

مرجع کامل

ASP.NET MVC 4

مهندس بهروز راد

انتشارات پندار پارس

سرشناسه : راد، بهروز، ۱۳۶۴ -
 عنوان و نام پدیدآور : مرجع کامل ASP.NET MVC 4/بهرروز راد.
 مشخصات نشر : تهران : پندار پارس ، ۱۳۹۱.
 مشخصات ظاهری : ۵۷۶ص. : مصور، جدول.
 شابک : ۲۵۰۰۰۰ : ریال: 3-23-6529-600-978
 وضعیت فهرست نویسی : فیبا
 موضوع : صفحه‌های سرور فعال
 موضوع : وب--سایت‌ها--طراحی
 موضوع : ای.اس.پی. (پروتکل شبکه کامپیوتری)
 رده بندی کنگره : ۵۱۰۵TK/۵۱۳۹۱۸۸۸۵/۲۷۳ص/
 رده بندی دیویی : ۲۷۶/۰۰۵
 شماره کتابشناسی ملی : ۲۸۹۱۷۴۳

انتشارات پندارپارس



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 4
 ناشر : انتشارات پندار پارس
 ترجمه و تالیف : مهندس بهروز راد
 چاپ نخست : تابستان ۹۱
 شمارگان : ۱۱۰۰ نسخه
 طرح جلد : فرزانه روزبهانی
 لیتوگرافی، چاپ، صحافی : ترام‌سنج، صالحان، خیام

قیمت : ۲۵۰۰۰ تومان شابک : ۳-۲۳-۶۵۲۹-۶۰۰-۹۷۸



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

فهرست

۱.....	بخش نخست : معرفی ASP.NET MVC
۳.....	فصل ۱ ایده‌ی اصلی
۳.....	تاریخچه‌ی مختصری از توسعه‌ی برنامه‌های مبتنی بر وب
۴.....	ASP.NET Web Forms
۵.....	مشکلات ASP.NET Web Forms چیست؟
۶.....	جایگاه توسعه‌ی وب در زمان حل
۶.....	استانداردهای وب و REST
۶.....	Agile و توسعه‌ی تست محور
۸.....	Ruby on Rails
۸.....	Sinatra
۸.....	Node.js
۹.....	مزایای اصلی ASP.NET MVC
۹.....	معماری MVC
۱۰.....	توسعه‌پذیری
۱۰.....	کنترل کامل بر HTML و HTTP
۱۱.....	تست‌پذیری
۱۱.....	سیستم مسیریابی قدرمند
۱۱.....	ساخته شده بر مبنای بهترین قسمت‌های ASP.NET
۱۲.....	API پیشرفته
۱۲.....	ASP.NET MVC، متن باز است
۱۳.....	چه کسی باید از ASP.NET MVC استفاده کند؟
۱۳.....	مقایسه با ASP.NET Web Forms
۱۴.....	مهاجرت از ASP.NET Web Forms به ASP.NET MVC
۱۴.....	مقایسه با Ruby on Rails
۱۴.....	مقایسه با MonoRail
۱۵.....	قابلیت‌های جدید در ASP.NET MVC 3 & 4
۱۵.....	نتیجه‌گیری
۱۷.....	فصل ۲ آمادگی برای شروع
۱۷.....	آماده‌سازی سیستم توسعه
۱۷.....	نصب Visual Studio 2010
۱۸.....	نصب ASP.NET MVC 3
۱۹.....	نصب ابزارهای اختیاری
۲۰.....	کدهای ASP.NET MVC
۲۰.....	IIS Express
۲۰.....	SQL Server 2008 R2 Management Studio Express
۲۰.....	آماده‌سازی Server
۲۱.....	نصب IIS
۲۲.....	نصب ابزارهای اضافه
۲۲.....	پیکربندی Web Deployment
۲۴.....	منابع بیشتر برای یادگیری
۲۵.....	نتیجه‌گیری
۲۷.....	فصل ۳ نخستین پروژه‌ی ASP.NET MVC
۲۷.....	ایجاد یک پروژه‌ی جدید ASP.NET MVC
۲۹.....	اضافه کردن نخستین Controller
۳۰.....	آشنایی با Routeها
۳۱.....	پردازش در صفحات وب
۳۱.....	ایجاد یک View
۳۳.....	اضافه کردن خروجی پویا
۳۴.....	ایجاد یک پروژه‌ی ساده برای کار با داده‌ها
۳۴.....	آماده‌سازی
۳۵.....	طراحی مدل داده‌ها

۳۶	افزودن یک کلاس برای مدل
۳۶	ارتباط بین متدهای Action
۳۷	ایجاد یک متد Action
۳۸	ایجاد یک View نودار
۳۸	ایجاد ساختار View
۴۱	مدیریت فرم‌ها
۴۲	استفاده از Model Binding
۴۳	نمایش Viewهای دیگر
۴۴	افزودن تعیین اعتبار
۴۶	مشخص کردن کنترل‌های نامعتبر
۴۷	تکمیل پروژه
۴۸	نتیجه‌گیری
۴۹	فصل ۴ معماری MVC
۴۹	تاریخچه‌ی MVC
۵۰	مفهوم الگوی MVC
۵۰	آشنایی با Domain Model
۵۱	پیاده‌سازی معماری MVC در ASP.NET
۵۲	مقایسه‌ی MVC با الگوهای دیگر
۵۲	آشنایی با الگوی Smart UI
۵۳	آشنایی با معماری Model-View
۵۳	آشنایی با معماری سه لایه‌ی کلاسیک
۵۴	آشنایی با اشکال مختلف الگوی MVC
۵۴	آشنایی با الگوی Model-View-Presenter
۵۵	آشنایی با الگوی Model-View-View Model
۵۵	متدولوژی Domain Driven Design
۵۶	مدل کردن یک Domain
۵۶	زبان یکپارچه
۵۷	Simplification و Aggregate
۵۸	ایجاد Repositoryها
۵۹	ایجاد بخش‌های تفکیک شده
۶۰	استفاده از تزریق وابستگی (Dependency Injection)
۶۲	مثالی از تزریق وابستگی در ASP.NET MVC
۶۲	استفاده از Dependency Injection Container
۶۴	شروعی برای آزمایش واحد خودکار
۶۴	آشنایی با Unit Testing
۶۶	استفاده از TDD و منطق Red-Green-Refactor
۷۱	به سوی آیین TDD بشتابید!
۷۱	آشنایی با Integration Testing
۷۲	نتیجه‌گیری
۷۳	فصل ۵ قابلیت‌های کلیدی زبان
۷۳	قابلیت‌های کلیدی #C
۷۳	استفاده از Automatic Properties
۷۵	استفاده از Object Initializer و Collection Initializer
۷۷	استفاده از Extension Methods
۷۸	اعمال Extension Method به یک Interface
۸۰	ایجاد Extension Methodها برای فیلترکردن
۸۱	استفاده از عبارتهای Lambda
۸۲	شکل‌های دیگر عبارتهای Lambda
۸۲	استفاده از Type Inference
۸۳	استفاده از Anonymous Type
۸۳	استفاده از LINQ
۸۷	آشنایی با مفهوم کوثری‌های با تأخیر در LINQ
۸۹	استفاده‌ی دوباره از یک کوثری با تأخیر

۹۰ LINQ و اینترفیس <T>IQueryable
۹۰ آشنایی با سینتکس موتور Razor
۹۱ ایجاد پروژه
۹۱ ایجاد Model
۹۱ ایجاد Controller
۹۲ ایجاد View
۹۳ تنظیم مسیر پیش‌فرض
۹۳ بررسی یک View ساده در Razor
۹۳ کار با شیء Model
۹۴ استفاده از کد در Razor
۹۶ تعریف یک بلاک کد در Razor
۹۷ انتقال مقادیر به View با استفاده از شیء ViewBag
۹۸ کار با قالب‌ها
۹۹ کار بدون قالب‌ها
۱۰۰ نتیجه‌گیری
۱۰۱ فصل ۶ ابزارهای مهم برای ASP.NET MVC
۱۰۱ استفاده از Ninject
۱۰۳ ایجاد پروژه
۱۰۳ اضافه کردن Ninject
۱۰۴ شروع کار با Ninject
۱۰۵ ایجاد زنجیره‌ای از وابستگی‌ها
۱۰۶ تعیین مقادیر Property‌ها و پارامترها
۱۰۷ استفاده از Self-Binding
۱۰۸ برگشت نوع مشتق شده
۱۰۹ استفاده از شرط در معرفی کلاس‌ها
۱۱۰ استفاده از Ninject در ASP.NET MVC
۱۱۱ آزمایش واحد با Visual Studio
۱۱۱ ایجاد پروژه
۱۱۳ ایجاد آزمایش‌های واحد
۱۱۶ اجرای آزمایش‌های واحد و مواجه شدن با خطا
۱۱۷ پیاده‌سازی قابلیت
۱۱۸ استفاده از Moq
۱۱۸ افزودن Moq به پروژه
۱۱۸ ایجاد یک Mock با Moq!
۱۱۹ استفاده از قابلیت انتخاب متد توسط Moq
۱۱۹ تعیین مقدار برای پارامتر متدها توسط Moq
۱۲۰ برگشت یک نتیجه
۱۲۰ آزمایش واحد با Moq
۱۲۲ تأیید با Moq
۱۲۲ نتیجه‌گیری
۱۲۳ بخش دوم بررسی کامل ASP.NET MVC
۱۲۵ فصل ۷ نمای کلی پروژه‌های ASP.NET MVC
۱۲۵ کار با پروژه‌های ASP.NET MVC
۱۲۸ کنترلرها در قالب‌های Internet Application و Intranet Application
۱۲۹ آشنایی با مفهوم قراردادهای ASP.NET MVC
۱۲۹ قوانین نام‌گذاری کلاس‌های کنترلر
۱۲۹ قوانین نام‌گذاری View‌ها
۱۳۰ قوانین نام‌گذاری برای قالب‌ها
۱۳۰ دیدگاه پروژه‌های ASP.NET MVC
۱۳۰ ایجاد پروژه
۱۳۱ اجرای دیباگر Visual Studio
۱۳۲ توقف Debugger در Breakpoint
۱۳۲ استفاده از Breakpoint‌ها

۱۳۴.....	توقف Debugger در زمان رخ دادن خطا
۱۳۵.....	استفاده از قابلیت Edit and Continue
۱۳۵.....	فعال سازی قابلیت Edit and Continue
۱۳۶.....	اصلاح پروژه
۱۳۶.....	Edit and Continue
۱۳۷.....	دیبگ آزمایش های واحد
۱۳۷.....	استفاده از تزریق وابستگی در کل پروژه
۱۳۹.....	نتیجه گیری
۱۴۱.....	فصل ۸ URL ها، ROUTING و AREA ها
۱۴۱.....	معرفی سیستم مسیریابی
۱۴۲.....	اسمبلی سیستم مسیریابی
۱۴۲.....	ایجاد پروژه ی مسیریابی
۱۴۳.....	آشنایی با URL Patterns
۱۴۴.....	ایجاد و معرفی یک Route ساده
۱۴۵.....	تعریف مقادیر پیش فرض
۱۴۶.....	ایجاد URL Pattern های ثابت
۱۴۷.....	حق تقدم Route ها
۱۴۸.....	تعریف Segment Variable های سفارشی
۱۴۹.....	استفاده از Segment Variable های سفارشی به عنوان پارامترهای یک متد Action
۱۵۰.....	تعریف Segment Variable های اختیاری
۱۵۱.....	تعریف مسیرهای با طول متغیر
۱۵۲.....	اولویت بندی کنترلرها به وسیله ی Namespace ها
۱۵۴.....	ایجاد قید برای Route ها
۱۵۴.....	ایجاد قید با استفاده از عبارت های باقاعده
۱۵۴.....	ایجاد قید برای یک Route بر مبنای چند مقدار
۱۵۵.....	ایجاد قید برای یک Route با استفاده از متدهای HTTP
۱۵۵.....	تعریف یک قید سفارشی
۱۵۶.....	رفتار Route ها با فایل های موجود بر روی فایل سیستم
۱۵۹.....	دور زدن سیستم مسیریابی
۱۵۹.....	ایجاد آدرس های خروجی
۱۶۰.....	چه کاری نباید انجام دهیم؟ تعریف دستی آدرس ها!
۱۶۰.....	آماده سازی پروژه
۱۶۰.....	ایجاد آدرس های خروجی در View ها
۱۶۱.....	آشنایی با نحوه ی تطبیق Route برای ایجاد آدرس خروجی
۱۶۲.....	تعیین کنترلر دلخواه در ایجاد آدرس خروجی
۱۶۲.....	پاس دادن مقادیر اضافه
۱۶۳.....	آشنایی با مفهوم «استفاده ی دوباره از Segment Variable ها»
۱۶۴.....	کار با صفت های HTML
۱۶۴.....	ایجاد آدرس های کامل برای لینک ها
۱۶۵.....	ایجاد آدرس ها (و نه لینک ها)
۱۶۵.....	ایجاد لینک ها و آدرس ها با استفاده از اطلاعات Route
۱۶۶.....	ایجاد آدرس های خروجی در متدهای Action
۱۶۶.....	ایجاد آدرس از یک Route مشخص
۱۶۷.....	نقطه ضعف استفاده از نام Route ها
۱۶۷.....	سفارشی سازی سیستم مسیریابی
۱۶۷.....	ایجاد رفتار تطبیقی سفارشی برای Route ها
۱۶۹.....	سفارشی سازی سیستم مسیریابی برای آدرس های ورودی
۱۷۰.....	سفارشی سازی سیستم مسیریابی برای ایجاد آدرس های خروجی
۱۷۱.....	ایجاد یک مدیر Route سفارشی
۱۷۲.....	کار با Area ها
۱۷۲.....	ایجاد یک Area
۱۷۴.....	کار با یک Area
۱۷۶.....	حل مشکل تداخل نام کنترلرها

۱۷۷	ایجاد لینک برای متدهای Action در Areaها	۱۷۷
۱۷۷	طراحی مناسب آدرس ها	۱۷۷
۱۷۸	آدرس های خود را ساده و کاربرپسند طراحی کنید	۱۷۸
۱۷۹	GET و POST: انتخاب صحیح	۱۷۹
۱۷۹	نتیجه گیری	۱۷۹
۱۸۱	فصل ۹ کنترلرها و اکشن ها	۱۸۱
۱۸۱	معرفی Controller	۱۸۱
۱۸۱	آماده سازی پروژه	۱۸۱
۱۸۱	ایجاد یک کنترلر با استفاده از اینترفیس IController	۱۸۱
۱۸۲	ایجاد یک کنترلر با اثری از کلاس Controller	۱۸۲
۱۸۴	دریافت ورودی	۱۸۴
۱۸۴	استخراج داده ها از اشیاء Context	۱۸۴
۱۸۶	استفاده از پارامترها در متد Action	۱۸۶
۱۸۶	آشنایی با نحوه ی پُرشدن پارامترهای متد Action	۱۸۶
۱۸۷	آشنایی با پارامترهای اختیاری و اجباری	۱۸۷
۱۸۷	تعیین مقدار پیش فرض برای پارامتر	۱۸۷
۱۸۸	تولید خروجی	۱۸۸
۱۸۹	آشنایی با نتایج Action	۱۸۹
۱۹۲	برگشت نتیجه در قالب HTML با ارسال یک View	۱۹۲
۱۹۳	آزمایش واحد: پردازش یک View	۱۹۳
۱۹۴	تعیین یک View با استفاده از مسیر آن	۱۹۴
۱۹۵	انتقال داده ها از یک متد Action به یک View	۱۹۵
۱۹۵	آماده سازی یک مدل برای یک View	۱۹۵
۱۹۶	آزمایش واحد: مدل برای View	۱۹۶
۱۹۷	انتقال داده ها با ViewBag	۱۹۷
۱۹۷	آزمایش واحد: ViewBag	۱۹۷
۱۹۸	انتقال داده ها با ViewData	۱۹۸
۱۹۸	آزمایش واحد: ViewData	۱۹۸
۱۹۹	هدایت کاربر به آدرسی دیگر	۱۹۹
۱۹۹	الگوی POST/REDIRECT/GET	۱۹۹
۱۹۹	هدایت کاربر به یک آدرس واقعی	۱۹۹
۲۰۰	آزمایش واحد: هدایت کاربر به یک آدرس واقعی	۲۰۰
۲۰۰	هدایت کاربر به مسیری ثبت شده در سیستم مسیریابی	۲۰۰
۲۰۱	آزمایش واحد: مسیرهای ثبت شده در سیستم مسیریابی	۲۰۱
۲۰۱	هدایت کاربر به یک متد Action	۲۰۱
۲۰۲	نگهداری داده ها در هنگام هدایت کاربر	۲۰۲
۲۰۳	برگشت داده های متنی	۲۰۳
۲۰۴	آزمایش واحد: نتایج حاصل از فراخوانی متد Content	۲۰۴
۲۰۴	برگشت داده ها با فرمت XML	۲۰۴
۲۰۵	برگشت داده ها با فرمت JSON	۲۰۵
۲۰۶	ارسال فایل ها و داده های باینری	۲۰۶
۲۰۶	ارسال یک فایل	۲۰۶
۲۰۸	ارسال آرایه ای از بایت ها	۲۰۸
۲۰۸	ارسال داده ها در قالب Stream	۲۰۸
۲۰۸	آزمایش واحد: نتایج حاصل از فراخوانی متد File	۲۰۸
۲۰۹	برگشت خطاها و کدهای HTTP	۲۰۹
۲۰۹	ارسال یک کد HTTP مشخص	۲۰۹
۲۰۹	برگشت نتیجه ی ۴۰۴	۲۰۹
۲۰۹	برگشت نتیجه ی ۴۰۱	۲۰۹
۲۱۰	آزمایش واحد: کدهای HTTP	۲۱۰
۲۱۰	ایجاد یک نتیجه ی Action سفارشی	۲۱۰
۲۱۲	نتیجه گیری	۲۱۲
۲۱۳	فصل ۱۰ فیلترها	۲۱۳

۲۱۳.....	استفاده از فیلترها
۲۱۴.....	Attributeها در .NET: یادآوری
۲۱۵.....	معرفی چهار نوع فیلتر
۲۱۵.....	اعمال فیلترها به کنترلرها و متدهای Action
۲۱۶.....	فیلترهای مرتبط با اعطای مجوز دسترسی به منبع
۲۱۶.....	ایجاد یک فیلتر مرتبط با اعطای مجوز دسترسی به منبع
۲۱۸.....	استفاده از فیلتر موجود مجوز دسترسی به منبع
۲۱۹.....	ایجاد منطق سفارشی برای دسترسی به منابع
۲۲۰.....	ایجاد منطق سفارشی، هنگام نداشتن مجوز دسترسی به منبع
۲۲۱.....	استفاده از فیلترهای مدیریت خطا
۲۲۱.....	ایجاد یک فیلتر مدیریت خطا
۲۲۲.....	استفاده از فیلتر موجود مدیریت خطا
۲۲۴.....	استفاده از فیلترهای Action و Result
۲۲۴.....	پیاده‌سازی متد OnActionExecuting
۲۲۵.....	پیاده‌سازی متد OnActionExecuted
۲۲۶.....	پیاده‌سازی یک فیلتر Result
۲۲۸.....	استفاده از فیلتر موجود Action و Result
۲۲۹.....	استفاده از قابلیت‌های دیگر فیلترها
۲۲۹.....	استفاده از فیلترها بدون Attributeها
۲۳۰.....	استفاده از فیلترهای Global
۲۳۱.....	ترتیب اجرای فیلترها
۲۳۴.....	استفاده از فیلترهای موجود
۲۳۴.....	استفاده از فیلتر RequireHttps
۲۳۵.....	استفاده از فیلتر OutputCache
۲۳۷.....	نتیجه‌گیری
۲۳۹.....	فصل ۱۱ سفارشی‌سازی کنترلرها
۲۳۹.....	اجزای دخیل در روند اجرای درخواست
۲۳۹.....	ایجاد یک Controller Factory
۲۴۰.....	ایجاد یک Controller Factory سفارشی
۲۴۲.....	ثبت یک Controller Factory سفارشی
۲۴۲.....	کار با Controller Factory موجود
۲۴۲.....	ایجاد حق تقدم برای Namespaceها
۲۴۳.....	سفارشی‌سازی فرایند ایجاد کنترلر در DefaultControllerFactory
۲۴۳.....	استفاده از Dependency Resolver
۲۴۵.....	استفاده از یک Controller Activator
۲۴۶.....	Override کردن متدهای کلاس DefaultControllerFactory
۲۴۶.....	ایجاد یک Action Invoker سفارشی
۲۴۸.....	استفاده از Action Invoker موجود
۲۴۹.....	استفاده از یک نام متد Action سفارشی
۲۵۰.....	استفاده از Action Method Selection
۲۵۱.....	ایجاد یک Action Method Selector سفارشی
۲۵۲.....	مکانیزم کارکرد Action Method Selector
۲۵۲.....	مدیریت متدهای Actionی که وجود ندارند
۲۵۳.....	استفاده از Action Method Selectorها برای پشتیبانی از سرویس‌های REST
۲۵۴.....	معرفی افعال HTTP به روشی دیگر
۲۵۴.....	معرفی افعال HTTP به روشی دیگر در یک فرم ASP.NET MVC
۲۵۵.....	افزایش کارایی با کنترلرهای خاص
۲۵۵.....	استفاده از کنترلرهای Sessionless
۲۵۵.....	مدیریت Sessionها از طریق IControllerFactory سفارشی
۲۵۶.....	مدیریت Sessionها با استفاده از DefaultControllerFactory
۲۵۷.....	استفاده از کنترلرهای نامتقارن
۲۵۸.....	ایجاد یک کنترلر نامتقارن
۲۶۰.....	عملیات پشت صحنه و Threadهای بلاک شده

۲۶۰.....	ایجاد متدهای Async و Completed
۲۶۰.....	شروع عملیات نامتقارن
۲۶۱.....	پایان عملیات نامتقارن
۲۶۲.....	پاس دادن مقادیر از متد Async به متد Completed
۲۶۲.....	مدیریت حداکثر زمان مجاز برای اجرای عملیات نامتقارن
۲۶۳.....	توقف عملیات نامتقارن
۲۶۴.....	استفاده از الگوی برنامه نویسی نامتقارن .NET
۲۶۵.....	چه هنگام باید از کنترلرهای نامتقارن استفاده نمود؟
۲۶۵.....	نتیجه گیری
۲۶۷.....	فصل ۱۲ VIEW ها
۲۶۷.....	ایجاد یک View Engine سفارشی
۲۶۹.....	ایجاد یک IView سفارشی
۲۶۹.....	ایجاد یک پیاده سازی از ایتترفیس IViewEngine
۲۷۰.....	معرفی یک View Engine سفارشی
۲۷۲.....	بهبود سرعت نمایش صفحات با حذف View Engine های اضافی
۲۷۳.....	استفاده از View Engine های دیگر
۲۷۳.....	کار با موتور Razor
۲۷۳.....	آشنایی با نحوه پردازش View توسط موتور Razor
۲۷۵.....	استفاده از الگوی DI برای View های Razor
۲۷۶.....	پیکربندی مکان های جستجوی View ها
۲۷۸.....	افزودن محتویات پویا به View های موتور Razor
۲۷۹.....	استفاده از Inline Code
۲۷۹.....	Inline Code و اصل جداسازی لایه ها
۲۸۰.....	معرفی Namespace به یک View
۲۸۰.....	استفاده از دستور using در یک View
۲۸۱.....	معرفی Namespace در فایل Web.config
۲۸۱.....	آشنایی با مفهوم کدگذاری HTML در موتور Razor
۲۸۳.....	استفاده از View هایی با نوع dynamic به عنوان Model
۲۸۴.....	استفاده از HTML Helper ها
۲۸۴.....	ایجاد یک Inline HTML helper
۲۸۵.....	ایجاد یک External HTML helper
۲۸۷.....	استفاده از HTML Helper های موجود
۲۸۷.....	ایجاد تگ form
۲۸۷.....	ایجاد فرم هایی که به خودشان ارسال می شوند
۲۸۸.....	استفاده از Input Helper ها
۲۸۹.....	استفاده از متدهای HTML helper با نوع Strongly Typed
۲۹۰.....	اضافه کردن صفت ها به تگ HTML
۲۹۰.....	ایجاد تگ select
۲۹۲.....	ایجاد لینک ها و آدرس ها
۲۹۳.....	استفاده از WebGrid Helper
۲۹۷.....	استفاده از Chart Helper
۲۹۹.....	استفاده از Helper های دیگر موجود
۲۹۹.....	استفاده از Section ها
۳۰۲.....	بررسی وجود Section ها (روش نخست)
۳۰۲.....	بررسی وجود Section ها (روش دوم)
۳۰۳.....	استفاده از Partial View ها
۳۰۳.....	ایجاد یک Partial View
۳۰۴.....	استفاده از Partial View های Strongly Typed
۳۰۵.....	استفاده از Child Action ها
۳۰۶.....	ایجاد یک متد Child Action
۳۰۶.....	فراخوانی متد Child Action
۳۰۷.....	نتیجه گیری
۳۰۹.....	فصل ۱۳ قالب های MODEL

۳۰۹.....	استفاده از Templated View Helpers	
۳۱۳.....	استفاده از CSS در HTML تولیدی	
۳۱۴.....	استفاده از Model Metadata	
۳۱۴.....	استفاده از Metadata برای کنترل ویرایش و رؤیت	
۳۱۶.....	مستثنا کردن یک Property از تولید کد HTML برای آن	
۳۱۶.....	استفاده از Metadata برای تگ <label>	
۳۱۷.....	استفاده از Metadata برای مقادیر	
۳۱۹.....	استفاده از Metadata برای انتخاب قالب نمایش	
۳۲۱.....	اعمال Metadata به یک کلاس Buddy	
۳۲۲.....	کار با Propertyهایی از نوع Complex	
۳۲۳.....	سفارشی‌سازی سیستم تولید کدهای HTML	
۳۲۳.....	ایجاد یک قالب سفارشی برای حالت ویرایش	
۳۲۶.....	آشنایی با ترتیب جست‌وجوی قالب	
۳۲۶.....	ایجاد یک قالب سفارشی برای حالت نمایش	
۳۲۷.....	ایجاد یک قالب عمومی	
۳۲۸.....	جایگزینی قالب‌های موجود	
۳۲۹.....	استفاده از خصیصه ViewData.TemplateInfo	
۳۲۹.....	توضیحی در ارتباط با فرمت داده‌ها	
۳۳۰.....	کار با پیشوندهای تولیدی برای تگ‌های HTML	
۳۳۰.....	فراهم کردن اطلاعات اضافی برای یک قالب	
۳۳۱.....	آشنایی با سیستم Metadata Provider	
۳۳۲.....	ایجاد یک Model Metadata Provider سفارشی	
۳۳۴.....	سفارشی‌سازی Data Annotations Model Metadata Provider	
۳۳۵.....	نتیجه‌گیری	
۳۳۷.....	MODEL BINDING	فصل ۱۴
۳۳۷.....	آشنایی با Model Binding	
۳۳۸.....	استفاده از Model Binder پیش‌فرض	
۳۳۹.....	Model Binding برای انواع داده‌های ساده	
۳۴۰.....	حساسیت Model Binding به قوانین زبان	
۳۴۰.....	Model Binding برای نوع‌های Complex	
۳۴۱.....	ایجاد کدهای HTML برای Model Binding آسان	
۳۴۲.....	تعیین پیشوندهای سفارشی	
۳۴۳.....	انتخاب Propertyهای مورد نظر برای Binding	
۳۴۴.....	Binding برای آرایه‌ها و مجموعه‌ها	
۳۴۴.....	Binding برای مجموعه‌ای از انواع داده‌های سفارشی	
۳۴۵.....	Binding برای مجموعه‌ها با اندیس‌های غیر ترتیبی	
۳۴۶.....	Binding برای نوع داده‌ی Dictionary	
۳۴۶.....	فراخوانی Model Binding با کدنویسی	
۳۴۷.....	محدود کردن فرایند Model Binding به منبعی مشخص	
۳۴۸.....	مدیریت خطاها در فرایند Model Binding	
۳۴۹.....	استفاده از Model Binding برای دریافت فایل‌های آپلودی	
۳۵۰.....	سفارشی‌سازی سیستم Model Binding	
۳۵۰.....	ایجاد یک Value Provider سفارشی	
۳۵۱.....	ایجاد یک Model Binder با الگوی DI	
۳۵۲.....	ایجاد یک Model Binder سفارشی	
۳۵۴.....	ایجاد Model Binder Providerها	
۳۵۵.....	استفاده از صفت ModelBinder	
۳۵۶.....	نتیجه‌گیری	
۳۵۷.....	MODEL VALIDATION	فصل ۱۵
۳۵۷.....	ایجاد پروژه	
۳۵۹.....	تعیین اعتبار یک مدل به شکل صریح	
۳۶۱.....	ایجاد ظاهر مناسب برای Check Box ها	
۳۶۱.....	نمایش پیغام‌های خطای مرتبط با تعیین اعتبار مقادیر	

۳۶۴.....	نمایش پیغام‌های خطای در سطح Property
۳۶۶.....	تکنیک‌های دیگر تعیین اعتبار مقادیر
۳۶۶.....	تعیین اعتبار مقادیر از طریق Model Binder
۳۶۹.....	تعیین قوانین تعیین اعتبار داده‌ها از طریق Metadataها
۳۷۱.....	ایجاد یک Attribute سفارشی برای تعیین اعتبار در سطح Property
۳۷۲.....	ایجاد یک Attribute سفارشی برای تعیین اعتبار در سطح مدل
۳۷۴.....	ایجاد مدل‌هایی که خود را تعیین اعتبار می‌کنند
۳۷۵.....	ایجاد یک Validation Provider سفارشی
۳۷۸.....	معرفی یک Validation Provider سفارشی
۳۷۸.....	تعیین اعتبار در سمت کلاینت
۳۷۹.....	فعال‌سازی/غیر فعال‌سازی تعیین اعتبار در سمت کلاینت
۳۸۱.....	آشنایی با CDN
۳۸۲.....	استفاده از یک CDN برای کتابخانه‌های JavaScript
۳۸۲.....	استفاده از قابلیت تعیین اعتبار داده‌ها در سمت کلاینت
۳۸۵.....	آشنایی با نحوه‌ی کارکرد فرایند تعیین اعتبار در سمت کلاینت
۳۸۶.....	قابلیت درونی ASP.NET MVC برای تعیین اعتبار در سمت کلاینت
۳۸۶.....	سفارشی‌سازی فرایند تعیین اعتبار در سمت کلاینت
۳۸۶.....	ایجاد مستقیم صفت‌های مرتبط با تعیین اعتبار در کدهای HTML
۳۸۸.....	ایجاد صفت‌هایی که از تعیین اعتبار سمت کلاینت پشتیبانی می‌کنند
۳۹۰.....	ایجاد قوانین تعیین اعتبار سفارشی در سمت کلاینت
۳۹۲.....	آشنایی با Remote Validation
۳۹۴.....	نتیجه‌گیری
۳۹۵.....	AJAX ۱۶ فصل
۳۹۵.....	استفاده از قابلیت AJAX Unobtrusive در ASP.NET MVC
۳۹۵.....	ایجاد پروژه
۳۹۷.....	فعال‌سازی/غیر فعال‌سازی AJAX Unobtrusive
۳۹۸.....	استفاده از فرم‌های AJAX Unobtrusive
۴۰۰.....	آشنایی با نحوه‌ی کارکرد قابلیت AJAX Unobtrusive
۴۰۰.....	ASP.NET MVC بدون AJAX Unobtrusive
۴۰۱.....	تنظیمات AJAX
۴۰۱.....	اصل «تنزل مطبوع» (Graceful Degradation)
۴۰۲.....	آگاه‌سازی کاربر در هنگام ایجاد یک درخواست AJAX
۴۰۴.....	تأیید کاربر، پیش از ارسال درخواست AJAX
۴۰۴.....	ایجاد لینک‌های آژاکسی
۴۰۶.....	اصل «تنزل مطبوع» برای لینک‌های آژاکسی
۴۰۷.....	کار با Callbackها در AJAX
۴۱۰.....	کار با JSON
۴۱۰.....	افزودن پشتیبانی از فرمت JSON به کنترلر
۴۱۲.....	پردازش داده‌های JSON در مرورگر
۴۱۳.....	تشخیص درخواست‌های AJAX در متد Action
۴۱۴.....	ارسال داده‌ها با فرمت JSON به سرور
۴۱۶.....	نتیجه‌گیری
۴۱۷.....	JQUERY ۱۷ فصل
۴۱۷.....	ایجاد پروژه
۴۱۹.....	ارجاع به jQuery
۴۲۰.....	مدیریت نسخه‌های مختلف jQuery
۴۲۱.....	نوشتن کدهای jQuery
۴۲۲.....	اجرای jQuery در یک محیط ایزوله
۴۲۳.....	استفاده از Firefox
۴۲۴.....	استفاده از Chrome
۴۲۵.....	مبانی jQuery
۴۲۶.....	آشنایی با Selectorهای jQuery
۴۲۷.....	نکته‌ای در ارتباط با Id هر تگ

۴۲۷	استفاده از چند Selector به طور هم‌زمان
۴۲۷	استفاده از Attribute Selector ها
۴۲۸	استفاده از فیلترها در jQuery
۴۲۹	استفاده از فیلترهای محتوا
۴۳۰	استفاده از فیلترهای فرم
۴۳۱	آشنایی با متدهای jQuery
۴۳۱	انتظار برای بارگذاری صفحه
۴۳۲	متدهای مرتبط با کار با CSS در jQuery
۴۳۵	کار با DOM
۴۳۸	استفاده از رویدادها در jQuery
۴۳۹	استفاده از جلوه‌های بصری در jQuery
۴۴۱	استفاده از jQuery UI
۴۴۲	ارجاع به کتابخانه‌ی jQuery UI
۴۴۲	استفاده از ابزار ThemeRoller
۴۴۳	ایجاد دکمه‌هایی با ظاهری زیباتر
۴۴۴	استفاده از کامپوننت Slider
۴۴۶	نتیجه‌گیری
۴۴۷	بخش سوم ادامه‌ی توانایی‌های پروژه‌های ASP.NET MVC
۴۴۹	فصل ۱۸ امنیت و آسیب‌پذیری
۴۴۹	تمامی ورودی‌های برنامه می‌توانند جعل شوند
۴۵۰	HTTP چگونه کار می‌کند؟
۴۵۰	یک درخواست GET ساده
۴۵۰	یک درخواست POST همراه با کوکی‌ها
۴۵۱	جعل درخواست‌های HTTP
۴۵۲	Cross-Site Scripting و HTML Injection
۴۵۳	آشنایی با حملات XSS
۴۵۴	کد گذاری HTML از طریق موتور Razor
۴۵۵	تعیین اعتبار درخواست
۴۵۶	غیرفعال‌سازی قابلیت تعیین اعتبار درخواست
۴۵۷	کدگذاری مقادیر رشته‌ای برای JavaScript
۴۵۹	Session Hijacking
۴۵۹	محافظت از طریق بررسی آدرس IP درخواست‌دهنده
۴۶۰	محافظت با تنظیم خصیصه‌ی HttpOnly کوکی‌ها
۴۶۰	حملات Cross Site Request Forgery
۴۶۱	حمله
۴۶۱	دفاع
۴۶۲	جلوگیری از حملات CSRF در ASP.NET MVC
۴۶۴	SQL Injection
۴۶۴	حمله
۴۶۴	دفاع با استفاده از کوئی‌های پارامترار
۴۶۵	دفاع با استفاده از ORM ها
۴۶۵	استفاده‌ی امن از ASP.NET MVC
۴۶۵	متدهای Action را سهوا در معرض دسترسی قرار ندهید
۴۶۶	اجازه ندهید Model Binding، مقادیر Property های حساس را تغییر دهد
۴۶۶	نتیجه‌گیری
۴۶۷	فصل ۱۹ تصدیق هویت و مجوز دسترسی به منابع
۴۶۷	استفاده از تصدیق هویت Windows
۴۷۰	استفاده از روش تصدیق هویت بر اساس فرم‌ها
۴۷۱	تنظیمات Forms Authentication
۴۷۳	مدیریت لاگین
۴۷۳	استفاده از Forms Authentication، بدون کوکی
۴۷۴	استفاده از سیستم عضویت، نقش‌ها و پروفایل‌ها
۴۷۶	پیکربندی و استفاده از سیستم عضویت

۴۷۶.....	پی‌کربندی SqlMembershipProvider
۴۷۷.....	استفاده از SqlMembershipProvider با SQL Server نسخه‌ی Express
۴۷۸.....	آماده‌سازی دستی SQL Server
۴۷۹.....	مدیریت سیستم عضویت
۴۷۹.....	استفاده از ابزار Web site Administration (WAT)
۴۸۰.....	استفاده از قسمت NET Users در IIS
۴۸۲.....	ایجاد یک Provider سفارشی برای سیستم عضویت
۴۸۳.....	پی‌کربندی و استفاده از نقش‌ها
۴۸۴.....	پی‌کربندی SqlRoleProvider
۴۸۵.....	مدیریت نقش‌ها
۴۸۵.....	ایجاد یک Provider سفارشی برای سیستم نقش‌ها
۴۸۷.....	پی‌کربندی و استفاده از پروفایل‌ها
۴۸۷.....	پی‌کربندی SqlProfileProvider
۴۸۷.....	پی‌کربندی، خواندن و نوشتن داده‌های پروفایلی
۴۸۸.....	ایجاد پروفایل‌های Anonymous
۴۸۹.....	ایجاد یک Provider سفارشی برای سیستم پروفایلی
۴۹۱.....	چرا نباید مجوزهای دسترسی به منبع را بر اساس آدرس آن تعیین کنیم؟
۴۹۲.....	محدودکردن دسترسی با استفاده از آدرس‌های IP و دامنه‌ها
۴۹۲.....	خطرهای محدودکردن دسترسی با استفاده از آدرس‌های IP و دامنه‌ها
۴۹۳.....	نتیجه‌گیری
۴۹۵.....	فصل ۲۰ قرار دادن پروژه بر روی سرور
۴۹۵.....	آماده‌سازی پروژه برای انتشار
۴۹۵.....	تشخیص خطا در Viewها، پیش از انتشار پروژه
۴۹۶.....	پی‌کربندی کامپایل پویا
۴۹۷.....	انتشار پروژه بر روی سرور با استراتژی Bin Deployment
۴۹۷.....	تغییر Web.config، با توجه به حالت کامپایل
۴۹۹.....	آشنایی با ساختار تبدیلی
۵۰۰.....	قراردادن تگ‌های مربوط به پی‌کربندی
۵۰۲.....	حذف تگ‌های مربوط به پی‌کربندی
۵۰۳.....	مقداردهی و حذف صفت‌ها
۵۰۳.....	جایگزینی تگ‌ها
۵۰۴.....	استفاده از صفت Locator
۵۰۶.....	قراردادن پایگاه داده‌ی پروژه بر روی سرور
۵۰۸.....	آشنایی با میانی IIS
۵۰۸.....	آشنایی با مفهوم «وب سایت‌ها»
۵۰۹.....	آشنایی با مفهوم Virtual Directory
۵۰۹.....	آشنایی با مفهوم Application Pool
۵۰۹.....	تنظیمات Binding مختلف برای سایت‌ها
۵۱۰.....	آماده‌سازی سرور برای انتشار
۵۱۱.....	سایت خود را باید کجا مستقر کنم؟
۵۱۱.....	انتشار یک پروژه
۵۱۲.....	انتشار پروژه با کپی‌کردن فایل‌های آن
۵۱۲.....	انتشار پروژه با قابلیت Deployment Package
۵۱۲.....	ایجاد Deployment Package
۵۱۳.....	استفاده از Deployment Package
۵۱۵.....	انتشار پروژه با قابلیت One-Click Publishing
۵۱۶.....	نتیجه‌گیری
۵۱۷.....	بخش چهارم قابلیت‌های ASP.NET MVC 4
۵۱۹.....	فصل ۲۱ ASP.NET MVC 4
۵۱۹.....	تغییرات صورت گرفته در قالب پروژهی پیش‌فرض
۵۲۰.....	قابلیت Adaptive Rendering با استفاده از Media Queries در CSS
۵۲۲.....	Media Queries در CSS
۵۲۳.....	نمایش Viewها بر اساس نوع دستگاه درخواست‌کننده

انتخاب View در زمان اجرا با قابلیت DisplayModes	۵۲۳
استفاده از حالت Mobile قابلیت DisplayModes	۵۲۳
تست دستی قابلیت DisplayModes برای دستگاه‌های مختلف	۵۲۳
ایجاد DisplayMode های سفارشی	۵۲۵
خلاقیت با قابلیت DisplayModes	۵۲۶
دادن اختیار به کاربر برای تغییر DisplayMode ها	۵۲۶
ادغام و کاهش حجم فایل‌های CSS و JavaScript	۵۲۹
صفت جدید AllowAnonymous	۵۳۲
انتقال تنظیم‌های آغازین به کلاس‌های مجزا	۵۳۳
قراردادن آسان کنترلرها در پوشه‌ی دلخواه	۵۳۴
کتابخانه‌ی Knockout	۵۳۴
بهبودهای صورت گرفته در موتور Razor	۵۳۴
عبارت "~/"	۵۳۴
تفسیر خودکار عبارت "~/"	۵۳۴
توسط موتور Razor	۵۳۴
صفت‌های شرطی	۵۳۵
نتیجه‌گیری	۵۳۶
فصل ۲۲ ASP.NET WEB API	۵۳۷
Web API چیست؟	۵۳۷
چرا Web Api؟	۵۳۷
تفاوت WCF و Web API	۵۳۷
ایجاد یک پروژه‌ی Web API	۵۴۰
اضافه کردن مدل	۵۴۱
اضافه کردن Controller	۵۴۲
فراخوانی Web API از طریق مرورگر	۵۴۴
فراخوانی Web API با استفاده از کتابخانه‌ی jQuery	۵۴۶
بازیبی فهرستی از محصولات	۵۴۷
بازیبی یک محصول با استفاده از مشخصه‌ی آن	۵۴۷
اجرای پروژه	۵۴۷
آشنایی با مفهوم مسیریابی در Web API	۵۴۸
مشاهده‌ی درخواست ارسالی و پاسخ دریافتی	۵۴۹
مدیریت کدهای وضعیت در Web API	۵۵۰
بازیبی رکورد	۵۵۰
ایجاد رکورد	۵۵۱
آپدیت رکورد	۵۵۱
حذف یک رکورد	۵۵۲
نتیجه‌گیری	۵۵۲

درباره‌ی نویسنده

بهروز راد با بیش از ۱۲ سال تجربه‌ی برنامه‌نویسی با تمرکز بر بستر وب، خالق کتاب‌های "مرجع کامل Regular Expressions"، "مرجع کامل CSS" و "مرجع کامل Entity Framework" است. او در حال حاضر 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 اعمال 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 ۹۱

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 مایکروسافت 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.

مجموعه‌ای از ارزش‌ها و اصول، برای توسعه‌ی نرم‌افزارهای کارا توسط تیم‌های خود سازماندهی می‌باشد. ارزش‌ها و اصول Agile در سال ۲۰۰۱ به وسیله 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 توسعه‌دهندگان وب به ارمغان آورده است. در این فصل در مورد هدف مایکروسافت از تولید ASP.NET MVC و مقایسه‌ی آن با تکنولوژی‌های پیشین و جانشین آن خواهیم پرداخت.

تاریخچه‌ی مختصری 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 مایکروسافت، قدرتمندتر و متأسفانه پیچیده‌تر شده‌اند. در جدول ۱-۱ این تکنولوژی‌ها را مشاهده می‌کنید. هر یک از آنها برای رفع نواقصی که در تکنولوژی قبلی وجود داشت ایجاد شدند.

جدول ۱-۱: تکنولوژی‌های توسعه‌ی وب مایکروسافت 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 این تکنولوژی انجام می‌پذیرد
1996	Active Server Pages (ASP)	چند منظوره	<ul style="list-style-type: none">کامپایل و تفسیر دستورات در زمان اجراکدهای در هم آمیخته (اسپاگتی)

بازهی انتشار	نام تکنولوژی	مزایا	معایب
2002-2003	ASP.NET Web Forms 1.0/1.1	<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 	<ul style="list-style-type: none"> • مصرف زیاد پهنای باند • کدهای HTML غیر بهینه • سخت بودن تست کدها
2005	ASP.NET Web Forms 2.0	---	---
2007	ASP.NET AJAX	---	---
2008	ASP.NET Web Forms 3.5	---	---
2009	ASP.NET MVC 1.0	---	---
2010	<ul style="list-style-type: none"> • ASP.NET MVC 2.0 • ASP.NET Web Forms 4.0 	---	---
2011	ASP.NET MVC 3.0	---	---
2012	<ul style="list-style-type: none"> • ASP.NET MVC 4.0 • ASP.NET Web Forms 4.5 	---	---

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 میکروسافت، 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 قابلیت‌های 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. فقط 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 اولیه‌ی 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 اعمالی 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، داده‌های 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 تست پذیری 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 تعجبی 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 آن موجود است، خروجی استانداردتر و سازگارتر با قوانین HTML را ارائه داد 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 عملیاتی 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 فقط 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 فقط 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 تست محور و آخرین شکل آن که با نام BDD (Behavior Driven Development) 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، "تست" می‌گویند. در این حالت، در هر زمان می‌توان پایداری و صحیح بودن رفتار قسمت‌های مختلف برنامه را با اجرای تست‌ها 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 تست 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 تست شوند. دلیل این عدم تست‌پذیری 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 کوثری 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 در پروژه‌های خود دارند، ASP.NET Web Forms تکنولوژی مناسبی نیست.

نوشتن آزمون 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 آزمایشات 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 خواهند 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 پُر رنگ‌تری 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.arenefastyet.com>

JavaScript به خودی 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 بُرد.

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. هر چند که مایکروسافت تلاش‌های خوبی را برای بهبود قابلیت‌های ASP.NET Web Forms با نیاز روز توسعه‌گران به کار برده است اما این تلاش‌ها به دلیل 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 پروفسور 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 به‌عنوان 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 استاندارد 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 بد 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 همونوا 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.

تست‌پذیری

معماری MVC به دلیل ذات جداگانه‌ی لایه‌های آن، امکان ایجاد برنامه‌هایی را با قابلیت نگهداری و تست‌پذیری بالا ارائه می‌دهد. اهمیت تست‌پذیری MVC ASP.NET آن قدر بالا است که در هنگام ایجاد یک پروژه‌ی جدید مبتنی بر آن، از شما در مورد انتخاب فریمورک دلخواه برای تست برنامه پرسش می‌شود. ASP.NET MVC، به خوبی با انواع فریمورک‌های تست مانند Nunit، xUnit، MSTest و میکروسافت سازگار است. تست‌پذیری، تنها به انجام آزمایش‌های واحد ختم نمی‌شود. همان‌طور که قبلاً 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 جستجو برایتان مهم است؟ اگر پاسخ شما مثبت است بهتر است بدانید در صورتی که آدرس شما حاوی کلماتی باشد که کاربران به دنبال آنها هستند، امتیاز بیشتر و در نتیجه، رتبه‌ی بالاتری توسط موتورهای جستجو به سایت شما تعلق می‌گیرد. برای نمونه، اگر کاربری عبارت "rent in ahwaz" را جستجو کند، احتمال اینکه آدرس قبل را به عنوان اولین نتیجه‌ی جستجوی 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 از قبیل 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 مطلع 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 می‌کند 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
- می‌توانستید به نسخه‌ی تغییر یافته نامی جدید اختصاص دهید اما نمی‌توانستید از علامت تجاری خود برای آن استفاده کنید
- مایکروسافت، کدهای شما را به شرطی که در نسخه‌ی آتی بگنجانند نمی‌پذیرفت و تنها کدهایی را که توسط برنامه‌نویسان شرکت مایکروسافت یا تیم تضمین کیفیت آن نوشته می‌شدند قبول می‌کرد

ملاحظه می‌کنید که مجوز MS-PL، محدودیت‌های بسیاری دارد. اما خوشبختانه در تاریخ ۲۷ مارس ۲۰۱۲، مایکروسافت، مجوز کد منبع ASP.NET MVC را از MS-PL به Apache 2.0 تغییر داد و همچنین موتور Razor و یک سری موارد دیگر هم متن‌باز شدند. این تغییرات به این معنا خواهند بود که پروژه از حالت فقط خواندنی MS-PL، به حالت متداول یک پروژه - متن‌باز که شامل دریافت تغییرات و وصله‌ها از جامعه‌ی برنامه‌نویس‌ها است، تغییر کرده است. مشخصات کلیدی مجوز Apache 2.0، به شرح زیر است:

- به کار مشتق شده از پروژه‌ی اصلی می‌توان نامی دیگر اختصاص داد
- نیاز نیست تا کار مشتق شده حتماً متن‌باز باقی بماند
- می‌توان برای کار مشتق شده، مجوز جدیدی انتخاب کرد
- می‌توان کار مشتق شده را فروخت و کسب درآمد کرد

کدهای ASP.NET MVC، از آدرس زیر، قابل دریافت و مرور به صورت آنلاین است:

<http://aspnet.codeplex.com>

چه کسی باید از ASP.NET MVC استفاده کند؟

همانند هر تکنولوژی جدید دیگر، صرفاً وجود ASP.NET MVC، دلیلی برای استفاده از آن نیست. تا اینجا در مورد برخی قابلیت‌ها و همچنین مقایسه‌ی آن با پلت‌فرم‌های دیگر خواندید. سعی شده است که این مقایسه‌ها منصفانه و بی‌غرض باشد. در بخش‌های پیش رو، ASP.NET MVC را با پلت‌فرم‌های دیگر مقایسه می‌کنیم. به این نکته دقت داشته باشید که در هنگام انتخاب یک پلت‌فرم برای تولید برنامه‌های مبتنی بر وب، باید میزان مهارت و دانش افراد تیم در مورد آن پلت‌فرم، میزان کاری که باید برای انتقال پروژه‌های موجود به پلت‌فرم جدید انجام شود و همچنین قابلیت اطمینان و میزان منابع در دسترس برای آن تکنولوژی را نیز در نظر بگیرید.

مقایسه با ASP.NET Web Forms

تا اینجا در مورد معایب و محدودیت‌های موجود در ASP.NET Web Forms و نحوه‌ی فائق آمدن ASP.NET MVC بر این محدودیت‌ها خواندید. هدف از بیان این مشکلات این نیست که فکر کنید ASP.NET Web Forms منسوخ شده یا از بین رفته است! مایکروسافت بارها بر این نکته تأکید کرده است که هر دوی این تکنولوژی‌ها در کنار یکدیگر با قدرت، توسعه داده و پشتیبانی خواهند شد و هیچ برنامه‌ای برای کنار گذاشتن ASP.NET Web Forms وجود ندارد. انتخاب بین هر یک از این دو، به استراتژی و اهداف شما نیز بر می‌گردد. به دو نکته‌ی زیر دقت کنید:

- در ASP.NET Web Forms، فرم‌های برنامه به طور کامل وضعیت خود را نگهداری می‌کنند و جزئیات پروتکل HTTP و HTMLی که توسط کنترل‌ها تولید می‌شوند را پنهان می‌کنند. این امکان با استفاده از View State و PostBackها پیاده‌سازی می‌شود. این حالت، برای تولید برنامه‌هایی به سبک برنامه‌های ویندوزی یا در اصطلاح Desktop based که با کشیدن و رها کردن کنترل‌ها بر روی فرم و نوشتن کد برای رویدادهای آنها می‌توان به سرعت فرمی را ایجاد کرد مناسب است.
- ASP.NET MVC، قابلیت ذاتی پروتکل HTTP در عدم توانایی در نگهداری وضعیت درخواست و فرم را با آغوش باز پذیرفته است و به جای آنکه در مقابل آن قرار بگیرد، با آن تعامل خوبی برقرار می‌کند. این قابلیت تعامل موجب می‌شود تا برنامه‌نویس دقیقاً بداند که یک برنامه‌ی مبتنی بر وب چگونه کار می‌کند. چنین درکی باعث تولید راه

حل های ساده، قدرتمند و مدرن در خلق برنامه های وب می شود. کدها خواناتر، با قابلیت نگهداری بالا و قابل توسعه خواهند بود.

مطمئناً شرایطی نیز پیش خواهد آمد که در آن، ASP.NET Web Forms به خوبی ASP.NET MVC و حتی بهتر از آن عمل می کند. برای نمونه، می توان به برنامه هایی اشاره کرد که در بستر شبکه ای داخلی یک شرکت اجرا می شوند و نمایش تعداد زیادی رکورد توسط آنها در صفحات مختلف به وفور وجود دارد یا فرم هایی که شامل مراحل چندگانه یا در اصطلاح Wizard هستند. قابلیت کشیدن و رها کردن کنترل ها در ASP.NET Web Forms در زمانی که نگران پهنای باند یا افزایش رتبه در موتورهای جستجو نیستید می تواند از قابلیت های برجسته ای آن محسوب شود.

اما اگر برنامه ای شما در اینترنت اجرا می شود، پهنای باند و سازگاری سایت با مرورگرهای مختلف و تست برنامه برایتان با اهمیت است و ASP.NET MVC می تواند شما را در این موارد راضی کند.

مهاجرت از ASP.NET Web Forms به ASP.NET MVC

خوشحال خواهید شد که بدانید اگر قصد انتقال پروژه ای بر مبنای ASP.NET Web Forms را به ASP.NET MVC داشته باشید، هر دوی این تکنولوژی ها می توانند به خوبی و خوشی در کنار هم و در یک پروژه استفاده شوند! بدین شکل می توان به اندازه ی کافی فرصت داشت تا به تدریج به ASP.NET MVC مهاجرت کرد.

مقایسه با Ruby on Rails

Rails را می توان به عنوان معیاری برای مقایسه ی پلت فرم های مختلف تولید برنامه های مبتنی بر وب در نظر گرفت. توسعه گران و شرکت هایی که حوزه و بستر کاری آنها بر اساس پلت فرم NET است، متوجه خواهند شد که قبول و یادگیری ASP.NET MVC آسان است. در مقابل، توسعه گران و شرکت هایی که با زبان Python یا Ruby در سیستم عامل های Linux یا Mac کار می کنند، استفاده از Rails برای آنها آسان تر خواهد بود. احتمال اینکه شخص یا گروهی از Rails به ASP.NET MVC یا عکس آن مهاجرت کند کم است. تفاوت های مهمی در حوزه ی این دو تکنولوژی با یکدیگر وجود دارد.

Rails پلت فرمی کل نگر است. تمامی ابزارهای لازم را در خود به طور پیش فرض جای داده است در حالی که ASP.NET MVC، ذاتاً همراه با خود ابزاری ندارد و از ابزارها و برنامه هایی که در NET وجود دارد یا توسط گروه های دیگر ساخته شده استفاده می کند. برای نمونه، ORM یی همراه با ASP.NET MVC وجود ندارد. ابزاری برای تست وجود ندارد. سیستمی برای انتقال پایگاه داده از جایی به جای دیگر وجود ندارد. این بدین دلیل است که NET، مجموعه ای غنی از تمامی ابزارهای لازم را برای ASP.NET MVC فراهم کرده است و برنامه نویس می تواند به دلخواه خود از هر یک از آنها استفاده کند. برای نمونه، برای ابزارهای ORM می توان از Subsonic، NHibernate، Entity Framework و امثال آنها استفاده کرد.

مقایسه با MonoRail

MonoRail، نخستین تلاش و پروژه ی خارج از شرکت مایکروسافت برای استفاده از معماری MVC در برنامه های ASP.NET و قسمتی از پروژه ی متن باز Castle بود که در سال ۲۰۰۳ معرفی گردید. MonoRail را می توان از جهات مختلف، نخستین نمونه ی ASP.NET MVC برشمرد.

اکنون، MonoRail رقیبی برای ASP.NET MVC به حساب نمی آید اما در زمان خود محبوبیت فراوانی داشت. پس از معرفی ASP.NET MVC، MonoRail رفته رفته طرفداران خود را از دست داد و توسعه گران وب در بستر NET، انرژی و وقت خود را صرف ASP.NET MVC کردند.