

# Leksion 2 MVC

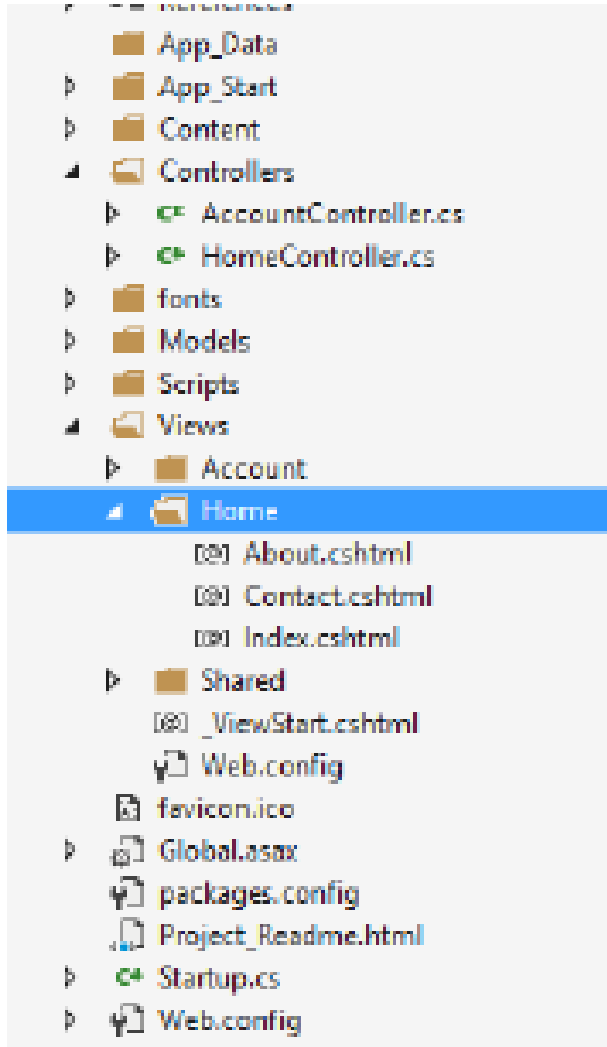
1. Vazhdim View
2. Modelet

# Konvencionet e View

---

- ▶ Në leksionin e kaluar u ilustruan disa shembuj se ti të përdoren View për të renderuar HTML. Megjithatë ka disa konvencione të cilat zbatohen në mënyrë që ASP.NET MVC të gjejë View e duhur për të shfaqur.
- ▶ Metodat e kontrolleve të para deri tani thjesht kthenin View() dhe nuk specifikonin emrin e saj. Kjo sepse shfrytëzohen avantazhet që vijnë nga konvencionet ASP.NET MVC Framework, të cilat përcaktojnë llogjikën e selektimit të View.
- ▶ Gjatë krijimit të një projekti të ri, projekti do të përmbante direktorinë View të strukturuar në një mënyrë të caktuar.

# Konvencionet e View



- ▶ Cdo direktori controller përmban një file view për cdo metode action dhe emri i file emertohet sipas metodës action, kështu view shoqërohen me metodën action përkatëse.
- ▶ Pra llogjikuesi që selekton view kërkon për një View me të njëjtin emër me action e një controller brenda folder /Views/.
- ▶ Kjo siguron bazën për mënyrën sesi lidhet një View me metodën action.
- ▶ Llogjika selektuese e View kërkon për një View me të njëjtin emër si action në direktorinë View/Emri i Kontrollerit/.

# Përtej ViewBag

- ▶ Kur të dhënat që do të kalohen në View nga kontrolleri janë të vogla atëherë mund të përdoret ViewBag. Megjithatë ViewBag nuk mund të përdoret në raste më komplekse kur përfshihen të dhëna reale.
- ▶ Supozojmë që duam të afishojmë në një View listën e instancave të klasës Album. Një mënyrë është të shtohen albumet në një ViewBag dhe të bridhen brenda View. Në kontrollerin përkatës do të vendosnit metodën List si më poshtë:

```
▶ public ActionResult List()  
▶ {  
▶     var albums = new List<Album>();  
▶     for(int i = 0; i < 10; i++) {  
▶         albums.Add(new Album {Title = "Product " + i});  
▶     }  
▶     ViewBag.Albums = albums;  
▶     return View();  
▶ }
```

# Përtej ViewBag

---

Në view mund të bridhen objektet e listës së Albumeve nëpërmjet:

- ▶ `<ul>`
- ▶ `@foreach (Album a in (ViewBag.Albums as IEnumerable<Album>))`  
`{`
- ▶ `<li>@a.Title</li>`
- ▶ `}`
- ▶ `</ul>`

Në këtë rast përdoret `IEnumerable<Album>` për të konvertuar në këtë tip Albumet që përmban ViewBag. `IEnumerable` përdoret në rastet kur bredhim objekte nga një koleksion dhe nuk jemi të bindur mbi strukturën e brendshme që kanë këto objekte, por edhe në rastet kur përmbajnë attribute me tipe të ndryshme(string, double etj).

# Përtej ViewBag

---

Një strongly typed view është e dizenuar që të punojë me një objekt nga modeli juaj, quhet ndryshe dhe view model.

Ju lejon që të kaloni objektin e modelit nga kontrolleri drejt view. Në metodën e Controller ju specifikoni modelin nëpërmjet mbivendosjes së metodës së View:

```
▶ public ActionResult List()  
▶ {  
▶ var albums = new List<Album>();  
▶ for (int i = 0; i < 10; i++)  
▶ {  
▶ albums.Add(new Album {Title = "Album " + i});  
▶ }  
▶ return View(albums);  
▶ }
```

---

# Përtej ViewBag

---

Hapi i rradhës është të përcaktohet tek View cili model do të përdoret duke vendosur deklarinin e modelit nëpërmjet @model.

- ▶ **@model**  
***IEnumerable<MvcMusicStore.Models.Album>***
- ▶ ***<ul>***
- ▶ ***@foreach (Album p in Model) {***
- ▶ ***<li>@p.Title</li>***
- ▶ ***}***
- ▶ ***</ul>***

# Përtej ViewBag

---

Nëse nuk dëshironi të specifikoni emrin e modelit atëherë mund të përdoret `@using` duke përfshirë kështu gjithë modelet e projektit si më poshtë:

- ▶ **`@using MvcMusicStore.Models`**
- ▶ `@model IEnumerable<Album>`
- ▶ `<ul>`
- ▶ `@foreach (Album p in Model) {`
- ▶ `<li>@p.Title</li>`
- ▶ `}`
- ▶ `</ul>`



# Përfshirja e modeleve

Një qasje akoma dhe më e mirë për namespace që do të përdoren në view është të deklarohen në web.config brenda direktorisë së View :

```
<system.web.webPages.razor>
```

- ▶ ...
- ▶ `<pages pageBaseType="System.Web.Mvc.WebViewPage">`
- ▶ `<namespaces>`
- ▶ `<add namespace="System.Web.Mvc" />`
- ▶ `<add namespace="System.Web.Mvc.Ajax" />`
- ▶ `<add namespace="System.Web.Mvc.Html" />`
- ▶ `<add namespace="System.Web.Routing" />`
- ▶ **`<add namespace="MvcMusicStore.Models" />`**
- ▶ `</namespaces>`
- ▶ `</pages>`
- ▶ `</system.web.webPages.razor>`

# Të kuptojmë ViewBag, ViewData, dhe ViewDataDictionary

---

- ▶ ViewBag ishte një mekanizëm që mundësonte kalimin e informacionit nga controller drejt View. Në realitet, si ViewBag si strongly typed model, të dyja kalohen nëpërmjet ViewDataDictionary.
- ▶ Teknikisht të gjitha të dhënat kalohen nga kontroller drejt View nëpërmjet ViewDataDictionary, një klasë e specializuar e quajtur ViewData. Ju mund të caktoni dhe të lexoni vlera nga një fjalor ViewData duke përdorur sintaksën:
- ▶ *`ViewData["CurrentTime"] = DateTime.Now;`*
- ▶ Ose
- ▶ *`ViewBag.CurrentTime = DateTime.Now;`*

# ViewBag dhe ViewData

---

ViewBag funksionon vetëm kur ju aksesoni një çelës identifikues valid në C#. Për shembull nëse e vendosni çelësin me hapsira, nuk mund të aksesohet vlera duke përdorur ViewBag sepse nuk do të kompilohej kodi. `ViewData["Key With Spaces"]`

ViewData dhe ViewBag kërkojnë që çelësi i aksesimit të ekzistojë, nëse ju gjatë vlerëdhënies ndryshoni çelësin e aksesit nga `CurrentTime` në `CurrentT`

► *`ViewData["CurrentT"] = DateTime.Now;`*

atëherë ky ndryshim duhet të pasqyrohet edhe në View

Tek View duhet që të aksesoni me të njëjtin çelës kjo i bën jo shumë fleksibël këto struktura, e njëta gjë do të vlente edhe për ViewBag. Për këtë në të ardhmen do të përdorim ViewModel.



# ViewModel

Në shumicën e rasteve rekomandohet që të ndërtohet një klasë model për view. Kjo do të quhej një view model. Mendojeni View model si një model që ekziston vetëm për ti dhënë informacion një view.

Për shembull, nëse keni një faqe për totalin e kartës së blerjes, ku ju duhet të shfaqni një listë me produkte dhe totalin e kostos për kartën, dhe gjithashtu një mesazh përdoruesit mund të krijohet një klasë model `ShoppingCartViewModel`

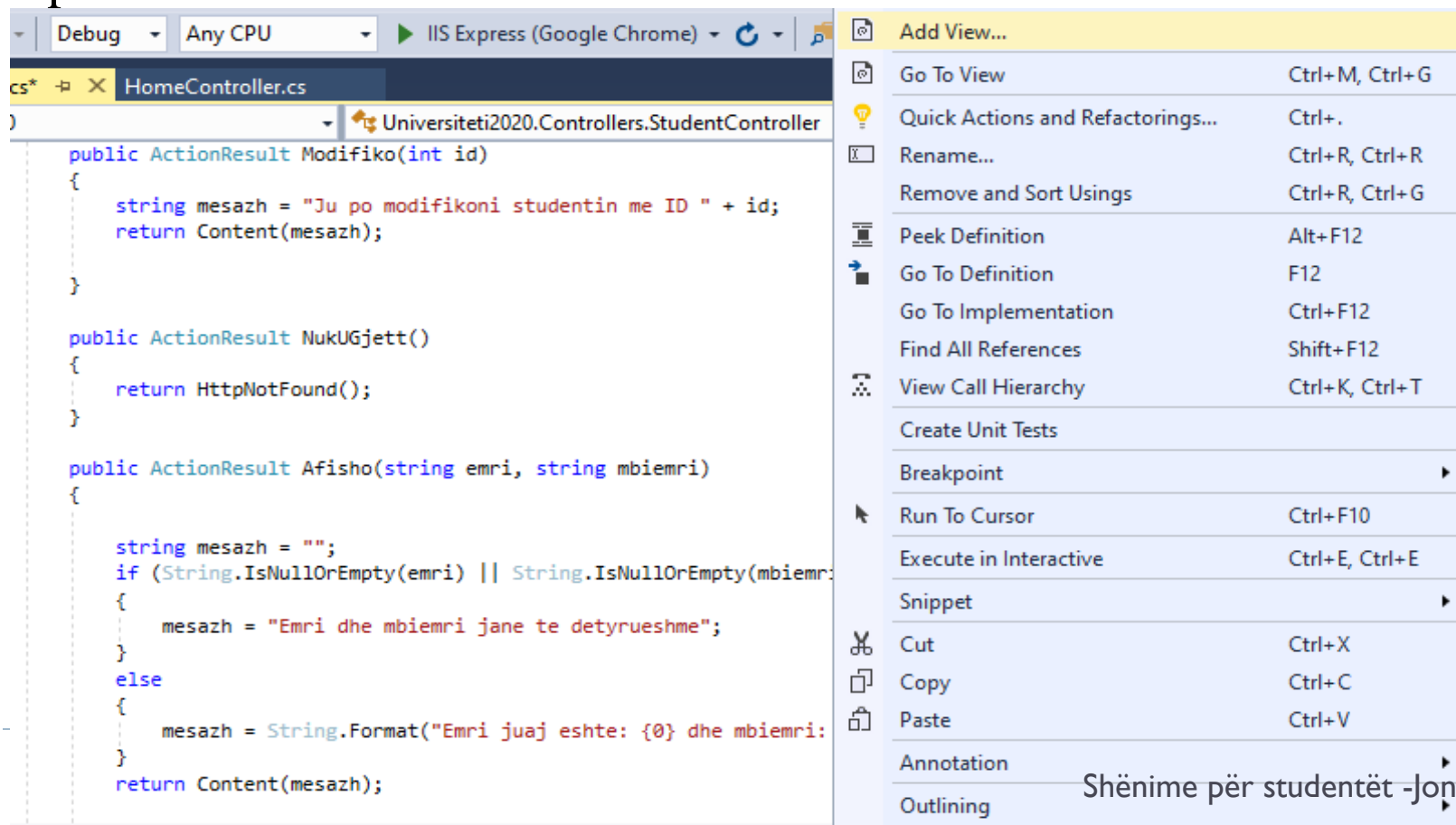
```
▶ public class ShoppingCartViewModel {  
▶ public IEnumerable<Product> Products { get; set; }  
▶ public decimal CartTotal { get; set; }  
▶ public string Message { get; set; }  
▶ }
```

Pwrfsirja e modelit nw View nwpwrmjet:

```
▶ @model ShoppingCartViewModel
```

# Shtimi i një View

Deri tani pamë View të gjeneruara në mënyrë automatike për Home dhe Account, por nëse dëshirojmë të shtojmë një View të re, atëherë mënyra më e thjeshtë do të ishte të shkonim tek action përkatës në Controller të klikonim me të djathtën dhe të për zgjidhnim Add View... Si në Figurën më poshtë:



The screenshot shows the Visual Studio IDE with the 'Add View...' menu option highlighted in the context menu of a Controller action. The code editor displays the following C# code for the 'Modifiko' action in 'Universiteti2020.Controllers.StudentController':

```
public ActionResult Modifiko(int id)
{
    string mesazh = "Ju po modifikoni studentin me ID " + id;
    return Content(mesazh);
}

public ActionResult NukUGjett()
{
    return HttpNotFound();
}

public ActionResult Afisho(string emri, string mbiemri)
{
    string mesazh = "";
    if (String.IsNullOrEmpty(emri) || String.IsNullOrEmpty(mbiemri))
    {
        mesazh = "Emri dhe mbiemri jane te detyrueshme";
    }
    else
    {
        mesazh = String.Format("Emri juaj eshte: {0} dhe mbiemri: {1}", emri, mbiemri);
    }
    return Content(mesazh);
}
```

The context menu on the right lists the following options:

- Add View...
- Go To View (Ctrl+M, Ctrl+G)
- Quick Actions and Refactorings... (Ctrl+.)
- Rename... (Ctrl+R, Ctrl+R)
- Remove and Sort Usings (Ctrl+R, Ctrl+G)
- Peek Definition (Alt+F12)
- Go To Definition (F12)
- Go To Implementation (Ctrl+F12)
- Find All References (Shift+F12)
- View Call Hierarchy (Ctrl+K, Ctrl+T)
- Create Unit Tests
- Breakpoint
- Run To Cursor (Ctrl+F10)
- Execute in Interactive (Ctrl+E, Ctrl+E)
- Snippet
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Annotation
- Outlining

# Shtimi i një View

Add View ×

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor \_viewstart file)

- ➤ **View name:** Duke qenë se krijimi i View u nis duke klikuar me të djathtën tek metoda action atëherë emri i View do të jetë i mbushur automatikisht me emrin e metodës action që për ne është Afisho.
- ➤ **Template:** Mund të përzgjidhet një template i cili gjeneron një view sipas një tip modeli të përgjedhur.

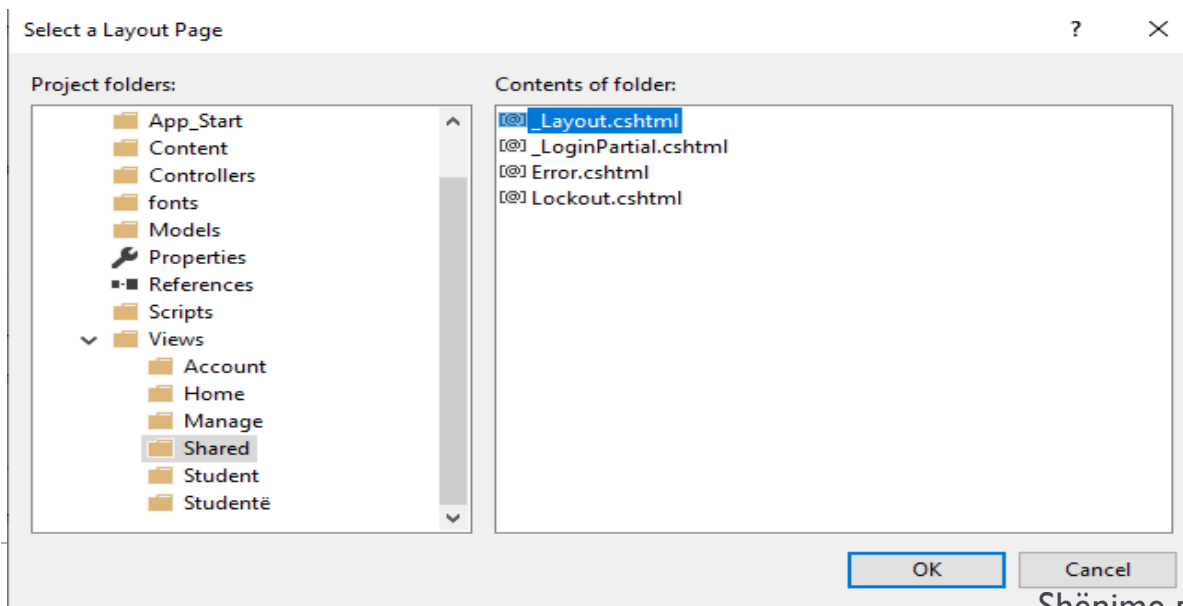
# Shtimi i një View

---

- ▶ **Reference script libraries:** Ky opsion nëse është i selektuar mundëson që view që po krijojmë të ketë referenca të një set të fileve Javascript. Default file `_Layout.cshtml` ka referenca për librarinë kryesore të jQuery por nuk ka rereferencë për librarinë e validimit të jQuery dhe në rastet kur nevojitet validimi sic janë view që mund të përmbajnë editim ose shtim rekordesh duke selektuar këtë opsion shtohen referencat për libraritë përgjegjëse për validimin.
- ▶ **Create as a partial view:** Kur përzgjidhet ky opsion view që po krijohet do të jetë e pjesshme për këtë edhe opsioni Layout është disable. Pamja e pjesshme duket si pamje e zakonshme vetëm se nuk ka tagun `<html>` ose `<head>` në fillim të pamjes. Zakonisht përdoret në rastet kur ajo pamje do të ripërdoret në disa skenarë.

# Shtimi i një View

- **Use a layout page:** Ky opsion përcakton nëse pamja që po krijohet lidhet me ndonjë dizajn të caktuar, në përgjithësi dëshirohet që aplikacioni të ketë pak a shumë të njëjtën pamje (Rikujto Master Pages), ne mund të përzgjedhim view default të specifikuar si me poshte





# Sintaksa Razor

- ▶ Prezantohet me ASP.NET MVC 3, i dha zgjidhje problemeve të pamjeve në web forms.
- ▶ Lehtësohet mirëmbajtja e kodit.
- ▶ `@{`
- ▶ `var items = new string[] { "one", "two", "three" };`
- ▶ `}`
- ▶ `<html>`
- ▶ `<head><title>Sample View</title></head>`
- ▶ `<body>`
- ▶ `<h1>Listing @items.Length items.</h1>`
- ▶ `<ul>`
- ▶ `@foreach(var item in items) {`
- ▶ `<li>The item name is @item.</li>`
- ▶ `}`
- ▶ `</ul>`
- ▶ `</body>`
- ▶ `</html>`

# Shprehje të kodit

---

Celësi i kalimit në sintaksën Razor është simboli @. Ky simbol përdoret për të kaluar nga kod markup dhe anasjelltas. Dy tipet kryesore të kalimit janë shprehjet e kodit dhe blloqet e kodit. Shprehjet vlerësohen dhe shkruhen në përgjigje.

Shembull:

▶ `<h />Listing @items.Length items.</h />`

Në shprehjen e mësipërmje `@items.Length` vlerësohet si shprehje ku dhe rezultati 3 do të shfaqej si output. Megjithatë siç vihet re në këtë rresht nuk është e nevojshme që të mbyllet me simbol @ siç ndodhte në Web forms ku duhej të mbyllej `<%: %>`

# Dykuptimësia në Razor

---

## Shembull

Në kodin e mëposhtëm ne duam që të na afishohet si string *MyApp.Models*

- ▶ `@{`
- ▶ `string rootNamespace = "MyApp";`
- ▶ `}`
- ▶ `<span>@rootNamespace.Models</span>`

Në fakt kodi i mësipërm do të gjeneronte një gabim që nuk ka atribut Model të stringut. Në këtë rast Razor nuk do e kuptonte këtë situatë, zgjidhja do të paraqitej si më poshtë:

`<span>@(rootNamespace).Models</span>`

Duke vendosur kllapa midis variablit dinamik arrihet të diferencohet nga pjesa tjetër statike e kodit.

---

# Dykuptimësia në Razor

---

Një tjetër rast që në të cilin mund të mendonim për dykuptimësinë do të ishte rasti i një adrese email. Shembull

▶ `<span>support@megacorp.com</span>`

Në pamje të parë do të mendonim që kjo shprehje do të shkaktonte gabim por në fakt nuk ndodh kështu sepse Razor mund të identifikojë modelin e një adrese email dhe do ta shfaqë këtë shprehje siç është.

Po në rastin kur ne kemi një shembull si më poshtë, ku `@item.Length` duam të trajtohet si variabël dinamik

▶ `<li>Item_@item.Length</li>`

Në këtë rast Razor do ta shfaqte siç është dhe jo sipas rezultatit `Item_3`, për të realizuar këtë mjafton të përdorim sërish kllapat ( )

`<li>Item_@(item.Length)</li>`

---

# Dykuptimësia në Razor

---

Në situatat kur stringu që ju dëshironi të afishoni përmban simbolin @ atëherë mund ta realizoni afishimin e tij duke vendsur dy simbole @@ si në shembullin më poshtë:

- ▶ `<p>`
- ▶ *You should folloë*
- ▶ `@@aspnet`
- ▶ `</p>`

Nëpërmjet sekuencave të @@ dhe () është e mundur që të kryhen afishimet e dëshiruara pa krijuar dykuptimësi në sintaksën Razor.

# HTML Encoding

---

Në situatat e ndërveprimit të përdoruesit me aplikacionin si shtimi i informacionit të regjistrimit në një formë ose postimi i një komenti, aplikacioni mund të rrezikohet nga sulmet cross-site script injection. Megjithatë në Razor nuk ekziston ky problem sepse shprehjet enkodohen në HTML automatikisht. Vini re kodin më poshtë:

- ▶ @{
- ▶ *string message = "<script>alert('haacked!');</script>";*
- ▶ }
- ▶ *<span>@message</span>*

Ky kod nuk do të afishonte mesazhin alert por do të enkodohej në HTML si më poshtë:

- ▶ *<span>&lt;script&gt;alert(&#39;haacked!&#39;);&lt;/script&gt;</span>*



# HTML Encoding

---

Megjithatë në rastet kur ju keni si qëllim që të afishoni tage HTML mund të ktheni një instancë të `System.Web.IHtmlString` dhe Razor nuk do ta enkodojë. Ose mund të krijoni një instancë të `HtmlString` ose përdorni `Html.Raw` :

- ▶ `@{`
- ▶ `string message = "<strong>This is bold!</strong>";`
- ▶ `}`
- ▶ `<span>@Html.Raw(message)</span>`

Mesazhi do të afishohet sic është pa enkodim HTML  
`<span><strong>This is bold!</strong></span>`

# Mbrojtja nëpërmjet enkodimit JavaScript

---

Kur vendosen variabla në Javascript me vlera që sigurohen nga përdoruesi përdorimi i enkodimit në Javascript përtej atij në HTML është thelbësor. Për të realizuar enkodimin e inputit do të përdorej @Ajax.JavaScriptStringEncode

- ▶ `<script type="text/javascript">`
- ▶ `$(function () {`
- ▶ `var message = 'Hello`  
`@Ajax.JavaScriptStringEncode(ViewBag.Username)';`
- ▶ `$("#message").html(message).show('slow');`
- ▶ `});`
- ▶ `</script>`



# Blloqet e kodit

---

Përtej shprehjeve të kodit, Razor do të suportonte edhe blloqet e kodit në një pamje.

```
@foreach(var item in stuff) {  
▶ <li>The item name is @item.</li>  
▶ }
```

Shpesh mendohet se kalimi nga kodi foreach drejt tagut <li> bëhet për shkak të karakterit rresht i ri (neë line), por në fakt nuk ndodh për këtë shkak sepse edhe po të shkruhej:

```
▶ @foreach(var item in stuff) {<li>The item name is @item.</li>}
```

Efekti do të ishte i njëjtë. Meqënëse Razor kupton strukturën e HTML ai e kryen kalimin në mënyrë automatike sërish tek kodi kur mbyllet tagu </li>

# Blloqet e kodit

---

Blloqet e kodit kanë nevojë për kllapa { } për të ndarë bllokun nga një simbol @,

Shembull:

- ▶ @{
- ▶ *string s = "One line of code.";*
- ▶ *ViewBag.Title "Another line of code";*
- ▶ }

Edhe kur thërrihen metoda që nuk kthyejnë vlerë duhet të vendosen kllapat sërish:

- ▶ @{Html.RenderPartial("SomePartial");}

# Shembuj të sintaksës Razor

---

Mënyra tipike e shfaqjes së një vlere nga një shprehje kodit në View

- ▶ `<span>@model.Message</span>`

Shprehjet e kodit vlerësohen dhe shkruhen në përgjigje vlera 3 do të vlerësohej si më poshtë:

- ▶ `<span>1 + 2 = @(1 + 2)</span>`

Blloqet e kodit në përgjithësi të dobishme në deklarin e variablave që do të përdoren më vonë:

- ▶ `@{`

- ▶ `int x = 123;`

- ▶ `string y = "because.";`

- ▶ `}`

# Miksimi i tekstit me kod

---

- ▶ `@if (showMessage) {`
- ▶ `<text>This is plain text</text>`
- ▶ `}`
- ▶ `ose`
- ▶ `@if (showMessage) { @:This is plain text.`
- ▶ `}`

Në të dyja rastet nëse shprehja if vlerësohet në true do të afishohet mesazhi, rasti i parë Razor kupton taget përkatëse dhe nuk i afishon ato, rasti i dytë sintaksë e veçantë për të kaluar nga kod në tekst. Funksionon vetëm për një rresht në një kohë.

# Komentet server side në Razor

---

Kodi më poshtë do të ishte një koment dhe do të injorohej.

- ▶ `@*`
- ▶ *This is a multiline server side comment.*
- ▶ `@if (showMessage) {`
- ▶ `<h1>@ViewBag.Message</h1>`
- ▶ `}`
- ▶ *All of this is commented out.*
- ▶ `*@`

# Layout

---

Layouts në Razor ndihmojnë në përcaktimin e një pamjeje përgjatë shumë view, ato krijojnë të njëjtin efekt si master pages në Web forms vetëm se ofrojnë sintaksë më të thjeshtë dhe më fleksibël.

Ju mund të përdorni layout për të përcaktuar një template për gjithë aplikacionin ose edhe për një pjesë të tij. Ky template përmban një ose më shumë placeholders që view në aplikacionin tuaj sigurojnë përmbajtje për to .

- ▶ `<!DOCTYPE html>`
- ▶ `<html>`
- ▶ `<head><title>@ViewBag.Title</title></head>`
- ▶ `<body>`
- ▶ `<h1>@ViewBag.Title</h1>`
- ▶ `<div id="main-content">@RenderBody()</div>`
- ▶ `</body>`
- ▶ `</html>`

# Shembull

## Ndryshimi i stilizimeve të aplikacionit

---

Për të ndryshuar stilet në aplikacion duhet të modifikojmë file template të bootstrap.css i cili ndodhet në librarinë Content/bootstrap.css ky file do të shfaqte stilin default që ne shohim kur i bëjmë run aplikacionit si në figurën më poshtë:

Application name   Home   About   Contact

Register   Log in

# ASP.NET

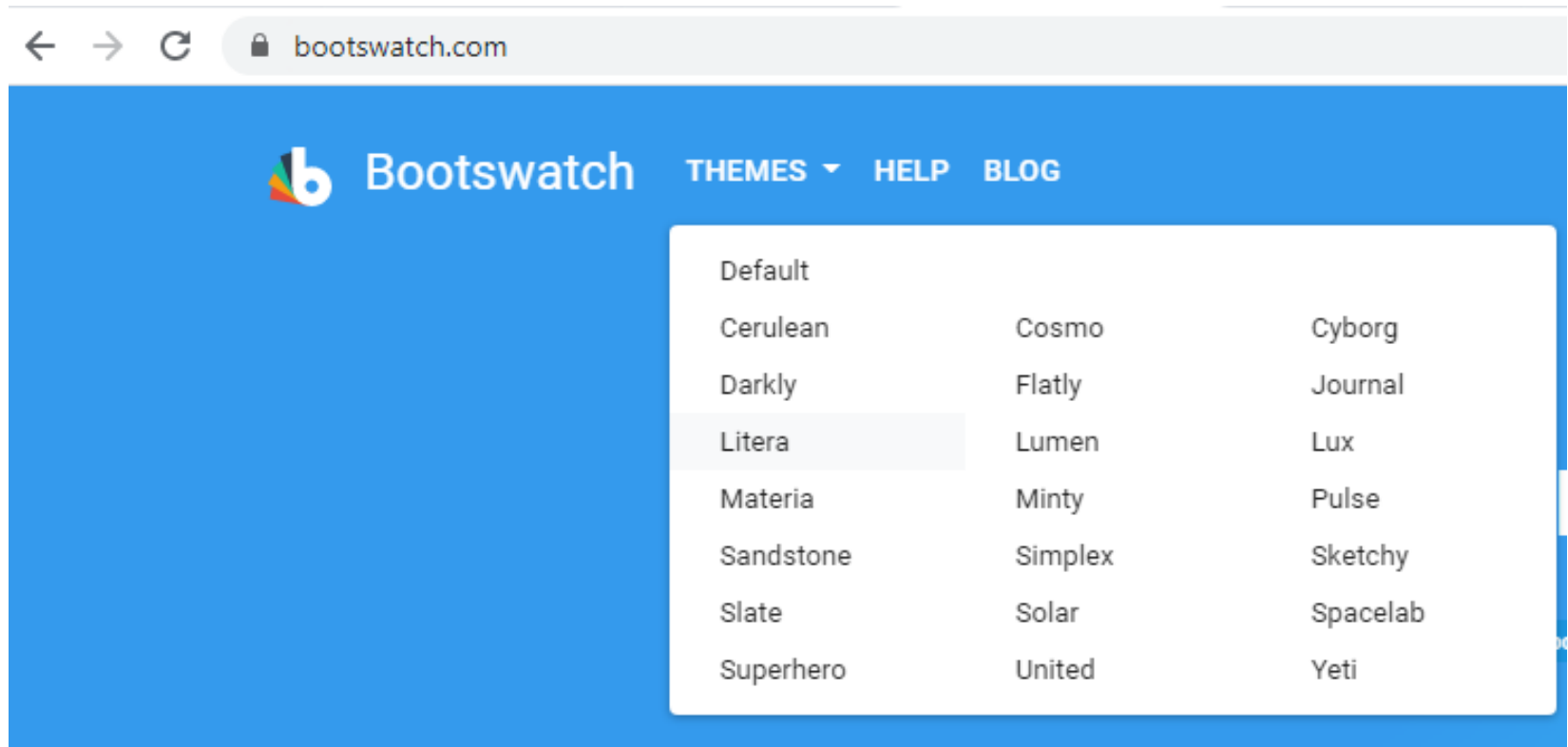
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

[Learn more »](#)

# Shembull

## Ndryshimi i stilizimeve të aplikacionit

- Stile të tjera mund të gjeni në website,  
<https://bootswatch.com>, për zgjedhim një tërë prej tyre

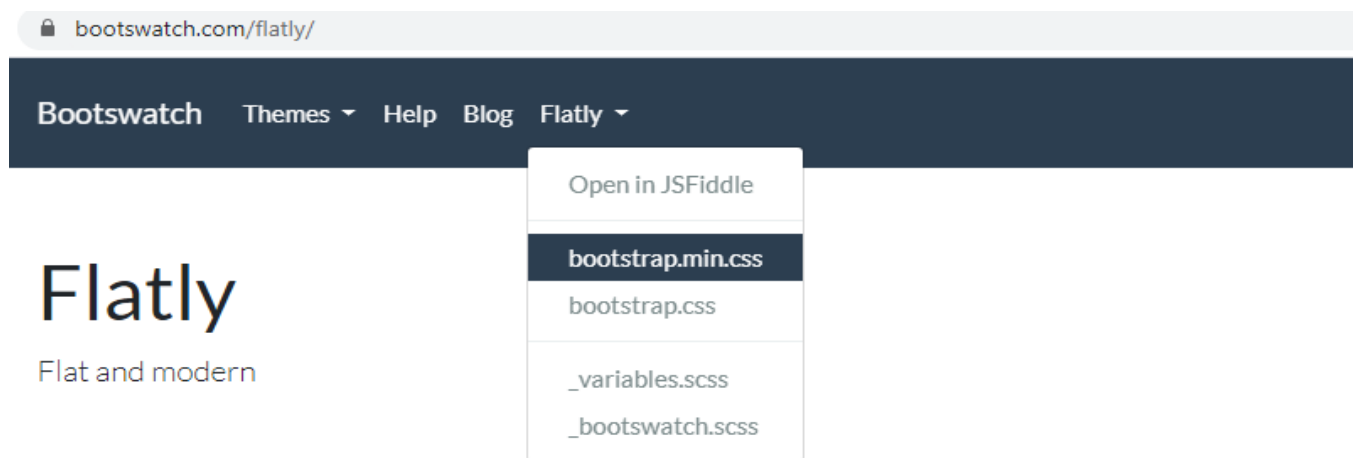




# Shembull

## Ndryshimi i stilizimeve të aplikacionit

- Klikoni mbi emrin e stilit të përzgjedhur dhe shkoni tek bootstrap.min.cs klikoni me të djathtën dhe hapeni në një tab të ri file.



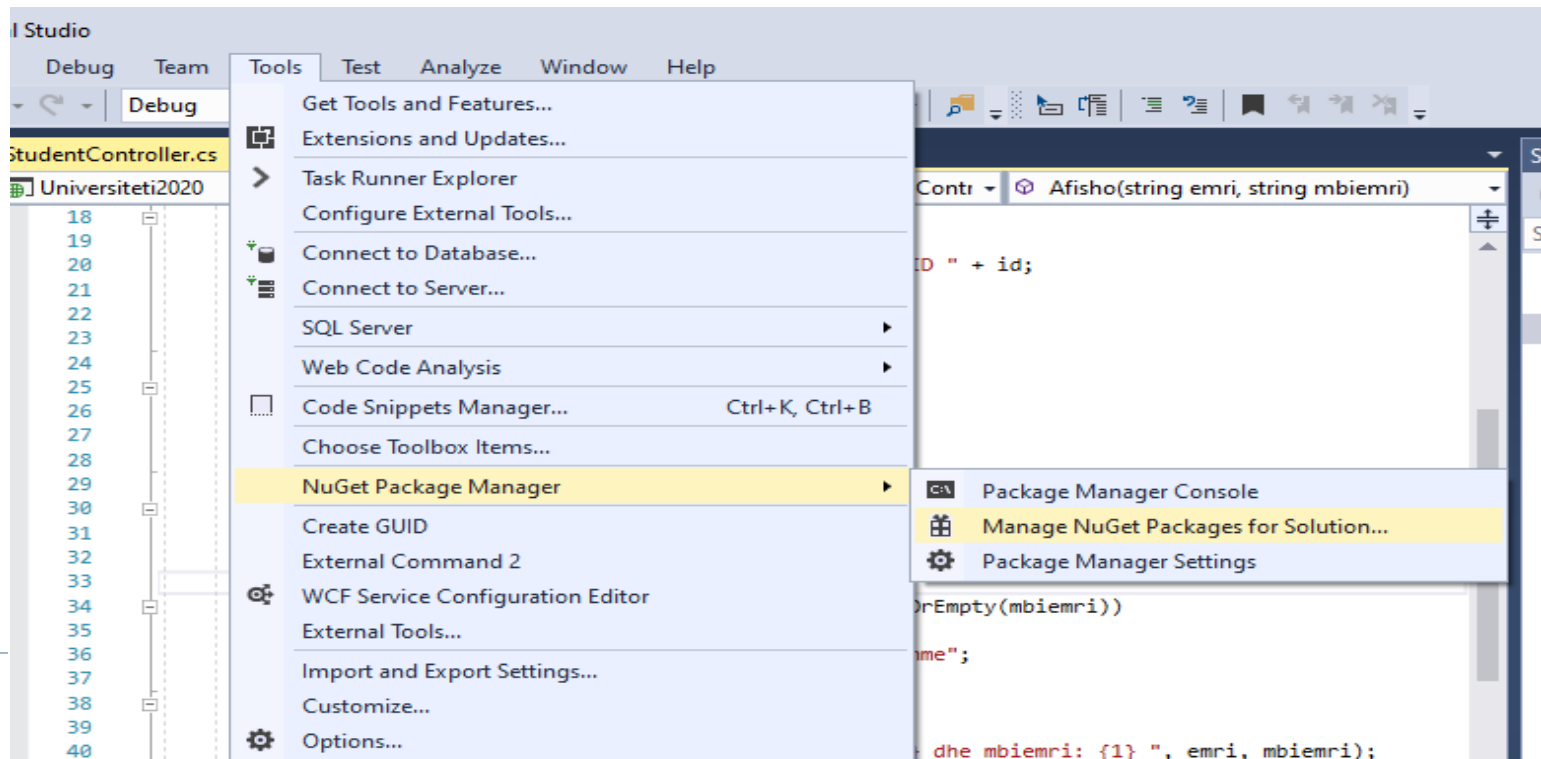
## Navbars



# Probleme të mundshme

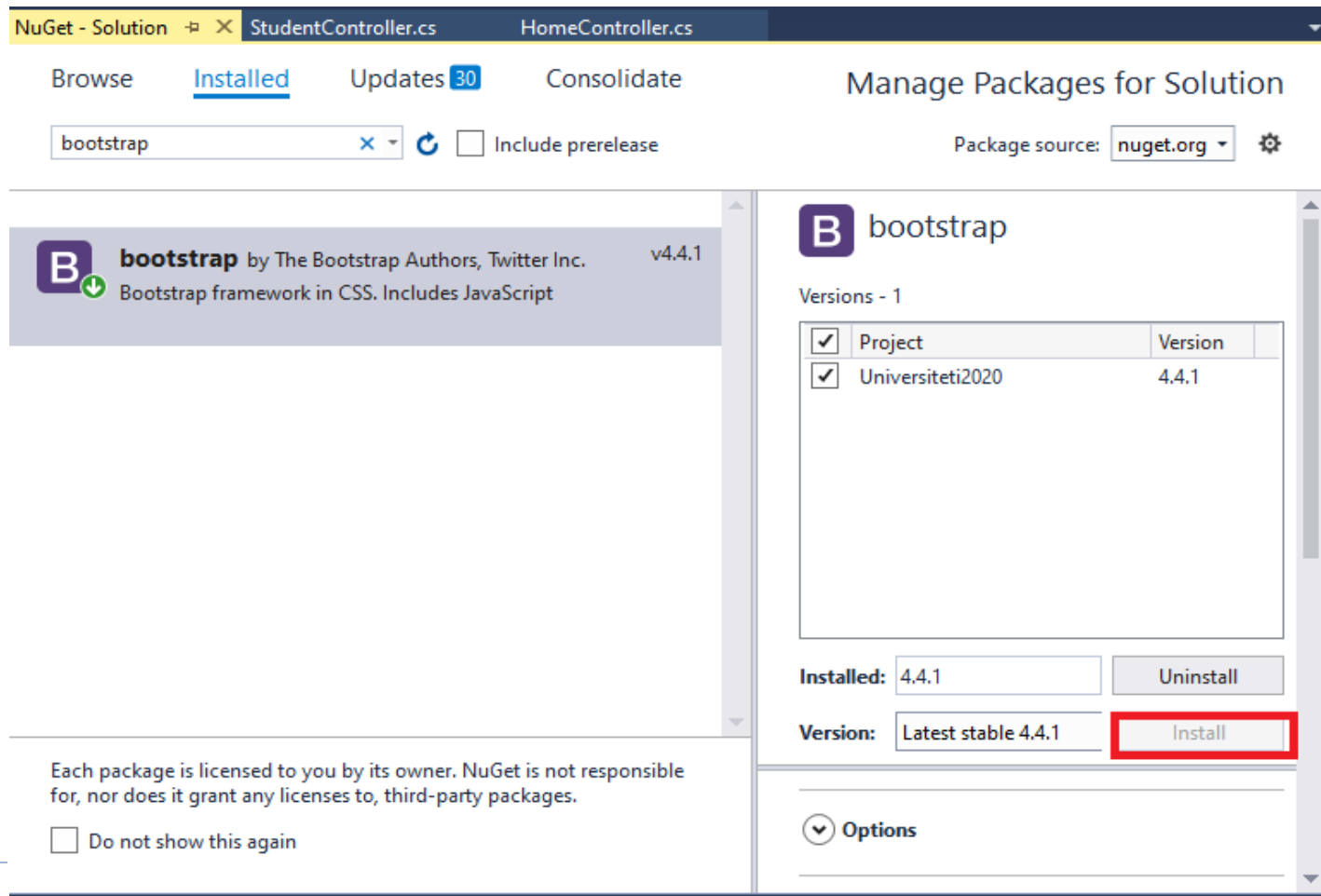
Versioni i bootstrap në Visual Studio 2017 është 3.3 ndërsa versionet e stileve aktuale janë sipas versionit bootstrap 4.4, kjo mosperputhje midis versioneve do te silte qe stilet te mos shfaqeshin ashtu sic duhet.

1. Zgjidhja e parë do te ishte migrimi i bootstrap në projektin tuaj sipas versionit me te fundit



# Probleme të mundshme

- Aktualisht në figuren me poshte eshte instaluar versioni me i fundit prandaj butoni Install eshte i çaktivizuar nese ju keni nje version me te vogel mjafton te shtypni Install



# Probleme të mundshme

- ▶ Pavaresisht migrimit ne nje version me te ri te bootstrap, serish stilet mund te mos shfaqen ashtu sic duhet atehere zgjidhja e dyte eshte qe ne file bootstrap.css qe hapet me pare ne nje tab te ri do te kishit url
- ▶ <https://bootswatch.com/4/flatly/bootstrap.css>
- ▶ Perpara se ti jepni save ketij file ndryshoni url per te marre bootstrap ne versionin 3
- ▶ <https://bootswatch.com/3/flatly/bootstrap.css>

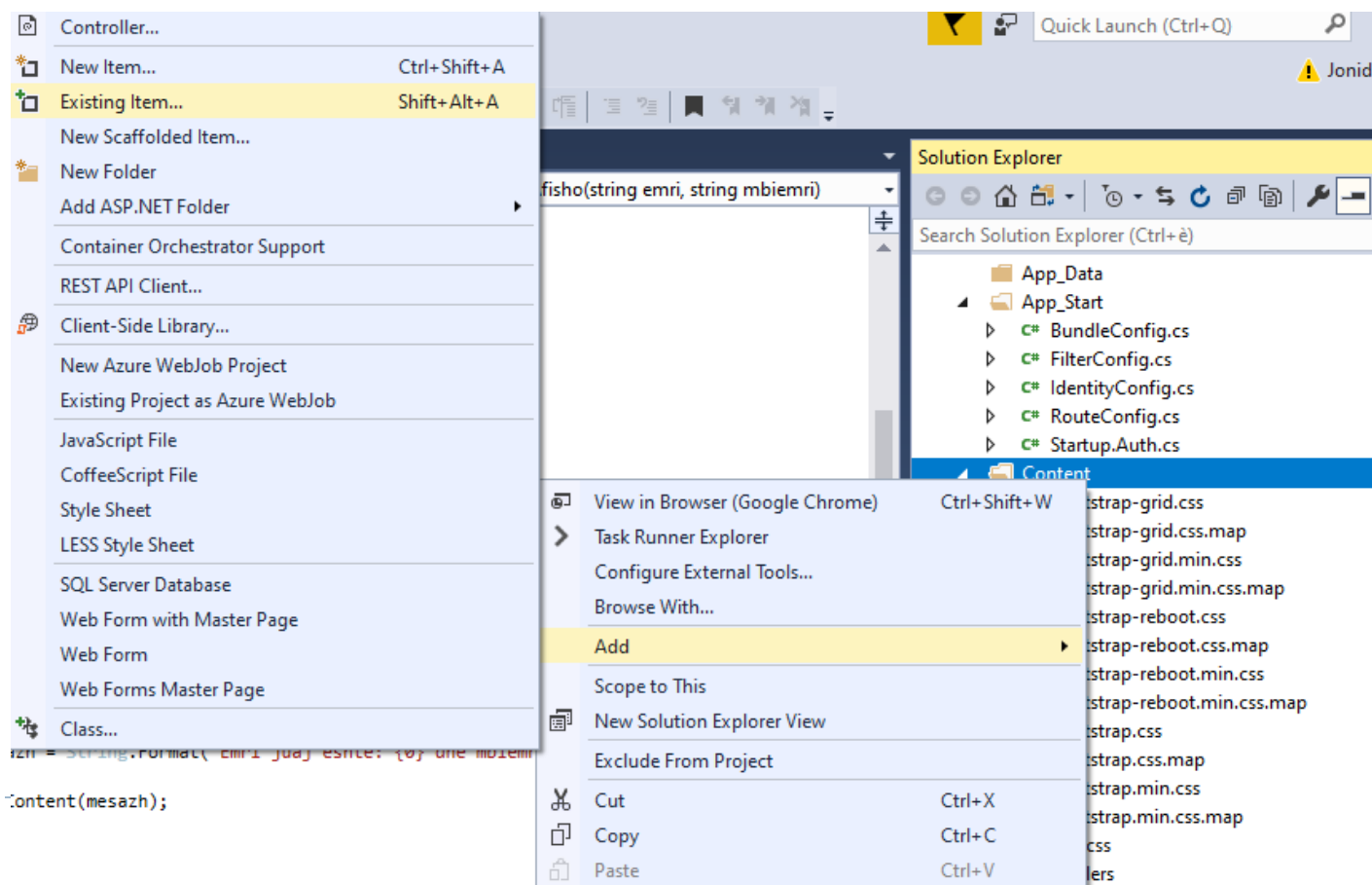
← → ↻ 🔒 bootswatch.com/3/flatly/bootstrap.css

```
@import url("https://fonts.googleapis.com/css?family=Lato:400,700,400italic&display=swap");
/*!
 * bootswatch v3.4.1
 * Homepage: http://bootswatch.com
 * Copyright 2012-2019 Thomas Park
 * Licensed under MIT
 * Based on Bootstrap
 */
/*!
 * Bootstrap v3.4.1 (https://getbootstrap.com/)
 * Copyright 2011-2019 Twitter, Inc.
 * Licensed under MIT (https://github.com/twbs/bootstrap)
 */
/*! normalize.css v3.0.3 | MIT License | github.com/
html {
  font-family: sans-serif;
  -ms-text-size-adjust: 100%;
  -webkit-text-size-adjust: 100%;
}
body {
  margin: 0;
}
article,
```

Back	Alt+Left Arrow
Forward	Alt+Right Arrow
Reload	Ctrl+R
Save as...	Ctrl+S
Print...	Ctrl+P
Cast...	
Translate to English	
View page source	Ctrl+U
Inspect	Ctrl+Shift+I

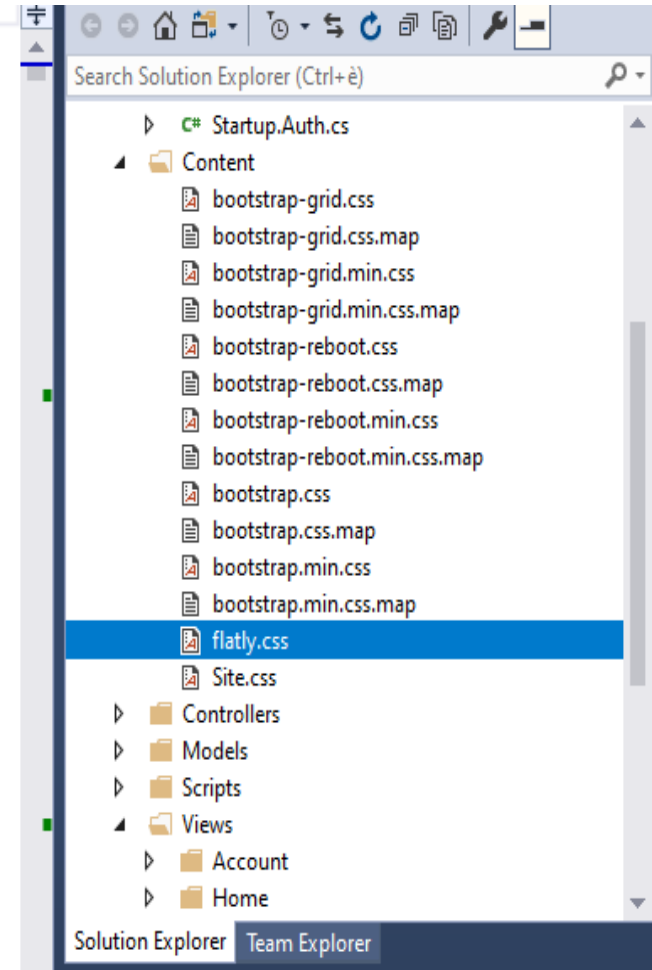
# Përfshirja e file të stilit

- ▶ Emertoni file sipas emrit të stilit dhe ruajeni diku lokalisht. Klikoni me të djathtë tek folder Content i projektit tuaj dhe shtoni file e ri të stilit si në figurën më poshtë:



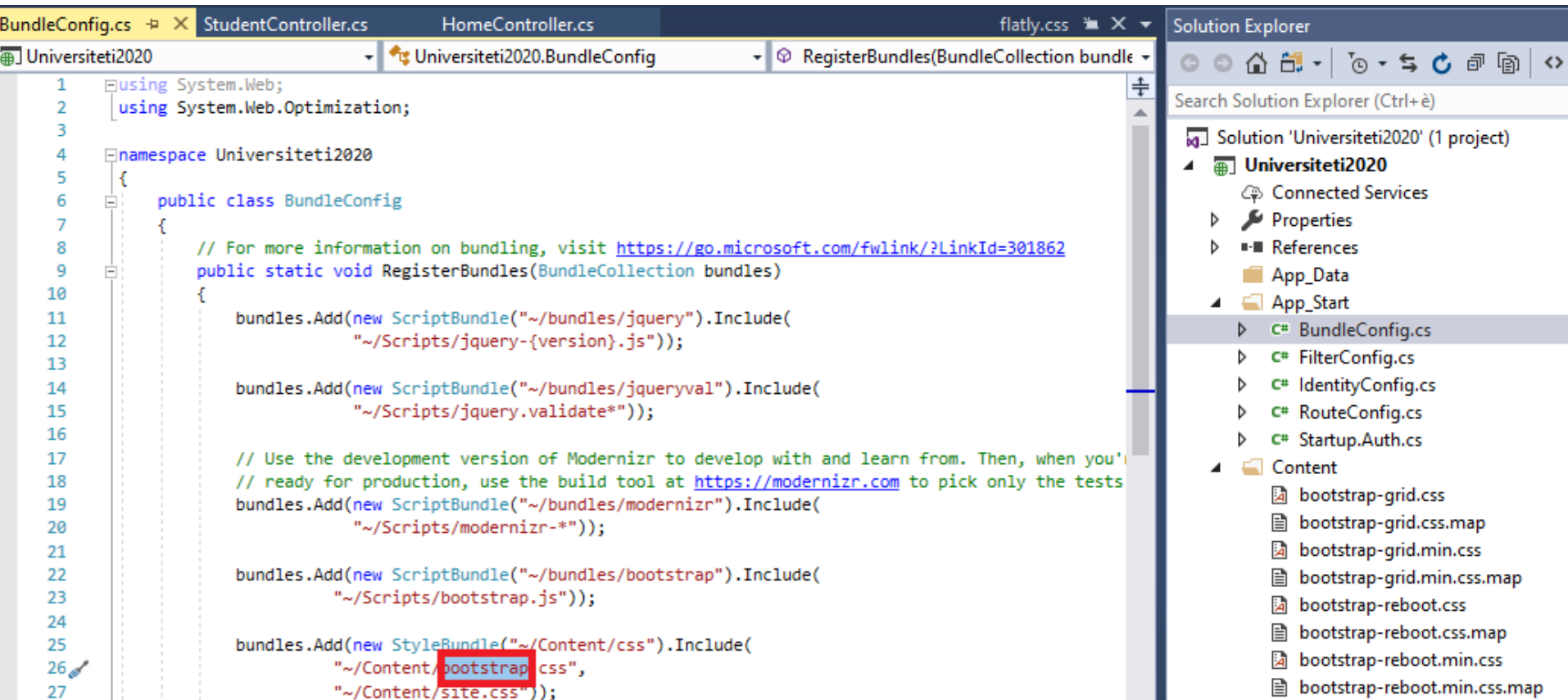
# Përfshirja e file të stilit

```
1 @import url("https://fonts.googleapis.com/css?family=Lato:400,700,400italic&display=swap");
2 /*!
3  * bootswatch v3.4.1
4  * Homepage: http://bootswatch.com
5  * Copyright 2012-2019 Thomas Park
6  * Licensed under MIT
7  * Based on Bootstrap
8  */
9 /*!
10 * Bootstrap v3.4.1 (https://getbootstrap.com/)
11 * Copyright 2011-2019 Twitter, Inc.
12 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
13 */
14 /*! normalize.css v3.0.3 | MIT License | github.com/necolas/normalize.css */
15 html {
16   font-family: sans-serif;
17   -ms-text-size-adjust: 100%;
18   -webkit-text-size-adjust: 100%;
19 }
20 body {
21   margin: 0;
22 }
23 article,
24 aside,
25 details,
26 figcaption,
27 figure,
28 footer,
29 header,
30 hgroup,
31 main,
32 ...
```



# Përfshirja e file të stilit

Per te ndryshuar referencen e bootstrap ekzistues me stilin e ri duhet te shkoni ne App\_Start → BundleConfig.cs dhe të modifikoni në rreshtin e fundit emrin e file nga bootstrap.css në flatly.css qe eshte file i stilit te ri (ose sipas emrit te file te stilit qe ju keni perzgjedhur)



The screenshot displays the Visual Studio IDE. The main window shows the `BundleConfig.cs` file in the `Universiteti2020` project. The code defines a `BundleConfig` class with a `RegisterBundles` method. The method registers several bundles, including jQuery, jQuery validation, Modernizr, and Bootstrap. The Bootstrap bundle is registered with the path `~/Content/bootstrap.css`, which is highlighted with a red box. The Solution Explorer on the right shows the project structure, including the `App_Start` folder and the `BundleConfig.cs` file.

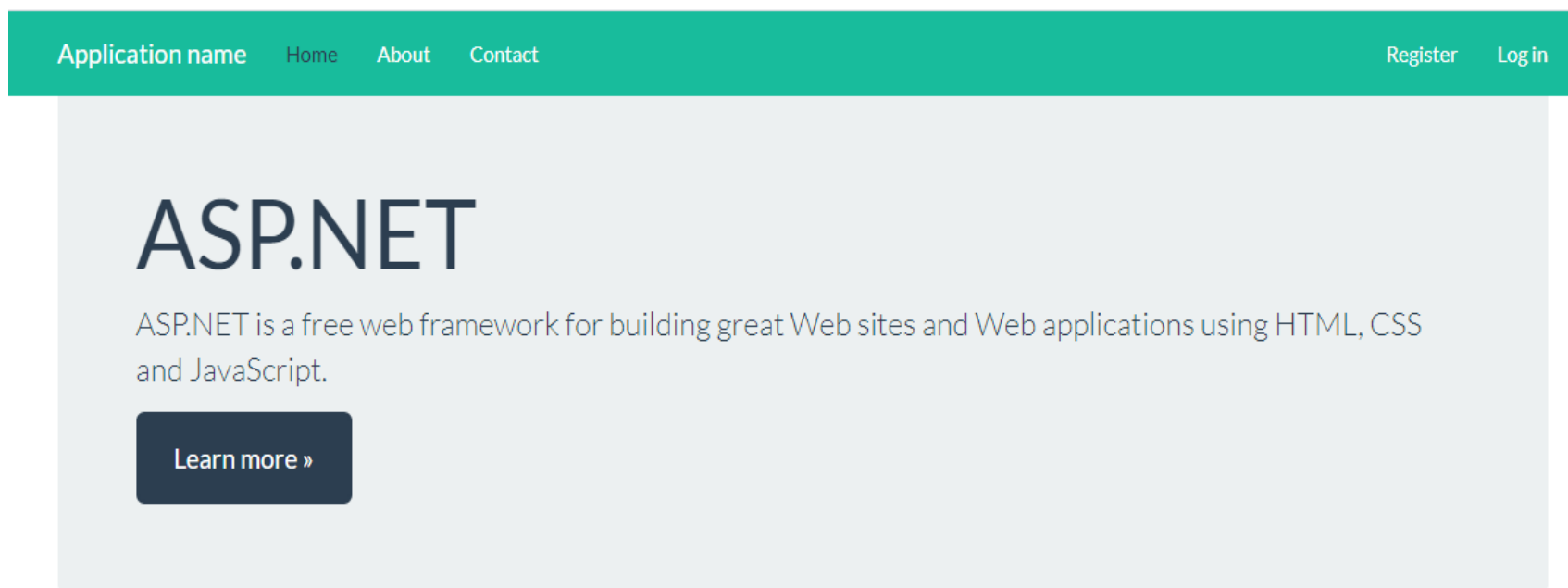
```
1 using System.Web;
2 using System.Web.Optimization;
3
4 namespace Universiteti2020
5 {
6     public class BundleConfig
7     {
8         // For more information on bundling, visit https://go.microsoft.com/fwlink/?LinkId=301862
9         public static void RegisterBundles(BundleCollection bundles)
10         {
11             bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
12                 "~/Scripts/jquery-{version}.js"));
13
14             bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
15                 "~/Scripts/jquery.validate*"));
16
17             // Use the development version of Modernizr to develop with and learn from. Then, when you're
18             // ready for production, use the build tool at https://modernizr.com to pick only the tests
19             bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
20                 "~/Scripts/modernizr-*"));
21
22             bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
23                 "~/Scripts/bootstrap.js"));
24
25             bundles.Add(new StyleBundle("~/Content/css").Include(
26                 "~/Content/bootstrap.css",
27                 "~/Content/site.css"));
28 }
```

Solution Explorer: Solution 'Universiteti2020' (1 project)

- Universiteti2020
  - Connected Services
  - Properties
  - References
  - App\_Data
  - App\_Start
    - BundleConfig.cs
    - FilterConfig.cs
    - IdentityConfig.cs
    - RouteConfig.cs
    - Startup.Auth.cs
  - Content
    - bootstrap-grid.css
    - bootstrap-grid.css.map
    - bootstrap-grid.min.css
    - bootstrap-grid.min.css.map
    - bootstrap-reboot.css
    - bootstrap-reboot.css.map
    - bootstrap-reboot.min.css
    - bootstrap-reboot.min.css.map

# Përfshirja e file të stilit

- Jepini save dhe run dhe me poshte do tju shfaqet aplikacioni me nje stil te ri:



## Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

[Learn more »](#)

## Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

[Learn more »](#)

## Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

[Learn more »](#)



# Modelet

---

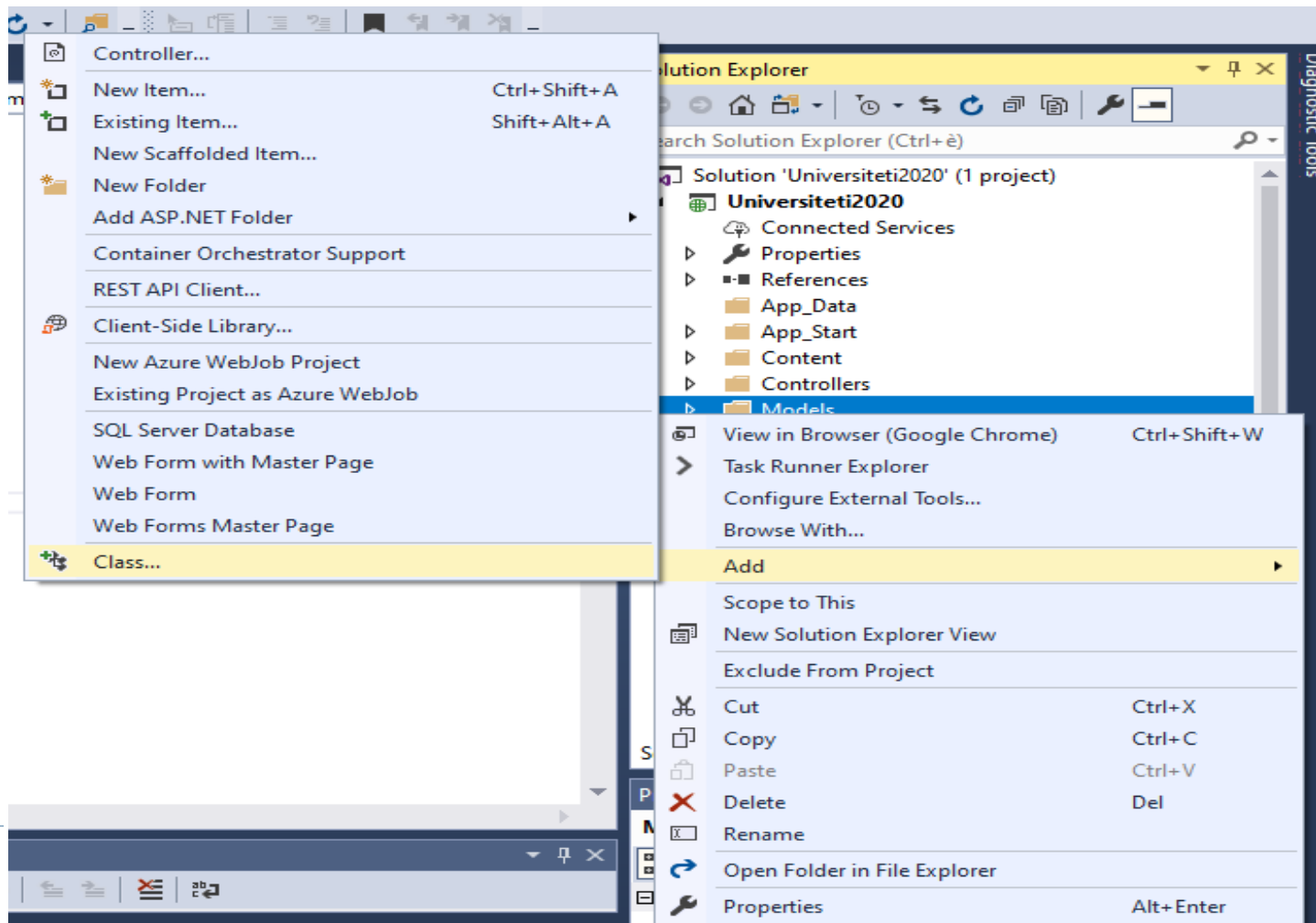
Modelet janë objekte që ju i përdorni për të dërguar informacion në bazën e të dhënave , për të performuar llogaritje të biznesit por edhe për të renderuar një View duke i kaluar si parameter një model. Këto objekte do të përfaqësojnë fushën në të cilën është i fokusuar aplikacioni dhe modelet do të ishin objekte të cilat ju dëshironi të shfaqni, të ruani, të modifikoni ose fshini.

Në vijim do të shikojmë se si krijohen modelet dhe se si mund të kalojmë nga modeli në controller dhe më pas në View.

Duam të shtojmë disa modele për projektin Universiteti2020. Universiteti duhet të ruajë të dhëna mbi degët, studentët dhe profesorët, kurset etj.

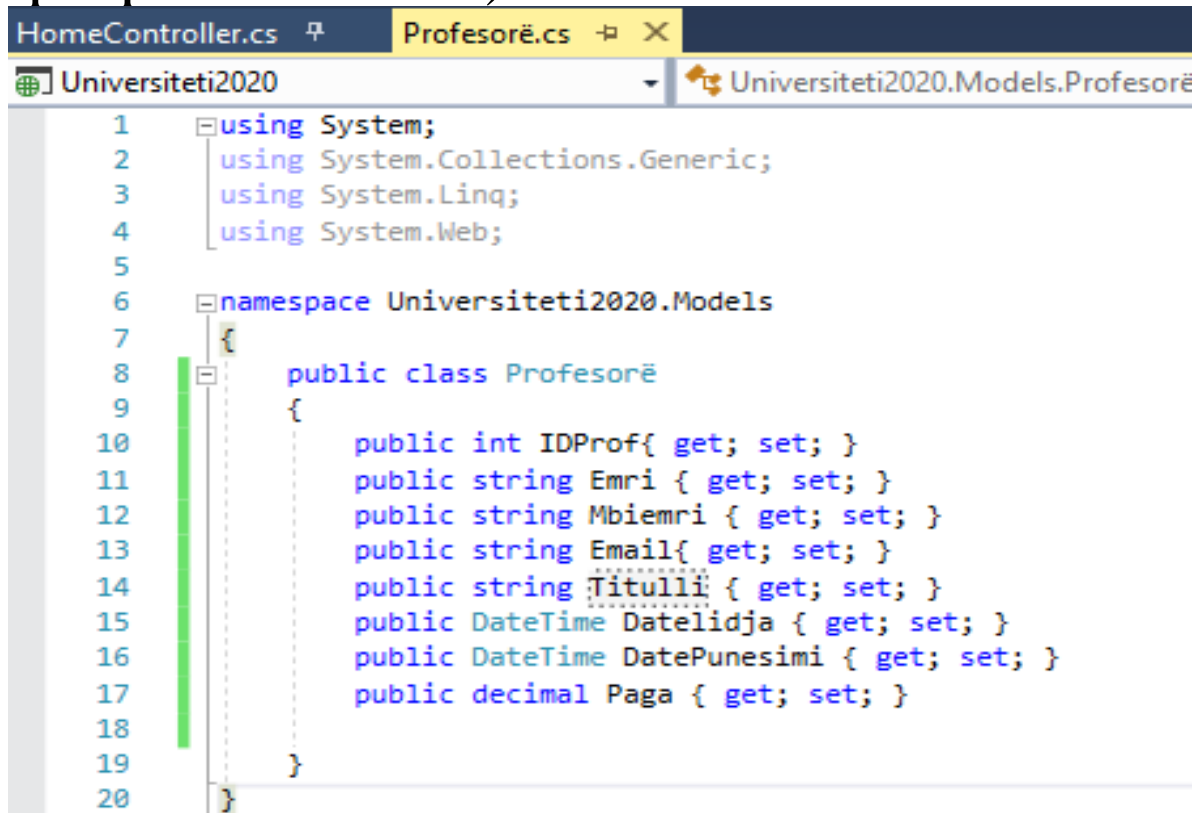
# Rast studimor Universiteti

- ▶ Do të krijojmë modelin e parë atë të profesorit. Shkoni në folderin Models klikoni me të djathtën dhe përzgjidhni Add→Class... dhe e emertojme Profesorë



# Rast studimor Universiteti

Per profesoret do te ruanim disa te dhena per kete do te deklarojme disa properties si me poshte:(kujtoni deklarimin e properties ne C#).



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5
6  namespace Universiteti2020.Models
7  {
8      public class Profesorë
9      {
10         public int IDProf{ get; set; }
11         public string Emri { get; set; }
12         public string Mbiemri { get; set; }
13         public string Email{ get; set; }
14         public string Titulli { get; set; }
15         public DateTime Datelidja { get; set; }
16         public DateTime DatePunesimi { get; set; }
17         public decimal Paga { get; set; }
18     }
19 }
20
```

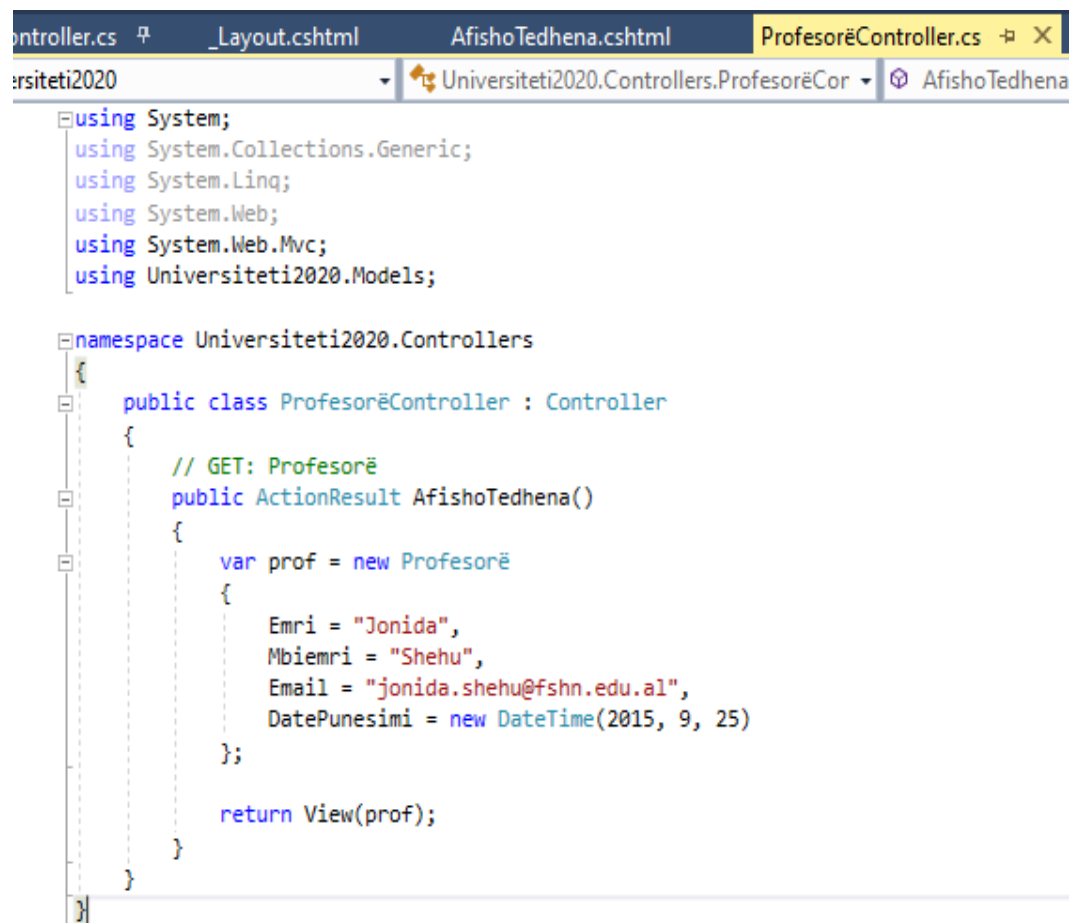
# Rast studimor Universiteti

Për të bërë lidhjen midis modelit dhe vieë duhet te krijojmë fillimisht një controller me të njëjtin emër si modeli, pra ProfesorëController (Rikujtoni krijimin e kontrollerave nga leksioni i mëparshëm)

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Mvc;
6
7  namespace Universitasiteti2020.Controllers
8  {
9      public class ProfesorëController : Controller
10     {
11         // GET: Profesorë
12         public ActionResult Index()
13         {
14             return View();
15         }
16     }
17 }
```

# Rast studimor Universiteti

- Krijoni një metodë AfishoTedhena() në controllerin ProfesorëController dhe krijoni aty një instancë të re liste nga klasa Profesorë. Kujdes të shtoni namespace e modeleve nëpërmjet *Using Universiteti2020.Models;*



```
controller.cs  _Layout.cshtml  AfishoTedhena.cshtml  ProfesorëController.cs  X
Universiteti2020  Universiteti2020.Controllers.ProfesorëCor  AfishoTedhena

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Universiteti2020.Models;

namespace Universiteti2020.Controllers
{
    public class ProfesorëController : Controller
    {
        // GET: Profesorë
        public ActionResult AfishoTedhena()
        {
            var prof = new Profesorë
            {
                Emri = "Jonida",
                Mbiemri = "Shehu",
                Email = "jonida.shehu@fshn.edu.al",
                DatePunesimi = new DateTime(2015, 9, 25)
            };

            return View(prof);
        }
    }
}
```

# Rast studimor Universiteti

- Klikoni me të djathtën tek metoda `AfishoTedhena()` dhe shtoni një view

Add View

View name: AfishoTedhena

Template: Empty (without model)

Model class:

Data context class:

Options:

☐ Create as a partial view

☐ Reference script libraries

☒ Use a layout page:

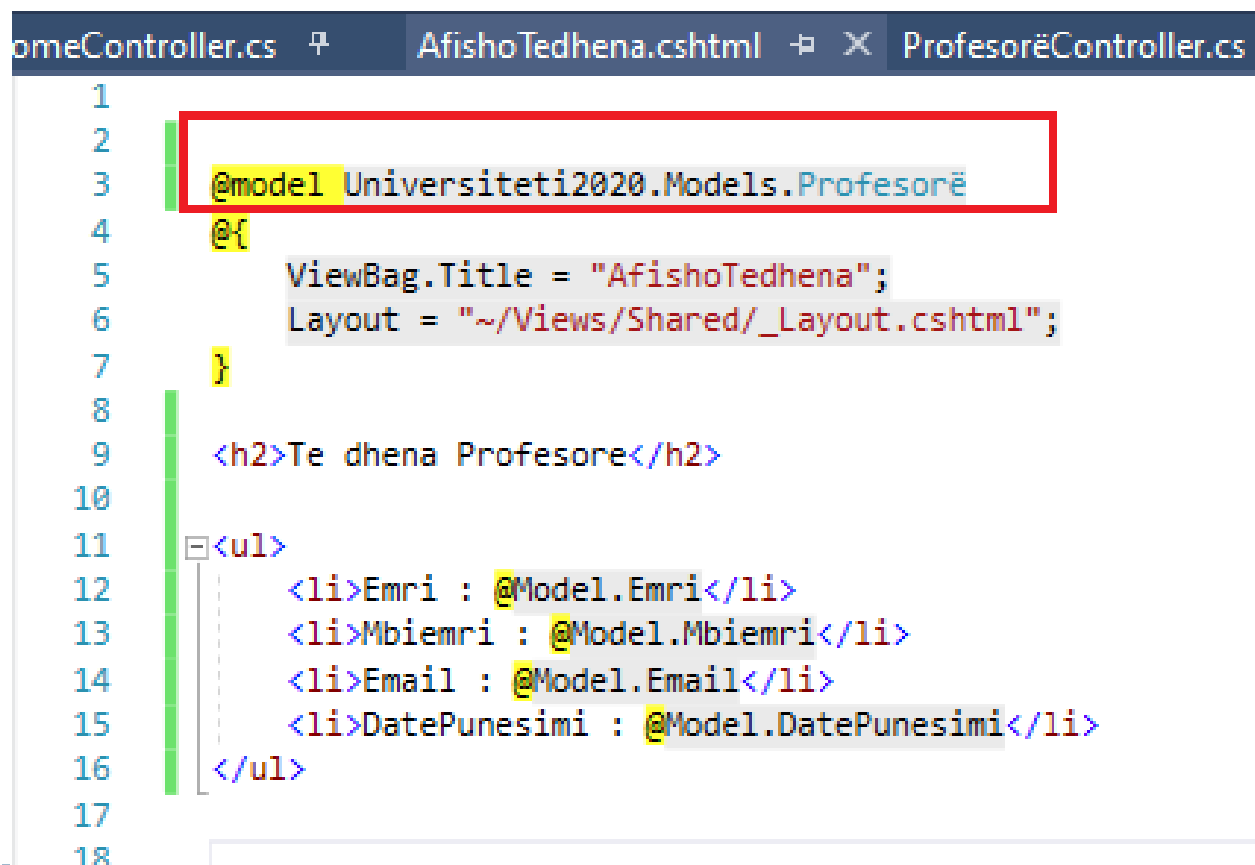
~/Views/Shared/\_Layout.cshtml

(Leave empty if it is set in a Razor \_viewstart file)

Add Cancel

# Rast studimor Universiteti

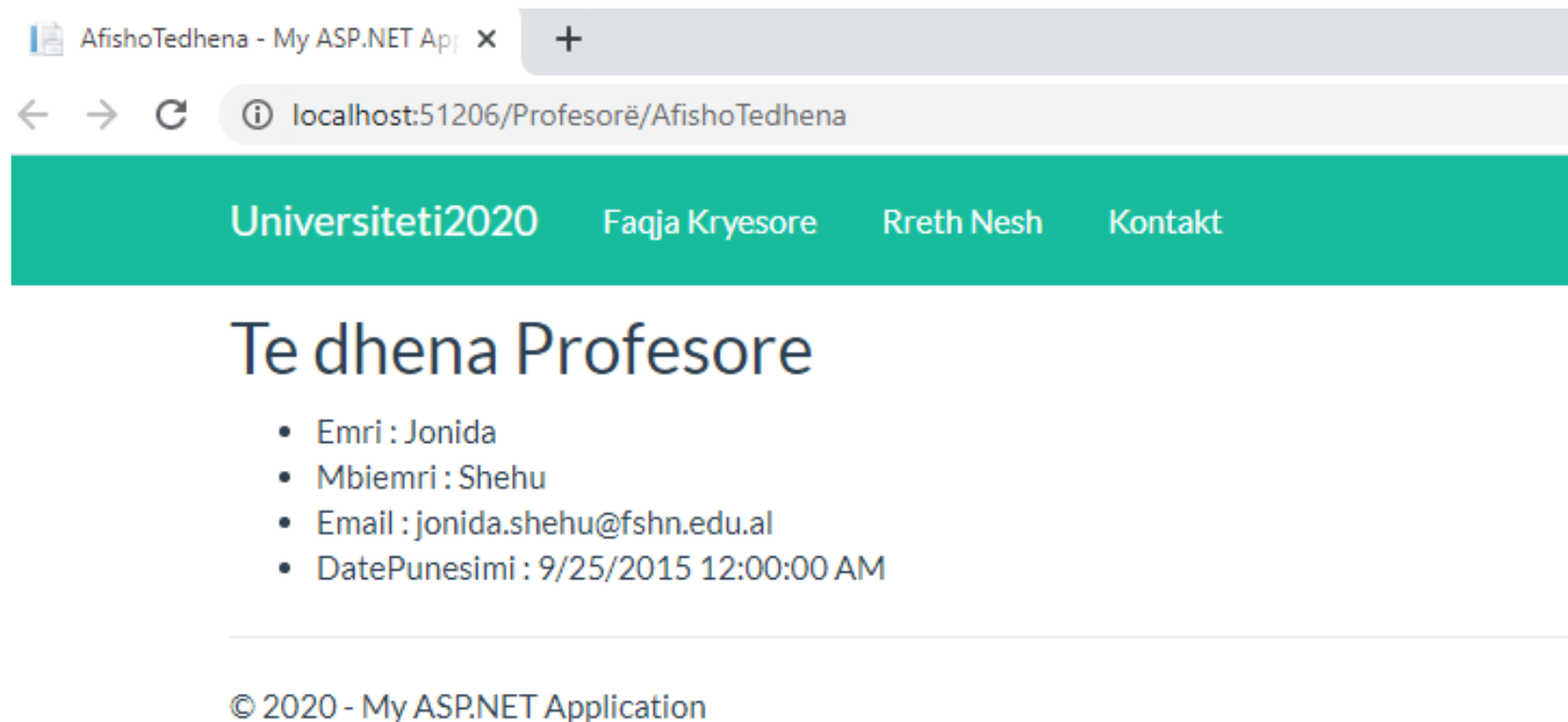
- Tek View afishoni te dhenat ne forme liste si me poshte, kujdes eshte e rendesishme te deklarohet modeli nga i cili po afishohen te dhenat vini re vijeimin e kuq ne figure:



```
omeController.cs  AfishoTedhena.cshtml  ProfesorëController.cs
1
2
3  @model Universiteti2020.Models.Profesorë
4  @{
5      ViewBag.Title = "AfishoTedhena";
6      Layout = "~/Views/Shared/_Layout.cshtml";
7  }
8
9  <h2>Te dhena Profesore</h2>
10
11  <ul>
12      <li>Emri : @Model.Emri</li>
13      <li>Mbiemri : @Model.Mbiemri</li>
14      <li>Email : @Model.Email</li>
15      <li>DatePunesimi : @Model.DatePunesimi</li>
16  </ul>
17
18
```

# Rezultati

---



The screenshot shows a web browser window with a single tab titled "AfishoTedhena - My ASP.NET Ap". The address bar displays "localhost:51206/Profesorë/AfishoTedhena". The page has a teal header with the text "Universiteti2020" and three navigation links: "Faqja Kryesore", "Rreth Nesh", and "Kontakt". The main content area features the heading "Te dhena Profesore" followed by a bulleted list of profile information. At the bottom, a horizontal line separates the content from the footer, which contains the copyright notice "© 2020 - My ASP.NET Application".

AfishoTedhena - My ASP.NET Ap x +

localhost:51206/Profesorë/AfishoTedhena

Universiteti2020 Faqja Kryesore Rreth Nesh Kontakt

## Te dhena Profesore

- Emri : Jonida
- Mbiemri : Shehu
- Email : jonida.shehu@fshn.edu.al
- DatePunesimi : 9/25/2015 12:00:00 AM

© 2020 - My ASP.NET Application