عنوان: **با ASP.MVC چه مزایایی را به دست خواهیم آورد**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۲:۸
آدرس: www.dotnettips.info
گروه‌ها: MVC

عموما در اکثر مطالب مقایسه‌ای بین وب فرم‌ها و ASP.NET MVC به جداسازی بهتر منطق کدها از فرم‌ها و قابلیت بهتر تهیه آزمون‌های واحد اشاره می‌شود. در این مطلب از دیدگاهی دیگر به این مساله خواهیم پرداخت؛ از لحاظ فنی و جدای از مسایل یاد شده، چه مزایای دیگری را می‌توان با استفاده از ASP.NET MVC نسبت به وب فرم‌ها به دست آورد؟

1 - آدرس‌های تمیزتر

در ASP.NET MVC به صورت پیش فرض از سیستم Routing موجود در زیر ساخت ASP.NET برای نمایش Urlهایی بدون پسوند استفاده می‌شود. همچنین این سیستم امکان تهیه آدرس‌هایی با سازگاری بهتر با موتورهای جستجو را نیز از ابتدا مدنظر داشته است.

بله. این زیر ساخت در اختیار وب فرم‌ها نیز هست؛ اما فرق است بین حالتی که از ابتدا مجبور شویم تمیزتر کار کنیم با زمانیکه این انتخاب را داریم و ... عموما هم از آن در وب فرم‌ها استفاده نمی‌شود.

2- عدم وابستگی الزامی به فایل‌های فیزیکی موجود در سیستم

کلیه درخواست‌ها در MVC برخلاف وب فرم‌ها در بدو امر به فایل‌های موجود در سیستم منتقل نمی‌شوند. درخواست‌ها به متدهای موجود در کنترلرها منتقل می‌شوند. همین مساله سبب می‌شود که آدرس‌ها الزاما به یک فایل فیزیکی موجود در سیستم اشاره نکنند. به این ترتیب می‌توان درخواست‌ها را بر اساس شرایط، به Viewهای مختلف هدایت کرد و نه اینکه هر درخواست ابتدا به یک view رسیده و سپس به متدی ارجاع داده شود.

این مساله از لحاظ امنیتی نیز مهم است. درخواست‌های رسیده به MVC سبب اجرای هیچ فرمی نخواهند شد. درخواست‌ها حتما باید از فیلتر یک کنترلر عبور کنند تا اجرایی شوند.

3- امکان مدیریت بهتر قسمت‌های مختلف سایت در پوشه‌های جداگانه

اگر به سورس اکثر سایت‌های مبتنی بر ASP.NET Web forms توجه کنیم، تمام فایل‌های آن‌ها در ریشه سایت قرار دارند منهای فایل‌های CSS و JS و تصاویر. در ASP.NET MVC از ابتدای کار، هر قسمت از سایت در پوشه‌های جداگانه‌ای قرار می‌گیرد و به این ترتیب مدیریت فایل‌ها و نظم دهی به آن‌ها ساده‌تر خواهد بود.

4- امکان تعریف تمام اجزای یک فرم یا view به صورت strongly typed

در ASP.NET MVC می‌توان یک کلاس را به یک فرم یا View نسبت داد. به این ترتیب کنترل‌های web forms تبدیل به خواص این کلاس در MVC خواهند شد. مزیت این امر امکان کنترل تمام اجزای فرم‌های سایت توسط کامپایلر است.

به این ترتیب اگر در طی یک حلقه، جدولی را ایجاد کنیم، تمام عناصر تشکیل دهنده این حلقه (چه کدهای آن و چه المان‌هایی که اطلاعات را در صفحه نمایش می‌دهند) نیز توسط کامپایلر قابل بررسی و خطایابی هستند.

5- مقدار دهی خودکار مدل متناظر با یک فرم یا View در ASP.NET MVC

روال متداول کار با وب فرم‌ها، قرار دادن تعدادی کنترل در صفحه و سپس دریافت دستی مقادیر آن‌ها در فایل code behind است. در MVC دیگر نیازی نیست تا این کارها را دستی انجام دهید. به یک فرم یا View کلاسی را انتساب خواهید داد. فریم ورک خواص آن‌را به صورت خودکار در حین ارسال به سرور مقدار دهی خواهد کرد. این مورد حتی در حین کار با jQuery Ajax نیز صادق است.

این مساله کار با ORMها را نیز ساده‌تر می‌کند. از این جهت که تمام آن‌ها نهایتا با یک سری مدل و کلاس کار خواهند کرد و تمام فیلدها و جداول به تعدادی کلاس و خاصیت تعریف شده در آن‌ها نگاشت می‌شوند.

به این ترتیب چون دیگر نیازی به ارجاع مستقیم به اشیاء بصری در فایل‌های code behind که در اینجا کنترلر نام گرفته‌اند نیست، نوشتن آزمون واحد برای این کلاس‌ها نیز به شدت ساده‌تر شده است.

6- ASP.NET MVC به همراه یک فرم ساز توکار ارائه می‌شود

اگر کسی به شما گفته است که سرعت کار با ASP.NET MVC پایین است به طور قطع دو فصل اول یک کتاب MVC را بیشتر مطالعه نکرده است. در MVC یک کلاس متناظر با فرمی را طراحی می‌کنید. توسط ابزار scaffolding همراه با VS.NET از روی این کلاس و مدل، با چند کلیک یک فرم تولید خواهد شد. فرمی که حتی مقدار دهی و انتساب عناصر بصری آن به کلاس متناظر با آن نیز خودکار است.
سرعت پیاده سازی یک برنامه با ASP.NET MVC به مراتب بیشتر است از کار با وب فرم‌ها.

7- حذف View State در MVC

از آنجائیکه فرم‌های ASP.NET Web forms از نوع strongly typed نیستند (در دات نت 4 و نیم اندکی بهبود در حد گریدهای آن حاصل شده البته)، برای اینکه پس از ارسال یک فرم به سرور باز هم کنترل‌های نمایش داده شده در صفحه همان مقادیر قبلی را نمایش دهند، مکانیزم View State به همراه ذخیره سازی اطلاعات فرم در فیلدهای مخفی فرم جاری طراحی شد.
در MVC نیازی به این مکانیزم نیست. زیرا فقط کافی است که اطلاعات مدل را مجددا به View ارسال کنیم. نمایش و انتساب نهایی آن در اینجا خودکار است. بنابراین نیازی به View State وجود ندارد.

8- کنترل بهتر بر روی اعتبار سنجی اطلاعات دریافتی

در وب فرم‌ها اگر بخواهیم سیستم اعتبارسنجی آن‌را غیرفعال کنیم، مثلا برای دریافت html از کاربر، نیاز است کلا آن‌را از کار بیندازیم (یا در سطح فرم یا در سطح کل برنامه). در MVC می‌توان این اعتبار سنجی را تنها در سطح یک خاصیت که قرار است اطلاعات HTML ایی را دریافت کند، غیرفعال کرد؛ نه برای کل فرم و نه در سطح کل برنامه.

9- امکان استفاده از فرم‌های و Viewهای Razor بجای موتور وب فرم‌ها

در وب فرم‌ها تا این زمان فقط محدود به تنها موتور نمایشی مخصوص به آن هستیم. اما در MVC این محدودیت برداشته شده و تا به حال چندین و چند View engine در این بین توسط مایکروسافت و سایر برنامه نویس‌ها طراحی شده است. مهم‌ترین آن‌ها Razor است که تمام برنامه نویس‌های MVC پس از مدتی به روان بودن و طراحی طبیعی و عالی آن اعتراف دارند.

10- امکان تعریف بیش از یک فرم در صفحه

طراحی ASP.NET Web forms از روز اول آن محدود به یک فرم در صفحه بوده است. این محدودیت در MVC برداشته شده و مزیت آن امکان ارسال اطلاعات قسمت‌های مختلف یک فرم به کنترلرهای مختلف و جداسازی بهتر کدهای قسمت‌های مختلف برنامه است.

11- امکان Refactoring بهتر کدهای تکراری در ASP.NET MVC به کمک مفهوم فیلترها

فیلترها در MVC یک سری attribute هستند که می‌توان آن‌ها را به متدهای کنترلرها اعمال کرد و به صورت خودکار توسط فریم ورک پیش یا پس از اجرای یک متد اجرا می‌شوند. به این ترتیب حجم قابل ملاحظه‌ای از if و elseها را می‌توان در این فیلترها کپسوله کرد و کدهای متدهای کنترلرها را تمیزتر و زیباتر نمود.

12- سازگاری کامل با jQuery و jQuery Ajax و کلا انواع و اقسام فریم‌ورک‌های جاوا اسکریپتی

در MVC وب کنترل‌ها کنار گذاشته شده‌اند و سعی شده است با وب به همان نحو که هست برخورد شود. به این ترتیب اگر نیاز داشتید، کل دکمه‌های فرم را با spanها جایگزین کرده و توسط فریم ورک‌های CSS ایی تزئین کنید، بدون نیاز به نگارش جدیدی از ASP.NET MVC، باز هم برنامه کار خواهد کرد.
یا برای کار با اجزای مختلف فرم از ده‌ها و صدها افزونه موجود برای jQuery به سادگی می‌توان استفاده کرد. برای نمونه کل سیستم اعتبار سنجی توکار MVC با اعتبار سنجی jQuery یکپارچه و جایگزین شده است.
یا برای کار با jQuery Ajax نیازی نیست تا متدی را static تعریف کنید و به این ترتیب از مزایای امنیتی توکار ASP.NET محروم شوید (مثلا دسترسی به شیء User اعتبار سنجی مبتنی بر فرم‌ها). یا اگر فرم شما 10 فیلد دارد، کل این فیلدها به صورت خودکار به خواص متناظر با آن‌ها نگاشت خواهد شد و نیازی نیست برای این مورد کد بنویسید. به علاوه باید درنظر داشت که jQuery Ajax نسبت به فریم ورک Ajax همراه با ASP.NET web forms بسیار سبک‌تر و سریعتر عمل می‌کند چون نیازی ندارد تا هر بار View state را نیز به سرور ارسال کند.
همچنین در اینجا دیگر ID کنترل‌های مورد استفاده در اسکریپت‌ها به صورت خودکار تولید نمی‌شوند و برنامه نویس از ابتدای امر کنترل کاملی را روی این مساله دارد.

13- امکانات فشرده سازی css و js بهتر

در MVC 4 سیستم bundling آن از نمونه مشابه موجود در وب فرم‌ها کامل‌تر است و جهت فشرده سازی و یکی کردن هر دو مورد فایل‌های css و js می‌تواند بکارگرفته شود؛ به همراه تنظیمات کش مرورگر و gzip خودکار حاصل. به علاوه این سیستم را سفارشی سازی نیز می‌توان ساخت و بهینه سازی عملکرد آن مطابق نیاز میسر است.

14- یکپارچگی بهتر EF Code first با MVC

عنوان شد که وجود مدل‌ها و فرم‌های strongly typed یکی از مزایای کار با MVC است و ORMها نیز نهایتا با همین کلاس‌ها هستند که کار می‌کنند. در MVC سیستم کد سازی به نام scaffolding وجود دارد (تهیه شده توسط خود مایکروسافت) که می‌تواند بر اساس مدل‌های EF code first شما، قسمت عمده‌ای از کدهای یک برنامه ASP.NET MVC را تولید کند. سپس می‌توانید به سفارشی سازی آن مشغول شوید.

15- تزریق وابستگی‌ها در MVC ساده‌تر است

در هر دو فریم ورک وب فرم‌ها و MVC امکان تزریق وابستگی‌ها وجود دارد. اما در MVC می‌توان در میانه کار وهله سازی کنترلرها، دخالت کرد و کنترل آن را کاملا در دست گرفت. همین امر سبب می‌شود حین کار با کتابخانه‌های تزریق وابستگی‌ها در ASP.NET MVC حجم کد نویسی به شدت کاهش پیدا کند.

16- امکانات امنیتی MVC بیشتر است

عنوان شد که در MVC می‌توان اعتبار سنجی را تنها در حد یک خاصیت غیرفعال کرد. فیلتر مبارزه با حملات CSRF جزئی از فریم ورک MVC است. به همراه فیلتر Authorize آن که باز هم اعمال سفارشی سیستم اعتبار سنجی مبتنی بر فرم‌ها را ساده‌تر می‌کند با امکان یکپارچگی بهتر با Role providerهای سفارشی.

و یا برای نمونه Razor به صورت پیش فرض امن طراحی شده است. خروجی Razor همواره و در بدو امر، html encoded است مگر اینکه برنامه نویس آگاهانه آن‌را تغییر دهد. این مورد مقاومت در برابر حملات XSS را بالا خواهد برد.

امکان استفاده از فیلترهای سفارشی که عنوان شد، جهت مسایل امنیتی بسیار کاربرد دارند. برای مثال بررسی referrer فرم ارسال به سرور را درنظر بگیرید. در وب فرم‌ها می‌توان این‌کار را با یک http module که روی کل برنامه تاثیر گذار است انجام داد. اما در MVC این فیلتر را تنها می‌توان بر روی یک فرم خاص عمومی برای مثال اعمال کرد و نه کل برنامه.

نظرات خوانندگان

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۶:۳۵

سلام. تشکر از مقالات خوبتون. میخواستم بپرسم برای من که بیش از 2 سالی هست با سی شارپ و Win App کار کردم و میخوام برنامه نویسی تحت وب رو از صفر کار کنم کدوم یک بهتره؟ اول از ASP.NET شروع کنم و بعد برم سراغ ASP.NET MVC یا نه مستقیما از ASP.NET MVC شروع کنم. البته به صورت مقدماتی با ASP.NET آشنا هستم. ممنون میشم راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۴ ۱۶:۳۸

مستقیما می‌تونید با MVC شروع کنید. البته پیشنیازهای HTML، CSS و جاوا اسکریپت را هم باید لحاظ کنید.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۸/۰۵ ۲۳:۴۹

واقعا به موضوعات خوب و قابل دفاعی در مورد ام وی سی اشاره کردید.
ممنونم

نویسنده: M.Q
تاریخ: ۱۳۹۱/۰۸/۰۹ ۱۳:۲۷

با سلام و تشکر

جناب نصیری شما قبلا هم 16 مورد از کاربردها و مزایای sliverlight را در چه زمانی بهتر است از silverlight استفاده شود؟ ارائه داده اید. من چندین بار این سوال را از برنامه نویسان پرسیده ام اما هنوز جوابقانع کننده ای دریافت نکرده ام.

چرا با وجود کاربردها و مزایای silverlight (مخصوصا موارد 6-11-13-16 در لینک فوق) هنوز mvc asp.net مطرح تر، بازار کار بیشتر و حتی طرفداران بیشتری در بین برنامه نویسان دارد؟ (صرف نظر از افزونه ای که برای مرورگرها دارد)

لطفا با توضیحات مفیدتون این ابهام را برای برنامه نویسانی که با من هم عقیده هستند رفع نمائید.

با تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۹ ۱۳:۴۳

- البته در آن مطلب هم عرض شد که کاربرد عمده برنامه‌های تجاری سیلورلایت، برنامه‌های قابل اجرا در شبکه‌های داخلی است (اینترانت) و نه وب سایت‌های عمومی. خصوصا از لحاظ بحث SEO که سیلورلایت اساسا برای این منظور طراحی نشده.
- باز هم در وب سایت‌های عمومی کمتر از سیلورلایت استفاده می‌شود چون مانند فلش، کل فایل‌های باینری آن قابل دریافت و آنالیز است. همین مورد برای برنامه‌های به شدت امن قابل قبول نیست. اما در یک شبکه داخلی شاید این مساله اهمیتی نداشته باشد.
- تیم مونو اخیرا پشتیبانی از سیلورلایت رو متوقف کرده. بنابراین سازگاری با پلتفرم‌های غیر ویندوزی آن را در نگارش‌های جدید سیلورلایت فراموش کنید.

- خود مایکروسافت چند وقت قبل اعلام کرد که در وب تغییر جهت دادیم به HTML5 . هرچند بعدش تکذبیه صادر شد اما الان تمام کسانی رو که قبلا فقط در مورد سیلورلایت مطلب می‌نوشتند (اعضای سابق این تیم)، صرفا در مورد فریم ورک‌های جدید js و برنامه‌های تک صفحه‌ای وب مطلب می‌نویسند (با رویکرد MVC4).

و ... کسانیکه سیلورلایت را فراگرفتند چیزی رو از دست ندادند، چون معماری برنامه‌های WinRT و ویندوز 8، بسیار بسیار شبیه است به سیلورلایت. تا این حد که دوره‌های XAML و سی شارپ ویندوز 8 مدام به مقایسه سیلورلایت و WPF با این معماری جدید می‌پردازند.

نویسنده: M.Q
تاریخ: ۱۳۹۱/۰۸/۰۹ ۱۴:۲۹

ممنون از شما جناب نصیری

به امید روزی که برنامه نویسی وب تنها با دانش یک تکنولوژی برای برنامه نویسان میسر شود.

نویسنده: sysman
تاریخ: ۱۳۹۱/۱۲/۰۳ ۱۹:۱۰

نوشتن یک برنامه enterprise با استفاده از سیلورلایت سریعتر و راحت‌تر است یا mvc؟ با توجه به عدم ارایه سیلورلایت ۶ آیا نوشتن چنین برنامه هایی که حداقل دارای طول عمری 10 ساله هستند کار عاقلانه ای است؟ یا به طور کلی بهتره بپرسم آیا در حال حاضر گزینه دیگری به جز سیلورلایت برای نوشتن یک برنامه Lob وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۳ ۲۰:۳۳

- باتوجه به جمله بندی که بکار بردید، به نظر تصمیم خودتون رو در مورد سیلورلایت گرفتید. اما اگر برای 10 سال آینده می‌خواهید سرمایه گذاری کنید، سیلورلایت چیزی شبیه به VB6 خواهد شد. یعنی بیشتر فقط پشتیبانی و هات فیکس خواهد داشت. خلاصه سرمایه گذاری روی فناوری که عده‌ای درحال درخواست از مایکروسافت هستند که لطفا آن‌را ادامه دهید و وضعیتش معلوم نیست، کار صحیحی برای 10 سال بعد نخواهد بود.
سیلورلایت و فلش با تبلیغ منفی استیوجابز به علت مشکلات فلش در پلتفرم آن‌ها از بین رفتند.
- سریعتر و راحت‌تر برای برنامه‌ای که قرار است 10 سال کار کند معنا ندارد. چون یک چنین سیستمی حداقل 2 سال طول خواهد کشید تا نگارش یک آن جا بیفتد و بعد کار نگهداری خواهید داشت. در قسمت نگهداری، کار با سیستمی ساده‌تر و لذت بخش‌تر خواهد بود که اجزای آن به خوبی از هم جدا شده باشند و اینجا است که MVC از روز اول بر روی این مساله تاکید دارد، آن هم به صورت یک فریم ورک توکار رسمی. هیچ وقت مایکروسافت MVVM رو با سیلورلایت و WPF به صورت یکپارچه و چیزی شبیه به MVC ارائه نداد. هرچند ده‌ها فریم ورک MVVM وجود دارند ولی در کل از نظر من بیشتر یک سری Helper هستند تا فریم ورکی شبیه به MVC.
- یک مقدار کار با jQuery و فریم ورک‌های جدید جاوا اسکریپتی را بدانید، اکثر قابلیت‌های سیلورلایت رو می‌تونید داشته باشید. ضمن اینکه الان مایکروسافت روی فریم ورک‌های SPA برای MVC به شدت مشغول تبلیغ است. برنامه‌های تک صفحه‌ای وب = Single page applications = SPA. این مورد جایگزین بهتری است برای سیلورلایت.

نویسنده: sysman
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۲:۱۱

ممنون از راهنمایی شما. هنوز تصمیم قطعی گرفته نشده. در حقیقت چند تستی که در خصوص سیلورلایت انجام دادم کمی من رو ناامید کرد. چون بستر ارتباطی اینترنت و اینترانت هست پس حجم داده‌های ارسال شده به کاربر باید حداقل مقدار ممکن باشد

که در مورد سیلورلایت با یک صفحه ساده و 4 عدد کنترل telerik حجم فایل xap به 2 مگ رسد؟! این یعنی فاجعه.
بین سیلورلایت و MVC تردید داشتیم تا حدی جواب سوالم رو گرفتم.
منظورم از سریعتر و راحت‌تر بیشتر تاکید بر روی امکان استفاده مجدد از یک کار تکراری در قسمتهای مختلف برنامه هست. مثلا نمایش لیست مقادیر بر اساس خصوصیات یک یا چند entity با شرطی خاص که ممکن هست بارها بارها در صفحات تکرار بشه.
نگرانی من بیشتر در خصوص حذف کارهای تکراری برای برنامه نویس‌های معمولی هست.
و هنوز هم شک دارم IIS بتونه 10000 کاربر همزمان رو جوابگو باشه. آیا شما تجربه عملی با این حجم کاربر دارید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۲:۱۸

برای مدیریت 10 هزار کاربر، یک سری روش‌هایی در ویندوزهای سرور وجود دارند مانند clustering ، load balancing و امثال آن (حتی این مباحث برای SQL Server هم به صورت اختصاصی وجود دارد؛ تحت سرفصلی به نام High availability). برای این موارد بهتر است با یک ادمین شبکه ویندوزی مشورت کنید.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۸/۲۳ ۰:۴۸

ممنون از شما بابت اطلاع رسانی خوبتون مثل همیشه واقعا asp.net با mvc تغییرشگرفی کرده
من همون اول که متوجه شدم از یکی بودن form اون هم runat=server رهایی پیدا کردیم خیلی خوشم اومد
این مسئله همیشه موجب آزار بود و هر دفعه باید یه جوری باهاش سر میکردیم و البته viewstate رو هم نباید فراموش کرد