

Leksion 1 MVC

- Të kuptojmë ASP.NET MVC
 - Si të krijojmë aplikacione me MV
- Si të krijojmë Controlle
 - Njohje me View

Hyrje

- ▶ ASP.NET MVC është një framework për ndërtimin e aplikacioneve web sipas strukturës Model-View-Controller.
- ▶ Kur u krijua ASP.NET për herë të parë në 2002, ai identifikohet me Web Forms, me të cilat u paraqitën lehtësi si përdorimi i kontrolleve server, mundësia për të ruajtur gjendjen e tyre por në të njëjtën kohë u vunë re dhe vështirësi si menaxhimi i ciklit jetësor të faqeve ose edhe personalizimi i HTML.
- ▶ Megjithatë mund të krijoheshin aplikacione web shumë shpejt, përgjigjia direkt e kërkesave HTTP, ndërtimi i faqeve duke përdorur dëgjuesit, module ose kod të shkruajtur. Të gjitha këto mund të realizoheshin por nuk ishin shumë praktike.
- ▶ ASP.NET MVC u lëshua në vitin 2007 dhe modeli MVC por bëhet ndër më të njohurit për ndërtimin e framework-eve web.



Struktura MVC

Model-View-Controller (MVC) ka qenë një strukturë arkitekturore e rëndësishme në informatikë për shumë vite. Fillimisht ajo quhej Thing-Model-View-Editor në 1979, më pas për thjeshtësi u emërtua Model-View-Controller.

Kjo strukturë qëndron shumë mirë në aplikacionet web dhe ka një mënyrë elegante në ndarjen e pjesëve në aplikacion ku logjika e aksesit të të dhënave ndahet nga pamjet.

Ajo përdoret në shumë framework si Java, C++, PHP dhe është e pavarur nga platforma.

Ndarja në 3 komponente:

- Modeli: Një set klasash që përshkruajnë të dhënat tuaja, dhe gjithashtu fushën dhe rregullat e biznesit
 - Pamja (View): Përcakton se si do të shfaqet ndërfaqja e aplikacionit.
 - Kontroller (Controller): Një set klasash që menaxhojnë komunikimin nga përdoruesi drejt rrjedhës së aplikacionit dhe logjikës specifike të tij.
-



Struktura MVC e aplikuar në framework Web

Struktura MVC përdoret shpesh në programimin web, në ASP.NET MVC:

- Modelet do të ishin klasa që përfaqësojnë fushën e interesit. Këto objekte të fushës shpesh enkapsulojnë të dhëna që ruhen në bazën e të dhënave por edhe kod që manipulon të dhënat dhe zbaton fushën specifike të llogjikës së biznesit
- View: është një template për të gjeneruar në mënyrë dinamike HTML.
- Controller: është një klasë e veçantë që menaxhon marrëdhënien midis View dhe Model. I përgjigjet inputit të përdoruesit , komunikon me modelin dhe përcakton se cilën View të shfaqë. Në ASP.NET MVC, kontrollerrat përfundojnë me prapashtesën Controller.



Sintaksa Razor

Razor do të konsiderohej si një inovacion në shfaqjen e HTML, sintaksa u shtua me ASP.NET MVC 3. Më parë për view në MVC 1 dhe MVC2 përdorej sintaksa e njëjtë me atë që përdoret në Web Forms si më poshtë:

- ▶ *<%@ Page Language="C#"*
 - ▶ *MasterPageFile="~/Views/Shared/Site.Master" Inherits=*
 - ▶ *"System.Web.Mvc.ViewPage<MvcMusicStore.ViewModels.StoreBrowseViewModel>" %>*
 - ▶ *<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">Browse Albums </asp:Content>*
 - ▶ *<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">*
 - ▶ *<div class="genre">*
 - ▶ *<h3><%= Model.Genre.Name %> Albums</h3>*
-



Sintaksa Razor

Razor u dizenjua specifikisht si një sintaksë pamjesh. Ka në fokus shfaqen e template të kodit për gjenerimin e HTML. Kjo sintaksë është e thjeshtë për tu përdorur dhe lexuar ndryshe nga trajta XML e pamjeve të Web Forms.

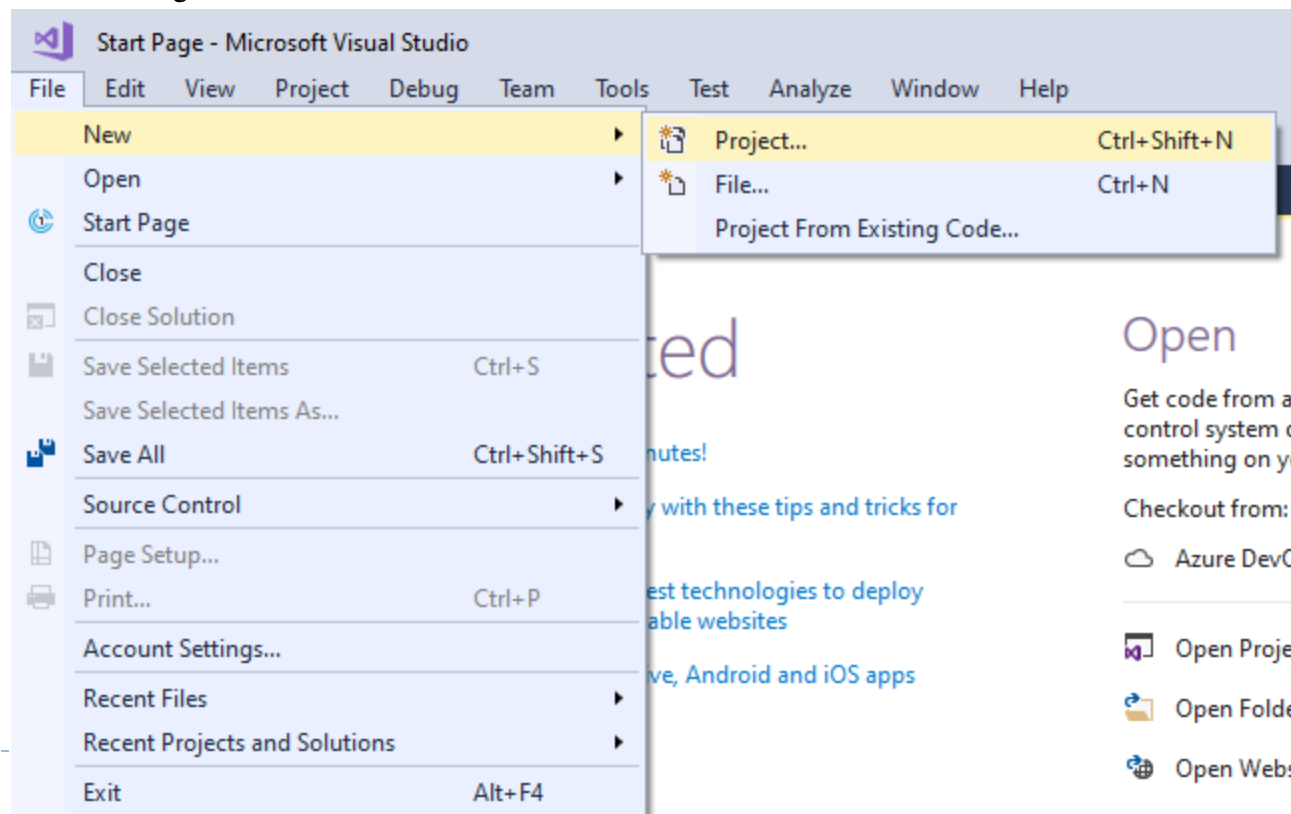
Sintaksa jepet si më poshtë:

```
@model MvcMusicStore.Models.Genre
@{ViewBag.Title = "Browse Albums";}
<div class="genre">
<h3><em>@Model.Name</em> Albums</h3>
<ul id="album-list">
@foreach (var album in Model.Albums)
{
  ▶ }
```

Krijimi i një Aplikacioni në ASP.NET MVC

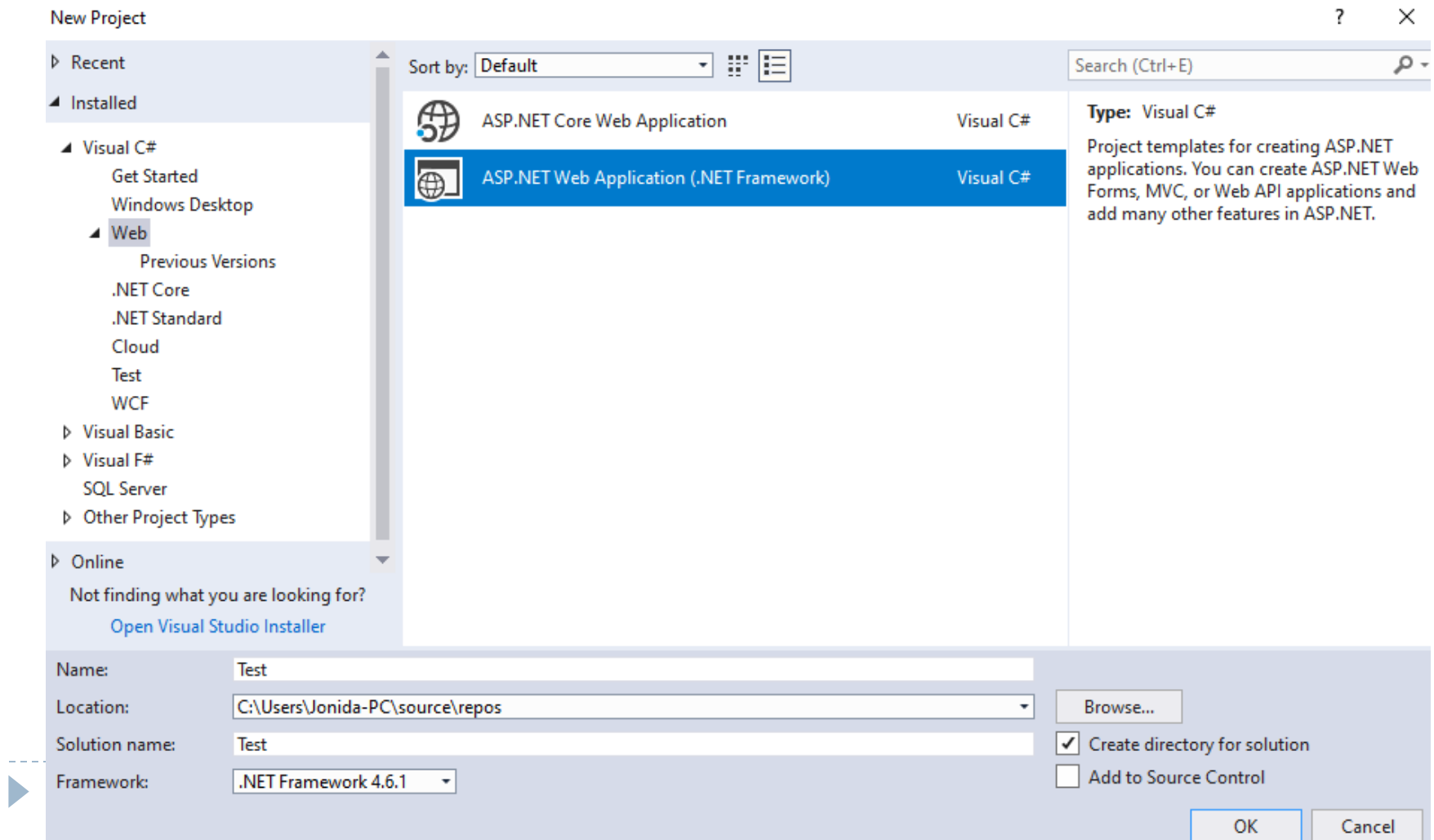
Krijimi i aplikacioneve në ASP.NET MVC mund të bëhet duke përdorur versione të Visual Studio 2013(Shembujt në libër janë më këtë version) edhe më lartë si Visual Studio 2017 ose 2019. Community Version. Hapat janë si më poshtë(Visual Studio 2017):

1. File→New Project



Krijimi i një Aplikacioni në ASP.NET MVC

- ▶ Tek seksioni i Templateve të instaluara në të majtë zgjidhet gjuha C# Web dhe përzgjidhet ASP.NET Web Application (.NET Framework), tek fusha Name përcaktoni një emër për projektin.



Krijimi i një Aplikacioni në ASP.NET MVC

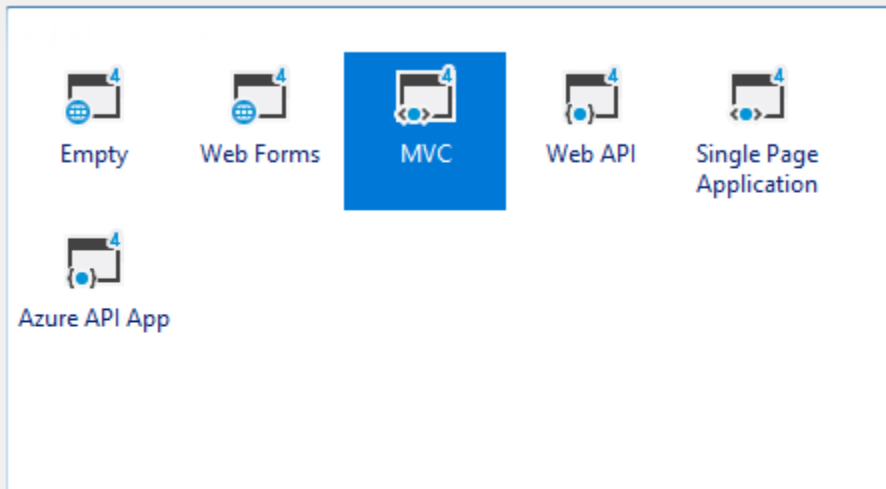
- ▶ Pasi krijohet aplikacioni ju shfaqet një dritare në të cilën keni mundësi të përzgjidhni
- ▶ Një template → Do të zgjedhim MVC
- ▶ Shtohen folder specifike dhe referencat kryesore sipas template.
- ▶ Mund të shtohen testet njësi.
- ▶ Mund të përzgjidhet mënyra e autentifikimit.



Krijimi i një Aplikacioni në ASP.NET MVC

New ASP.NET Web Application - Test

? X



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

[Learn more](#)

Add folders and core references for:

☐ Web Forms ☒ MVC ☐ Web API

☐ Enable Docker Compose support (Requires [Docker for Windows](#))

☐ Add unit tests

Test project name:

Authentication: **No Authentication**

[Change Authentication](#)

OK

Cancel

Autentifikimi

New ASP.NET Web Application - Test

? X



A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating

Change Authentication

X

- ☐ No Authentication
- ☒ Individual User Accounts
- ☐ Work or School Accounts
- ☐ Windows Authentication

For applications that store user profiles in a SQL Server database. Users can register, or sign in using their existing account for Facebook, Twitter, Google, Microsoft, or another provider.

[Learn more](#)

OK

Cancel

☐ Add unit tests

Test project name:

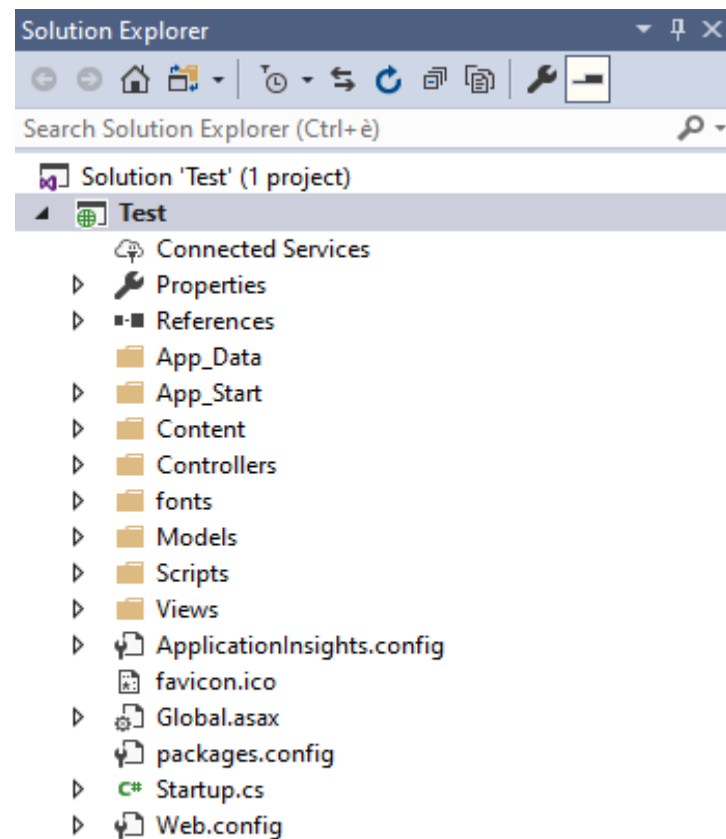
OK

Cancel

Struktura e aplikacionit

- ▶ Kur krijohet një aplikacion i ri ASP.NET MVC, Visual Studio shton në mënyrë automatike direktori dhe file.

Direktoria	Qëllimi
/Controllers	Vendosen klasat e Controller që menaxhojnë kërkesa URL
/Models	Mban klasat që përfaqësojnë dhe manipulojnë të dhënat.
/Views	Vendosen file të ndërfaqjes që janë përgjegjëse për shfaqjen e output si HTML.
/Scripts	Vendosen file Javascript (.js)
/fonts	Template të Bootstrap që përshijnë fonte Web të personalizuar.



Struktura e aplikacionit

Direktoria	Qëllimi
/Content	Vendosen file e CSS, imazheve dhe përmbajtje tjetër përveç skripteve.
/App_Data	Vendosen file e të dhënave
/App_Start	Vendosen file të konfigurimeve si Routing, Web API etj.

- ▶ **Global.asax** përmban funksione për evente që ndodhin gjatë ciklit jetësor të aplikacionit.
- ▶ **Packages.config** që përdoret nga NuGet Package Manager shërben për të menaxhuar varësitë e aplikacionit. Nëse aplikacioni ynë ka nevojë për disa paketa ose librari, menaxhuesi i paketave bën të mundur shkarkimin dhe instalimin e këtyre librave, gjithashtu updatimi i paketave të instaluar mund të bëhet shumë lehtësisht.
- ▶ **Web.config** është një file XML që përmban konfigurime në lidhje me aplikacionin si për shembull connectionStrings që shërbejnë për të përcaktuar lidhjen me bazën e të dhënave ose appSettings që përmban konfigurime të ndryshme.



Konvencionet në ASP.NET

- ▶ Aplikacionet ASP.NET MVC varen shumë nga konvencionet, kjo lehtëson zhvilluesit nga konfigurimet. Kur ju krijoni ose deklaroni një View të re ASP.NET e vendos atë nën direktorinë `\Views\`, ky konvencion ju lehtëson kur ju referoni një View nga një klasë Controller.
- ▶ Koncepti i konvencioneve mbi konfigurimet u bë i famshëm nga Ruby on Rails disa vite më parë dhe nënkuptonte që nëse tashmë dimë se si të ndërtojmë një aplikacion web atëherë le ta përfshijmë atë eksperiencë në framework në mënyrë që mos të na duhet të konfigurojmë çdo gjë sërish.
- ▶ Ky koncept është aplikuar në ASP.NET MVC, mjafton të vështrohen tre direktoritë kryesore.
 - ▶ Controllers
 - ▶ Models
 - ▶ Views

Konvencionet në ASP.NET

Emërtimet e këtyre direktorive nuk është e thënë që të përcaktohen në file web.config, ato vendosen aty automatikisht. Kjo ju kursen kohë por edhe ju lehtëson që të mos modiikoni një file XML si web.config.

- ▶ Çdo emër klase Controlleri do të përfundonte me fjalën Controller, psh ProduktController, KlientController dhe duhet të ruhet në direktorinë Controllers.
- ▶ Ka vetëm një direktori Views për të gjitha pamjet në aplikacion.
- ▶ Views që përdorin controllers qëndrojnë në një nëndirektori tek Views, dhe emërtohen sipas emrit të Controller por pa prapashtesën Controller. Shembull ProductController është controller dhe view do të qëndronte tek /Views/Product



Konvencionet thjeshtojnë komunikimin

- ▶ Ju shkruani kod që të komunikoni, por i drejtoheni dy tipe audiencash të ndryshme:
 1. Duhet ti komunikoni instruksione kompjuterit në mënyrë të qartë dhe pa dykuptimësi.
 2. Dëshironi që zhvilluesit të kenë mundësi që të lexojnë dhe kuptojnë kodin tuaj për mirëmbajtje të ose zgjerime të mëtejshme.



Controllers

- ▶ Në këtë pjesë do të shqyrtohet se si controllers i përgjigjen kërkesave HTTP dhe se si kthejnë informacionin në browser.
- ▶ Kontrollet në strukturën MVC janë përgjegjëse për tju përgjigjur inputit të përdoruesit, dhe shpesh bëjnë ndryshime në model në përgjigje të inputit. Në këtë mënyrë kontrollet janë përgjegjëse për rrjedhën e aplikacionit, punojnë me të dhënat që vijnë dhe sigurojnë të dhëna output në view përkatëse.
- ▶ Më parë web serverat i përgjigjeshin file statikë HTML të ruajtura në disk. Ndërkohë sot web serverat i përgjigjen HTML së gjeneruar në mënyrë dinamike nga skripte dinamike.
- ▶ Me MVC ndryshon paksa sepse URL është ajo që tregon mekanizmin e navigimit se cila klasë kontrolli do të përfaqësohet dhe cila metodë ose action do të thërritet së bashku me argumentat e metodës. Më pas është metoda e kontrollerit ajo që vendos se cilën view të shfaqë.

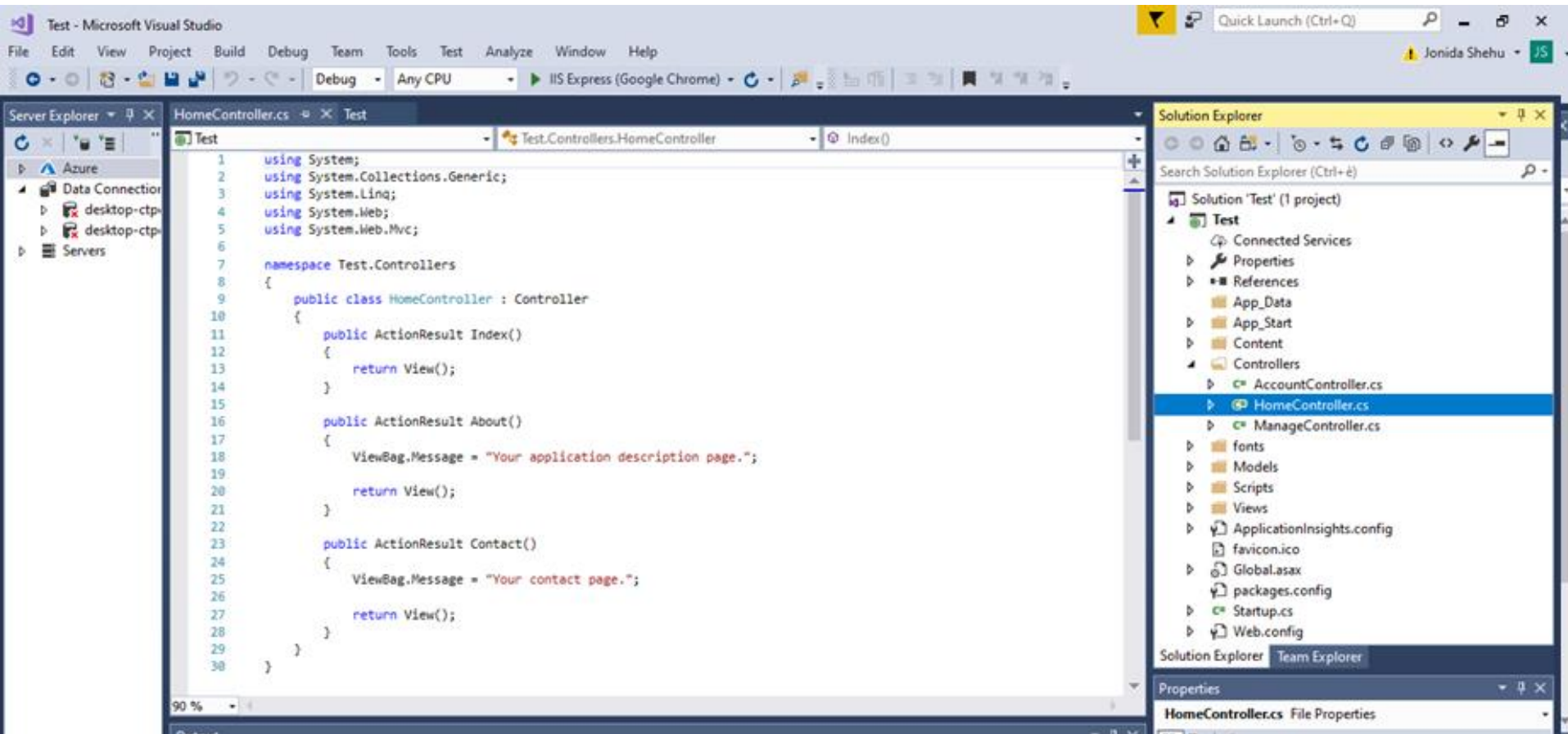


Shembull HomeController

- ▶ Kur ne përzgjedhëm në projektin Test të krijuar më parë template MVC dhe si autentifikim përzgjedhëm Individual User Accounts, automatikisht krijohen dy controllers:
- ▶ HomeController → përgjegjës për faqen kryesore dhe faqen about, së bashku me faqen contact
- ▶ AccountController → përgjegjës për kërkesa që lidhen me account, si login dhe regjistrim.



Shembull HomeController



Shembull HomeController

- ▶ Metoda Index në klasën HomeController është përgjegjëse për të vendosur se cfarë ndodh kur ju kërkonit faqen kryesore të një website. Nëse tek metoda About ndryshohet teksti “Pershendetje studente, kjo eshte faqja rreth nesh” dhe tek metoda Contact() ndryshojme “Ketu mund te kontaktohet”

- ▶ *public class HomeController : Controller*

- ▶ *{ public ActionResult Index()*

- ▶ *{ return View();*

- ▶ *}*

- ▶ *public ActionResult About()*

- ▶ *{*

- ▶ *ViewBag.Message = " Pershendetje studente, kjo eshte faqja rreth nesh!";*

- ▶ *return View();*

- ▶ *}*

- ▶ *public ActionResult Contact()*

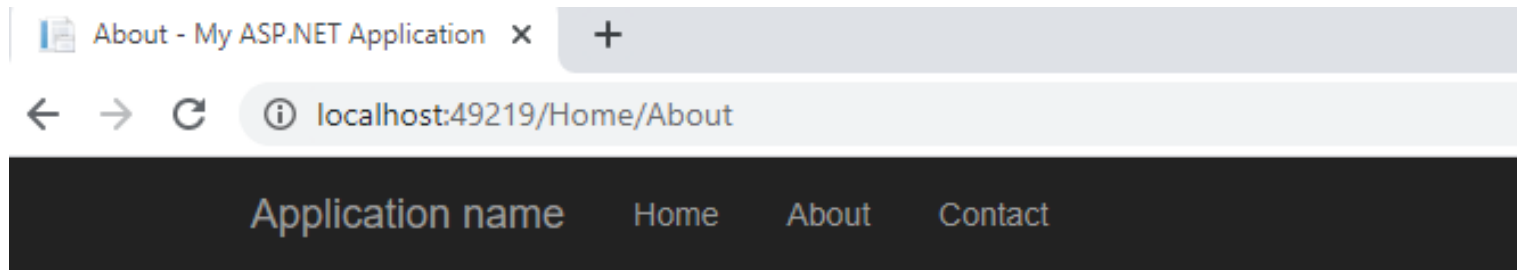
- ▶ *{*

- ▶ *ViewBag.Message = “Ketu mund te kontaktohet ”;*

- ▶ *return View();*

- ▶ *} }*

About



About.

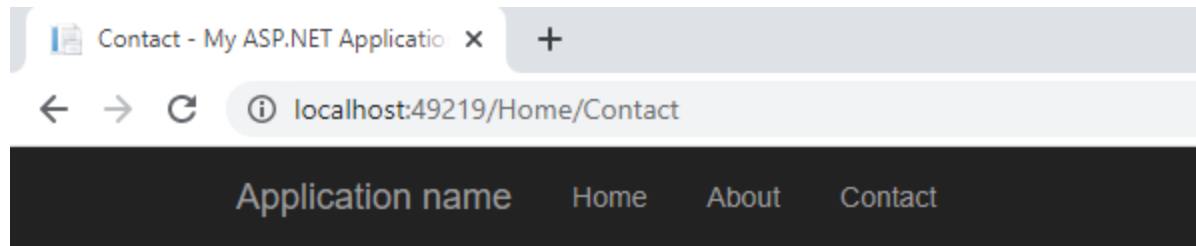
Pershendetje studente, kjo eshte faqja rreth nesh!

Use this area to provide additional information.

© 2020 - My ASP.NET Application



Contact



Contact.

Ketu mund te kontaktohet

One Microsoft Way
Redmond, WA 98052-6399
P: 425.555.0100

Support: Support@example.com

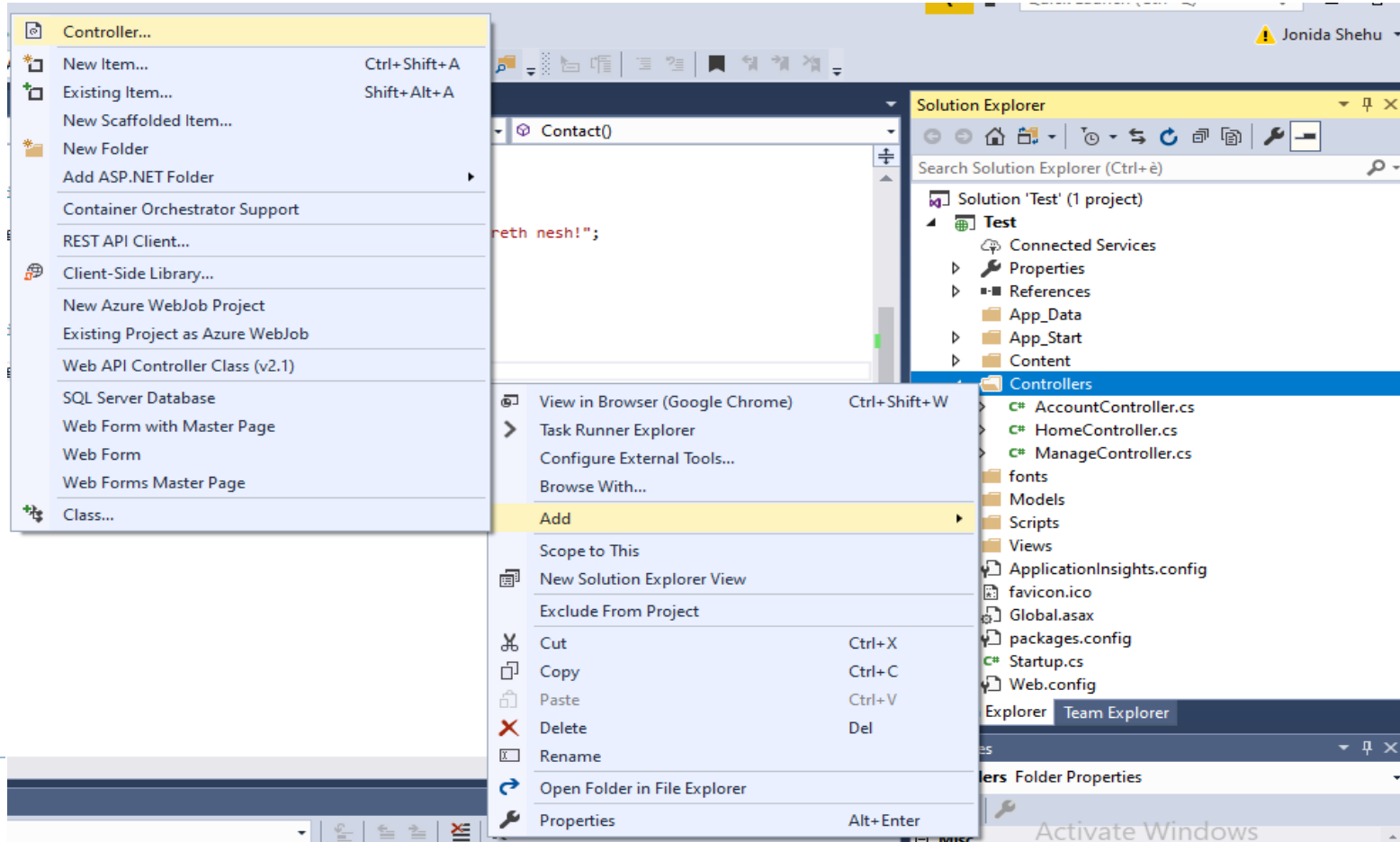
Marketing: Marketing@example.com

© 2020 - My ASP.NET Application



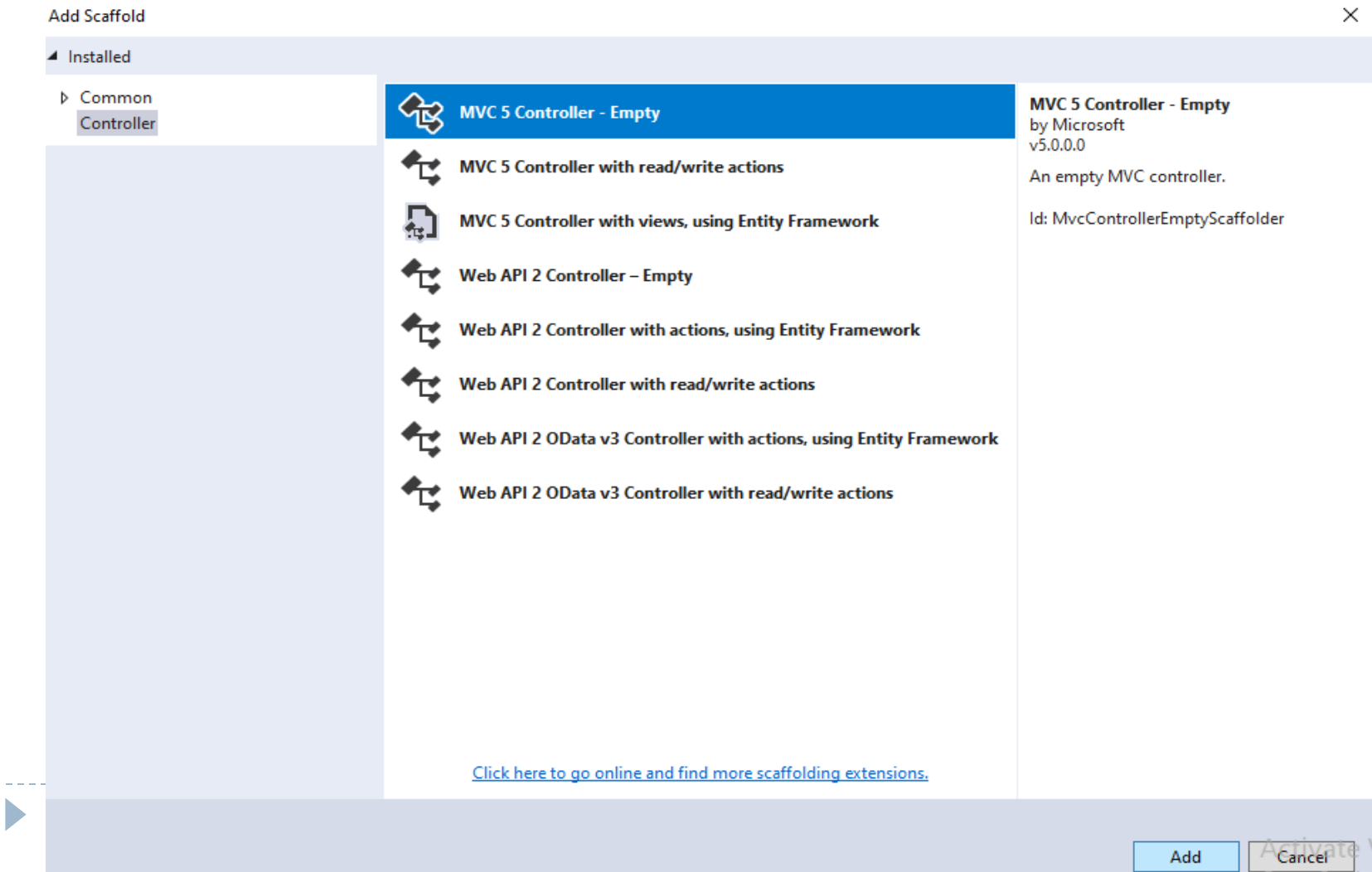
Krijimi i një Controller të ri

- ▶ Për të krijuar një controller të ri shkoni në Solution Explorer, në direktorinë Controllers, klikoni me të djathtën dhe përzgjidhni Add→Controller...

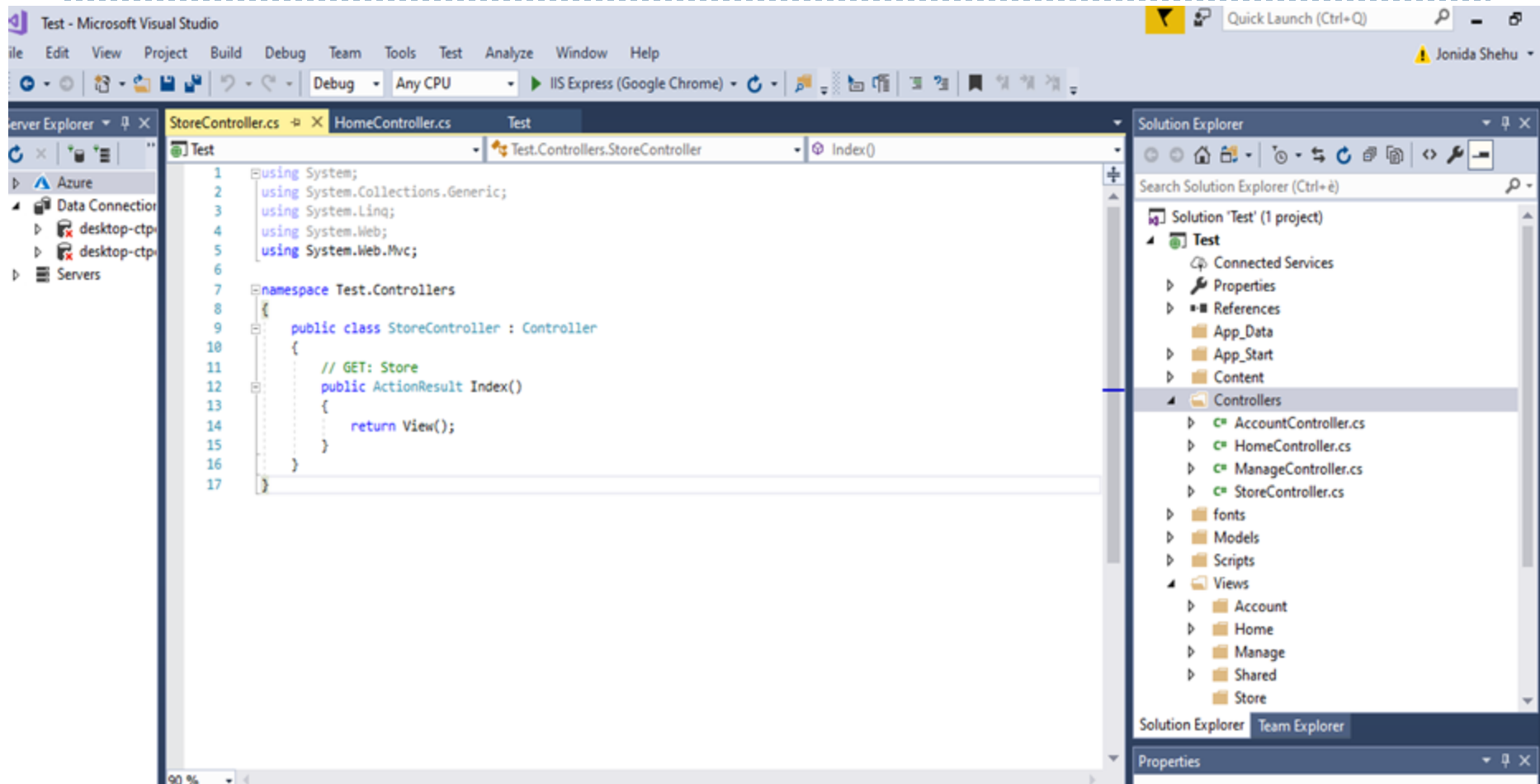


Krijimi i një Controller të ri

► Përzgjidhni MVC 5 Controller - Empty



Krijimi i një Controller të ri



Metodat dhe tipet e kthimit të një action tek Controller

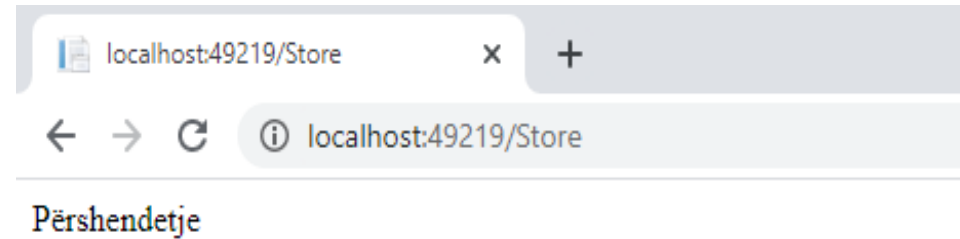
Tipi	Metoda ndihmëse	Përshkrimi
ViewResult	View()	Për të kthyer një View
PartialViewResult	PartialView()	Për të kthyer një View të pjesshme
RedirectResult	Redirect()	Për të ridrejtuar në një URL
RedirectToRouteResult	RedirectToAction()	Të ridrejtohet një action në vend të një URL
ContentResult	Content()	Për të kthyer tekst të thjeshtë
JsonResult	Json()	Për të kthyer një objekt Json
HttpNotFoundResult	HttpNotFound()	Për të kthyer mesazhin 404 nuk u gjet
FileResult	File()	Për të kthyer një file
EmptyResult		Kur nuk kërkohet të kthehet asnjë vlerë

Shembuj

Tipi i kthimit ActionResult do të përfshinte si nën tipe të gjitha ato të përcaktuara në tabelën në slide e mëparshëm.

Shembull 1 Afishimi si tekst

```
▶ public class StoreController : Controller
▶ {
▶     // GET: Store
▶     public ActionResult Index()
▶     {
▶         return Content("Përshendetje");
▶     }
▶ }
```



Shembuj

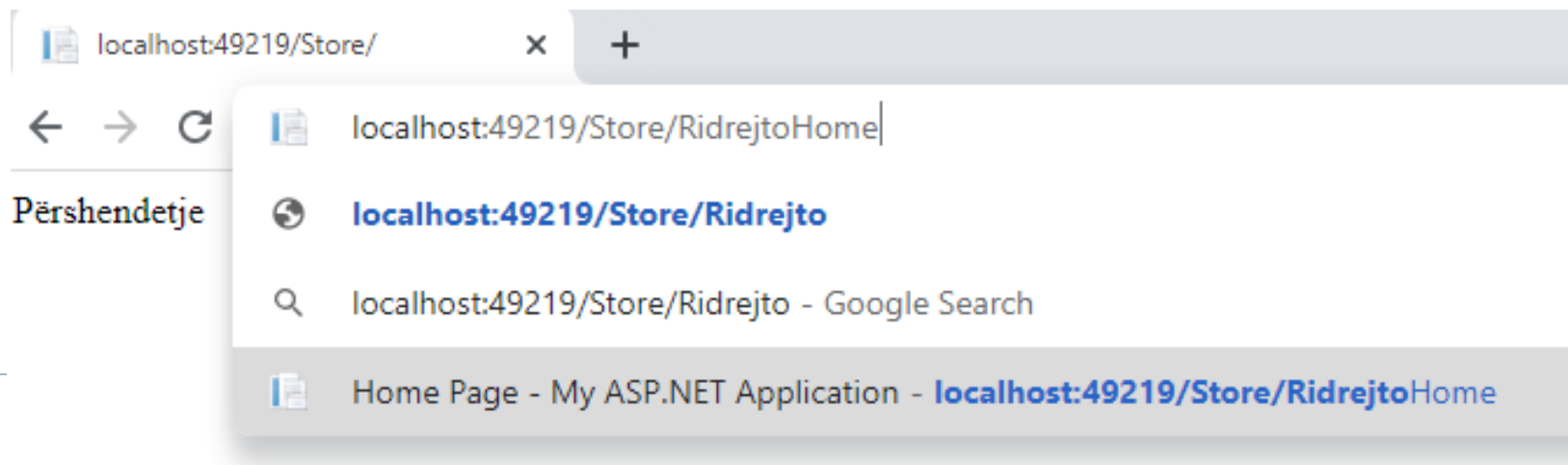
► Shembull 2 Ridrejtimi në faqe brenda app/ faqe të jashtme

Ridrejtimi në faqen Home/Index

```
public ActionResult RidrejtoHome()  
{  
    return Redirect("/Home/Index");  
}
```

```
public ActionResult RidrejtoFSHN()  
{  
    return  
    Redirect("http://www.fshn.edu.al");  
}
```

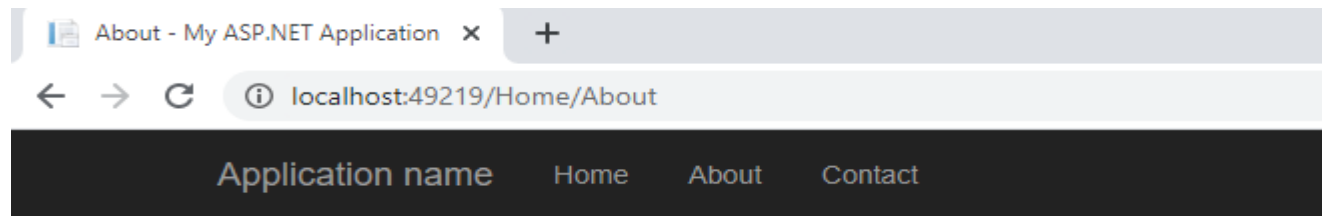
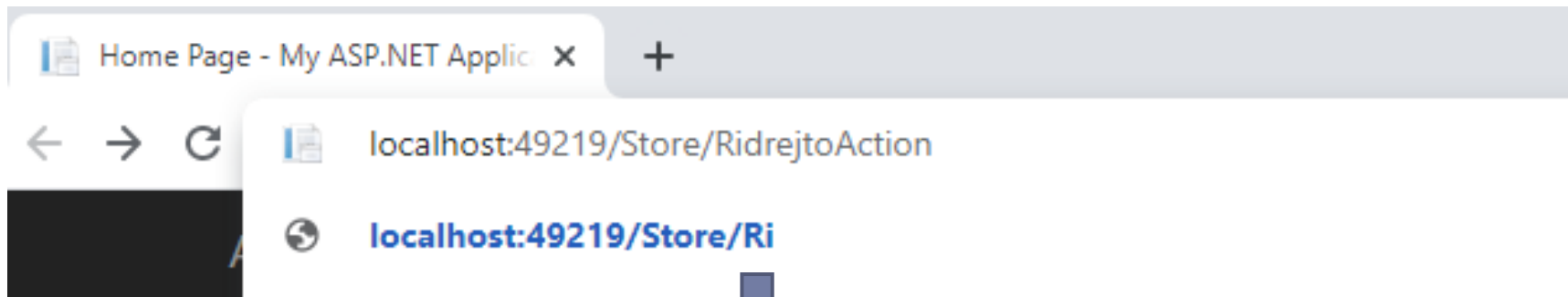
Në browser tek url pas Store/ shtohet emri i metodës Store/**RidrejtoHome**, si në Figurë, pasi të shtypni enter do të ridrejtoheni në faqen kryesore



Shembuj

Shembull 3 Ridrejtimi në action About të controllit Home

```
▶ public ActionResult RidrejtoAction()  
▶ {  
▶     return RedirectToAction("About", "Home");  
▶ }
```



About.

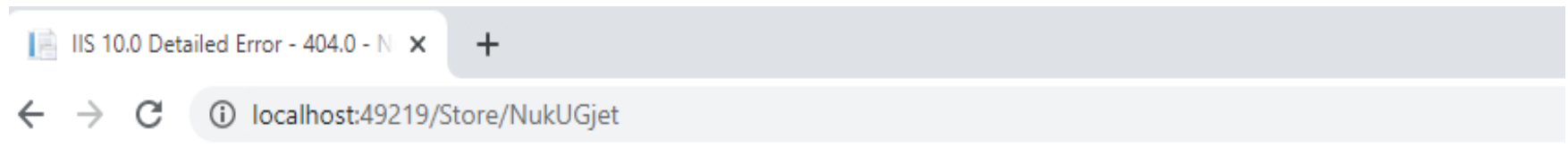
Pershendetje studente, kjo eshte faqja rreth nesh!

Use this area to provide additional information.

Shembuj

Shembull 4 Kthimi i faqes 404 Not found

```
▶ public ActionResult NukUGjet()  
▶ {  
▶     return HttpNotFound();  
▶ }
```



HTTP Error 404.0 - Not Found

The resource you are looking for has been removed, had its name changed, or is temporarily unavailable.

Most likely causes:

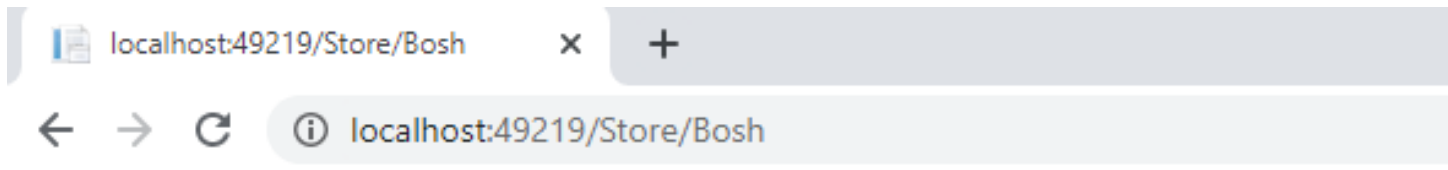
- The directory or file specified does not exist on the Web server.
- The URL contains a typographical error.
- A custom filter or module, such as URLScan, restricts access to the file.

Things you can try:

- Create the content on the Web server.
- Review the browser URL.

Shembuj

- ▶ Shembull 5 Kthimi i input bosh
- ▶ *public ActionResult Bosh() {*
- ▶ *return new EmptyResult();*
- ▶ *}*



Diskutim rreth Controller

- ▶ ➤ Kur kërkohet Store/Bosh në URL, kjo shkaktoi ekzekutimin e metodës Bosh të klasës së controller Store pa konfigurime të tjera.
- ▶ ➤ Klasa e Controller e krijuar nga Visual Studio identifikohet që është controller sepse trashëgon nga `System.Web.Mvc.Controller`.
- ▶ ➤ Megjithëse modelet dhe view janë shumë të rëndësishme në ASP.NET, ka raste që përdorimi i tyre edhe mund të mos jetë i nevojshëm sepse kërkesat kalojnë përmes controlles



Parametrat në metodat e Controller

- ▶ Parametrat mund të kalohen në disa mënyra në metodën e një Controller:
- 1. Në query string /Store/Browse?Genre=House
- 2. Në URL /Store/Details/2
- 3. Sipas të dhënave të formës id=2

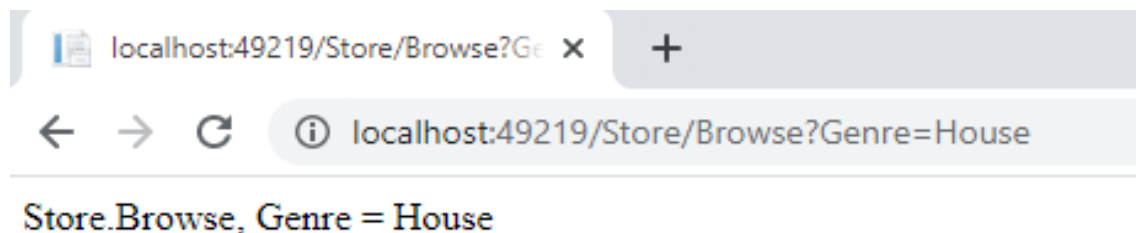


Parametrat në metodat e Controller

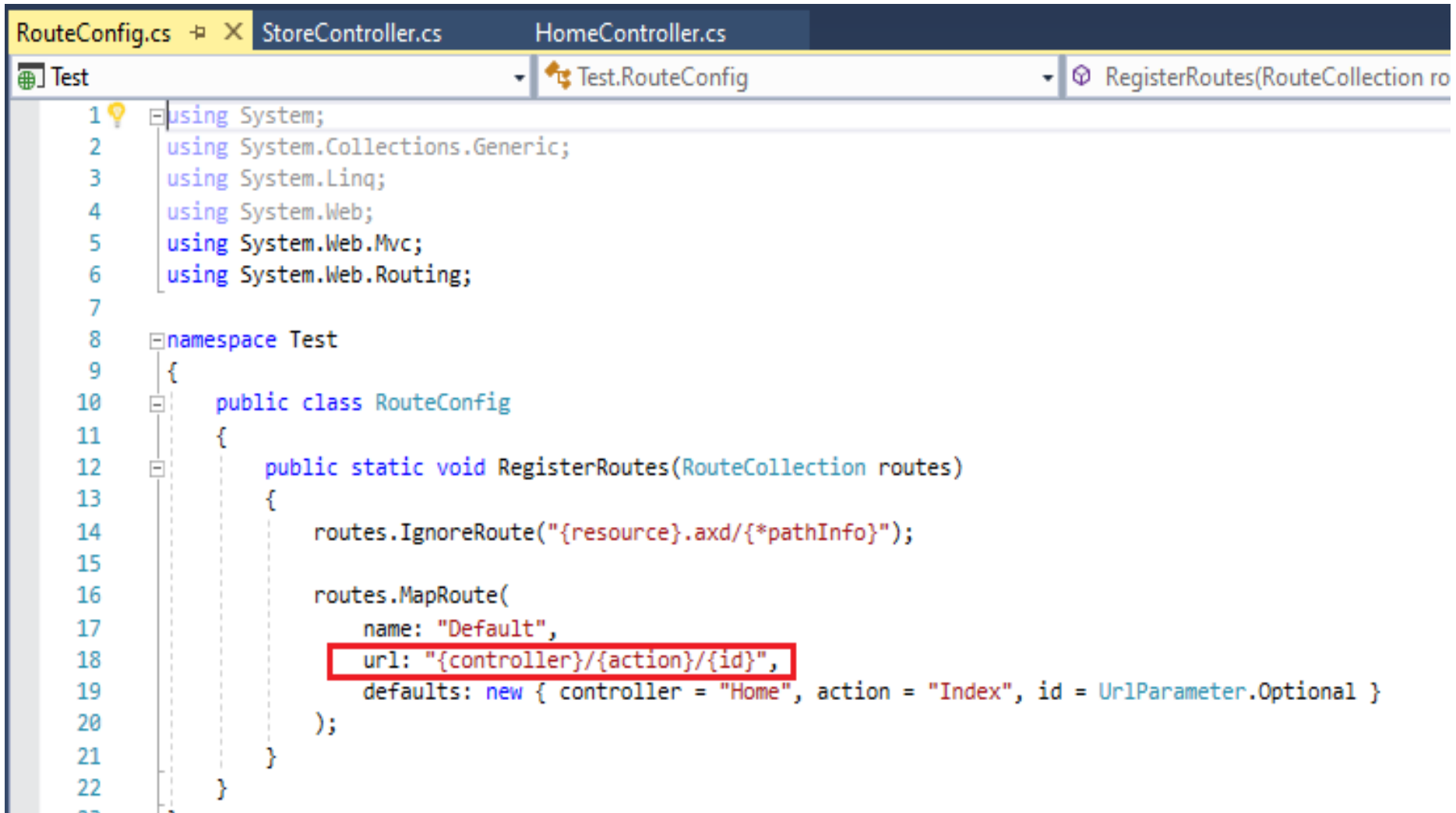
Metodat mund të bëhen edhe më dinamike duke shtuar parametra.

Krijoni metodën Browse që kërkon sipas zhanerit të muzikës si më poshtë. ASP.NET do ti kalonte në mënyrë automatike çdo query string ose parametra post të formës me emrin zhaneri metodës që thërritet. Pra nëpërmjet kësaj metode mund të aksesohet parametri në query string.

- ▶ *public string Browse(string genre)*
- ▶ {
- ▶ *string message = HttpUtility.HtmlEncode("Store.Browse, Genre = " + genre);*
- ▶ *return message;*
- ▶ }



A mund të kalojmë zhanerin si URL?



```
RouteConfig.cs X StoreController.cs HomeController.cs
Test Test.RouteConfig RegisterRoutes(RouteCollection ro
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6 using System.Web.Routing;
7
8 namespace Test
9 {
10     public class RouteConfig
11     {
12         public static void RegisterRoutes(RouteCollection routes)
13         {
14             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
15
16             routes.MapRoute(
17                 name: "Default",
18                 url: "{controller}/{action}/{id}",
19                 defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
20             );
21         }
22     }
23 }
```

Parametrat në metodat e Controller

Krijoni metodën `Details` që lexon dhe shfaq një parametër input të quajtur `ID`. Ndryshe nga zhaneri `ID` nuk do të kalojë si parametër query string. Ajo do të përfshihet vetë në URL.

Sipas → `/Store/Details/5`.

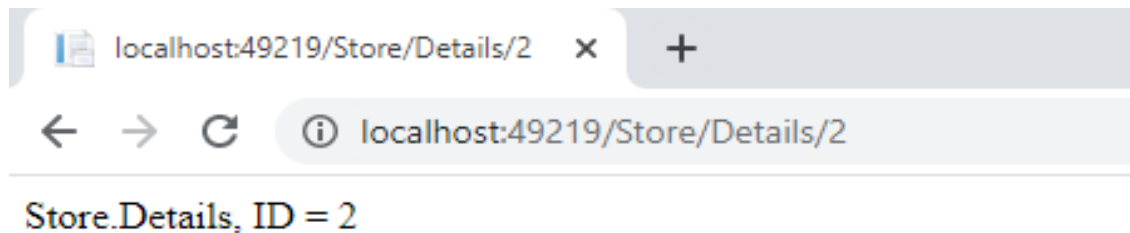
ASP.NET MVC ju lejon të realizoni këtë gjë pa konfiguruar asgjë. Konvencionet e routing të ASP.NET MVC's janë që të trajtojnë një segment të URL pas emrit të metodës së action si parameter të quajtur `ID`. Nëse metoda juaj action ka një parametër të quajtur `ID`, atëherë ASP.NET MVC automatikisht e kalon këtë segment URL si parametër.

```
public string Details(int id)
```

- ▶ {
- ▶ *string message = "Store.Details, ID = " + id;*
- ▶ *return message;*
- ▶ }



Rezultati



Siç tregojnë edhe shembujt e mësipërm, actions të një controller do të funksiononin njësoj sikur web browser të thërriste metodat në mënyrë direkte në klasën tuaj të controller.

Klasa, metoda dhe parametrat specifikohen të gjitha si segmente path ose si query stringje në URL, rezultati është një string që kthehet në browser.

Kjo është një lehtësi e madhe sepse injorohet fakti se si routing bashkengjit URL me actions , se si pothuajse gjithnjë përdoren view si template për të gjeneruar stringje zakonisht HTML dhe fakti se si të kthehet ActionResult që duhet në varësi të rastit.

Përfundime

- ▶ Kontrollerat ofrojnë shumë mundësi për personalizime dhe zgjerime por në përgjithësi ato thërren përmes URL, ekzekutojnë kodin dhe kthejnë pamjen.



Pamjet (View)

- ▶ Shumica e actions të controllers, kanë synimin e afishimit të informacionit dinamik në format HTML dhe jo thjeshtë afishimin e stringjeve.
 - ▶ Për këtë nevojitet një sistem templatesh që përfaqësohet nëpërmjet pamjeve. Pamjet (View) janë përgjegjëse për të siguruar ndërfajen përdoruesit. Pasi controller ka ekzekutuar logjikën që duhet për URL e kërkuar, ia delegon shfaqjen view.
 - ▶ View nuk mund të aksesohen në mënyrë direkte,, ajo prezantohet gjithnjë nga controller, që paraqet edhe të dhënat që view do të shfaqë.
 - ▶ Në disa raste të thjeshta view kërkon pak ose aspak informacion nga controller. Shpesh, controller duhet ti sigurojë informacion pamjes për këtë i kalon një objekt të dhënash të quajtur model.
-



Shembull View

Pamja më poshtë ndodhet në Views/Home/Index.cshtml

- ▶ @{
- ▶ ViewBag.Title = "Home Page";
- ▶ }
- ▶ <div class="jumbotron">
- ▶ <h1>ASP.NET</h1>
- ▶ <p class="lead">ASP.NET is a free web framework for building
- ▶ great Web sites and Web applications using HTML,
- ▶ CSS and JavaScript.</p>
- ▶ <p>
- ▶ Learn more »</p>
- ▶ </div>
- ▶ <div class="row">
- ▶ <div class="col-md-4">
- ▶ <h2>Getting started</h2>
- ▶ <p>

ViewPager

- ▶ ViewPager është një atribut që konsiderohet si objekt dinamik dhe ju lejon që të ndani të dhëna në mënyrë dinamike në aplikacion midis kontroller dhe View. ViewPager përdoret për të ruajtur një sasi të vogël të dhënash, si karta blerje, opsione liste etj, përdoret në rastet kur të dhënat nuk kërkojnë domosdoshmërisht implikimin e një modeli.



View

Ndërkohë që në controller home action Index do të kthente këtë View

```
▶ public ActionResult Index() {  
▶ return View();  
▶ }
```

Nëse do të shkonit tek rrënja e site metoda Index e HomeController do të shfate View përkatëse të faqes kryesore. Që në këtë rast do të përmban thjesht HTML statike, por situata nuk është gjithnjë kështu. Për ta bërë pak më dinamike situatën mund të kalojmë të dhëna nga controller drejt View. Kjo mund të bëhet duke përdorur ViewBag, tek HomeController vendosni kodin:

```
▶ public ActionResult About()  
▶ {  
▶ ViewBag.Message = "Your application description page.";  
▶ return View();  
▶ }
```



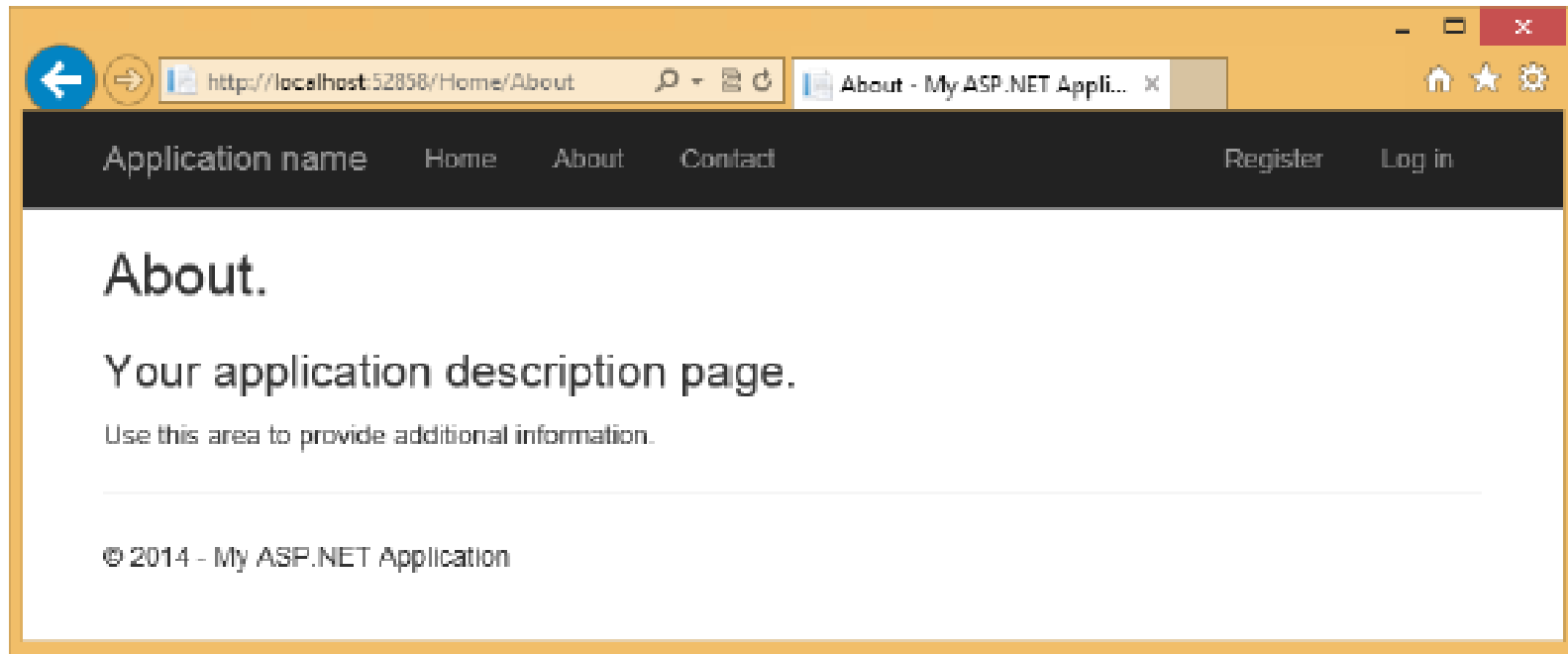
Home About view—About.cshtml

- ▶ @{
- ▶ ViewBag.Title = "About";
- ▶ }
- ▶ <h2>@ViewBag.Title.</h2>
- ▶ <h3>@ViewBag.Message</h3>
- ▶ <p>Use this area to provide additional information.</p>

Në kodin më sipër vendoset titulli i faqes në ViewBag.Title, dhe afishohen edhe vlerat e ViewBag.Title dhe ViewBag.Message. Simboli @ përpara vlerave ViewBag është pjesa më e rëndësishme e sintaksës Razor syntax, sepse i tregon Razor engine që ky është kod dhe jo text HTML.



Afishimi



Konvencionet e View

- ▶ Actions të kontrollerave që kemi parë deri tani thjesht thërrisin metodën `View()` për të shfaqur pamjen, nuk e specifikonin emrin e file të View. Kjo sepse përfitohet nga konvencionet e ASP.NET MVC Framework, që përcaktojnë llogjikën e përzgjedhjes së View.
- ▶ Cdo folder 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/`.

