闲来继续学习蒋金楠大师的ASP.NET MVC框架揭秘一书，当前主要阅读的内容是Model元数据的解析，即使是阅读完的现在，仍然有不少细节不是特别明白。好在这部分内容主要是关于Razor引擎的呈现的，通过注解的方式对Model进行自定的修饰，最终使得页面在渲染时（即从cshtml文件转化为html时），相关的数据能够按照指定的形式转化并显示。由于接下来的项目中不再打算使用Razor引擎，该引擎虽然很不错，但也有一些问题，例如存在HTML5代码与HtmlHelper的混写，使得UI层很难与业务代码层完全的分离。所以不太利于程序开发的分工，在当前的互联网需求迅速变化的场景下变得不是也别适合。而且相关工作人员的招聘等问题上也比较难得处理，不可能要求每一个工程师都具有全栈能力，企业也可能负担的其相关成本，自己对微软2012年推出的blend开发模式不太了解，不知道是不是只是支持WPF，WinForm等，可能有些理解上问题，请给予指正。个人观点上，更倾向于直接使用HTML5开发，框架主要负责路由，过滤器等功能，当前携程等网站也主要采用这种方式。因此，该章节算是选学了，主要介绍一些主要的概念。

首先要提及的是元数据，一说元数据，大家第一反应可能都是一样的，就是中间语言IL中对类的描述信息。一般我们可以通过自定义特性的方式对其进行扩充，这儿的Model元数据只要用于控制Model对象（ViewModel对象），在View上的呈现形式。其使用System.Web.Mvc.ModelMetadata来表示Model元数据，并且ModelMetadata是一种迭代，支持自包含的结构，有点像组合模式。如果ModelMetadata是一个复杂类型，可以通过一个相应的XXXTypeConverter辅助类将其转化为简单类型。

接下来，简单介绍一下与该框架相关数据注解特性，框架就是依靠这些注解特性和相应的模板方法来控制Model数据的显示，如下表所示。

|  |  |
| --- | --- |
| 特性名称 | |
| **UIHintAttribute** |  |
| HiddenInputAttribute | ScaffoldColumnAttribute |
| DisplayTypeAttribute | DisplayFormatAttribute |
| EditableAttribute | ReadOnlyAttribute |
| DisplayAttribute | DisplayNameAttribute |
| RequiredAttribute |  |
| AllowHtmlAttribute |  |
| RadioButtonListAttribute | CheckBoxListAttribute |
| DropdownListAttribute | ListBoxAttribute |

相关特性大部分都是顾名思义，比较容易理解，也与EF中的注解特性相似，就不一一介绍了。需要注意的是，如果想自定义一个特性，那么就要实现IMetadataAware接口，与它同名的接口也经常出现在.NET相关数据绑定中，代码如下所示。

|  |
| --- |
| [AttributeUsage(AttributeTargets.Class | AttributeTargets.Property)]  public class DisplayTextAttribute : Attribute, IMetadataAware  {  private static Type staticResourceType;  public string DisplayName { get; set; }  public Type ResourceType { get; set; }  public DisplayTextAttribute()  {  this.ResourceType = staticResourceType;  }  public void OnMetadataCreated(ModelMetadata metadata)  {  this.DisplayName = this.DisplayName ?? (metadata.PropertyName ?? metadata.ModelType.Name);  if (null == this.ResourceType)  {  metadata.DisplayName = this.DisplayName;  return;  }  PropertyInfo property = this.ResourceType.GetProperty(this.DisplayName, BindingFlags.NonPublic | BindingFlags.Public | BindingFlags.Static);  metadata.DisplayName = property.GetValue(null, null).ToString();  }  public static void SetResourceType(Type resourceType)  {  staticResourceType = resourceType;  }  } |

Tip：注意在项目中定义相关资源文件，将英文的属性名转化为中文。

常见模板方法为：

|  |  |  |
| --- | --- | --- |
| 模板方法 | | |
| HtmlHelper<TModel> | Display | DisplayFor |
|  | Editor | EditorFor |
|  | DisplayForModel | EditForModel |
|  | Label | LabelFor |
|  | DisplayText | DisplayTextFor |
|  | DropdownList | DropdownListFor |
|  | ListBox | ListBoxFor |

以上可以很清楚的看到显示模式和编辑模式两种不同的显示形式，由于框架是根据元数据对象中的数据类型属性值去寻找对应模板的，因此将需要将模板的View定义放在EditorTemplates目录下。接下来通过一个表格介绍一些框架预定义的模板。

|  |  |  |
| --- | --- | --- |
| 预定义方法 | | |
| EmailAddress | HiddenInput | Html |
| Text&String | Url | MultilineText |
| Password | Decimal | Boolean |
| Collection | Object |  |

最后用一个简图介绍下与ModelMetadata相关的类加强理解与记忆。



可以看到，ModelMetadata是会进行缓存的，并且通过原型模式进行创建，在.NET中经常可以看到Provider，这儿主要起一个提供者的作用，但与工厂类等其他创建型模式有什么区别仍然不是很清楚，还需要加强理解。

注：本文主要供自己学习，不妥之处望见谅。

参考资料：

[1]蒋金楠. ASP.NET MVC4框架揭秘[M]. 上海:电子工业出版社, 2012. 123-182