

در ادامه می‌خواهیم نحوه‌ی ایجاد یک فرم‌ساز ساده را ASP.NET MVC بررسی کنیم.

مدل‌های برنامه ما به صورت زیر می‌باشند:

```
namespace SimpleFormGenerator.DomainClasses
{
    public class Form
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public virtual ICollection<Field> Fields { get; set; }
    }
    public class Field
    {
        public int Id { get; set; }
        public string TitleEn { get; set; }
        public string TitleFa { get; set; }
        public FieldType FieldType { get; set; }
        public virtual Form Form { get; set; }
        public int FormId { get; set; }
    }
    public enum FieldType
    {
        Button,
        Checkbox,
        File,
        Hidden,
        Image,
        Password,
        Radio,
        Reset,
        Submit,
        Text
    }
}
```

توضیح مدل‌های فوق:

همانطور که مشاهده می‌کنید برنامه ما از سه مدل تشکیل شده است. اولین مورد آن کلاس فرم است. این کلاس در واقع بیانگر یک فرم است که در ساده‌ترین حالت خود از یک Id، یک عنوان و تعدادی از فیلدها تشکیل می‌شود. کلاس فیلد نیز بیانگر یک فیلد است که شامل: آی‌دی، عنوان انگلیسی فیلد، عنوان فارسی فیلد، نوع فیلد (که در اینجا از نوع enum انتخاب شده است که خود شامل چندین آیتم مانند Text، Radio و... است) و کلید خارجی کلاس فرم می‌باشد. تا اینجا مشخص شد که رابطه فرم با فیلد، یک رابطه یک به چند است؛ یعنی یک فرم می‌تواند چندین فیلد داشته باشد. کلاس کانتکست برنامه نیز به این صورت می‌باشد:

```
namespace SimpleFormGenerator.DataLayer.Context
{
    public class SimpleFormGeneratorContext : DbContext, IUnitOfWork
    {
        public SimpleFormGeneratorContext()
            : base("SimpleFormGenerator") {}
        public DbSet<Form> Forms { get; set; }
        public DbSet<Field> Fields { get; set; }
        public DbSet<Value> Values { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Value>()
                .HasRequired(d => d.Form)
                .WithMany()
        }
    }
}
```

```
.HasForeignKey(d => d.FormId)
.WillCascadeOnDelete(false);

    }
}
}
```

همانطور که مشاهده می‌کنید مدل‌های برنامه را در معرض دید EF قرار داده‌ایم. تنها نکته‌ای که در کلاس فوق مهم است متد OnModelCreating است. از آنجائیکه رابطه کلاس Field و Value یک رابطه یک‌به‌یک است باید ابتدا و انتهای روابط را برای این دو کلاس تعیین کنیم.

تا اینجا می‌توانیم به کاربر امکان ایجاد یک فرم و همچنین تعیین فیلدهای یک فرم را بدهیم. برای اینکار ویوهای زیر را در نظر بگیرید:
ویو ایجاد یک فرم:

```
@model SimpleFormGenerator.DomainClasses.Form
@{
    ViewBag.Title = "صفحه ایجاد یک فرم";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div>
            <span>عنوان</span>
            <div>
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-danger" })
            </div>
        </div>

        <div>
            <div>
                <input type="submit" value="ذخیره" />
            </div>
        </div>
    </div>

    <div>
        @Html.ActionLink("بازگشت", "Index")
    </div>
}
```

ویو ایجاد فیلد برای هر فرم:

```
@model SimpleFormGenerator.DomainClasses.Field
@{
    ViewBag.Title = "CreateField";
}

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div>
            <span>عنوان انگلیسی</span>
            <div>
                @Html.EditorFor(model => model.TitleEn, new { htmlAttributes = new { @class = "form-control" } })
            </div>
        </div>
    </div>
}
```


ویوی نمایش فرم تولید شده برای کاربر نهایی:

```
@using SimpleFormGenerator.DomainClasses
@model IEnumerable<SimpleFormGenerator.DomainClasses.Field>

@{
    ViewBag.Title = "نمایش فرم";
}

<div>
    <div>
        <div>
            @using (Html.BeginForm())
            {
                @Html.AntiForgeryToken()
                for (int i = 0; i < Model.Count(); i++)
                {
                    if (Model.ElementAt(i).FieldType == FieldType.Text)
                    {
                        <text>
                            <input type="hidden" name="[@i].FieldType"
value="@Model.ElementAt(i).FieldType" />
                            <input type="hidden" name="[@i].Id" value="@Model.ElementAt(i).Id" />
                            <input type="hidden" name="[@i].FormId" value="@Model.ElementAt(i).FormId"
/>
                            <div>
                                <label>@Model.ElementAt(i).TitleFa</label>
                                <div>
                                    <input type="text" name="[@i].TitleEn" />
                                </div>
                            </div>
                        </text>
                    }
                }
                <div data-formId="@ViewBag.FormId">
                    <div>
                        <input type="submit" value="ارسال فرم" />
                    </div>
                </div>
            }
        </div>
        <div>
            @Html.ActionLink("بازگشت", "Index")
        </div>
    </div>
</div>
```

همانطور که در کدهای فوق مشخص است از اکشن متدی که در ادامه مشاهده خواهید کرد لیستی از فیلدهای مربوط به یک فرم را برای کاربر به صورت رندر شده نمایش داده‌ایم. در اینجا باید براساس فیلد FieldType، نوع فیلد را تشخیص دهیم و المنت متناسب با آن را برای کاربر نهایی رندر کنیم. برای اینکار توسط یک حلقه for در بین تمام فیلدها پیمایش می‌کنیم:

```
for (int i = 0; i < Model.Count(); i++)
{
    // code
}
```

سپس در داخل حلقه یک شرط را برای بررسی نوع فیلد قرار داده‌ایم:

```
if (Model.ElementAt(i).FieldType == FieldType.Text)
{
    // code
}
```

بعد از بررسی نوع فیلد، خروجی رندر شده به این صورت برای کاربر نهایی به صورت یک عنصر HTML نمایش داده می‌شود:

```
<input type="text" name="[@i].TitleEn" />
```

همانطور که در کدهای قبلی مشاهده می‌کنید یکسری فیلد را به صورت مخفی بر روی فرم قرار داده‌ایم زیرا در زمان پست این اطلاعات به سرور از آنجائیکه مقادیر فیلدهای فرم تولید شده ممکن است چندین مورد باشند، به صورت آرایه‌ای از عناصر آنها را نمایش خواهیم داد:

```
[@i].FieldType
```

خوب، تا اینجا توانستیم یک فرم‌ساز ساده ایجاد کنیم. اما برای ارسال این اطلاعات به سرور به یک مدل دیگر احتیاج داریم. این جدول در واقع محل ذخیره‌سازی مقادیر فیلدهای یک فرم و یا فرم‌های مختلف است.

```
public class Value
{
    public int Id { get; set; }
    public string Val { get; set; }
    public virtual Field Field { get; set; }
    [ForeignKey("Field")]
    public int FieldId { get; set; }
    public virtual Form Form { get; set; }
    [ForeignKey("Form")]
    public int FormId { get; set; }
}
```

این جدول در واقع شامل: آی‌دی، مقدار فیلد، کلید خارجی فیلد و کلید خارجی فرم می‌باشد. بنابراین برای ارسال ویو قبلی به سرور اکشن‌متد ShowForm را در حالت Post به این صورت خواهیم نوشت:

```
[HttpPost]
public ActionResult ShowForm(IEnumerable<Field> values)
{
    if (ModelState.IsValid)
    {
        foreach (var value in values)
        {
            _valueService.AddValue(new Value { Val = value.TitleEn, FormId = value.FormId,
FieldId = value.Id});
            _uow.SaveAllChanges();
        }
    }
    return View(values);
}
```

سورس مثال جاری را نیز می‌توانید از [اینجا](#) دریافت کنید.

نظرات خوانندگان

نویسنده: مرتضی اسدی
تاریخ: ۹:۳۳ ۱۳۹۳/۰۹/۱۵

با تشکر از مطلب مفید شما
برای اعتبار سنجی فیلدها ایده خاصی دارید؟

نویسنده: سیروان عقیقی
تاریخ: ۲۲:۶ ۱۳۹۳/۰۹/۱۵

به دو روش می‌تونید اینکار رو انجام بدید: 1- از یک جدول دیگر برای اعمال اعتبارسنجی استفاده کنید که کاربر خودش بتونه rule اعمال کنه، رابطه این جدول با جدول فیلد هم به صورت یک به چند هست یعنی یک فیلد می‌تونه چند تا validation rule داشته باشه:

```
public class FieldValidation
{
    public int Id { get; set; }
    public string Rule { get; set; }
    public virtual Field Field { get; set; }
}
```

روش دوم:

می‌تونید از یک فیلد اضافی تحت عنوان "متن خطا" در جدول فیلد استفاده کنید و در ویوی مربوطه به این صورت از اون استفاده کنید:

```
<div class="col-md-4">
    <input type="text" name="@i.TitleEn" data-val="true" data-val-
required="عنوان را وارد نمایید" id="@i.TitleEn" value="" />
    <span class="field-validation-valid text-danger" data-valmsg-
for="@i.TitleEn" data-valmsg-replace="true"></span>
</div>
```