، مفاهیم جالب و قابل توجهی را که امروزه ASP.NET MVC از آنها الهام گرفته است معرفی نمود و موجب شد تا بسترهای تولید برنامه‌های وب در آن زمان، در مقابل آن حرفی برای گفتن نداشته باشند.

Ruby on Rails (یا به بیان عامیانه‌ی آن، Rails)، از معماری MVC بهره می‌برد. استفاده از معماری MVC و تعامل زیبا با پروتکل HTTP و نه تقابل با آن، و معرفی مفهوم Convention over Configuration (قرارداد بر پیکربندی ارجحیت دارد) و شامل کردن یک ابزار ORM در درون خود، باعث محبوبیت بسیار Rails شد.

Rails نشان داد که استانداردهای وب و معماری REST می‌توانند به آسانی پیاده‌سازی شوند و همچنین توسعه‌ی مبتنی بر تفکر Agile و TDD نیز زمانی می‌توانند به خوبی پاسخگو باشند که بستر استفاده از آنها در محیط توسعه فراهم باشد.

Sinatra

Rails به سرعت طرفداران زیادی پیدا کرد اما زمان زیادی طول نکشید که فریمورک‌های مختلفی بر اساس معماری Ruby عرضه شدند. Sinatra یکی از آنها بود که در سال ۲۰۰۷ معرفی شد.

در آدرس زیر می‌توانید فهرستی از فریمورک‌های توسعه‌ی وب را ببینید:

http://en.wikipedia.org/wiki/Comparison_of_Web_application_frameworks

Node.js

یکی دیگر از موارد مهمی که هر روز بیش از پیش به آن اهمیت داده می‌شود و نقش پررنگ‌تری در توسعه‌ی برنامه‌های مبتنی بر وب پیدا می‌کند، زبان برنامه‌نویسی JavaScript است. اهمیت AJAX به ما اهمیت JavaScript را می‌آموزد و jQuery قدرت JavaScript را نمایش می‌دهد. جنگ مرورگرها برای ارائه‌ی موتور پردازش سریع‌تر برای JavaScript نشان از توجه و اهمیت ویژه‌ای است که وب به آن نشان می‌دهد. موتور متن‌باز V8 مرورگر Chrome شرکت Google که هم‌اکنون سریع‌ترین موتور JavaScript است، نمونه‌ای از این دست است. برای مشاهده‌ی سرعت موتور JavaScript مرورگرهای مختلف می‌توانید از آدرس زیر استفاده کنید:

<http://www.arewefastyet.com>

JavaScript به‌خودی خود به‌عنوان راه حلی برای اجرای سناریوها در سمت کلاینت شناخته شده است. Node.js، راه حلی است برای اجرای آن در سمت سرور! امروزه از JavaScript می‌توان برای کوئری گرفتن از پایگاه‌های داده‌ی غیر ارتباطی مانند CouchDB و MongoDB استفاده کرد و یا به‌عنوان یک راه حل چند منظوره برای اجرای دستورات در سمت سرور به وسیله‌ی Node.js بهره برد.

Node.js در سال ۲۰۰۹ معرفی شد و به سرعت مورد توجه قرار گرفت و با خود دو نوآوری را به ارمغان آورد:

- استفاده از JavaScript: توسعه‌گران، تنها نیازمند استفاده از یک زبان (JavaScript) برای انجام اعمال در سمت کلاینت، سرور و حتی کوئری گرفتن از پایگاه داده‌ای مانند CouchDB هستند.
- اعمال کاملاً نامتقارن: متدهایی که در کتابخانه‌ی Node.js وجود دارند، همگی به صورت نامتقارن اجرا می‌شوند؛ بنابراین، نیاز به انتظار برای پایان یافتن یک عملیات و سپس شروع یک عملیات جدید نیست. متدها پارامتری با نام callback دارند که متدی را به‌عنوان ورودی می‌پذیرد و پس از پایان عملیات، این متد به طور خودکار فراخوانی می‌شود. در این حالت، منابع سیستم به طور بهینه مصرف شده و می‌توان به تعداد زیادی درخواست همزمان، با موفقیت پاسخ داد. برخی بسترهای دیگر تنها تا ۱۰۰ درخواست به ازای هر CPU را پاسخگو هستند.

ASP.NET MVC از مفهومی با نام Asynchronous Controllers پشتیبانی می‌کند (با این مفهوم در فصل ۱۱ آشنا خواهید شد) که عملیات را به صورت نامتقارن اجرا و موجب افزایش توان پاسخ‌دهی CPUها به درخواست‌ها می‌شود. همچنین ASP.NET MVC به خوبی با کدهای JavaScriptی که در مرورگر اجرا می‌شوند تعامل دارد و زیرساخت‌های مناسبی را بدین منظور ارائه می‌دهد. با قابلیت‌های JavaScript در ASP.NET MVC در فصل‌های ۱۵، ۱۶ و ۱۷ آشنا خواهید شد.

مزایای اصلی ASP.NET MVC

ASP.NET Web Forms، یک محصول تجاری خوب است اما همان طور که گفته شد، دنیای توسعه‌ی وب تغییر کرده است و نیازهای متفاوتی را طلب می‌کند. هرچند که مایکروسافت تلاش‌های خوبی را برای بهبود قابلیت‌های ASP.NET Web Forms با نیاز روز توسعه‌گران به کار برده است اما این تلاش‌ها به‌دلیل معماری نامناسب اولیه‌ی آن، چندان موفق نبوده است.

در ماه اکتبر سال ۲۰۰۷، در کنفرانس ALT.NET که در شهر Austin ایالت Texas آمریکا برگزار شد، آقای Scott Guthrie که رهبری چندین تیم مرتبط با توسعه‌ی تکنولوژی‌های مرتبط با .NET را در مایکروسافت بر عهده دارد، ASP.NET MVC را معرفی کرد. ASP.NET MVC، تلاش مایکروسافت برای پاسخگویی به تکنولوژی‌های جذابی همانند Rails و نقدهایی بود که در مورد ASP.NET Web Forms مطرح می‌شد. در ادامه در مورد دلایل فائق آمدن این تکنولوژی بر محدودیت‌های ASP.NET Web Forms خواهید خواند.

معماری MVC

نکته‌ی مهمی که باید به آن توجه داشت، تفاوت بین الگوی MVC و فریمورک ASP.NET MVC است. الگوی MVC، جدید نیست. طرح اولیه‌ی MVC در سال ۱۹۷۳ در مرکز تحقیقات صنعتی Oslo در کشور نروژ توسط پروفیسور Trygve Reenskaug ایجاد شد. نام نخستین آن، Thing Model View Editor بود. نام آن سپس به Model View Controller یا به اختصار، MVC تغییر و در سال ۱۹۷۸ در زبان SmallTalk در شرکت Xerox PARC رسماً استفاده شد و امروزه به دلایل زیر به عنوان یک معماری محبوب برای طراحی برنامه‌های مبتنی بر وب مورد توجه است:

- نحوه‌ی تعامل کاربر با یک برنامه‌ی مبتنی بر معماری MVC، بسیار ساده است. کاربر، عملی را انجام می‌دهد و برنامه در پاسخ به کاربر، داده‌هایی را به سمت فرم برنامه ارسال و فرم را با این داده‌ها آپدیت و این چرخه ادامه پیدا می‌کند. این فرایند، یک روش بسیار مناسب برای برنامه‌های مبتنی بر وب است که نحوه‌ی تعامل کاربر با برنامه را به شکل تعدادی درخواست و پاسخ ساده‌ی پروتکل HTTP در می‌آورد.
- برنامه‌های مبتنی بر وب، ناگزیر به استفاده از تکنولوژی‌های مختلفی همچون پایگاه‌های داده، HTML، کدهای سمت سرور و همانند آن هستند که معمولاً این تکنولوژی‌ها توسط برنامه‌نویسان در لایه‌های مختلفی قرار می‌گیرند. الگوهایی که می‌توان از آنها برای لایه‌بندی این تکنولوژی‌ها استفاده کرد، ذاتاً در مفاهیم موجود در MVC وجود دارند.

ASP.NET MVC از الگوی MVC استفاده می‌کند و به خوبی، مفهوم جداسازی لایه‌ها را ارائه می‌دهد. در حقیقت، ASP.NET MVC شکل مدرن‌تری از الگوی MVC را پیاده‌سازی کرده است تا برای برنامه‌های مبتنی بر وب مناسب باشد. در مورد معماری MVC در فصل ۴ بیشتر می‌خوانید.

ASP.NET MVC یک رقیب جدی برای Ruby on Rails و بسترهای مشابه به حساب می‌آید و الگوی MVC را به یکی از قابلیت‌های محبوب دنیای .NET تبدیل کرده است. توسعه‌گرانی که Rails و ASP.NET MVC را تجربه کرده‌اند معتقدند که ASP.NET MVC، حرف‌ها و توانایی‌های بسیار بیشتری نسبت به Rails ارائه می‌دهد.

توسعه‌پذیری

کامپیوتر شما از اجزای مستقلی تشکیل شده است که این اجزا بر اساس استانداردی مشخص با یکدیگر تعامل دارند. می‌توانید به راحتی کارت گرافیک یا هارد دیسک را جدا کرده و آنها را با نوع دیگری کارت گرافیک یا هارد دیسک تعویض کنید. ASP.NET MVC نیز به همین شکل است و از تعدادی اجزای مستقل از هم مانند سیستم

Routing، View Engine، Controller Factory و همانند آنها تشکیل شده است که می‌توان آنها را با پیاده‌سازی دلخواه خود تعویض کرد.

ASP.NET MVC برای تعویض هر یک از اجزای خود، سه راه حل ارائه می‌دهد:

- از پیاده‌سازی پیش‌فرض آن جزء استفاده کرد که برای بیشتر سناریوها و پروژه‌ها مناسب است.
- از کلاس جزء مورد نظر ارث بُرد و پیاده‌سازی دلخواه را برای قسمت‌های مورد نظر آن جزء ارائه داد.
- جزء مورد نظر را به طور کامل از ابتدا با پیاده‌سازی interface یا کلاس abstract آن پیاده‌سازی کرد.

این قابلیت را می‌توان همانند قابلیت تغییر Provider Model (مانند Membership Provider) در ASP.NET 2.0 به بعد دانست اما با قابلیت‌های بیشتر. در مورد دلایل و نحوه‌ی تغییر و تعویض اجزای مختلف ASP.NET MVC در فصل ۷ خواهید خواند.

کنترل کامل بر HTML و HTTP

تیم سازنده‌ی ASP.NET MVC، به خوبی از اهمیت تولید یک خروجی HTML تمیز و استاندارد آگاه بودند. HTML helperهایی که در ASP.NET MVC وجود دارند، خروجی تمیز و استاندارد تولید می‌کنند. در مقایسه با ASP.NET Web Forms که در آن ایجاد قالب با CSS برای کنترل‌ها مشکل بود، در ASP.NET MVC این کار به آسانی انجام می‌پذیرد. کنترل‌های پیچیده‌ای مانند Date Picker یا منوها که از طریق HTML helperهای ASP.NET MVC قابل تولید نیستند را می‌توان به راحتی با استفاده از کنترل‌هایی که به وفور و رایگان از طریق توسعه‌گران با استفاده از کتابخانه‌ی jQuery یا YUI ایجاد شده‌اند در برنامه استفاده کرد. برنامه‌نویسان JavaScript، خوشحال خواهند شد که بدانند ASP.NET MVC به طور کامل با jQuery سازگار است و مایکروسافت آن را به عنوان قالب پیش‌فرضی که هنگام ایجاد یک پروژه‌ی ASP.NET MVC ایجاد می‌کند در پروژه قرار می‌دهد. در مورد jQuery در فصل ۱۷ خواهید خواند.

فرم‌هایی که توسط ASP.NET MVC تولید می‌شوند، فیلد مخفی View State را ندارند؛ بنابراین حجم بسیار کمتری نسبت به نمونه‌های مشابه خود در ASP.NET Web Forms دارند. تکنیک‌های مختلفی برای بارگذاری سریع‌تر صفحات استفاده می‌شوند که عدم وجود View State در معماری ASP.NET MVC، یکی از آنها است.

همانند Ruby on Rails، ASP.NET MVC نیز هم‌نوا با HTTP کار می‌کند و کنترل کاملی بر درخواست‌هایی که بین مرورگر و سرور رد و بدل می‌شوند دارد. توسعه‌گران حرفه‌ای وب، از چنین رفتاری استقبال خوبی خواهند کرد.

تست پذیری

معماری MVC به دلیل ذات جداگانه‌ی لایه‌های آن، امکان ایجاد برنامه‌هایی را با قابلیت نگهداری و آزمایش-پذیری بالا ارائه می‌دهد. اهمیت آزمایش‌پذیری ASP.NET MVC آن قدر بالا است که در هنگام ایجاد یک پروژه‌ی جدید مبتنی بر آن، از شما در مورد تمایل برای ایجاد یک پروژه‌ی آزمون در کنار پروژه ای که ایجاد می‌شود پرسش می‌شود. ASP.NET MVC، به خوبی با انواع فریمورک‌های تست مانند NUnit، xUnit و MSTest مایکروسافت سازگار است. آزمایش‌پذیری، تنها به انجام آزمایش‌های واحد ختم نمی‌شود. همان‌طور که پیش‌تر

خواندید، می‌توان از ابزارهای خودکارسازی محیط کاربری نیز برای ایجاد اسکریپت‌هایی که رفتار کاربر را در محیط برنامه شبیه‌سازی می‌کنند استفاده کرد. در هنگام استفاده از این ابزارها در محیط ASP.NET MVC، نیاز نیست تا نگران تغییر ساختار کدهای HTML صفحه، اسامی کلاس‌های CSS یا id تگ‌های HTML باشید که توسط ASP.NET MVC تولید می‌شوند.

سیستم مسیریابی قدرمند

شکل آدرس‌های اینترنتی با پیشرفت تکنولوژی‌های مبتنی بر وب تغییر کرده است. آدرس‌های نازیبایی همانند زیر:

`/App_v2/User/Page.aspx?action=show%20prop&prop_id=82742`

روز به روز کمتر می‌شوند و به جای آن می‌توان از آدرسی زیباتر و ساده‌تر همانند زیر استفاده کرد:

`/to-rent/ahwaz/2303-manategh-street`

دلایل خوبی برای ایجاد چنین آدرس‌های زیبایی و فراموشی فرمت قدیمی آدرس‌های اینترنتی وجود دارد. نخست اینکه آیا افزایش رتبه‌ی سایت‌های شما در موتورهای جست‌وجو برایتان مهم است؟ اگر پاسخ شما مثبت است بهتر است بدانید چنانچه آدرس شما حاوی کلماتی باشد که کاربران به دنبال آنها هستند، امتیاز بیشتر و در نتیجه، رتبه‌ی بالاتری توسط موتورهای جست‌وجو به سایت شما تعلق می‌گیرد. برای نمونه، اگر کاربری عبارت "rent in ahwaz" را جست‌وجو کند، احتمال اینکه آدرس قبل را به عنوان نخستین نتیجه‌ی جست‌وجوی خود مشاهده کند بسیار زیاد است. دوم اینکه درک و شاید حفظ چنین آدرس‌هایی توسط کاربران بسیار راحت‌تر صورت می‌پذیرد. دلیل سوم این است که زمانی که کاربری چنین آدرس‌های کوتاه و زیبایی را مشاهده می‌کند، مشتاق به قرار دادن آن لینک یا به اشتراک‌گذاری آن با دیگران یا حتی خواندن از پشت تلفن برای فردی دیگر خواهد بود. چهارم آنکه نیاز نیست تا ساختار پوشه‌ها و فایل‌های موجود در پروژه‌تان را کسی متوجه شود؛ با استفاده از سیستم مسیریابی می‌توان ساختار دسترسی به قسمت‌های مختلف سایت از طریق لینک‌ها را بدون خرابی لینکی که از قبل به منابع پروژه داده شده است، تغییر داد.

پیاده‌سازی چنین آدرس‌های زیبایی در ASP.NET Web Forms آسان نبود، اما ASP.NET MVC با استفاده از فضای نام `System.Web.Routing`، این قابلیت را به شکل پیش‌فرض ارائه می‌دهد. با این فضای نام، آزادی کاملی برای ایجاد انواع آدرس‌ها به شکل دلخواه خواهید داشت. در مورد مسیریابی یا همان `Routing`، در فصل ۸ به تفصیل بحث شده است.

ساخته شده بر مبنای بهترین قسمت‌های ASP.NET

بستر جاری میکروسافت برای تولید برنامه‌های مبتنی بر ASP.NET، بستری پخته و کارآمد است که از مجموعه‌ای از قابلیت‌ها و کامپوننت‌های منحصر به فرد برای ایجاد برنامه‌های تحت وب مؤثر و کارا بهره می‌برد.

نخستین نکته‌ی آشکاری که باید به آن اشاره کرد این است که از آنجا که ASP.NET MVC مبتنی بر پلت فرم .NET است، قابلیت نوشتن کد برای آن در هر زبان مبتنی بر پلت فرم .NET و استفاده از تمامی امکانات و ابزارها و افزونه‌های ساخته شده برای آن را خواهید داشت.

دوم اینکه برخی از قابلیت‌های آماده در ASP.NET Web Forms همانند Master Pages، Forms Authentication، Membership Provider، Profiles و Localization، می‌توانند در ASP.NET MVC نیز استفاده شوند و بدین ترتیب حجم کدهای تولیدی را تا حد زیادی کاهش دهند. همچنین می‌توانید از کنترل‌هایی در ASP.NET Web Forms که از View State استفاده نمی‌کنند و مبتنی بر PostBack نیستند نیز در ASP.NET MVC استفاده کنید.

API پیشرفته

.NET، یک فریمورک پیشرفته است که سبک خاصی از برنامه‌نویسی مدرن را معرفی نموده است. ASP.NET MVC از نسخه‌ی ۳ به بعد بر مبنای .NET 4.0 نوشته شده است؛ بنابراین می‌توان از تمامی امکانات معرفی شده در آن همچون Extension Methods، Lambda Expressions، Anonymous Types، Dynamic Types، LINQ و مانند آنها استفاده نمود.

ASP.NET MVC، متن باز است

بر خلاف تکنولوژی‌های پیشین مبتنی بر وب مایکروسافت، ASP.NET MVC به صورت متن باز ارائه شده است. می‌توانید کدهای آن را دانلود، اصلاح، کامپایل و نسخه‌ی خاص خود را داشته باشید! این مورد از آنجا بسیار حائز اهمیت است که می‌توانید Debug را برای کدهای ASP.NET MVC نیز در پروژه‌ی خود داشته باشید، از کارکرد اجزای داخلی ASP.NET MVC آگاه شوید، نحوه‌ی کدنویسی برنامه‌نویسان مایکروسافت را بیاموزید یا اگر باگی در آن پیدا کردید، پیش از آنکه صبر کنید تا مایکروسافت آن باگ را بر طرف کند، خود اقدام به بر طرف نمودن آن کنید. با این حال باید تغییراتی که در کدهای آن می‌دهید را به خاطر بسپارید و آن را در نسخه‌های جدیدی که ارائه می‌شود نیز اعمال کنید. ASP.NET MVC، نخست از مجوزی با عنوان MS-PL (Microsoft Public License) استفاده می‌کرد که مفاد آن از لینک زیر در دسترس است:

<http://www.opensource.org/licenses/ms-pl.html>

با توجه به این مجوز، قوانین زیر در مورد کد منبع ASP.NET MVC وجود داشتند:

- می‌توانستید کدهای ASP.NET MVC را به دلخواه تغییر دهید
- می‌توانستید نسخه‌ی تغییر یافته را منتشر کنید
- نسخه‌ی تغییر یافته نیاز نیست تا دوباره به صورت متن باز منتشر شود
- برای نسخه‌ی تغییر یافته نمی‌توانستید مجوزی جدید انتخاب کنید
- می‌توانستید نسخه‌ی تغییر یافته را بفروشید و کسب درآمد کنید