M V C

هادی حسینی پور

تکنیک های پیشرفته برنامه سازی

استاد عضدانلو

MVCچیست!؟

مفهوم کلیدی این فریم ورک همان سه حرف آخر آن یعنی MVC است. پس کمی در مورد آن توضیح می دهم. همانطور که گفتم، MVCیک الگوی طراحی است که همانطور که از نامش پیداست، یک پروژه نرم افزاری را به سه قسمت منطقی Model, View و Controllerتقسیم می کند. شاید شما در حال حاضر با معماری ۳ لایه نرم افزاری آشنا باشید. اگر اینطور است، شما مشکلی در درک الگوی طراحی MVC نخواهید داشت .MVC مفهوم جدیدی نیست، خیلی وقت است که در جاوا، رابی، PHPو بسیاری پلت فرم های دیگر از این الگو برای طراحی نرم افزار استفاده می شده است. اما خب برای توسعه دهندگان ASP.NET تازه است.

سه قسمت اصلی الگوی: MVC

Model :مدل قسمتی از یک اپلیکیشن است که وظایف سنگین دسترسی به داده ها، پیاده سازی منطق و موجودیت ها را بر عهده دارد. به طور معمول یک مدل وظیفه Mapکردن جداول اطلاعاتی یک دیتابیس را به کلاس های شیء گرا و برعکس را بر عهده می گیرد. احتمالاً شما همین الان هم در پروژه های خود، مدل را پیاده سازی می کنید و به آن لایه دسترسی به داده می گویید Model !باید طوری پیاده سازی شود که به هیچ وجه به رابط کاربری وابستگی نداشته باشد.

View :احتمالاً کاربرد View را حدس زده اید! رابط کاربری همان View است .در واقع بخشی که یک کاربر نهایی با آن تعامل خواهد داشت و اطلاعات را نمایش می دهد، Viewنام دارد. همانطور که رابط کاربری برای Model هیچ اهمیتی ندارد، اینکه چطور داده ها اعتبارسنجی یا ذخیره می شوند یا منطق اپلیکیشن شما چطور پیاده سازی شده است، برای View مهم نیست.

Controller : فضای خالی میان Model و View را Controller پر می کند. از آنجا که Model و View هیچ ارتباطی با هم ندارند و برای یکدیگر هیچ اهمیتی قائل نیستند، Controllerداده ها را از Model به Viewبرای نمایش به کاربر انتقال می دهد. کنترلرها تصمیم می گیرند که اطلاعاتی که شما وارد کرده اید را به کجا برسانند و همینطور چه چیزی را باید در خروجی مشاهده کنید. در واقع کنترل کننده و هماهنگ کننده میان Model و View است.

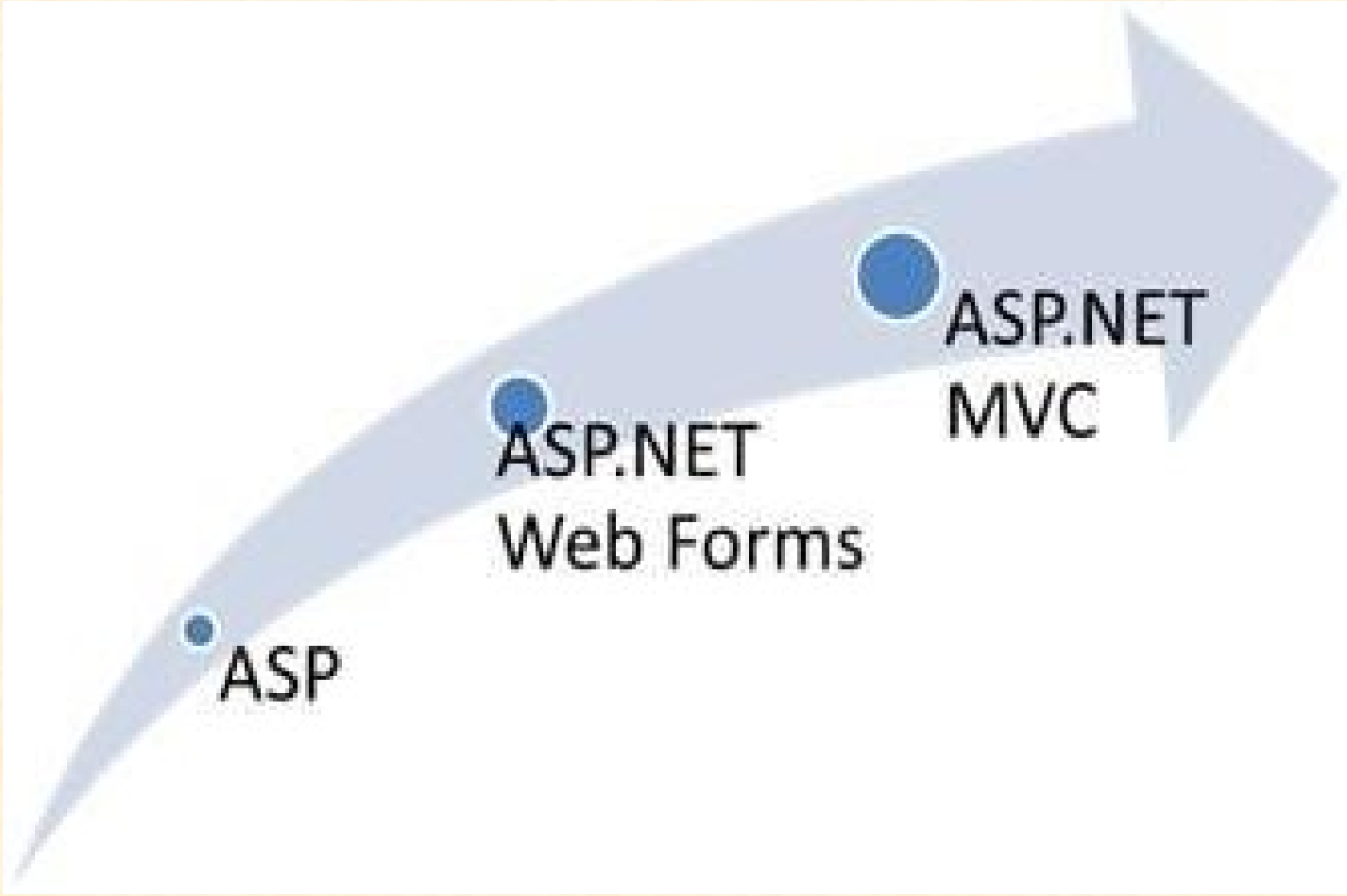
تاریخچه mvc :

معماری mvcدر دهه ۷۰ میلادی معرفی شد اما در آن زمان پیاده سازی برنامه های stand alone با استفاده از این معماری چندان مورد استقبال برنامه نویسان قرار نگرفت. اما با ظهور اینترنت و برنامه های مبتنی بر وب، این معماری شانس دوباره ای یافت asp.net mvc .فریم ورک مبتنی بر معماری mvc مایکروسافت میباشد که از دلایل محبوبیت این معماری نظام بخشیدن به پروژه های طراحی سایت میباشد. معمولاً به علت همکاری چندین تکنولوژی مختلف با هم در برنامه های مبتنی بر وب ساختار پروژه های بزرگ پیچیده میشوند و اعمال تغییرات و همچنین رفع خطا های پروژه مشکل و زمانبر میشوند که معماری mvc با جداسازی لایه های مختلف برنامه نویسی تا حد زیادی این مشکل را رفع کرده است. از دیگر مزایای این فریم ورک میتوان به کنترل کامل بر روی html نهایی، پشتیبانی از فریم ورک های گوناگون برای unit testing ، کنترل بر روی آدرس های url و تعامل راحتتر با فریم ورک های javascript اشاره کرد.

معماری mvc یا همان model view controller که اخیراً توسط مایکروسافت در asp.net به کار گرفته شده است در واقع انقلابی در زمینه بهبود پروژه های تحت وب بود. جهت آشنایی بیشتر با مزایای استفاده از این معماری در asp.net کافی است تکنولوژی قدیمی تر یعنی asp.net webformsرا با آن مقایسه کنیم:

-1در asp.net webforms طراحی بهینه و ساختارمند و همچنین رعایت مسائل امنیتی بسیار پیچیده تر گاهاً غیر ممکن بود. در حالی که با استفاده از معماری mvc بسیاری از این موارد به سهولت انجام می پذیرد.

-2در asp.net mvc انعطاف و قابلیت کنترل بسیار زیادی برای برنامه نویسان فراهم است چرا که در تکنولوژی webforms در بیشتر مواقع برنامه نویسان از مجموعه ای از ابزارهای فراهم شده استفاده میکردند اما در mvc هیچگونه ابزار و واسطی که ماهیتاً همراه با محدودیت هستند وجود ندارد و برنامه نویسان کاملاً به صورت دستی اقدام به طراحی و پیاده سازی میکنند.


ASP.NET
MVC
ASP.NET
Web Forms
ASP

-3در نهایت باتوجه به حذف asp.net webforms در نسخه ٦ .net استفاده از آن یک اشتباه بزرگ محسوب میشود .

Mvc-4 یک معماری است نه یک تکنولوژی، بنابراین از ریسک آزاد است و در آینده مانند asp.net webformsبه تاریخ نخواهد پیوست.

در حال حاضر وب وان از آخرین نسخه معماری mvc در asp.net c# و موتور نمایش razor که از سوی شرکت مایکروسافت که از قوی ترین ها در زمینه انواع علوم رایانه ای است، استفاده میکند. با توجه به اینکه visual basic asp.net توسط شرکت های بسیار اندکی استفاده میشود و همچنین در نسخه ٦ .net نیز حذف شده است در اینجا مورد بررسی قرار نمی گیرد.

لازم به ذکر است visual basic طبق بررسی های انجام گرفته در واحد فنی هیچگاه در وب وان مورد استفاده قرار نگرفته است.

مزایای استفاده از: mvc

-با تقسیم یک برنامه به سه قسمت مدل، نمایشگر و کنترلگر، مدیریت برنامه یا پروژه آسانتر میشود.

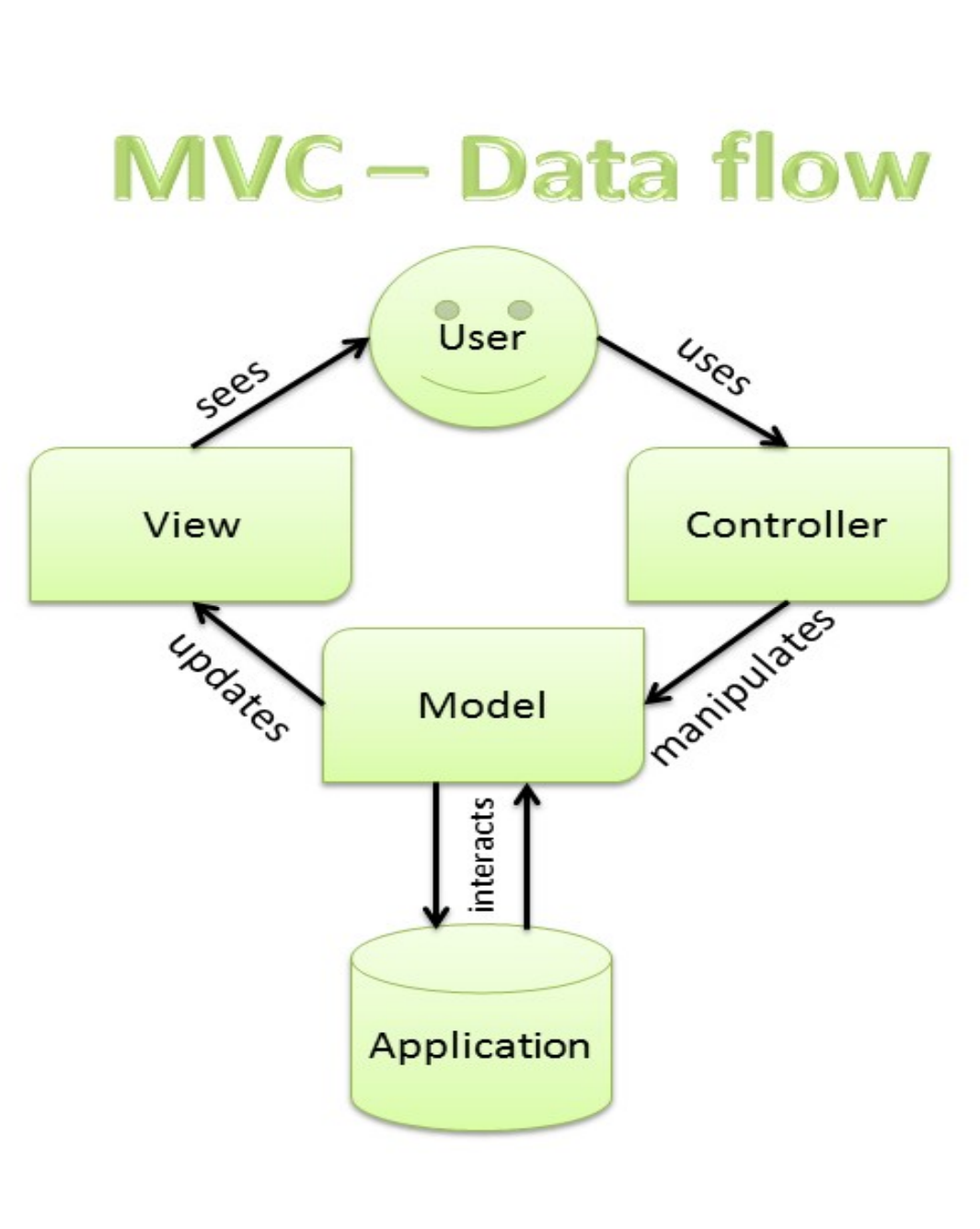
-از viewstate و فرم های سروری استفاده نمی کند . به همین خاطر برای برنامه نویسانی که تسلط کامل بر رفتار برنامه را می خواهند عالی است.

-از الگوی کنترلگر جلو استفاده میکند که درخواست های برنامه را توسط یک کنترلگر پردازش میکند. این مسئله باعث میشود تا بتوانیم برنامه هایی را طراحی کنیم که از زیر ساخت های غنی مسیریابی پشتیبانی میکند.

-پشتیبانی بهتری از طراحی و توسعه آزمون محور دارد.

-برای برنامه های پشتیبانی شده توسط تیم های بزرگ برنامه نویسان که کنترل بسیار بر رفتار برنامه را می خواهند، بهتر کار میکنند .

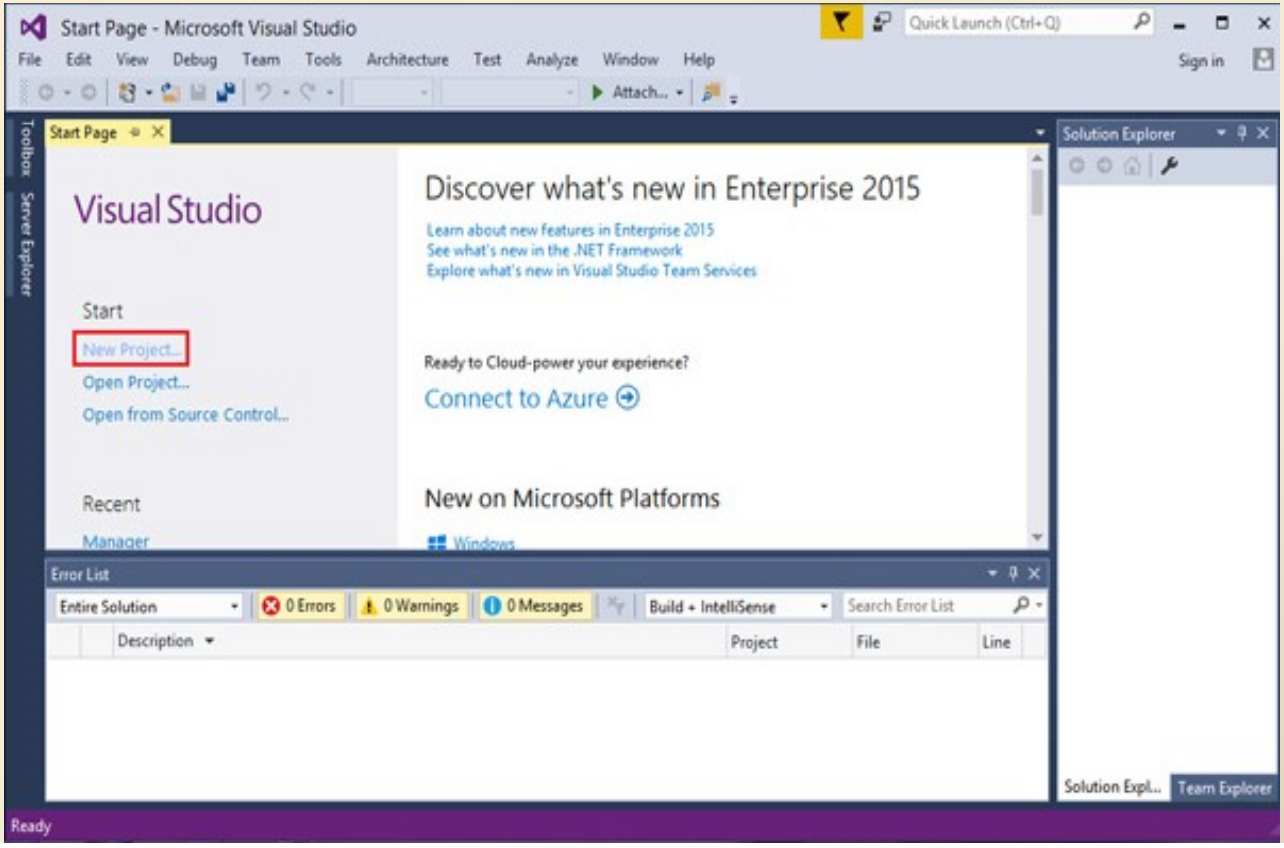
mvcبه زبان ساده تر :



ساخت پروژه mvc

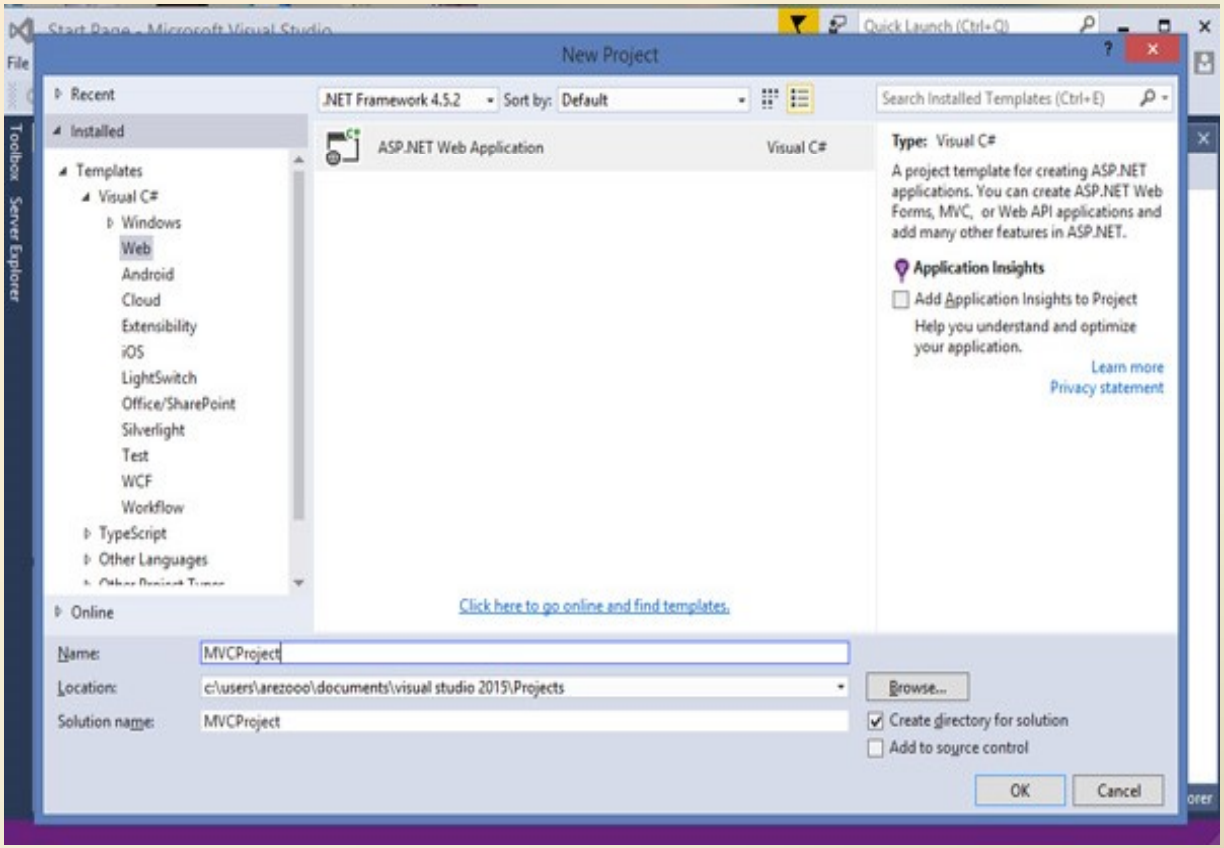
گام اول ساخت پروژه MVC

در گام نخست نرم افزار Visual Studioرا اجرا کنید (توصیه میکنم از ویژوال استودیو ۲۰۱۵ نسخه Enterprise استفاده نمایید) در پنجره Start Pageکه در تصویر زیر مشاهده می‌نمایید بر روی گزینه New Projectکلیک نمایید شما همچنین می‌توانید از طریق منوی بالا پنجره Start Pageو با انتخاب گزینه Fileو سپس Newو بعد انتخاب Projectهمین فرآیند را انجام دهید.



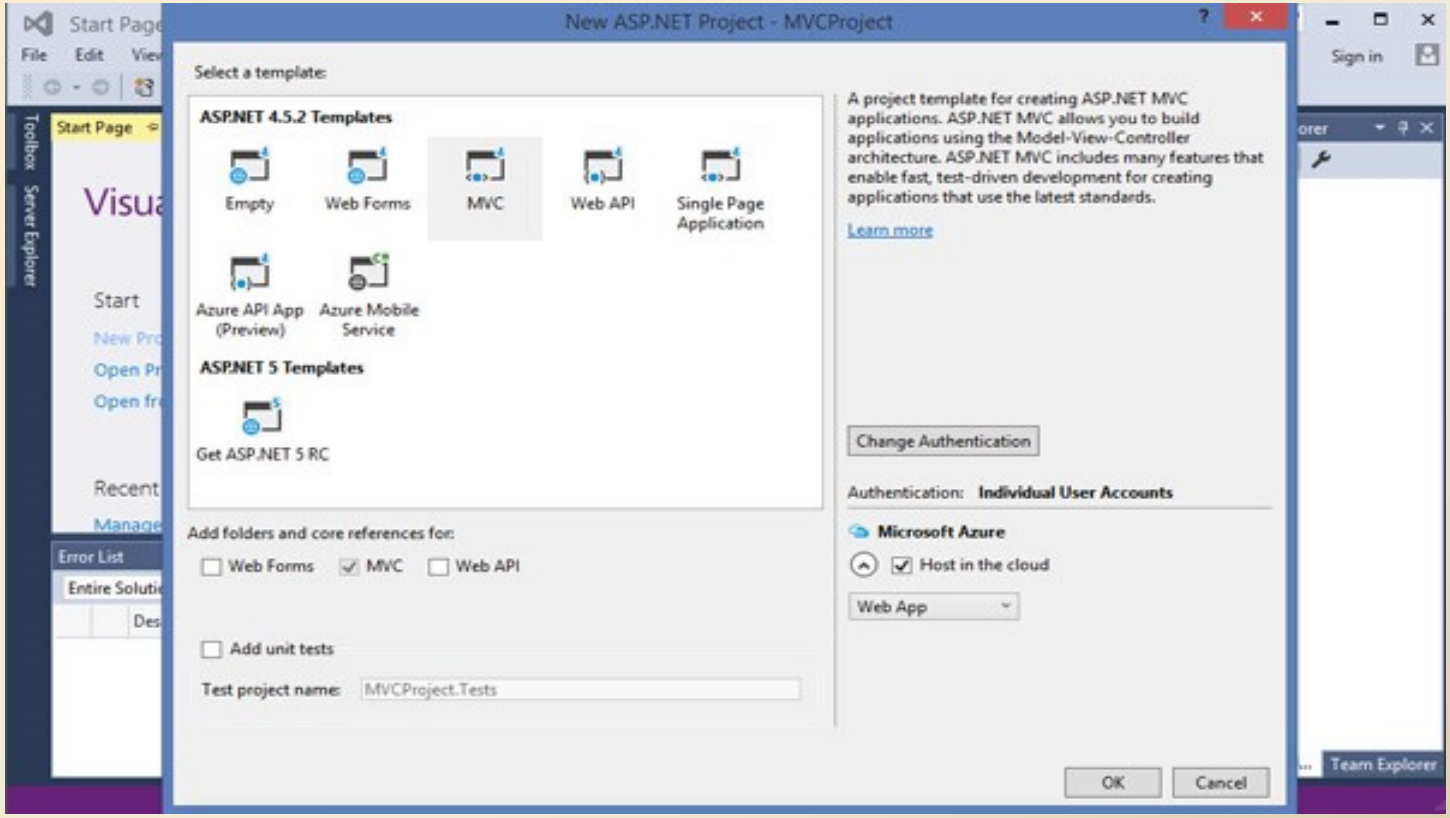
گام دوم ساخت پروژه MVC

در گام دوم پنجره New Projectبرای شما باز می‌شود در پنجره باز شده از منوی سمت چپ گزینه Webرا انتخاب کنید و سپس در قسمت Nameکه در پایین پنجره مشاهده می‌نمایید یک نام دلخواه برای پروژه خود انتخاب نمایید هم چنین شما می‌توانید ازقسمت Locationو با کلیک بر روی گزینه Browsمسیر پیش فرضی که پروژه شما در آنجا ذخیره می‌شود را تغییر دهید و در نهایت بر روی گزینه Okکلیک نمایید.



گام سوم ساخت پروژه MVC

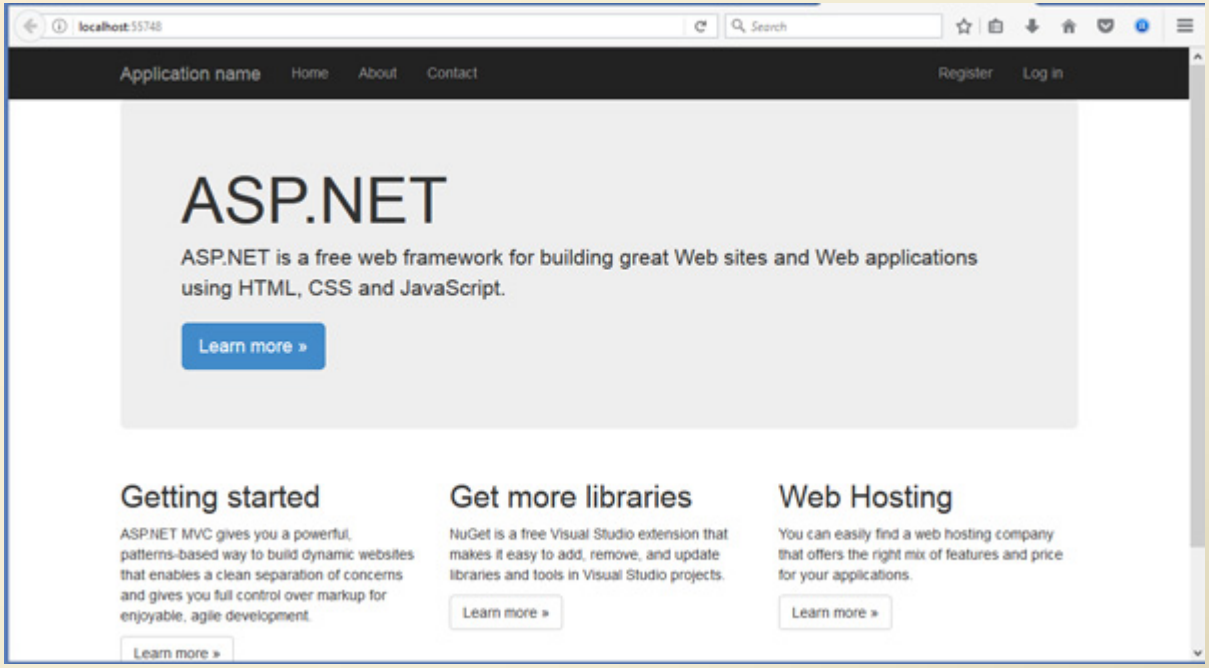
در این گام پنجره New ASP.NET Projectبرای شما باز می‌شود بر روی گزینه MVCکلیک نمایید تا یک پروژه شسته رفته با تمام قابلیت های احراز هویت نظیر (عضویت کاربر در سایت ، ورود کاربر به سایت با وارد نمودن ایمیل و رمز عبور ، تغییر رمز عبور توسط کاربر و ...) برای شما ایجاد شود. در صورتی که می‌خواهید یک پروژه خالی از نوع MVCایجاد کنید گزینه Emptyرا انتخاب نموده و از قسمت add folders and core references forتیک گزینه MVCرا بزنید. و در نهایت بر روی گزینه Okکلیک نمایید.



گام چهارم ساخت پروژه MVC

حال اندکی صبر کنید تا ویژوال استودیو پروژه شما را ایجاد نمایید. پس از این که پروژه برای شما ساخته شد با زدن دکمه F5می توانید پروژه خود را اجرا نمایید . پروژه ای که ایجاد نموده اید به صورت پیش فرض دارای قابلیت های ثبت نام کابر ((Register، ورود کاربر به سایت ((Login، تغییر رمز عبور توسط کاربر ((ChangePasswordو ... و هم چین دارای فایل های jqueryو Bootstrapمی‌باشد.

قالب پیش فرض پروژه با صفحه نمایش های مختلف کاملا سازگار بوده و Responsiveهست برای درک بهتر این موضوع صفحه نمایش مرورگر خود را کوچک کنید تا شاهد این انعطاف پذیری باشید . به همین راحتی شما موفق شدید که با تنها چند کلیک یک پروژه از نوع MVCبسازید.



ایجاد یک فرم‌ساز ساده را ASP.NET MVC

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 enumانتخاب شده است که خود شامل چندین آیتم مانند Text, Radioو... 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);
 } } }
```

تنها نکته‌ایی که در کلاس فوق مهم است متد OnModelCreatingاست. از آنجائیکه رابطه کلاس Fieldو Valueیک 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 فرمusing (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("" ,"بازگشتIndex") </div>
```

ویوی 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 انگلیسیspan> <div>
    @Html.EditorFor(model => model.TitleEn, new { htmlAttributes = new {
    @class = "form-control" } }) @Html.ValidationMessageFor(model =>
    model.TitleEn, "", new { @class = "text-danger" }) </div> </div> <div>
    <span>/>عنوان فارسیspan> <div> @Html.EditorFor(model => model.TitleFa,
    new { htmlAttributes = new { @class = "form-control" } })
    @Html.ValidationMessageFor(model => model.TitleFa, "", new { @class =
    "text-danger" }) </div> </div>
```

```
<div> <span>نوع فیلد</span> <div> @Html.EnumDropDownListFor(model =>
    model.FieldType, htmlAttributes: new { @class = "form-control" })
    @Html.ValidationMessageFor(model => model.FieldType, "", new { @class =
    "text-danger" }) </div> </div> <div> <span>فرم</span> <div>
    @Html.DropDownList("FormId", (SelectList)ViewBag.FormList)
    @Html.ValidationMessageFor(model => model.FormId, "", new { @class =
    "text-danger" }) </div> </div> <div> <div> <input type="submit"
    value="ذخیره" /> </div> </div> </div> } <div> @Html.ActionLink("بازگشت",
    "Index") </div>
```

در ویوی فوق کاربر می‌تواند برای فرم انتخاب شده فیلدهای موردنظر را تعریف کند:

ویوی نمایش فرم تولید شده برای کاربر نهایی:

```
@using SimpleFormGenerator.DomainClasses @model
    IEnumerable<SimpleFormGenerator.DomainClasses.Field> @{ ViewBag.Title =
    "> { ;"نمایش فرمdiv> <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 forدر 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 HTMLنمایش 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 آرایه‌ایی از عناصر آنها را نمایش خواهیم داد:

```
[@i].FieldTyp
```

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 ShowFormرا در حالت Postبه 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);
 }
```

پایان