به نام خدا

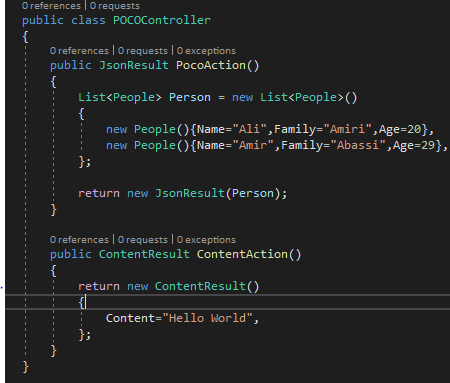
# 1-Poco , ActionMethod

C03-S01-ControllerAndActionMethod

ساخت PocoControler

چون از controller ارث بری نمیکنند بسیار سریعتر اجرا میشوند ولی باید همه چیز را برایشان تعریف کرد چون به پایه کلاسهای متد کنترلر دسترسی ندارند. برای زمانی که ما میخواهیم یک رشته یا یک Json را در خروجی داشته باشیم میتوانیم از آن استفاده کنیم.

داخل کنترلر یک کلاس ایجاد میکنیم و ActionResult که میخواهیم را ایجاد میکنیم. مثلا ما یک JSonResult میخواهیم دریافت کنیم.



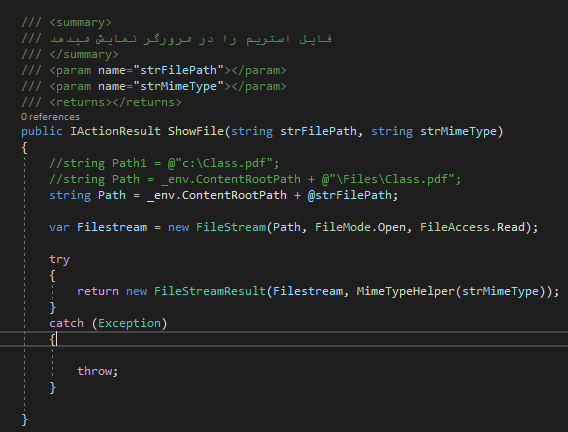
در متد دوم چون به کلاس پایه کنترلر دسترسی نداشتیم و میخواستیم یک ContentResult برگردانیم از کلمه کلیدی new استفاده کردیم و از طریق این نمونه پارامترهای سازنده مثل Content را مقدار دهی میکنیم.

# 2- FileResult

با استفاده از این متد میتوانیم یک فایل استریم را در مرورگر به صورت فایل بازگشت بدهیم و در مرورگر نمایش بدهیم.

یک کلاس ایجاد میکنیم برای دانلود یا نمایش فایل در مرور گر در لایه Common

FileResult خودش شامل FileStreamResult و FileContentResult هم میشه ولی پون IActionResult از همه آنها پشتیبانی میکنه میتونیم نوع متد رو از این نوع تعریف کنیم.



آدرس فایل و نوع آنرا دریافت میکند و با متد ShowFile در مرورگر نمایش میدهد و یا با متد DownLoad دانلود میکند.

آدرس پروژه را با IHostingEnvironment env بدست می آوریم.

string Path = \_env.ContentRootPath + @strFilePath;

برای دانلود باید فایل را به آرایه از بایتها تبدیل کنیم و سپس با خواندن مقادیر فایل آنرا برای دانلود به مرورگر ارسال نماییم.



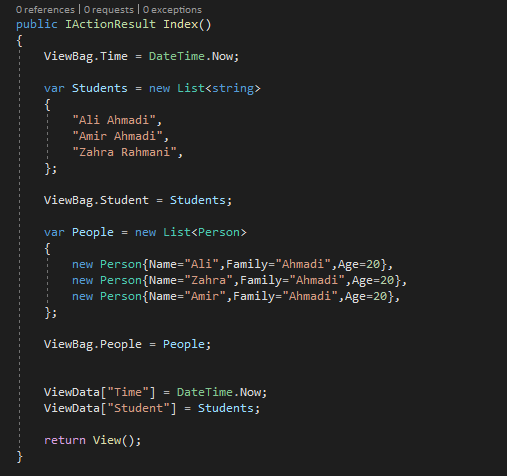
# ViewBag , TempData , ViewData

C04-S01-ViewBagViewDataTempData

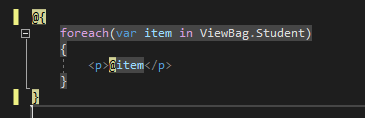
ViewBag برای انتقال داده ها از کنترلر به View استفاده میکنیم.

مثل ViewBag.Time = DateTime.Now; که کافیه توی View از @ViewBag.Time استفاده کنیم تا مقادیر آنرا نمایش دهد.

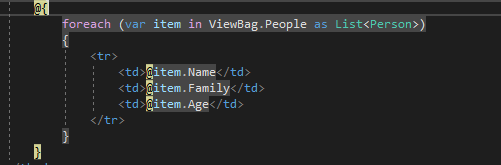
اگر بخواهیم یک لیست ارسال کنیم باید از ForEach استفاده کنیم.



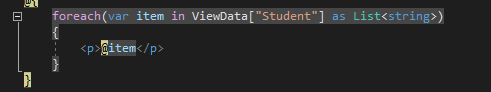
برای نمایش



در قسمت دوم برای اینکه به پراپرتی های تعریف شده دسترسی داشته باشیم و نخواهیم بریم بگردیم دنبالش که ببینم چی بود و اونها رو نمایش بدیم از کلمه کلید as توی view استفاده میکنیم و بعد نوعش رو مینویسیم.

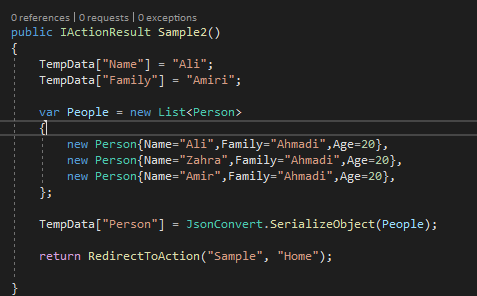


ViewData از کلاس ViewDataDictionary ارثبری میکنه و مثل viewBag داده ها رو منتقل میکنه با این تفاوت که در viewBag الزامی برای تعیین نوع داده ارسال نیست و میتون as رو بکار نبریم ولی برای ViewData حتما باید نوعش رو مشخص کنیم.



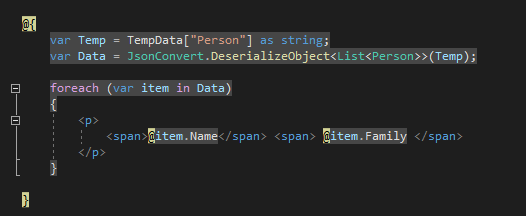
TempData

میتوان اطلاعات رو بین Controller ها و ActionMethod ها جابجا کرد.



در TempData لیست را نمیتوان انتقال داد. از روش تبدیل به Json میتوان آنرا منتقل کرد.

با Serialize تبدیل به Json کردیم و با Deserialize آنرا بر میگردانیم و در View نمایش میدهیم.



**نکته:** TempData موقتی هست و بعد از خوانده شدن یا پردازش درخواست از بین میروند چون در Session ذخیره میشوند.

# Font Bootstrap yekan

C04-S02-SendFormData

فایلهای یکان را داخل پوشه wwwroot در پوشه fonts کپی کردیم.

سپس کدهای مربوط به فونت را داخل فایل Site.css مینویسیم و کد مربوط به فونت را هم داخل تگ body اضافه میکنیم.

@font-face {

font-family: "yekan";

src: url(/fonts/Yekan.eot);

src: url(/fonts/Yekan.eot?#iefix) format("embedded-opentype"), url(/fonts/Yekan.woff) format("woff"), url(/fonts/Yekan.ttf) format("truetype"), url(/fonts/Yekan.svg#BYekan) format("svg");

font-weight: normal;

font-style: normal

}

و در body

font-family:yekan;

و معرفی انواع کنترلهای روی فرم Input

Input text , ….

# Method Post Method Get

در صورتیکه از TagHelper ها استفاده نمیکنیم و میخواهیم از تگهای خود Html استفاده کنیم برای CheckBox

Input type=”checkbox”

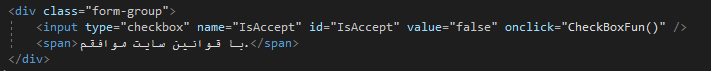
چک باکس به سمت سرور ارسال نمیشود. برای اینکه بتوانیم این مقدار را ارسال کنیم. روش اول مقدار را برابر با True قرار میدهیم. یعنی

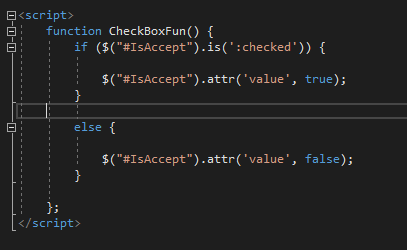
<input type=”checkbox” value=”true” checked />

که باعث میشود همیشه تیک خورده باشد و مقدار را true ارسال میکند.

اما اگر نخواهیم که با بازشدن صفحه تیک خورده باشد و کاربر مجبور باشد که تیک بزند یا نزند از JQuery استفاده میکنیم.

یک تگ id به تگ Input اضافه میکنیم و یک تگ onclick





<script>

// تعریف تابع

function CheckBoxFun() {

if ($("#IsAccept").is(':checked')) {

$("#IsAccept").attr('value', true);

}

else {

$("#IsAccept").attr('value', false);

}

};

</script>

به وسیله # میتوانیم به مقدار id دسترسی پیدا کنیم و از علامت . برای دسترسی به کلاس اون تگ

و چک کردیم که اگر تیک خورده بود صفت value ار true قرار میدهد و در غیر اینصورت مقدار false را جایگزین میکند.

# دریافت عکس کاربر

در تگ فرم باید مقدار enctype را برای ارسال فایل multipart تعریف کنیم.

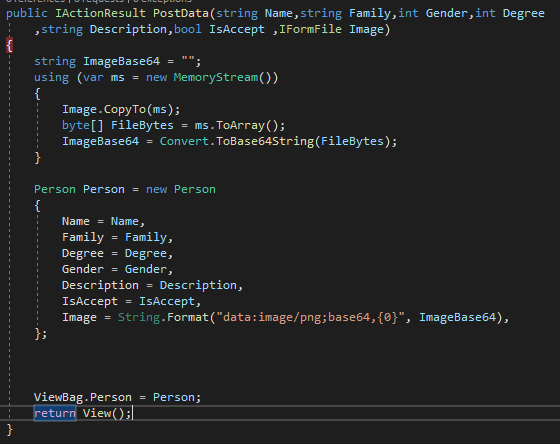


صفت accept مشخص میکند که چه فایلهایی را کاربر میتواند انتخاب نماید.



اگر بخواهیم در Action Method فایل را دریافت کنیم باید نوع پارامتر را IFormFile تعیین کنیم که از کلاس AspNetCore.Http ارثبری میکند.

حالا عکس را به فرمت base64 تبدیل میکنیم و سپس به فرم ارسال میکنیم.



یک نمونه از MemoryStream می سازیم و فایل ارسال شده را در نمونه سازی کپی میکنیم. Image.CopyTo(ms)

این کار فایل را به حافظه رم منتقل میکند.

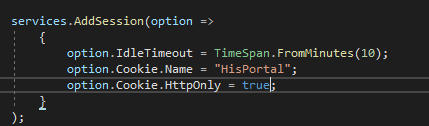
حالا اون رو به آرایه ای بایتها تبدیل میکنیم و فایل تبدیل شده را به base64 تبدیل میکنیم و در زمان از تابع

Image = String.Format("data:image/png;base64,{0}", ImageBase64)

استفاده میکنیم.

FileUploader از نوع bootstrap هم در این فایل تعریف شده است.

# Session & cookie



option.Cookie.HttpOnly = true;

این ویژگی باعث میشه که کدهای جاوا اسکریپت (مخرب) نتونن دسترسی به اطلاعات کوکی داشته باشند. از طریق حملات XSS اگر این گزینه فعال نباشه میشه اطلاعات ذخیره شده کوکی رو به سرور دیگه ارسال کرد.[[1]](#footnote-1)

در متد Configure هم باید میان افزارش رو اضافه کنیم. ترتیب قرار گیری هم مهمه. باید قبل از app.UseMVC باشد.



در .Net Core 3 به بعد از app. UseEndpoints به جای UseMVC ایجاد شده است.[[2]](#footnote-2)

## SetSession & GetSession



میتوان اطلاعات رابه صورت تک به تک درون Session ذخیره کرد.

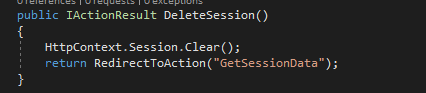
برای اینکه بتوانیم اطلاعات پیچیده مانند لیست را در session ذخیره کنیم از روش Json استفاده میکنیم که در مثال فوق مشخص است.

HttpContext.Session.SetString("Person", JsonConvert.SerializeObject(person));

و

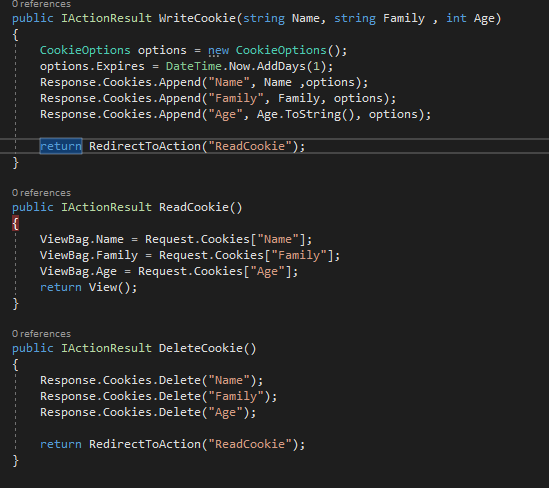
ViewBag.Person = JsonConvert.DeserializeObject<Person>(HttpContext.Session.GetString("Person"));

و برای حذف Session



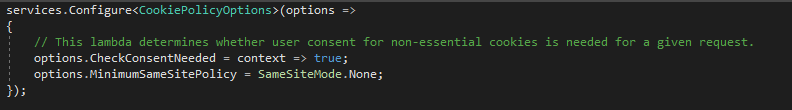
## WriteCookie Read Delete

خواند نوشتن و حذف کوکی



قوانین GDPR قوانین مربوط به کوکی ها

در زمان اجرا به کاربر پیام میدهد که کوکی میخواهد توی سیستم ذخیره شوند و ... اگر بخواهیم پیام cookie رو حذف کنیم باید مقدار options.CheckConsentNeeded = context => true; را به false تغییر بدهیم.



# TagHelper

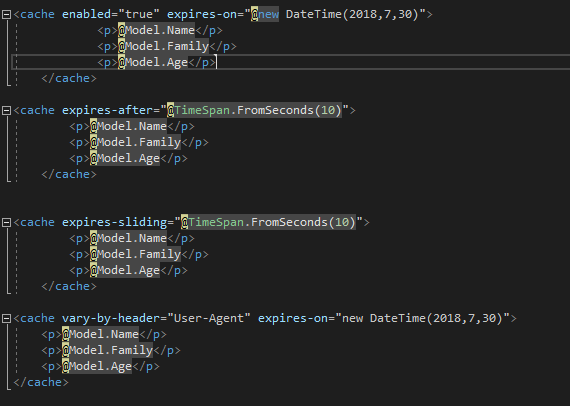
## Caching – Cache TagHelper

C06-S01-CacheTagHelper

ذخیره یکسری داده درحافظه وب سرور

هدف کم کردن تعداد مراجعات به منابع سرور و دیتابیس که بعد از دریافت اونها رو نگهدرای کنیم که برای مراجعه بعدی نیاز نباشه مجدد از دیتابیس بخونیم و سرعت بالا تر میره. در حافظه وب سرور نگهداری میشه.

این اطلاعات تاریخ انقضا دارند یعنی میتونیم بهش زمان بدیم که تا کی بمونند.



**enabled**="true" حالت فعال بودن کش یا نبودن

**expires-on**="@new DateTime(2018,7,30) حذف شدن بعد از تاریخ ...... / زمان نگهداری تا تاریخ

**expires-after**="@TimeSpan.FromSeconds(10)

[" نگهداری](mailto:\" نگهداری expires-sliding=\"@TimeSpan.FromSeconds(10))

[expires-sliding="@TimeSpan.FromSeconds(10)](mailto:\" نگهداری expires-sliding=\"@TimeSpan.FromSeconds(10))"> اگر چه زمانی داده ها از وب سرویس خوانده نشد حافظه کش خالی شود. در واقع زمان ارسال درخواست ها چک میکند اگر این زمان سپری شد و هیچ درخواستی وجود نداشت حافظه کش شده را حذف میکند.

vary-by-header="User-Agent" expires-on="new DateTime(2018,7,30)">

یکی از محتویات header در خواست User-Agent است. محتوا داخل وب سرور برای هر User-Agent ذخیره میشود. این ویژگی اطلاعات و مشخصات مرور گر را در خودش نگهدای میکند. در Insepect element در تب Network میتوانیم درخواستهای ارسال شده را ببینیم. این درخواستها و اطلاعات برای هر کاربر اختصاصی عمل میکند. به عنوان مثال اگر یک متد را در دو مرور گر یا دو کلاینت اجرا کنیم در مرورگری که داده ها کش شده است این اطلاعات را نمایش میدهد و در مرورگر دیگر با خطا مواجه خواهیم شد.

## Anchor TagHelper

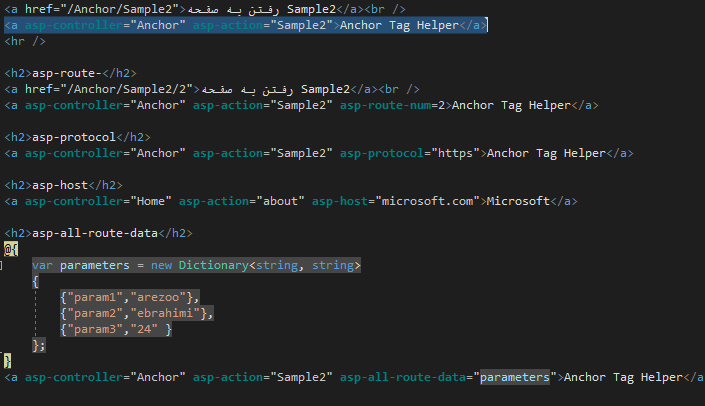
C06-S02-AnchorTagHelper

Anchor یعنی متنهای لینک دار که تبدیل به تگ a میشود.

<a asp-controller="Anchor" asp-action="Sample2">Anchor Tag Helper</a>

کدهای سمت سرور را به کدهای Html تبدیل میکنند.

از دیگر ویژگی های این Anchor TagHelper ها



asp-route-id

ارسال یک id به یک اکشن متد. میتوان نام ارسال به سمت اکشن متد را هم تعیین کرد مثلا بجای id از num یا نام دلخواه استفاده کنیم.

asp-route-num=2

اگر از id استفاده کنیم بلافاصله بعد از نام Action method / و id درج میشود ولی مقادیر دیگه نام مقادیر به همراه نام و ؟ ارسال میشود. که این مربوط به مسائل مربوط به routing یا مسیریابی است.

Sample2?num=2

asp-protocol="https"

پروتکل به کار رفته در آدرس را تعیین میکند که http است یا https

asp-host="microsoft.com"

میتوان ادرس هاست را تعریف کنیم. اکشن متدی که به آن نیاز داریم و نام کنترلر

در مثال فوق با کلیک بر روی لینک به صفحه about در کنترلر home و هاست Microsoft منتقل خواهیم شد.

<a asp-controller="Anchor" asp-action="Sample2" asp-all-route-data="parameters">Anchor Tag Helper</a>

ارسال چند پارامتر به یک اکشن متد. از کلاس Dictionary باید استفاده کنیم و برای هر کدام یک کلید و یک مقدار تعریف میکنیم.

هر کلاس Dictionary از یک key و یک value تشکیل شده است. در این مثال از param به عنوان کلید و مقادیر فوق استفاده کردیم.

asp-all-route-data="parameters" این ویژگی یک نمونه از کلاس dictionary را بعنوان پارامتر قبول میکند.

# Form TagHelper

C06-S03-TagHelpers



<form asp-controller="Home" asp-action="Register" enctype="multipart/form-data">

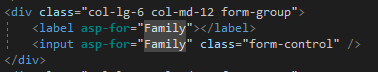
اطلاعات فرم به کدام اکشن متد و کدام کنترلر ارسال شود. برای ارسال داده ها به اکشن متدها استفاده میشود. و به صورت پیش فرض از متد post استفاده میکند.

در زمان اجرا یک توکن و اینپوت به صورت hidden ایجاد میکند

<input name = “\_\_RequestVerificationToken” value= ………………….”

این توکن به منظور جلوگیری از حمله CSRF یا Cross-Site Request Forgery انجام میشود.

وقتی که بجای Form TagHelper از خود تگ Form استفاده بکنیم دیگه این اینپوت ساخته نشده و امنیت کاهش پیدا خواهد کرد.



asp-for

مقدار input را داخل پراپرتی مدل قرار میدهد.

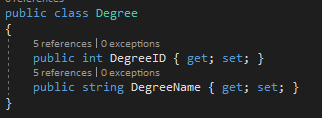
### اعتبار سنجی سمت کاربر

asp-validation-for="Image" برای اعتبار سنجی مقدادیری که در مدل ها تعریف کردیم استفاده میشود و کاربرد دارد

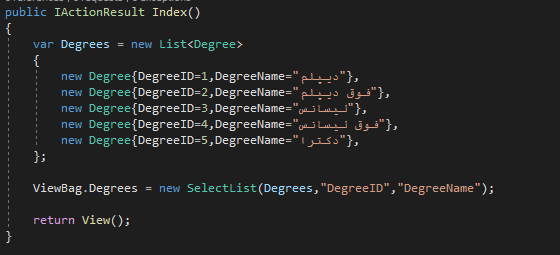
در قسمت بعدی کاملا مفصل توضیح داده شده است.

## Dopdown list combo box listbox

برای این پراپرتی ها از تگ SelectList استفاده میکنیم که در Anchor ها asp-items هستند



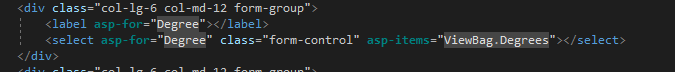
یک کلاس تعریف کردیم برای سطح تحصیلات



در اکشن متد ایندکس برای سطح تحصیلات با استفاده از کلاس degree یک لیست ساختیم

SelectList یک لیست را به عنوان پارامتر اول و پارامتر دوم به عنوان مقدار value و پارامتر سوم یک text را میپذیرد.

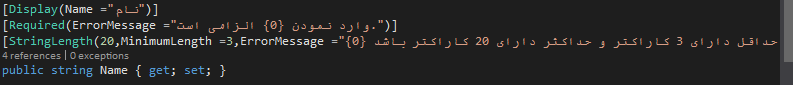
حال در view با viewbag اون لیست رو مقدار دهی میکنیم.

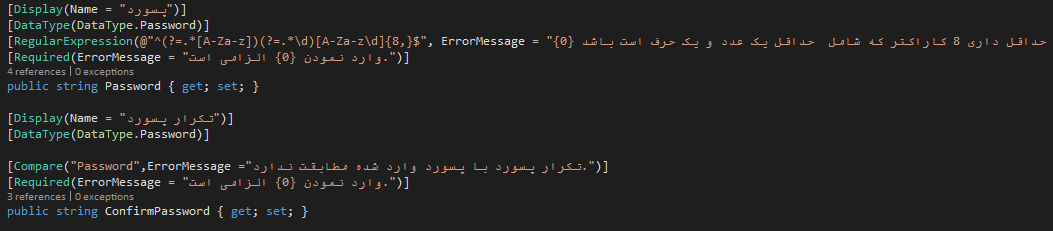


# روش اعتبار سنجی فرم ورود اطالعات

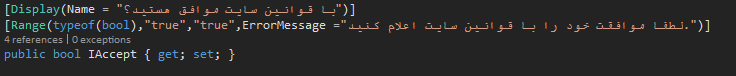
## DataAnnotations

C06-S04-DataAnnotations





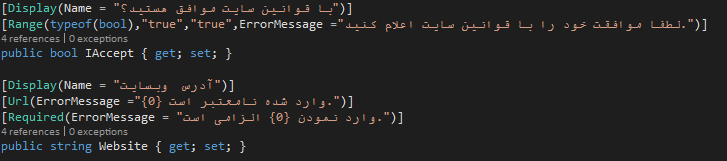
Compare مقایسه کردن دو پراپرتی مثل پسورد و تکرار پسورد



Range مشخص کردن محدوده قابل وارد کردن توسط کاربر

اعتبارسنجی Checkbox به روشی که فقط از Required استفاده کنیم درست نیست و اجباری در نظر نمیگیرد برای این کار باید از روش Range استفاده کنیم .

پارامتر اول نوعش رو مشخص میکنیم، پارامتر دوم مقدار maximum و مقدار سوم minimum است.



[StringLength(20,MinimumLength =3,ErrorMessage =""]

asp-validation-for="Image" برای اعتبار سنجی مقدادیری که در مدل ها تعریف کردیم استفاده میشود و کاربرد دارد

این پیام های خطا درون view به کاربر نمایش داده میشود

برای اینکه این پیامهای خطا به درستی عمل کنند باید پکیج jquery-validation و jquery-validation-unobtrusive نصب شده باشد و همچنین jquery.validate.js و jquery.validate.unobtrusive.js در صفحه لود شده باشد. در واقع لینک آن را به layout باید اضافه کنیم.

<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script

در صورتیکه کاربر جاوا اسکریپت سمت کاربر یعنی مرورگر را غیر فعال کند دیگه فرم اعتبار سنجی نمیشود کاربر میتواند مقادیری که خودش میخواهد وارد کند و اعتبار سنجی را عملا دور بزند. بانوشتن دستور about:config در آدرس بار مرورگر میتوانیم جاوا را غیر فعال کرده و عملا اعتبار سنجی را ساقط کنیم.

اگر بخواهیم Checkbox ها هم اعتبار سنجی سمت کلاینت انجام بدهند میتوان از کد Jquery زیر استفاده کرد.

<script>

// extend jquery range validator to work for required checkboxes

var defaultRangeValidator = $.validator.methods.range;

$.validator.methods.range = function (value, element, param) {

if (element.type === 'checkbox') {

// if it's a checkbox return true if it is checked

return element.checked;

} else {

// otherwise run the default validation function

return defaultRangeValidator.call(this, value, element, param);

}

}

</script>

برای تعریف اعتبار سنجی سفارشی قسمت Custom Attribute Validation را مطالعه کنید.

### ModelState اعتبار سنجی سمت سرو

برای اعتبار سنجی سمت سرور از ModelState استفاده میشود.

# Custom TagHelper

C06-S05-CustomTagHelper

ایجاد TagHelper سفارشی

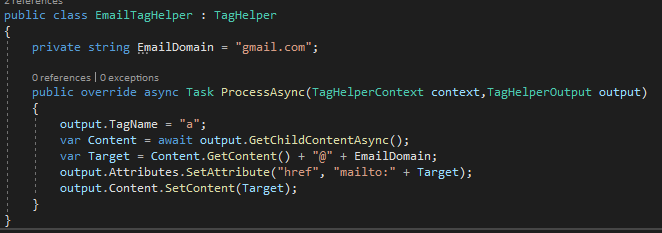
میخواهیم یک تگ هلپر بنویسیم مثل <email> که هرجا مثلا نوشتیم بیاد خودکار خودش اون رو تبدیل به تگ a بکنه

<a href="mailto:Marketing@example.com">[Marketing@example.com</a](mailto:Marketing@example.com%3c/a)>

و هرجایی خواستیم از این استفاده کنیم

یک کلاس تعریف میکنیم که از TagHelper ارثبری کنه

در کلاس TagHelper یک متد داره به نام Process که باید خودمون اون رو Overide کنیم دوباره بازنویسیش کنیم.



و در View تعریفش میکنیم



# سفارشی سازی Attribute

C06-S06-CustomValidationAttribute

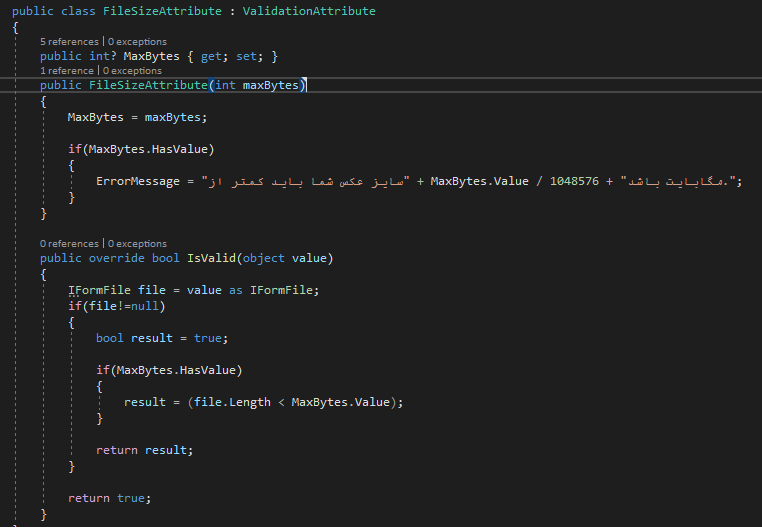
میخواهیم یک ویژگی سفارشی بسازیم مثلا یک عکس را اعتبار سنجی کند اگر اندازه اش از یک مقداری بیشتر بود به کاربر پیام بدهد. این اتریبیوت ها بالای پراپرتی های مدل یا کنترلر و ... تعریف میشوند. مثل [HttpPost] و .....

FileSizeAttribute

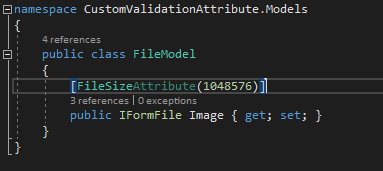
یک کلاس میسازیم که این کلاس باید از کلاس ValidationAttribute ارثبری کند. یک متد نوشتیم و پارمتر ورودی اون رو از نوع اint گرفتیم. میخواهیم حجم فایل رو به این متد پاس کنیم. و شرط گذاشتیم که اگر این فایل حجمش بیشتر از 1 مگابایت بود به کاربر پیام خطا بدهد.

این کلاس ValidationAttribute یک متد به نام IsValid دارد که یک Object را به عنوان ورودی میپذیرد.

ما یک نمونه از کلاس IFormFile ساختیم که نمونه Object ورودی رو به نمونه از کلاس IFormFile تبدیل کنیم. حالا چک کردیم که اگر مقدار داشت و خالی نبود بیاد چک کنه که حجم فایل چقدره و مقدار true یا false رو برگشت بده



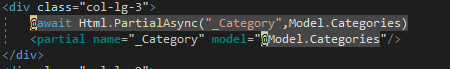
حالا attribute ساخته شده رو به پراپرتی کلاس مدلی که ساخته بودیم منتقل کردیم.



# Partial View

C08-S02-PartialView

برای ارسال داده ها درون View هایی که از پارشال ویو درونشون استفاده کرده ایم باید یک مدل یا ویو مدل ViewModel بسازیم و مقادیر را به اونها پاس کنیم. در غیر اینصورت با خطا مواجه خواهیم شد. پس در دو حالت زیر مدل را به partial view ارسال کرده ایم.



هر دو یک کار انجام میدهند یکی در قالب TagHelper و دیگری یک کد C#

در partial view ها باید مدلی که ارسا ل میکنیم را هم متناسب با مدل تغییر دهیم.

# ViewComponents

C08-S03-ViewComponent

از این view ها میتوانی مانند partial view ها استفاده کرد ولی با قدرت بیشتر. در واقع نوع خاصی از view هست که برای تقسیم view های بزرگ به اجزای کوچکتر و برای استفاده مجدد از یک view از اون استفاده میشود. بر روی کنترلر تکیه نمیکنند و کلاس خاص خودشون رو پیاده سازی میکنند و از Dependency injection میشه توشون استفاده کرد. یا وقتی که کدها قابلیت استفاده مجدد دارند و آنقدر پیچیده هستند که نمیشه از Partial view استفاده کرد. برای سبد خرید، منو های پیچیده، مقالات تازه منتشر شده و ... . ولی دقیقا نمیدونم خاصیتشون چیه و درک نکردم. ولی راه ساختنشون اینه که یک کلاس که از کلاس ViewComponent ارثبری میکند میسازیم.

یک پوشه میسازیم درون پوشه یک کلاس مثلا ما برای Category ها میخواهیم بسازیم

...

....

...

...

# Area

C08-S04-Areas

# DependencyInjecti

C09-S03-DependencyInjectionInView

## Repository

یک واسط بین کنترل و مدل است . Repository دسترسی به داده هاست. در لغت به معنای مخزن یا محل ذخیره سازی داده ها. مثلا اگر کنترلر نیاز به داده خاصی داشته باشه / حذف / درج / ویرایش و .... Repository این کار رو برایش انجام میده. در واقع یک لایه ای بین کنترلر و دیتا بیس است. تا کنترلر مستقیما با دیتابیس در ارتباط نباشد.

### سازنده Static

سازنده استاتیک برای کلاسهای استاتیک مورد استفاده قرار میگیرد. در این کلاسها بدون نمونه سازی میتوانیم به پراپرتی ها و متدهای آن دسترسی داشته باشیم.

ثابت هستند و در قسمتی از حافظه ذخیره میشوند. برای مقدار دهی اعضای استایتک استفاده میشود و هر کلاس فقط یک سازنده استاتیک میتواند داشته باشد.

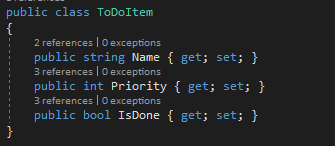
برخلاف سازنده معمولی سازنده استاتیک پارامتر ورودی نمیتواند داشته باشد ولی سازنده معمولی محدودیت پارامتر ندارند.

سازنده استاتیک فقط یکبار و قبل از ایجاد اولین نمونه از کلاس فراخوانی میشود. ولی سازنده غیر استاتیک هر باری که کلاس اجرا شود فراخوانی میشوند و فیلدها مقدار دهی میشوند.

به وسیله دستور @Inject سرویسی که ساخته ایم را درون view مقدار دهی میکنیم.

یک کلاس ساختیم و یک سازنده برای اینکه مقداری داشته باشیم تا بتونیم اون رو بررسی و چک بکنیم در کلاس سازنده یک حلقه گذاشتیم که مقدار 50 تا عمل انجام شده و اولویت بندی رو انجام بده. این برنامه یک برنامه ایه که کارهای انجام شده و انجام نشده رو نشون میده و اولیتهاشون رو نمایش میده. حالا میخواهیم این مقادیر رو توی View نمایش بدیم.

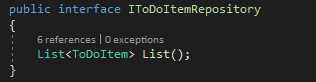
هربار که این کلاس صدا زده بشه سازنده اون تعداد 50 تا رکوردی که گفتیم رو میسازه.



و الگوی Repository اون

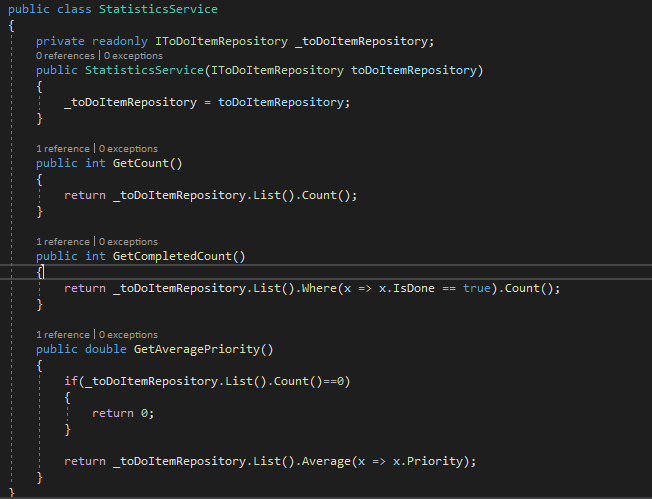


و برایش یک اینترفیس ساختیم و این سرویس رو توی Startup تعریف کردیم.

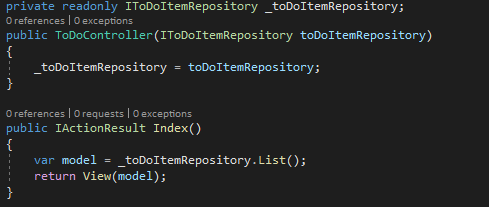


توی این سرویس از تزریق وابستگی انجام شده.

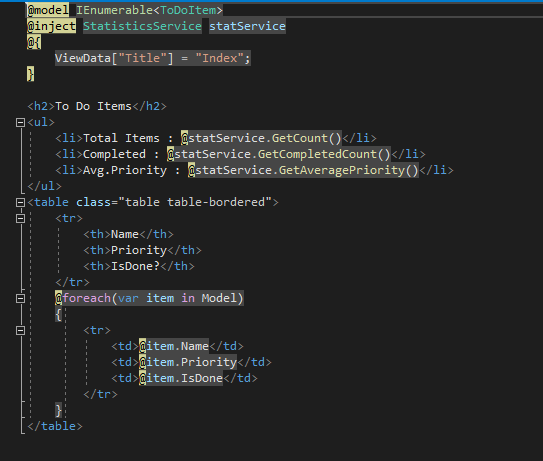
سه تا متد ساختیم برای بدست آوردن تعداد کل، تعداد انجام شده و متوسط اولیت ها



توی کنترلر سازنده رو صدا میزنیم و مقدارش رو به view میفرستیم.



حالا توی view



از تزریق وابستگی استفاده میکنیم و مقدادیر رو نمایش میدهیم.

# ServiceLife

C09-S04-ServiceLife

طول عمر سرویس ها

AddTransiant

به ازای هر درخواست یک نمونه جدید از کلاس ایجاد شده و در اختیار برنامه قرار میگیرد.

AddScoped

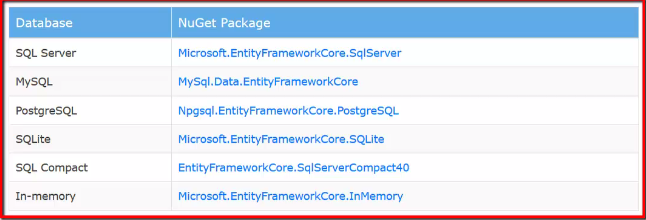
به ازای هر درخواست Http Request فقط یک نمونه از این کلاس ایجاد شده و در طول حیات این درخواست به اشتراک گذاشته میشود.

AddSingleton

در طول حیات برنامه وب، فقط و فقط یک نمونه از کلاس ایجاد شده و هر زمان که نیاز باشد به اشتراک گذاشته میشود.

# Install EF Core

## C10-S02-Attributes-1

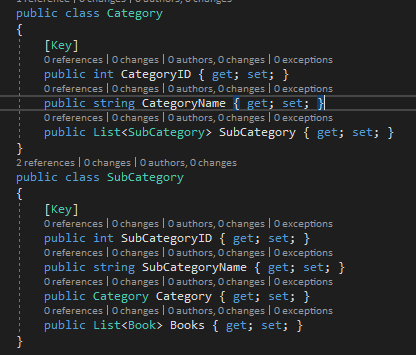


ارتباط یک به چند

1 به n

از سه روش میتوان این رابطه را پیاده سازی کرد.

روش اول:



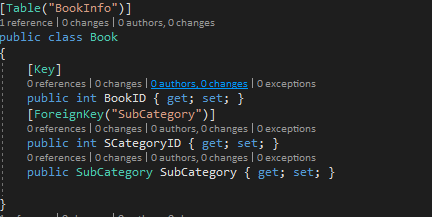
هر دسته بندی اصلی متیوانیم لیستی از زیر دسته ها را داشته باشیم ولی هر زیر دسته متعلق به یک دسته بندی اصلی هست. پس در کلاس اول

Public List<subcategory> SubCategory {get;set;}

و در کلاس دوم

Public Category Category {get;set;}

روش دوم:



استفاده از کلید فرعی بیشتر زمانی کاربرد دارد که میخواهیم نام کلید خارجی هم نام با کلید اصلی اون جدولی که باهاش ارتباط دارد نباشد.

هر کتاب تنها میتواند در یک زیر دسته قرار بگیرد.

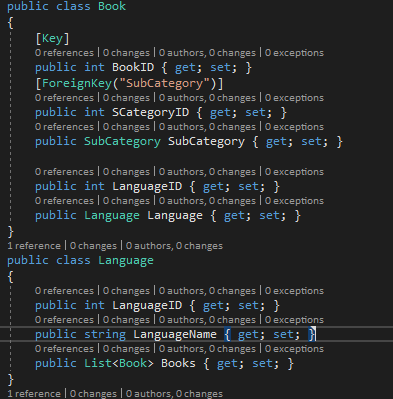
یک کلید فرعی برایش تعریف کردیم public int SCategoryID { get; set; } و سپس ویژگی ForeignKey رو بهش اختصاص دادیم. یک نمونه هم از کلاس SubCategory میسازیم برای ارتباط

در کلاس SubCategory هم یک لیست از نوع Book تعریف کردیم.

public List<Book> Books { get; set; }

و همینطور که مشخص است نام کلید خارجی متفاوت است و همنام کلاس نیست.

روش سوم:



هر کتاب میتواند به یک زبان نوشته شده باشد پس در جدول زبان ما لیستی از کتابها رو داریم. در این روش نام کلید خارجی دقیقا همنام کلید اصلی جدولی است که با اون رابطه داره

ارتباط چند به چند

Database Attributes

[Key] تعریف کلید اصلی

در حالت کلی خودکار کلید اصلی به صورت Identity ایجاد میشود اگر نخواهیم این اتفاق بیفتند باید از ویژگی

DatabaseGenerated استفاده کنیم. که سه نوع مقدار دارد

[DatabaseGenerated(DatabaseGeneratedOption.None)]

None

Identity به صورت خودکار مقدار میگیرد، و فقط در زمان درج مقدار میگیرد و قابل تغییر نیست.

Computed وقتی که یک سطر به جدول اضافه میشه یا ویرایش میشه به صورت خودکار db اون رو ویرایش میکنه یا اضافه میکنه.

[InverseProperty] در حالتی که یک جدول دو بار رابطه 1 به n با یک جدول دیگه داشته باشد از این خاصیت باید استفاده کنیم و نام آن رو باید درون این پراپرتی تعریف کنیم. مثل آدرس اول و دوم مشتری ها

[Required] اجباری کردن فیلد

# FluentApi

C10-S05-FluentAPI1

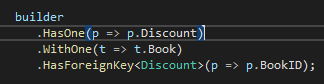
1. public byte[] RowVersion { get; set; }

از نوع timestamp در fluentapi به صورت زیر تعریف میشود.

<https://www.learnentityframeworkcore.com/configuration/fluent-api/valuegeneratedonaddorupdate-method>

1. protected override void OnModelCreating(ModelBuilder modelBuilder)
2. {
3. modelBuilder.Entity<Contact>()
4. .Property(p => p.LastAccessed)
5. .ValueGeneratedOnAddOrUpdate();
6. }
7. public class Contact
8. {
9. public int Id { get; set; }
10. public string FullName { get; set; }
11. public string Email { get; set; }
12. public DateTime LastAccessed { get; set; }
13. }

رابطه یک به یک



رابطه یک به n

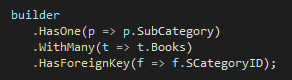
در رابطه یک به چند مثل کتاب و دسته بندی ها

و یا رابطه دسته بندی و زیر دسته ها

هر کتاب میتواند در یک دسته قرار بگیرید و هر دسته میتواند لیستی از کتابها داشته باشد.

ا زسه روش میتوانیم استفاده کنیم.

روش اول:



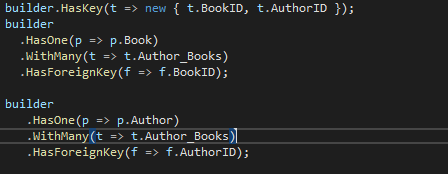
builder

.HasOne(p => p.Book)

.WithMany(t => t.Author\_Books)

.HasForeignKey(f => f.BookID);

رابطه n به n



در رابطه چند به چند ابتدا دو کلید اصلی که دارند رو تعریف میکنیم و سپس مثل رابطه یک به چند همون ترکیب را قرار میدهیم.

هر کتاب چند نویسنده میتونه داشته باشد و هر نویسنده میتونه چندین کتاب نوشته باشه.



C10-S11-AdminPanel رابطه چند به چند در Book\_Category

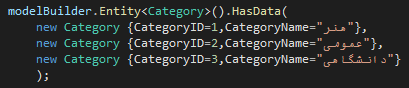
تعریف کلید اصلی

تعریف فیلد برای عکس image

تغییر نام جدول از نام پیش فرض به نام دیگر

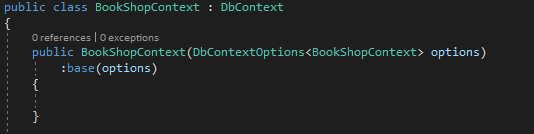
HasColumnName هم برای تغییر نام فیلد در db

HasData برای وارد کردن فیلدها در هنگام ساخت db کاربرد دارد.



### قابلیت MethodChaning

این قابلیت این امکان رو می دهد که برای هر پراپرتی عملیات Mapping رو توی چند خط انجام بدهیم. و همه در یک خط قابل قبول است. و باعث میشه که عملیات بسیار ساده تر و کدها کوتاهتر بشود.



از طریق سازنده باید مقدار ConnectionString رو به سازنده کلاس DbContext ارسال کنیم.

برای این منظور یک سازنده تعریف میکنیم. پارامتر ورودی این سازنده مقدار ConnectionString رو میگیره و با استفاده از کلمه کلیدی base آن را به سازنده کلاس DbContext ارسال میکنه.

در واقع سازنده به کلاس سازنده bookshop تزریق میشود و مقدار دریافت شده را به سازنده dbContext می فرستند.

# Update-Database

میتوان نام هر migration کی اد کرده بودیم را برای update تعیین کنیم در حالت کلی اگر تعیین نکنیم همه migration ها را روی db اعمال میکند.

### خطای عدم دسترسی کاربر Sa بعد از ساخت db توسط update-database

USE Portal\_New

GO

ALTER DATABASE Portal\_New set TRUSTWORTHY ON;

GO

EXEC dbo.sp\_changedbowner @loginame = N'sa', @map = false

GO

sp\_configure 'show advanced options', 1;

GO

RECONFIGURE;

GO

sp\_configure 'clr enabled', 1;

GO

RECONFIGURE;

GO

# CRUD

C10-S08-CRUD1

## C10-S12-TreeViewCategory

هر کتاب میتواند در چند دسته قرار بگیرد و هر دسته میتواند چندین کتاب را در خود جای دهد. بنابراین رابطه کتاب و دسته بندی را به رابطه چند به چند تقسیم کردیم. در این حالت یک جدول واسط تعریف میشود. در قسمت رابطه چند به چند این مطلب توضیح داده شده است.

حالا میخواهیم اطلاعات DropDownList ها به کاربر نمایش بدهیم

SelectList

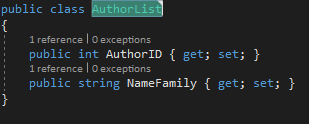
ViewBag.PublisherID = new SelectList(\_context.Publishers, "PublisherID", "PublisherName");

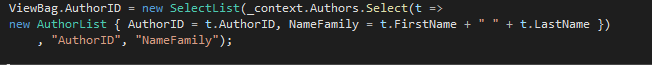
پارامتر اول ارتباط با بانک اطلاعاتی را میگیرد

پاارمتر دوم ID های جدولی که میخواهیم برگردانیم و پارامتر سوم نام نمایشی جدول است. دراینجا ما شناسه منتشرکنندگان و نام آنها را توسط ViewBag به view ارسال کردیم.

حالا اگر بخواهیم نام و نام خانوادگی نویسنده را هم ارسال کنیم چه کاری باید انجام بدهیم؟ چون نمیتوانیم چند مقدار توی پارامتر سوم ارسال کنیم

برای چسباندن نام و نام خانوادگی به همدیگه یک viewmodel می سازیم





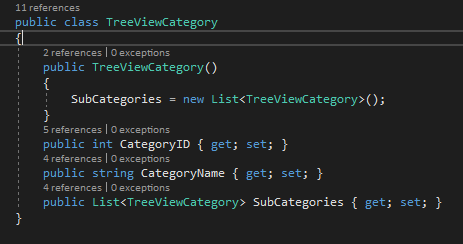
ViewBag.AuthorID = new SelectList(\_context.Authors.Select(t => new AuthorList { AuthorID = t.AuthorID, NameFamily = t.FirstName + " " + t.LastName }), "AuthorID", "NameFamily");

توسط select و linq کلاسی که ساخته ایم AuthorList رو مقدار دهی میکنیم. و حالا AuthorID را به عنوان value در select option قرار میدهیم و namefamily رو به عنوان text اون ارسال میکنیم.

حالا برای نمایش دسته بندی های ما نیاز به

شناسه دسته بندی، نام دسته بندی و نیاز به زیر دسته های یک دسته بندی داریم.

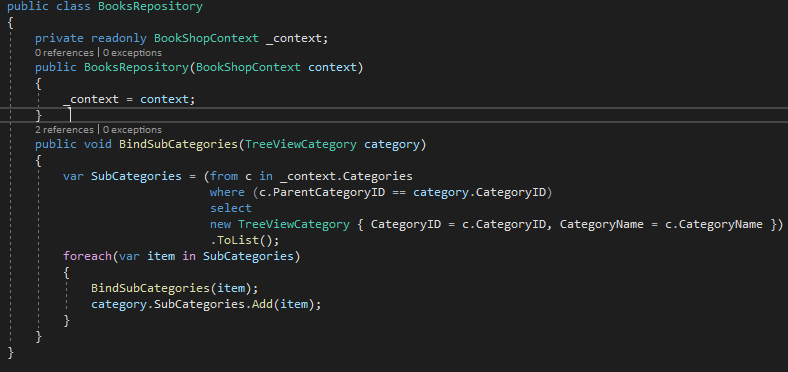
بنابراین یک کلاس تعریف میکنیم به نام TreeViewCategory



پراپرتی شناسه و نام که مشخص هستند یک لیستی هم از همین جدول برای زیر دسته ها نیاز داریم. مثل دسته هنر که شامل نقاشی / نقاشی روی کاغذ، سفال و ... هست. پس نوعشون لیستی از Treeviewcategory هست.

یک سازنده میسازیم در این سازنده یک نمونه از لیست Treeviewcategory میسازیم و اون رو داخل subcategories میریزیم.

حالا یک کلاس Repository میسازیم برای bind کردن داده های SubCategory



در این متد یک نمونه از TreeViewCategory را به عنوان پارامتر ورودی دریافت میکند

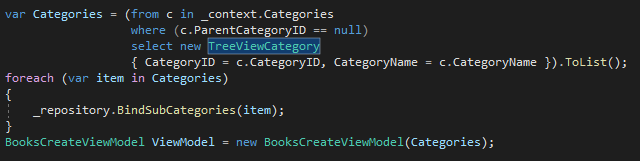
Treeviewcategory سه تا پراپرتی دارد شناسه و نام و subCategories

یک متغیر تعریف کردیم و زیر دسته های دسته Category رو از db استخراج کردیم و ریختیم توی متغیر subcategories

حالا هر کدوم از این زیر دسته ها رو با foreach پیمایش کردیم و یکی یکی این زیر دسته ها رو به پراپرتی subcategories نمونه category اضافه (add) کردیم.

پس الان پراپرتی subCategories نمونه category شامل زیر دسته های دسته بندی Category هست.

حالا به کنترلر برمیگردیم و دسته بندی های پدر یعنی اونهایی که مقدار ParentId = null هست رو داخل متغیر میریزیم.

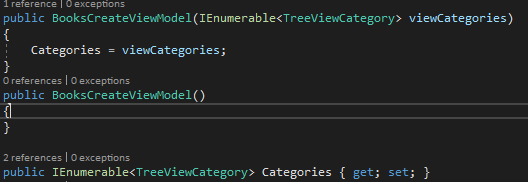


در متغیر اول مقادیری از دسته بندی که والد ندارند را اسختراج کرده ایم. سپس با حلقه یکی یکی آنها را چک میکنیم که زیر مجموعه دارند یا خیر.

با تزریق وابستگی Bookrepository به متد Bindsubcategories دسترسی پیدا کرده و زیرشاخه ها را به آن اضافه میکنیم.

اکنون با ویومدلی که ساختیم BooksCreateViewModel متغیر Categories رو به سازنده اش ارسال میکنیم.

برای اینکار یک سازنده برای ویو مدل میسازیم



یک پراپرتی از نوع لیستی از TreeViewCategory به ویومدل اضافه میکنیم و از طریق سازنده اون رو مقدار دهی میکنیم.

حالا این ویومدل ساخته شده را به view پاس میدهیم.

# نمایش پیام خطا برای dropdown list ها

تغییر خطای پیش فرض در startup.cs یک سرویس اضافه میکنیم.

services.AddLocalization(options => { options.ResourcesPath = "Resources"; });

services.AddMvc(options =>

{

var F = services.BuildServiceProvider().GetService<IStringLocalizerFactory>();

var L = F.Create("ModelBindingMessages", "BookShop");

options.ModelBindingMessageProvider.SetValueMustNotBeNullAccessor(

(x) => L["انتخاب یکی از موارد لیست الزامی است."]);

});

## گروه بندی لیست ها و نمایش غیر تکراری در جدول

در جدول کتابها چون بیش از یک نویسنده داریم و کتابها تکرار میشوند یعنی به تعداد نویسندگان ردیف و کتاب درج میشود.

باید ابتدا کتابها را بر اساس نویسندگان گروهبندی کنیم و سپس با حلقه for each انها را به هم مرتبط کرده و لیست نهایی را جهت نمایش به view ارسال کنیم.

C10-S14-ShowBooks

## Eager loading

بارگذاری مشتاقانه

زمانی استفاده میشود که بخواهیم داده های مرتبط با یک موجودیت را به محض خوانده شدن اون موجودیت به صورت همزمان بارگذاری میکنیم.

مثل اطلاعات کتاب که همراه با آن اطلاعات نویسندگان، زبان، مترجمان و .... را بار گزاری میکنیم.

به وسیله دستور Include و ThenInclude

## Explicit Loading

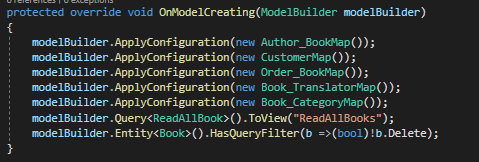
بارگذاری صریح

در این روش ابتدا داده های مرتبط با هر entity را به صورت صریح بارگذاری میکنیم. با یک کوئری از دیتابیس میگیریم و سپس داده های مرتبط با اون رو با load از دیتابیس میگیریم.

# Global Query Filter فیلتر کردن همیشگی داده ها

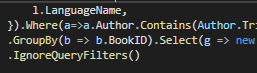
C10-S22-QueryFilters

مثلا ما همیشه میخواهیم که در همه جداول isactive = false نمایش داده نشود. و هر جا که خواستیم آنها را ببینم. یعنی نمیخواهیم در هر فرم یا گزارش و ... موارد حذف شده یا غیر فعال شده نمایش داده شوند.



در این روش از HasQueryFilter مقداری که میخواهیم را قرار میدهیم و هر کجا که ارجاعی به جدول فوق داشته باشیم این شرط اعمال میشود.

و هر جا که نخواستیم این شرط اجرا بشود از متد .IgnoreQueryFilters() استفاده میکنیم.



# TransAction

C10-S23-Transaction

برای مدیریت خطاها زمانی که یک عملیاتی توی دیتابیس انجام میشود ولی در مراحل بعدی با خطا مواجه میشود و یا انجام نمیشود عملیات تعریف کنیم. در این حالت عملیات شکست خورده بازگشته داده میشود.

مثلا یکسری داده داریم که باید به صورت پشت سر هم و در جدولهای متعدد ذرج بشوند. در مرحله اول کار انجام میشود ولی در مرحله دوم با خطا مواجه شده و مرحله سوم و بعد اصلا انجام نخواهد شد. در این حالت داده های ما با مشکل روبرو خواهند شد. برای جلوگیری از این مشکلات از Transaction استفاده میکنیم. در این حالت عملیات انجام شده بازگشت داده شده و یا بی اثر یا خنثی خواهند شد.

با دستور BeginTransaction در ابتدای شروع عملیات و Commit برای پایان عملیات کار را انجام میدهیم.



.....



# درج همزمان داده های مرتبط SaveRelatedData

C10-S24-SavingRelatedData

# EntityState

Scalar Property

پراپرتی هایی که مستقیما به یک ستون از جدول مپ میشوند. فیلد دیتابیس یا درواقع به خود موجدیت برمیگردند یا map میشود. مثل id , title , ….

Navigation property

پراپرتی هایی که برای ارتباط جداول میسازیم مثل:



و دو نوع هستند

Reference Navigation property پراپرتی هایی که از نوع یک entity هستند. مثل پراپرتی discount که بالا آوردیم.

Collection Navigation property پراپرتی هایی که از نوع یک لیست از entity هستند مثل Author\_book

## وضعیت entity ها

هر entity میتواندیکی از پنج وضعیت زیر را داشته باشد.

Added

Modified

Deleted

UnChanged

Detached

# ScalarFunctionMapping

C10-S30-ScalarFunMapping

به وسیله اتریبیوت DbFunction میتوانیم یک اسکالر فانکشن را به یک متد مپ کرد.

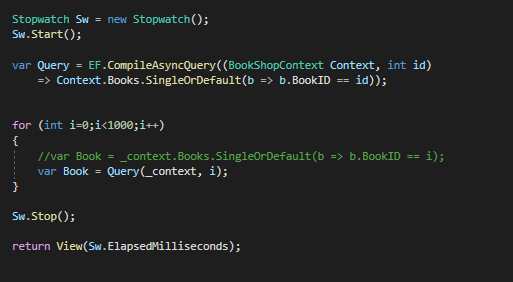
هم نام با فانکشن دیتابیس بهتر است تعریف شوند. و استاتیک هستند. و مقدار آن برابر مقدار برگشتی فانکش دیتابیس است.



## CompiledQueries

C10-S31-CompiledQueries

زمانیکه میخواهیم از یک کوتری چندین بار استفاده کنیم از این قابلیت استفاده میکنیم . باعث میشود که یک کوئری کامپایل شده و در دفعات بعد مستقیما از آن استفاده شود و سرعت برنامه چندین برابر بیشتر شده و در هر بار استفاده دیگه کامپایل نمیشه و از همان کوئری استفاده میشود.



در مثال فوق کوئری 1000 بار اجرا میشود. در حالت کامپایل شده زمان اجرا به نصف کاهش پیدا میکند.

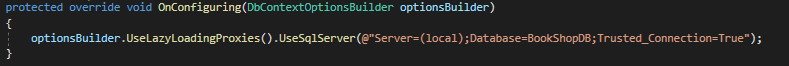
## LazyLoading

C10-S32-LazyLoading

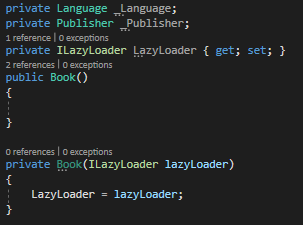
بارگذاری صریح. با لود شدن هر کوئری تمام navigation property ها مقدار دهی میشوند.

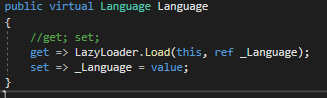
برای فعال شدن این حالت از دو روش : پروکسی ها و یا اینترفیس آن استفاده میکنیم.

برای استفاده از پروکسی ها Install-package Microsoft.EntityFramworkCore.Proxies نصب میکنیم و سپس در کلاس Context در متد Onfiguring از متد UseLazyLoadingProxies() استفاده میکنیم. در این حالت باید همه Navigation property ها را از نوع virtual تعریف کنیم، چون این قابلیت برای همه navigation ها فعال میشود. در غیر این صورت با پیام خطا مواجه خواهیم شد.



برای استفاده از اینترفیس



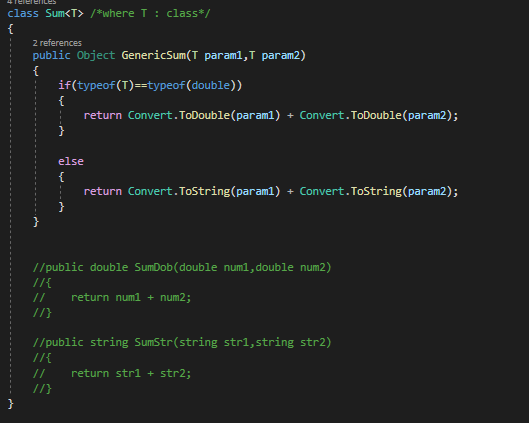


# GenericTypes

C11-S01-GenericTypes

برای جلوگیری از کدهای تکراری، کوتاه شدن برنامه استفاده میشود مثل List<T> که میتواند هر نوع داده ای را قبول کند.

و میتوان از عبارت where برای محدود کردن نوع T استفاده کرد. در این حالت میتوان struct یا کلاس و .... را در مقابل نوع داده ای تعریف نمود

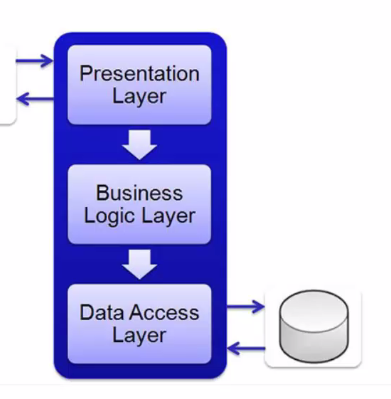


این یک کلاس جنریک است. در این کلاس دو مقدار را با هم جمع میکند. اگر عددی باشد جمع میشود و اگر رشته ای باشد به هم میچسباند.

# الگوی Repository

C11-S02-Repository

در معماری سه لایه از این الگو استفاده میشود.



لایه Present اطلاعات لازم رو به کاربر نمایش میدهد. View

Dataaccess کار با بانک اطلاعاتی )ذخیره و بازیابی و .... ) را بر عهده دارد. Model ها و کلاس Context

Bussiness لایه بین دولایه فوق است، درخواستها را به سمت لایه های دیگر میفرستد تا با بانک اطلاعاتی و کاربر در ارتباط باشد. لایه منتطقی برنامه است. Controllers

این الگو لایه ای بین لایه Bussiness و Data است که دسترسی سایر لایه ها به پایگاه داده پنهان میکنه و دیگه مستقیما با پایگاه داده در ارتباط نیستند. از تکرار شدن کوئری های جلوگیری کرده و قابلیت تست پذیری و کوتاه کردن برنامه رو به ما میدهد.

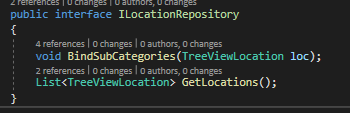
UnitOfWork

IUnitOfWork

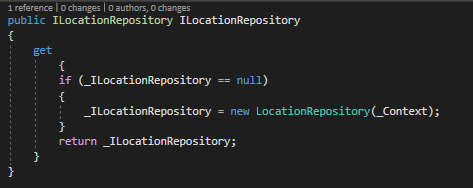
IBaseRepository

BaseRepository

برای اینکه بتوانیم به متد های LocationRepository و ... از طریق Iunitofwork دسترسی داشته باشیم یک اینترفیس تعریف میکنیم به نام ILocationRepository



در کلاس UnitOfWork



برای این منظور مجبور شدم خود کانتکست رو به Repository پاس بدم. باید یک روشی پیدا کنم که از خود uw استفاده کنم.

## RandomID

برای مواردی که در جدول مثل شهر و استان چون Id ها را به صورت دستی وارد کرده بودیم از تابع Random استفاده کردیم.



یک id رندوم ساخته میشود. و چک میکنیم که در جدول وجود دارد یا خیر. اگر موجود بود یکی جدید تولید میکنیم و دوباره چک میکنیم. با حلقه while

هر وقت که به Id جدید رسیدیم به مرحله بعد میره و اون رو توی جدول درج میکنه. و درنهایت id جدید رو به action ایندکس پاس میده.

## ActionName

برا اینکه نام یک متد را تغییر دهیم از این اتریبیوت استفاده میکنیم. مثل delete که در حالت post به آن نام delete را میدهیم. برای بهبود سئو هم خیلی مناسب است.

# RazorPages

در داخل پوشه page ها ساخته میشوند. نام پیش فرض برای اینکه موتور razor بتواند آن را تشخیص، پیدا و اجرا کند و pages است.

برای تغییر نام پوشه پیش فرض Page به هر نام دلخواهی مثلا Mypages باید در startup در سرویس MVC() از متد

AddRazorPagesOptions استفاده کنیم.

.AddRazorPagesOptions(options =>

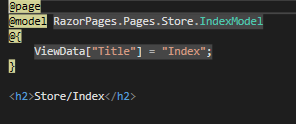
{

options.RootDirectory = "/MyPages";

});

این صفحه ها بسیار سریع هستند و نیازی به کنترلر ندارند و مستقیما اجرا میشوند.

در هنگام آدرس دهی باید کل آدرس مسیر فایل را به view بدهیم و یا اینکه فایلهای \_viewstart , \_viewimport را به پوشه pages کپی کنیم. و در آنجا آدرس کل پروژه را وارد کنیم.



برای layout هم باید پوشه Share را در Pages بسازیم و سپس فایل layout را درون آن قرار دهیم.

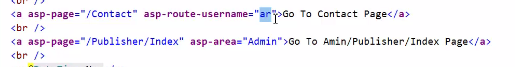
برای RedirectToPage به صفحه دیگر و یا area دیگر به صورت زیر آدرس دهی میکنیم.



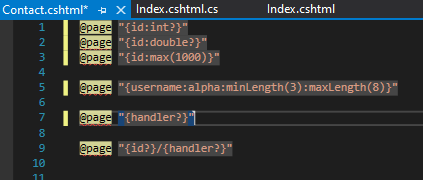
برای رفتن به صفحه دیگر در view از دستور زیر:

<a **asp-page**="/Contact" **asp-route-id**="1">Go To Contact Page</a>

می توانیم در پیج محدود مقادیر ارسال را تعیین کنیم. مثل نوع عددی، رشته ای باشد و ....



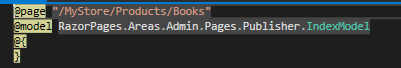
مقادیر ارسال به صفحه contact



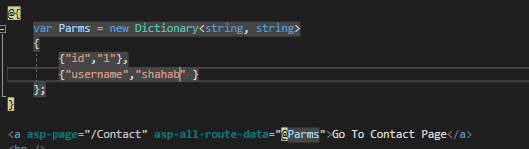
مثلا برای نام کاربری چندین محدودیت اعمال کردیم. الفبایی باشد/ حداقل 3 حرف و حداکثر 8 حرف داشته باشد. یا برای id حداکثر مقدار 1000 در نظر گرفتیم.

Handler? یعنی پارامتر ارسالی به پیچ میتواند optional یا اختیاری باشد.

برای routing یا تغییرات مسیریابی در خود صفحه



مثلا ما میخواهیم دو تا پارامتر به صفحه contact ارسال کنیم.



را وارد میکنیم.

اکنون در صفحه در آدرس بار به صورت

https://localhost:44334/Contact/shahb?id=1

نمایش داده میشود. و نام کاربری و شناسه به آن ارسال شده است. اگر ما بخواهیم مسیر و نمایش را به شکل

https://localhost:44334/Contact?username=shahb&id=1

تغییر بدهیم. باید در page نیز مقدار

@page "{handler?}"

را وارد کنیم. به معنی optional بودن پارامتر ارسالی به razor page است. و نتیجه به صورت فوق نمایش داده خواهد شد.

حالا اگر بخواهیم قسمت آخر آدرس به صورت ?username=shahb&id=1

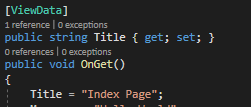
باشد در پیج دستور را به صورت زیر تغییر میدهیم.

@page "{id?}/{handler?}"

یعنی id اختیاری و مقداری هم که بعد از آن قرار میگیرد optional است.

### ViewData

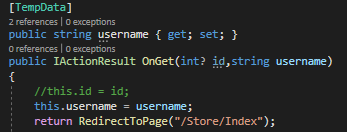
برای انتقال داده به صفحه از ViewData استفاده میکنیم.





### TempData

برای زمانیکه می خواهیم داده ها را به یک action method دیگر یا صفحه دیگر منتقل کنیم.



### Post

حالا اگر بخواهیم داده را از مروگر(سمت کلاینت) بگیریم. باید از متد OnPost یا OnPostAsync استفاده کنیم.

هر پراپرتی که میخواهد داده را از سمت کلاینت بگیرد باید اتریبیوت BindProperty را برایش به کار ببریم.

Bindproperty برای بایندکردن داده ها به متد post استفاده میشود. اگر بخواهیم به متد get هم چیزی ارسال کنیم باید پارامتر

SupportGet = true را به آن ارسال کنیم. در این حالت نیازی نیست که نیست که متد post تعریف کنیم، مثال این موضوع در Paginating یا صفحه بندی Razor Page آمده است.



در این صفحه اعتبار سنجی هم انجام داده ایم.



برای اینکه بتوانیم دو تا متد post داشته باشیم در page از کلمه کلیدی asp-page-handler استفاده کردیم. برنامه در حالت عادی به دنبال OnPost یا OnPostAsync میگردد ولی متد ما متفاوت است. بنابراین قسمت بعد از onpost که LowerCase یا UpperCase است را به asp-page-handler پاس میدهیم. با این کار برنامه متوجه میشود که کدام متد را باید اجرا کند.

### صفحه بندی Razor Page

C12-S06-PagingInRazorPages

به دلیل اینکه در این نوع صفحات از رفلکشن ها نمیتوان استفاده کرد به روش زیر عمل میکنیم.

در کلاس BaseRepository دو تا متد تعریف میکنیم.

public async Task<List<TEntity>> GetPaginateResultAsync(int currentPage,int pageSize=1)

{

var entities = await FindAllAsync();

return entities.Skip((currentPage - 1) \* pageSize).Take(pageSize).ToList();

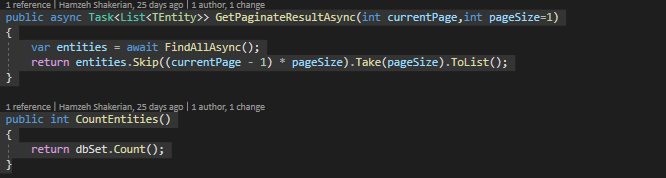
}

public int CountEntities()

{

return dbSet.Count();

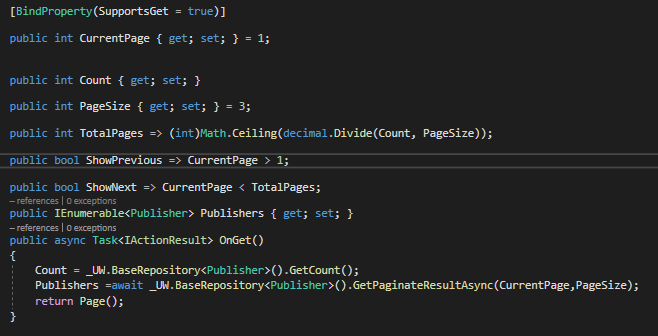
}

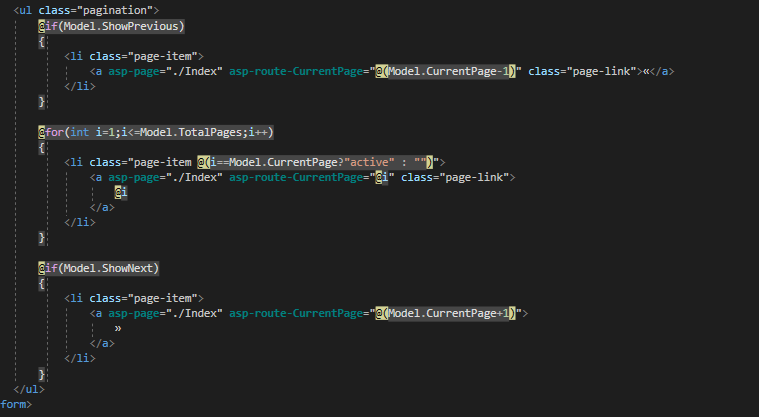


در متد اول شماره صفحه را میگیریم currentpage یکی از آن کم میکنیم و در شماره صفحه pagesize ضرب میکنیم.

چون میخواهیم در صفحه به اندازه pagesize به ما ردیف ها را نمایش دهد و همینطور باید تعداد ردیفهای موجود را هم بگیریم که بدانیم چند ردیف برای نمایش داریم.

در صفحه Razor Page چند پراپرتی تعریف میکنیم و همینطور از متد Bind Property برای ارسال داده به متد get استفاده میکنیم.





چطوری کار میکنه؟

هر بار که کاربر روی علائم صفحه کلیک میکنه متد Onget صدا زده میشه و شماره صفحه به اون ارسال میشه. در واقع پراپرتی CurrentPage مقدار دهی میشود. مقدار 1 رو برایش قرار دادیم تا همیشه به صفحه 1 اشاره کند و صفحه اول رو به نشون بده و کلید زیر صفحه که حالت Active براش گذاشتیم آبی نشون داده بشه.

Count

PageSize چون میخواستیم در هر صفحه 3 تا ردیف رو نشون بده مقدار 3 رو براش پیش فرض قرار دادیم.

TotalPage مقدار تمام صفحات رو میگیره. اگر تعداد کل Entity ها Count رو تقسیم بر pagesize کنیم تعداد صفحه بدست میاد. برای تقسیم از متد Divide استفاده کردیم و چون عدد بدست آمده اعشاری خواهد بود باید رندش کنیم. پس به وسیله تابع Ceiling اون رو گرد می کنیم و حاصل را به مقدار int تبدیل میکنیم.

در متد onget پراپرتی count و publisher رو مقدار دهی کردیم.

ShowPreviouse از عبارت لاندا برای مقدار دهی استفاده کردیم. اگر تعداد CurrentPage بیشتر از 1 بود مقدار true و در غیر اینصورت False میگیرد. به دلیل اینکه میخواهیم بدانیم صفحه های قبلی هم وجود داره یا نه و همینطور برای اینکه به صفحه بعد یا قبل بریم.

# Identity

C13-S01-IntroIdentity

RoleBaseAuthorization در این روش بر اساس نقش احراز هویت انجام میشود. مثلا کاربرانی که نقش ادمین دارند به بخش های مدیریتی دسترسی پیدا میکنند.

ClaimBase احراز هویت بر اساس ادعا، دسترسی کاربر براساس ادعا محدود میشود. Claim ها اطلاعات کاربر هستند که توسط یک منبع معتبر صادر میشوند. هر claim میتواند شامل هر نوع داده ای باشد. مثل نام تاریخ تولد و .... مثلا میتوانیم کاربرانی که تاریخ تولدشون یک تاریخ خاصی هستند به صفحاتی دسترسی داشته باشند و ...

PolicyBase – تعریف سیاست برای احراز هویت. میتوانیم مشخص کنیم که احراز هویت رو براساس نقش / ادعا و یا روش سفارشی خودمون باشد.

### جداول Identity

SecurityStamp مسائل امنیتی جدول کاربران. یک کد رندوم درون این فیلد قرار میگیره و با تغییر رمز کاربر این ستون تغییر میکند.

ConcurrencyStamp برای به روز کردن اطلاعات کاربر. برای ویرایش کردن اطلاعات ردیف کاربر مشابه TimeStamp کار میکند. برای ویرایش داده ها از این ستون استفاده میشود و چک میشود که در زمان ویرایش مقادیر تغییر نکرده باشد. کد همگام سازی و یک کد رندوم که توسط برنامه تولید میشود.

AccessFaildCount تعداد تلاشهای ناموفق کاربر

### جدول Roles

جدول نقشها. مثل ادمین و کاربران و .....

### جدول UserClaims و RoleClaims

اگر بخواهیم از روش Claim Base استفاده کنیم از این جدول و جدول RoleClaims استفاده میکنیم. احراز هویت مبتنی بر ادعا

ClaimType نوع ادعا، تولد ، فامیلی ، اسم و ...

ClailmValue مقدار نوع Claim . مثلا اگر کلیم تایپ از نوع نام باشد مقدار Value میشه حمزه

### جدول UserLogins

این جدول زمانی به کار میاد که به کاربران بخواهیم اجازه بدهیم از طریق حساب کاربری Gmail یا یاهو وارد سایت بشوند.

LoginProvider سرویس هست که کاربر از طریق آن وارد میشود. مثل Facebook یا gmail و ...

ProviderKey شناسه منحصر به فردی که به کاربر اختصاص داده میشود.

DisplayName نام نمایشی سرویسی که کاربر از طریق اون وارد سایت میشود.

### UserToken

احراز هویت بر اساس توکن.

LoginProvider نام سرویسی قرار میگیرد که عمیلات لاگین از طریق آن انجام میشود.

Name Value نام و مقدار توکن را در خود جای میدهد.

## Customized Identity

C13-S02-CustomizeIdentityTable

سفارشی کردن جداول کاربران

## CRUDRoles

C13-S07-CRUDRoles

برای گرفتن / اضافه کردن / حذف نقش ها در دیتا بیس از کلاس RoleManager استفاده میکنیم.

یک کنترلر می سازیم و RoleManager<IdentityRole> را به آن تزریق میکنیم. برای نمایش یک RolesViewModel ساختیم.

# JQuery

C16-S01-Jquery

Function FuncName(){

///

})

## انواع Selector

$(“span”) انتخاب یک تگ

$(“#id”) انتخاب بر اساس شناسه

$(“.test”) انتخاب بر اساس کلاس

## EventMethods

### اجرا شدن با لود شدن صفحه

$(document).ready( function (){

});

### اجرا شدن با کلیک کردن

$(“p”).click( function (){

/// action goes here !

);

گرفتن محتوا Content

### به وسیله Id

<p id=”test” <span> hmzh.ir </span></p>

< id=”btn” button>

ما میخواهیم زمانی که روی کلید کلیک شد متن داخل تگ span را بگیرم.

$(”#btn”).click(function(){

alert(“Text:” + $(“#test”).text())

});

نتیجه: Text : hmzeh.ir

با کلیک رو باتن تابع کلیک فراخوانی میشود و تابع متن داخل تگ p که آدرس سایت هست به وسیله id رو میگیره و با دستور alert به کاربر نمایش میدهد.

### به وسیله html

و اگر بخواهیم محتوای html اون تگ رو بگیریم از روش زیر استفاده میکنیم.

$(”#btn”).click(function(){

alert(“HTML :” + $(“#test”).html())

});

نتیجه: HTML : ” <span> hmzh.ir </span>

### گرفتن value

فرض میگیریم یک تگ input داریم که مقدار "Ali" را در آن نوشته ایم.

$(”#btn”).click(function(){

alert(“value :” + $(“#test”).val())

});

نتیجه : value : Ali

## نوشته مقادیر در تگها

حالا اگر بخواهیم یک محتوا را داخل یک تگ قرار بدهیم

<p id=”test” <span> hmzh.ir </span></p>

تغییر hmzh.ir به mirsoltan پس داخل تگ span تغییر میکند.

$(”#btn”).click(function(){

$(“#test”).text(“ Mirsoltan ”);

});

تغییر تگ html فوق از span به b

$(”#btn”).click(function(){

$(“#test”).html(“<b> Mirsoltan </b>”);

});

برای نوشته در تگ input

$(”#btn”).click(function(){

$(“#inp”).val(“ Mirsoltan ”);

});

مخفی کردن یا نمایش یک تگ

$(”#btn”).click(function(){

$(“#inp”).hide();

});

$(”#btn”).click(function(){

$(“#inp”).show();

});

### اضافه / حذف کردن کلاس به تگ

$(“inp”).addClass(“”);

$(“inp”).removecClass(“”);

# Ajax

## get

## ارسال درخواست

$.get(URL,data,function(data,status,xhr),dataType)

URL اجباری هست. مشخص میکنه ما میخواهیم درخواست رو به چه آدرسی ارسال کنیم

Data اطلاعاتی که به همراه درخواست به سمت سرور ارسال میشود.

Function تابعی هست که زمان ارسال درخواست اجرا میشود.

data در برگیرنده اطلاعات خروجی درخواست

Status وضعیت درخواست success , error , time out , ….

Xhr •xhr - contains the XMLHttpRequest object

datatype اختیاری هست . نوع داده ارسالی از سمت سرور را مشخص میکند. که میتواند از نوع

مثال: یک درخواست به index ارسال میکند و در نهایت دیتایی که از سمت سرور دریافت میشود را در قالب alert نمایش میدهد.

$("button").click(function(){

$.get(“/Home/Index", function(data , status){

alert("Data: " + data + "\nStatus: " + status);

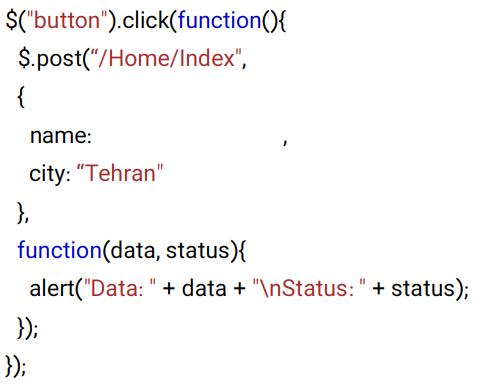
});

});

## Post

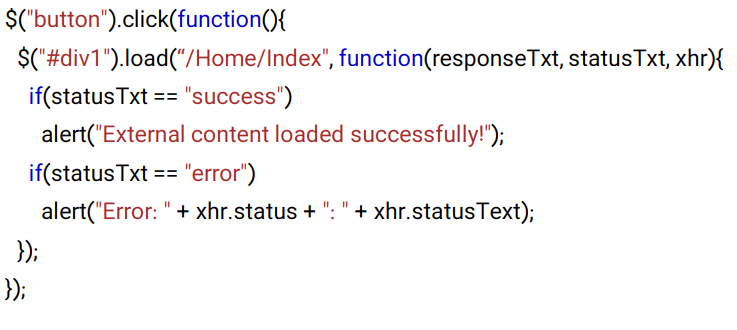
Post هم دقیقا مانند get هست.

$(selector).post(URL,data,function(data,status,xhr),dataType)



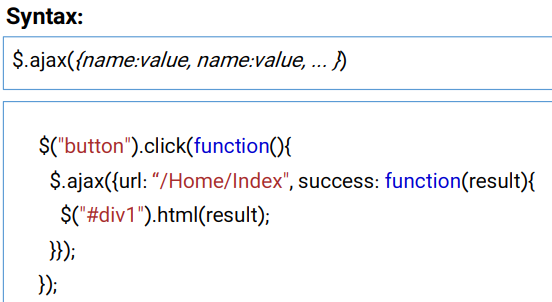
## Load

$(selector).load(url,data,function(response,status,xhr))



متد ajax

مهمترین متد برای ارسال درخواست



پارامترها:

async به صورت پیش فرض true هست

Cache برای کش شدن یا نشدن صفحه در خواست مقدار پیش فرض true

contentType نوع محتوای داده ارسالی به سمت سرور را مشخص میکند.

Complete بعد از تمام شدن درخواست اجرا میشود.

Data داده های ارسالی به سمت سرور

Error در زمان مواجه شدن با خطا ارجا میشود.

Type نوع get یا post را تعیین میکند.

url آدرس درخواست

1. https://virgool.io/@alimfazeli/%D8%B9%D9%84%D8%AA-%D9%88%D8%AC%D9%88%D8%AF-cookie-%D9%87%D8%A7%DB%8C-httponly-%D9%88-%DA%A9%D8%A7%D8%B1%D8%A8%D8%B1%D8%AF-%D8%A7%D9%88%D9%86-%D9%87%D8%A7-hszfkm5eqfsr [↑](#footnote-ref-1)
2. https://www.dotnettips.info/post/3031/%D8%A8%D8%B1%D8%B1%D8%B3%DB%8C-%D9%82%D8%A7%D8%A8%D9%84%DB%8C%D8%AA-endpoint-routing-%D8%AF%D8%B1-asp-net-core [↑](#footnote-ref-2)