TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tarkvarateaduse instituut

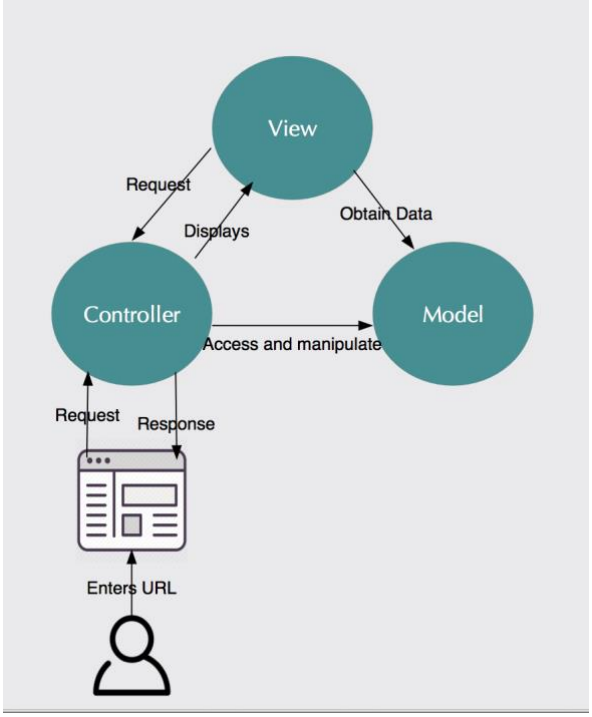
ASP .NET MVC CORE
JUHEND

Tallinn 2018

SISUKORD

Labor 1- Kontrolleri näiteprojekt

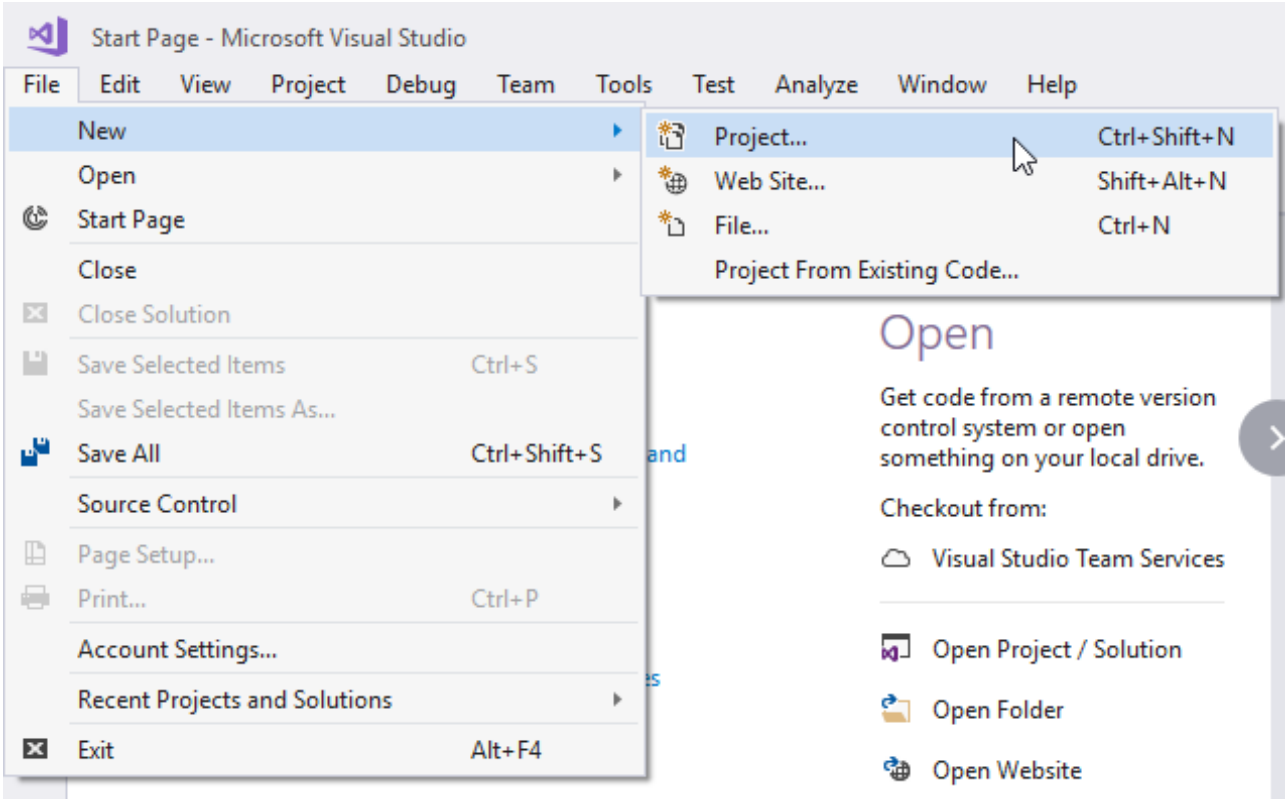
MVC lahutab rakenduse kolmeks erinevaks osaks- mudel(Model), vaade(View) ja kontroller(Controller). Mudel- esindab andme ja äriloogika kihti. Mudeli objektid tagastavad ja säilitavad mudeli oleku andmebaasis. Vaade- kasutajaliidese kiht. Vaade esitab andmeid kasutajale kasutades selleks mudelit. Kontroller- Kontroller käsitleb kasutaja päringut. Tüüpiliselt suhtleb kasutaja vaatega, mis tekitab vastava URL päringu, selle päringuga tegeleb edasi kontroller. Kontroller tagastab vastusena õige vaate vastavate andmetega. Kasutaja suhtlemisel serveriga on vaja kontrollerit, et tegeleda kasutaja päringuga ja sellele vastuse saatmisega.



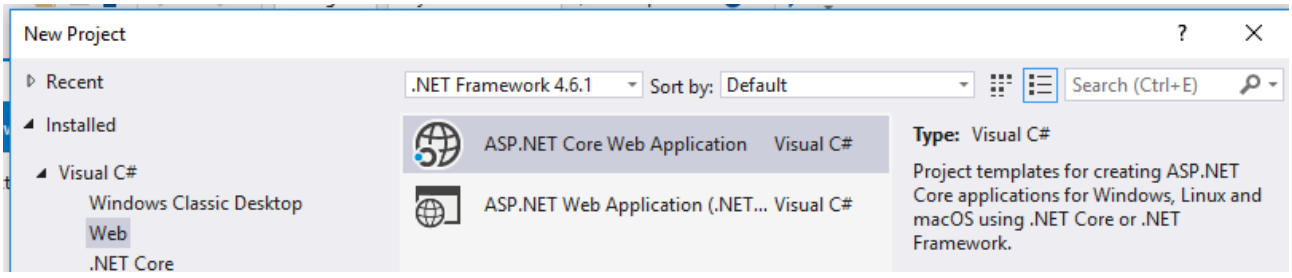
Pilt 1. MVC arhitektuur. Adapteeritud http://www.tutorialsteacher.com/mvc/mvc-architecture

1. Loo ASP.NET MVC Core projekt

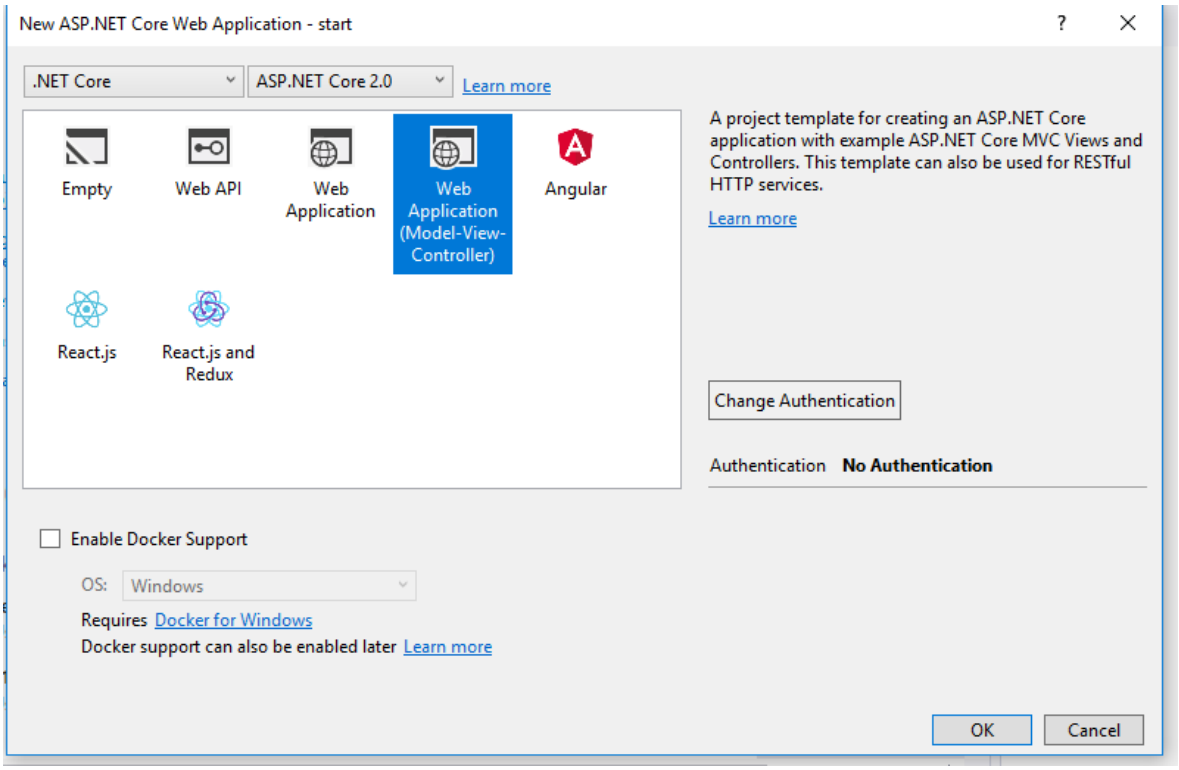
1.1. Ava Visual Studio. Vali File -> New -> Project



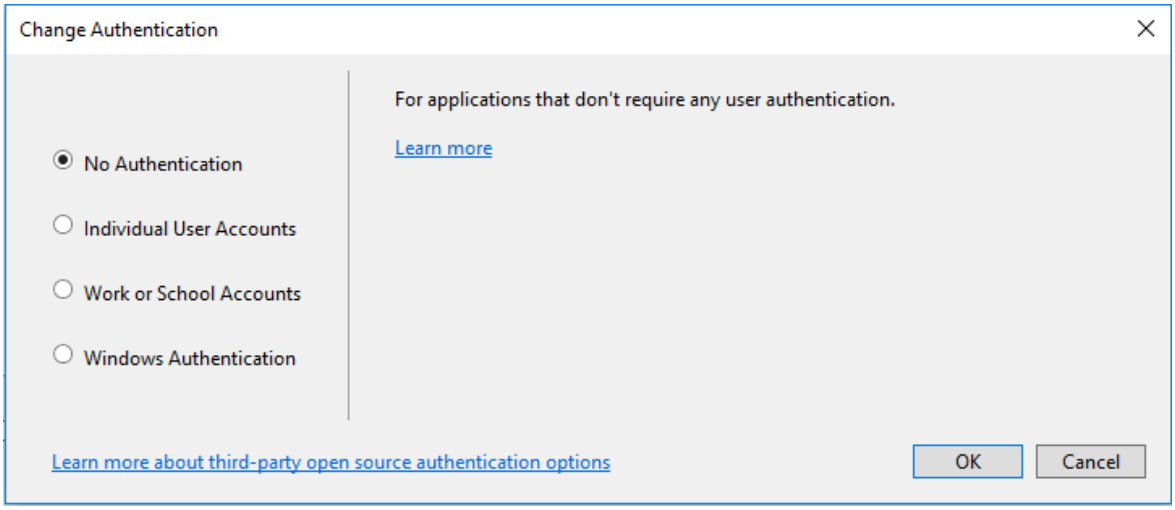
1.2. Vali ASP.NET Core Web Application. Määra projektile nimi ja salvestamise asukoht, seejärel vajuta ‘ok’.



1.3. Vali MVC mall, seejärel vajuta ‘Change Authentication’.



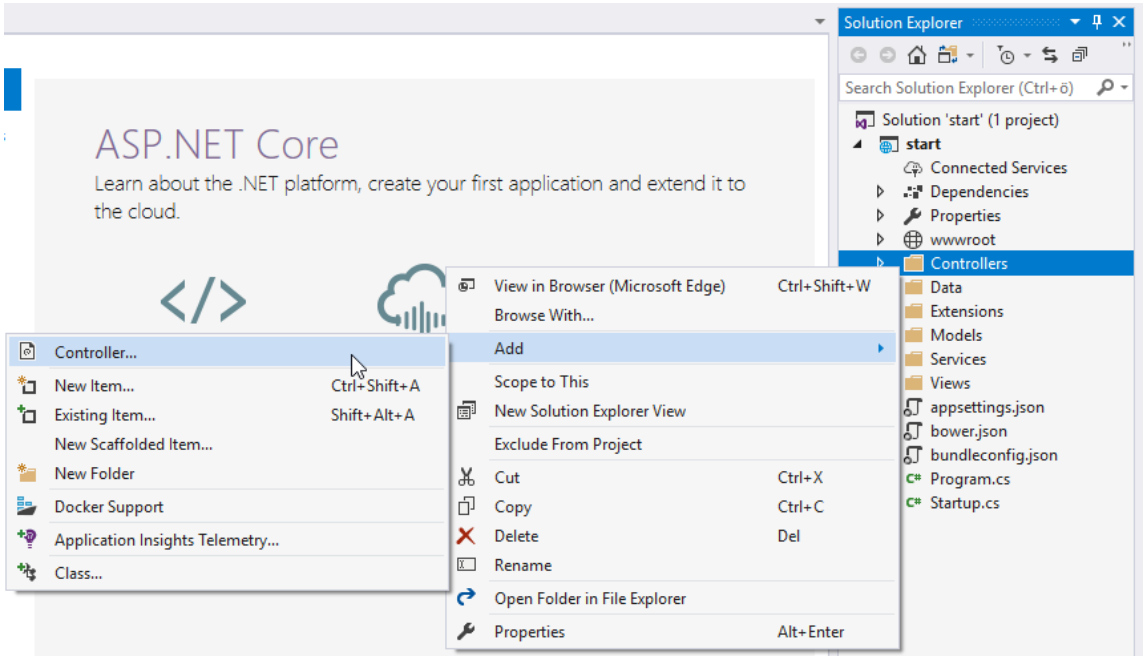
1.4. Vali ‘No Authentication’ vasakul olevast valikust, seejärel vajuta ‘ok’.



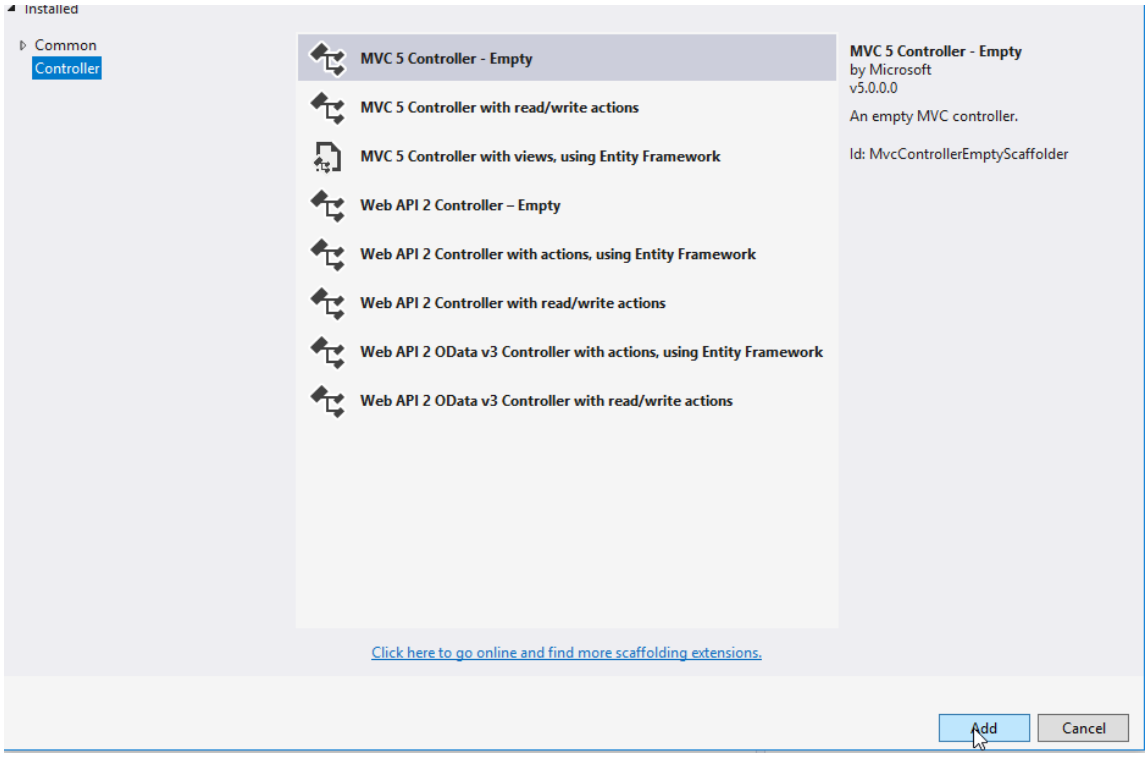
1.5. Vajuta ‘ok’.

2. Lisa uus kontroller

Tee lahenduse aknas (Solution explorer) parem klikk kontrolleri kaustal (Controllers) ja vali Add -> Controller

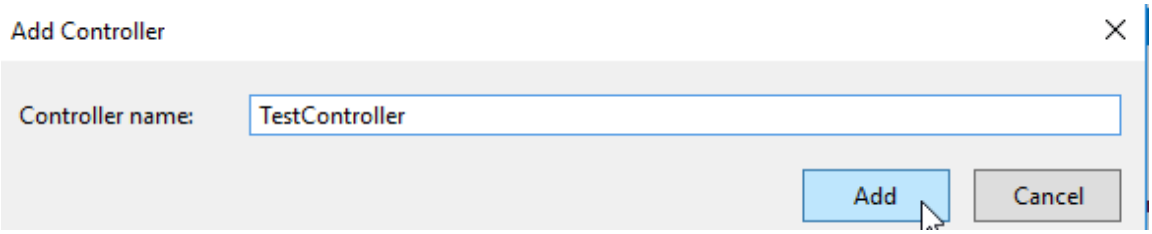


2.1. Vali MVC 5 Controller – Empty ja vajuta ‘Add’.



2.2. Pane kontrollerile nimeks ‘TestController’* ja valuta ‘Add’.

*Väga oluline selle sammu juures on see, et **ei kustutaks** võtmesõna ’Controller’



3. Loo '*action*' meetod.

3.1. Ava loodud 'TestController'-i klass. Seal on vaikimisi meetod nimega 'Index'. Eemalda see meetod ja lisa uus meetod, mille nimeks on 'GetString' ja mis on järgmine:

```
namespace Labor1.Controllers
{
    public class TestController : Controller
    {
        public string GetString()
        {
            return "Hello world!";
        }
    }
}
```

Action meetod on tavapärane avalik meetod, mis on järgmiste omadustega:

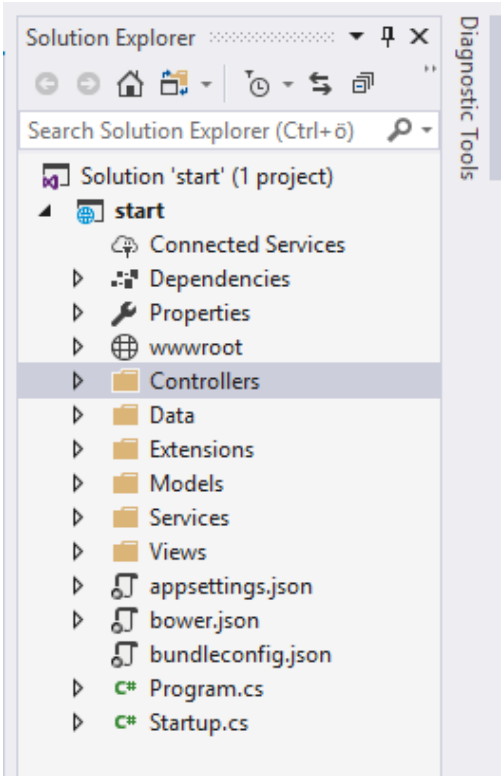
- On alati avalik
- Ei saa olla ülekoormatud (*overloaded*)
- Ei saa olla staatiline

4. Käivita kood ja testi tulemust. Koodi käivitamiseks vajuta klahvi F5. Aadressi ribale kirjuta pordi järele /Kontrolleri nimi/Meetodi nimi.

TestController vs Test controller

TestController on klassi nimi, samas kui Test on kontrolleri nimi. Aadressiribale lisatakse alati kontrolleri nimi ilma sõnata 'Controller'

Visual Studio loob vaikimisi järgmise failide struktuuri:



Pilt 2. MVC failide struktuur

Iga MVC projekti puhul on failide paigutus kaustadesse väga oluline.

Kataloog Controllers- sisaldab kontrollerite klassi faile. Kontrollerite nimed peavad alati lõppema sõnaga “Controller”.

Kataloog Models- sisaldab mudeli klassi faile. Tüüpiliselt sisaldab mudeli klass avalikke omadusi(public properties), mida kasutavad rakendused hoidmaks ja käsitlemaks rakenduse andmeid.

Kataloog Views- sisaldab faile, mille abil genereeritakse HTML vormingus kasutajale kuvatavad vaated. Tüüpiliselt on vaade .cshtml laiendiga fail, kuhu saab kirjutada nii HTML koodi kui ka C# koodi. Vaadete kaust sisaldab iga kontrolleri jaoks eraldi kausta. Näiteks kõik vaated, mida käsitletakse HomeController-i poolt peaksid asuma kaustas View/Home. Shared kaust Views kaustas sisaldab vaateid, mis on ligipääsetavad kõikidele kontrolleritele.

Labor 2- Vaadete demonstreerimine

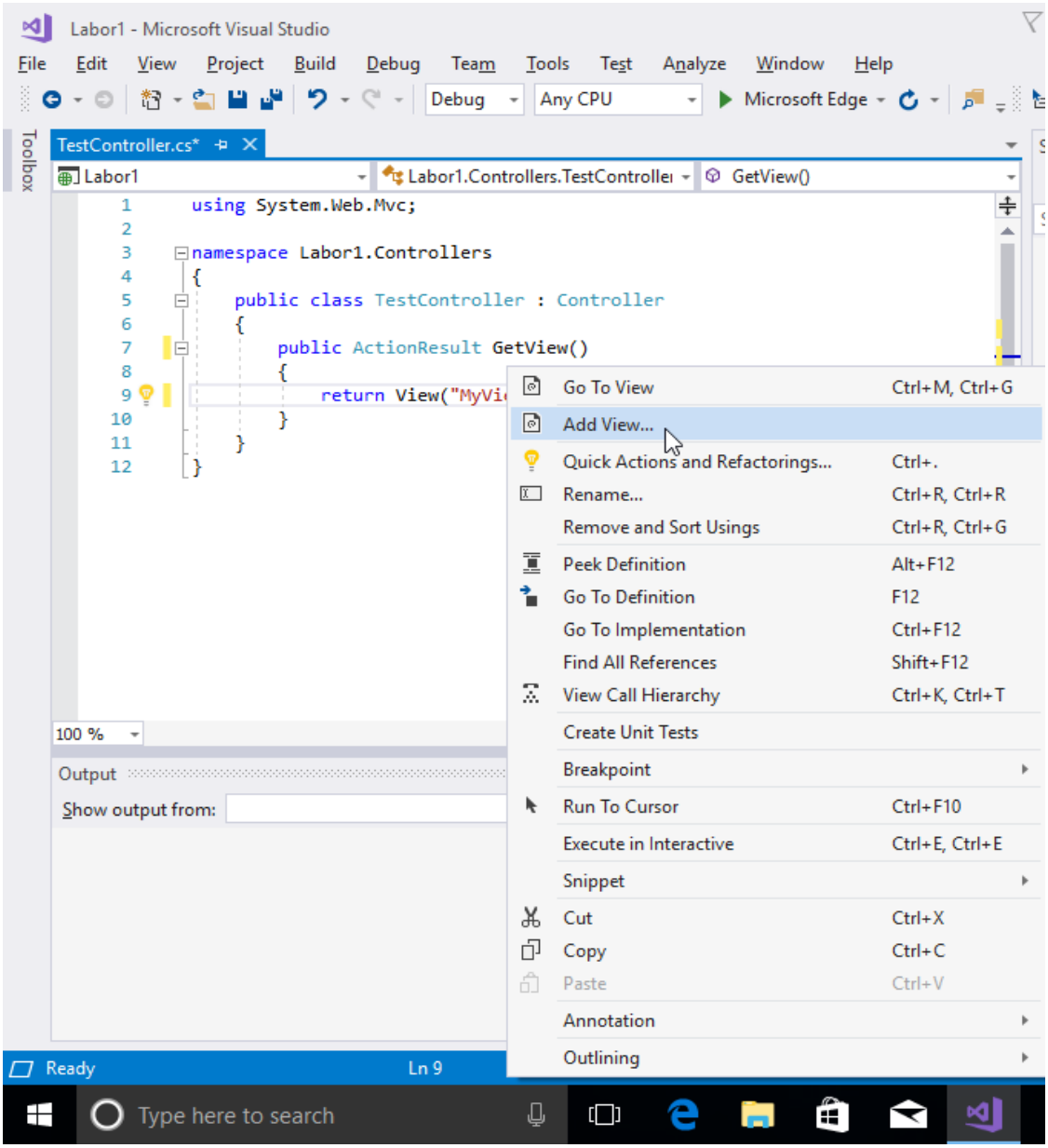
Tavaliselt on vaade HTML formaadis, kuna brauser mõistab seda formaati kõige paremini. Kasutajale kuvatavat kihti nimetatakse tehnilise terminiga UI ehk User Interface, Asp.Net MVC projektis kutsutakse seda kihti vaateks(view). Vaade kuvab kasutajale seda, mida kasutaja näeb brauseris.

1. Loo uus action meetod Test kontrolleri klassi sisse järgevalt:

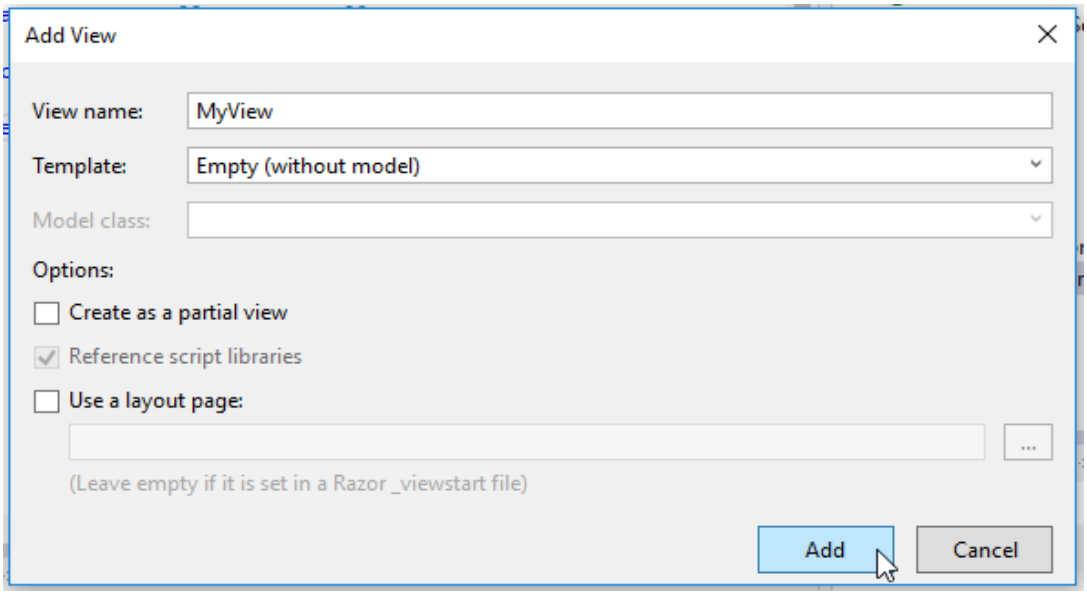
```
public ActionResult GetView()
{
    return View("MyView");
}
```

2. Loo vaade

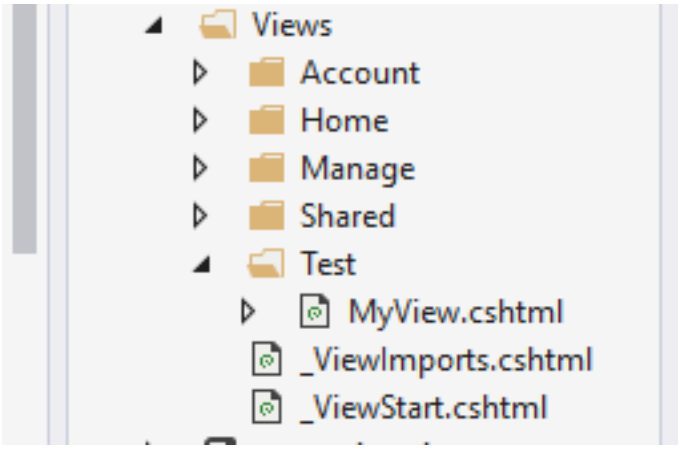
   2.1 Tee hiire parema sõrmise klikk action meetodil ja vali “Add View”



2.2 Avanenud aknas määra vaate nimeks “MyView”, seejärel eemalda linnuke “Use a layout” ees olevast kastist ja vajuta “Add”.



2.3 Seejärel tekib loodud vaade kausta “Views/Test” projekti lahenduse aknas (Solution Explorer)



Iga vaade on seotud kindla kontrolleriga ja peab seetõttu olema asetatud vastava kontrolleri nimega kausta (nt /Views/Test).

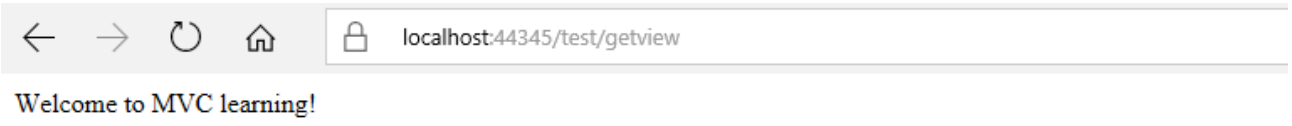
Kui soovid kasutada ühte vaadet mitme kontrolleri jaoks, tuleb see paigutada jagatud kausta (/Views/Shared)

3 Lisa loodud vaatele sisu

MyView.cshtml TestController.cs

```
1
2   @{
3       Layout = null;
4   }
5
6   <!DOCTYPE html>
7
8   <html>
9   <head>
10      <meta name="viewport" content="width=device-width" />
11      <title>MyView</title>
12  </head>
13  <body>
14      Welcome to MVC learning!
15  </body>
16  </html>
```

4 Testi loodud vaadet. Vajuta klahvi F5, mis käivitab rakenduse



Welcome to MVC learning!

Loe lisaks: https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-2.1

Labor 3 - ViewData kasutamine

ViewData on sõnastik, mis sisaldab andmeid, mida kontroller ja vaade vahetavad. Kontroller lisab andmed sõnastikku ja vaade loeb neid sealt.

1. Loo uus klass nimega ‘Employee’ kausta ‘Models’, mis näeb välja järgnevalt:

```
namespace Lab3.Models
{
    public class Employee
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public int Salary { get; set; }
    }
}
```

2. Loo Employee objekt meetodi GetView (Controllers/TestController) sisse

```
public ActionResult GetView()
{
    Employee emp = new Employee();
    emp.FirstName = "Sukesh";
    emp.LastName = "Marla";
    emp.Salary = 20000;
```

3. Ära unusta lisamast viidet:

```
using Lab3.Models;
```

4. Salvesta Employee objekt ViewData sõnastikku ja tagasta vaade.

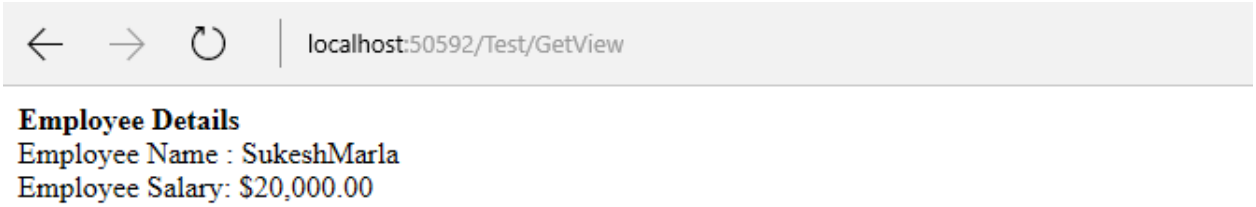
```
public ActionResult GetView()

{
    Employee emp = new Employee();
    emp.FirstName = "Sukesh";
    emp.LastName = "Marla";
    emp.Salary = 20000;

    ViewData["Employee"] = emp;
    return View("MyView");
}
```

5. Kuva Employee objekti andmed vaates. Selleks ava Views/Test/MyView.cshtml fail ja tagasta Employee objekti andmed ViewData sõnastikust ja kuva need.

```
1   @{
2       Layout = null;
3   }
4
5   <!DOCTYPE html>
6
7   <html>
8   <head>
9       <meta name="viewport" content="width=device-width"/>
10      <title>MyView</title>
11  </head>
12  <body>
13  <div>
14      @{
15          Labor.Models.Employee emp = (Labor.Models.Employee)
16              ViewData["Employee"];
17      }
18      <b>Employee Details</b><br />
19      Employee Name: @emp.FirstName@emp.LastName <br/>
20      Employee Salary: @emp.Salary.ToString("C")
21
22  </div>
23  </body>
24  </html>
25
```

Razor syntax võimaldab kirjutada vaates serveri poolset C# või VB koodi kombineeritult HTML-iga. Serveri poolse muutuja kuvamiseks tuleb kirjutada @muutuja_nimi (nt kuupäeva ja kellaaja kuvamiseks kirjuta @DateTime.Now)

6. Testi väljundit. Selleks vajuta F5 klahvi ja navigeeri vastavale URL-ile.



Labor 4 - ViewBag-i kasutamine

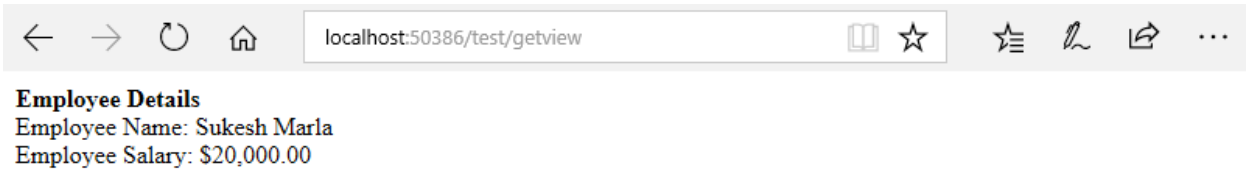
1. Loo View Bag, selleks asenda GetView meetodis ViewData ViewBag-iga järgmiselt:

```
public ActionResult GetView()
{
    Employee emp = new Employee();
    emp.FirstName = "Sukesh";
    emp.LastName = "Marla";
    emp.Salary = 20000;

    ViewBag.Employee = emp;
    return View("MyView");
}
```

2. Kuva employee andmed vaates, selleks muuda vaadet järgmiselt:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>MyView</title>
</head>
<body>
@{
    var emp = (Employee)
        ViewBag.Employee;
}
<b>Employee Details</b><br />
Employee Name: @emp.FirstName @emp.LastName <br />
Employee Salary: @emp.Salary.ToString("C")
</body>
</html>
```

3. Testi väljundit. Selleks vajuta F5 klahvi ja navigeeri vastavale URL-ile.



ViewBag-I kasutatakse ajutiste andmete edastamiseks kontrollerist vaatesse. ViewBag-ile saab omistada mitmeid omadusi ja väärtuseid, kuid kui omistada sama nimega omadust korduvalt, arvestatakse vaid viimasena omistatud väärtust. Kuna ViewBag ja ViewData kasutavad sisemiselt sama sõnastikku, ei tohi nende võtmeväärtused olla kattuvad, see annab Runtime erandi.

Näiteks ei saa koos kasutada järgnevat võtmeväärtust Id:

```
ViewBag.Id = 1;
```

```
ViewData.Add("Id", 1);
```

Labor 5 - Kindla tüübiga vaated *(Strongly typed view)*

1. Muuda vaade kindlat tüüpi, selleks lisa järgmine annotatsioon vaatele kõige esimesele reale

```
@model Employee
```

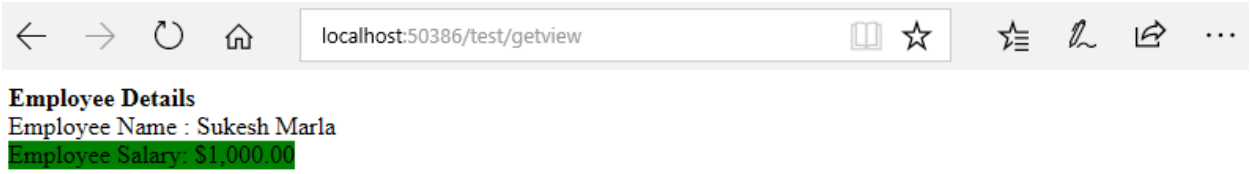
2. Kuva objekti andmed kasutades objektile viitamiseks @Model, kui seejärel kirjutada punkt pakub Visual Studio valikut kõikidest Employee klassi omadustest.

```
@model Employee
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>MyView</title>
</head>
<body>
<div>
    <b>Employee Details</b><br />
    Employee Name : @Model.FirstName @Model.LastName <br />
    @if (Model.Salary > 15000) {
        <span style="background-color:yellow">
            Employee Salary: @Model.Salary.ToString("C")
        </span>
    } else {
        <span style="background-color:green">
            Employee Salary: @Model.Salary.ToString("C")
        </span>
    }
</div>
</body>
</html>
```

3. Muuda GetView meetodit järgmiselt:

```
public ActionResult GetView()
{
    Employee emp = new Employee();
    emp.FirstName = "Sukesh";
    emp.LastName = "Marla";
    emp.Salary = 1000;
    return View("MyView", emp);
}
```

4. Testi väljundit. Selleks vajuta F5 klahvi ja navigeeri vastavale URL-ile.



Employee Details
Employee Name : Sukesh Marla
Employee Salary: $1,000.00

Võimalused andmete edastamiseks kontrollerist vaatesse:

- ViewBag või ViewData
- DynamicType
- Stongly Typed View

Kindla tüübiga vaate eelised:

1. Automaatne koodi genereerimine

   (https://msdn.microsoft.com/en-us/library/43f44291(v=vs.140).aspx)

2. Veakontroll kompileerimisel

Loe lisaks vaadetest:

https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview#creating-a-view

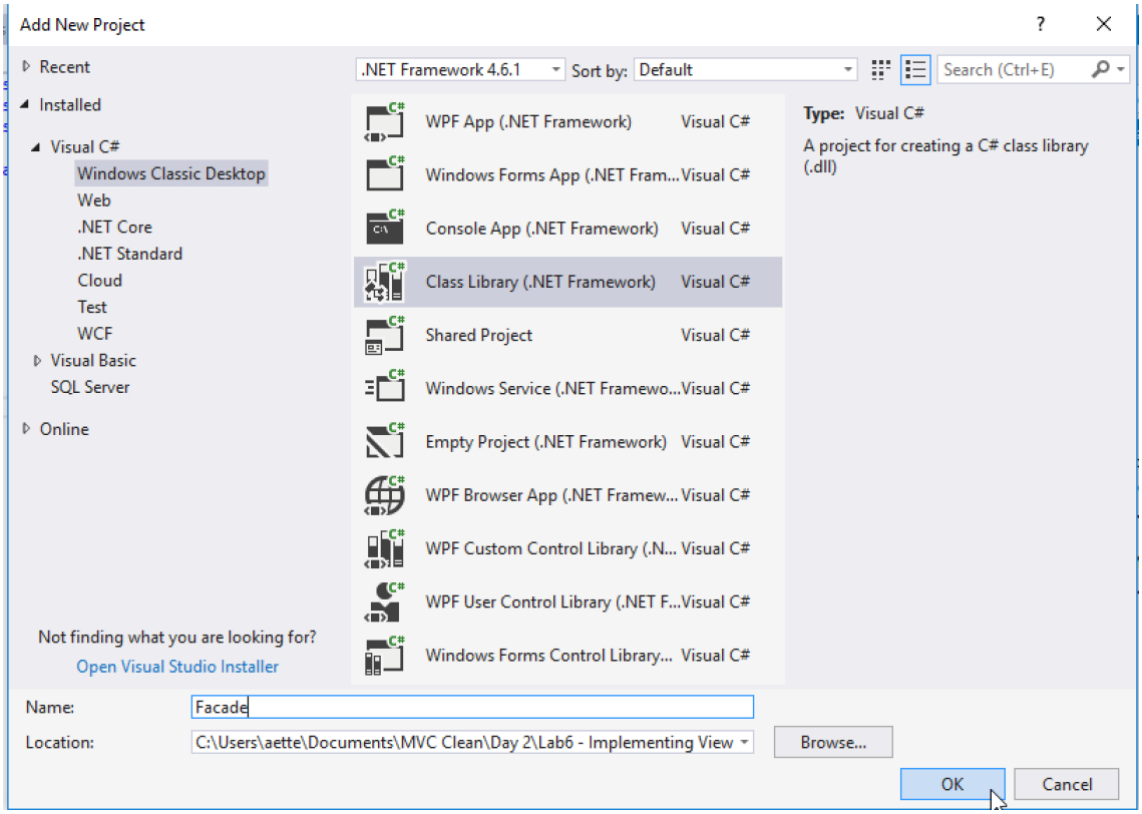
Labor 6 - ViewModel-i rakendamine

4. Korrastame projekti.

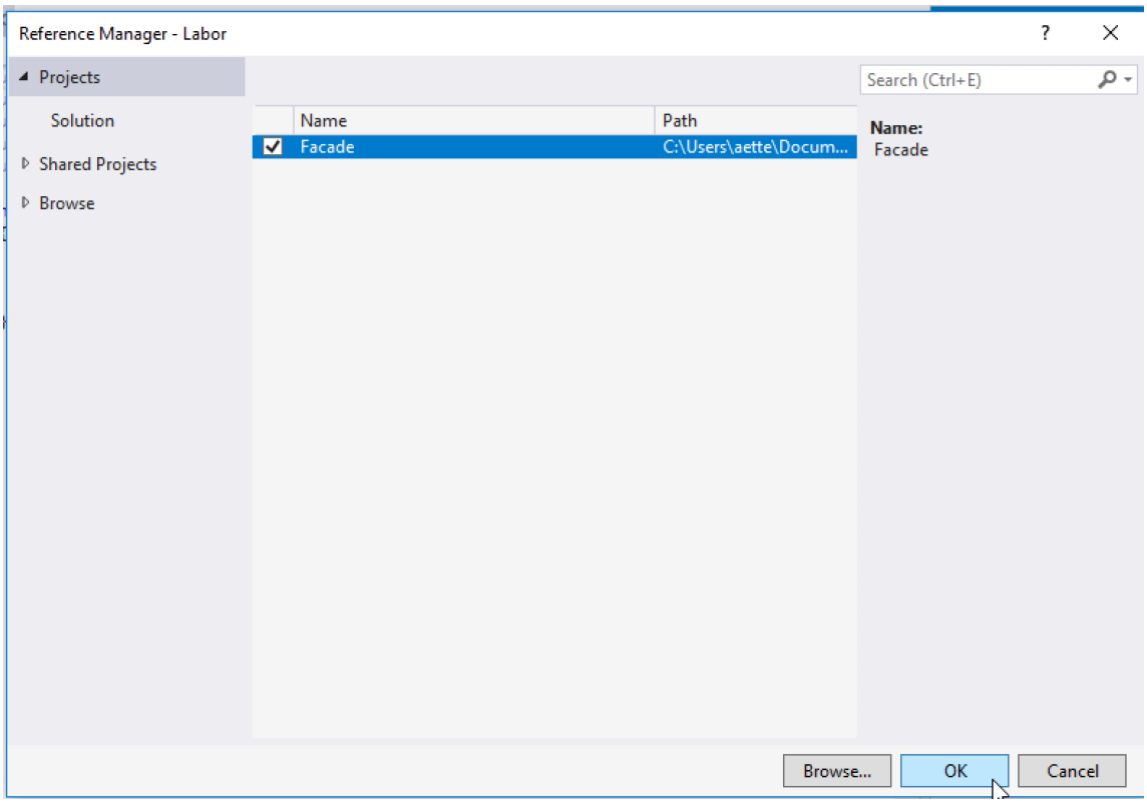
   a. Ava Controllers kaust ning kustuta seal AccountController ja ManageController. Kustuta ka Views kaustas paiknevad Account ja Manage kaustad.

   b. Kustuta Extensions ja Services kaustad.

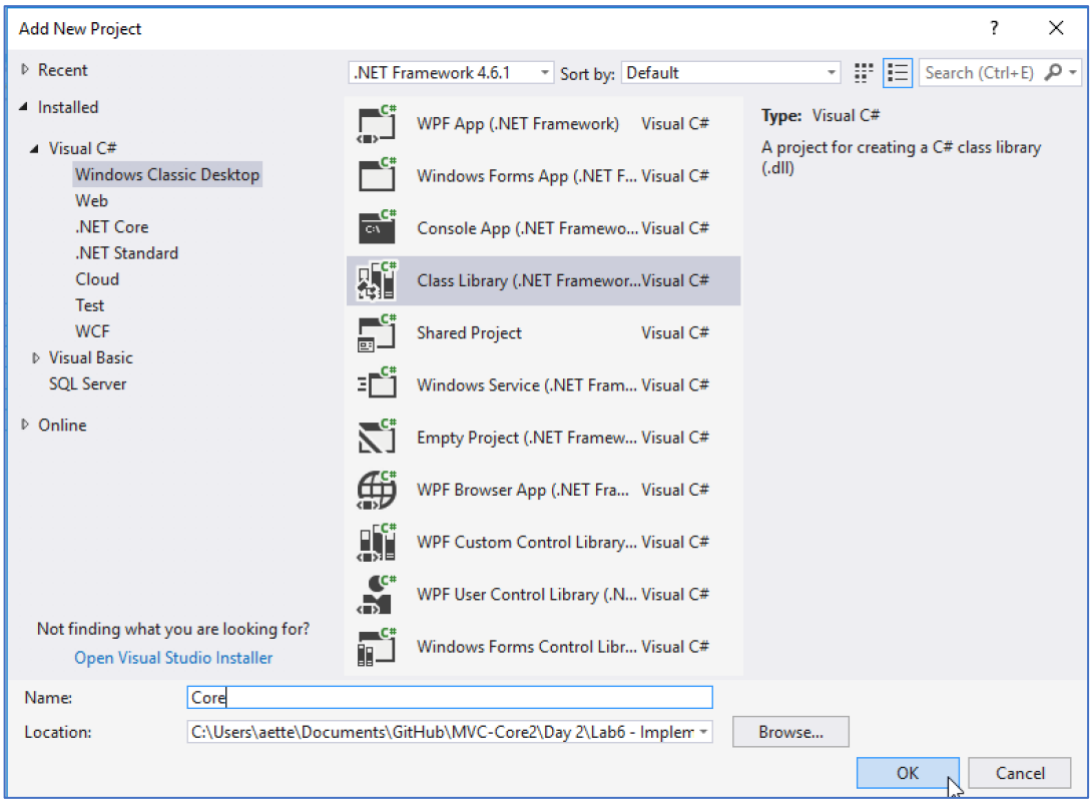
5. Vali Solution -> Add -> New Project. Avanenud aknas vali Windows Classic Desktop -> Class Library. Pane projektile nimeks Facade.



6. Lisa Labor projektis viide Facade projektile. Selleks tee hiire parema sõrmise klikk Labor projektil ning vali Add -> Reference. Avanenud aknas vali linnuke Facade projekti ees ning vajuta ok.

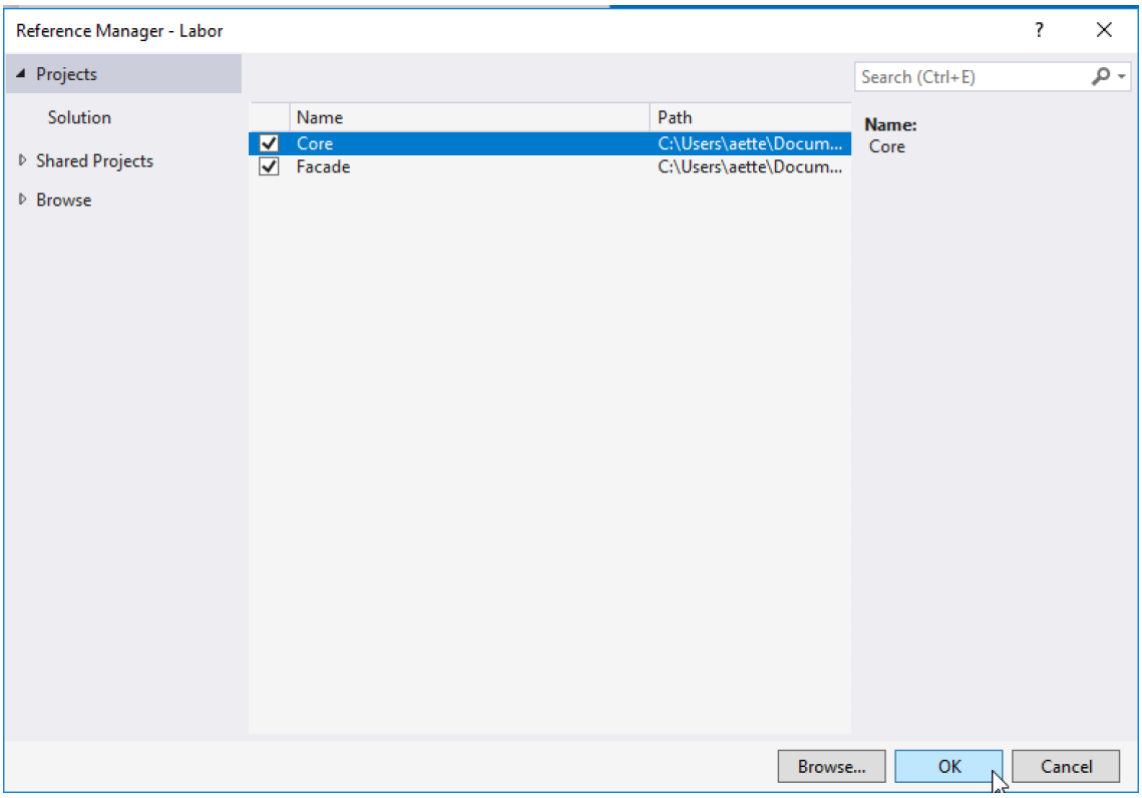


7. Loo uus projekt. Selleks tee hiire parema sõrmise klikk Solution-il ning vali Add New Project. Avanenud aknas vali vasakul asuvalt menüüribalt Windows Classic Desktop ning Class Library, pane projektile nimeks ’Core’.

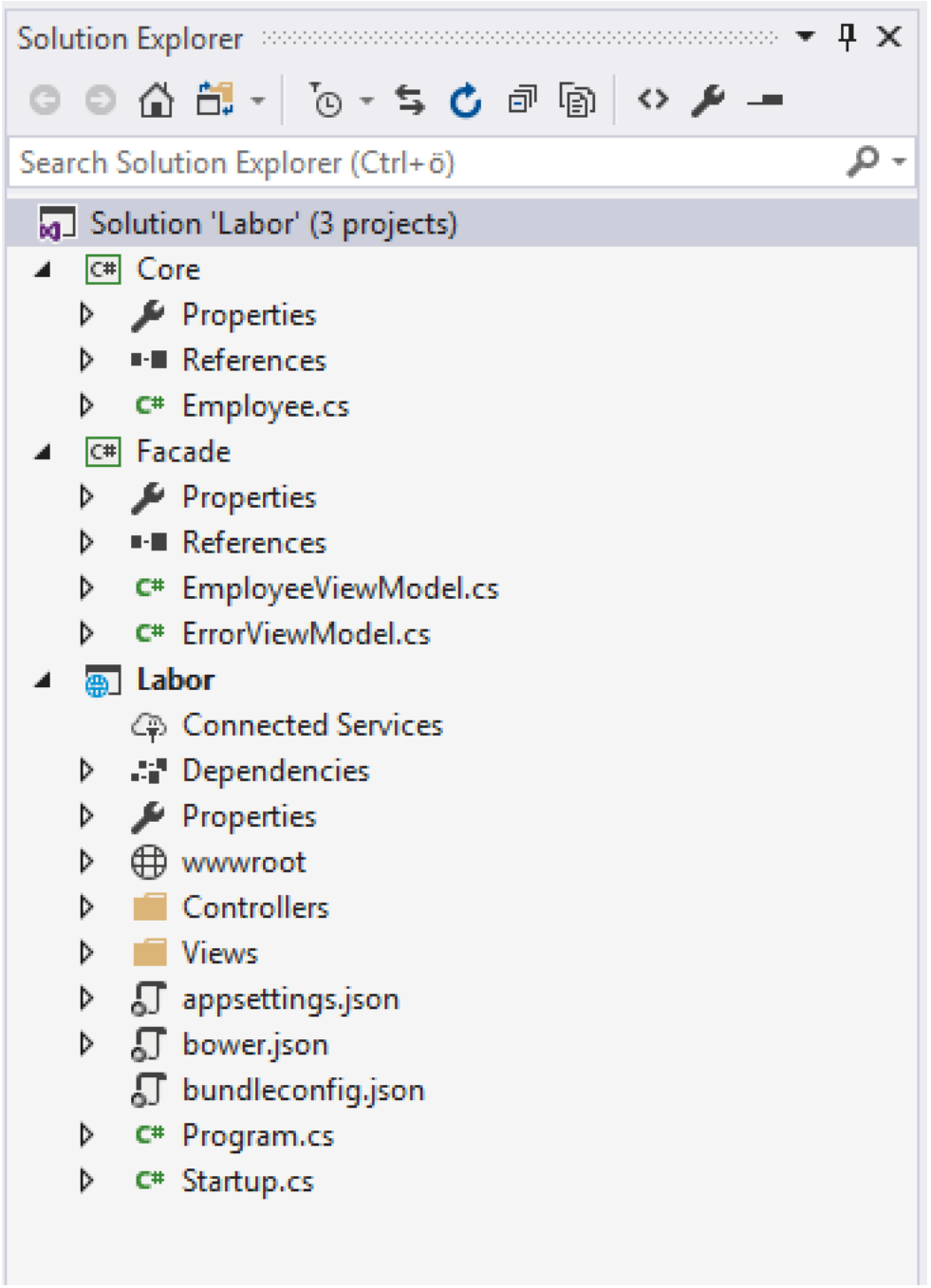


8. Lisa viide loodud projektidele. Selleks tee hiire parema sõrmise klikk Labor-il ning vali Add -> Reference. Avanenud aknas märgi linnuke Core ette ning vajuta ’OK’.

   Lisa samamoodi viide Facade projektis projektile Core.



9. Vii Labor projekti all olevast Models kaustas asuv Employee klass Core projekti alla ning samas kaustas asuv ErrorViewModel klass Facade projekti. Seejärel kustuta Models ja Data kaust. Nüüd peaks failide struktuur olema järgmine.



NB! Peale asukohtade muutmist kontrolli, et klassid asuksid õiges nimeruumis, nt Employee klass asus enne nimeruumis Labor.Models (nimeruum on peale klassi liigutamist sama). See tuleb muuta vastavalt sellele, kuhu nimeruumi klass liigutati, ehk Core. Kontrolli nimeruumi ka ErrorViewModeli puhul, see peaks olema Facade. hiire

```
namespace Core
{
    public class Employee
    {
```

10. Ava Core projektis asuv Employee klass ning muuda koodi järgnevalt

```
public class Employee
{
    public Employee()
    {
    }
    public Employee(string firstName,
        string lastName, int salary)
    {
        FirstName = firstName;
        LastName = lastName;
        Salary = salary;
    }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Salary { get; set; }
}
```

11. Loo uus klass nimega ‘EmployeeViewModel’ Facade projekti, mis näeb välja järgmine

```
public class EmployeeViewModel
{
    public EmployeeViewModel(Employee emp, string userName)
    {
        setName(emp);
        setSalary(emp);
        setColor(emp);
        setUserName(userName);
    }
    public string EmployeeName { get; set; }
    public string Salary { get; set; }
    public string SalaryColor { get; private set; } = "red";
    public string UserName { get; set; }

    internal void setName(Employee e)
    {
        EmployeeName = e.FirstName + " " + e.LastName;
    }
    internal void setColor(Employee e)
    {
        if (!ReferenceEquals(null, e))
            SalaryColor = e.Salary > 15000 ?
                "yellow" : "green";
        else SalaryColor = "red";
    }
    internal void setSalary(Employee e)
    {
        Salary = e.Salary.ToString("C");
    }
    internal void setUserName(string userName)
    {
        UserName = userName ?? string.Empty;
        ;
    }
}
```

12. Muuda Labor projektis asuvat TestController klassi. Märka, et ka kasutuslausetes tuleb muudatusi teha.

```
using Core;
using Facade;
using Microsoft.AspNetCore.Mvc;

namespace Labor.Controllers
{
    public class TestController : Controller
    {
        public ActionResult GetView()
        {
            var emp = new Employee("Sukesh", "Marla", 20000);

            var vmEmp = new EmployeeViewModel(emp, "Admin");
            return View("MyView", vmEmp);
        }
    }
}
```

13. Kuva andmed vaates, selleks asenda MyView.cshtml-is olev kood järgnevaga.

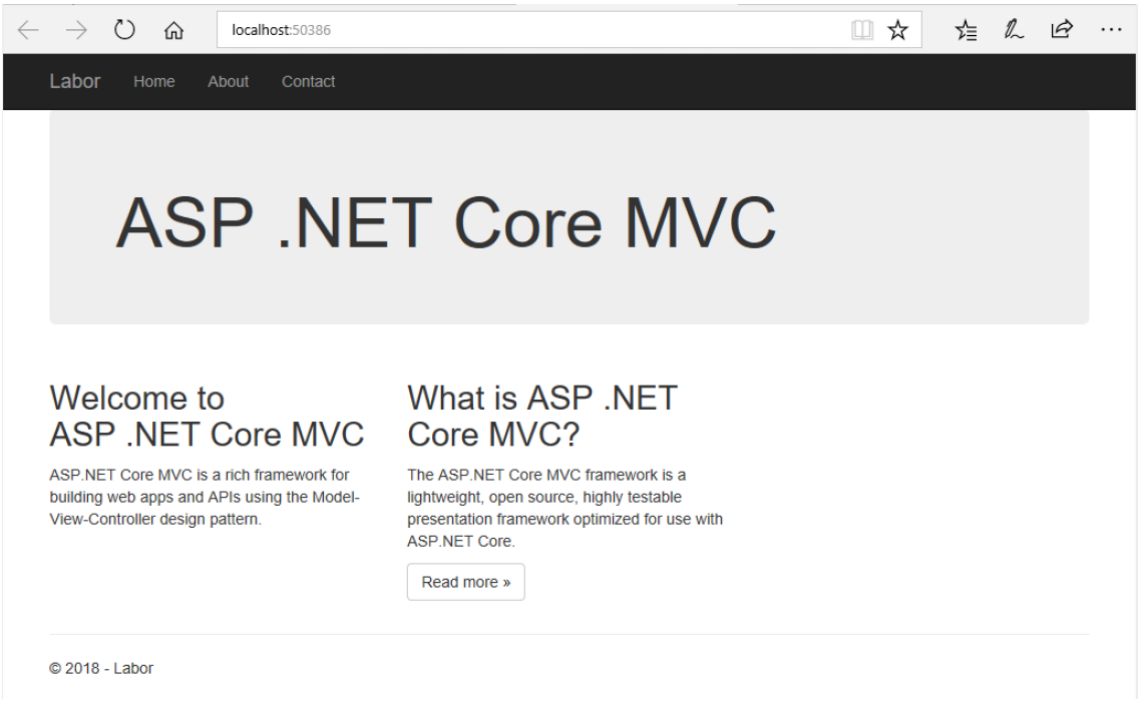
```
@model Facade.EmployeeViewModel
@{
    Layout = null;
}

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>MyView</title>
</head>
<body>
    Hello @Model.UserName
    <hr />
    <div>
        <b>Employee Details</b><br />
        Employee Name : @Model.EmployeeName <br />
        <span style="background-color: @Model.SalaryColor">
            Employee Salary: @Model.Salary
        </span>
    </div>
</body>
</html>
```

14. Ava Views/Home kaustas asuv Index.cshtml fail ning muuda seda järgnevalt

```
@{
    ViewData["Title"] = "ASP .NET Core MVC";
}

<div class="jumbotron">
    <h1>@ViewData["Title"]</h1>
</div>

<div class="row">
    <div class="col-md-4">
        <h2>Welcome to ASP .NET Core MVC</h2>
        <p>
            ASP.NET Core MVC is a rich framework for building web apps
            and APIs using the Model-View-Controller design pattern.
        </p>
    </div>

    <div class="col-md-4">
        <h2>What is ASP .NET Core MVC?</h2>
        <p>
            The ASP.NET Core MVC framework is a lightweight, open source,
            highly testable presentation framework optimized for use with ASP.NET Core.
        </p>
        <p>
            <a class="btn btn-default"
               href="https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.1">
            Read more &raquo;</a>
        </p>
    </div>
</div>
```

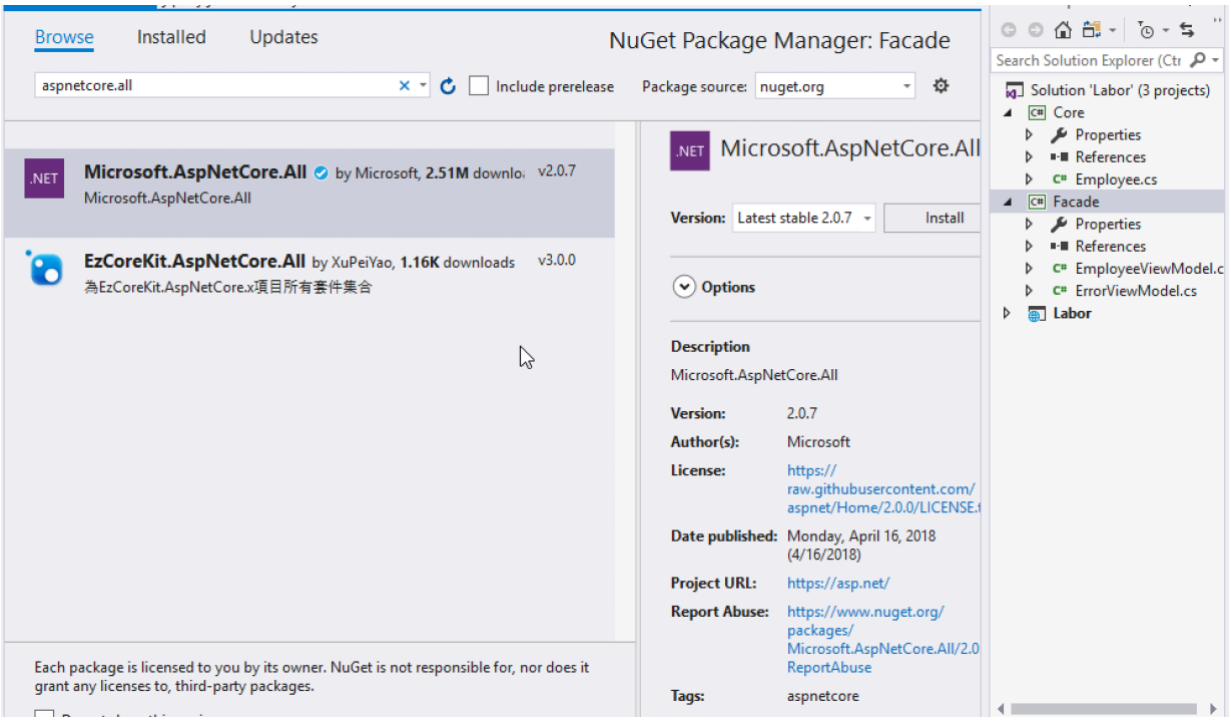
15. Ava HomeController ning eemalda sealt ebavajalik kasutuslause (*using.Labor.Models*) ning lisa kasutus Facade projektile (*using Facade*). Vajuta ka Ctr + F ning otsi sõna ’Model’ järgi, kas kuhugi on jäänud veel viide selles kaustas asuvatele failidele ning eemalda need.

16. Testi väljundit. Selleks vajuta F5 klahvi ja navigeeri vastavale URL-ile.

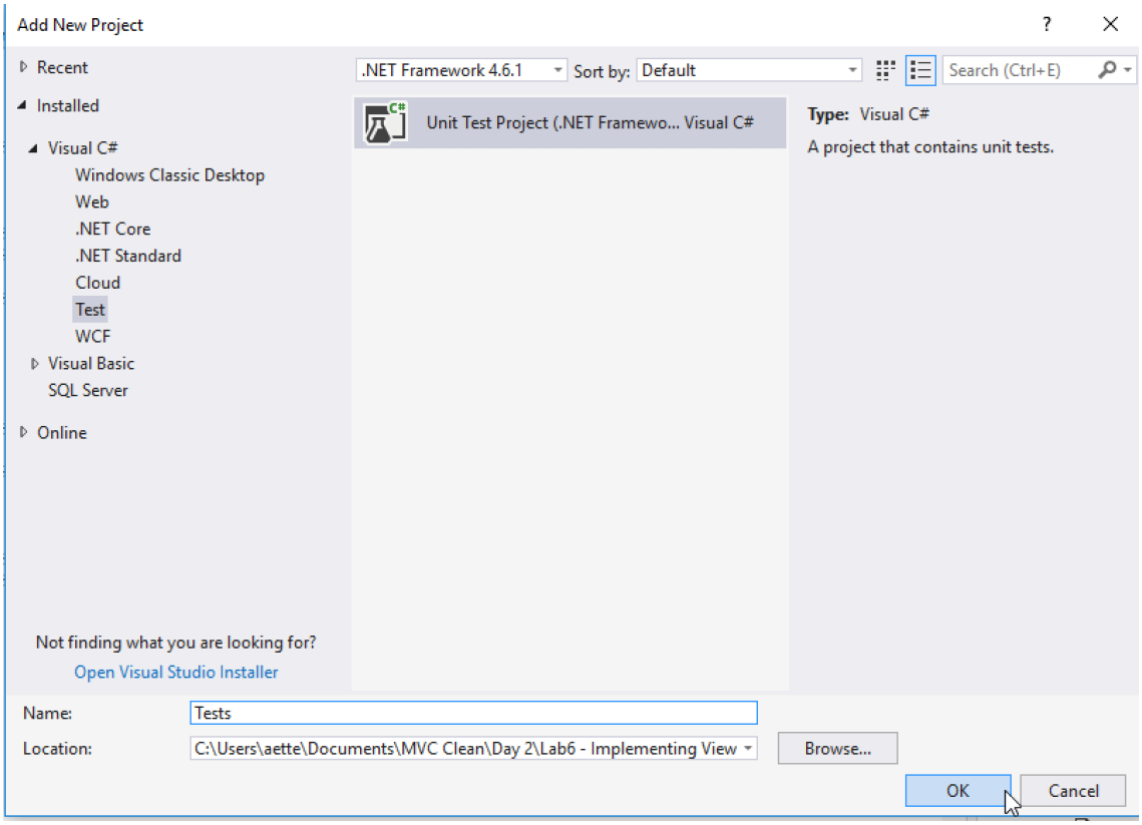


Igal vaatel peaks alati olema oma ViewModel, isegi kui see sisaldab samu omadusi, mis mudel- see lihtsustab tulevikus muudatuste tegemist.

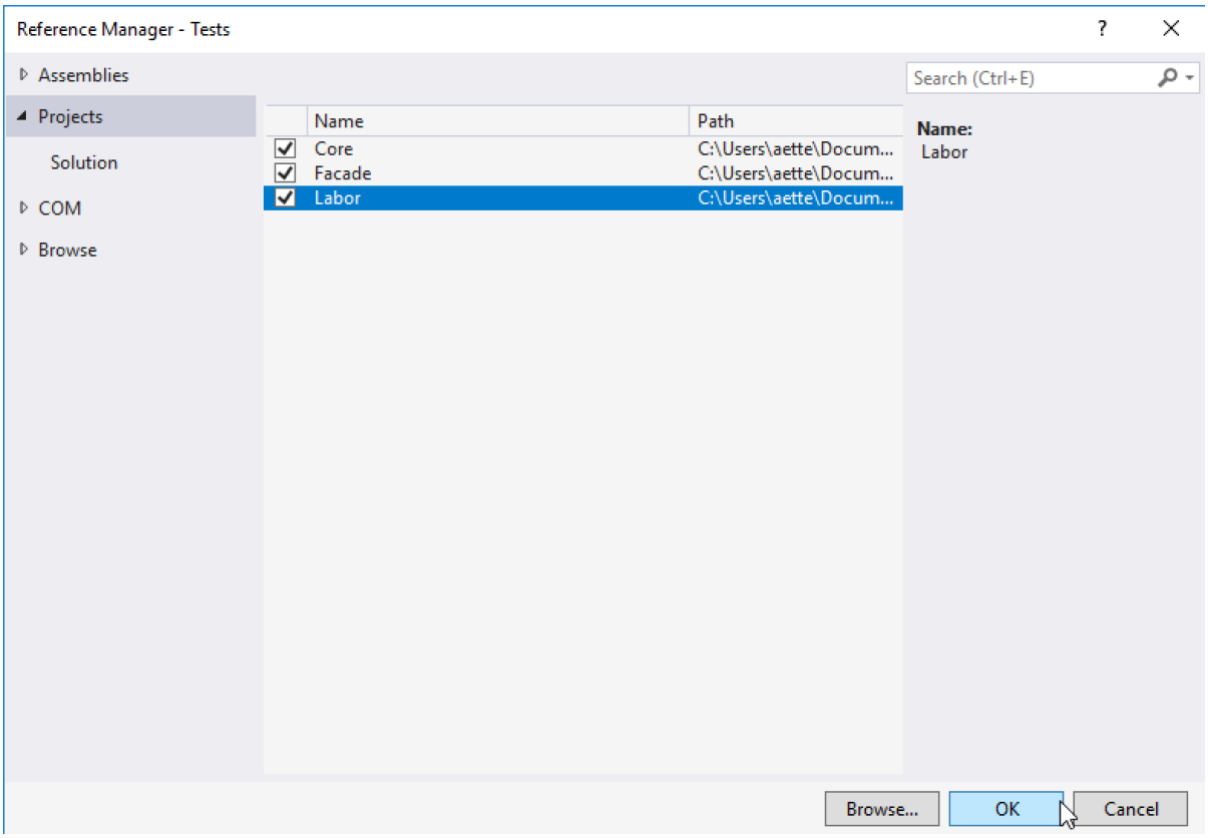
17. Kui testimine ei õnnestunud. Vaata, kas projekti Facade on installitud Microsoft.AspNetCore.All NutGet Package ning ega sellel paketil ei ole Update-e, mida oleks vaja teha.



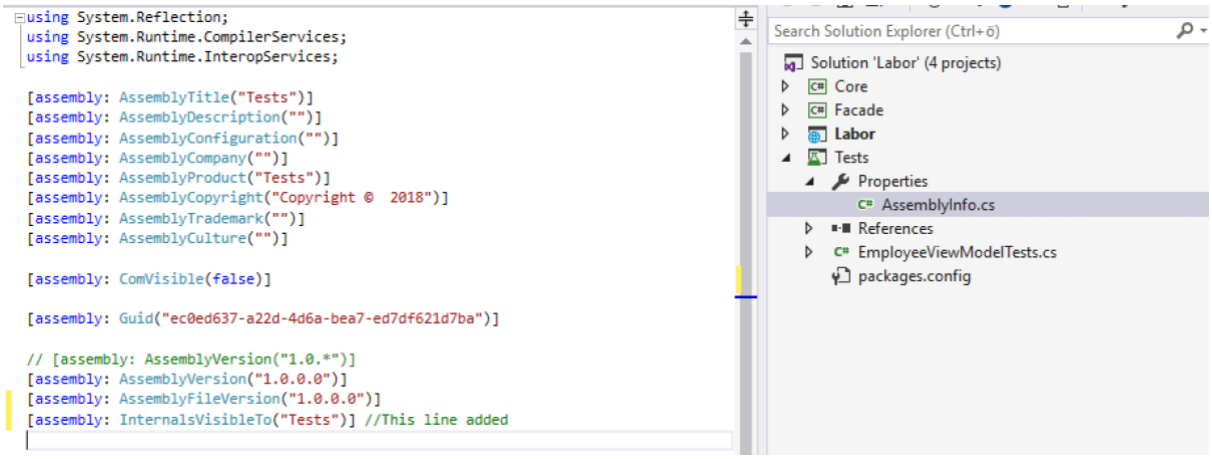
18. Lisame projektile ka ühiktestid. Selleks tee hiire parema sõrmise klikk Solutions kaustal ning vali Add -> New Project. Avanenud aknas vali Test -> Unit Test Project, määra nimeks Tests ning vajuta ’ok’.



19. Tests projekti puhul lisada viide(Add -> Reference) kõikidele teistele projektidele.



20. Ava Tests projektis asuv Properties -> Assemblyinfo klass ning lisa sinna omadus, et sisemised (internal) meetodid oleksid nähtavad.



21. Loo uus klass Tests projekti alla nimega EmployeeViewModelTests

Labor 7 - Kogumikuga vaade

1. Muuda ‘EmployeeViewModel’ klassi- eemalda UserName omadus, lisa kontroll, kui Employee objekt on null ning Salary-le vaikimisi väärtus.

```
namespace Facade
{
    public class EmployeeViewModel
    {
        public EmployeeViewModel(Employee emp)
        {
            if (emp is null) return;
            setName(emp);
            setSalary(emp);
            setColor(emp);
        }
        public string EmployeeName { get; set; }
        public string Salary { get; set; } = 0.ToString("C");
        public string SalaryColor { get; private set; } = "red";

        internal void setName(Employee e)
        {
            EmployeeName = e.FirstName + " " + e.LastName;
        }
        internal void setColor(Employee e)
        {
            if (!ReferenceEquals(null, e))
                SalaryColor = e.Salary > 15000 ?
                    "yellow" : "green";
            else SalaryColor = "red";
        }
        internal void setSalary(Employee e)
        {
            Salary = e.Salary.ToString("C");
        }

    }
}
```

2. Loo uus klass nimega ‘EmployeeListViewModel’ Facade projekti alla

```
public class EmployeeListViewModel
{
    public List<EmployeeViewModel> Employees { get; set; }
    public string UserName { get; set; }
}
```

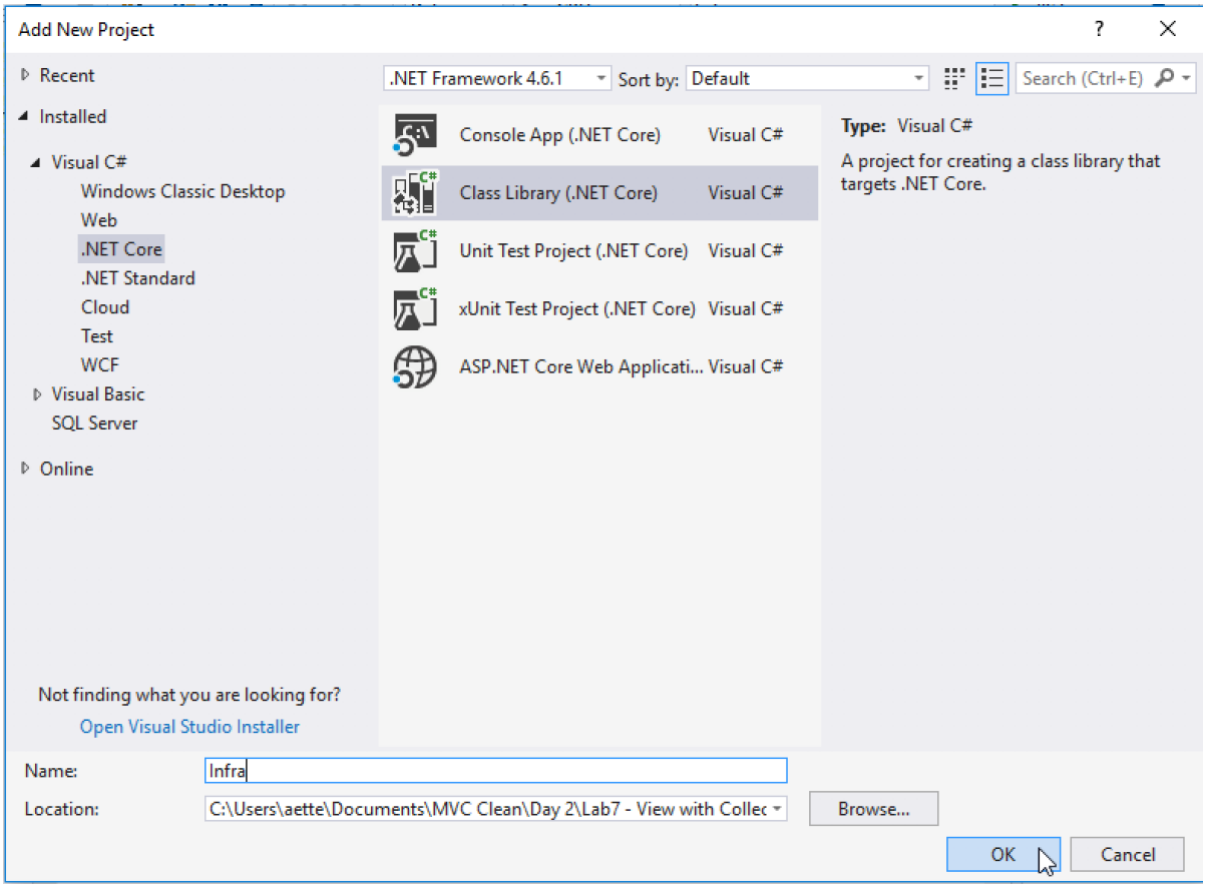
3. Muuda MyView.cshtml EmployeeListViewModel tüüpi tugevalt trükitud vaateks

```
@model Facade.EmployeeListViewModel
@{
    Layout = null;
}
```

4. Kuva töötajad(employees) vaates

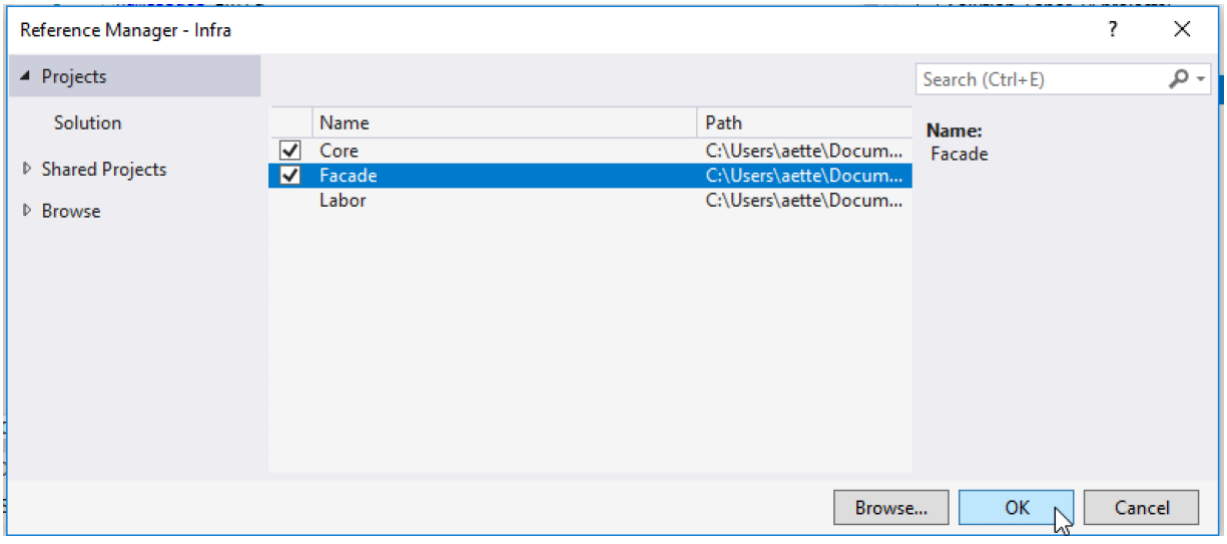
```
<body>
    Hello @Model.UserName
    <hr />
    <div>
        <table>
            <tr>
                <th>Employee Name</th>
                <th>Salary</th>
            </tr>
            @foreach (EmployeeViewModel item in Model.Employees)
            {
                <tr>
                    <td>@item.EmployeeName</td>
                    <td style="background-color:@item.SalaryColor">@item.Salary</td>
                </tr>
            }
        </table>
    </div>
```

5. Lisa uus projekt. Solution -> Add -> New Project. Vali avanenud aknas vasakult menüüribalt .NETCore ning Class Library. Määra projektile nimi ’Infra’.



6. Kontrolli, et Infra projektile oleks lisatud NuGet Package AspNetCore.All. Kui puudub, siis installeeri see.

7. Lisa Infra projekti viide Core ja Facade projektidele. Selleks tee hiire parema sõrmise klikk Infra projektil ning vali Add-> Reference. Lisa linnuke Core ning Facade projekti ees olevasse kasti ning vajuta ok.



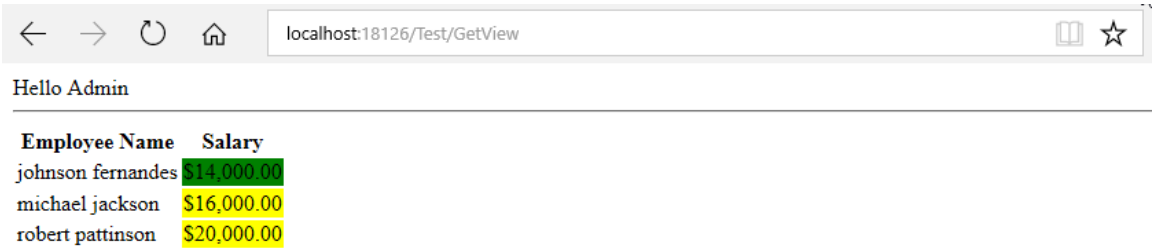
8. Lisa äriloogika kiht töötajate jaoks, selleks loo Infra projekti alla uus klass Employees, mille sisse lisa meetod Get()

```
namespace Infra
{
    public static class Employees
    {
        public static List<Employee> Get()
        {
            var employees = new List<Employee> {
            new Employee("John", "Doe", 14000),
            new Employee("Michael", "Jackson", 16000),
            new Employee("Robert", "Pattinson", 20000)
            };
        return employees;
        }
    }
}
```

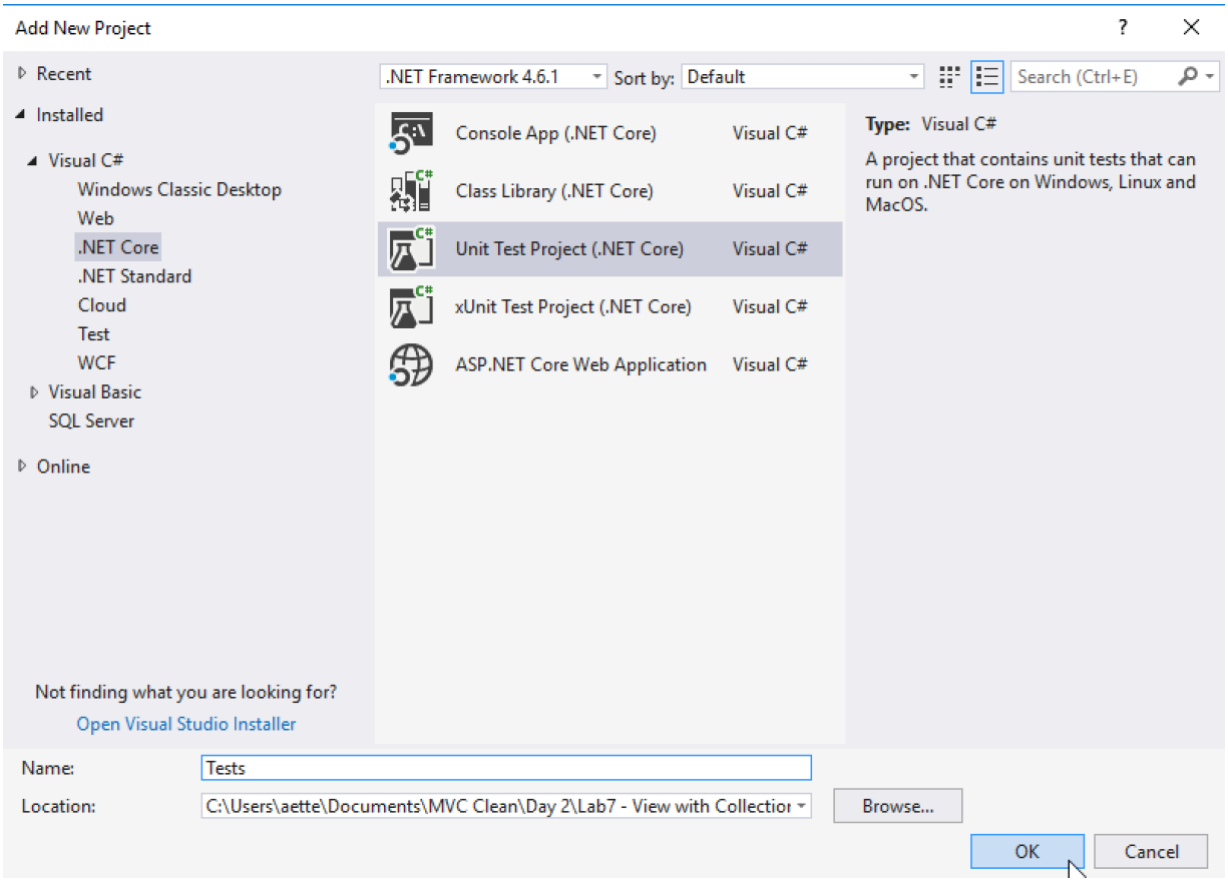
9. Tagasta töötajate andmed Test kontrollerile

```
public class TestController : Controller
{
    public ActionResult GetView()
    {
        var model = new EmployeeListViewModel();
        var employees = Employees.Get();
        var list = new List<EmployeeViewModel>();
        foreach (var e in employees)
        {
            var employee = new EmployeeViewModel(e);
            list.Add(employee);
        }
        model.Employees = list;
        model.UserName = "Admin";
        return View("MyView", model);
    }
}
```

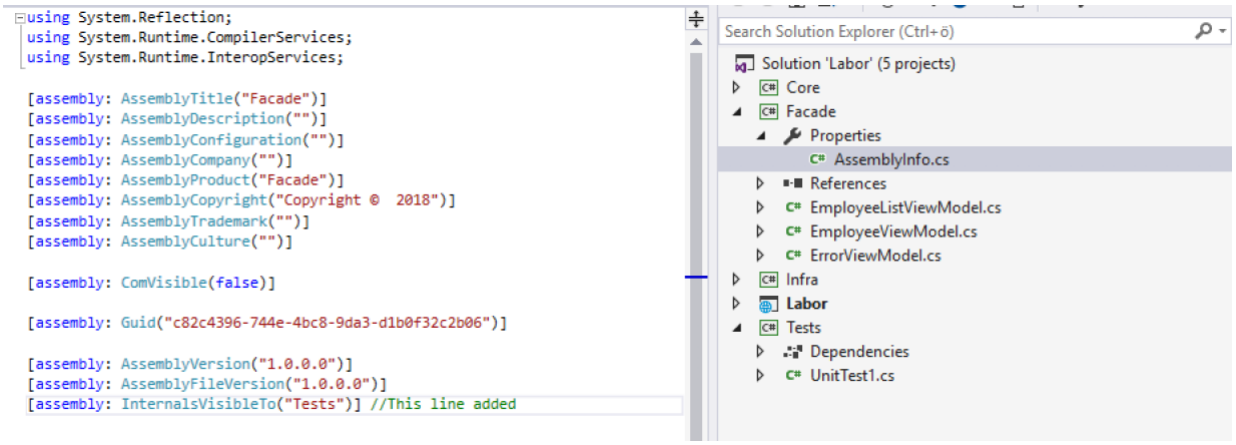
10. Testi väljundit. Selleks vajuta F5 klahvi ja navigeeri vastavale URL-ile.



11. Lisa uus projekt lahendusele. Seekord vali .NET Core -> Unit Test Project, pane projektile nimeks Tests ning vajuta ’ok’.

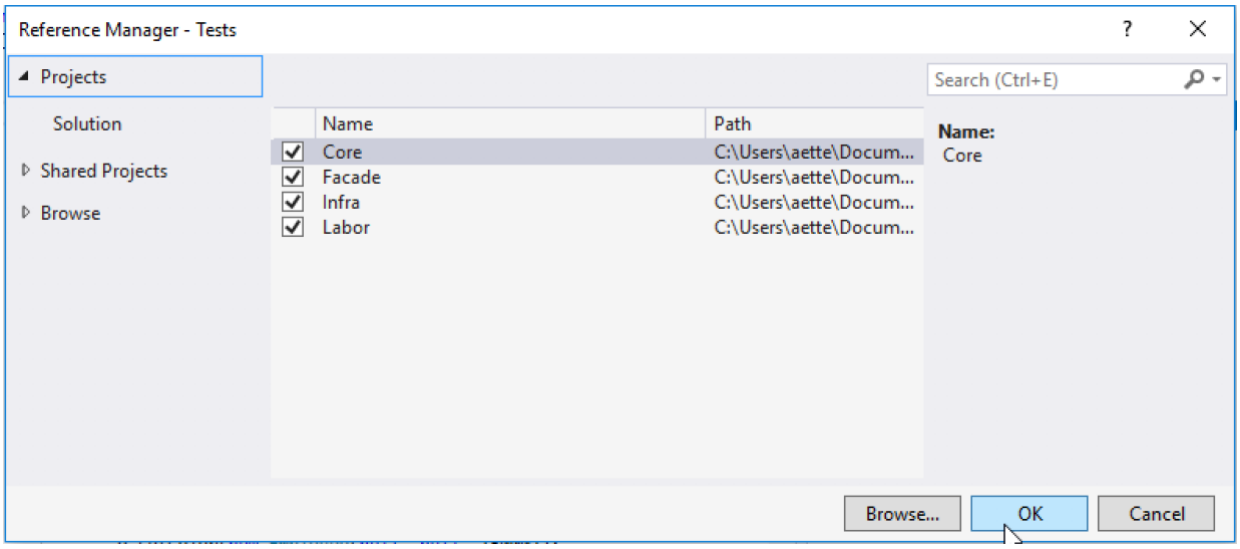


12. Ava Facade -> Properties -> AssemblyInfo klass ning lisa sinna omadus, et sisemised meetodid oleksid Tests projektile nähtavad.



13. Kontrolli, et NuGet Package Manager-is oleksid kõik uuendused tehtud.

14. Lisa projektile Tests viide kõikidele teistele projektidele

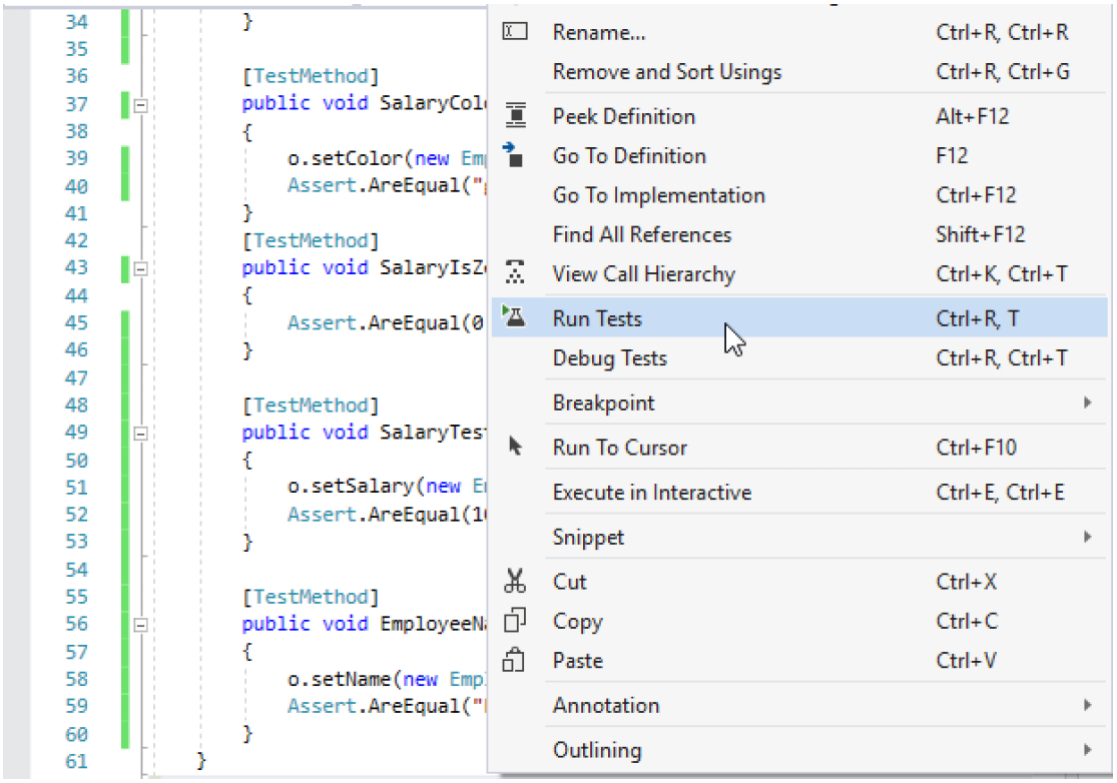


15. Lisa Tests projekti uus klass nimega EmployeeViewModelTests ning sinna sisse loo testmeetodid.

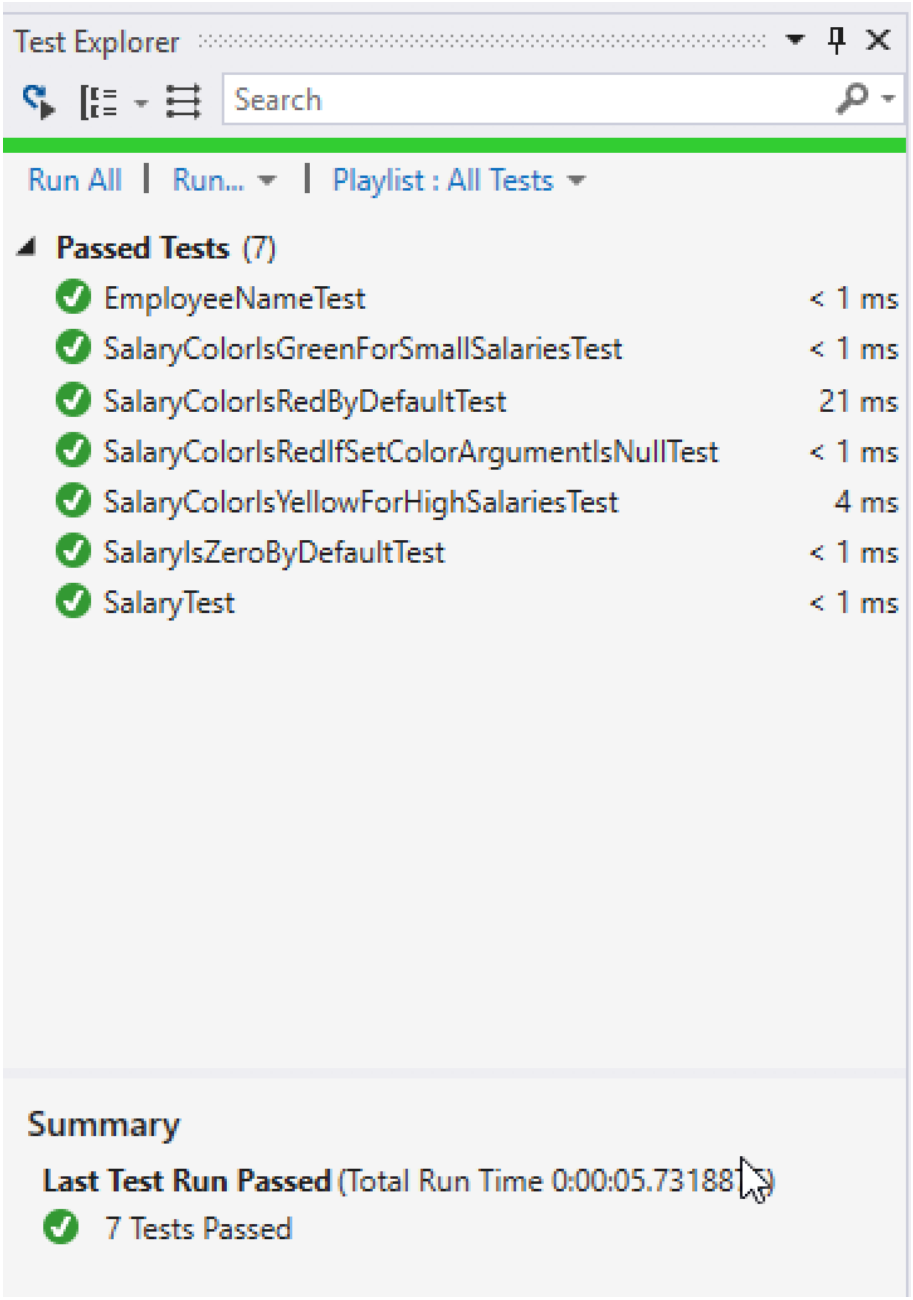
```
using Core;
using Facade;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Tests
{
    [TestClass]
    public class EmployeeViewModelTests
    {
        private EmployeeViewModel o;

        [TestInitialize]
        public void TestInitialize()
        {
            o = new EmployeeViewModel(null);
        }

        [TestMethod]
        public void SalaryColorIsRedByDefaultTest()
        {
            Assert.AreEqual("red", o.SalaryColor);
        }
        [TestMethod]
        public void SalaryColorIsRedIfSetColorArgumentIsNullTest()
        {
            o.setColor(null);
            Assert.AreEqual("red", o.SalaryColor);
        }
        [TestMethod]
        public void SalaryColorIsYellowForHighSalariesTest()
        {
            o.setColor(new Employee(null, null, 15001));
            Assert.AreEqual("yellow", o.SalaryColor);
        }
```

```
        [TestMethod]
        public void SalaryColorIsGreenForSmallSalariesTest()
        {
            o.setColor(new Employee(null, null, 15000));
            Assert.AreEqual("green", o.SalaryColor);
        }
        [TestMethod]
        public void SalaryIsZeroByDefaultTest()
        {
            Assert.AreEqual(0.ToString("C"), o.Salary);
        }

        [TestMethod]
        public void SalaryTest()
        {
            o.setSalary(new Employee(null, null, 100));
            Assert.AreEqual(100.ToString("C"), o.Salary);
        }

        [TestMethod]
        public void EmployeeNameTest()
        {
            o.setName(new Employee("First", "Last", 0));
            Assert.AreEqual("First Last", o.EmployeeName);
        }
    }
}
```

16. Tee hiire parema sõrmise klikk ning vali Run Tests.



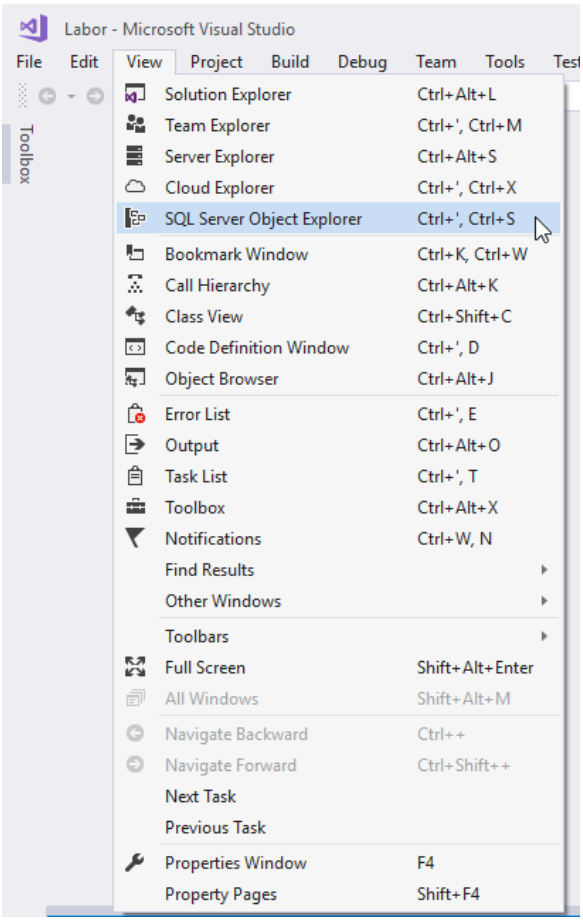
17. Tulemus peab olema järgmine



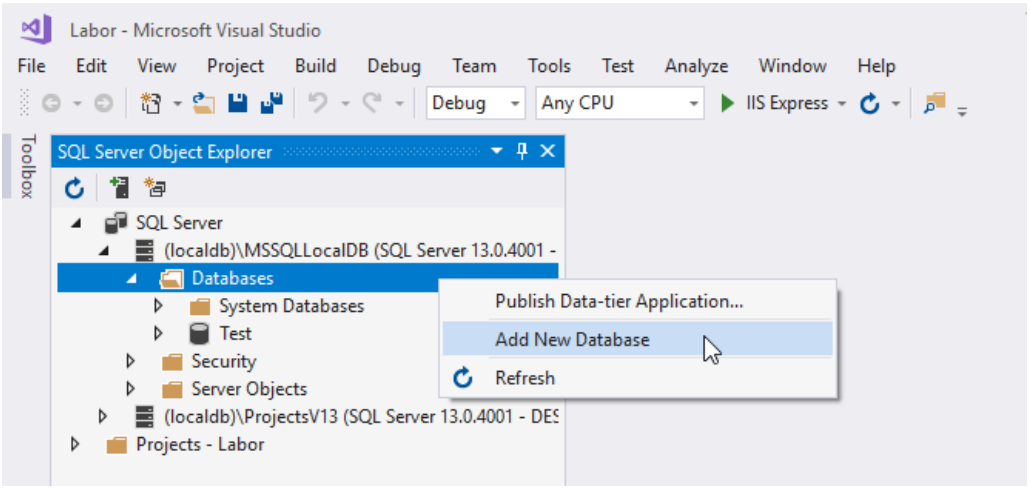
Labor 8 - Andmeedastuskihi lisamine

22. Loo uus andmebaas.

1.1 Vali ülevalt menüüst ‘View’ -> ‘SQL Server Object Explorer’



1.2 Avanenud SQL Server Object Explorer-is tee hiire parema sõrmise klikk ‘Databases’ kaustal ja vali ‘Add New Database’



1.3 Määra andmebaasile nimeks ‘Sales’

23. Loo uus klass nimega SalesDbContext projekti Infra alla.

```
using Core;
using Microsoft.EntityFrameworkCore;

namespace Infra
{
    public class SalesDbContext : DbContext
    {
        public SalesDbContext(DbContextOptions<SalesDbContext> options) : base(options) { }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
            builder.Entity<Employee>()
                .ToTable("Employees");
        }

        public DbSet<Employee> Employees { get; set; }
    }
}
```

24. Muuda Employee klassi järgmiselt

```
public class Employee
{
    public Employee()
    {
    }
    public Employee(string firstName,
        string lastName = null, int salary = 0)
    {
        FirstName = firstName?? string.Empty;
        LastName = lastName?? string.Empty;
        Salary = salary;
    }
    public int EmployeeId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Salary { get; set; }
}
```

25. Muuda Labor projektis Startup klassis olevat ConfigurationServices meetodit järgnevalt

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SalesDbContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"),
        b => b.MigrationsAssembly("Labor")));
    services.AddMvc();
}
```

26. Ava appsettings.json fail ja lisa andmebaasiga ühendamiseks vajalik ConnectionString parameeter.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=Sales;Trusted_Connection=True;MultipleActiveResultSets=true"
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```

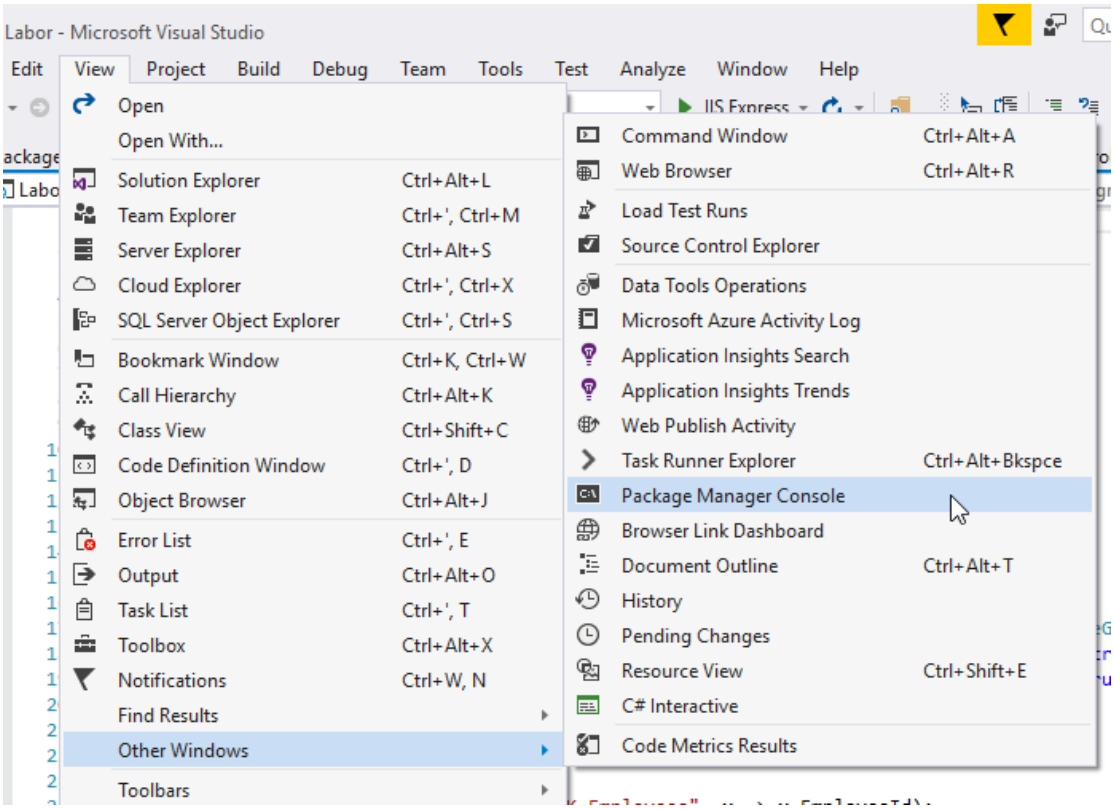
27. Muuda Infra projektis Employees klassis asuvat Get meetodit järgnevalt:

```
public class Employees
{
    public static List<Employee> Get(SalesDbContext db)
    {
        return db.Employees.ToList();
    }
}
```

28. *TestControlleris* lisa enne *GetView* action meetodit järgnevad read ning anna *Get* meetodit välja kutsudes sisendparameeter:

```
public class TestController : Controller
{
    private readonly SalesDbContext db;
    public TestController(SalesDbContext db) { this.db = db; }
    public ActionResult GetView()
    {
        var model = new EmployeeListViewModel();
        var employees = Employees.Get(db);
        var list = new List<EmployeeViewModel>();
        foreach (var e in employees)
        {
            var employee = new EmployeeViewModel(e);
            list.Add(employee);
        }
        model.Employees = list;
        model.UserName = "Admin";
        return View("MyView", model);
    }
}
```

29. Ava Package Manager Console, selleks vali üleval asuvalt menüüribalt View -> Other Windows -> Package Manager Console



30. Avanenud konsooli aknas käivita järgmised käsud (enne järgmise käsu käivitamist, oota kindlasti ära esimese käsu tulemus. Esimene käsk õnnestus, kui viimasele real asuv väärtus on *'To undo this action, use Remove-Migration.'* Teine käsk õnnestus kui viimasel real asuvaks väärtuseks on '*Done*.'

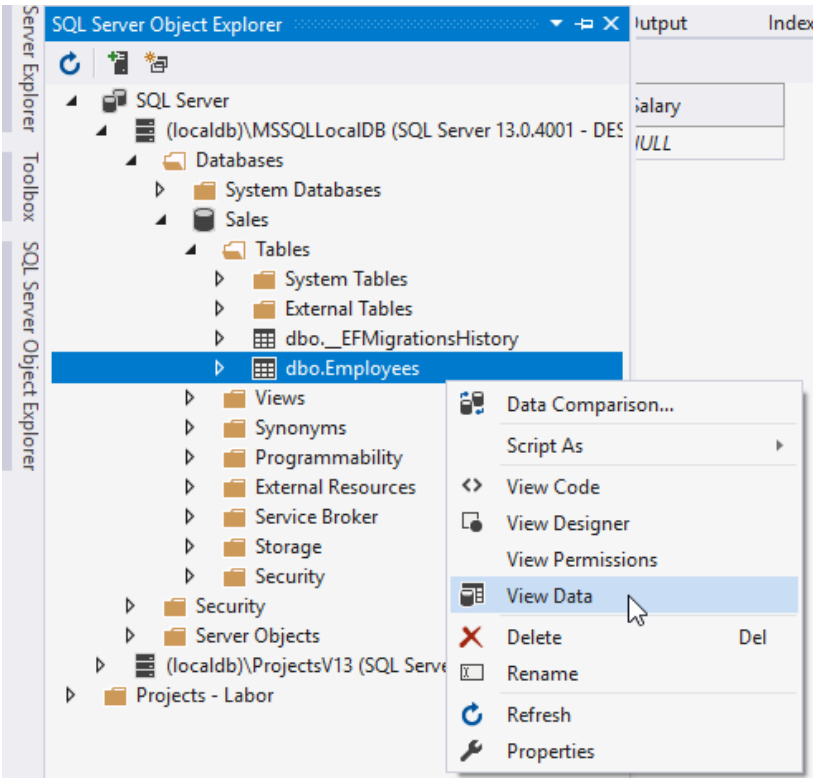
PM> Add-Migration initial

PM> Update-Database

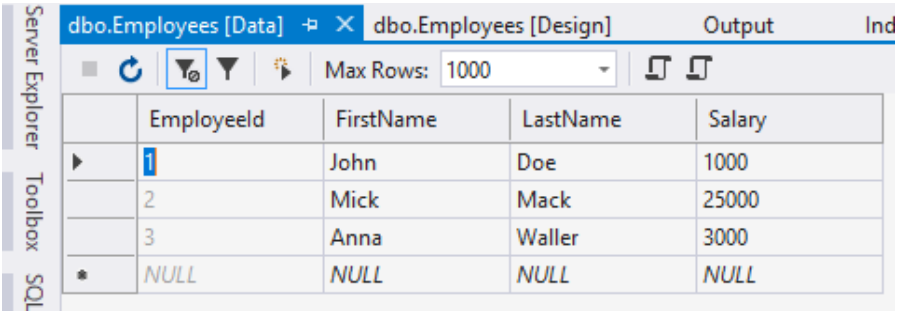
31. Vajuta F5 ja testi genereeritud koodi

32. Ava SQL Server Object Explorer ja lisa loodud tabelisse 'Employees' testandmed

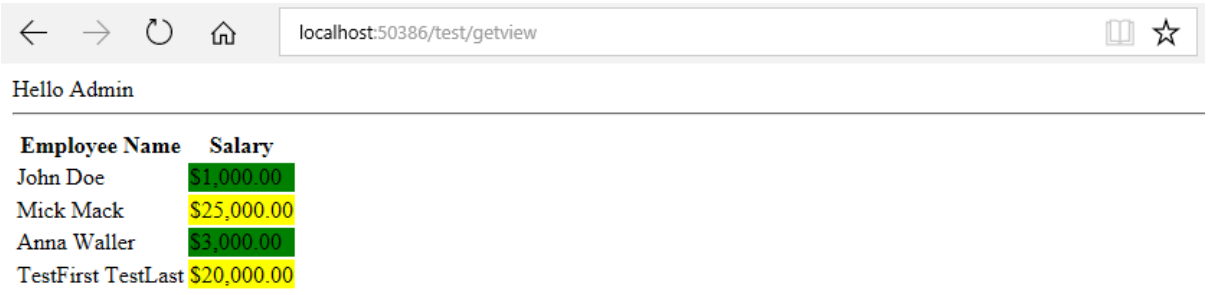
    a. Tee hiire parema sõrmise klikk tabelil Employees ja vali 'View Data'



b. Tee kahekordne hiireklõps vastava veeru real ning sisesta testandmed



33. Vajuta uuesti F5 ja testi, kas sisestatud testandmed on nähtavad



Loe lisaks: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-2.1

Labor 9 - Andmete sisestamise ekraani lisamine

1. Enne alustamist korrasta projekt, selleks:
    a. Nimeta ümber:
        i. “TestController” “EmployeeController”-iks
        ii. “GetView()” action meetod “Index()”-iks
        iii. “Test” kaust (asub Views kaustas) “Employee”-ks
        iv. “MyView” vaade “Index”-iks
    b. Eemalda UserName omadus EmployeeListViewModel-ist
    c. Eemalda UserName väärtuse kuvamine vaatest
        i. Selleks ava Views/Employee/Index.cshtml vaade ja eemalda järgmine koodilõik

        Hello @Model.UseName

        <hr/>

    d. Muuda Index action meetodit EmployeeController-is järgnevalt

```
model.Employees = list;
model.UserName = "Admin"; //Remove this line!
return View("Index", model); //Change the name of the view!
```

2. Kui projekt korrastatud, loo uus action meetod ‘AddNew’ EmployeeController-is, mis tagastaks CreateEmployee vaate

```
public ActionResult AddNew()
{
    return View("CreateEmployee");
}
```

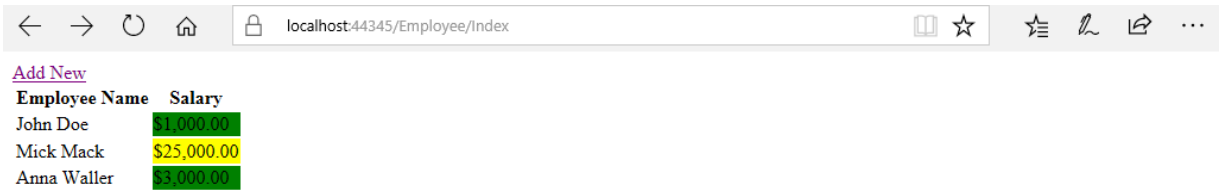
3. Loo vaade “CreateEmployee” View/Employee kaustas, mis näeb välja järgmine

```
@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>CreateEmployee</title>
</head>
<body>
    <div>
        <form action="/Employee/SaveEmployee" method="post">
            First Name: <input type="text" id="TxtFName" name="FirstName" value="" /><br />
            Last Name: <input type="text" id="TxtLName" name="LastName" value="" /><br />
            Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><br />
            <input type="submit" name="BtnSave" value="Save Employee" />
            <input type="button" name="BtnReset" value="Reset" />
        </form>
    </div>
</body>
</html>
```

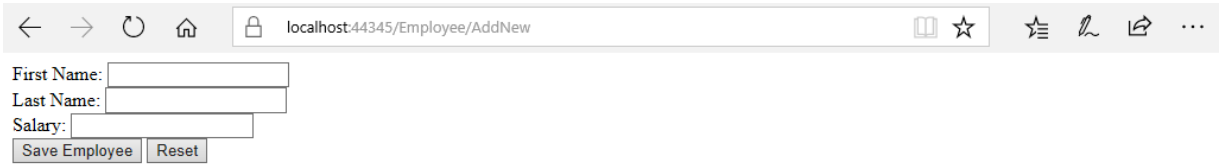
4. Loo link, mis viitaks AddNew action meetodi URL-ile Index.cshtml vaate sees

```
<a href ="/Employee/AddNew">Add New</a>
```

5. Vajuta F5 ja testi loodud koodi.



Vajutades lingil “Add New”, toimub suunamine järgmisele URL-ile.



6. Eemalda Index vaatelt Layout ning asenda see pealkirjaga.

```
@{
    ViewBag.Title = "Employees";
}
```

7. Muuda Index vaates pealkirja osa ning lisa tabelile klassi table tunnus.

```
<!DOCTYPE html>
<html>
<h2>Employees</h2>
<body>
<div>
    <p>
        <a href="/Employee/AddNew">Add New</a>
    </p>
    <table class="table">
        <tr>
            <th>Employee Name</th>
            <th>Salary</th>
        </tr>
        @foreach (var item in Model.Employees)
        {
            <tr>
                <td>@item.EmployeeName</td>
                <td style="background-color: @item.SalaryColor">@item.Salary</td>
            </tr>
        }
    </table>
</div>
</body>
</html>
```

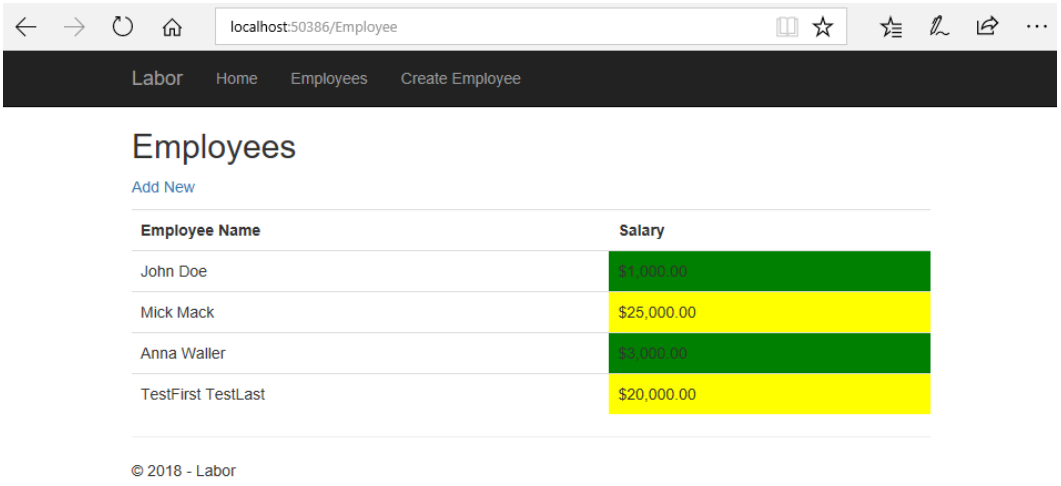
8. Tee sama laadi muudatused ka CreateEmployee vaates, lisa ka paragrahvi sildid vormi elementidele.

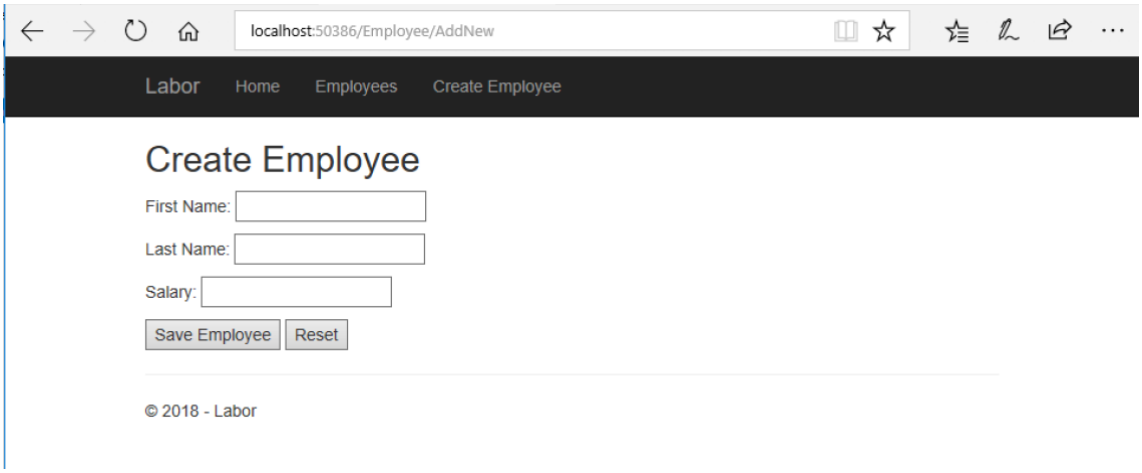
```
@{
    ViewBag.Title = "Create Employee";
}
<!DOCTYPE html>
<html>
<h2>Create Employee</h2>
<body>
<div>
    <form action="/Employee/SaveEmployee" method="post">
        <p>First Name: <input type="text" id="TxtFName" name="FirstName" value=""/><p/>
        <p>Last Name: <input type="text" id="TxtLName" name="LastName" value=""/><p/>
        <p>Salary: <input type="text" id="TxtSalary" name="Salary" value=""/><p/>
                <input type="submit" name="BtnSave" value="Save Employee"/>
                <input type="button" name="BtnReset" value="Reset"/>
    </form>
</div>
</body>
</html>
```

9. Muuta Shared/_Layout.cshtml faili. Muuda pealkirjas ViewData objekt ViewBag-iks ning muuda navigatsiooni menüüriba.

```
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>@ViewBag.Title - Labor</title>

    <environment>...</environment>
    <environment>...</environment>
</head>
<body>
<nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a asp-area="" asp-controller="Home" asp-action="Index" class="navbar-brand">Labor</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>
                    <a asp-area="" asp-controller="Home" asp-action="Index">Home</a>
                </li>
                <li>
                    <a asp-area="" asp-controller="Employee" asp-action="Index">Employees</a>
                </li>
                <li>
                    <a asp-area="" asp-controller="Employee" asp-action="AddNew">Create Employee</a>
                </li>
            </ul>
        </div>
```

10. Vajuta F5 ja testi rakendust.


localhost:50386/Employee
Labor
Home
Employees
Create Employee
Employees
Add New
Employee Name
Salary
John Doe
$1,000.00
Mick Mack
$25,000.00
Anna Waller
$3,000.00
TestFirst TestLast
$20,000.00
© 2018 - Labor

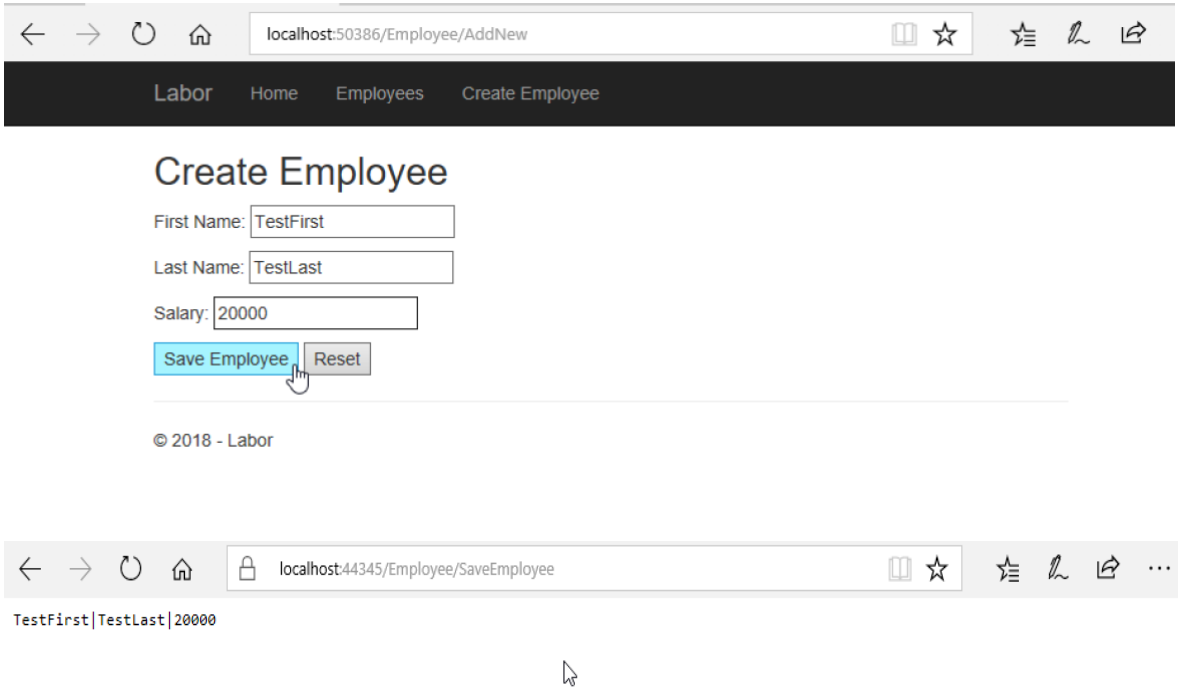

localhost:50386/Employee/AddNew
Labor
Home
Employees
Create Employee
Create Employee
First Name:
Last Name:
Salary:
Save Employee
Reset
© 2018 - Labor

Labor 10 - Sisestatud andmete töötlemine kontrolleris

1. Loo uus action meetod SaveEmployee() Employee kontrolleris

```
public string SaveEmployee(Employee e)
{
    return e.FirstName + "|" + e.LastName + "|" + e.Salary;
}
```

2. Vajuta klahvi F5 ja testi rakendust



Labor 11- *Reset* ja *Cancel* nuppude funktsionaalsus

1. Lisa CreateEmployee vaatesse nupud *Reset* ja *Cancel* ning päisesse JavaScripti funktsioon ResetForm()

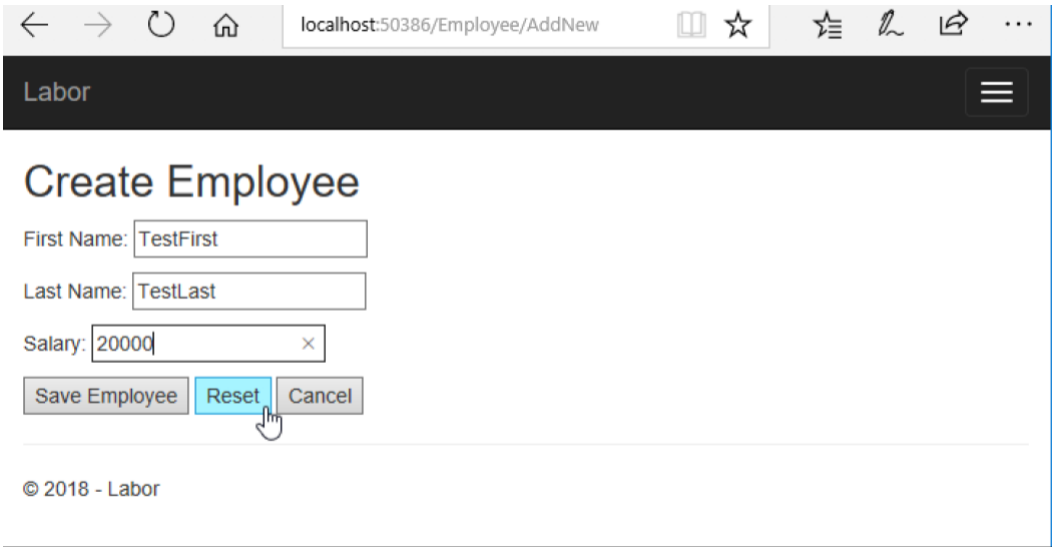
```
@{
    ViewBag.Title = "Create Employee";
    <script>
        function ResetForm() {
            document.getElementById("TxtFName").value = "";
            document.getElementById("TxLtName").value = "";
            document.getElementById("TxtSalary").value = "";
        }
    </script>
}

<h2>Create Employee</h2>
    <div>
        <form action="/Employee/SaveEmployee" method="post">
            <p>First Name: <input type="text" id="TxtFName" name="FirstName" value="" /><p />
            <p>Last Name: <input type="text" id="TxtLName" name="LastName" value="" /><p />
            <p>Salary: <input type="text" id="TxtSalary" name="Salary" value="" /><p />
            <input type="submit" name="BtnSubmit" value="Save Employee" />
            <input type="button" name="BtnReset" value="Reset" onclick="ResetForm()" />
            <input type="submit" name="BtnSubmit" value="Cancel" />
        </form>
    </div>
```

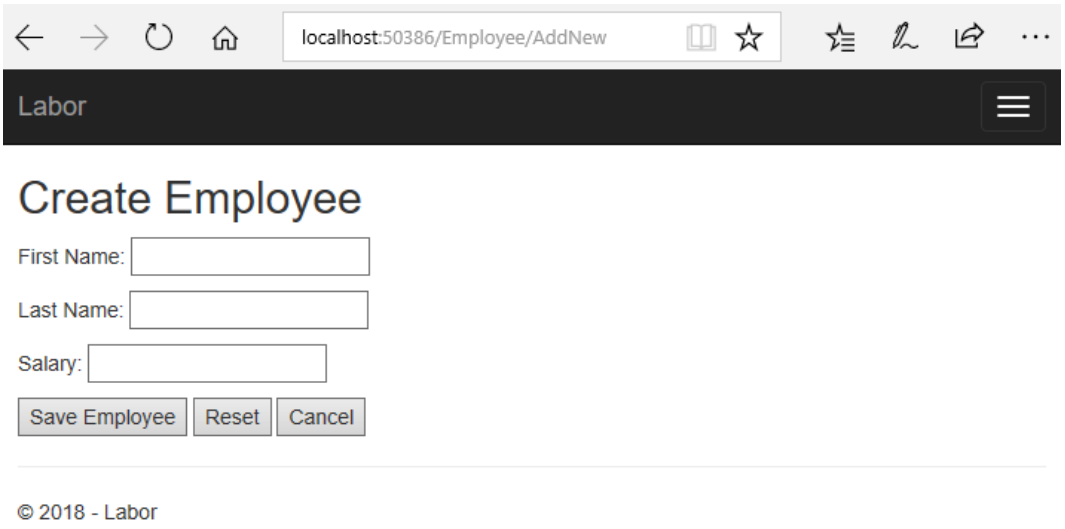
2. Muuda SaveEmployee action meetodit nii, et see tuleks toime ka *Cancel* nupu vajutusega

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            return Content(e.FirstName + "|" + e.LastName + "|" + e.Salary);
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

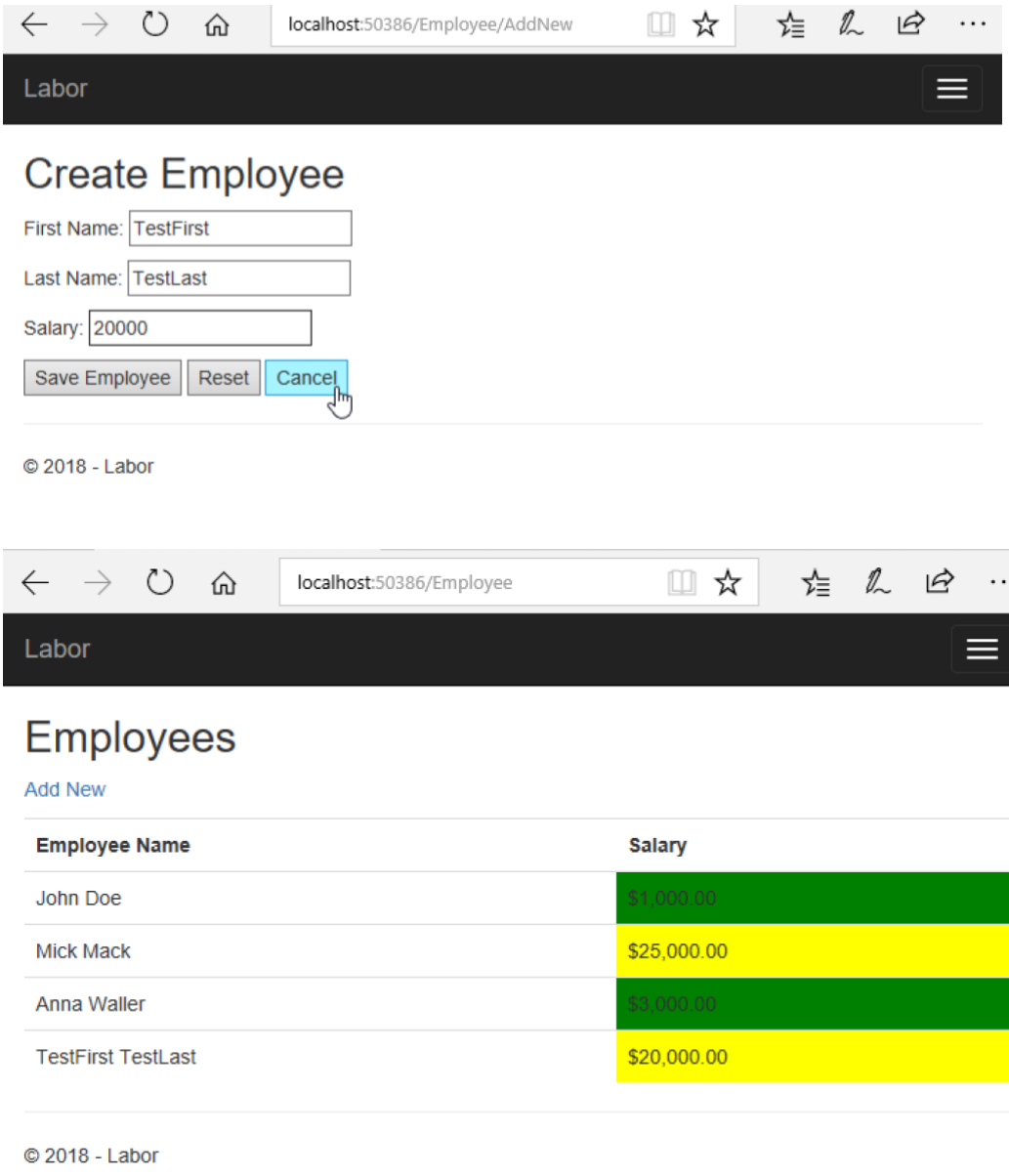
3. Vajuta klahvi F5 ja testi rakendust

a. Testi *Reset* nupul vajutust

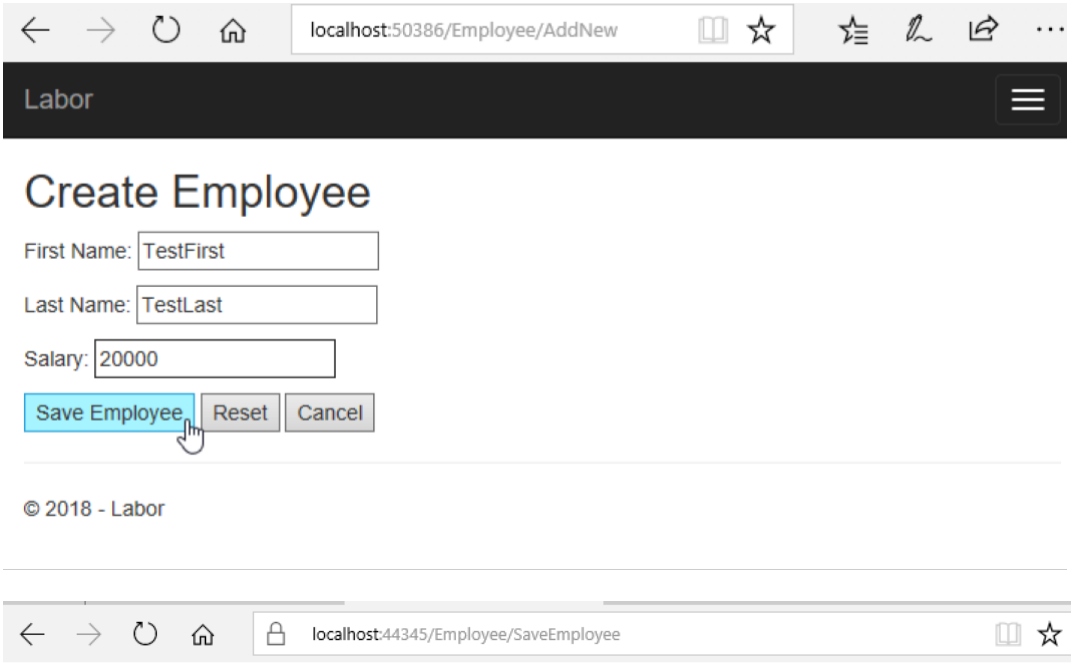




b. Testi *Cancel* nupul vajutust



c. Testi *Save Employee* nupul vajutust




```
TestFirst|TestLast|20000
```

Labor 12 - *Andmete andmebaasi salvestamine ja vaate uuendamine*

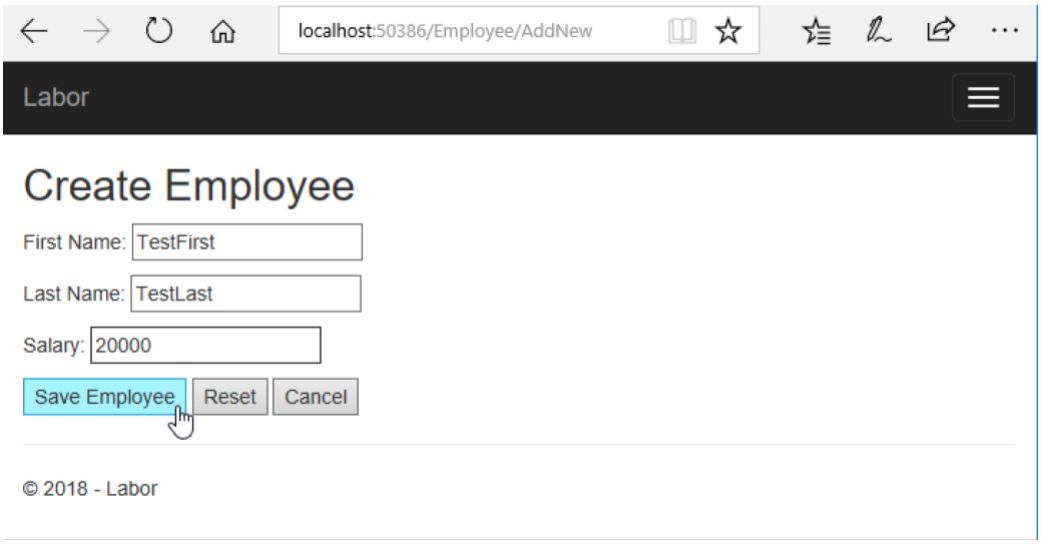
1. Loo Employee tüüpi Save meetod Infra projektis asuvasse Employees klassi.

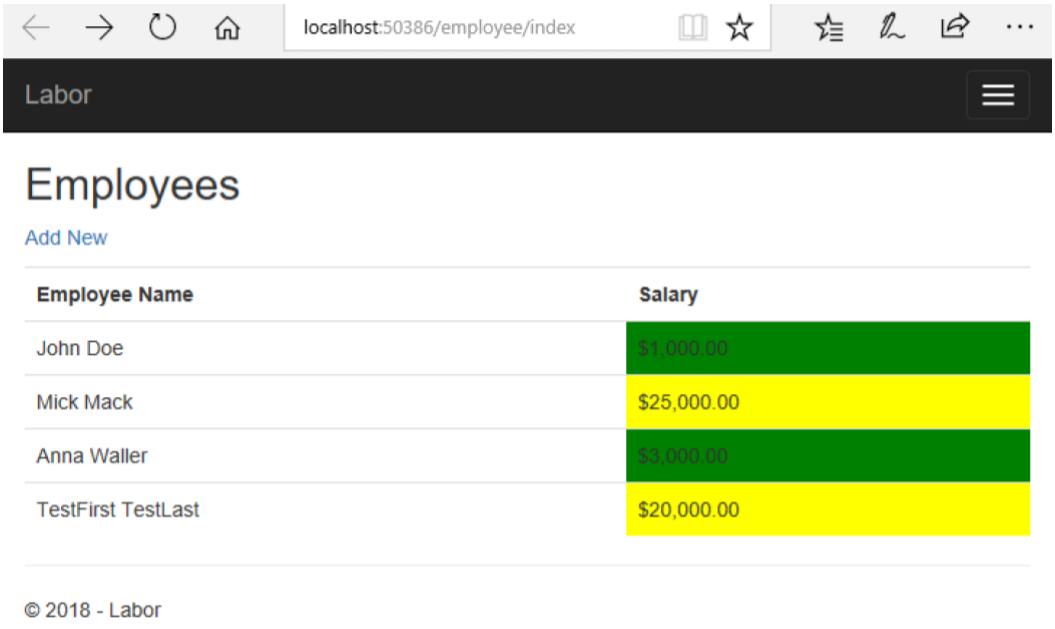
```
namespace Infra
{
    public class Employees
    {
        public static List<Employee> Get(SalesDbContext db)
        {
            return db.Employees.ToList();
        }

        public Employee Save(Employee e, SalesDbContext db)
        {
            db.Employees.Add(e);
            db.SaveChanges();
            return e;
        }
    }
}
```

2. Muuda SaveEmployee action meetodit

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            Employees emp = new Employees();
            emp.Save(e, db);
            return RedirectToAction("Index");
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

3. Vajuta klahvi F5 ja testi rakendust





Loe lisaks: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/working-with-sql?view=aspnetcore-2.1&tabs=aspnetcore2x

Labor 13- *Serveri poolne valideerimine*

1. Lisa Employee klassi omadustele annotatsioonid

```
public class Employee
{
    private const string requiredField = "Required field!";
    private const string lenghtIsToBig = "Length should be less than 20 characters!!";

    public Employee()
    {
    }
    public Employee(string firstName,
        string lastName, int salary)
    {
        FirstName = firstName;
        LastName = lastName;
        Salary = salary;
    }
    public int EmployeeId { get; set; }
    [Required(ErrorMessage = requiredField)]
    [StringLength(20, ErrorMessage = lenghtIsToBig)]
    public string FirstName { get; set; }
    [StringLength(20, ErrorMessage = lenghtIsToBig)]
    public string LastName { get; set; }
    public int Salary { get; set; }
}
```

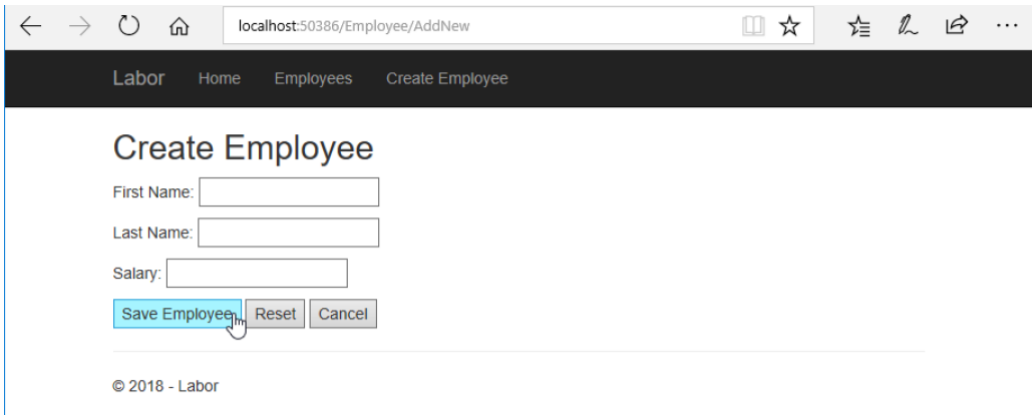
2. Muuda SaveEmployee action meetodit

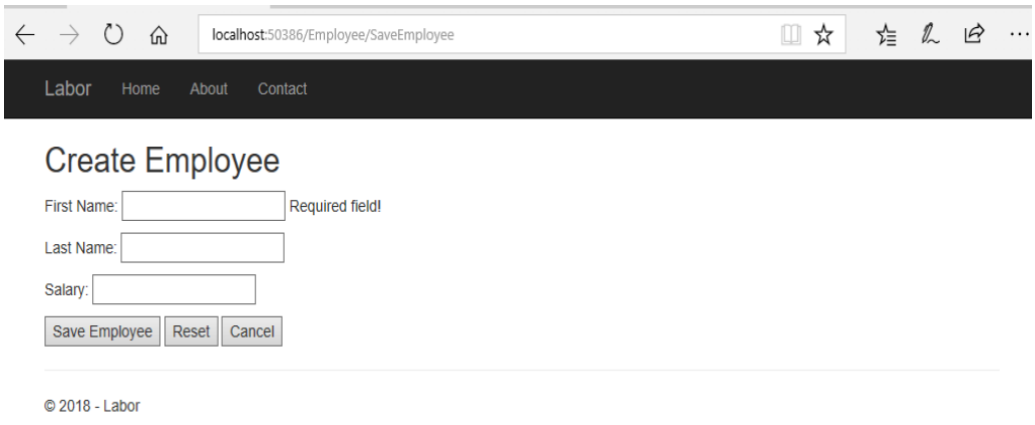
```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    switch (BtnSubmit)
    {
        case "Save Employee":
            if (ModelState.IsValid)
            {
                Employees emp = new Employees();
                emp.Save(e, db);
                return RedirectToAction("Index");
            }
            else
            {
                return View("CreateEmployee");
            }
        case "Cancel":
            return RedirectToAction("Index");
    }
    return new EmptyResult();
}
```

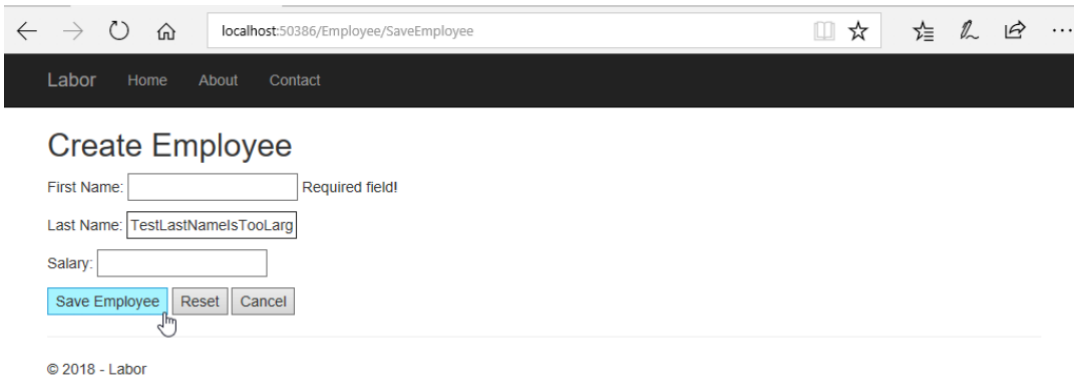
3. Lisa CreateEmployee vaatesse valideerimissõnumid

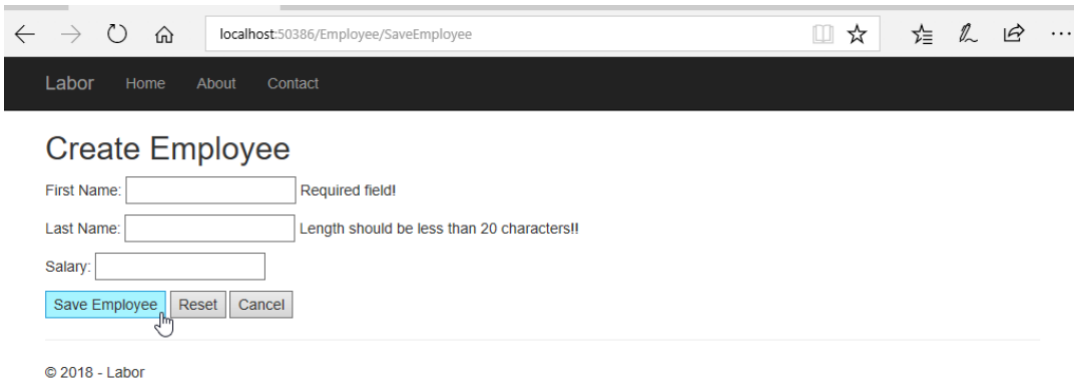
```
<body>
    <div>
        <form action="/Employee/SaveEmployee" method="post">
            <p>First Name: <input type="text" id="TxtFName" name="FirstName" value=""/>
                @Html.ValidationMessage("FirstName") </p>
            <p>Last Name: <input type="text" id="TxtLName" name="LastName" value=""/>
                @Html.ValidationMessage("LastName")</p>
            <p>Salary: <input type="text" id="TxtSalary" name="Salary" value=""/></p>
            <input type="submit" name="BtnSubmit" value="Save Employee" />
            <input type="button" name="BtnReset" value="Reset" onclick="ResetForm()"/>
            <input type="submit" name="BtnSubmit" value="Cancel" />
        </form>
    </div>
</body>
```

4. Vajuta klahvi F5 ja testi rakendust


localhost:50386/Employee/AddNew
Labor
Home
Employees
Create Employee
Create Employee
First Name:
Last Name:
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name:
Required field!
Last Name:
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name:
Required field!
Last Name:
TestLastNameIsTooLarg
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name:
Required field!
Last Name:
Length should be less than 20 characters!!
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor

Labor 14- *Serveri poolse valideerimise kohandamine*

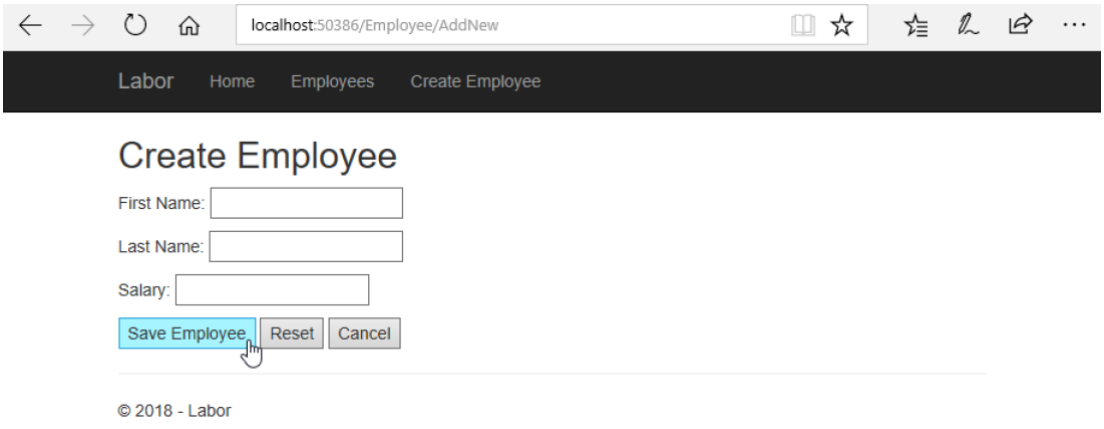
1. Lisa uus klass nimega NameValidation Core projekti alla, mis pärineb ValidationAttribute klassist.

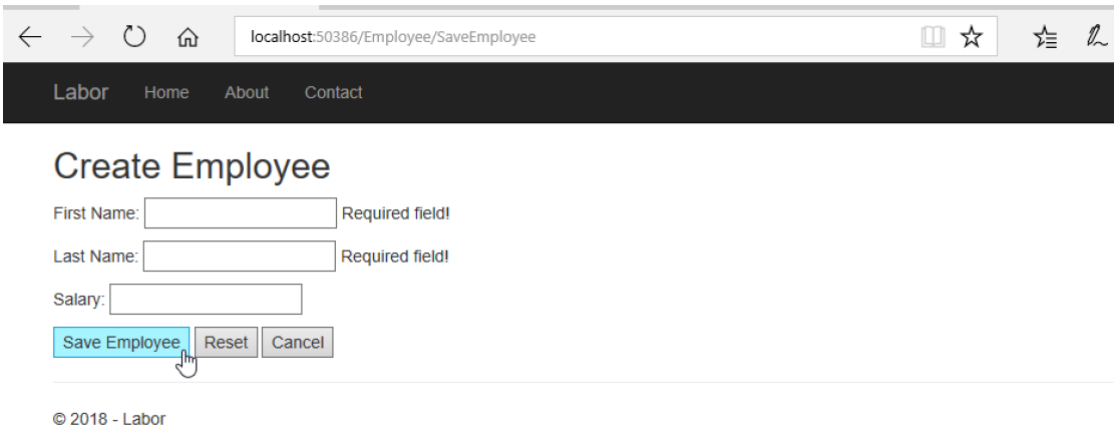
```
public class NameValidation : ValidationAttribute
{
    protected const string requiredField = "Required field!";
    protected const string lenghtIsToBig = "Length should be less than 20 characters!!";
    protected const string useOnlyLetters = "Use only letters!";

    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        if (value == null) return error(requiredField);
        var s = value.ToString();
        if (s.Length > 20) return error(lenghtIsToBig);
        if (!onlyLetters(s)) return error(useOnlyLetters);
        return ValidationResult.Success;
    }
    protected static bool onlyLetters(string s)
    {
        if (string.IsNullOrEmpty(s)) return false;
        if (string.IsNullOrEmpty(s.Trim())) return false;
        return s.All(char.IsLetter);
    }

    protected static ValidationResult error(string s)
    {
        return new ValidationResult(s);
    }
}
```

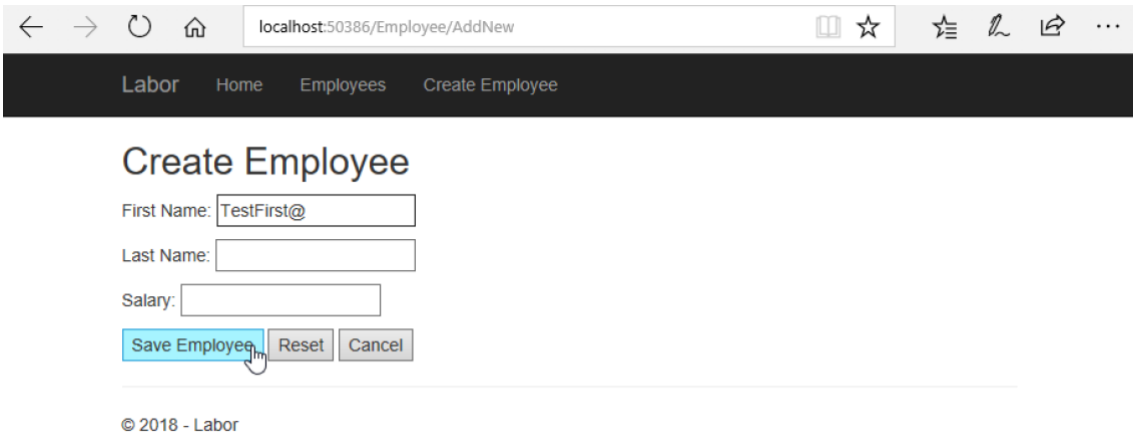
2. Eemalda varem lisatud valideerimisreeglid ning lisa NameValidation reeglid asemele.

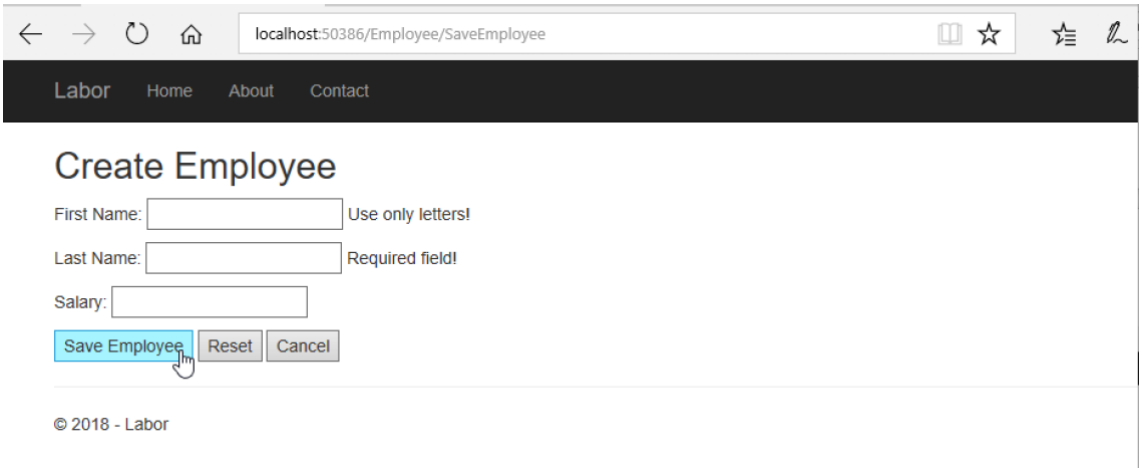
```
public class Employee
{
    public Employee()
    {
    }
    public Employee(string firstName,
        string lastName, int salary)
    {
        FirstName = firstName;
        LastName = lastName;
        Salary = salary;
    }
    public int EmployeeId { get; set; }
    [NameValidation]
    public string FirstName { get; set; }
    [NameValidation]
    public string LastName { get; set; }
    public int Salary { get; set; }
}
```

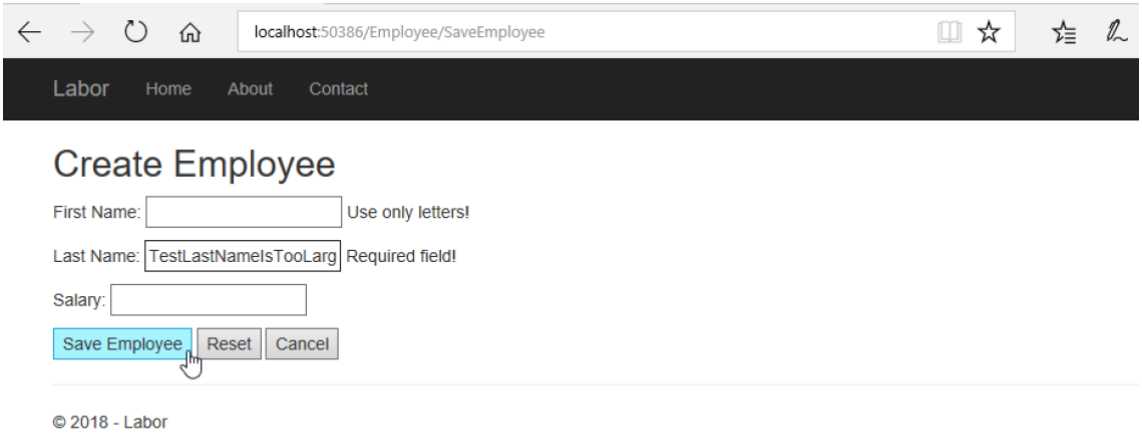
3. Vajuta klahvi F5 ja testi rakendust

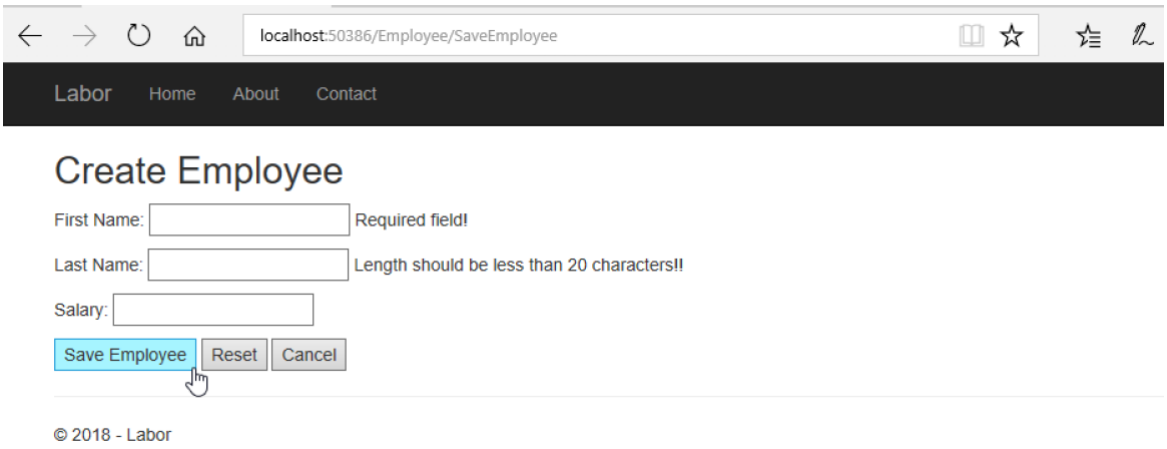





localhost:50386/Employee/AddNew
Labor
Home
Employees
Create Employee
Create Employee
First Name: TestFirst@
Last Name:
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name: Use only letters!
Last Name: Required field!
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name: Use only letters!
Last Name: TestLastNameIsTooLarg Required field!
Salary:
Save Employee
Reset
Cancel
© 2018 - Labor



Loe lisaks: https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-2.1

Labor 15- *Väärtuste säilitamine Validation Error-i korral*

1. Loo Facade projekti uus klass CreateEmployeeViewModel

```
namespace Labor.ViewModels
{
    public class CreateEmployeeViewModel
    {

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Salary { get; set; }
    }
}
```

2. Muuda SaveEmployee action meetodit ning loo privaatne meetod save().

```
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    if (BtnSubmit != "Save Employee") return RedirectToAction("Index");
    if (!ModelState.IsValid) return View("CreateEmployee");
    return save(e);

}

private ActionResult save(Employee e)
{
    Employees emp = new Employees();
    emp.Save(e, db);
    return RedirectToAction("Index");
}
```

3. Loo uus klass SalaryValidation Core projekti, mis pärineb NameValidation klassist.

```
class SalaryValidation: NameValidation
{
    protected const string salaryRange = "Salary must be between {0} and {1}";
    protected int minSalary = 5000;
    protected int maxSalary = 50000;
    protected override ValidationResult IsValid(object value,
        ValidationContext validationContext)
    {
        if (value == null) return error(requiredField);
        var i = value as int?;
        if (i < minSalary) return error(salary);
        if (i > maxSalary) return error(salary);
        return ValidationResult.Success;
    }

    private string salary{
        get{
            var min = minSalary.ToString("C");
            var max = maxSalary.ToString("C");
            var s = string.Format(salaryRange, min, max);
            return s;
        }
    }
}
```

4. Lisa Employee klassis Salary objektile SalaryValidation reegel

```
public Employee()
{
}
public Employee(string firstName,
    string lastName = null, int salary = 0)
{
    FirstName = firstName ?? string.Empty;
    LastName = lastName ?? string.Empty;
    Salary = salary;
}
public int EmployeeId { get; set; }
[NameValidation]
public string FirstName { get; set; }
[NameValidation]
public string LastName { get; set; }
[SalaryValidation]
public int Salary { get; set; }
```

5. Muuda CreateEmployee vaade CreateEmployeeViewModel tüüpi vaateks

```
@model Facade.CreateEmployeeViewModel
```

6. Lisa *value*= parameetrid CreateEmployee input väljadele ning Salary valideerimissõnum

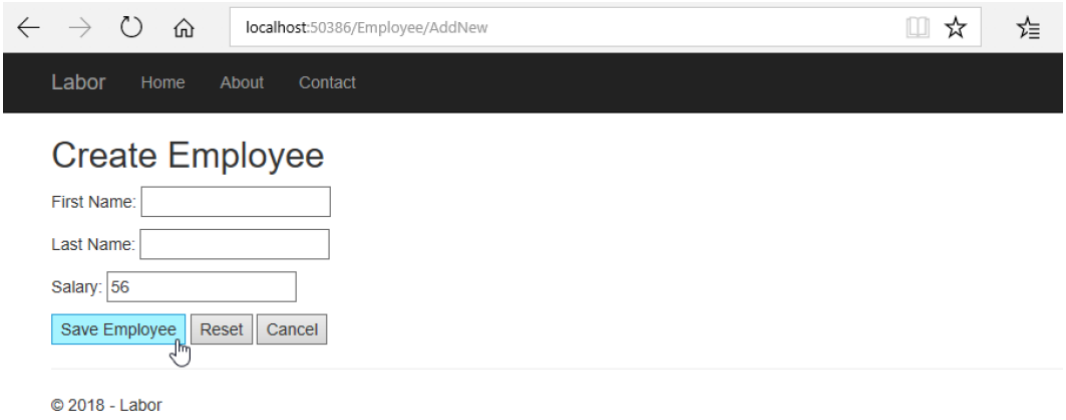
```
<div>
    <form action="/Employee/SaveEmployee" method="post">
        <p>First Name: <input type="text" id="TxtFName" name="FirstName" value="@Model.FirstName"/>
            @Html.ValidationMessage("FirstName") </p>
        <p>Last Name: <input type="text" id="TxtLName" name="LastName" value="@Model.LastName"/>
            @Html.ValidationMessage("LastName")</p>
        <p>Salary: <input type="text" id="TxtSalary" name="Salary" value="@Model.Salary"/>
            @Html.ValidationMessage("Salary")</p>
        <input type="submit" name="BtnSubmit" value="Save Employee" />
        <input type="button" name="BtnReset" value="Reset" onclick="ResetForm()"/>
        <input type="submit" name="BtnSubmit" value="Cancel" />
    </form>
</div>
```

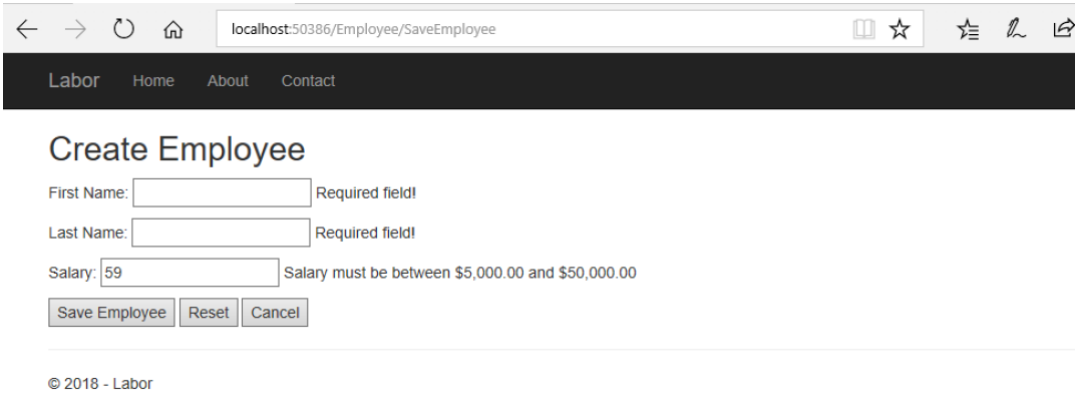
7. Muuda AddNew() action meetodit

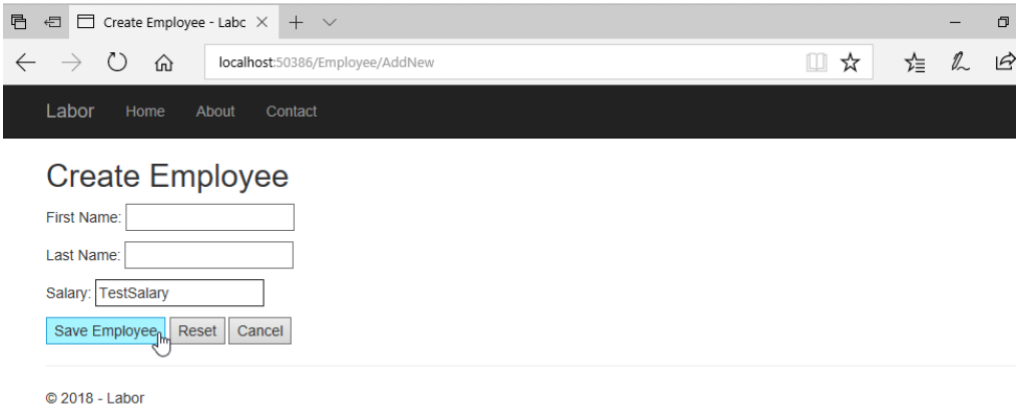
```
public ActionResult AddNew()
{
    return View("CreateEmployee", new CreateEmployeeViewModel());
}
```

8. Vajuta uuesti F5 klahvi ja testi rakendust

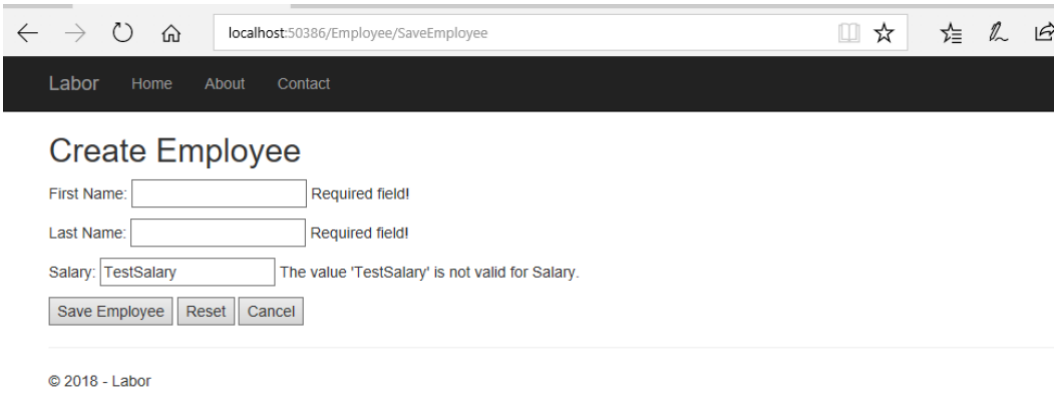
a.





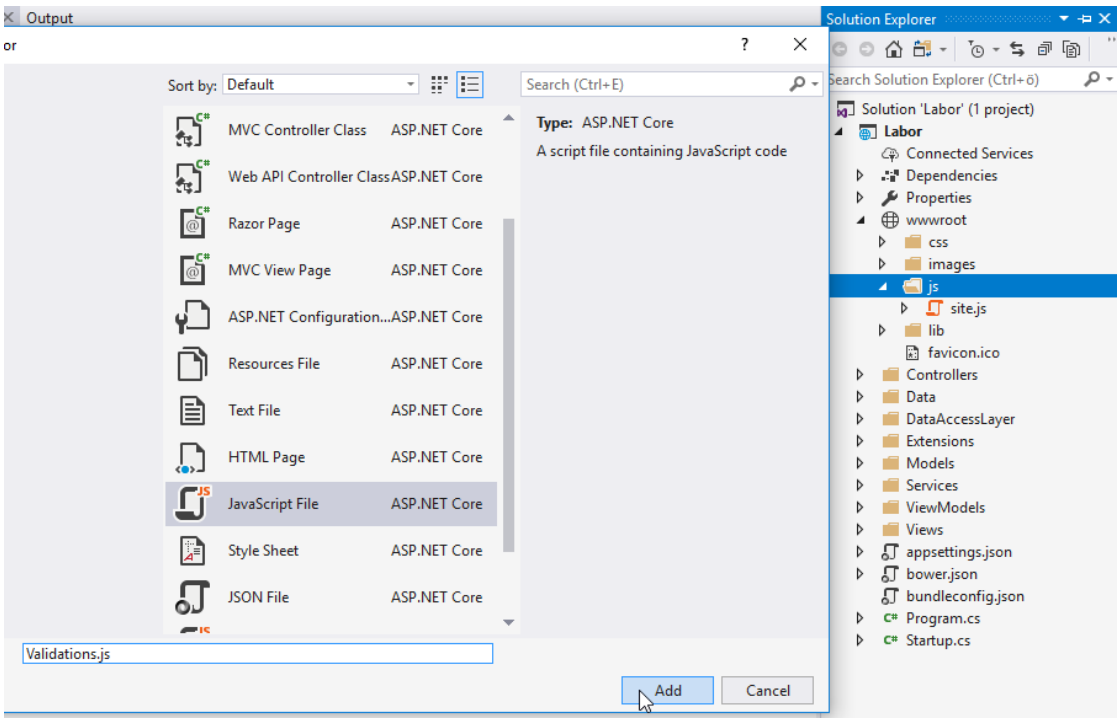

Create Employee - Labc
localhost:50386/Employee/AddNew
Labor
Home
About
Contact
Create Employee
First Name:
Last Name:
Salary: TestSalary
Save Employee
Reset
Cancel
© 2018 - Labor

b.


localhost:50386/Employee/SaveEmployee
Labor
Home
About
Contact
Create Employee
First Name: Required field!
Last Name: Required field!
Salary: TestSalary The value 'TestSalary' is not valid for Salary.
Save Employee
Reset
Cancel
© 2018 - Labor

Labor 16 - *Kliendi poolse valideerimise lisamine*

1. Lisa wwwroot/js kausta uus NewItem ning vali tüübiks JavaScript File, pane sellele nimeks Validations.js



2. Lisa Validations.js faili funktsioonid.

```
function IsFieldEmpty() {
    if ((document.getElementById("TxtFName").value == "")
        || (document.getElementById("TxtLName").value == "")
        || (document.getElementById("TxtSalary").value == "")) {
        return "Required field!";}
    return "";
}

function ContainsOnlyLetters() {
    var letters = /^[a-zA-Z]*$/;
    if ((document.getElementById("TxtFName").value.match(letters))
        && (document.getElementById("TxtLName").value.match(letters))) {
        return "";
    }
    return "Use only letters!";
}

function LengthIsValid() {
    if ((document.getElementById("TxtFName").value.length > 20)
        || (document.getElementById("TxtLName").value.length > 20)) {
        return "Length should be less than 20 characters!";
    }
    return "";
}

var minSalary = 500;
var maxSalary = 50000;
function IsSalaryInValid() {
    if ((isNaN(document.getElementById("TxtSalary").value))
        || ((document.getElementById("TxtSalary").value) < minSalary)
        || ((document.getElementById("TxtSalary").value) > maxSalary)) {
        return "Salary must be between " + minSalary + " and " + maxSalary + "!";
    } return "";
}


function IsValid() {
    var FieldEmptyEmptyMessage = IsFieldEmpty();
    var OnlyLettersMessage = ContainsOnlyLetters();
    var LengthIsValidMessage = LengthIsValid();
    var SalaryInvalidMessage = IsSalaryInValid();
    var FinalErrorMessage = "Errors:";

    if (FieldEmptyEmptyMessage != "")
        FinalErrorMessage += "\n" + FieldEmptyEmptyMessage;
    if (OnlyLettersMessage != "")
        FinalErrorMessage += "\n" + OnlyLettersMessage;
    if (LengthIsValidMessage != "")
        FinalErrorMessage += "\n" + LengthIsValidMessage;
    if (SalaryInvalidMessage != "")
        FinalErrorMessage += "\n" + SalaryInvalidMessage;
    if (FinalErrorMessage != "Errors:") {
        alert(FinalErrorMessage);
        return false;
    } else {
        return true;
    }
}
```

3. Lisa viide Validations.js failile CreateEmployee vaate päisesse

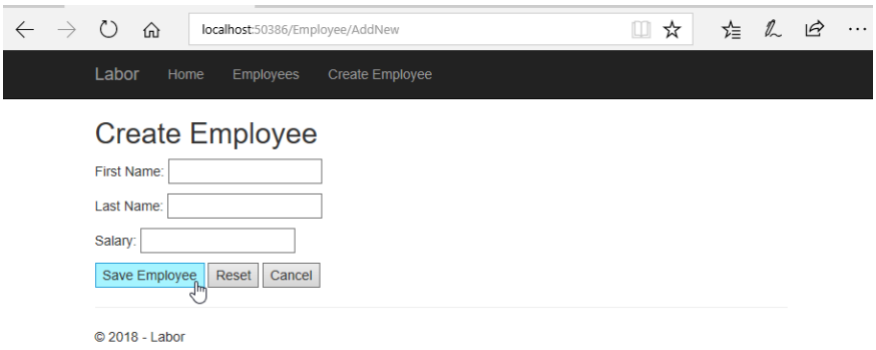
```
@{
    ViewBag.Title = "Create Employee";
    <script src="~/js/Validations.js"></script>
    <script>
        function ResetForm() {
            document.getElementById('TxtFName').value = "";
            document.getElementById('TxtLName').value = "";
            document.getElementById('TxtSalary').value = "";
        }
    </script>
}
```

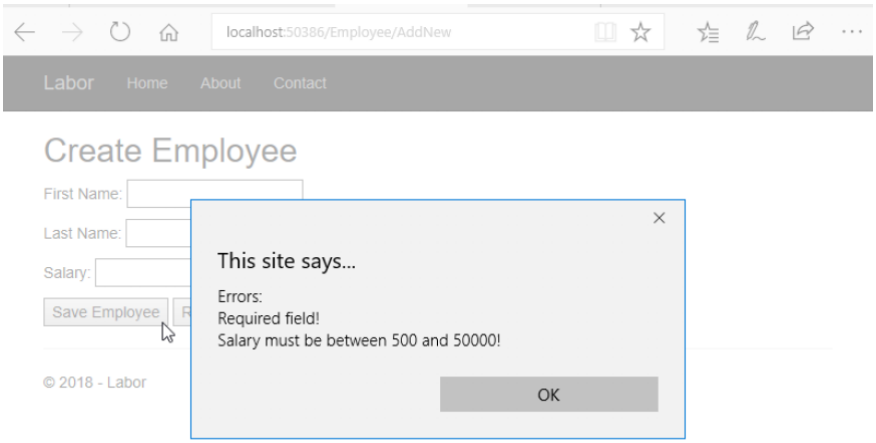
4. Save Employee vajutamisel kutsu välja JavaScripti funktsioon IsValid()

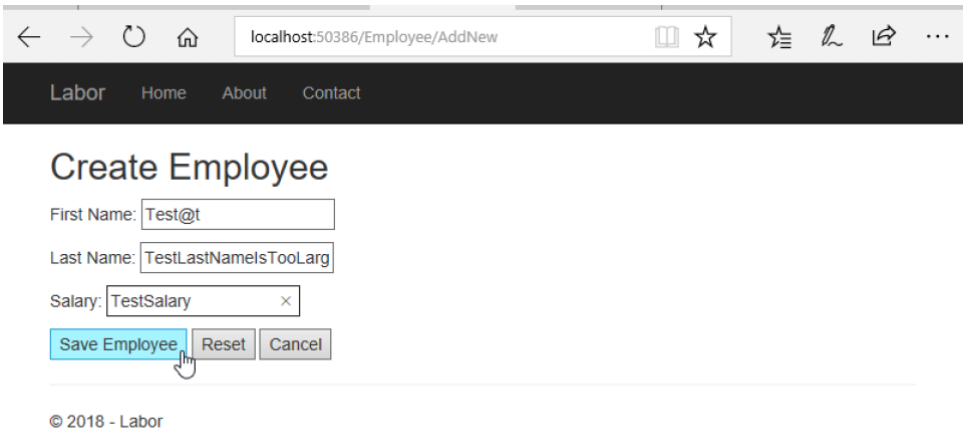
```
<td colspan="2">
    <input type="submit" name="BtnSubmit" value="Save Employee" onclick="return IsValid();" />
    <input type="submit" name="BtnSubmit" value="Cancel" />
    <input type="button" name="BtnReset" value="Reset" onclick="ResetForm();" />
</td>
```

5. Vajuta uuesti F5 klahvi ja testi rakendust

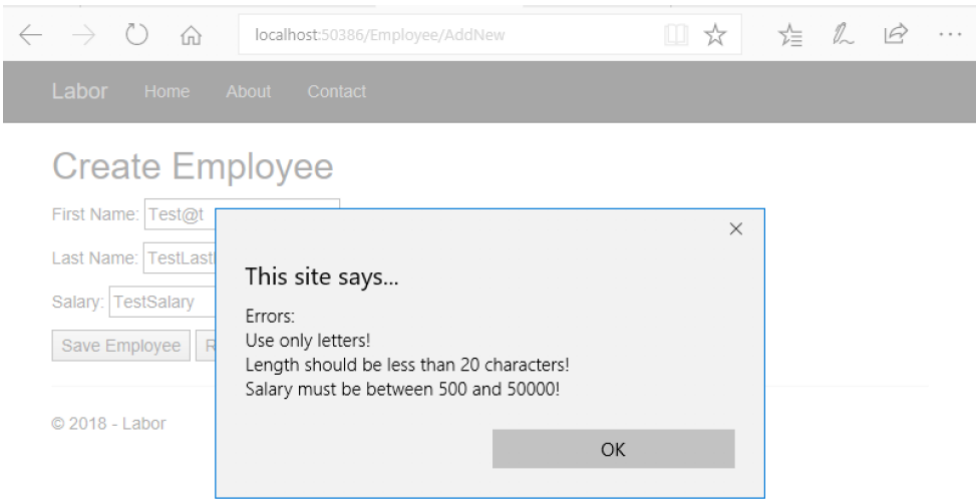
a.






localhost:50386/Employee/AddNew
Labor
Home
About
Contact
Create Employee
First Name: Test@t
Last Name: TestLastNameIsTooLarg
Salary: TestSalary
Save Employee
Reset
Cancel
© 2018 - Labor

b.


localhost:50386/Employee/AddNew
Labor
Home
About
Contact
Create Employee
First Name: Test@t
Last Name: TestLast
Salary: TestSalary
Save Employee
© 2018 - Labor
This site says...
Errors:
Use only letters!
Length should be less than 20 characters!
Salary must be between 500 and 50000!
OK

Labor 17 - Autentimise lisamine

1. Loo uus kontroller nimega AuthenticationController Controllers kausta, mis oleks järgmise sisuga

```
namespace Labor.Controllers
{
    public class AuthenticationController : Controller
    {
        public ActionResult Login()
        {
            return View();
        }
    }
}
```

2. Loo uus klass UserDetails Core projekti, mis sisaldaks UserName ja Password omadusi

```
    public class UserDetails
    {
        public string UserName { get; set; }
        public string Password { get; set; }
    }
}
```

3. Ava Infra projekti all olev Employees klass ning lisa sinna meetod IsValidUser()

```
public static bool IsValidUser(UserDetails user)
{ return user.UserName == "Admin" && user.Password == "Admin"; }
```

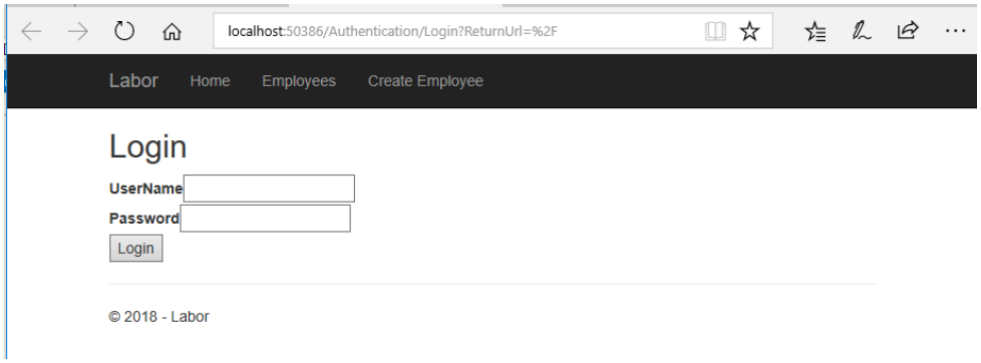
4. Lisa AuthenticationController-ile meetod DoLogin(), mis näeb välja järgmine

```
[HttpPost]
public async Task<IActionResult> DoLogin(UserDetails u)
{
    if (Employees.IsValidUser(u))
    {
        await setIdentity(u);
        return RedirectToAction("Index", "Employee");
    }
    ModelState.AddModelError("CredentialError", "Invalid Username or Password");
    return View("Login");
}

private async Task setIdentity(UserDetails u)
{
    List<Claim> claims = new List<Claim> { new Claim(ClaimTypes.Name, u.UserName) };
    ClaimsIdentity identity = new ClaimsIdentity(claims, "cookie");
    ClaimsPrincipal principal = new ClaimsPrincipal(identity);
    await HttpContext.SignInAsync(
        scheme: "AuthScheme",
        principal: principal);
}
```

5. Loo uus kaust nimega Authentication Views kausta, millesse lisa uus vaade nimega Login. Muuda vaade tugevalt trükitud UserDetails tüüpi vaateks ning lisa vaatele järgmine sisu

```
@model Core.UserDetails
@{
    ViewBag.Title = "Login";
}
<h2>Login</h2>
<div>
    @Html.ValidationMessage("CredentialError", new { style = "color:red;" })
    @using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
    {
        @Html.LabelFor(c => c.UserName)
        @Html.TextBoxFor(x => x.UserName)
        <br />
        @Html.LabelFor(c => c.Password)
        @Html.PasswordFor(x => x.Password)
        <br />
        <input type="submit" name="BtnSubmit" value="Login" />
    }
</div>
```

6. Vajuta klahvi F5 ja testi loodud vaadet



7. Muuda Startup.cs faili ConfigureServices meetodit järgnevalt

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SalesDbContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"),
        b => b.MigrationsAssembly("Labor")));

    services.AddAuthentication("AuthScheme")
        .AddCookie("AuthScheme", options =>
    {
        options.LoginPath = new PathString("/Authentication/Login");
    });

    services.AddMvc();
}
```

8. Muuda Startup.cs failis olevat Configure meetodit lisades sinna autentimise kasutamise.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseBrowserLink();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }

    app.UseStaticFiles();

    app.UseAuthentication();

    app.UseMvc(routes =>
    {
        routes.MapRoute(
            name: "default",
            template: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

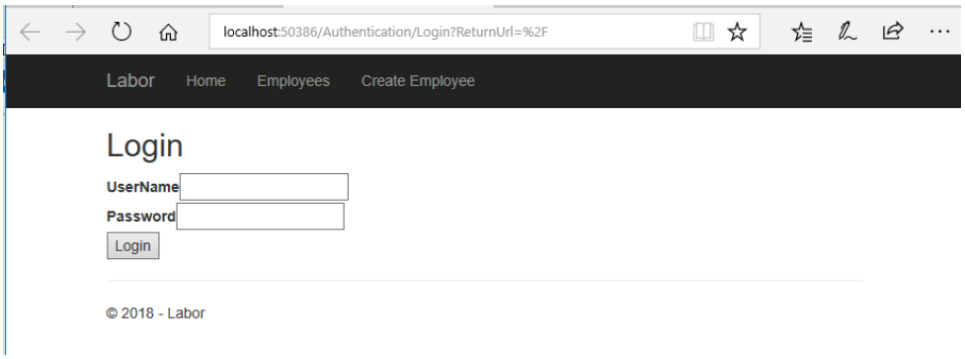
9. Muuda Home kontrolleris asuvad meetodid turvaliseks- lisa autentimine. Tee sama ka Employee kontrolleris asuvate Index ja AddNew meetoditega.

```
[Authorize]
public IActionResult Index()...
[Authorize]
public IActionResult About()...
[Authorize]
public IActionResult Contact()...
```

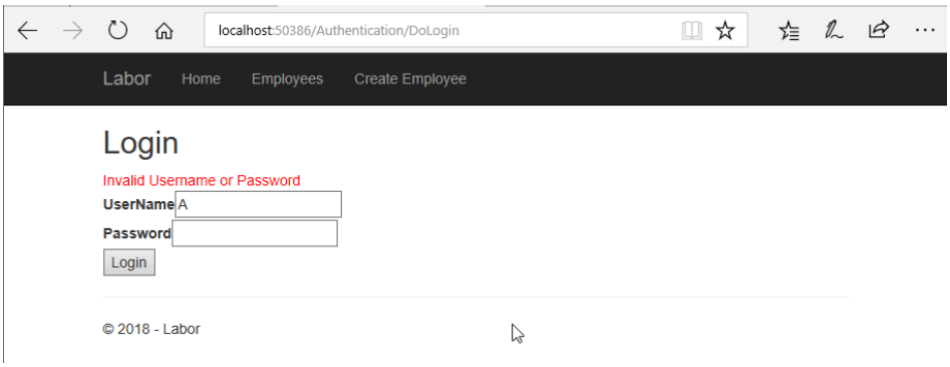
```
[Authorize]
public ActionResult Index()...
[Authorize]
public ActionResult AddNew()...
```

10. Vajuta F5 klahvi ja test rakendust

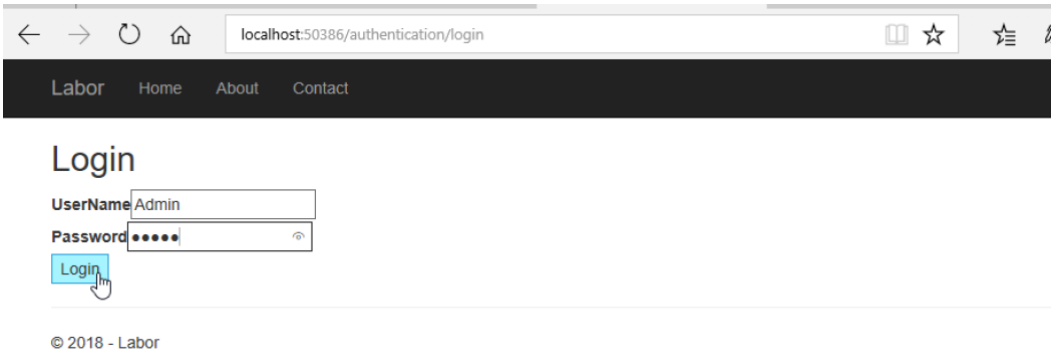
    a. Navigeeri URL-ile lõpuga /Employee/Index ja seejärel /Employee/AddNew

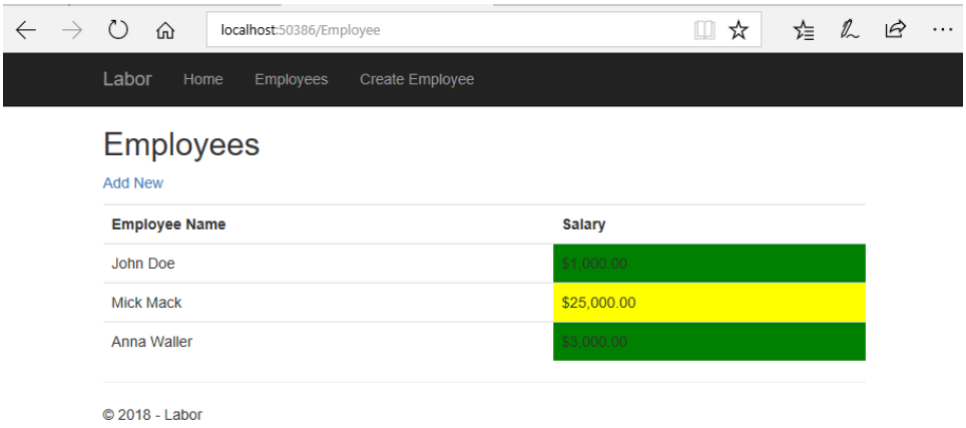


    b. Sisesta vale kasutajanimi



    c. Logi sisse ja kontrolli, kas suunamine toimib





Loe lisaks: https://docs.microsoft.com/en-us/aspnet/core/security/authentication/cookie?view=aspnetcore-2.1&tabs=aspnetcore2x

Labor 18 - *Väljalogimine*

1. Lisa EmployeeListViewModel-i sisse UserName omadus

```
public class EmployeeListViewModel
{
    public List<EmployeeViewModel> Employees { get; set; }
    public string UserName { get; set; }
}
```

2. Väärtusta UserName omadus EmployeeControlleri Index meetodis

```
var model = new EmployeeListViewModel();
model.UserName = User.Identity.Name;
```

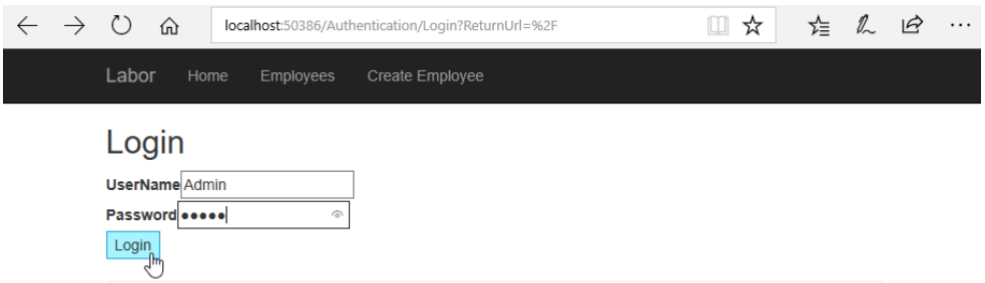
3. Ava Views/Employee kaustas asuv Index.cshtml fail ja loo link väljalogimiseks ning lisa kasutaja nime kuvamine.

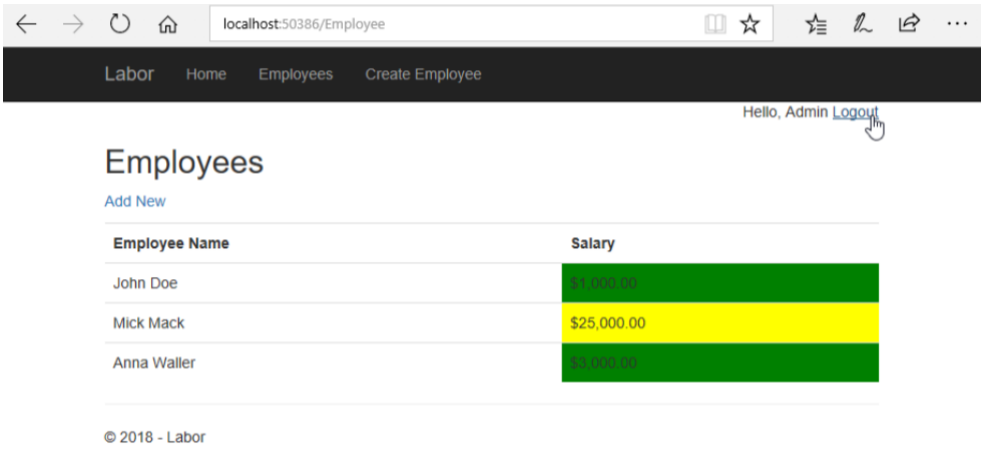
```
<div style="text-align: right"> Hello, @Model.UserName
    <a href="/Authentication/Logout">Logout</a>
</div>
```

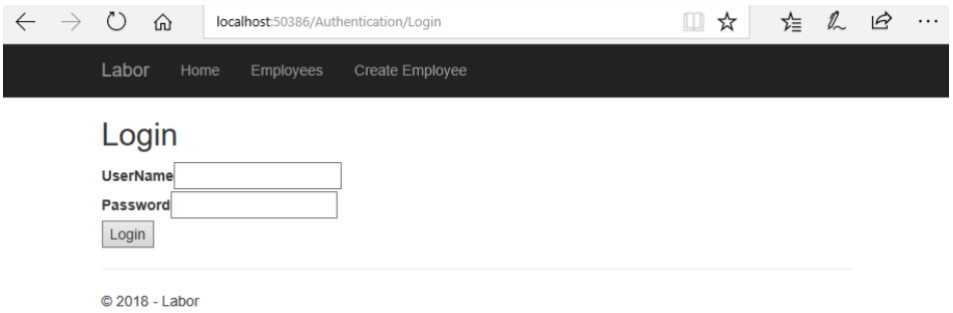
4. Loo Logout() action meetod Authentication kontrollerisse

```
public async Task<IActionResult> Logout()
{
    await HttpContext.SignOutAsync(
        scheme: "AuthScheme");
    return RedirectToAction("Login");
}
```

5. Vajuta F5 ja testi välja logimist. Logout lingile vajutades, toimub suunamine tagasi sisse logimise lehele.


localhost:50386/Authentication/Login?ReturnUrl=%2F
Labor
Home
Employees
Create Employee
Login
UserName
Admin
Password
Login
© 2018 - Labor


localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Admin Logout
Employees
Add New
Employee Name
Salary
John Doe
$1,000.00
Mick Mack
$25,000.00
Anna Waller
$3,000.00
© 2018 - Labor


localhost:50386/Authentication/Login
Labor
Home
Employees
Create Employee
Login
UserName
Password
Login
© 2018 - Labor

Labor 19 – Valideerimisreeglid sisse logimisel

1. Lisa Core projekti uus klass nimega UserValidation, mis pärineb NameValidation klassist

```
namespace Core
{
    public class UserValidation: NameValidation
    {
        protected const string requiredLength = "UserName length should be between 2 and 7";
        protected override ValidationResult IsValid(object value, ValidationContext validationContext)
        {
            if (value == null) return error(requiredField);
            var s = value.ToString();
            if (s.Length < 2) return error(requiredLength);
            if (s.Length > 7) return error(requiredLength);
            if (!onlyLetters(s)) return error(useOnlyLetters);
            return ValidationResult.Success;
        }
    }
}
```

2. Lisa UserValidation UserDetails klassis olevale UserName omadusele.

```
public class UserDetails
{
    [UserValidation]
    public string UserName { get; set; }
    public string Password { get; set; }
}
```

3. Muuda Login.cshtml faili, et kuvada valideerimise veateated vaates.

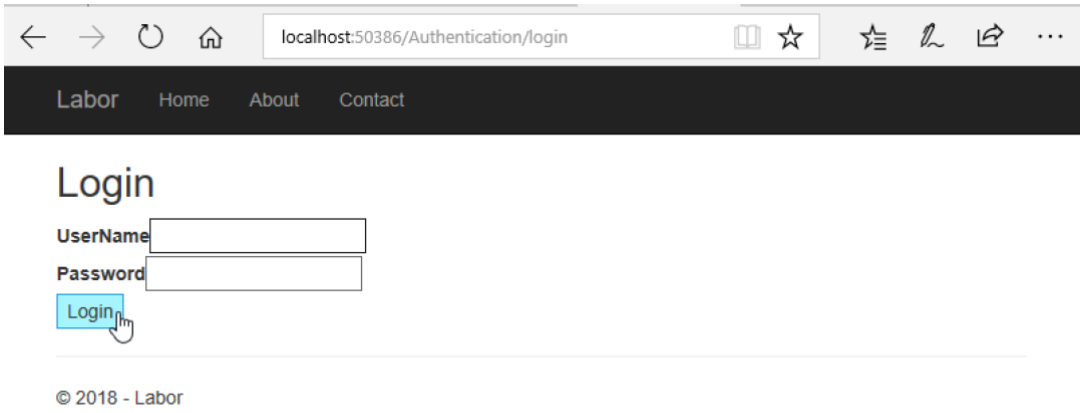
```
<body>
    <div>
        @Html.ValidationMessage("CredentialError", new { style = "color:red;" })
    @using (Html.BeginForm("DoLogin", "Authentication", FormMethod.Post))
            {
        @Html.LabelFor(c => c.UserName)
        @Html.TextBoxFor(x => x.UserName)
        @Html.ValidationMessageFor(x => x.UserName)
        <br />
                @Html.LabelFor(c => c.Password)
                @Html.PasswordFor(x => x.Password)
                <br />
                <input type="submit" name="BtnSubmit" value="Login" />
    }
    </div>
```

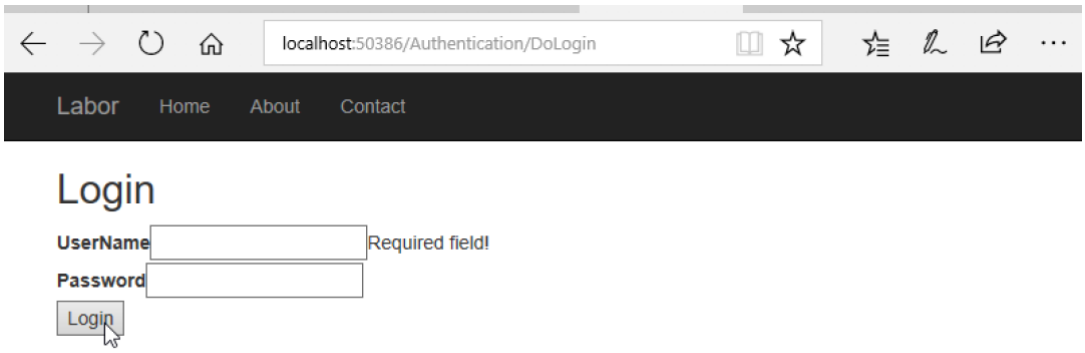
Seekord on kasutatud Html.ValidationMessageFor meetodit, mida saab kasutada vaid kindla tüübiga vaadete puhul.

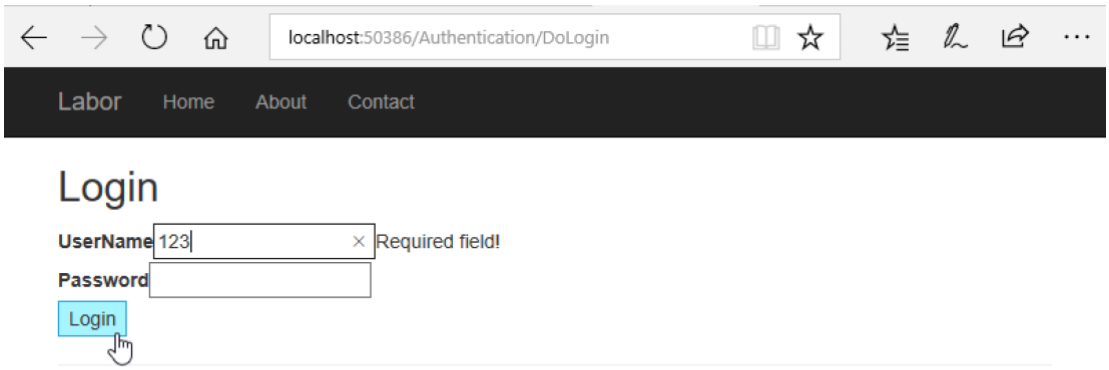
4. Muuda DoLogin action meetodit- lisa valideerimine ka sinna

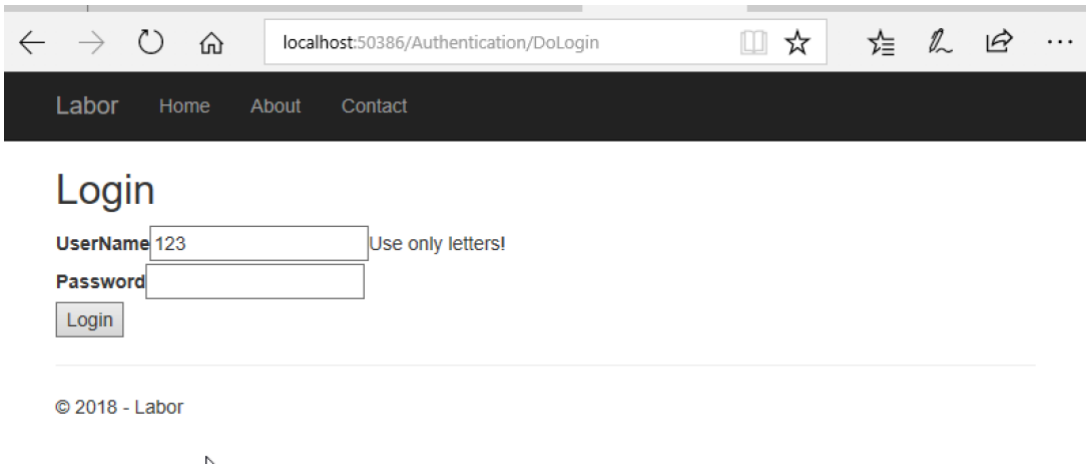
```
[HttpPost]
public async Task<IActionResult> DoLogin(UserDetails u)
{
    if (ModelState.IsValid)
    {
        if (Employees.IsValidUser(u))
        {
            await setIdentity(u);
            return RedirectToAction("Index", "Employee");
        }
        ModelState.AddModelError("CredentialError", "Invalid Username or Password");
        return View("Login");
    }
    return View("Login");
}
```

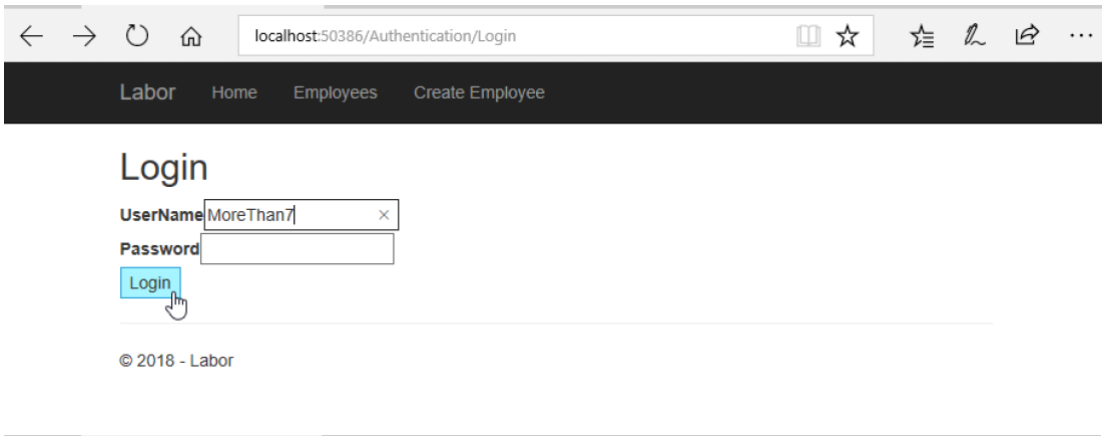
5. Vajuta F5 ja testi rakendust

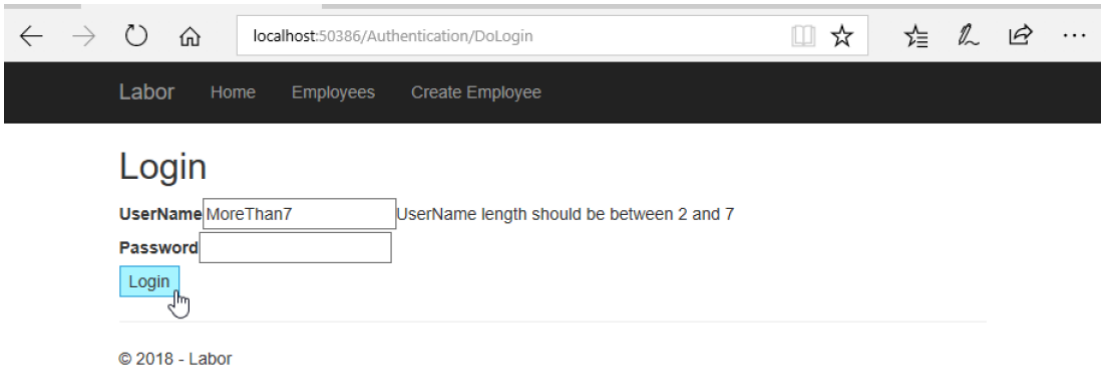



localhost:50386/Authentication/DoLogin
Labor
Home
About
Contact
Login
UserName
Required field!
Password
Login
© 2018 - Labor


localhost:50386/Authentication/DoLogin
Labor
Home
About
Contact
Login
UserName
123
Required field!
Password
Login
© 2018 - Labor

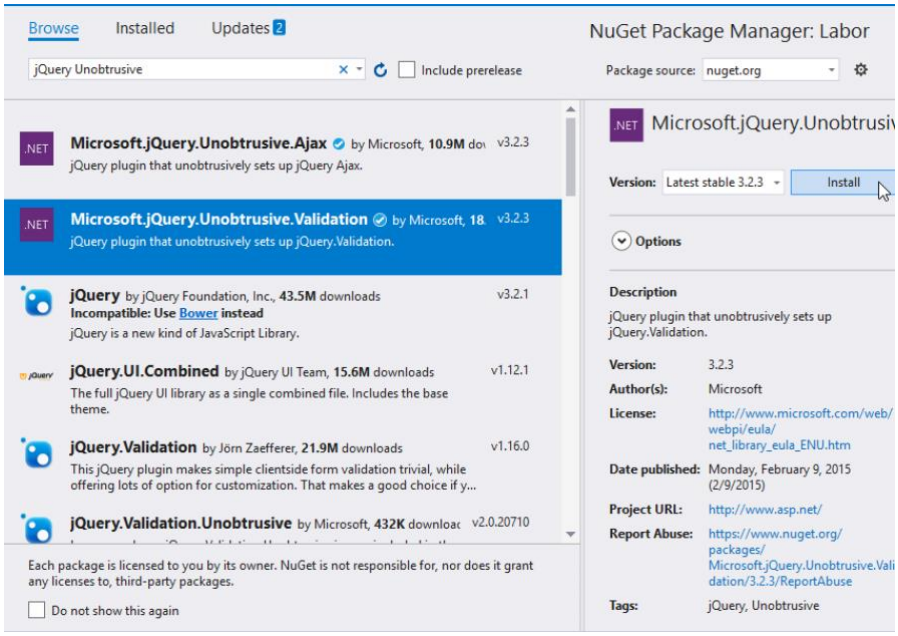

localhost:50386/Authentication/DoLogin
Labor
Home
About
Contact
Login
UserName
123
Use only letters!
Password
Login
© 2018 - Labor


localhost:50386/Authentication/Login
Labor
Home
Employees
Create Employee
Login
UserName
MoreThan7
Password
Login
© 2018 - Labor


localhost:50386/Authentication/DoLogin
Labor
Home
Employees
Create Employee
Login
UserName
MoreThan7
UserName length should be between 2 and 7
Password
Login
© 2018 - Labor

Labor 20 – Kliendipoolse valideerimise lisamine sisse logimisel

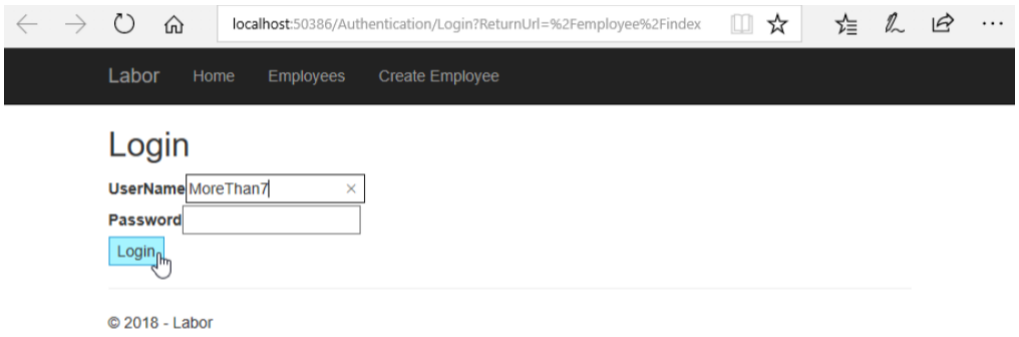
1. Lisa projekti Labor '*jQuery Unobtrusive Validation'* NutGet pakett

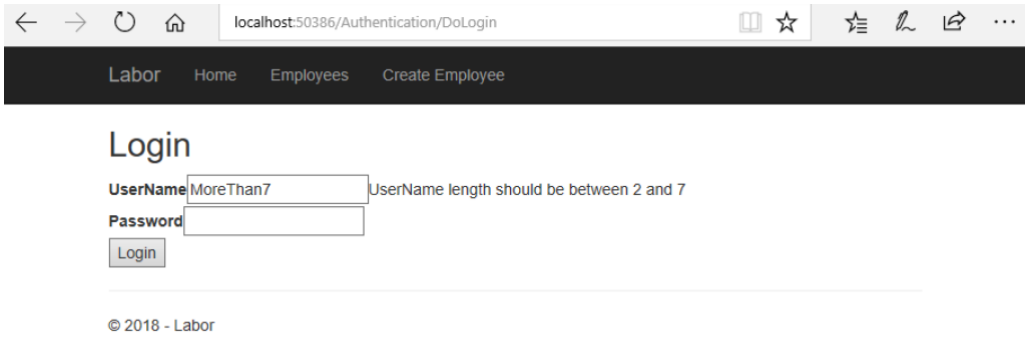


2. Lisa jQuery valideerimise failid Login.cshtml vaatesse

```
@{
    ViewBag.Title = "Login";
    <script src="~/lib/jquery/"></script>
    <script src="~/lib/jquery-validation/"></script>
    <script src="~/lib/jquery-validation-unobtrusive/"></script>
}
```

3. Vajuta F5 ja testi rakendust





Loe rohkem *jQuery Unobtrusive Validation-i* kohta*:* https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-2.1

Labor 21 – Jaluse lisamine

Osaline vaade (*Partial view*)

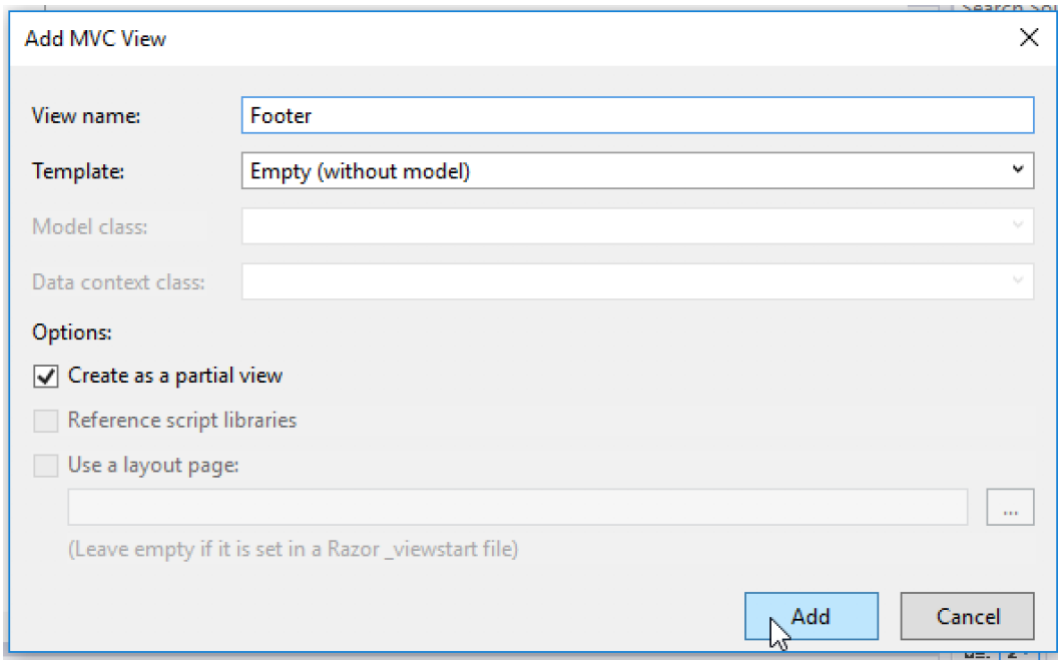
- Vaade, mida saab taaskasutada ja mida ei kuvata kunagi otseselt.
- Lisatakse teiste vaadete sisse ja kuvatakse osana sellest vaatest.

1. Loo ViewModels kausta uus klass FooterViewModel osalise vaate jaoks

```
namespace Labor.ViewModels
{
    public class FooterViewModel
    {
        public string CompanyName { get; set; }
        public string Year { get; set; }
    }
}
```

2. Tee hiire parema sõrmise klikk /Views/Shared kaustal ning vali Add -> View

   NB! Nimeta vaade 'Footer'-iks ning märgi linnuke *Create as a partial view* ees olevasse kasti. Vajuta *Add*



3. Ava Footer.cshtml fail ning lisa sellele järgmine sisu

```
@using Labor.ViewModels
@model FooterViewModel

    <div style="text-align:right; background-color: silver;
        color: darkcyan;border:1px solid gray;
        margin-top: 2px; padding-right: 10px">

        @Model.CompanyName © @Model.Year

    </div>
```

4. Ava EmployeeListViewModel klass ja lisa uus omadus, mis hoiaks jaluse andmeid

```
public class EmployeeListViewModel
{
    public List<EmployeeViewModel> Employees { get; set; }
    public string UserName { get; set; }
    public FooterViewModel FooterData { get; set; }
}
```

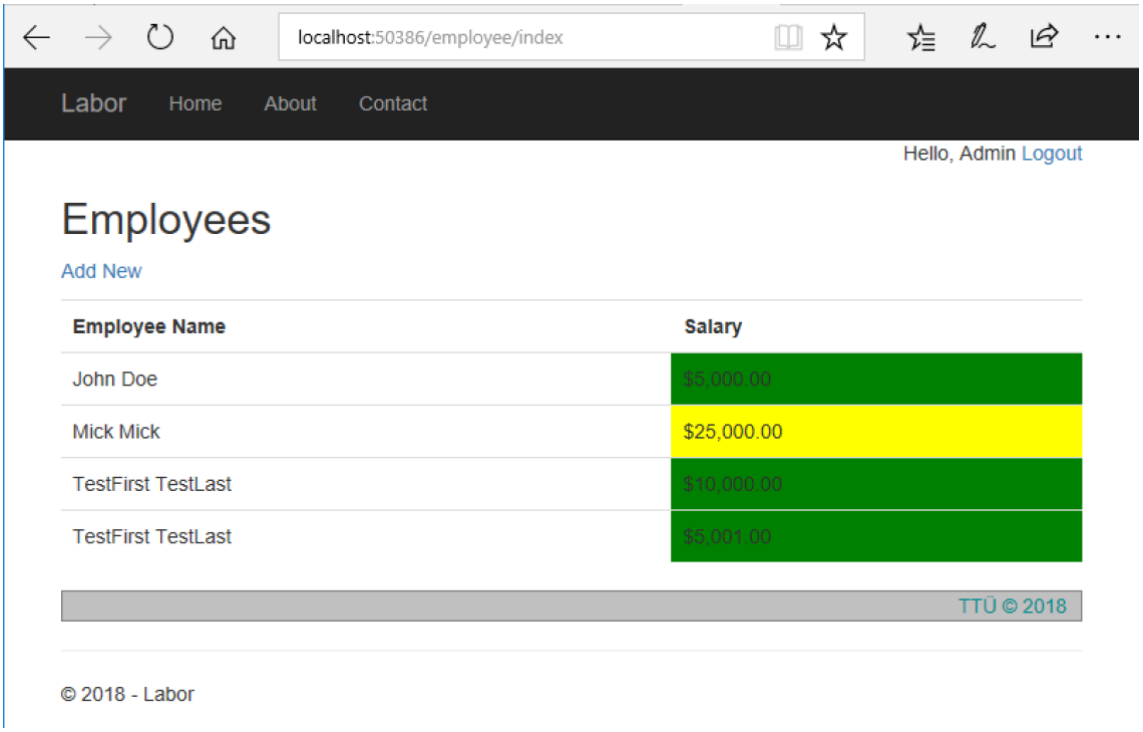
5. Ava Employee kontroller ja lisa väärtusta FooterData omadus Index action meetodis järgnevalt

```
[Authorize]
public ActionResult Index()
{
    var model = new EmployeeListViewModel();
    model.UserName = User.Identity.Name;
    var employees = Employees.Get(db);
    var list = new List<EmployeeViewModel>();
    foreach (var e in employees)...
    model.Employees = list;
    model.FooterData = new FooterViewModel();
    model.FooterData.CompanyName = "TTÜ";
    model.FooterData.Year = DateTime.Now.Year.ToString();
    return View("Index", model);
}
```

6. Ava Views/Employee kaustas olev Index.cshtml ning kuva jalus vahetult pärast töötajate tabelit

```
@{
    Html.RenderPartial("Footer", Model.FooterData);
}
```

7. Vajuta F5 ja testi rakendust



Labor 22 – Rolli põhine turvatus

8. Peida *AddNew* link mitte-admin kasutajate vaatest

   a. Lisa startup.cs faili ConfigureServices meetodisse sessiooni jaoks vajalikud parameetrid.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SalesDbContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"),
        b => b.MigrationsAssembly("Labor")));

    services.AddAuthentication("AuthScheme")
        .AddCookie("AuthScheme", options =>
    {
        options.LoginPath = new PathString("/Authentication/Login");
    });
    services.AddDistributedMemoryCache();
    services.AddSession();

    services.AddMvc();
}
```

   b. Lisa startup.cs faili Configure meetodisse sessiooni kasutus. NB! See peab kindlasti asuma enne app.UseMvc määramist.

```
app.UseAuthentication();
app.UseSession();

app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

   c. Lisa kasutajaks ka mitte-admin kasutaja

```
public static bool IsValidUser(UserDetails user)
{
    if (user.UserName == "Admin" && user.Password == "Admin") return true;
    if (user.UserName == "Mari" && user.Password == "Mets") return true;
    return false;
}
```

d. Muuda DoLogin() meetodit, lisa sisselogimisel sessiooni võtmele (*SessionKeyName*) väärtus.

```
[HttpPost]
public async Task<IActionResult> DoLogin(UserDetails u)
{
    if (ModelState.IsValid)
    {
        if (Employees.IsValidUser(u))
        {
            await setIdentity(u);
            HttpContext.Session.SetString("SessionKeyName", u.UserName);
            return RedirectToAction("Index", "Employee");
        }
        ModelState.AddModelError("CredentialError", "Invalid Username or Password");
        return View("Login");
    }
    return View("Login");
}
```

e. Loo uus klass Views -> Employee kausta nimega NewLinkViewComponent.cs, mis pärineb ViewComponent klassist ning on järgmise sisuga

```
public class NewLinkViewComponent: ViewComponent
{
    public async Task<IViewComponentResult> InvokeAsync()
    {
        if (HttpContext.Session.GetString("SessionKeyName") == "Admin") return View("AddNewLink");
        return Content(string.Empty);
    }
}
```

f. Loo *Views -> Shared* kausta uus kaust *Components*, mille sisse loo omakorda kaust *NewLink.* Loo sellesse kausta uus osaline vaade nimega AddNewLink. Vaadet luues vaata, et märgiksid linnukese '*Create as a partial view'* ees olevasse kasti. Vaate sisu on järgmine:

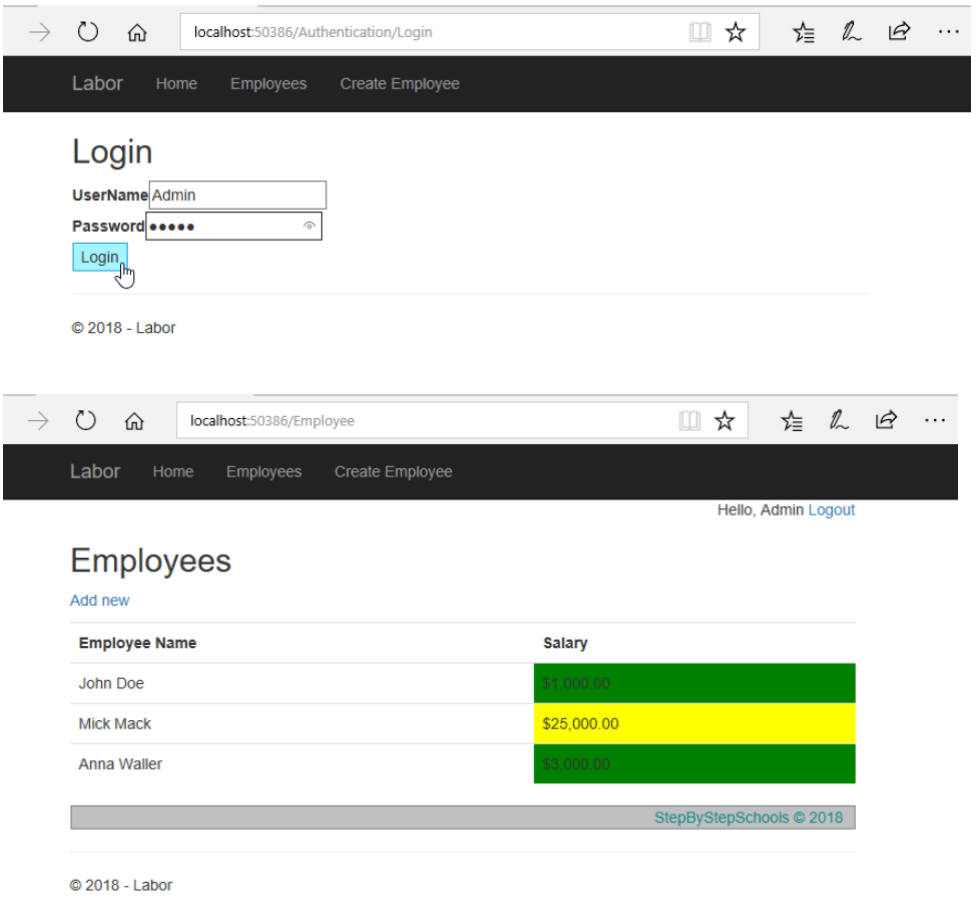
```
<a href="/Employee/AddNew">Add new</a>
```

g. Ava Views-> Employee kaustas asuv Index.cshtml fail ning eemalda sealt *Add new* link. Lisa selle asemel järgmine kood

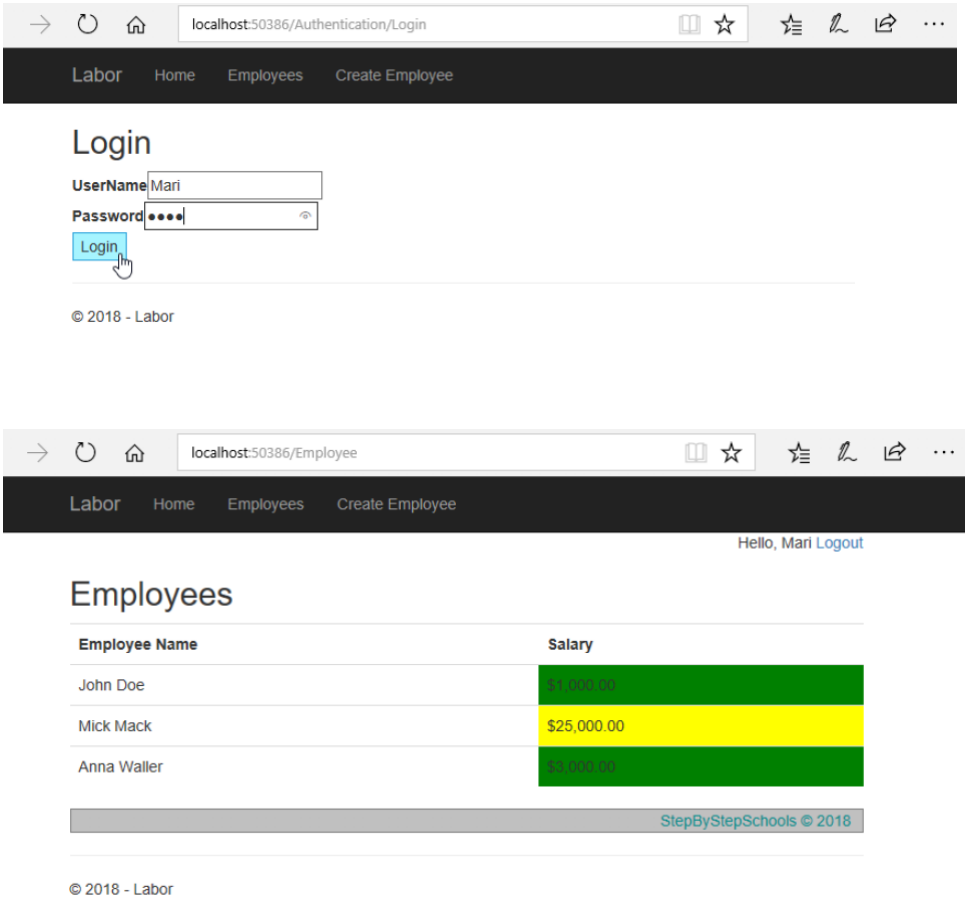
```
<p>
    @await Component.InvokeAsync("NewLink")
</p>
```

h. Vajuta F5 ja testi rakendust

1.


localhost:50386/Authentication/Login
Labor
Home
Employees
Create Employee
Login
UserName
Admin
Password
Login
© 2018 - Labor
localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Admin Logout
Employees
Add new
Employee Name
Salary
John Doe
$1,000.00
Mick Mack
$25,000.00
Anna Waller
$3,000.00
StepByStepSchools © 2018
© 2018 - Labor

2.


localhost:50386/Authentication/Login
Labor
Home
Employees
Create Employee
Login
UserName
Mari
Password
Login
© 2018 - Labor
localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Mari Logout
Employees
Employee Name
Salary
John Doe
$1,000.00
Mick Mack
$25,000.00
Anna Waller
$3,000.00
StepByStepSchools © 2018
© 2018 - Labor

9. Eemalda ka võimalus mitte-admin kasutajatel lisada töötajad navigeerides otse URL-ile Employee/AddNew. (Ava rakendus ja veendu, et see on hetkel võimalik)

    a. Loo projekti Labor alla uus kaust nimega ‘Filters’, mille sisse loo uus klass AdminFilter. Kui klass loodud, muuda seda nii, et see pärineks ActionFilterAttribute klassist

```
public class AdminFilter: ActionFilterAttribute
{

}
```

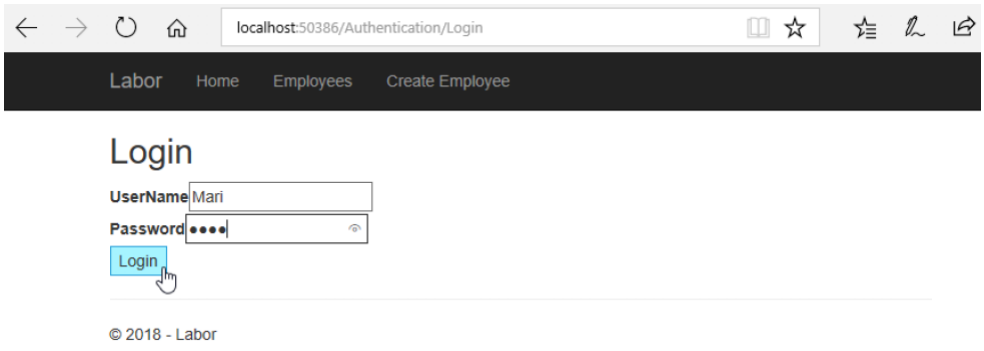
    b. Loo AdminFilter klassi *override* meetod *OnActionExecuting*, mis näeb välja järgmine

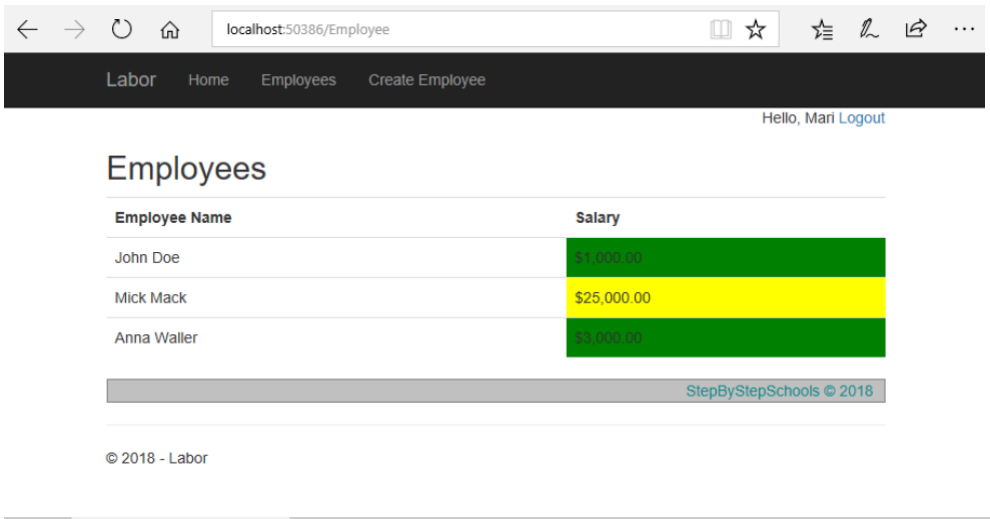
```
public class AdminFilter: ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        if (!(filterContext.HttpContext.Session.GetString("SessionKeyName") == "Admin"))
        {
            filterContext.Result = new ContentResult()
            {
                Content = "Unauthorized to access specified resource."
            };
        }
    }
}
```

    c. Lisa filter AddNew ja SaveEmployee meetoditele Employee kontrolleris

```
[Authorize]
[AdminFilter]
public ActionResult AddNew()...

[AdminFilter]
public ActionResult SaveEmployee(Employee e, string BtnSubmit)...
```

    d. Vajuta klahvi F5 ning testi rakendust logides sisse mitte-admin kasutajana ning seejärel navigeeri URL-ile employee/AddNew





Unauthorized to access specified resource.

Labor 23 – CSRF rünnaku vältimine

1. Lisa Employee kontrollerisse SaveEmployee meetodile ValitadeAntiForgeryToken atribuut.

```
[AdminFilter]
[ValidateAntiForgeryToken]
public ActionResult SaveEmployee(Employee e, string BtnSubmit)
{
    if (BtnSubmit != "Save Employee") return RedirectToAction("Index");
    if (!ModelState.IsValid) return Create(e);
    return save(e);
}
```

2. Lisa CreateEmployee vaatesse AntiForgeryToken valideerimisreegel.

```
<h2>Create Employee</h2>
    <div>
        <form action="/Employee/SaveEmployee" method="post">
            @Html.AntiForgeryToken()
            <p>First Name: <input type="text" id="TxtFName" name="FirstName" value="@Model.FirstName" />
            @Html.ValidationMessage("FirstName")</p>
            <p>Last Name: <input type="text" id="TxtLName" name="LastName" value="@Model.LastName" />
            @Html.ValidationMessage("LastName")</p>
            <p>Salary: <input type="text" id="TxtSalary" name="Salary" value="@Model.Salary" />
            @Html.ValidationMessage("Salary")</p>
            <input type="submit" name="BtnSubmit" value="Save Employee" onclick="return IsValid();" />
            <input type="button" name="BtnReset" value="Reset" onclick="ResetForm()" />
            <input type="submit" name="BtnSubmit" value="Cancel" />
        </form>
    </div>
```

Labor 24 – Kirjete kustutamine

1. Loo Employee kontrollerisse uus Action meetod Delete

```
public async Task<IActionResult> Delete(int? id)
{
    if (id == null) return NotFound();
    Employee employee = db.Employees.Find(id);
    if (employee == null) return NotFound();
    return View("Delete", employee);
}
```

2. Lisa EmployeeViewModel-ile EmployeeId omadus

```
public class EmployeeViewModel
{
    public EmployeeViewModel(Employee emp)
    {
        setName(emp);
        setSalary(emp);
        setColor(emp);
    }
    public int EmployeeId { get; set; }
    public string EmployeeName { get; set; }
    public string Salary { get; set; }
    public string SalaryColor { get; private set; } = "red";
```

3. Lisa Views/Employee/Index vaatesse link Delete lehele

```
@foreach (var item in Model.Employees)
{
    <tr>
        <td>@item.EmployeeName</td>
        <td style="background-color: @item.SalaryColor">@item.Salary</td>
        <td>
            <a asp-action="Delete" asp-route-id="@item.EmployeeId">Delete</a>
        </td>
    </tr>
}
```

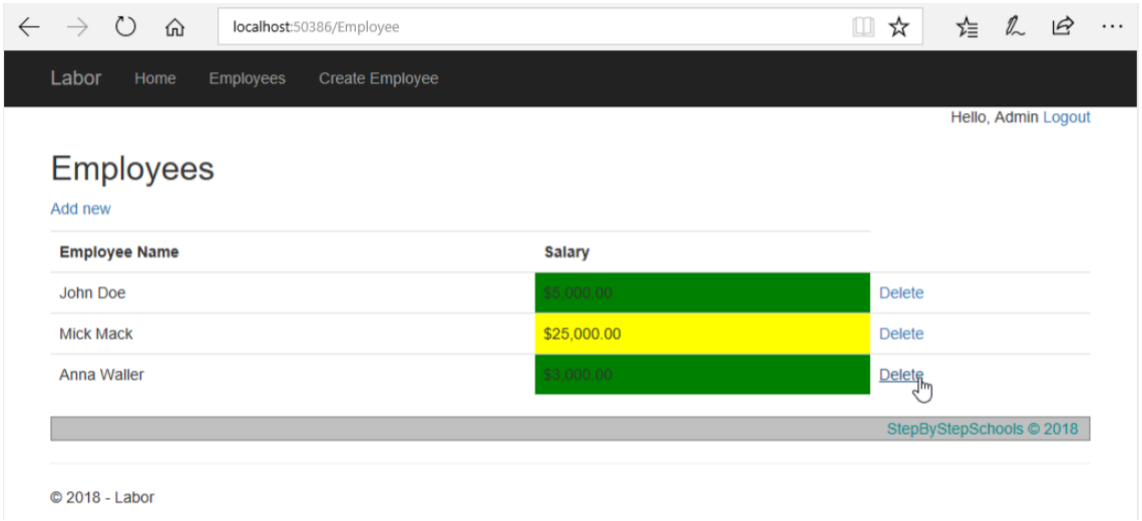
4. Loo Views/Employee kausta uus vaade nimega Delete järgmiselt

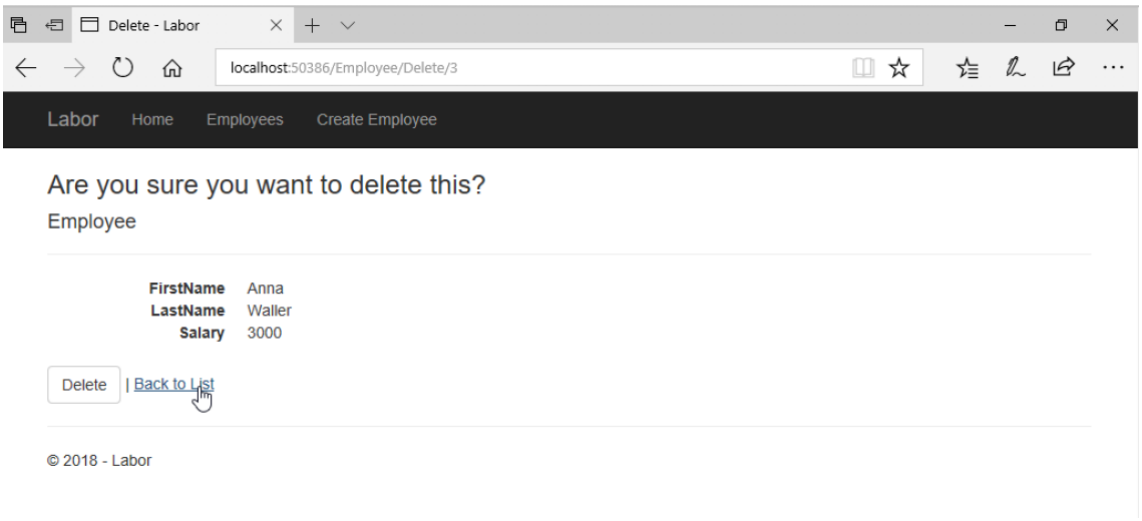
```
@model Core.Employee

@{
    ViewBag.Title = "Delete";
}
<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Employee</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.FirstName)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.FirstName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.LastName)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.LastName)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Salary)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Salary)
        </dd>
    </dl>
    <form asp-action="Delete">
        <input type="hidden" asp-for="EmployeeId" />
        <input type="submit" value="Delete" class="btn btn-default" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

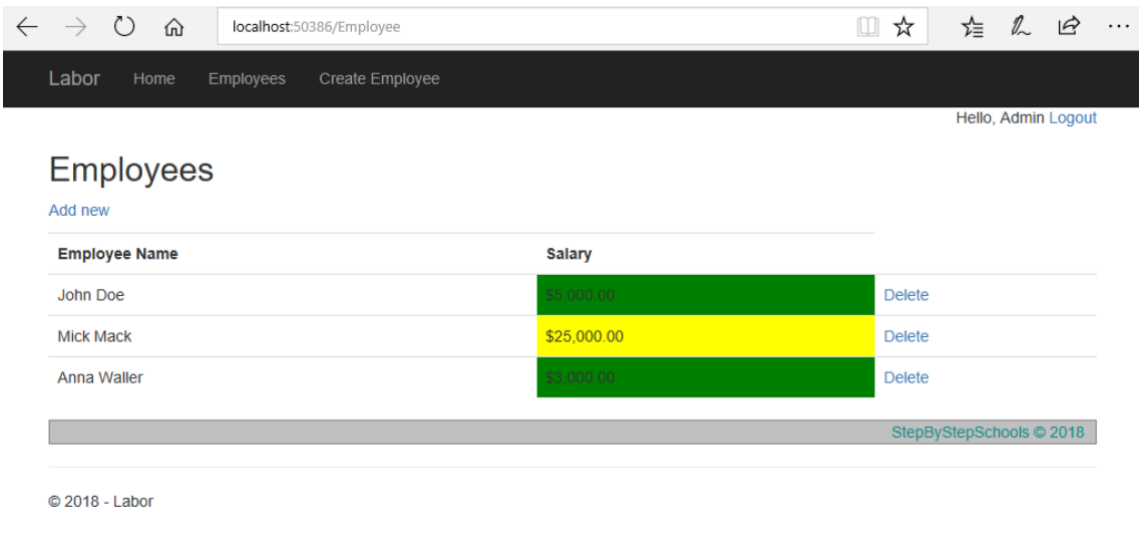
5. Väärtusta EmployeeId muutuja Index meetodis

```
foreach (var e in employees)
{
    var employee = new EmployeeViewModel(e);
    employee.EmployeeId = e.EmployeeId;
    list.Add(employee);
}
```

6. Vajuta klahvi F5 ning testi rakendust



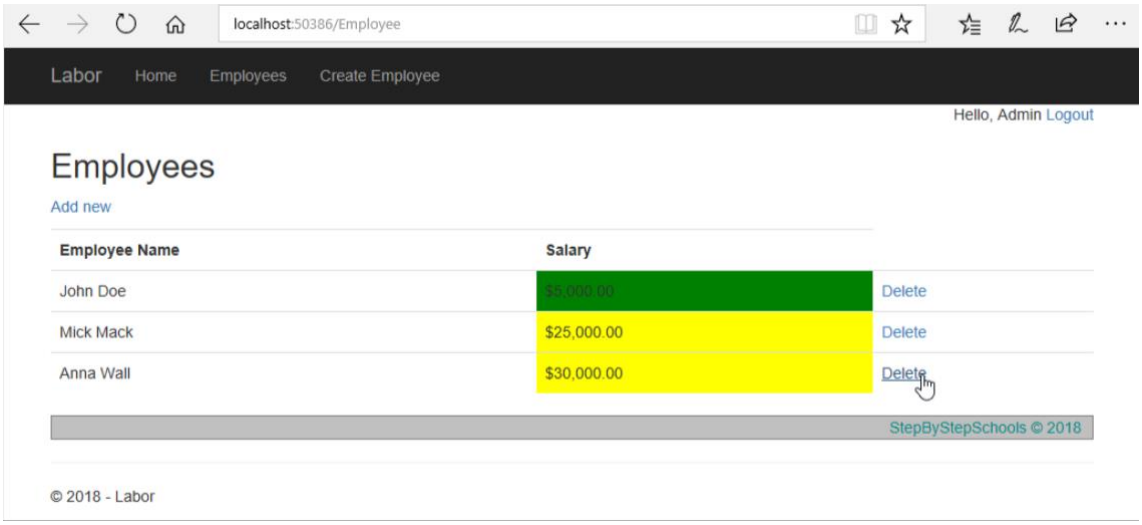


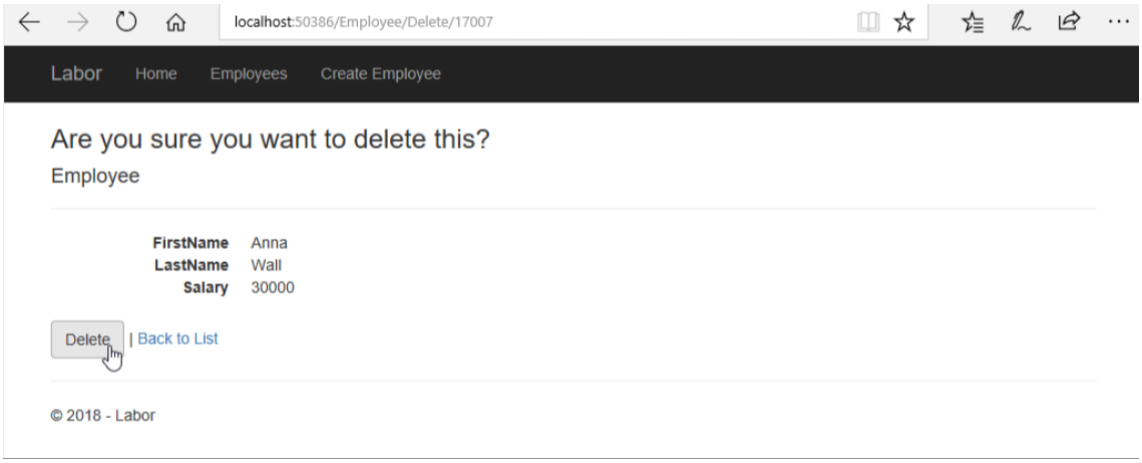


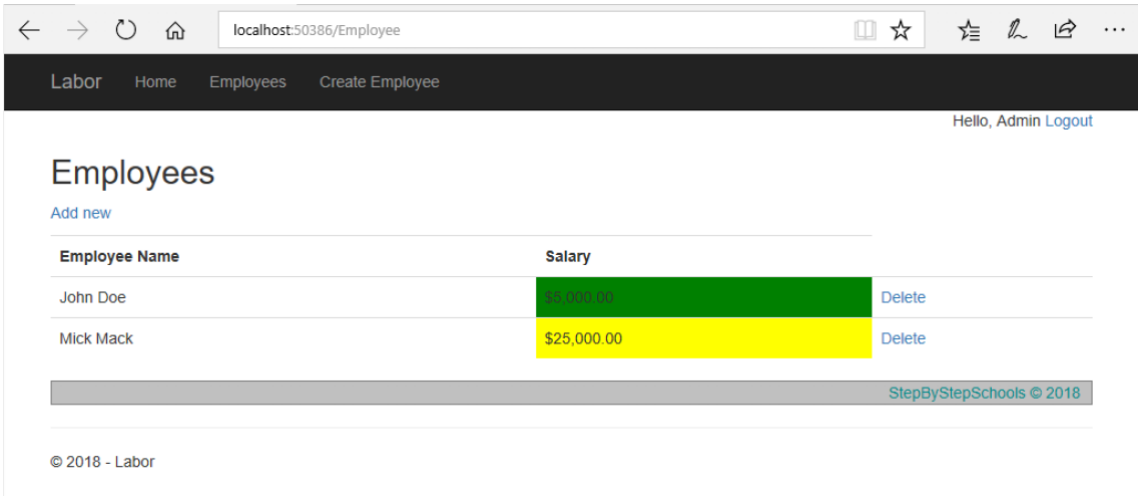
7. Lisa Employee kontrollerisse DeleteConfirmed action meetod

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var employee = await db.Employees.SingleOrDefaultAsync
        (m => m.EmployeeId == id);
    db.Employees.Remove(employee);
    await db.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

8. Vajuta F5 ja testi rakendust






localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Admin Logout
Employees
Add new
Employee Name
Salary
John Doe
$5,000.00
Delete
Mick Mack
$25,000.00
Delete
StepByStepSchools © 2018
© 2018 - Labor

Labor 25 – Kirjete muutmine

1. Loo Edit meetod Employee kontrollerisse

```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null) return NotFound();
    var employee = await db.Employees.SingleOrDefaultAsync(m => m.EmployeeId == id);
    if (employee == null) return NotFound();
    return View("Edit", employee);
}
```

2. Loo uus vaade Edit Employee kontrollerisse

```
@model Core.Employee
@{
    ViewBag.Title = "Edit";
    <script src="~/js/Validations.js"></script>
}
<h4>Employee</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="EmployeeId" />
            <div class="form-group">
                <label asp-for="FirstName" class="control-label"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName" class="control-label"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Salary" class="control-label"></label>
                <input asp-for="Salary" class="form-control" />
                <span asp-validation-for="Salary" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Back to List</a>
</div>
```

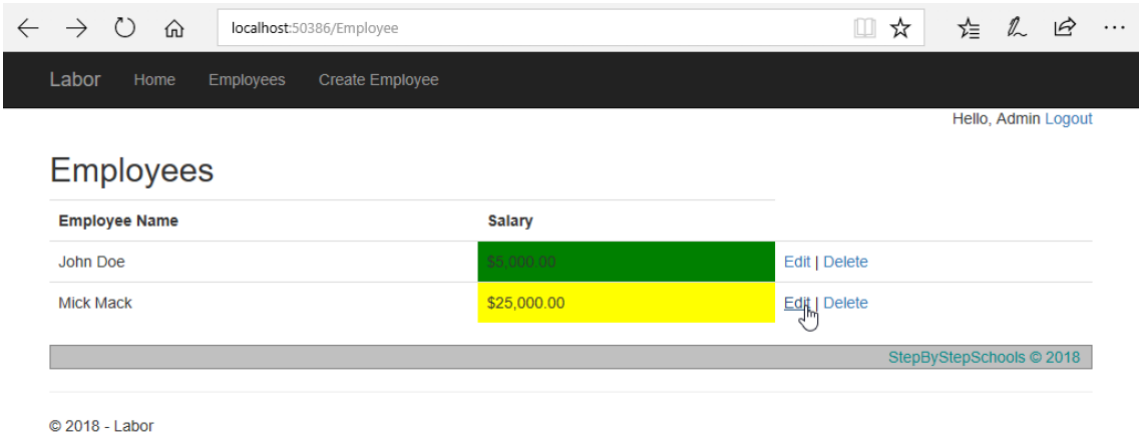
3. Lisa link Edit vaatesse

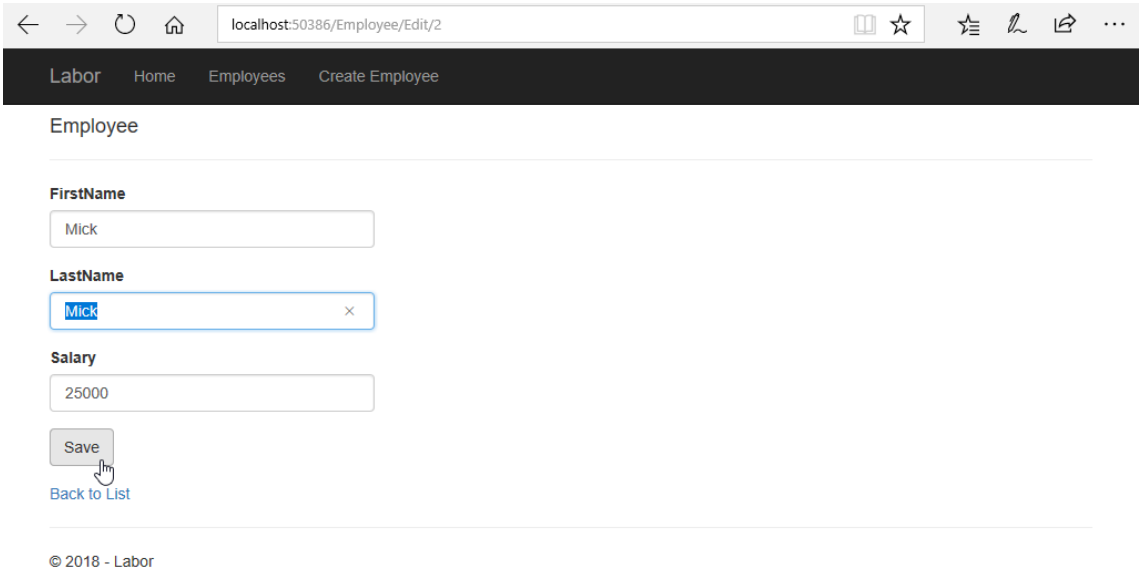
```
@foreach (var item in Model.Employees)
{
    <tr>
        <td>@item.EmployeeName</td>
        <td style="background-color: @item.SalaryColor">@item.Salary</td>
        <td>
            <a asp-action="Edit" asp-route-id="@item.EmployeeId">Edit</a> |
            <a asp-action="Delete" asp-route-id="@item.EmployeeId">Delete</a>
        </td>
    </tr>
}
```

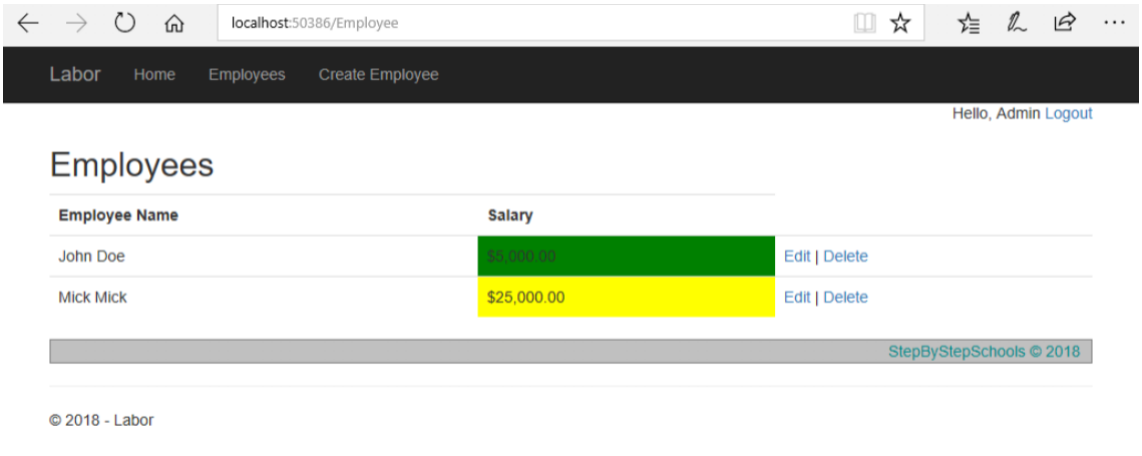
4. Lisa Edit meetod kirje muutmiseks Employee kontrollerisse

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
    [Bind("EmployeeId,FirstName,LastName,Salary")] Employee employee)
{
    if (id != employee.EmployeeId) return NotFound();
    if (ModelState.IsValid)
    {
        try
        {
            db.Update(employee);
            await db.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!EmployeeExists(employee.EmployeeId)) return NotFound();
                throw;
        }
        return RedirectToAction(nameof(Index));
    }
    return View(employee);
}

private bool EmployeeExists(int id)
{
    return db.Employees.Any(e => e.EmployeeId == id);
}
```

5. Vajuta klahvi F5 ning testi rakendust


localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Admin Logout
Employees
Employee Name
Salary
John Doe
$5,000.00
Edit | Delete
Mick Mack
$25,000.00
Edit | Delete
StepByStepSchools © 2018
© 2018 - Labor


localhost:50386/Employee/Edit/2
Labor
Home
Employees
Create Employee
Employee
FirstName
Mick
LastName
Mick
Salary
25000
Save
Back to List
© 2018 - Labor


localhost:50386/Employee
Labor
Home
Employees
Create Employee
Hello, Admin Logout
Employees
Employee Name
Salary
John Doe
$5,000.00
Edit | Delete
Mick Mick
$25,000.00
Edit | Delete
StepByStepSchools © 2018
© 2018 - Labor

Labor 26 – Failide ülesse laadimine

1. Loo uus klass nimega FileUploadViewModel projekti Labor.

```
namespace Labor
{
    public class FileUploadViewModel
    {
        public IFormFile FileToUpload { get; set; }
    }
}
```

2. Lisa uus meetod UploadEmployees Infra projektis asuvasse klassi Employees.

```
public void UploadEmployees(List<Employee> employees, SalesDbContext db)
{
    db.Employees.AddRange(employees);
    db.SaveChanges();
}
```

3. Loo uus kontroller nimega BulkUploadController Controllers kausta. Index meetod tagastab vaate faili ülesse laadimiseks. GetEmployees meetod võtab sisendiks üles laetud faili, avab selle lugemiseks ning leiab failist vastavad töötaja andmed. Kusjuures faili lugemisel on arvestatud, et faili esimene rida on päis. Upload meetod kasutab GetEmployees meetodit ning loob uue töötaja, salvestab töötaja andmebaasi ning tagastab Employee/Index vaate.

```
namespace Labor.Controllers
{
    public class BulkUploadController : Controller
    {
        private readonly SalesDbContext db;

        public BulkUploadController(SalesDbContext database)
        {
            db = database;
        }
        [AdminFilter]
        public ActionResult Index()
        {
            return View(new FileUploadViewModel());
        }

        [AdminFilter]
        public ActionResult Upload(IFormFile fileUpload)
        {
            List<Employee> employees = GetEmployees(fileUpload);
            Employees e = new Employees();
            e.UploadEmployees(employees, db);
            return RedirectToAction("Index", "Employee");
        }


    private List<Employee> GetEmployees(IFormFile file)
    {
        List<Employee> employees = new List<Employee>();
        StreamReader csvreader = new StreamReader(file.OpenReadStream());
        csvreader.ReadLine();
        while (!csvreader.EndOfStream)
        {
            var line = csvreader.ReadLine();
            var values = line.Split(',');
            Employee e = new Employee();
            e.FirstName = values[0];
            e.LastName = values[1];
            e.Salary = int.Parse(values[2]);
            employees.Add(e);
        }
        return employees;
    }
}
```

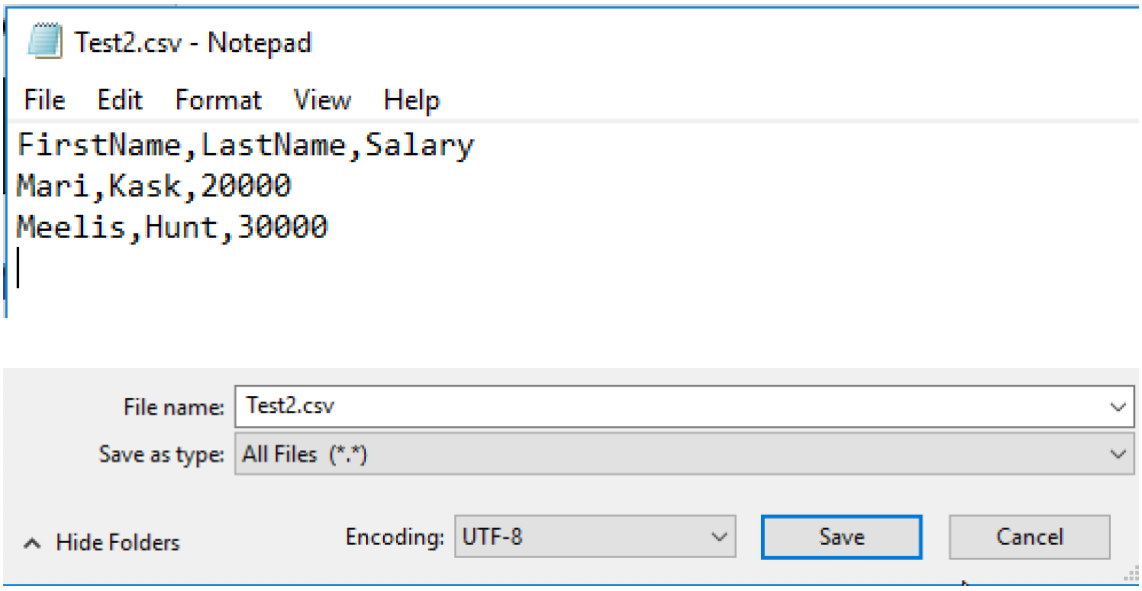
4. Lisa Views kausta uus kaust nimega BulkUpload ning sinna sisse omakorda uus vaade nimega Index.

```
@model FileUploadViewModel
@{
    ViewBag.Title = "Bulk Upload";
}
<h3>Bulk Upload</h3>
<div>
    <a href="/Employee/Index">Back</a>
    <form action="/BulkUpload/Upload" method="post" enctype="multipart/form-data"
        Select File : <input type="file" name="fileUpload" value="" />
        <input type="submit" name="name" value="Upload" />
    </form>
</div>
```

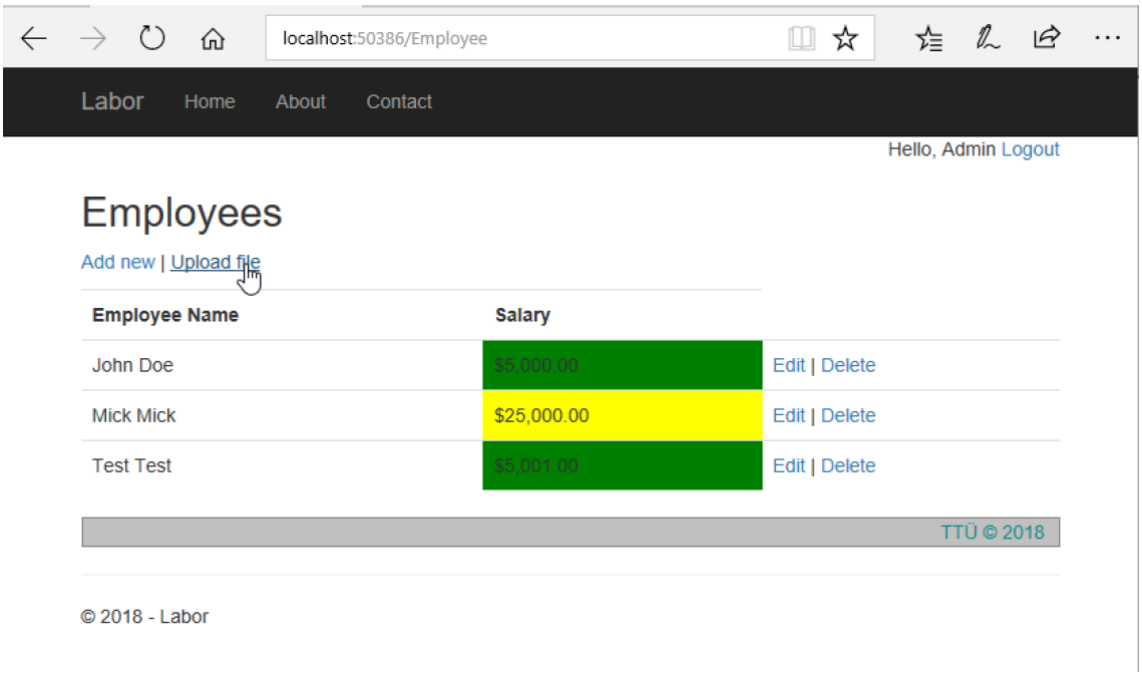
5. Ava Shared/Components/NewLink kaustas asuv AddNewLink osaline vaade ning lisa sinna link ka failide ülesse laadimise vaatele.

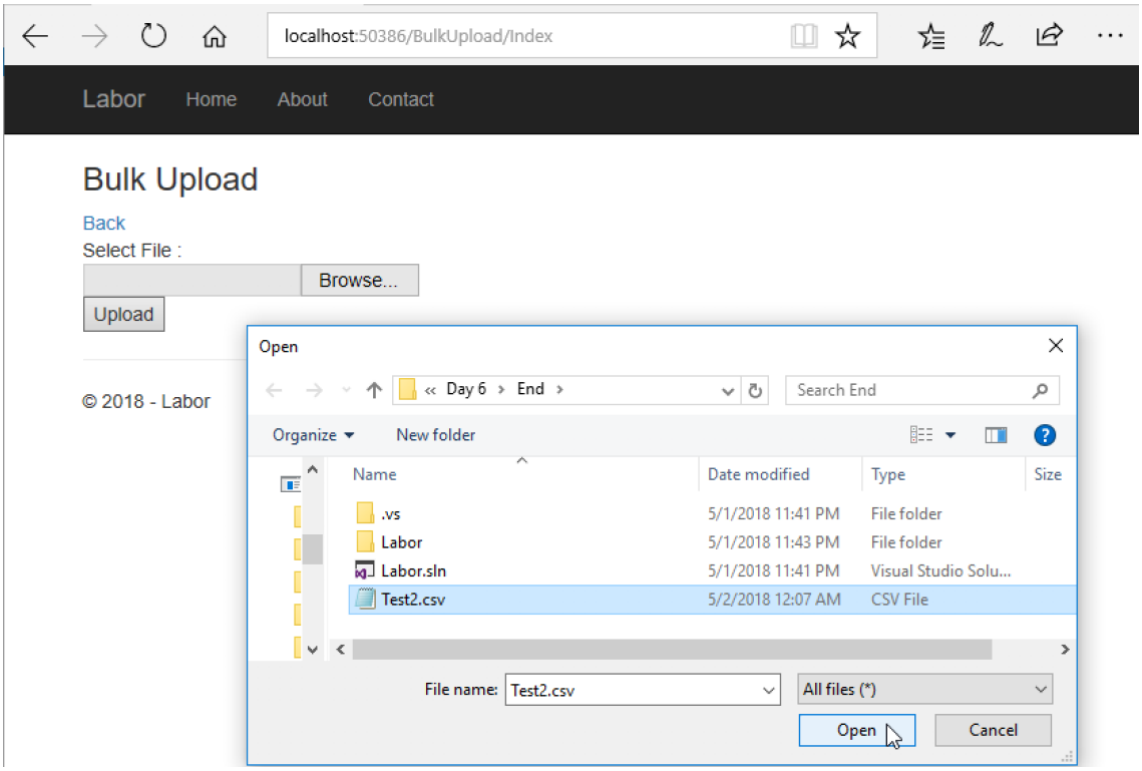
```
<a href="/Employee/AddNew">Add new</a>
<a href="/BulkUpload/Index">Upload file</a>
```

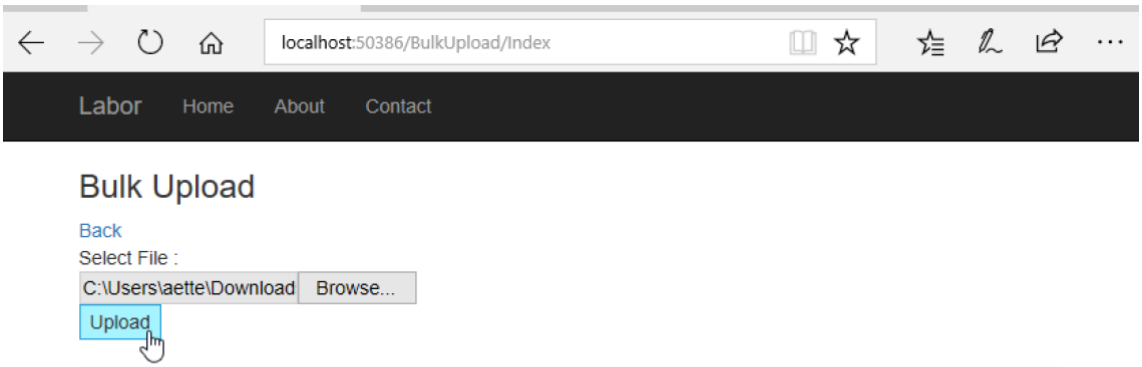
6. Loo uus .csv fail. Salvesta fail UTF-8 kodeeringus ning laiendiga .csv.

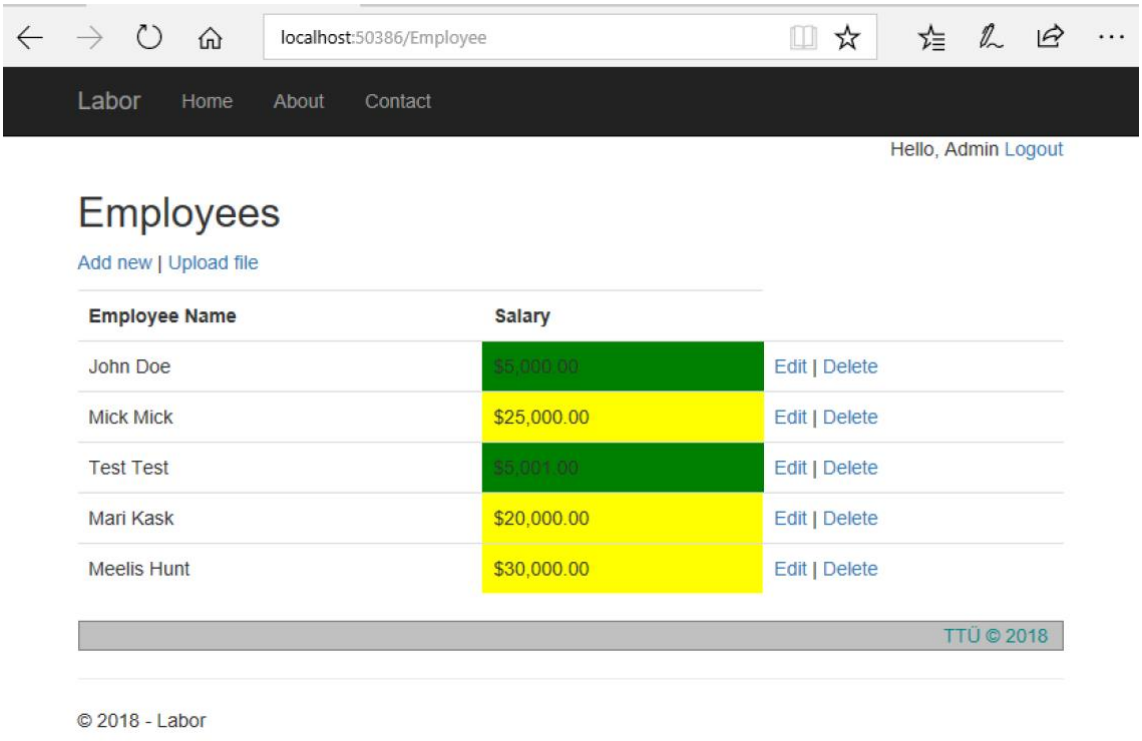


7. Testi failide ülesse laadimist vajutades klahvi F5.


localhost:50386/Employee
Labor
Home
About
Contact
Hello, Admin Logout
Employees
Add new | Upload file
Employee Name
Salary
John Doe
$5,000.00
Edit | Delete
Mick Mick
$25,000.00
Edit | Delete
Test Test
$5,001.00
Edit | Delete
TTÜ © 2018
© 2018 - Labor


localhost:50386/BulkUpload/Index
Labor
Home
About
Contact
Bulk Upload
Back
Select File :
Browse...
Upload
© 2018 - Labor
Open
Day 6 › End ›
Search End
Organize
New folder
Name
Date modified
Type
Size
.vs
5/1/2018 11:41 PM
File folder
Labor
5/1/2018 11:43 PM
File folder
Labor.sln
5/1/2018 11:41 PM
Visual Studio Solu...
Test2.csv
5/2/2018 12:07 AM
CSV File
File name:
Test2.csv
All files (*)
Open
Cancel


localhost:50386/BulkUpload/Index
Labor
Home
About
Contact
Bulk Upload
Back
Select File :
C:\Users\aette\Download
Browse...
Upload
© 2018 - Labor


localhost:50386/Employee
Labor
Home
About
Contact
Hello, Admin Logout
Employees
Add new | Upload file
Employee Name
Salary
John Doe
$5,000.00
Edit | Delete
Mick Mick
$25,000.00
Edit | Delete
Test Test
$5,001.00
Edit | Delete
Mari Kask
$20,000.00
Edit | Delete
Meelis Hunt
$30,000.00
Edit | Delete
TTÜ © 2018
© 2018 - Labor

Labor 27 – Testide kirjutamine

1. Loo uus klass nimega EmployeeTests Tests projekti alla ning kirjuta ühiktestid Employee klassi omadustele.

```
[TestClass]
public class EmployeeTests
{
    [TestMethod]
    public void FirstNameTest()
    {
        Employee e = new Employee("eesnimi", null, 0);
        Assert.AreEqual("eesnimi", e.FirstName);
    }

    [TestMethod]
    public void LasttNameTest()
    {
        Employee e = new Employee(null, "perenimi", 0);
        Assert.AreEqual("perenimi", e.LastName);
    }

    [TestMethod]
    public void SalaryTest()
    {
        Employee e = new Employee(null, null, 100);
        Assert.AreEqual(100, e.Salary);
    }
}
```

2. Ava FooterViewModel ning muuda seda järgnevalt

```
public class FooterViewModel
{
    public FooterViewModel(string company)
    {
        setCompanyName(company);
        setYear();
    }
    public string CompanyName { get; set; } = "TTÜ";
    public string Year { get; set; }

    internal void setCompanyName(string c)
    {
        CompanyName = c;
    }
    internal void setYear()
    {
        Year = DateTime.Now.Year.ToString();
    }
}
```

3. Ava Employee kontroller ning eemalda sealt jaluse väärtustamine

```
model.Employees = list;
model.FooterData = new FooterViewModel("TTÜ");//change this line
model.FooterData.CompanyName = "TTÜ";//remove this line
model.FooterData.Year = DateTime.Now.Year.ToString();//remove this line
return View("Index", model);
```

4. Lisa Tests projekti uus klass FooterViewModelTests ning kirjuta sinna testmeetodid.

```
[TestClass]
public class FooterViewModelTests
{
    private FooterViewModel o;

    [TestInitialize]
    public void TestInitialize()
    {
        o = new FooterViewModel(null);
    }
    [TestMethod]
    public void CompanyNameTest()
    {
        o.setCompanyName("TTÜ");
        Assert.AreEqual("TTÜ", o.CompanyName);
    }
    [TestMethod]
    public void YearTest()
    {
        Assert.AreEqual(DateTime.Now.Year.ToString(), o.Year);
    }
}
```

5. Lisa Tests projekti uus klass nimega HomeControllerTests ning sinna sisse lisa testmeetodid.

```
[TestClass]
public class HomeControllerTests
{
    [TestMethod]
    public void Index()
    {
        HomeController controller = new HomeController();
        ViewResult result = controller.Index() as ViewResult;
        Assert.IsNotNull(result);
    }

    [TestMethod]
    public void About()
    {
        HomeController controller = new HomeController();
        ViewResult result = controller.About() as ViewResult;
        Assert.AreEqual("Your application description page.",
            result.ViewData["Message"]);
    }

    [TestMethod]
    public void Contact()
    {
        HomeController controller = new HomeController();
        ViewResult result = controller.Contact() as ViewResult;
        Assert.IsNotNull(result);
    }
}
```

6. Vali Test Exploreris Run All ning testi loodud teste.