عنوان: **یک تکنیک جالب در نحوه نام گذاری فیلدهای دیتابیس به منظور استفاده بهینه از فایل های T4 در MVC5**
نویسنده: صادق نجاتی
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱۰:۴۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Kendo UI, T4

بدون شک دوستانی که با تکنولوژی محبوب ASP.NET MVC5 کار کرده اند این نکته را می‌دانند که اگر فایل‌های T4 که وظیفه Scaffolding را به عهده دارند به پروژه خود اضافه کنند می‌توانند نحوه تولید خودکار Controllerها و Viewهای متناظر را سفارشی کنند. مثلا می‌توان این فایل‌ها را طوری طراحی کرد که Controller و Viewهای تولیدی به طور اتوماتیک چند زبانه و یا Responsive تولید شوند (این موضوعات بحث اصلی مقاله نیستند) و اما بحث اصلی را با یک مثال آغاز می‌کنیم :
فرض کنید در دیتابیس خود یک Table دارید که قرار است اطلاعات یک Slider را در خود نگه دارد. این Table دارای یک فیلد از نوع nvarchar برای ذخیره آدرس تصویر ارسالی توسط کاربر است.
در حالت عادی اگر از روی مدل این Table اقدام به تولید خودکار Controller و View متناظر کنید، یک editor (تکست باکس) برای دریافت آدرس تصویر تولید خواهد شد که برنامه نویس یا طراح باید به طور دستی آن را (به طور مثال) با Kendo uploader جایگزین نماید. ما می‌خواهیم برای فیلدهایی که قرار است آدرس تصویر را در خود نگه دارد به طور اتوماتیک از Kendo uploader استفاده شود. راه حل چیست؟
بسیار ساده است. ابتدا باید در نظر داشت که هنگام طراحی Table در دیتا بیس فیلد مورد نظر را به این شکل نامگذاری کنید : ExampleIMGURL (نحوه نام گذاری دلخواه است) مقصود آن است که نام هر فیلدی که قرار است آدرس یک تصویر را در خود نگه دارد باید حاوی کلمه (IMGURL) باشد. مجددا ذکر می‌شود که نحوه نامگذاری اختیاری است. سپس فایل Create.t4 را باز کنید و کد :

```
@Html.EditorFor(model => model.<#= property.PropertyName #>)
```

را با کد زیر جایگزین کنید :

```
<#
if (GetAssociationName(property).Contains ("IMGURL")
{
#>
     @Html.Kendo().Upload().Name("<#= property.PropertyName #>")
<#
}
else
{
#>
     @Html.EditorFor(model => model.<#= property.PropertyName #>)
<#
}
#>
```

کد بالا چک می‌کند اگر نام فیلد مد نظر حاوی " IMGURL " باشد یک کندو آپلودر تولید کرده در غیر این صورت یک ادیتور ساده تولید می‌کند. البته این فقط یک مثال است و بدون شک دامنه استفاده از این تکنیک وسیع‌تر است.

اگر این مطلب مفید واقع شد با در نظر گرفتن نظرات ارسالی به تکنیک‌های آتی اشاره خواهد شد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱۱:۳۰

قابلیت سفارشی سازی EditorFor در ASP.NET MVC پیش بینی شده‌است و با استفاده از UIHint قابل انتساب به خواص مدل مورد نظر است. البته این مورد برای حالت Code first یا حالتیکه از ViewModels استفاده کنید بیشتر کاربرد دارد.
یک مثال:
فایلی را به نام Upload.cshtml ، در مسیر Views/Shared/EditorTemplates با محتوای ذیل ایجاد کنید:

```
@model string
@Html.Kendo().Upload().Name("@ViewData.ModelMetadata.PropertyName")
```

سپس برای استفاده از آن فقط کافی است خاصیت مدنظر را با ویژگی UIHint مزین کنید:

```
[UIHint("Upload")]
public string ImageUrl {set;get;}
```

نویسنده: صادق نجاتی
تاریخ: ۱۳۹۳/۰۲/۲۹ ۱۱:۵

ضمن تشکر از آقای نصیری؛
بدون شک نقش UIHint در سفارشی سازی انکار ناپذیر است. ولی همانطور که گفته شد دامنه استفاده از این تکنیک وسیع‌تر است. مثلا حالتی را در نظر بگیرید که می‌خواهیم از طریق Scaffolding برای یک جدول بانک اطلاعاتی که یک فیلد آن آدرس یک تصویر را نگهداری می‌کند View ایجاد نماییم. خوب ما در صفحه Index می‌خواهیم تصویر مورد نظر با اندازه 100 * 100 پیکسل نمایش دهیم (چون قرار است لیستی از تصاویر نمایش داده شود باید در اندازه قابل نمایشی باشد) ولی در صفحه Details باید اندازه بزرگتری از تصویر را به نمایش بگذاریم. حال اگر از UIHint استفاده کنیم تنها یکی از موارد قبل (سفارشی سازی در لیست و جزئیات) محقق خواهد شد. اگر بخواهیم انجام این کارها را به صورت اتوماتیک به Scaffolding بسپاریم باید مطابق آنچه گفته شد ، فایل‌های T4 را (List.t4 و Details.t4) سفارشی سازی نماییم.