



INDONESIA
.NET DEVELOPER COMMUNITY



Seri Belajar
ASP.NET
ASP.NET MVC Untuk Pemula
M Reza Faisal, Microsoft MVP ASP.NET/IIS

Kata Pengantar

Puji dan syukur diucapkan kepada Allah SWT atas selesainya ebook sederhana yang berjudul ASP.NET MVC Untuk Pemula.

Pada ebook ini akan dipaparkan apa saja hal-hal yang mesti diketahui web developer dalam memulai membangun aplikasi web dengan ASP.NET MVC dengan bantuan tool Visual Studio 2015. Pada ebook ini akan diberikan contoh-contoh yang dapat diikuti oleh pembaca sehingga konsep yang diberikan pada buku ini dapat langsung dicoba. Harapannya hal ini akan membantu web developer untuk cepat memahami paparan pada ebook.

Akhir kata, selamat membaca dan semoga ebook ini bermanfaat bagi para web developer pemula untuk membuat aplikasi web. Kritik dan saran akan sangat berarti dan dapat ditujukan via email.

Banjarmasin, Januari 2016

M Reza Faisal

(reza.faisal@gmail.com)

Daftar Isi

<i>Kata Pengantar</i>	I
<i>Daftar Isi</i>	II
<i>Daftar Gambar</i>	V
1 Pendahuluan	1
One ASP.NET	1
ASP.NET 5	7
Visual Studio 2015	8
Referensi	9
2 Framework Web Forms & MVC	11
ASP.NET Web Forms	11
Page Controller	11
Implementasi	12
ASP.NET MVC	14
Konfigurasi	15
Controller	16
View	18
Model	19
Kesimpulan	22
Referensi	24
3 Persiapan	25
Project	25
Database	27
Theme	31
Referensi	34
4 Model	35
Definisi	35
Jenis Model	36
Data Access Layer	36
View Model	40
PegawaiViewModel	40

LoginViewModel	42
Kesimpulan	43
Referensi	43
5 View-Controller	44
Definisi.....	44
View	44
Controller	44
Cara Kerja	44
Razor	53
Sintaks Dasar	53
Layout dan Antarmuka.....	56
HTML Helper	85
ViewData dan ViewBag	96
Implementasi	101
Mengelola Divisi	101
Mengelola Pegawai.....	119
Kesimpulan	138
Referensi	138
6 Keamanan	140
ASP.NET Identity.....	140
Otentikasi	143
Persiapan Database.....	143
Model.....	144
Controller	150
View	171
Uji Coba	179
Pengelolaan Role	182
Controller	182
View	183
Pengelolaan User Aplikasi	187
Model.....	187
Controller	188
View	194
Otorisasi	202

Otorisasi Menu	203
Otorisasi Controller	204
Referensi	206
7 Penutup.....	207

Daftar Gambar

Gambar 1. Keluarga ASP.NET.....	1
Gambar 2. Blank Solution.....	2
Gambar 3. Solution ASPNET.MVC.UntukPemula pada Solution Explorer.	2
Gambar 4. Membuat folder pada solution.	3
Gambar 5. Membuat project ASP.NET Web Application.	3
Gambar 6. Window pemilihan template ASP.NET Project.	4
Gambar 7. Project Web Forms.	4
Gambar 8. Project MVC.	5
Gambar 9. Project Web API.....	5
Gambar 10. Framework Web Forms, MVC dan Web API dalam sebuah project.....	6
Gambar 11. Struktur folder.	6
Gambar 12. Referensi.	7
Gambar 13. Template project ASP.NET 5 pada Visual Studio 2015.	8
Gambar 14. Cross-platform development.	9
Gambar 15. Struktur Page Controller.	12
Gambar 16. Membuat halaman web form.	12
Gambar 17. Halaman sederhana ASP.NET Web Forms.	14
Gambar 18. Hasil skenario pada halaman sederhana ASP.NET Web Forms.	14
Gambar 19. Cara kerja MVC.	15
Gambar 20. Membuat MVC 5 Controller – Home.	16
Gambar 21. Memberi nama class controller – Home.	17
Gambar 22. Pesan error.....	17
Gambar 23. Membuat view.	18
Gambar 24. Halaman view.....	19
Gambar 25. Contoh tampilan halaman web yang akan dibuat.....	19
Gambar 26. Class model Visitor.	20
Gambar 27. Halaman sederhana ASP.NET MVC.	21
Gambar 28. Hasil halaman sederhana ASP.NET MVC.	22
Gambar 29. Isi Toolbox saat membuat file *.cshtml.....	23
Gambar 30. Membuat project PengelolaanPegawai.Web.	25
Gambar 31. Memilih template project web yang akan digunakan.	26
Gambar 32. ProjectPengelolaan.Web.	26

Gambar 33. Membuat file database.....	27
Gambar 34. File PegawaiDB.mdf.....	28
Gambar 35. Server Explorer – Data Connection.....	28
Gambar 36. Membuat tabel Divisi.....	29
Gambar 37. Membuat tabel Pegawai.....	29
Gambar 38. Tabel Divisi dan Pegawai pada PegawaiDB.mdf.....	30
Gambar 39. Menu Show Table Data	31
Gambar 40. Mengisi data pada tabel Divisi.	31
Gambar 41. Mengisi data pada tabel Pegawai.....	31
Gambar 42. AdminLTE Control Panel Template.	32
Gambar 43. SB Admin 2.....	32
Gambar 44. Charisma Responsive Admin Template.....	33
Gambar 45. Metro Dashboard – A Modern and Clean Admin Template.....	33
Gambar 46. Antarmuka theme Metro Dashboard pada layar yang lebih kecil.	34
Gambar 47. Antarmuka theme Metro Dashboard pada layar smartphone.....	34
Gambar 48. Komponen model pada pattern MVC.....	35
Gambar 49. Membuat PegawaiModel.	36
Gambar 50. Memilih cara untuk membuat model.....	37
Gambar 51. Memilih koneksi data.	37
Gambar 52. Memilih tabel yang digunakan.	38
Gambar 53. PegawaiModel.edmx.	38
Gambar 54. Class model Pegawai.	41
Gambar 55. Isi data Pegawai.....	41
Gambar 56. Isi data Divisi.	41
Gambar 57. Membuat class PegawaiViewModel.....	42
Gambar 58. Form login.	43
Gambar 59. Membuat class controller.	46
Gambar 60. Class controller dengan nama Home.	46
Gambar 61. Folder Home pada folder View.....	49
Gambar 62. View dengan nama file Index.cshtml.	49
Gambar 63. Folder View\Home.....	50
Gambar 64. Tampilan aksi Index.....	51
Gambar 65. Tampilan aksi Login.	52
Gambar 66. Tampilan aksi PageNotFound.	52
Gambar 67. Cara kerja komponen view dan controller.	52

Gambar 68. Halaman Dashboard.....	57
Gambar 69. Halaman Tasks.....	57
Gambar 70. Halaman Forms.....	57
Gambar 71. Halaman Tables.....	58
Gambar 72. Struktur folder dan file theme Bootstrap Metro Dashboard.....	59
Gambar 73. Folder css, font, img dan js pada project PengelolaanPegawai.Web.....	59
Gambar 74. Membuat file layout MasterLayout.cshtml.....	60
Gambar 75. Antarmuka MasterLayout.....	60
Gambar 76.....	70
Gambar 77. Halaman ASP.NET dengan Theme Bootstrap Metro Dashboard.....	81
Gambar 78. File-file view untuk controller Metro.....	82
Gambar 79. File _ViewStart.cshtml.....	84
Gambar 80. Window Add View.....	84
Gambar 81. Hasil render halaman view Forms didalam view Latihan.....	88
Gambar 82. Menampilkan konten view Forms pada halaman view Latihan.....	88
Gambar 83. Contoh form – textbox.....	90
Gambar 84. Contoh form – textbox dengan perbaikan antarmuka.....	91
Gambar 85. Contoh form – textarea.....	92
Gambar 86. Contoh form – textarea untuk editor WYSIWYG.....	92
Gambar 87. Contoh form – dropdownlist.....	93
Gambar 88. Contoh form – dropdownlist dengan theme Bootstrap Metro Dashboard.....	93
Gambar 89. Contoh form – radiobutton.....	94
Gambar 90. Contoh form – listbox.....	95
Gambar 91. Contoh form – listbox dengan theme Bootstrap Metro Dashboard.....	95
Gambar 92. Contoh form – tombol.....	96
Gambar 93. Menambah MVC 5 Controller with view, using Entity Framework.....	101
Gambar 94. Add Controller – Divisi.....	102
Gambar 95. Daftar divisi.....	105
Gambar 96. Daftar divisi dengan theme Bootstrap Metro Dashboard.....	107
Gambar 97. Detail data divisi.....	108
Gambar 98. Detail divisi dengan theme Bootstrap Metro Dashboard.....	109
Gambar 99. Form input data divisi.....	110
Gambar 100. Form create dengan theme Bootstrap Metro Dashboard.....	113
Gambar 101. Form edit.....	113
Gambar 102. Form edit create dengan theme Bootstrap Metro Dashboard.....	116

Gambar 103. Halaman konfirmasi sebelum record yang dipilih dihapus.	116
Gambar 104. Halaman konfirmasi dengan menggunakan theme Bootstrap Metro Dashboard.	119
Gambar 105. Window Add Controller – Pegawai.	119
Gambar 106. Daftar pegawai.	122
Gambar 107. Antarmuka daftar pegawai dengan menggunakan theme Bootstrap Metro Dashboard.	123
Gambar 108. Antarmuka detail pegawai.	125
Gambar 109. Antarmuka detail pegawai dengan theme Bootstrap Metro Dashboard.	126
Gambar 110. Antarmuka form input data pegawai.	127
Gambar 111. Antarmuka form input dengan theme Bootstrap Metro Dashboard.	129
Gambar 112. Form input pegawai – dropdownlist pilihan divisi.	131
Gambar 113. Form input pegawai – pilihan tanggal.	131
Gambar 114. Form edit pegawai.	132
Gambar 115. Form edit pegawai dengan theme Bootstrap Metro Dashboard.	133
Gambar 116. Antarmuka konfirmasi sebelum data dihapus.	135
Gambar 117. Antarmuka konfirmasi penghapusan yang telah memanfaatkan theme.	136
Gambar 118. Menambahkan ASP.NET Identity pada project via NuGet.	140
Gambar 119. Tiga komponen utama ASP.NET Identity.	141
Gambar 120. Window preview.	141
Gambar 121. Window License Acceptance.	142
Gambar 122. Status installasi komponen ASP.NET Identity.	142
Gambar 123. Komponen Microsoft.Owin.Host.SystemWeb.	143
Gambar 124. Tabel-tabel hasil generasi otomatis dari Entity Framework Code First.	144
Gambar 125. Form registrasi pengguna/user.	172
Gambar 126. Form login.	178
Gambar 127. Kondisi awal pada database.	179
Gambar 128. Form register pengguna/user yang telah diisi.	179
Gambar 129. Tabel tambahan untuk pengelolaan data pengguna/user.	180
Gambar 130. Informasi user yang aktif dan tombol logout.	181
Gambar 131. Daftar role.	184
Gambar 132. Form menambah role.	185
Gambar 133. Form untuk mengedit role.	186
Gambar 134. Daftar user.	195
Gambar 135. Detail user yang dipilih.	197
Gambar 136. Form input data user.	199

Gambar 137. Form update user yang dipilih.....	201
Gambar 138. Konfirmasi untuk menghapus user.	202
Gambar 139. Menu aplikasi Pengelolaan Pegawai.	203

Pendahuluan

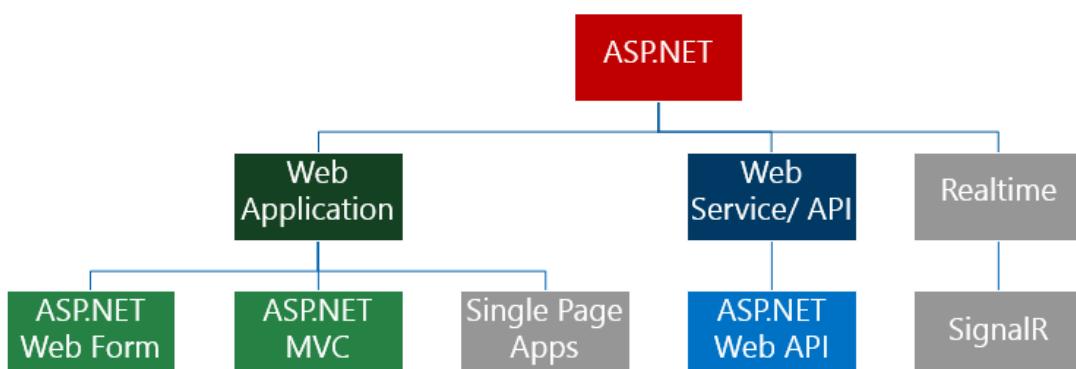
One ASP.NET

Sejak Microsoft .NET dan Visual Studio pertama kali diperkenalkan tahun 2002, maka dimulai pulalah kelahiran ASP.NET Web Forms sebagai framework yang memudahkan web developer untuk membangun aplikasi web. ASP.NET dibuat untuk menggantikan teknologi web Microsoft terdahulu yaitu Active Server Pages (ASP).

ASP.NET Web Forms dibuat agar programmer mempunyai pengalaman yang sama saat membuat program berbasis desktop dengan Win Forms. Tetapi seiring dengan perkembangan teknologi dan tuntutan pasar, Microsoft menambahkan anggota baru para keluarga ASP.NET, yaitu ASP.NET MVC.

Tidak hanya berhenti sampai disitu, ASP.NET semakin berkembang dan semakin lengkap untuk memenuhi perkembangan teknologi web yang saat ini. Berikut adalah ASP.NET sekarang ini, yaitu :

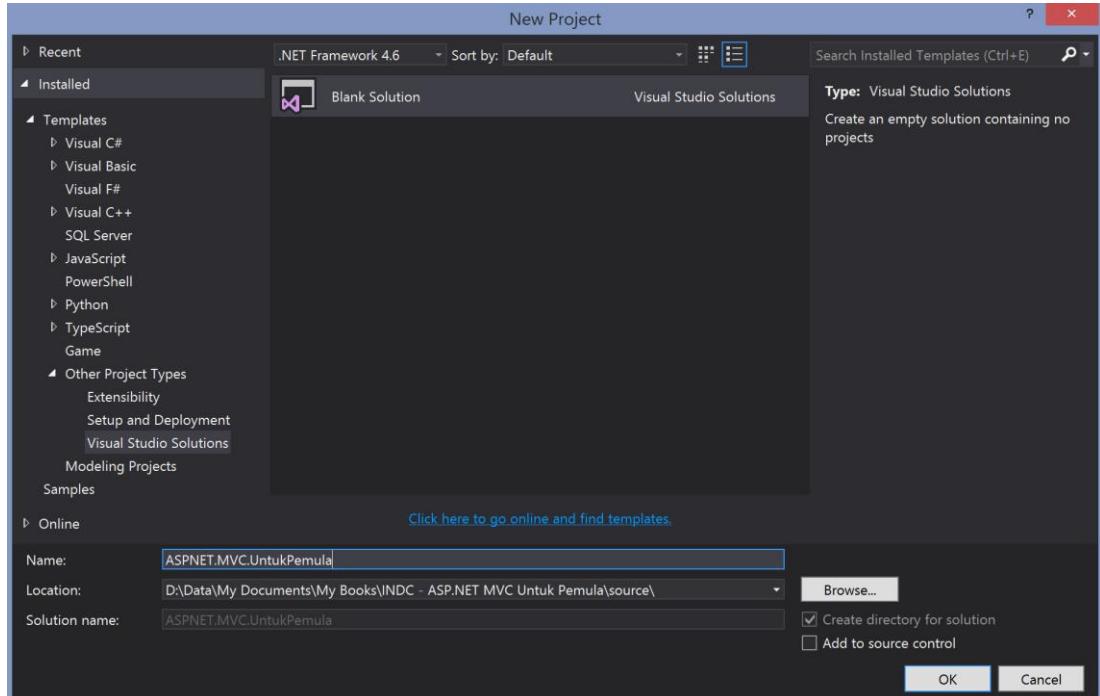
1. Web Application, pada bagian ini terdiri lagi atas :
 - a. Web Forms.
 - b. MVC.
 - c. Web Pages/Razor.
 - d. Mobile.
 - e. Single Page Application.
2. Web Service/API's.
 - a. Web API.
3. Real Time.
 - a. SignalR.



Gambar 1. Keluarga ASP.NET.

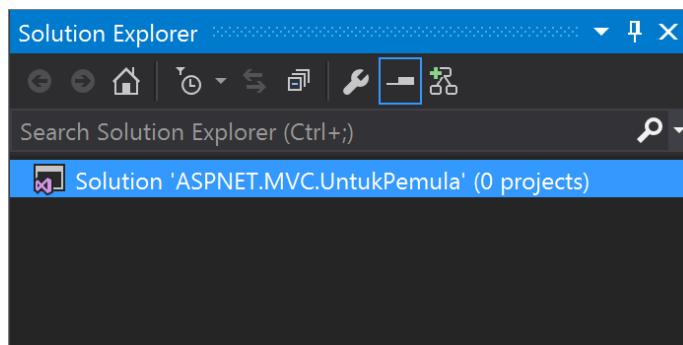
One ASP.NET adalah istilah yang memberikan cara untuk mempermudah dan memungkinkan developer untuk menggunakan sebuah project yang mana didalamnya terdapat multi framework. Artinya dalam sebuah project dimungkinkan didalamnya terdapat ASP.NET Web Forms, ASP.NET Web MVC dan ASP.NET Web API.

Berikut ini akan diperlihatkan implementasi konsep One ASP.NET dengan cara membuat project ASP.NET Web Application. Langkah pertama untuk membuat project ini dapat dimulai dengan membuat Solution kosong dengan cara memilih File > New > Project kemudian pada window New Project pilih Templates > Other Project Types > Visual Studio Solutions kemudian pilih Blank Solution.



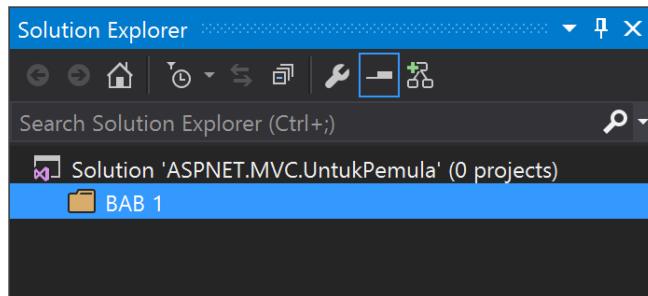
Gambar 2. Blank Solution.

Pada kolom Name, tulis nama solution yang diinginkan. Pada contoh ini dibuat Solution dengan nama ASPNET.MVC.UntukPemula. Kemudian klik tombol OK. Hasilnya dapat dilihat pada area Solution Explorer.



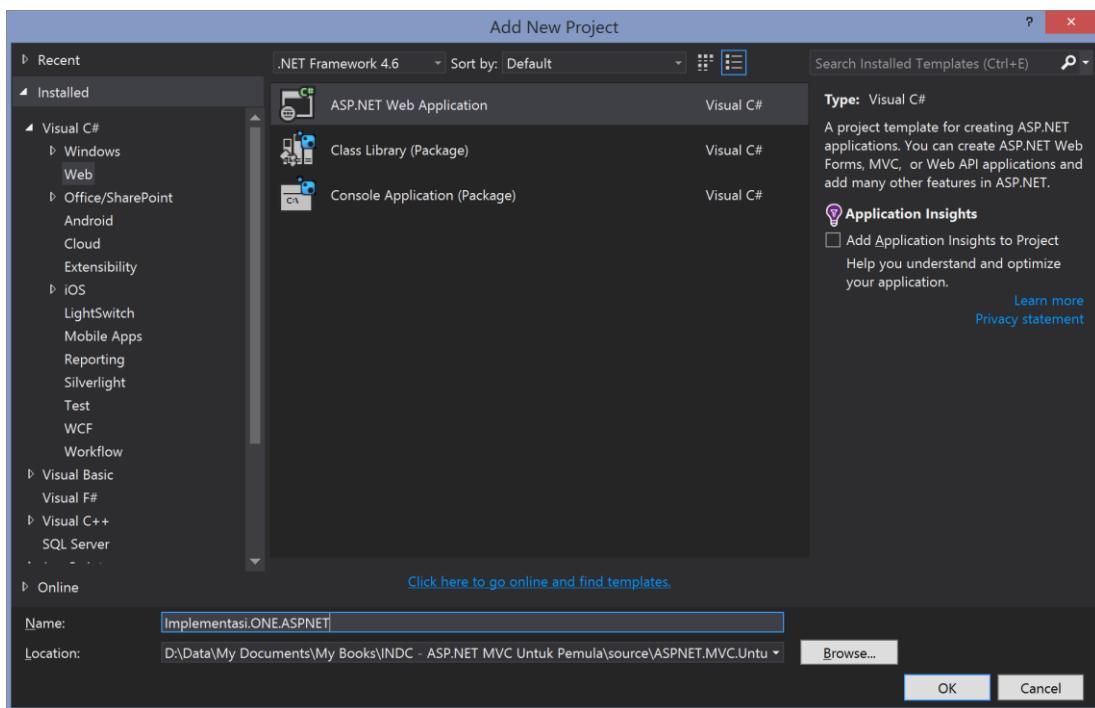
Gambar 3. Solution ASPNET.MVC.UntukPemula pada Solution Explorer.

Pada Solution dapat ditambahkan dapat ditambahkan folder atau project. Folder dan project yang dapat ditambah pada suatu Solution dapat lebih dari satu. Sebagai contoh untuk menambahkan folder dilakukan dengan cara klik kanan pada solution kemudian pilih Add > New Solution Folder. Kemudian berikan nama untuk folder tersebut.



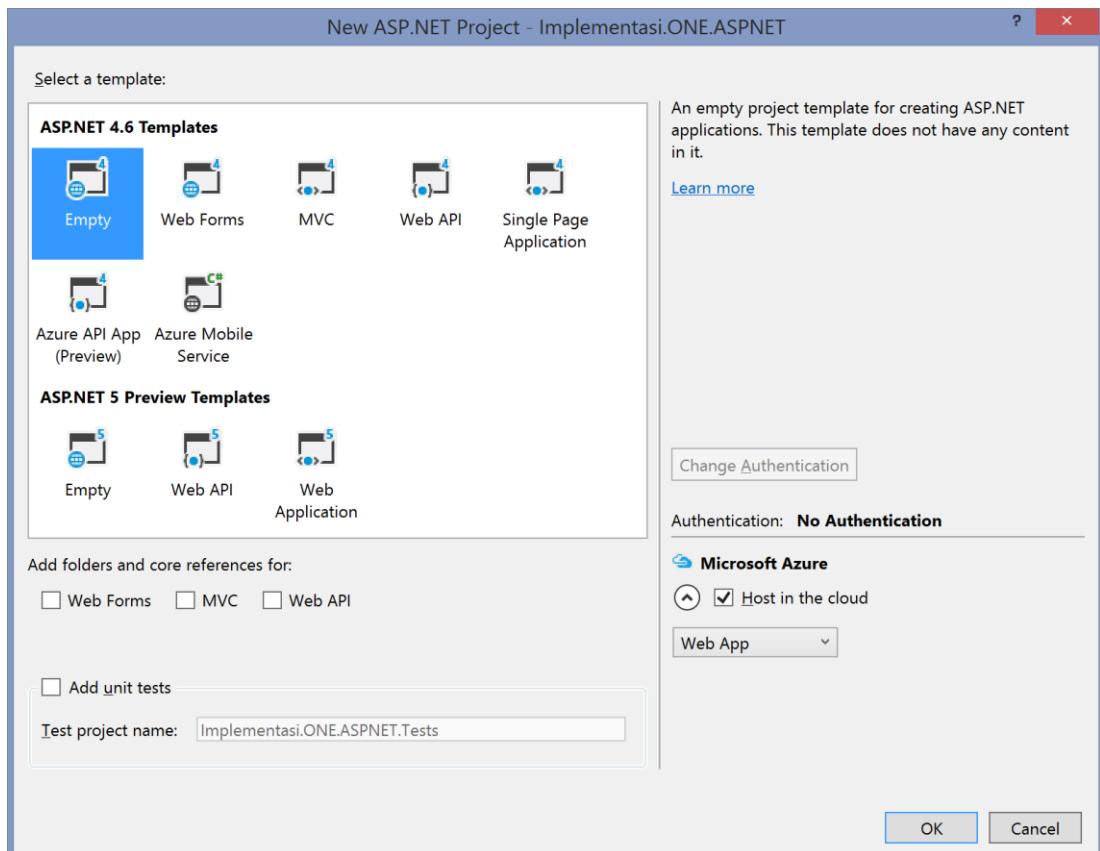
Gambar 4. Membuat folder pada solution.

Selanjutnya akan dicontohnya cara membuat project pada solution. Pada contoh ini diperlihatkan cara membuat project di dalam folder yang ada pada solution. Klik kanan pada folder tersebut kemudian pilih Add > New Project. Pada window Add New Project pilih Visual C# > Web > ASP.NET Web Application. Kemudian berikan nama project pada kolom Name.



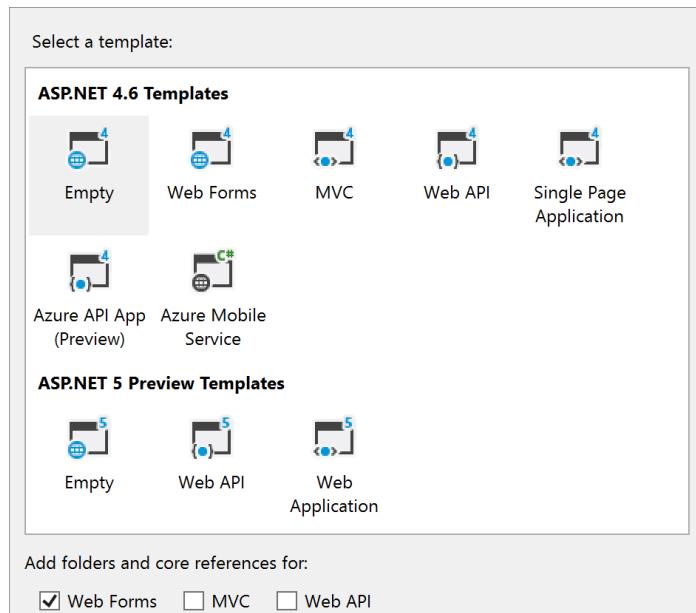
Gambar 5. Membuat project ASP.NET Web Application.

Setelah itu klik tombol OK. Kemudian akan ditampilkan window seperti gambar berikut ini.

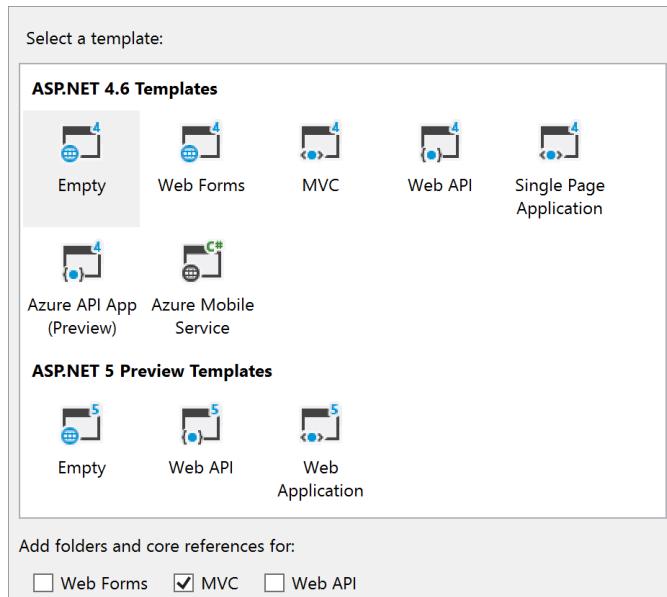


Gambar 6. Window pemilihan template ASP.NET Project.

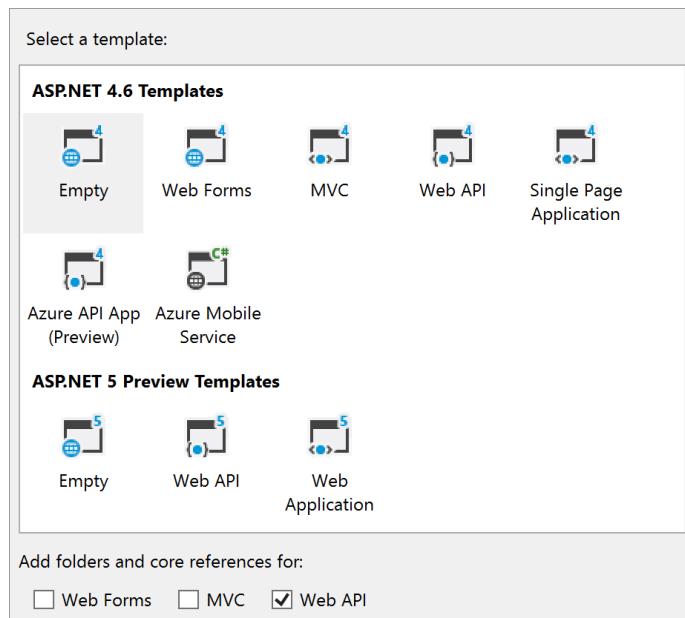
Misalnya dipilih ASP.NET 4.6 Templates > Empty, kemudian pada bagian Add folders and core references for dapat dipilih framework yang diinginkan. Pada gambar di bawah ini diperlihatkan pada project dapat ditambahkan framework Web Forms atau MVC atau Web API.



Gambar 7. Project Web Forms.

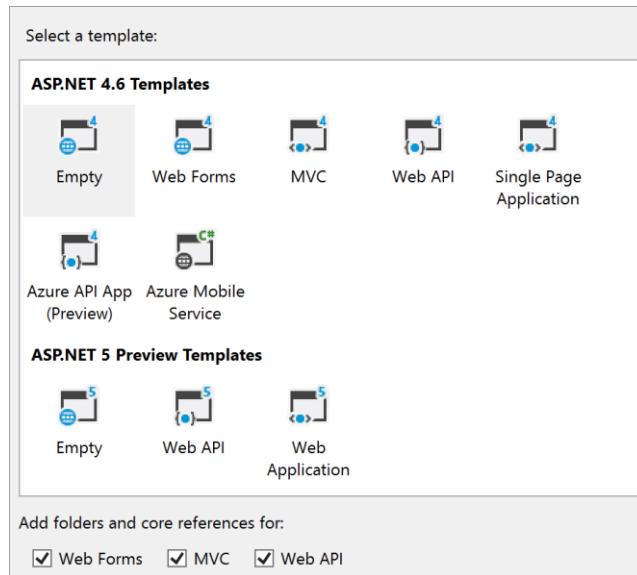


Gambar 8. Project MVC.



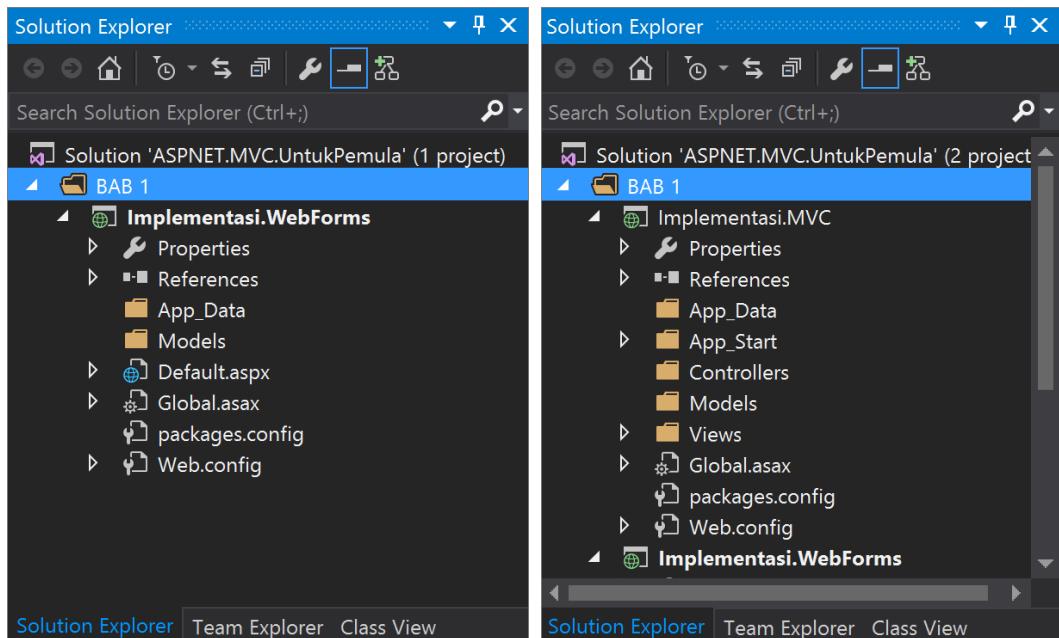
Gambar 9. Project Web API.

Selain itu dalam satu project dapat terdiri dari kombinasi lebih dari satu framework atau gabungan dari seluruh framework seperti pada gambar berikut ini.



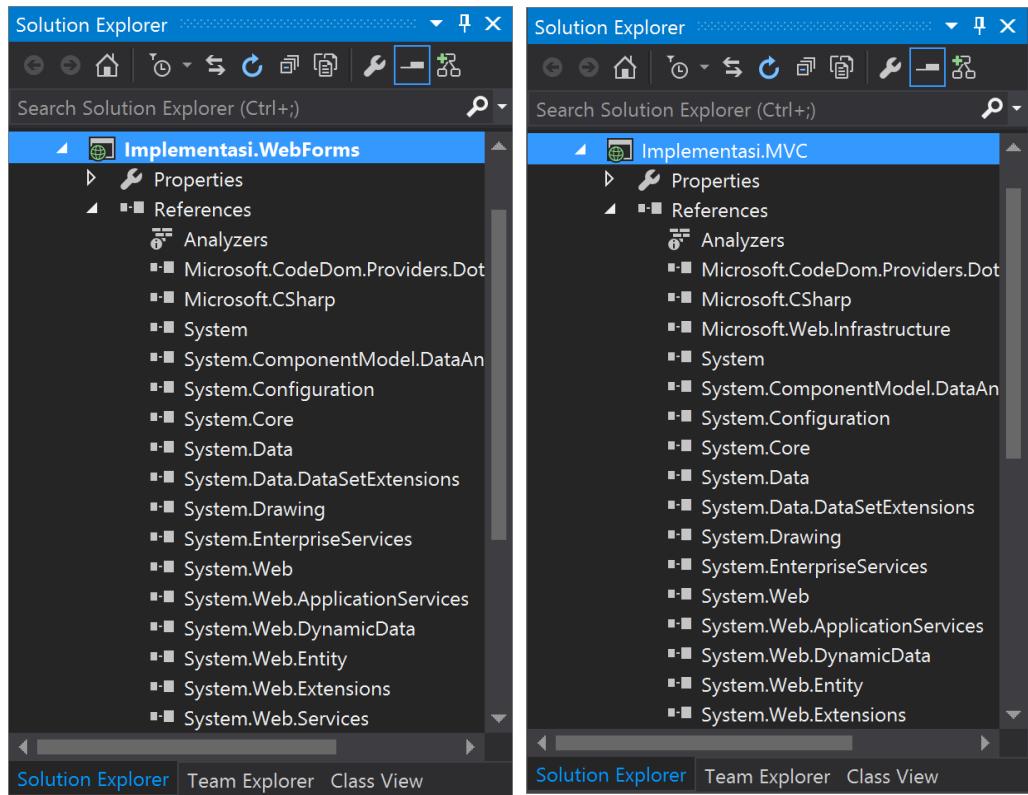
Gambar 10. Framework Web Forms, MVC dan Web API dalam sebuah project.

Setelah framework dipilih dan diklik tombol OK maka secara otomatis akan disiapkan folder dan referensi sesuai dengan framework yang dipilih. Pada gambar di bawah ini dapat dilihat perbedaan folder pada project jika developer memilih framework yang berbeda.



Gambar 11. Struktur folder.

Dan pada gambar di bawah ini diperlihatkan perbedaan referensi pada framework yang berbeda.

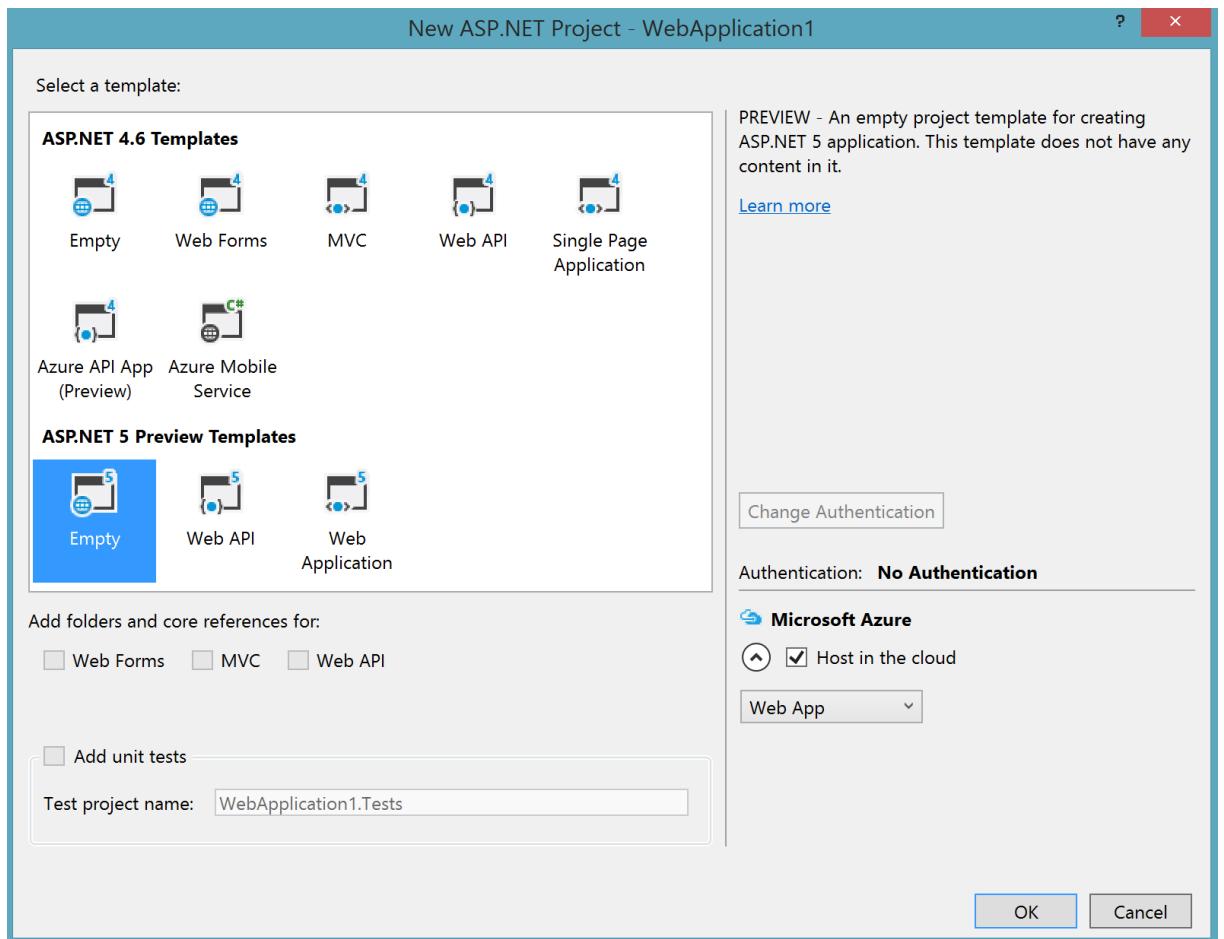


Gambar 12. Referensi.

Jika awalnya hanya dipilih satu framework saja pada project, developer dapat menambahkan file dari framework lain ke dalam project tersebut. Saat file tersebut ditambahkan maka secara otomatis akan ditambahkan folder dan referensi yang sesuai dengan framework dari file tersebut.

ASP.NET 5

Sekarang, saat ebook ini ditulis, ASP.NET telah mencapai versi 5 walaupun belum RTW. ASP.NET 5 merupakan desain ulang dari ASP.NET yang mengalami banyak perubahan. Hal yang paling signifikan adalah ASP.NET 5 menjadi framework yang bersifat open-source dan cross-platform, yang artinya aplikasi web yang dibangun dengan ASP.NET 5 dapat berjalan di berbagai platform. Saat ini platform yang telah didukung adalah Windows, Linux dan Mac.



Gambar 13. Template project ASP.NET 5 pada Visual Studio 2015.

Saat ini template project ASP.NET 5 pada Visual Studio 2015 masih dalam tahap preview semoga pada beberapa waktu kedepan template ASP.NET 5 sudah rampung.

Tetapi pada ebook ini tidak akan membahas bagaimana membangun aplikasi web dengan ASP.NET 5.

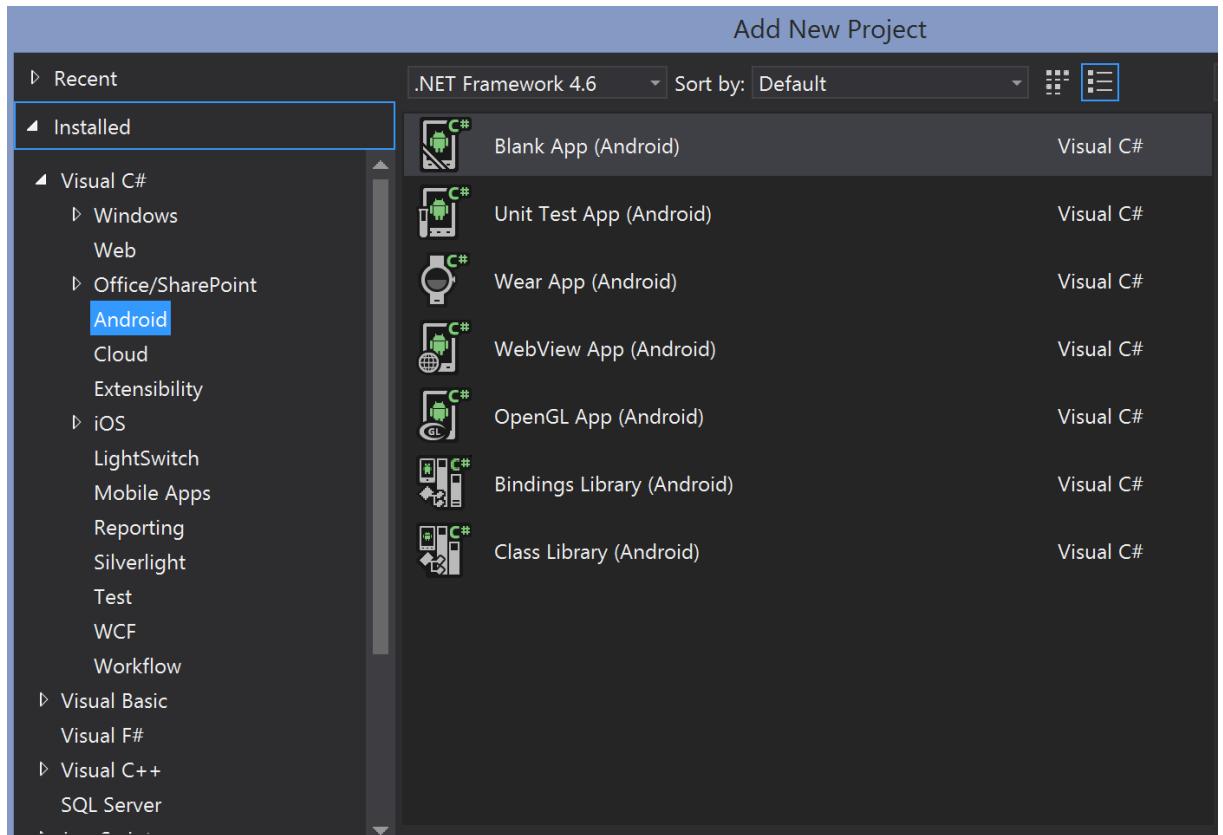
Visual Studio 2015

Visual Studio adalah Integrated Development Environment (IDE) dari untuk membangun aplikasi console dan Graphical user interface (GUI) dengan menggunakan bahasa yang didukung pada .NET Framework. Aplikasi yang dapat dibangun diantaranya adalah aplikasi desktop, web, mobile, cloud, office dan lain-lain.

Visual Studio selain mempunyai feature untuk :

1. Designer antarmuka untuk Winform, WPF dan Web. Selain itu juga dapat digunakan untuk mendesign Class, Data dan Mapping.
2. Code editor dengan dukungan IntelliSense.
3. Debugger.
4. Cross-platform development, saat ini dengan Visual Studio 2015 dimungkinkan untuk membangun aplikasi pada berbagai platform. Untuk aplikasi web, telah dimungkinkan untuk dijalankan tidak hanya pada platform Windows saja tetapi juga

pada Mac dan Linux. Untuk aplikasi mobile telah dimungkinkan untuk membangun aplikasi mobile pada platform Windows Phone, Android dan iOS.



Gambar 14. Cross-platform development.

Seperti Visual Studio pada versi sebelumnya, Visual Studio 2015 juga merupakan IDE dengan multi target .NET Framework, artinya developer dapat membangun aplikasi dengan menggunakan .NET Framework 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.2 sampai 4.6.

Untuk mendapatkan Visual Studio dapat mengunjungi alamat berikut ini <https://www.visualstudio.com/downloads/download-visual-studio-vs>. Pada alamat tersebut dapat diunduh :

1. Visual Studio 2015 Community yang bersifat gratis.
2. Visual Studio 2015 Enterprise yang berbayar.
3. Visual Studio Code yang bisa digunakan sebagai tool development pada platform Windows, Linux dan Mac.

Referensi

<http://www.asp.net/>

<http://docs.asp.net/en/latest/index.html>

<https://www.visualstudio.com/downloads/download-visual-studio-vs>

2

Framework Web Forms & MVC

ASP.NET Web Forms dan ASP.NET MVC adalah framework untuk memudahkan developer untuk membangun aplikasi web. Karena keduanya berada di atas .NET Framework maka class library-nya juga dapat digunakan pada ASP.NET Web Forms dan ASP.NET MVC. Selain itu framework memungkinkan developer untuk memanfaatkan komponen atau helper yang telah disediakan dengan cara menggunakan secara langsung atau melakukan modifikasi agar sesuai dengan kebutuhan dari aplikasi web yang akan dibangun. Di dalam kedua framework tersebut juga memiliki aturan yang harus diikuti agar tidak terjadi kesalahan (error) dan menghasilkan aplikasi web yang benar.

Pada bagian ini akan diperlihatkan perbedaan yang mendasar antara ASP.NET Web Forms dan ASP.NET MVC yang akan ditunjukkan secara langsung dalam bentuk contoh.

ASP.NET Web Forms

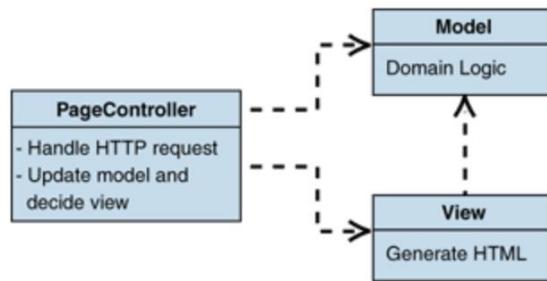
ASP.NET Web Forms menggunakan pattern Page Controller, pattern ini termasuk kedalam software design pattern. Pattern ini memungkinkan request akan ditangani oleh sebuah page yang filenya ada secara fisik. File fisik tersebut sesuai dengan URL yang tertulis pada address bar di web browser. Selain itu karakteristik yang dimiliki oleh ASP.NET Web Forms adalah sebagai berikut :

1. Form server-side secara default akan melakukan post-back terhadap page itu sendiri.
2. Stateful.
3. Componen model.
4. Pengujian dapat dilakukan dengan unit testing atau web testing. Test Driven Development (TTD) sulit diimplementasikan.

Page Controller

Uraian lebih dalam mengenai Page Controller dijelaskan sebagai berikut ini. Page Controller dapat didefinisikan sebagai objek yang menangani request pada sebuah page atau sebuah action. Secara singkat, cara kerja pattern ini melakukan mekanisme penanganan event pada client kemudian mengirimkannya ke server dan akhirnya akan dipanggil method secara otomatis untuk menampilkan view yang sesuai permintaan. Pattern ini memisahkan antara logic dan kode yang berkaitan dengan view.

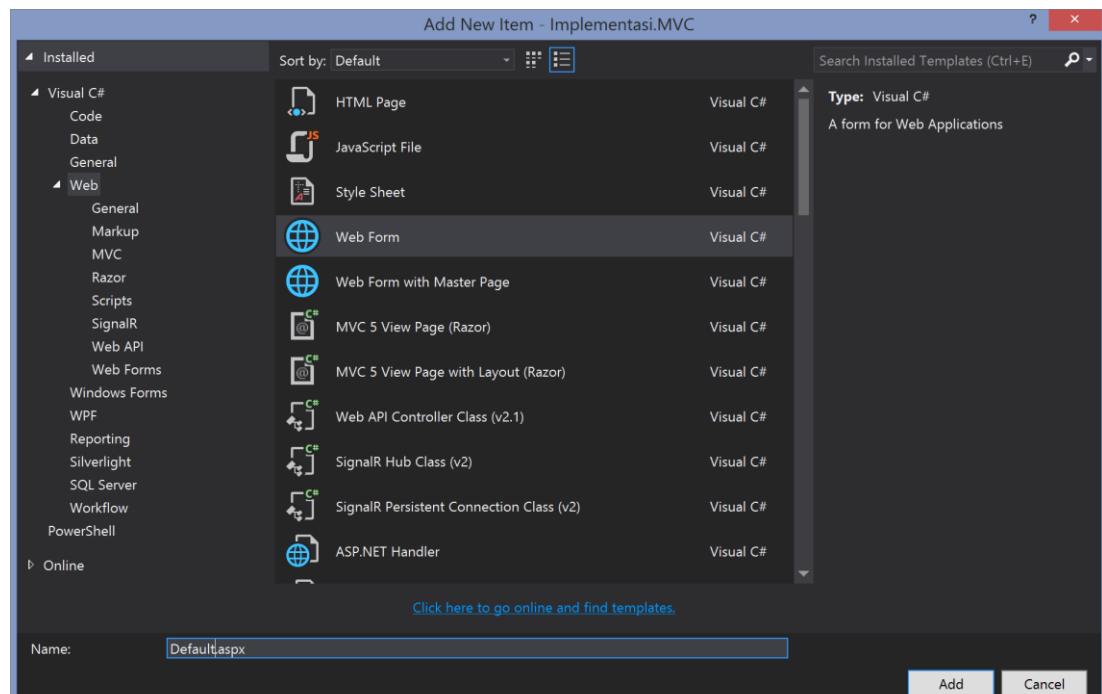
Paparan di atas dapat digambarkan dengan ilustrasi seperti berikut ini.



Gambar 15. Struktur Page Controller.

Implementasi

Pattern ini secara default digunakan pada framework ASP.NET Web Form. Untuk membuat sebuah halaman ASP.NET Web Form dapat dilakukan dengan cara yang mudah, yaitu dengan klik kanan pada project kemudian pilih Add > New Item. Kemudian pada Visual C# > Web pilih Web Form. Berikan nama file pada kolom Name dan klik tombol Add.



Gambar 16. Membuat halaman web form.

Pada ASP.NET Web Forms setiap page *.aspx adalah Page Controller dan mempunyai prilaku untuk melakukan postback ke page itu sendiri. Selain file berekstension *.aspx juga terdapat file *.aspx.cs atau *.aspx.vb yang berfungsi sebagai code-behind, file tipe ini dianggap sebagai kesatuan dari page. Walau keduanya terlihat sebagai 2 file yang berbeda tetapi keduanya adalah satu kesatuan. Tetapi tidak menutup kemungkinan dibuat halaman ASP.NET Web Forms yang terdiri atas file *.aspx saja tanpa file code-behind.

Secara nyata dapat dilihat pada contoh kode di bawah ini. Berikut ini adalah kode file *.aspx.

```

Default.aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Implementasi.WebForms.Default" %>

<!DOCTYPE html>

```

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Nama :
            <asp:TextBox ID="TextBox_Nama" runat="server"></asp:TextBox>
            <br /><br />
            Alamat :
            <asp:TextBox ID="TextBox_Alamat" runat="server"></asp:TextBox>
            <br /><br />
            <asp:Label ID="Label_Hello" runat="server"
Text="Label"></asp:Label>
            <br />
            <asp:Button ID="Button_Kirim" runat="server" Text="Kirim"
OnClick="Button_Kirim_Click" />
        </div>
    </form>
</body>
</html>

```

Dan berikut ini adalah kode file *.aspx.cs.

```

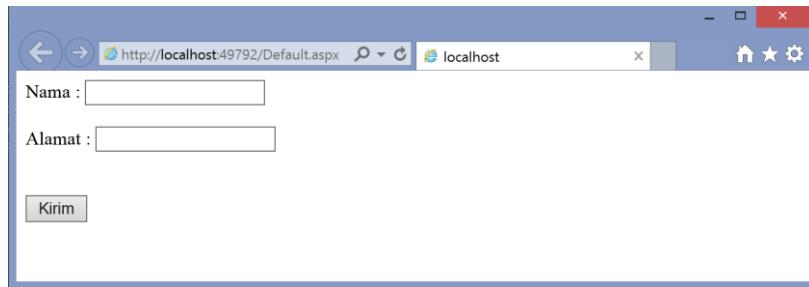
Default.aspx.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Implementasi.WebForms
{
    public partial class Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                Label_Hello.Text = String.Empty;
            }
        }

        protected void Button_Kirim_Click(object sender, EventArgs e)
        {
            Label_Hello.Text = "Hello " + TextBox_Nama.Text + " di " +
TextBox_Alamat.Text;
        }
    }
}

```

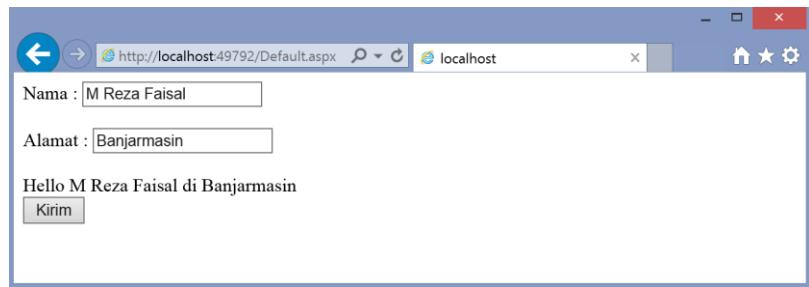
Dan berikut adalah tampilan awal dari halaman ini.



Gambar 17. Halaman sederhana ASP.NET Web Forms.

Dari gambar di atas, dapat dilihat pada address bar web browser yang menampilkan alamat sesuai dengan nama file yang dibuat. Hal ini memperlihatkan file fisik yang dibuat dapat diakses langsung dengan cara di atas.

Selanjutnya pada halaman di atas akan dilakukan skenario sebagai berikut, pertama diinputkan nilai pada textbox, kemudian klik tombol Kirim. Maka hasilnya dapat dilihat pada gambar di bawah ini.



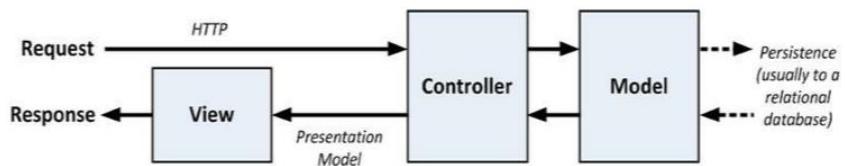
Gambar 18. Hasil skenario pada halaman sederhana ASP.NET Web Forms.

Skenario ini memperlihatkan apa yang telah dipaparkan di atas mengenai cara kerja pattern Page Controller. Saat tombol Kirim diklik maka dilakukan postback terhadap halaman tersebut, dan request ke server sambil mengirimkan parameter. Pada halaman di atas, parameter yang dikirimkan adalah yang nilai-nilai yang dimiliki oleh server control seperti textbox atau label. Kemudian akan dipanggil method yang merupakan event handler dari tombol Klik yaitu method Button_Kirim_Click yang terdapat pada file Default.aspx.cs. Selanjutnya dilakukan menampilkan view sesuai dengan permintaan, dalam skenario ini perubahan view dapat dilihat dengan menampilkan tulisan "Hello M Reza Faisal" pada halaman tersebut.

ASP.NET MVC

Pattern MVC (Model-View-Controller) adalah termasuk sebagai Architectural Pattern. Sedangkan software design pattern yang digunakan pada ASP.NET MVC adalah Front Controller, artinya kontrol akan bersifat terpusat (center controller) pada sebuah class saja. Berbeda dengan ASP.NET Web Forms yang memiliki controller pada setiap halamannya.

Cara kerja MVC dapat dilihat pada gambar berikut ini.



Gambar 19. Cara kerja MVC.

Berikut ini akan dicontohnya membuat halaman ASP.NET MVC, dengan contoh tersebut akan dapat dilihat bagaimana cara kerja pattern ini pada framework ASP.NET MVC. Contoh akan diberikan dalam bentuk langkah-langkah pembuatan halaman ASP.NET MVC sampai bisa ditampilkan pada web browser. Akan dipaparkan beberapa hal tentang dasar-dasar ASP.NET MVC yang bermanfaat untuk diketahui di awal perkenalan dengan framework ini.

Konfigurasi

Pada ASP.NET MVC, alamat yang ditulis pada address bar web browser tidak menunjukkan sebagai file fisik yang benar ada, tetapi karena konfigurasi dan proses routing yang dapat dilihat pada file Global.asax.cs dan RouteConfig.cs (pada folder App_Start).

```
Global.asax.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Implementasi.MVC
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

File Global.asax (Global.asax.cs) juga sering ditemui pada project ASP.NET Web Forms, file ini berfungsi sebagai file application yang bertanggung jawab pada event-event di level application yang dibangkitkan oleh ASP.NET atau HttpModules. Pada file ini dapat dilihat terdapat penanganan event saat aplikasi dijalankan yaitu pada method Application_Start. Pada method tersebut dapat dilihat pemanggilan method milik class RouteConfig. Berikut ini adalah isi dari class RouteConfig yang berada pada file RouteConfig.cs pada folder App_Start.

```
RouteConfig.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Implementasi.MVC
{
```

```

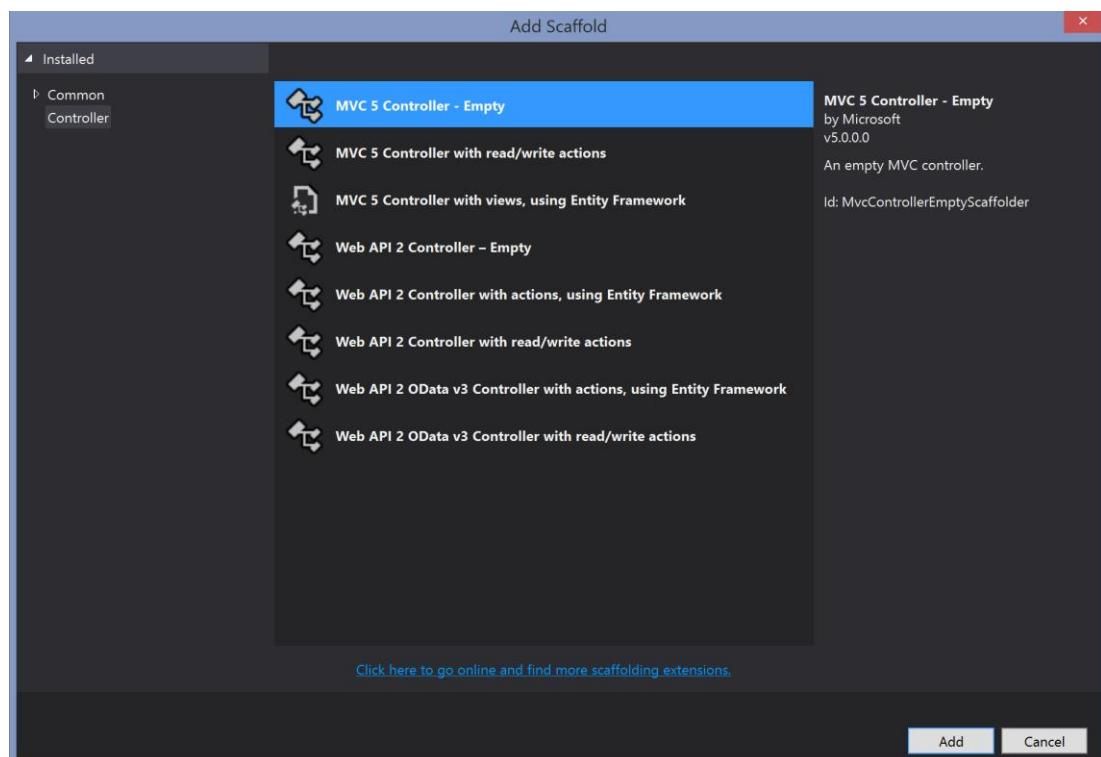
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
        );
    }
}

```

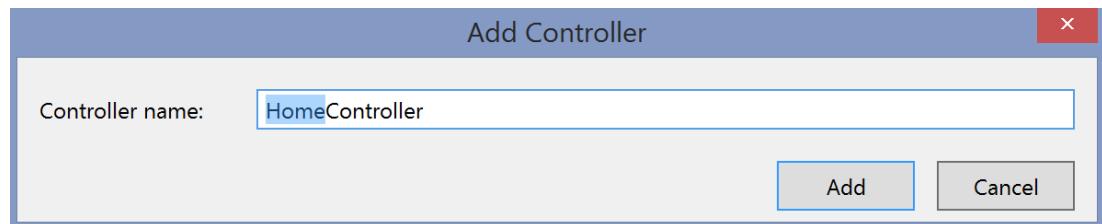
Controller

Dari kode di atas dapat dilihat bahwa nama controller default yang digunakan pada aplikasi ini adalah Home, artinya jika mengikuti kode tersebut perlu dibuat class controller dengan nama Home. Sesuai dengan aturan framework, class ini disimpan pada folder Controllers. Untuk menambahkan class ini klik kanan pada folder Controller kemudian pilih Add > Controller, maka akan ditampilkan window seperti berikut ini.



Gambar 20. Membuat MVC 5 Controller – Home.

Pilih MVC 5 Controller – Empty kemudian klik tombol Add, maka akan ditampilkan window Add Controller yang berfungsi untuk memberi nama class controller yang akan dibuat. Berikan nilai HomeController pada kolom isian Controller name, kemudian klik tombol Add. Harus diperhatikan dalam pemberian nama class yang harus mengikuti aturan yang telah ditetapkan oleh framework ini.



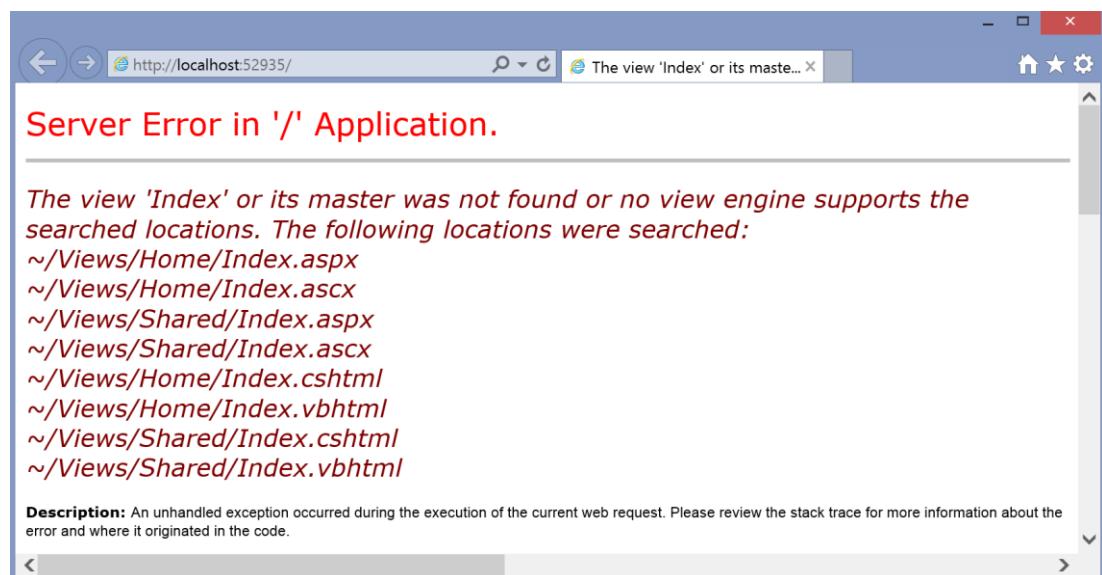
Gambar 21. Memberi nama class controller – Home.

Dan berikut ini adalah isi dari class controller HomeController yang telah dibuat tersebut. Pada class controller ini telah dibuatkan action Index yang berisi perintah untuk menambilkan View.

```
HomeController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Implementasi.MVC.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Jika pada framework ASP.NET Web Forms, cukup membuat sebuah halaman web saja kemudian sudah dapat dieksekusi dan dilihat pada web browser, maka pada framework ASP.NET MVC hal itu belum bisa dilakukan. Sebagai bukti jika jika project ini dieksekusi maka akan dilihat tampilan pada web browser seperti berikut ini.



Gambar 22. Pesan error.

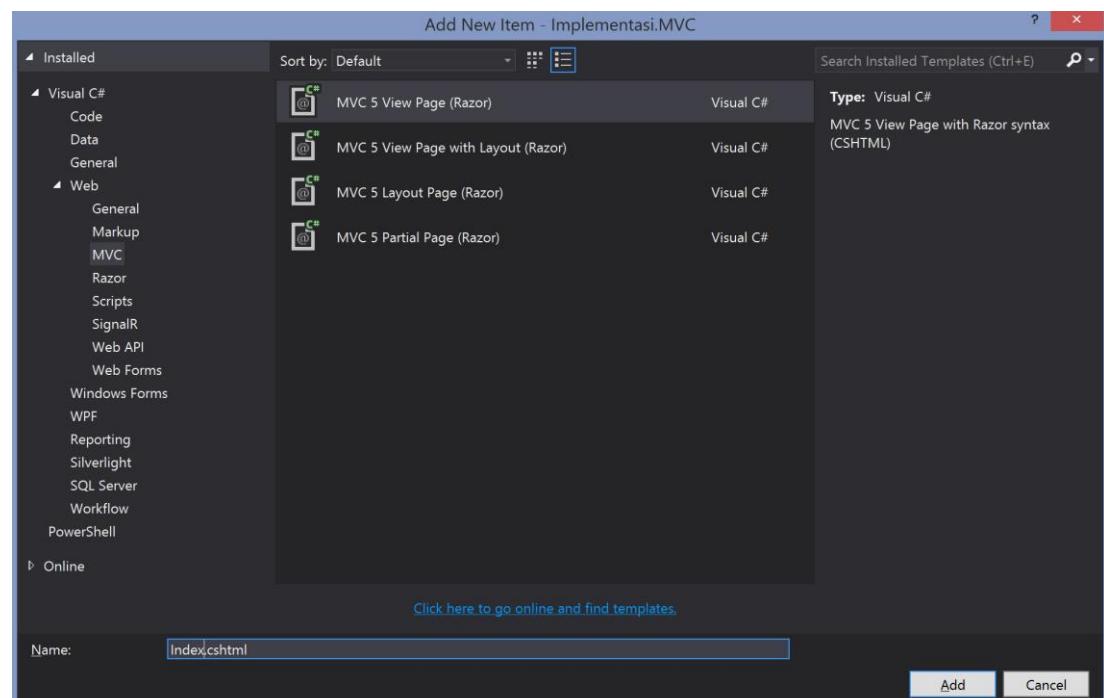
View

Dari output diatas dapat dilihat jika controller yang dibuat telah dikenali dan telah menjalankan fungsinya yaitu memanggil method Index seperti yang terlihat dari pesan error di atas. Jika dilihat isi dari method Index pada class HomeController, dapat diketahui fungsi dari method ini adalah mengembalikan output dari method View yang berfungsi untuk menampilkan konten dari file yang berfungsi sebagai view.

Pesan kesalahan di atas memberikan informasi bahwa tidak ditemukan file yang view yang dimaksud. File view yang dimaksud adalah salah satu dari file berikut ini :

- ~/Views/Home/Index.aspx.
- ~/Views/Home/Index.ascx.
- ~/Views/Shared/Index.aspx.
- ~/Views/Shared/Index.ascx.
- ~/Views/Home/Index.cshtml.
- ~/Views/Home/Index.vbhtml.
- ~/Views/Shared/Index.cshtml.
- ~/Views/Shared/Index.vbhtml.

Pada contoh ini, akan dibuat view khusus untuk class HomeController pada folder Home dengan nama file Index.cshtml. File *.cshtml adalah file yang ditulis gabungan antara kode HTML dan sintaks Razor yang akan dibahas lebih lanjut pada bab selanjutnya. Untuk menambahkan file view ini, klik kanan pada folder Home yang telah dibuatkan secara otomatis kemudian pilih Add > New item. Maka akan ditampilkan window seperti berikut ini, pilih MVC 5 View Page (Razor) dan berikan nama file yang diinginkan pada kolom input Name, kemudian klik tombol Add.



Gambar 23. Membuat view.

Kemudian modifikasi file Index.cshtml sehingga berisi baris kode seperti berikut ini.

```
Index.cshtml
@{
    Layout = null;
}
```

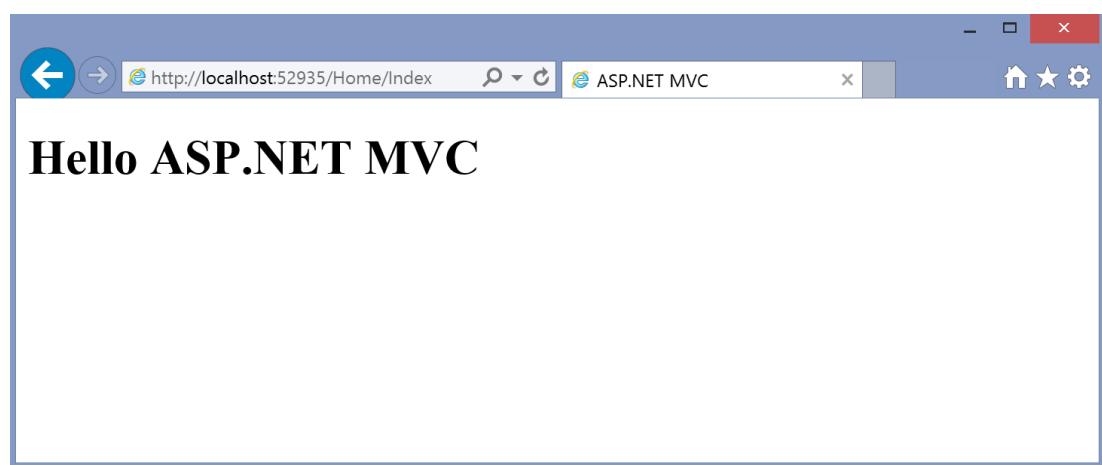
```

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ASP.NET MVC</title>
</head>
<body>
    <div>
        <h1>Hello ASP.NET MVC</h1>
    </div>
</body>
</html>

```

Selanjutnya kembali dilakukan eksekusi project seperti yang dilakukan di atas maka akan dapat dilihat tampilan pada web browser seperti berikut ini.

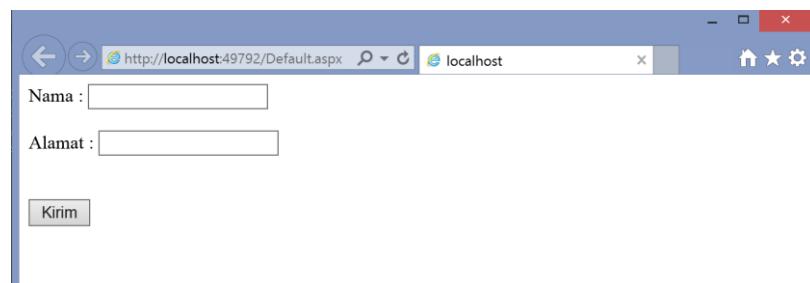


Gambar 24. Halaman view.

Dari contoh ini maka class controller telah berfungsi sempurna dengan memanggil method Index yang memanggil dan menampilkan halaman Index.cshtml.

Model

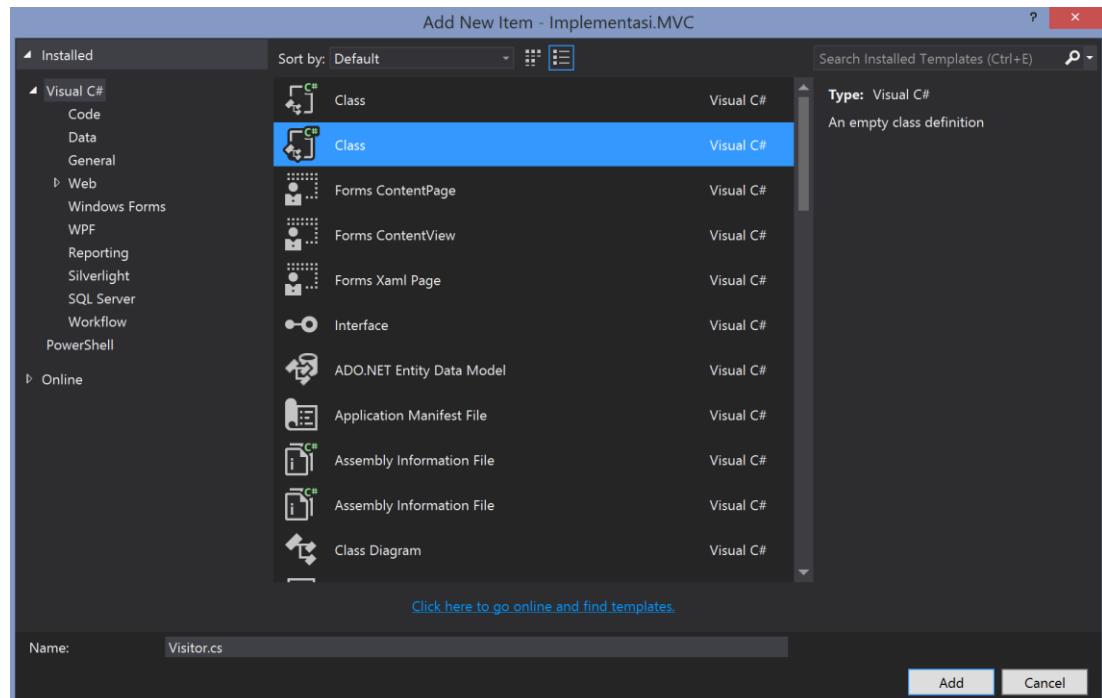
Pada bagian ini akan diperkenalkan tentang model. Untuk itu akan diberikan contoh dengan membuat halaman aplikasi web sederhana yang mempunyai fungsi seperti halaman aplikasi web yang telah dibuat dengan framework ASP.NET Web Forms di atas. Akan dibuat halaman seperti pada gambar di bawah ini dengan ASP.NET MVC.



Gambar 25. Contoh tampilan halaman web yang akan dibuat.

Untuk membuat halaman seperti gambar di atas dengan ASP.NET MVC, maka terlebih dahulu dibuat class model pendukung untuk membuat halaman seperti tersebut. Salah satu fungsi model pada framework ini adalah sebagai sarana untuk pertukaran informasi pada aplikasi.

Class model yang akan disimpan pada folder Models. Caranya adalah klik kanan pada folder Models kemudian Add > Class.



Gambar 26. Class model Visitor.

Pada window Add New Item berikan nama class model pada kolom Name kemudian klik tombol Add.

```
Visitor.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Implementasi.MVC.Models
{
    public class Visitor
    {
        public String Nama { set; get; }
        public String Alamat { set; get; }
    }
}
```

Selanjutnya perlu dilakukan modifikasi pada bagian view dan controller. Pada bagian view dilakukan modifikasi pada file Index.cshtml menjadi seperti berikut ini.

```
Index.cshtml
@{
    Layout = null;
}

@model Implementasi.MVC.Models.Visitor
```

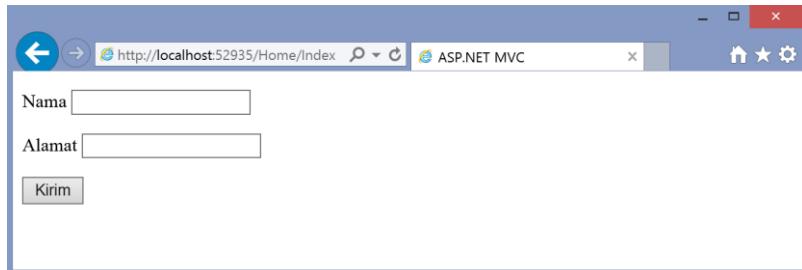
```

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>ASP.NET MVC</title>
</head>
<body>
    <div>
        @using (Html.BeginForm())
        {
            <p>
                @Html.LabelFor(m=>m.Nama)
                @Html.TextBoxFor(m=>m.Nama)
            </p>
            <p>
                @Html.LabelFor(m=>m.Alamat)
                @Html.TextBoxFor(m=>m.Alamat)
            </p>
            <p>
                @ViewBag.Pesan
            </p>
            <p>
                <input type="submit" value="Kirim" />
            </p>
        }
    </div>
</body>
</html>

```

Hasil dari modifikasi kode di atas membuat tampilan halaman web seperti berikut ini.



Gambar 27. Halaman sederhana ASP.NET MVC.

Untuk membuat form seperti yang terlihat pada gambar digunakan sintaks Razor yang dapat dilihat pada kode di atas. Penjelasan lebih lanjut tentang Razor akan dijelaskan pada bagian selanjutnya.

Setelah langkah di atas perlu dilakukan modifikasi pada class controller untuk menangani aksi ketika tombol Kirim ditekan. Berikut ini adalah hasil modifikasi pada class controller HomeController.cs.

```

HomeController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

using Implementasi.MVC.Models;

namespace Implementasi.MVC.Controllers
{

```

```

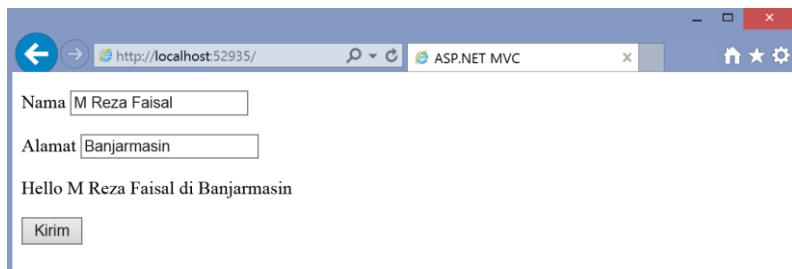
public class HomeController : Controller
{
    // GET: Home
    public ActionResult Index()
    {
        return View();
    }

    [HttpPost]
    public ActionResult Index(Visitor data)
    {
        ViewBag.Pesan = "Hello " + data.Nama + " di " + data.Alamat;
        return View();
    }
}

```

Pada kode di atas ditambahkan method baru untuk menangani aksi post ketika tombol Kirim diklik. Pada method ini terdapat perintah untuk menampung pesan pada ViewBag.Pesan yang nanti akan ditampilkan pada bagian View. Pada bagian selanjutnya akan dilakukan pembahasan lebih dalam tentang controller.

Selanjutnya adalah mencoba menjalankan halaman yang telah dibuat.



Gambar 28. Hasil halaman sederhana ASP.NET MVC.

Setelah tombol Kirim diklik maka akan dieksekusi method Index yang baru ditambahkan. Pada percobaan ini dapat dilihat model yang dibuat memang dapat digunakan sebagai sarana pengiriman informasi. Dapat dilihat pada baris kode di bawah ini, objek data yang bertipe dari class model Visitor dikirimkan saat tombol Kirim diklik. Kemudian dilakukan pengambilan informasi yang diperlukan dari objek tersebut untuk mengisi nilai pada ViewBag.Pesan yang kemudian akan ditampilkan kembali pada halaman view.

```

HomeController.cs
[HttpPost]
public ActionResult Index(Visitor data)
{
    ViewBag.Pesan = "Hello " + data.Nama + " di " + data.Alamat;
    return View();
}

```

Kesimpulan

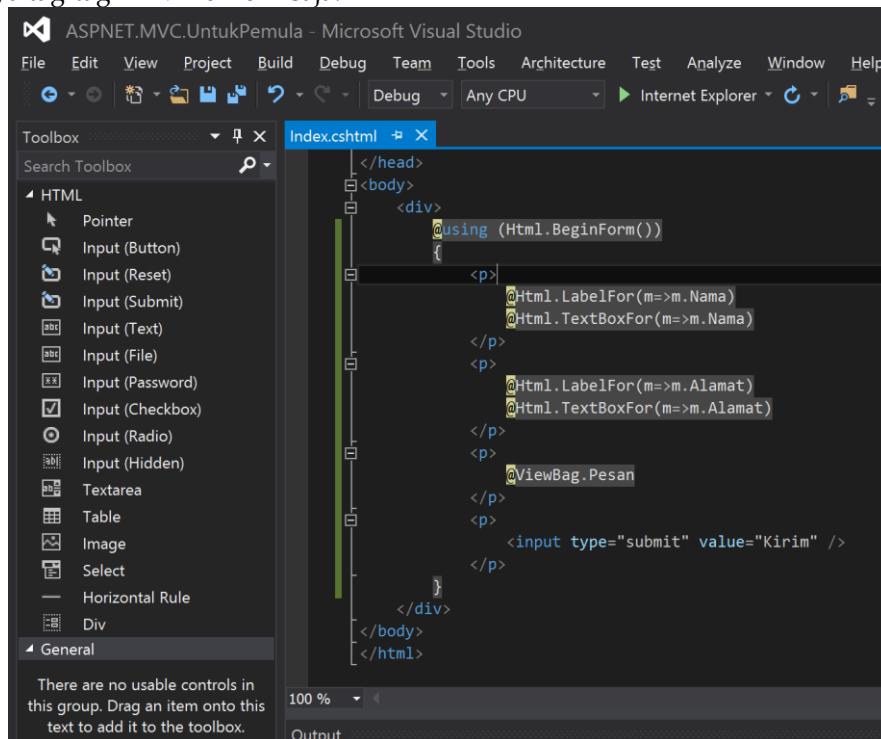
Ada beberapa hal yang dapat dilihat dari paparan ini yang menjadi karakteristik dari ASP.NET Web Forms, yaitu :

1. Sebuah halaman web form dapat dibuat dengan mudah.
2. Setiap halaman web form terdiri atas 2 file penting yaitu *.aspx dan file code behing (*.aspx.cs atau *.aspx.vb).

3. Antarmuka halaman web ditulis pada file *.aspx dapat terdiri atas HTML dan server control dari komponen yang tersedia.
4. File web forms yang dibuat dapat langsung diakses dan dilihat hasilnya dengan cara menuliskan pada address bar pada web browser.
5. Pada server control dapat memiliki event yang akan mengeksekusi event handler dengan cara yang telah dijelaskan pada paparan di atas.
6. Karakteristik stateful dapat dilihat dari nilai pada textbox sebagai control server yang tetap ada walau telah dilakukan proses post, dan pengelolaan nilai pada control server tersebut dilakukan secara otomatis oleh framework ASP.NET Web Forms.

Sedangkan pada paparan tentang framework ASP.NET MVC dapat disimpulkan beberapa hal sebagai berikut :

1. Diperlukan konfigurasi untuk mengelola pola url yang diinginkan untuk mengakses halaman. Pola url ini yang akan dituliskan pada address bar web browser untuk mengakses halaman yang diinginkan.
2. Untuk membuat sebuah halaman dengan menggunakan framework ini diperlukan minimal 2 file sebagai yaitu file yang berfungsi sebagai controller dan file yang berfungsi sebagai view.
3. Sebagai pelengkap dapat ditambahkan model untuk melengkapi bagian controller dan view, model dapat dipergunakan sebagai sarana untuk melakukan pertukaran informasi.
4. Pada framework ini tidak memiliki komponen server control seperti pada framework ASP.NET Web Forms. Terlihat pada saat file *.cshtml aktif, yang terlihat pada toolbox hanya tag-tag HTML umum saja.



Gambar 29. Isi Toolbox saat membuat file *.cshtml.

Dari kedua contoh di atas, dapat menjadi pengetahuan awal tentang perbedaan antara framework ASP.NET Web Forms dan MVC. Pengetahuan ini dapat bermanfaat bagi web developer yang telah lebih dahulu mengenal framework ASP.NET Web Forms, sehingga dapat diketahui hal-hal mendasar yang membedakan antara kedua framework tersebut. Sedangkan bagi software developer yang belum pernah menggunakan kedua framework ini,

pengetahuan ini dapat menjadi pengetahuan tambahan tentang framework dalam pengembangan aplikasi web.

Referensi

<https://msdn.microsoft.com/en-us/library/ff649595.aspx>

<https://msdn.microsoft.com/en-us/library/ff649415.aspx>

<https://msdn.microsoft.com/en-us/library/1xaas8a2%28v=vs.71%29.aspx>

Professional ASP.NET MVC 5, Wrox.

3

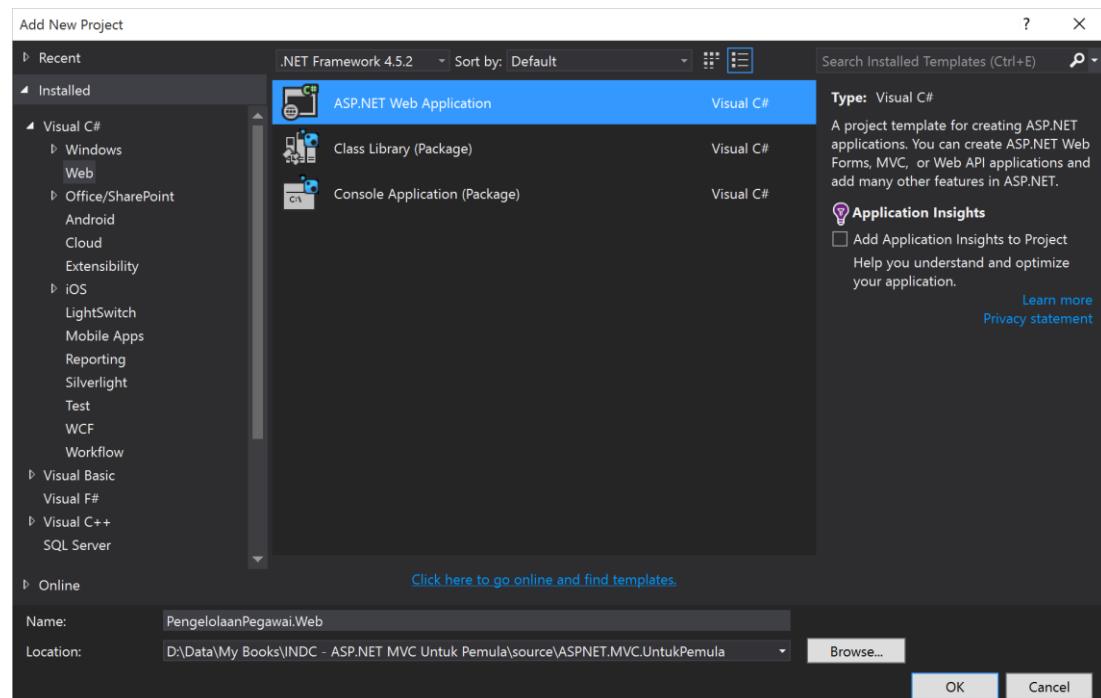
Persiapan

Pada bagian ini berisi hal-hal yang dipersiapkan sebelum mengenal lebih dalam mengenai framework ASP.NET MVC. Pada ebook ini diberikan contoh pembangunan aplikasi pengelolaan pegawai, sehingga persiapan yang dilakukan akan sesuai dengan kebutuhan dari aplikasi tersebut.

Untuk mengikuti ebook ini, maka pembaca diharapkan melakukan langkah-langkah persiapan yang akan dilakukan di ebook ini, hal ini untuk keseragaman sehingga pembaca dapat lebih mengerti perbahasan pada bagian selanjutnya.

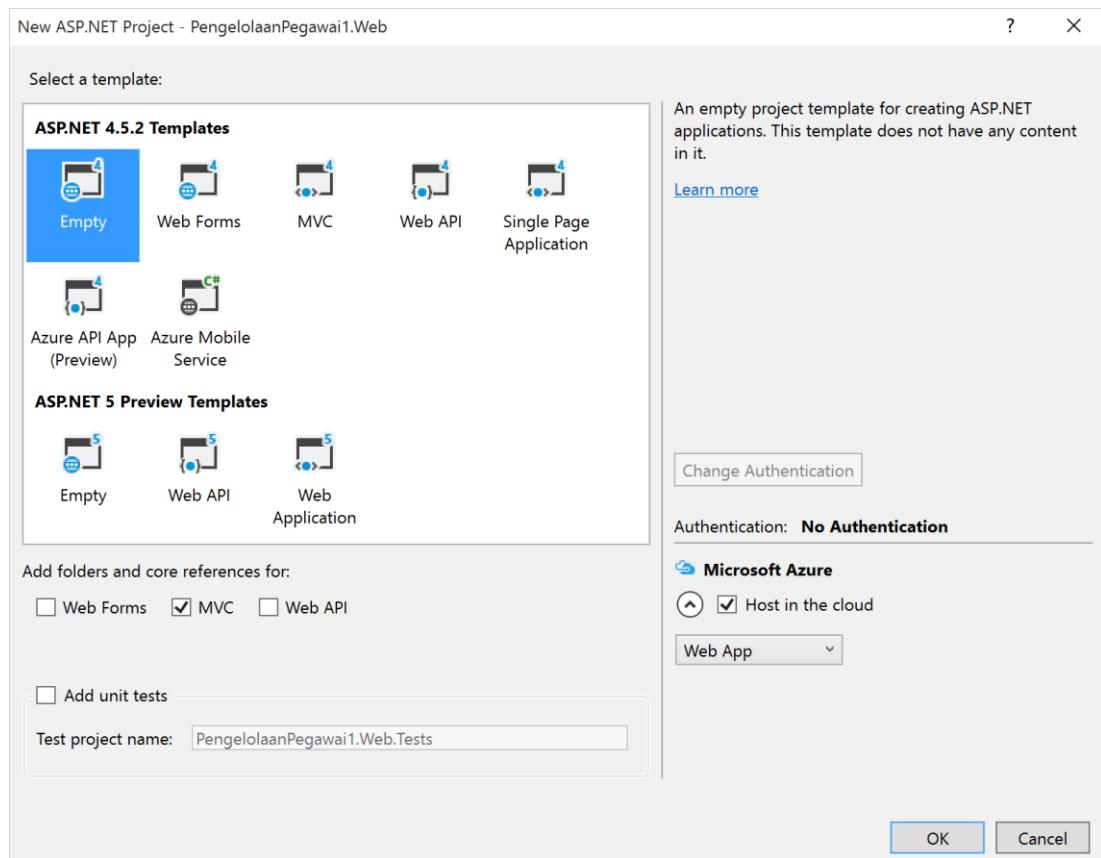
Project

Hal pertama yang harus dipersiapkan adalah membuat project, nama project yang akan dibuat adalah PengelolaanPegawai.Web. Untuk menambahkan project dapat dilakukan dengan memilih menu File > New > Project dengan cara ini maka akan dibuat solution dan project dengan nama yang sama. Sedangkan jika telah dibuat solution maka untuk membuat project dapat dilakukan dengan cara klik kanan pada solution yang ada pada bagian Solution Explorer kemudian pilih Add > New Project.



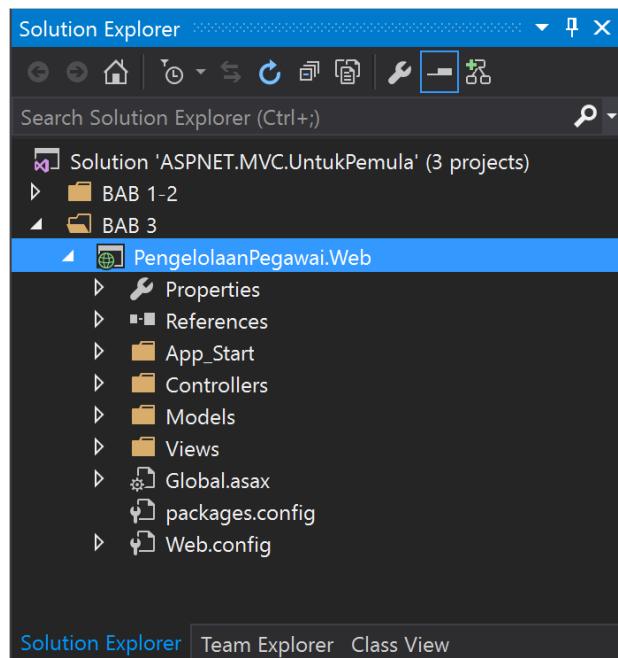
Gambar 30. Membuat project PengelolaanPegawai.Web.

Pada window Add New Project pilih Visual C# > Web kemudian pilih ASP.NET Web Application. Isi kolom input Name dengan PengelolaanPegawai.web, kemudian klik tombol OK. Maka akan ditampilkan window New ASP.NET Project seperti di bawah ini.



Gambar 31. Memilih template project web yang akan digunakan.

Pada window di atas, pilih Empty pada bagian ASP.NET 4.4.2 Template kemudian centang MVC pada bagian Add folders and core reference for. Kemudian klik tombol OK. Maka hasilnya dapat dilihat seperti pada gambar di bawah ini.



Gambar 32. Project Pengelolaan.Web.

Database

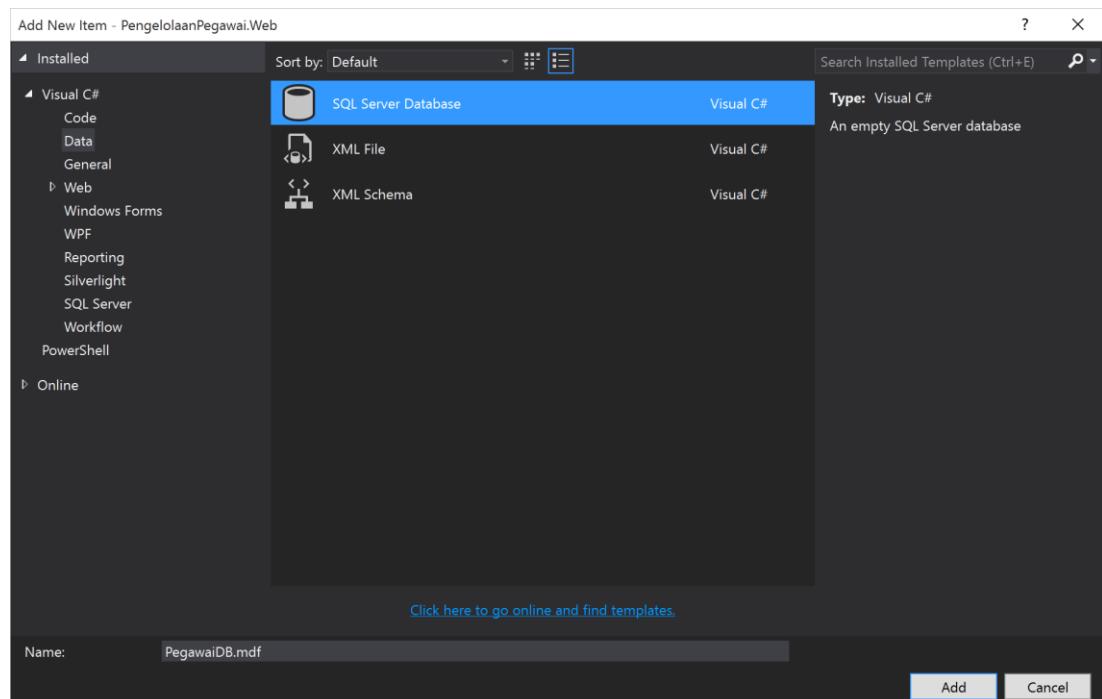
Untuk membuat aplikasi untuk mengelola pegawai, maka diperlukan database yang akan menyimpan data. Data yang akan dikelola pada aplikasi ini adalah divisi dan pegawai, sehingga database harus berisi dua tabel untuk menyimpan kedua data tersebut.

Pada aplikasi ini digunakan database SQL Express Edition, maka mesti dipastikan SQL Express Edition telah diinstall sebelum mengikuti langkah ini.

Dengan menggunakan SQL Express Edition sebagai database yang digunakan pada pembangunan aplikasi ini, maka file database dapat disimpan dalam folder yang ada pada project. Sehingga project dalam dijalankan pada mesin lain yang telah memiliki SQL Express Edition.

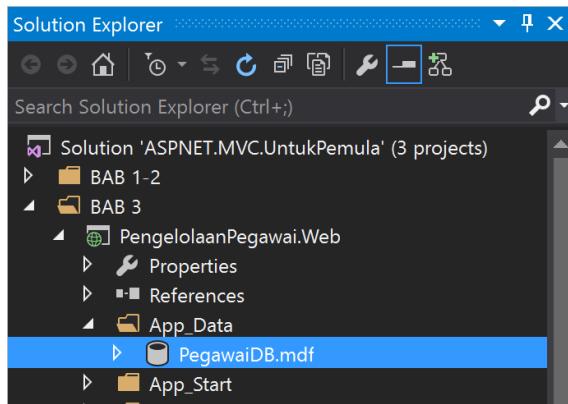
Langkah pertama yang dilakukan adalah membuat folder untuk menyimpan file database tersebut. Penamaan folder ini harus mengikuti aturan yang dimiliki oleh ASP.NET. Langkah pertama untuk membuat folder ini adalah klik kanan pada project yang telah dibuat di atas, kemudian pilih Add > Add ASP.NET Folder > App_Data. Folder App_Data adalah folder yang umumnya digunakan untuk menyimpan file data dengan format *.mdf (file SQL Server Express) atau *.xml.

Langkah kedua adalah menambahkan SQL Server Database ke dalam folder App_Data dengan cara klik kanan pada folder App_Data kemudian pilih Add > New Item. Pada window Add New Item pilih Visual C# > Data > SQL Server Database dan pada kolom input Name berikan nilai PegawaiDB.mdf sebagai nama database, kemudian klik tombol Add.



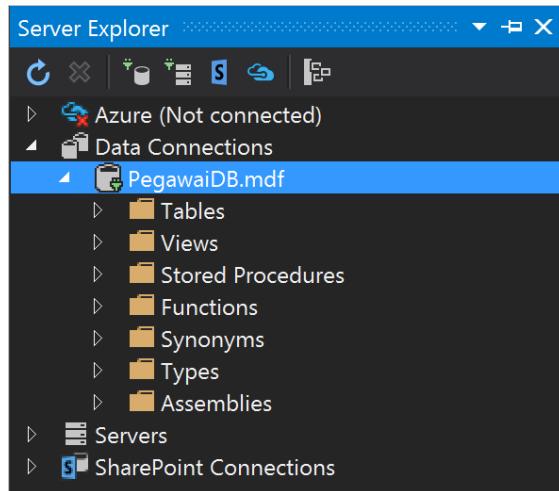
Gambar 33. Membuat file database.

Maka dapat dilihat file PegawaiDB.mdf pada Solution Explorer seperti pada gambar di bawah ini.



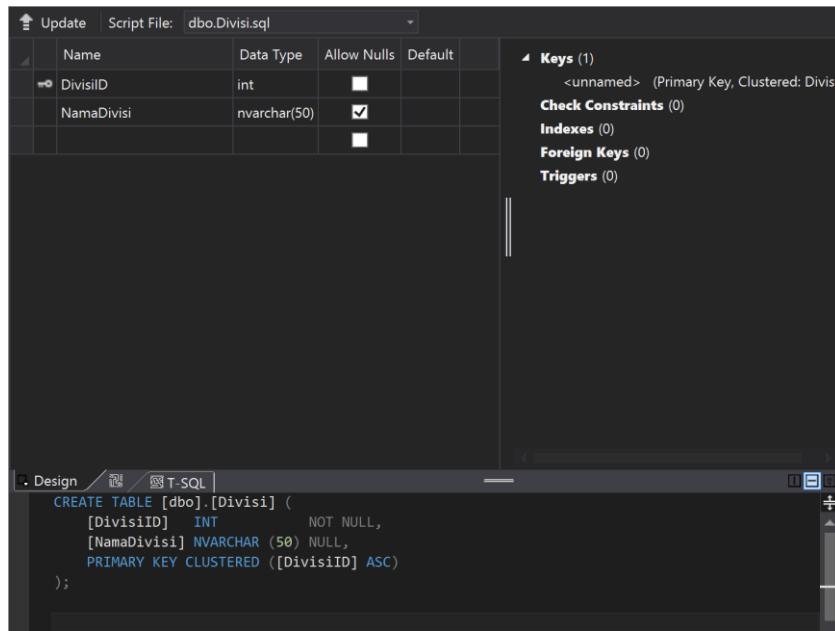
Gambar 34. File PegawaiDB.mdf.

Setelah database dibuat maka langkah selanjutnya adalah membuat tabel. Pada Visual Studio telah dilengkapi fitur sebagai tool pengelolaan database. Maka pada langkah ketiga dimulai dengan mengaktifkan tool tersebut dengan cara mengklik double pada file PegawaiDB.mdf. Jika SQL Server Express maka tool ini dapat dilihat pada area Server Explorer seperti gambar di bawah ini.



Gambar 35. Server Explorer – Data Connection.

Untuk menambah tabel pada database PegawaiDB.mdf dapat dilakukan dengan cara klik kanan pada folder Tables kemudian pilih Add New Table, maka akan ditampilkan seperti pada gambar berikut.



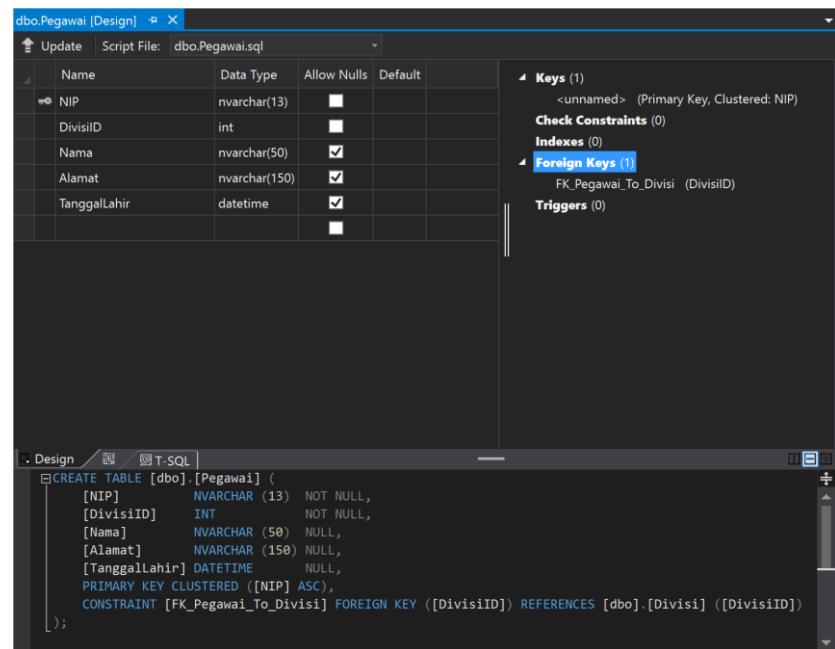
Gambar 36. Membuat tabel Divisi.

```

Divisi.sql
CREATE TABLE [dbo].[Divisi] (
    [DivisiID] INT NOT NULL,
    [NamaDivisi] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([DivisiID] ASC)
);
  
```

Buat field-field untuk tabel Divisi seperti pada contoh di atas, kemudian pada area T-SQL ubah nama tabel dengan nama Divisi. Untuk membuat tabel sesuai dengan desain yang telah dibuat, klik tombol Update yang ada pada posisi kiri atas.

Setelah tabel Divisi berhasil dibuat, maka dilanjutkan dengan membuat tabel Pegawai dengan nama field seperti pada gambar di bawah ini.



Gambar 37. Membuat tabel Pegawai.

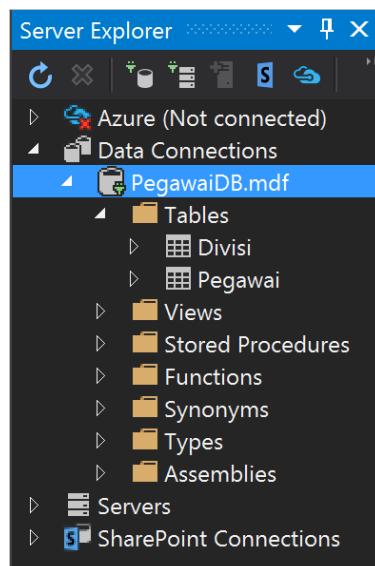
```

Pegawai.sql
CREATE TABLE [dbo].[Pegawai] (
    [NIP]          NVARCHAR (13) NOT NULL,
    [DivisiID]      INT          NOT NULL,
    [Nama]          NVARCHAR (50) NULL,
    [Alamat]        NVARCHAR (150) NULL,
    [TanggalLahir]  DATETIME    NULL,
    PRIMARY KEY CLUSTERED ([NIP] ASC),
    CONSTRAINT [FK_Pegawai_To_Divisi] FOREIGN KEY ([DivisiID]) REFERENCES
    [dbo].[Divisi] ([DivisiID])
);

```

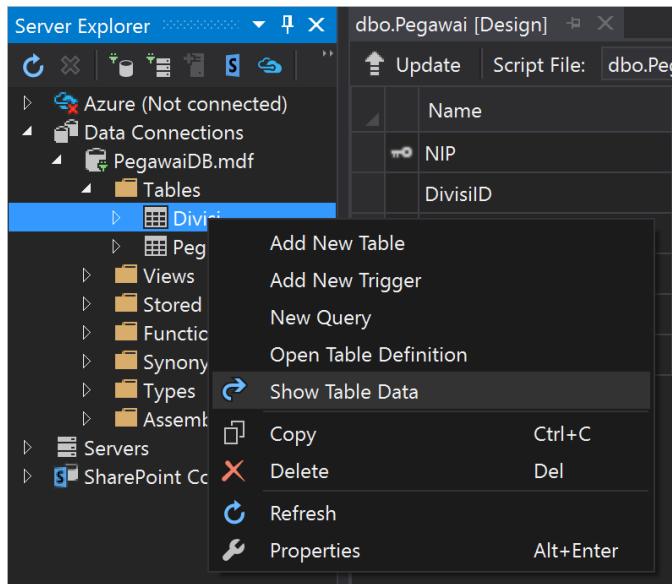
Pada tabel ini perlu dibuat foreign key yang menjadi relasi ke tabel Divisi. Foreign key pada tabel Pegawai adalah DivisiID, untuk membuat foreign key ini dapat dilakukan dengan cara klik kanan pada Foreign Key kemudian pilih Add New Foreign Key kemudian pada T-SQL akan ditambahkan baris baru dengan awalan CONSTRAINT. Ubah baris baru tersebut menjadi seperti pada gambar di atas. Setelah nama tabel pada T-SQL diganti dengan nama Pegawai, klik tombol Update.

Jika proses pembuatan kedua tabel itu sukses akan dapat dilihat tabel Divisi dan Pegawai di dalam database PegawaiDB.mdf.



Gambar 38. Tabel Divisi dan Pegawai pada PegawaiDB.mdf.

Selanjutnya dapat dilakukan pengisian data awal pada kedua tabel tersebut. Untuk tabel Divisi, pengisian data awal dapat dilakukan dengan klik kanan pada tabel Divisi kemudian pilih Show Table Data.



Gambar 39. Menu Show Table Data.

Isi record pada tabel ini dengan nilai-nilai seperti pada gambar di bawah ini. Untuk mengisi data ini dapat dilakukan dengan cara seperti mengisi nilai-nilai dengan menggunakan MS Excel.

	DivisiID	NamaDivisi
1	Divisi Kepengawaian	
2	Divisi Umum	
NULL	NULL	

Gambar 40. Mengisi data pada tabel Divisi.

Dengan cara yang sama, lakukan pengisian data pada tabel Pegawai dengan nilai seperti pada gambar berikut.

	NIP	DivisiID	Nama	Alamat	TanggalLahir
123	1	Mohammad	Banjarmasin	12/12/1980 12:00:00 AM	
234	1	Faisal	Banjarbaru	7/7/1988 12:00:00 AM	
NULL	NULL	NULL	NULL	NULL	NULL

Gambar 41. Mengisi data pada tabel Pegawai.

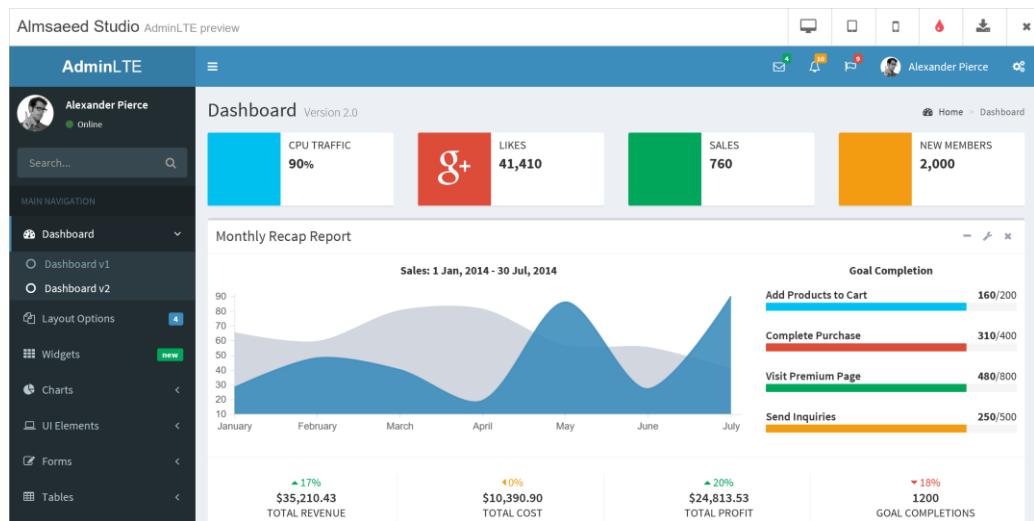
Theme

Dalam pembangunan software antarmuka merupakan hal penting, dengan antarmuka (user interface) yang bagus dan user experience yang mudah akan membuat software mudah digunakan oleh penggunanya. Untuk aplikasi web, antarmuka yang dibuat dapat digunakan dalam berbagai lingkungan. Agar aplikasi web dapat digunakan oleh banyak pengguna,

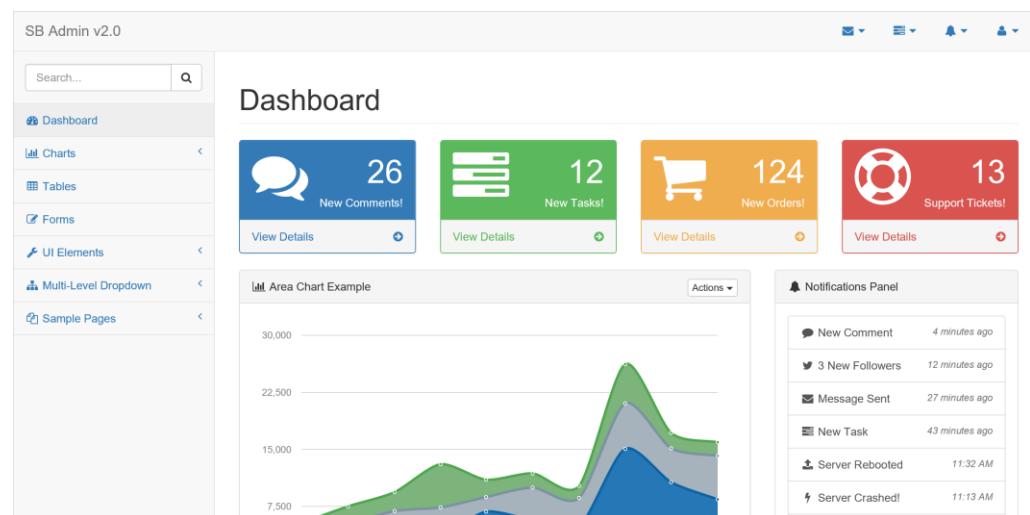
maka aplikasi web harus dibangun dengan antarmuka yang dapat digunakan pada lingkungan pengguna. Saat ini lingkungan pengguna sangat beragam, beragam jenis web browser (Internet Explorer, Microsoft Edge, Firefox, Chrome, Safari, Opera dan lain-lain), beragam ukurang layar dan beragama device (komputer desktop, laptop, smartphone dan lain-lain).

Salah satu framework yang dapat digunakan agar aplikasi web dengan tujuan di atas adalah Bootstrap. Bootstrap merupakan framework front-end yang menggunakan HTML, CSS dan JavaScript yang dapat membuat aplikasi web yang responsive dan dapat digunakan pada lingkungan perangkat mobile.

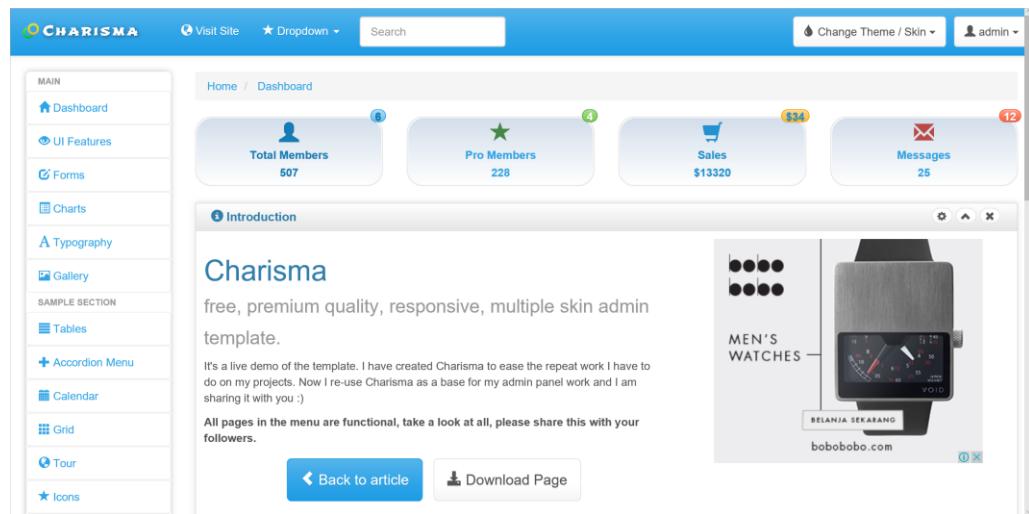
Aplikasi pengelolaan pegawai yang akan dibangun akan menggunakan theme bootstrap yang dengan fungsi khusus yaitu Bootstrap Admin Theme. Sekarang ini telah banyak tersedia theme gratis yang dapat digunakan, sebagai contoh yang dapat dilihat pada link berikut ini <http://speckyboy.com/2014/05/16/free-bootstrap-admin-themes/>. Berikut adalah contoh antarmuka theme yang ada pada link tersebut.



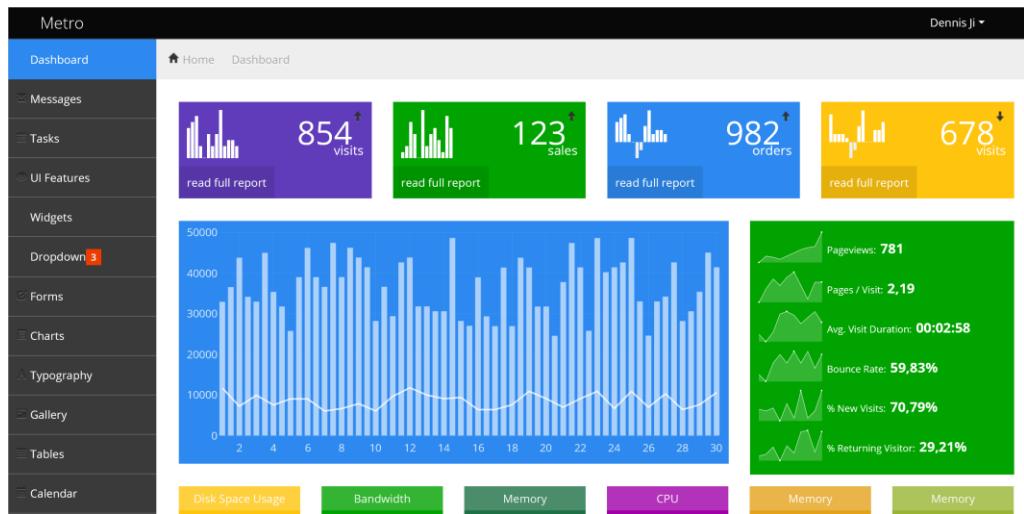
Gambar 42. AdminLTE Control Panel Template.



Gambar 43. SB Admin 2.

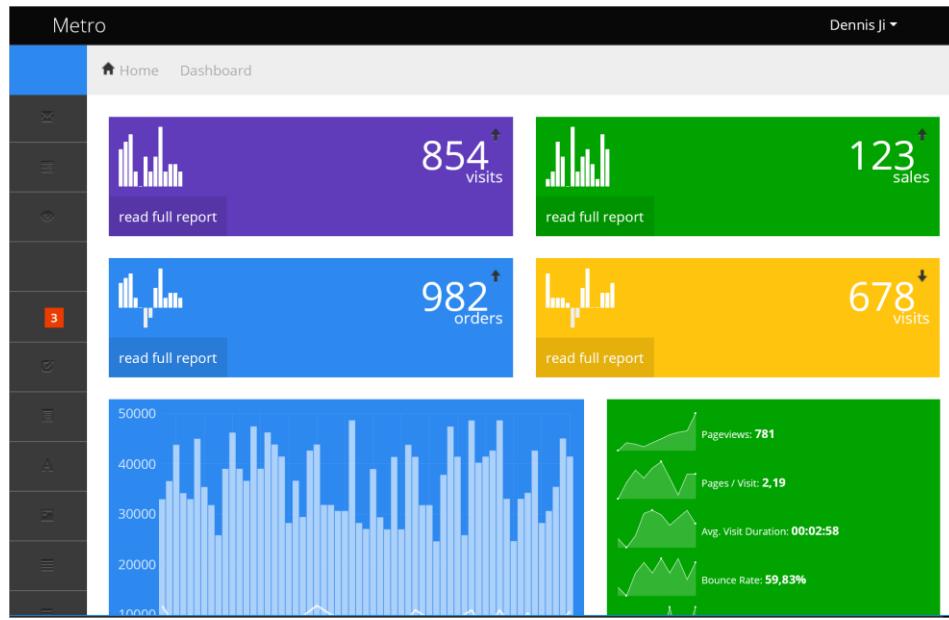


Gambar 44. Charisma Responsive Admin Template.



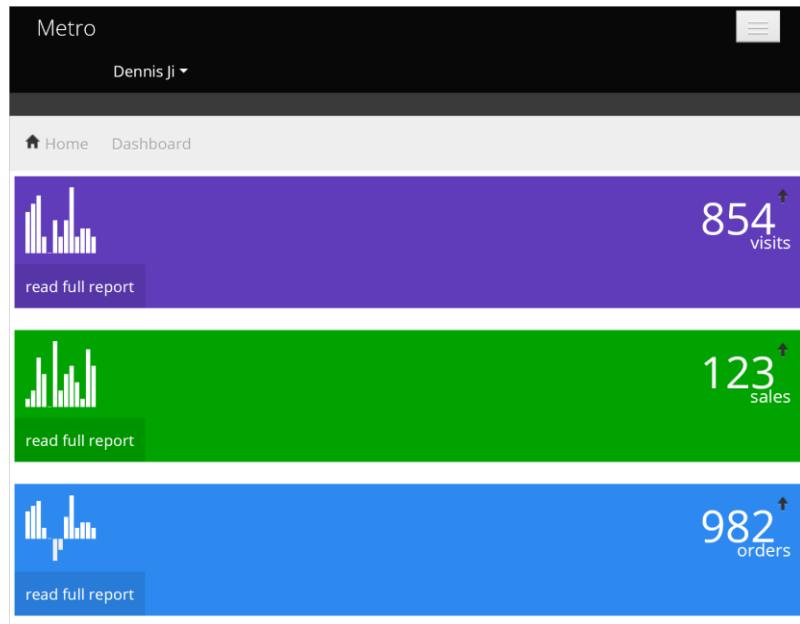
Gambar 45. Metro Dashboard – A Modern and Clean Admin Template.

Dari beberapa theme yang tersedia tersebut, dipilih yang akan digunakan pada pembangunan aplikasi pengelolaan pegawai pada ebook ini. Theme yang dipilih adalah Metro Dashboard yang dapat dilihat pada gambar di atas. Gambar di atas adalah antarmuka theme ini ketika diakses oleh ukuran layar yang besar. Ketika theme ini diakses oleh perangkat dengan ukuran layar yang lebih kecil maka dapat dilihat penyesuaian antarmuka seperti berikut ini.



Gambar 46. Antarmuka theme Metro Dashboard pada layar yang lebih kecil.

Sedangkan jika theme ini diakses oleh device dengan layar yang lebih kecil, seperti pada smartphone maka antarmuka akan menyesuaikan dengan ukuran dan menu seperti yang umumnya ditemui pada antarmuka aplikasi mobile.



Gambar 47. Antarmuka theme Metro Dashboard pada layar smartphone.

Implementasi theme ini pada aplikasi yang akan dibangun pada ebook ini akan diterangkan lebih lanjut pada bagian selanjutnya yaitu saat membahas tentang View pada Bab 6.

Referensi

<http://www.w3schools.com/bootstrap/>

<http://speckyboy.com/2014/05/16/free-bootstrap-admin-themes/>

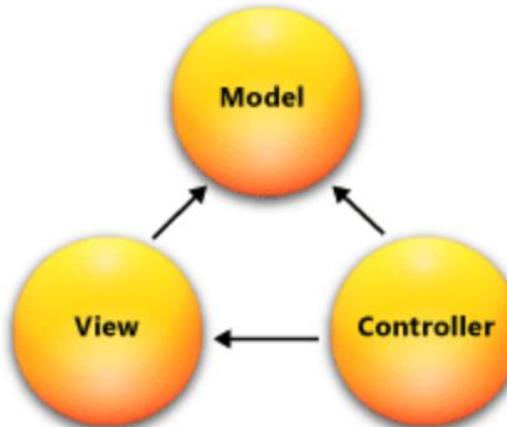
4

Model

Pada bab ini dibahas tentang Model sebagai komponen pertama pada pattern Model-View-Controller (MVC). Di dalam bab ini akan dijelaskan sedikit tentang konsep model dan perannya pada framework ASP.NET MVC, dan kemudian akan ditunjukkan pembuatan model dengan bantuan Visual Studio. Contoh model yang dibuat pada bagian ini akan disesuaikan dengan keperluan dari aplikasi pengelolaan pegawai yang dibangun.

Definisi

Berikut ini adalah ilustrasi pattern Model-View-Controller (MVC).



Gambar 48. Komponen model pada pattern MVC.

Pada project model dapat dilihat sebagai class. Di dalam setiap class model tersebut terdapat method-method sebagai logika yang mengelola permasalahan pengelolaan data.

Dari penjelasan tersebut maka sering ditemui pernyataan bahwa object model adalah bagian aplikasi yang mengimplementasikan logika sebagai data domain aplikasi. object model mengambil dan menyimpan nilai state mode pada database. Sebagai contoh, sebuah object Pegawai dapat mengambil data pegawai dari database, melakukan operasi pada database dan kemudian mengupdate kembali informasi tersebut ke tabel Pegawai yang ada pada database PegawaiDB pada SQL Server.

Selain penjelasan di atas, model juga berfungsi sebagai media untuk pertukaran informasi antara komponen controller dan view. Fungsi ini adalah fungsi utama dari model. Selain itu, model juga dapat digunakan oleh komponen view sebagai acuan untuk membuat antarmuka. Pada framework ASP.NET MVC, dengan fungsi-fungsi Razor maka dapat dibuat antarmuka dengan menggunakan komponen model. Hal ini dapat dilihat dalam contoh yang telah dikerjakan pada Bab 2.

Jenis Model

Pada Bab 2 telah ditunjukkan bagaimana cara untuk membuat class model dan penggunaannya. Pada bab ini akan diperlihatkan dua jenis model yang dapat dibuat, yaitu :

1. Class model yang dibuat dengan menggunakan bantuan tool Entity Framework. Komponen model ini juga akan berfungsi sebagai Data Access Layer, yang berisi method-method untuk memudahkan dalam melakukan operasi database seperti melakukan koneksi dan operasi CRUD (create, retrieve, update, delete).
2. Class model yang dibuat secara manual seperti yang telah dilakukan pada Bab 2, class ini akan dibuat untuk membantu membuat antarmuka sesuai dengan kebutuhan.

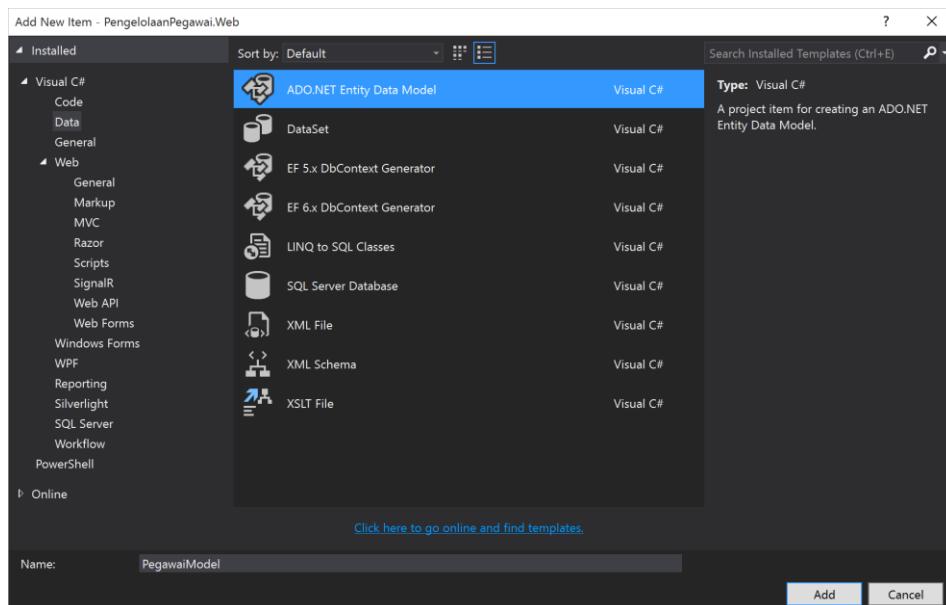
Data Access Layer

Pada bagian ini akan ditunjukkan membuat model yang akan berfungsi sebagai Data Access Layer dengan bantuan Entity Framework. Sebagai informasi, Data Access Layer yang dibuat dengan Entity Framework dapat dibedakan menjadi dua konsep, yaitu :

1. Code First, konsep ini dimulai dengan membuat class context dan model terlebih dahulu. Saat aplikasi dijalankan pertama kali maka akan secara otomatis akan membuat database dan tabel-tabel sesuai dengan model yang telah dibuat.
2. Database First, konsep ini dimulai dengan membuat database dan tabel-tabel yang diperlukan terlebih dahulu. Kemudian class context dan class model akan dibuat secara otomatis dengan bantuan generator pada Entity Framework.

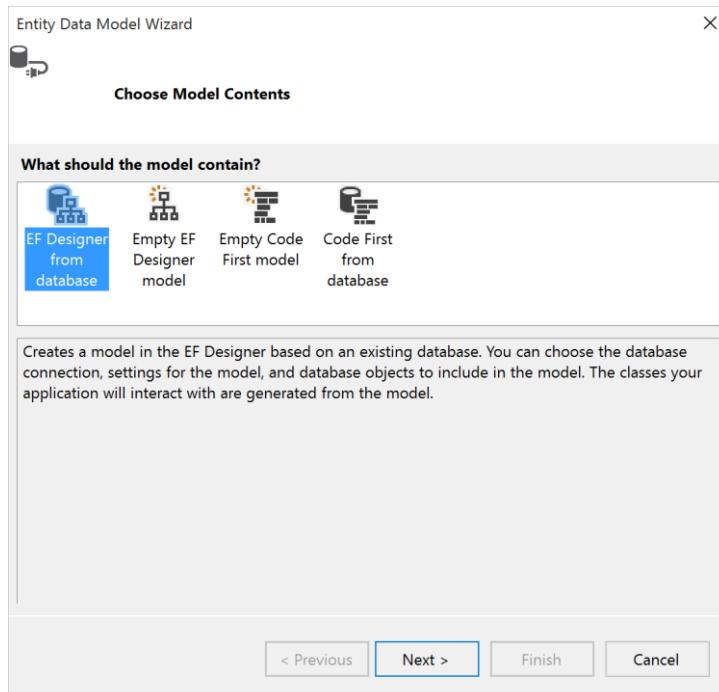
Pada bagian ini akan digunakan konsep yang kedua, yaitu Database First. Penggunaan konsep ini dikarenakan sebelumnya telah dilakukan persiapan pada Bab 3, dimana telah dibuat database dan tabel-tabel untuk keperluan pembuatan aplikasi pengelolaan pegawai.

Seperti class model yang sebelumnya telah dibuat, class model ini juga harus disimpan pada folder Models. Maka langkah pertama yang dilakukan adalah dengan klik kanan pada folder Models kemudian pilih Add > New Item.



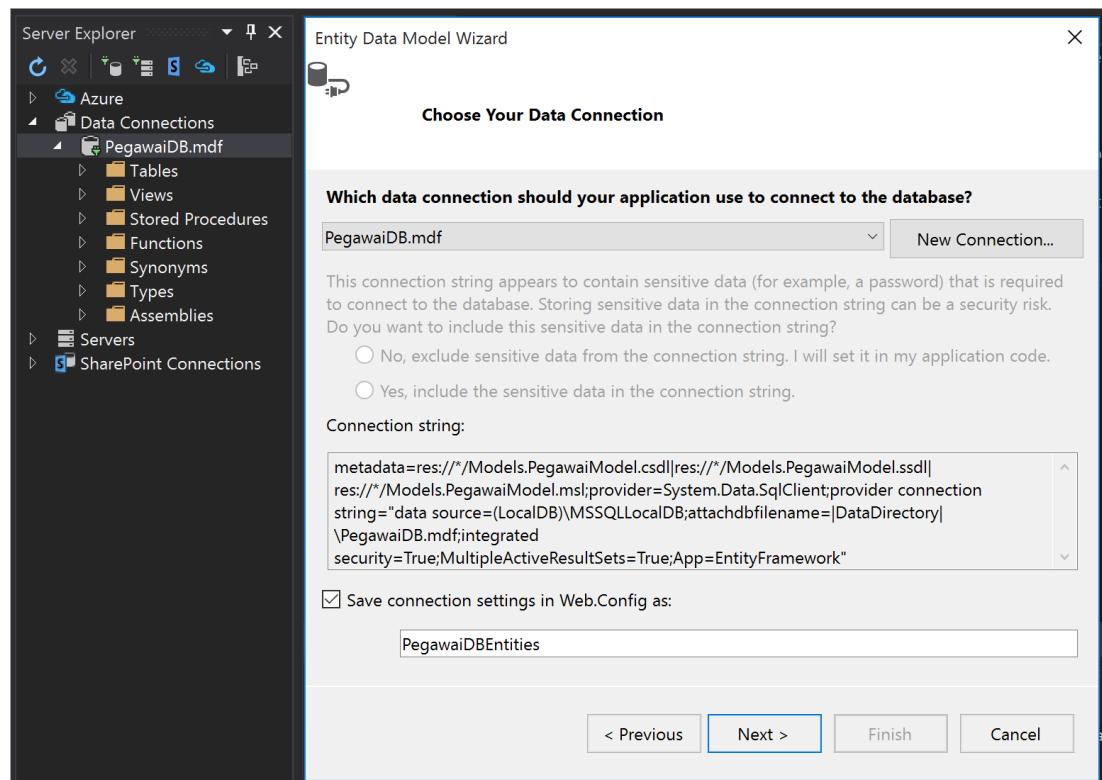
Gambar 49. Membuat PegawaiModel.

Pada window Add New Item, pilih Visual C# > Data > ADO.NET Entity Data Model. Kemudian pada kolom input Name isikan nilai PegawaiModel. Kemudian akan ditampilkan window Entity Data Model Wizard.



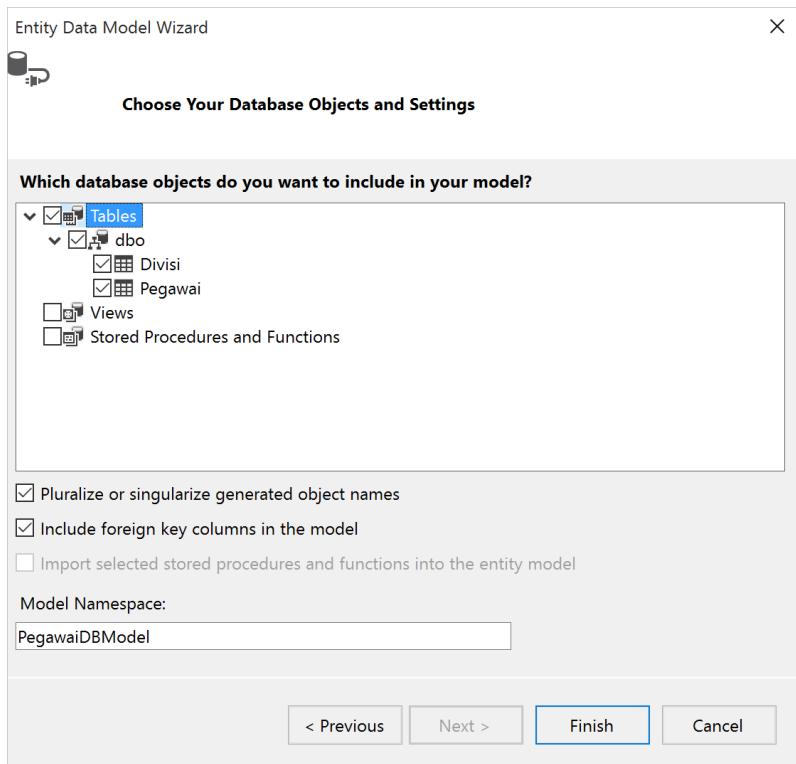
Gambar 50. Memilih cara untuk membuat model.

Karena akan digunakan konsep Database First, maka dipilih EF Designer from database kemudian klik tombol Next.



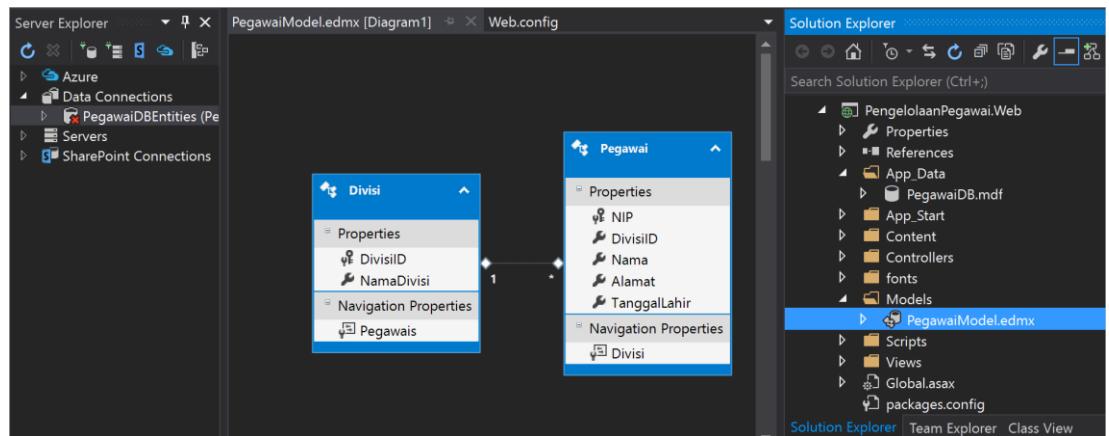
Gambar 51. Memilih koneksi data.

Pada window di atas dapat dipilih PegawaiDB.mdf sebagai database yang akan digunakan sebagai acuan membuat class context dan class model. Pada bagian ini, secara otomatis akan dibuatkan connection string dengan nama PegawaiDBEntities pada Web.Config. Kemudian klik tombol Next.



Gambar 52. Memilih tabel yang digunakan.

Pilih seluruh tabel dengan cara dicentang, kemudian klik tombol Finish. Jika semua proses yang telah dilakukan di atas berhasil, maka akan dapat dilihat hasilnya seperti gambar di bawah ini.



Gambar 53. PegawaiModel.edmx.

Pada gambar di atas dapat dilihat pada folder Models terdapat file hasil pembuatan secara otomatis yaitu PegawaiModel.edmx. Di dalam file tersebut terdapat class context yaitu PegawaiModel.Context.cs dengan isi sebagai berikut.

```
PegawaiModel.Context.cs
```

```
//-----  
// <auto-generated>
```

```

//      This code was generated from a template.
//
//      Manual changes to this file may cause unexpected behavior in your
//      application.
//      Manual changes to this file will be overwritten if the code is
//      regenerated.
// </auto-generated>
//-----
-----

namespace PengelolaanPegawai.Web.Models
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;

    public partial class PegawaiDBEntities : DbContext
    {
        public PegawaiDBEntities()
            : base("name=PegawaiDBEntities")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Divisi> Divisis { get; set; }
        public virtual DbSet<Pegawai> Pegawais { get; set; }
    }
}

```

Sedangkan setiap tabel yang telah dipilih akan dibuatkan masing-masing sebuah class model. Untuk tabel Divisi akan dibuat class Divisi yang disimpan pada file Divisi.cs.

Divisi.cs
<pre> //----- // ----- // <auto-generated> // This code was generated from a template. // // Manual changes to this file may cause unexpected behavior in your // application. // Manual changes to this file will be overwritten if the code is // regenerated. // </auto-generated> //----- // ---- namespace PengelolaanPegawai.Web.Models { using System; using System.Collections.Generic; public partial class Divisi { [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")] public Divisi() { this.Pegawais = new HashSet<Pegawai>(); } } } </pre>

```

    }

    public int DivisiID { get; set; }
    public string NamaDivisi { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
    "CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Pegawai> Pegawais { get; set; }
}
}

```

Sedangkan tabel Pegawai akan dibuatkan class Pegawai yang disimpan pada file Pegawai.cs dengan isi sebagai berikut.

```

Pegawai.cs
//-----
-----
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your
// application.
//     Manual changes to this file will be overwritten if the code is
// regenerated.
// </auto-generated>
//-----
-----

namespace PengelolaanPegawai.Web.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Pegawai
    {
        public string NIP { get; set; }
        public int DivisiID { get; set; }
        public string Nama { get; set; }
        public string Alamat { get; set; }
        public Nullable<System.DateTime> TanggalLahir { get; set; }

        public virtual Divisi Divisi { get; set; }
    }
}

```

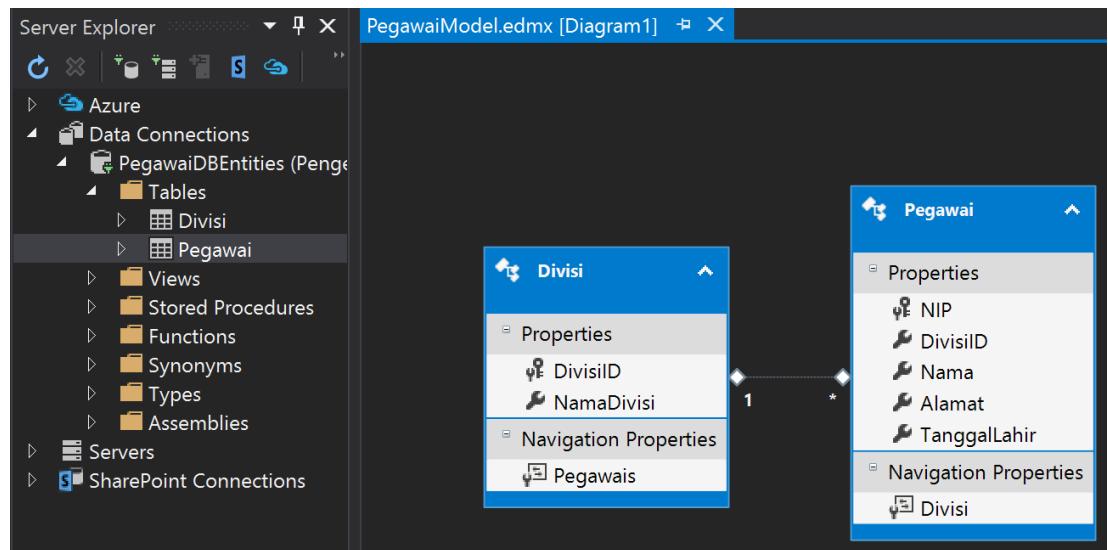
Penggunaan class context dan class model akan dijelaskan pada pembahasan tentang komponen controller dan view pada bab selanjutnya.

View Model

Istilah view model diberikan kepada class model yang berfungsi sebagai mengatur tampilan pada halaman view.

PegawaiViewModel

Sebagai contoh untuk data Pegawai, telah dibuat class model Pegawai seperti yang dilihat pada diagram berikut.



Gambar 54. Class model Pegawai.

Dengan property-property tersebut maka akan ditampilkan data seperti tabel berikut ini.

	NIP	DivisiID	Nama	Alamat	TanggalLahir
▷	123	1	Mohammad	Banjarmasin	12/12/1980 12:00:00 AM
	234	1	Faisal	Banjarbaru	7/7/1988 12:00:00 AM

Gambar 55. Isi data Pegawai.

Pada data tersebut dapat dilihat terdapat field DivisiID dengan nilai untuk masing-masing record yang berisi nilai berupa angka yang sesuai dengan primary key pada data Divisi.

	Divisi...	NamaDivisi
▷	1	Divisi Kepergawain
	2	Divisi Umum

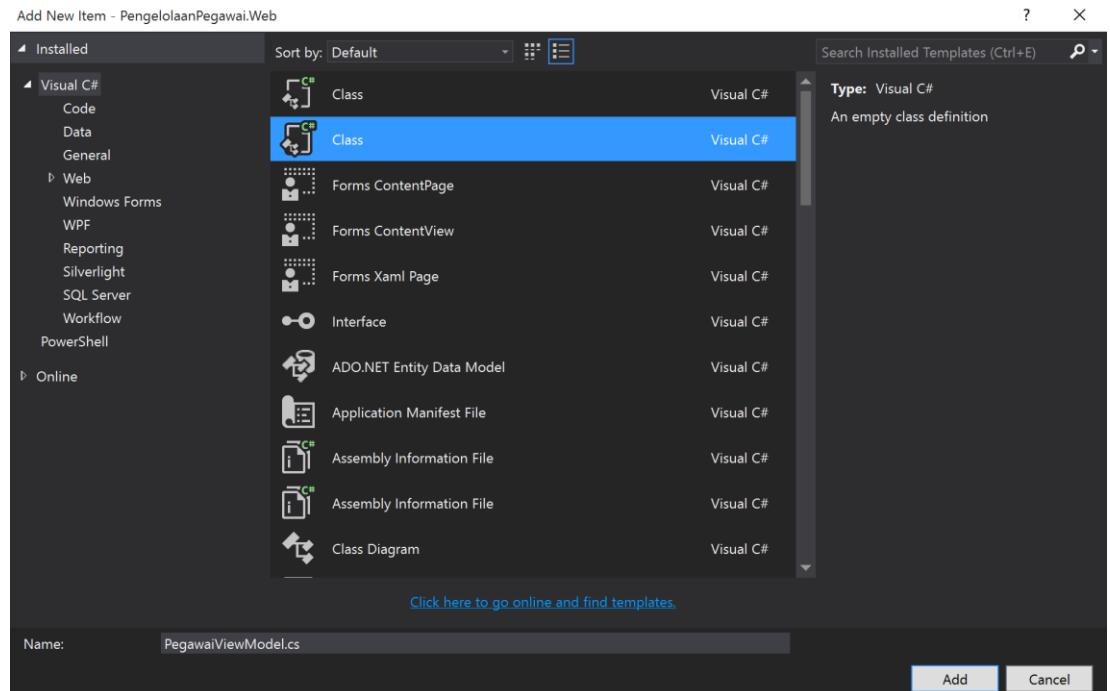
Gambar 56. Isi data Divisi.

Jika pada aplikasi ditampilkan data pegawai seperti pada gambar di atas, maka informasi yang disampaikan kepada pengguna menjadi kurang lengkap. Akan lebih bagus jika data yang ditampilkan pada halaman aplikasi seperti berikut ini.

NIP	Nama Divisi	Nama	Alamat	TanggalLahir
123	Divisi Kepergawain	Mohammad	Banjarmasin	12/12/1980 12:00:00 AM
234	Divisi Kepergawain	Faisal	Banjarbaru	7/7/1988 12:00:00 AM

Untuk menampilkan data seperti itu maka perlu dibuat model yang sesuai antarmuka untuk menampilkan data di atas.

Cara untuk membuat class model ini dapat dimulai dengan klik kanan pada folder Models, kemudian pilih Add > Class. Kemudian pilih Visual C# > Class dan pada kolom input Name isikan nilai PegawaiViewModel.cs. Penambahan kata "ViewModel" pada nama class model sering digunakan untuk menandakan bahwa class tersebut berfungsi sebagai sarana untuk mengirimkan data dan tampilan sesuai dengan keinginan.



Gambar 57. Membuat class PegawaiViewModel.

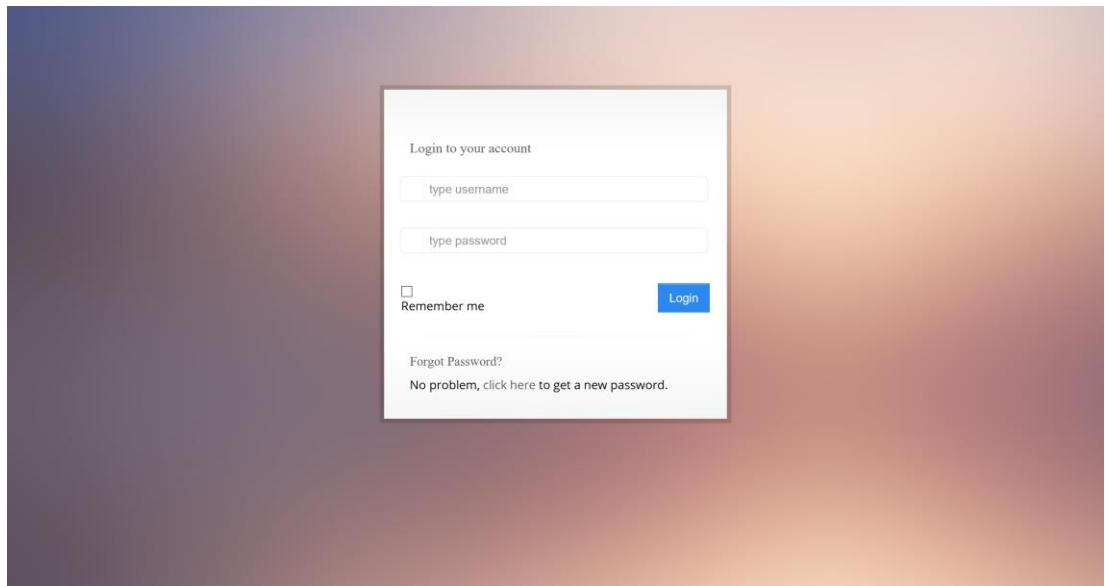
Berikut ini adalah isi dari file PegawaiViewModel.cs, pada class ini terdapat property dengan nama NamaDivisi yang akan berfungsi untuk menyimpan data nama divisi.

```
PegawaiViewModel.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PengelolaanPegawai.Web.Models
{
    public class PegawaiViewModel
    {
        public string NIP { set; get; }
        public string NamaDivisi { set; get; }
        public string Nama { set; get; }
        public string Alamat { set; get; }
        public DateTime TanggalLahir { set; get; }
    }
}
```

LoginViewModel

Class model lain yang dapat dibuat untuk melengkapi kebutuhan aplikasi yang akan dibuat adalah untuk keperluan membuat form login seperti gambar berikut ini.



Gambar 58. Form login.

Jika diperlukan model untuk membuat antarmuka pada view seperti gambar di atas untuk keperluan pertukaran informasi login, maka dapat dibuat class LoginViewModel dengan isi sebagai berikut.

```
>LoginViewModel.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PengelolaanPegawai.Web.Models
{
    public class LoginViewModel
    {
        public string Username { set; get; }
        public string Password { set; get; }
        public bool IsRemember { set; get; }
    }
}
```

Kesimpulan

Dari pembahasan dan contoh di atas, maka didapat beberapa kesimpulan sebagai berikut :

1. Fungsi utama model adalah sebagai sarana untuk pertukaran informasi antara view dan controller.
2. Model dapat digunakan sebagai acuan untuk membuat antarmuka pada bagian view.
3. Model dapat dibuat dengan bantuan tool Entity Framework, dengan tool ini akan dibuatkan class context dan class model yang menjadi Data Access Layer pada aplikasi.

Referensi

[https://msdn.microsoft.com/en-us/library/dd381412\(v=vs.108\).aspx](https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx)

5

View-Controller

Pada bagian ini membahas dua komponen pada pattern MVC yaitu View dan Controller. Kedua komponen ini sengaja dibahas di dalam bab ini karena agak susah untuk menentukan mana yg lebih dahulu jika dibahas salah satu saja secara bergantian. Keduanya perlu dibahas bersamaan karena contoh-contohnya saling terkait satu sama lain.

Definisi

Pada akan diterangkan definisi view dan controller.

View

View adalah komponen yang bertugas untuk menampilkan antarmuka aplikasi. Biasanya antarmuka dibuat berdasarkan model. Sebagai contoh antarmuka daftar untuk menampilkan atau form untuk input data Pegawai dibuat berdasarkan object Pegawai. Pada komponen ini informasi akan ditampilkan.

Controller

Controller adalah komponen yang menangani interaksi user, menentukan view yang akan ditampilkan dan menghubungkan dengan model. Sebagai contoh, user dapat berinteraksi dengan controller dengan cara menulis nama controller dan aksi yang diinginkan pada address bar di web browser, kemudian respon dengan memilihkan view yang akan ditampilkan sesuai dengan aksi yang telah dipanggil. Sebelum memilihkan view yg akan ditampilkan, aksi pada controller dapat berisi logika yang berisi baris perintah seperti perintah untuk melakukan pengambilan data ke database kemudian data akan dimasukkan ke dalam collection yang berisi object model, dan kemudian menampilkannya pada view yang diinginkan.

Cara Kerja

Pada bagian ini akan dijelaskan bagaimana cara kerja framework ASP.NET MVC hanya dengan memanfaatkan komponen view dan controller saja. Dengan mengetahui cara kerja dan pemanfaatan kedua komponen ini, maka pembaca sudah dapat membuat halaman yang bisa dieksekusi agar hasilnya dapat dilihat pada web browser.

Pada pattern MVC, komponen controller adalah komponen utama dan paling penting. Hal ini juga diterapkan pada framework ASP.NET MVC dimulai dengan mengatur cara untuk mengakses dan memanfaatkan controller.

Pada ASP.NET MVC terdapat class RouteConfig.cs pada folder App_Start yang berfungsi untuk mengatur bagaimana cara mengakses dan memanfaatkan controller.

```

RouteConfig.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace PengelolaanPegawai.Web
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}

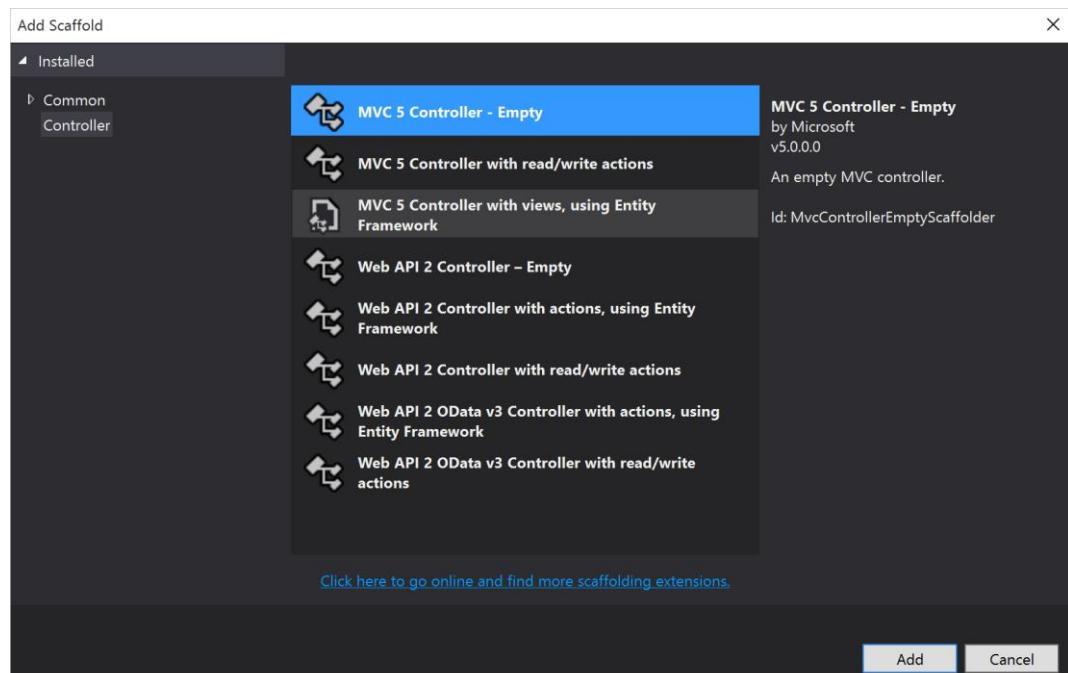
```

Pada class tersebut dapat dilihat format yang harus diikuti untuk mengakses controller dan memilih aksi yang diinginkan, seperti pada baris yang dicetak tebal dengan tanda garis bawah. Dengan mengetahui format tersebut maka dapat diketahui alamat yang harus dituliskan pada address bar di web browser untuk memanggil aksi pada controller, yaitu dengan format seperti berikut :

```
http://[nama server/ip address server]/[nama controller]/[nama aksi]/[id]
```

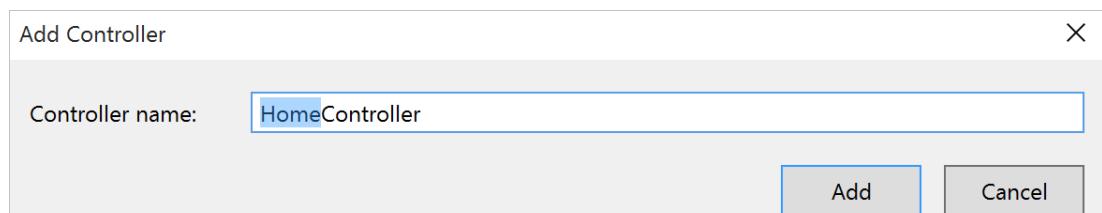
Selanjutnya akan dibuat sebuah class controller yang berisi beberapa aksi, dan akan diperlihatkan memanggil setiap aksi pada controller tersebut dengan format alamat di atas. Framework ASP.NET MVC mengatur untuk meletakkan semua class controller pada folder Controllers. Aturan lain yang harus diketahui adalah penamaan file dan class controller, dimana setiap nama file dan nama class controller harus diakhiri dengan kata Controller. Sehingga jika ingin memiliki controller dengan nama Home maka nama file dan class menjadi HomeController. Contoh lain jika ingin membuat controller dengan nama Login, maka nama file dan class yang harus dibuat adalah LoginController

Untuk membuat class controller ini dapat dimulai dengan mengklik kanan pada folder Controllers kemudian pilih Add > Controller dan pada window Add Scaffold pilih MVC 5 Controller – Empty dan diakhiri dengan mengklik tombol Add.



Gambar 59. Membuat class controller.

Kemudian pada window Add Controller tuliskan nama class controller yang diinginkan, pada contoh di bawah ini nama controller yang dibuat adalah Home.



Gambar 60. Class controller dengan nama Home.

Dan akan didapat class controller dengan isi sebagai berikut.

```
HomeController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PengelolaanPegawai.Web.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Pada class HomeController di atas terdapat sebuah action method atau aksi yaitu Index. Pada framework ASP.NET MVC, walaupun nama file dan class controller adalah HomeController

tetapi untuk memanggil class ini cukup dengan Home. Selain itu aturan lain yang perlu diketahui adalah bahwa aksi Index merupakan aksi default yang akan dipanggil.

Dengan petunjuk tersebut maka untuk memanggil aksi Index pada class controller Home dapat dilakukan dengan dua cara yaitu :

1. `http://localhost:51165/Home/.`
2. `http://localhost:51165/Home/Index/.`

Jika pada class di atas ditambahkan aksi baru seperti pada kode berikut ini.

```
// GET: Login
public ActionResult Login()
{
    return View();
}

// GET: PageNotFound
public ActionResult PageNotFound()
{
    return View();
}
```

Maka kedua aksi ini dapat dieksekusi dengan cara seperti berikut :

1. `http://localhost:51165/Home/Login/.`
2. `http://localhost:51165/Home/PageNotFound/.`

Framework ASP.NET MVC mengatur bahwa setiap aksi dengan nama yang unik akan memiliki sebuah view. Sehingga jika melihat pada class HomeController yang memiliki 3 aksi maka paling tidak diperlukan 3 buah view yaitu view untuk aksi Index, Login dan PageNotFound. Lalu bagaimana jika pada class controller dimiliki kode lengkap seperti berikut ini, berapa view yang harus dibuat ?

```
HomeController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

using PengelolaanPegawai.Web.Models;

namespace PengelolaanPegawai.Web.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home
        public ActionResult Index()
        {
            return View();
        }

        // GET: Login
        public ActionResult Login()
        {
            return View();
        }

        // POST: Login
        [HttpPost]
        public ActionResult Login(LoginViewModel data)
```

```

    {
        return View();
    }

    // GET: PageNotFound
    public ActionResult PageNotFound()
    {
        return View();
    }
}

```

Walaupun terdapat 4 aksi pada class di atas, tetapi cukup dibuat 3 view saja. Sesuai dengan aturan di atas yaitu setiap aksi dengan nama yang unik akan memiliki sebuah view. Karena pada class controller terdapat aksi dengan nama yang sama yaitu :

```

// GET: Login
public ActionResult Login()
{
    return View();
}

// POST: Login
[HttpPost]
public ActionResult Login(LoginViewModel data)
{
    return View();
}

```

Maka kedua aksi Login di atas cukup memiliki sebuah view saja yang digunakan bersamaan, sehingga cukup dibuat 3 view saja untuk class controller tersebut.

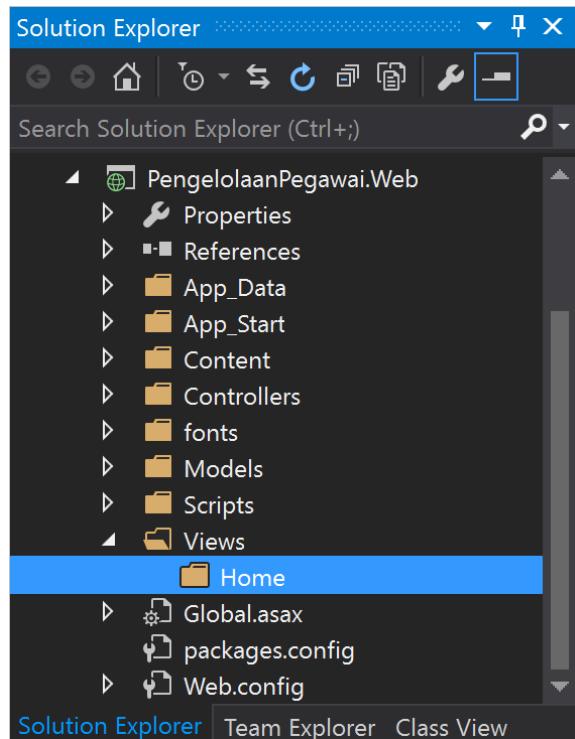
Framework ASP.NET MVC memberikan kemudahan dalam pengelolaan nama view untuk setiap aksi yaitu nama file view sesuai dengan nama aksi tersebut, tetapi tidak menutup kemungkinan untuk memilih view yang ingin ditampilkan dengan nama file yang berbeda dengan nama aksi. Format file view yang dapat digunakan ada bermacam-macam yaitu :

1. *.aspx.
2. *.ascx.
3. *.cshtml.
4. *.vbhtml.

Pada ebook ini akan digunakan file view dengan format *.cshtml, sehingga nama-nama file untuk setiap view untuk setiap aksi di atas adalah :

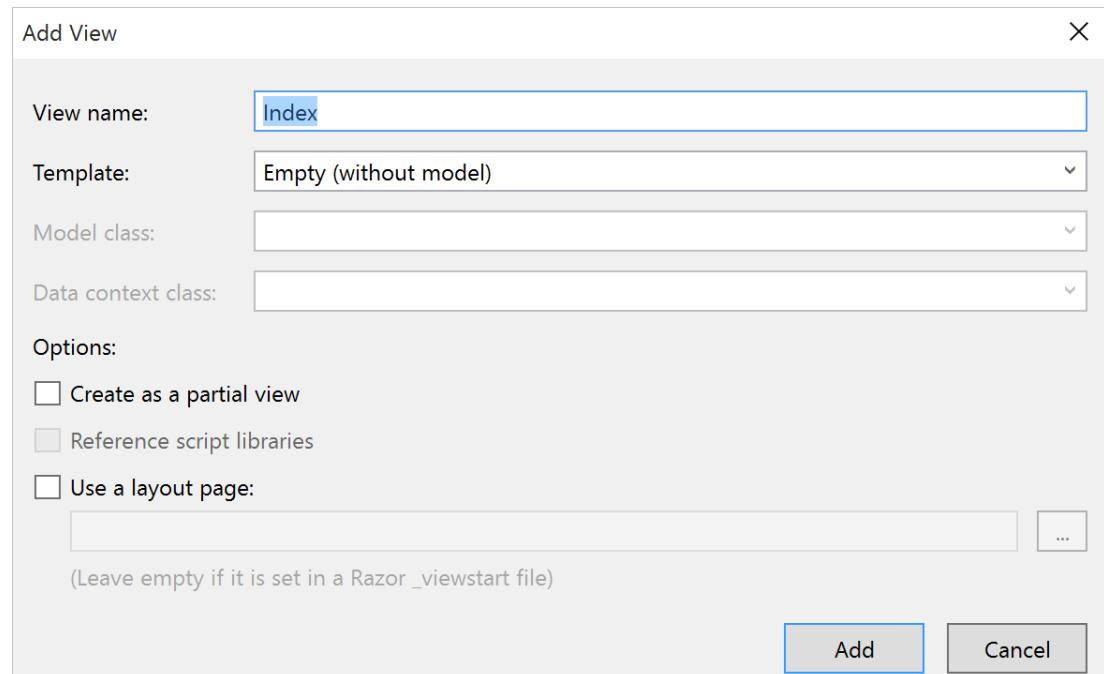
1. Index.cshtml.
2. Login.cshtml.
3. PageNotFound.cshtml.

Ketiga file ini disimpan di dalam folder Views dan dikelompokkan pada folder sesuai dengan nama controllernya, yaitu Home (seperti yang dapat dilihat pada gambar di bawah ini).



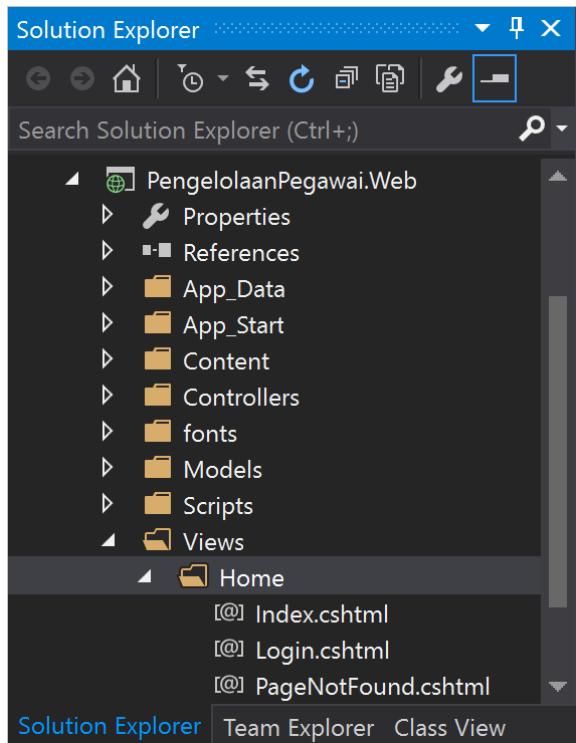
Gambar 61. Folder Home pada folder View.

Sehingga untuk membuat file view dapat dilakukan dengan cara klik kanan pada folder Views\Home kemudian pilih Add > View kemudian akan ditampilkan window berikut ini.



Gambar 62. View dengan nama file Index.cshtml.

Selanjutnya lakukan langkah yang sama untuk membuat view dengan nama file Login.cshtml dan PageNotFound.cshtml. Sehingga dapat dilihat hasil seperti berikut ini pada folder View\Home.



Gambar 63. Folder View\Home.

Berikut adalah isi file tersebut setelah dimodifikasi.

Index.cshtml

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        Index
    </div>
</body>
</html>
```

Login.cshtml

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Login</title>
</head>
```

```
<body>
    <div>
        Login
    </div>
</body>
</html>
```

```
PageNotFound.cshtml
```

```
@{
    Layout = null;
}

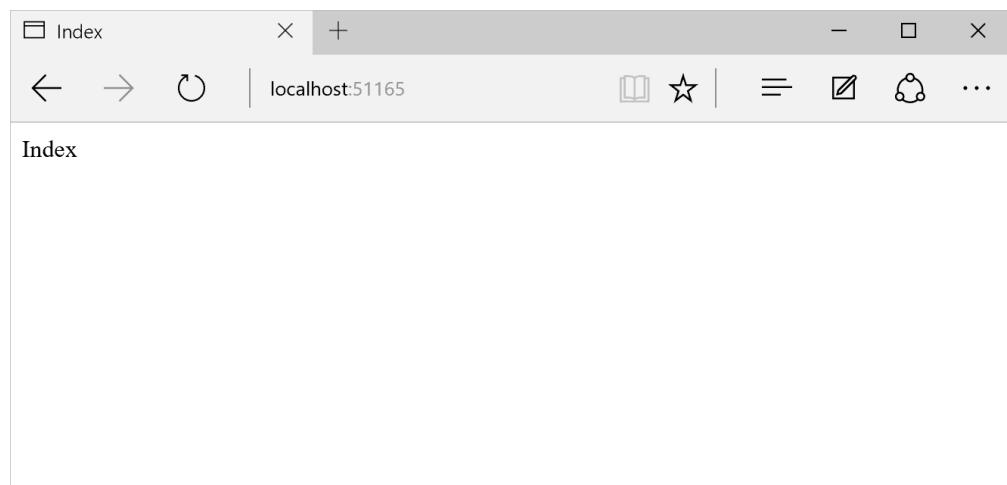
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>PageNotFound</title>
</head>
<body>
    <div>
        Halaman tidak ditemukan.
    </div>
</body>
</html>
```

Yang diubah dari ketiga file di atas adalah hanya memberikan sebaris kalimat yang akan ditampilkan pada layar web browser.

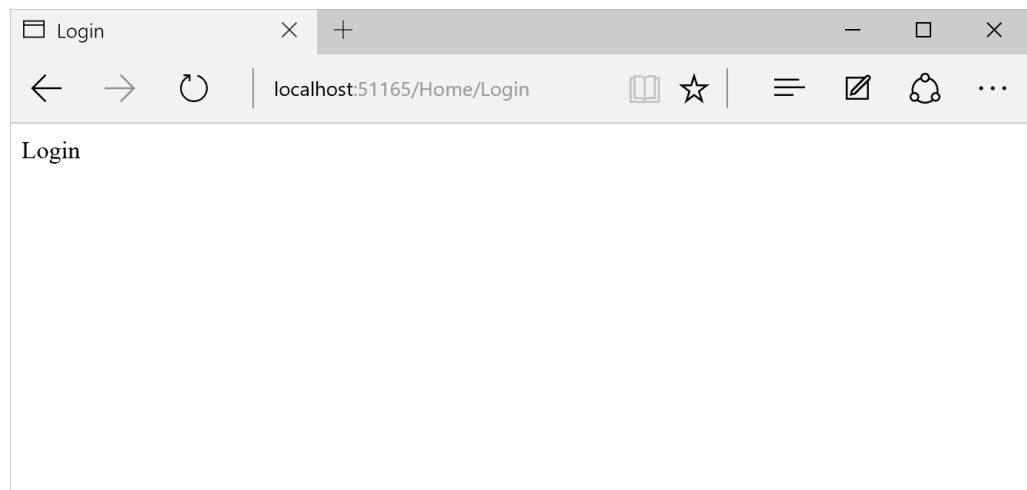
Untuk mengeksekusi project ini maka bisa dilakukan dengan cara klik kanan pada project yaitu PengelolaanPegawai.Web kemudian pilih Debug > Start new instance. Maka secara otomatis web browser default akan dijalankan dengan url default dari project ini yaitu <http://localhost:51165/> (alamat default ini dapat berbeda). URL tersebut merupakan alamat root untuk mengakses project ini. Seperti yang dapat dilihat pada file App_Start\RouteConfig.cs yang menyatakan bahwa controller default yang akan dieksekusi adalah Home dan aksi default yang akan dieksekusi adalah Index. Sehingga walaupun pada address bar tidak ditulis dengan alamat berikut <http://localhost:51165/Home/Index>.

Berikut ini adalah tampilan aksi Index tersebut.



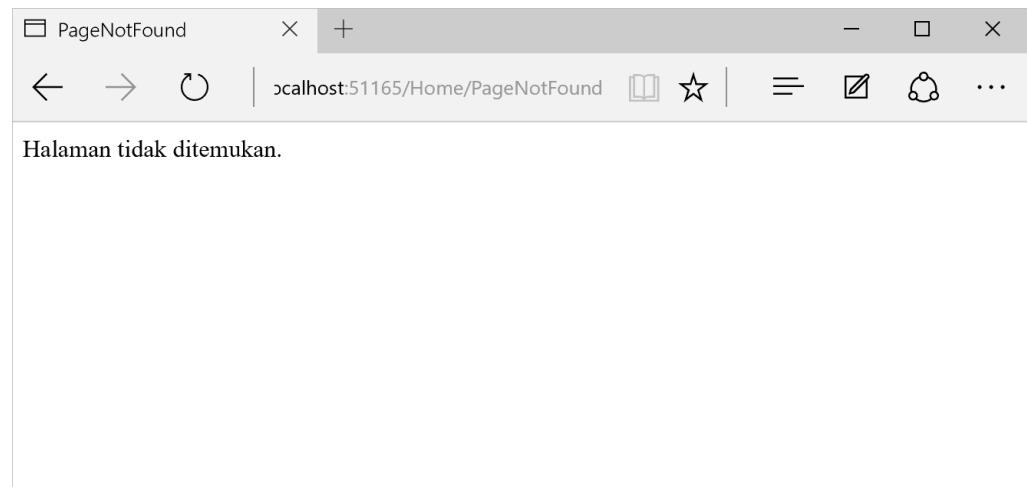
Gambar 64. Tampilan aksi Index.

Sedangkan untuk mengeksekusi aksi Login digunakan URL berikut <http://localhost:51165/Home/Login>, dengan tampilan seperti berikut.



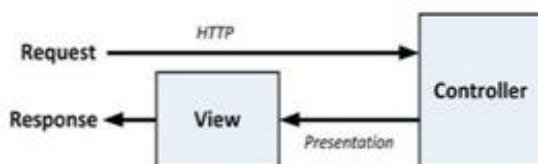
Gambar 65. Tampilan aksi Login.

Dan dengan URL <http://localhost:51165/Home/PageNotFound> akan dieksekusi aksi PageNotFound dengan tampilan seperti berikut ini.



Gambar 66. Tampilan aksi PageNotFound.

Dari paparan di atas maka sudah dapat dilihat peran dan cara kerja View-Controller yang dapat disimpulkan lebih singkat lagi dengan gambar berikut ini.



Gambar 67. Cara kerja komponen view dan controller.

Dari paparan ini maka diharapkan pembaca memiliki pengetahuan dasar untuk membuat halaman yang dapat dilihat hasilnya pada web browser, sehingga pada bagian selanjutnya sudah dapat mempelajari bagai menulis sintaks Razor pada bagian view dan melihat hasil kode yang ditulis pada web browser.

Razor

Teknologi umum yang digunakan pada pengembangan aplikasi web adalah HTML, CSS dan JavaScript. Teknologi ini tela digunakan membuat front-end yang dalam hal ini adalah komponen view. Untuk membuat komponen view pada ASP.NET MVC dapat menggunakan beberapa framework yaitu Web Forms (*.aspx, *.ascx) atau Web Pages (*.cshtml, *.vbhtml). Pada bagian ini hanya akan dijelaskan pembuatan komponen view dengan menggunakan framework Web Pages dengan Razor.

Razor merupakan bahasa yang digunakan pada Web Pages, bahasa ini dapat ditulis dengan basis bahasa pemrograman C# atau VB.NET. Pada ebook ini hanya akan dijelaskan dan dicontohkan penggunaan Rozor dengan basis bahasa pemrograman C#.

Sintaks Dasar

Pada bahasa pemrograman web biasanya digunakan penanda yang membedakan antara baris HTML biasa dan baris sintaks Razor. Baris yang memiliki penanda akan diparsing oleh runtime pada web server sesuai dengan perintah pada baris tersebut dan kemudian akan dirender sebagai kode HTML. Pada Razor, tanda yang digunakan adalah karakter @. Sintaks pada pemrograman web mempunyai beberapa jenis seperti blok kode dan ekspresi.

Blok Kode

Blok kode adalah area yang diakhiri dengan penanda dan didalamnya berisi baris-baris perintah. Pada pemrograman PHP dapat dilihat blok kode yang digunakan adalah sebagai berikut:

```
<?php . . . ?>
```

Sedangkan pada ASP.NET Web Forms digunakan kode blok sebagai berikut:

```
<% . . . %>
```

Pada Razor digunakan blok kode sebagai berikut:

```
@{ . . . }
```

Jika melihat file view Index.cshtml yang telah dibuat sebelumnya, dapat ditemui kode Razor seperti di bawah ini.

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

Kode tersebut bertujuan untuk menentukan layout view yang akan digunakan. Pada ASP.NET Web Forms, layout ini dikenal sebagai Master Page yang memungkinkan semua halaman web memiliki tampilan atau layout yang sama. Hal ini akan lebih detail dijelaskan pada bagian selanjutnya.

Contoh lain dapat dilihat dari barisan kode berikut ini.

```
@{  
    int bil1 = 1;  
    int bil2 = 3;  
    int hasil = bil1 + bil2;
```

[1]

Dari barisan kode di atas berisi beberapa perintah untuk pembuatan variable (bil1 dan bil2) dan mengisi nilai pada variable tersebut. Kemudian dapat dilihat juga baris berupa operasi penjumlahan.

Ekspresi

Ekspresi adalah baris yang berfungsi untuk menampilkan nilai dari suatu variable. Contoh ekspresinya dapat dilihat pada contoh pada paparan tentang ASP.NET MVC pada Bab 2 pada halaman Index.cshtml. Contoh-contoh ekspresi dapat dilihat kembali pada kode di bawah ini berupa baris-baris yang dicetak tebal.

```
@using (Html.BeginForm())
{
    <p>
        @Html.LabelFor(m=>m.Nama)
        @Html.TextBoxFor(m=>m.Nama)
    </p>
    <p>
        @Html.LabelFor(m=>m.Alamat)
        @Html.TextBoxFor(m=>m.Alamat)
    </p>
    <p>
        @ViewBag.Pesan
    </p>
    <p>
        <input type="submit" value="Kirim" />
    </p>
}
```

Contoh lain dapat dilihat pada baris berikut ini. Contoh ini merupakan lanjutan dari contoh pada bagian blok kode di atas. Pada contoh di bawah ini diperlihatkan bagaimana menampilkan nilai-nilai variable (bil1, bil2 dan hasil) dengan menggunakan ekspresi.

```
Jumlah @bil1 dan @bil2 adalah @hasil
```

Ekspresi juga dapat digunakan untuk mengisi nilai pada atribut HTML seperti contoh berikut ini.

```
@{
    ...
    string heading1 = "title";
}
...
<h1 class="@heading1">Halaman Index</h1>
```

Hasilnya dapat dilihat dengan melihat kode HTML pada web browser seperti kode berikut ini yaitu pada teks cetak tebal berwarna merah.

```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        <h1 class="title">Halaman Index</h1>
    </div>

```

<h2>Contoh Ekspresi</h2>

Komentar

Komentar sering digunakan untuk memberikan keterangan pada kode pemrograman atau untuk menonaktifkan baris perintah tertentu agar tidak dijalankan. Berikut ini adalah contoh yang dapat digunakan untuk membuat komentar.

```
@{  
    int bil1 = 1; /* variable (contoh komentar sebaris) */  
}
```

Untuk contoh komentar multi baris dapat ditulis seperti berikut ini.

```
<h2>Contoh Ekspresi</h2>  
/*  
Nilai-nilai pada blok kode di atas  
ditampilkan menggunakan ekspresi  
(contoh komentar multi baris)  
*@  
Jumlah @bil1 dan @bil2 adalah @hasil
```

Percabangan

Percabangan merupakan salah satu hal dasar pada pemrograman yang berfungsi untuk menjalankan perintah tertentu jika suatu kondisi terpenuhi. Berikut ini adalah contoh percabangan sederhana yang masih merupakan lanjutan dari contoh di atas.

```
/* contoh kondisi */  
<h2>Contoh Kondisi</h2>  
if (hasil < 6)  
{  
    <text>Nilai kecil</text>  
}  
else  
{  
    <text>Nilai besar</text>  
}
```

Dari contoh di atas, jika nilai variable hasil kurang dari 6 maka kondisi akan dipenuhi dan isi dari blok kondisi ini akan dijalankan. Hasilnya akan dapat dilihat ditampilkan tulisan "Nilai kecil". Sebaliknya jika kondisi di atas tidak dipenuhi maka blok else akan dijalankan, sehingga akan dapat dilihat tulisan "Nilai besar" pada halaman web.

Pengulangan

Pengulangan berfungsi untuk mengulang suatu perintah jika suatu kondisi dipenuhi dan pengulangan akan berhenti jika kondisi benar tidak dipenuhi lagi. Berikut ini adalah contoh pengulangan dengan menggunakan kata kunci while.

```
/* contoh pengulangan */  
<h2>Contoh Pengulangan</h2>  
int i = 1;  
  
while (i <= hasil)  
{  
    <text>@i</text><br />  
    i++;
```

```
}
```

Pada contoh di atas, akan dilakukan pengulangan untuk menampilkan nilai variable i selama kondisi terpenuhi yaitu dimana jika nilai variable i lebih kecil atau sama dengan nilai variable hasil.

Selain menggunakan kata kunci while, pengulangan juga dapat dilakukan dengan menggunakan kata kunci lain yaitu for dan foreach.

Akses Method Generic

Akses method dari class .NET atau class buatan sendiri dapat dilakukan pada blok kode atau ekspresi. Berikut ini adalah contoh cara akses pada blok kode.

```
@{  
    string sekarang = DateTime.Now.ToString();  
}
```

Sedangkan contoh ekspresi untuk mengakses class .NET.

```
Waktu sekarang adalah @DateTime.Now.ToString()
```

Akses File

Pada aplikasi web, sudah umum untuk menggunakan style CSS dan kode JavaScript untuk membantu membuat antarmuka. Biasanya kode style CSS dan JavaScript tersebut dapat disimpan pada sebuah file. Dan untuk mengakses file tersebut dari halaman web, biasanya digunakan kode seperti baris berikut ini.

```
<link href="css/bootstrap-responsive.min.css" rel="stylesheet">  
<script src="js/jquery-1.9.1.min.js"></script>
```

Pada atribut href atau src dapat dilihat bagaimana cara menentukan lokasi file yang akan diakses. Cara di atas tidak menujuk lokasi file secara absolut, sehingga akan bergantung dari letak halaman web tersebut. Untuk mengakses lokasi file tersebut secara absolut maka pada Razor bisa digunakan dengan cara seperti berikut ini.

```
<link href="~/css/bootstrap-responsive.min.css" rel="stylesheet">  
<script src="~/js/jquery-1.9.1.min.js"></script>
```

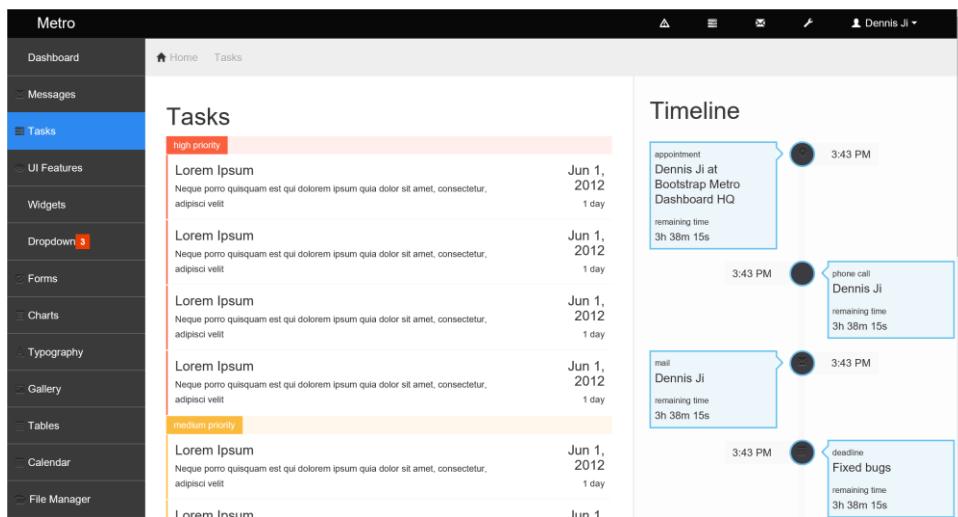
Cara ini juga bisa digunakan untuk nilai atribut yang mengakses file seperti untuk mengakses file gambar atau akses file pada link.

Layout dan Antarmuka

Aplikasi web terdiri atas beberapa halaman web yang ditulis dengan kode HTML dan CSS bahkan dibantu dengan bahasa pemrograman front-end seperti JavaScript. Antarmuka setiap halaman web tersebut biasanya seragam. Sebagai contoh dapat dilihat beberapa halaman web pada aplikasi web di bawah ini.



Gambar 68. Halaman Dashboard.



Gambar 69. Halaman Tasks.

The screenshot shows the 'Forms' section of the Metro application. The sidebar is identical. The main area has a header 'Home Forms'. A modal window titled 'Form Elements' is open, containing fields for 'Auto complete' (with placeholder 'Start typing to activate auto complete!'), 'Date input' (set to '02/16/12'), 'File input' (with 'Browse...' button and 'No file selected' message), and 'Textarea WYSIWYG'. At the bottom of the modal are 'Save changes' and 'Cancel' buttons.

Gambar 70. Halaman Forms.

Username	Date registered	Role	Status	Actions
Dennis Ji	2012/01/01	Member	Active	
Dennis Ji	2012/01/01	Member	Active	
Dennis Ji	2012/01/01	Member	Active	
Dennis Ji	2012/01/01	Member	Active	
Dennis Ji	2012/02/01	Staff	Banned	
Dennis Ji	2012/02/01	Staff	Banned	
Dennis Ji	2012/03/01	Member	Pending	
Dennis Ji	2012/03/01	Member	Pending	
Dennis Ji	2012/01/21	Staff	Active	

Gambar 71. Halaman Tables.

Dari contoh 4 halaman web di atas dapat dilihat terdapat keseragaman pada header, menu dan footer (tidak terlihat pada gambar). Yang berbeda dari keempat halaman web itu hanya pada bagian konten saja. Oleh karena itu perlu ditulis kode HTML dan CSS yang seragam untuk 4 halaman tersebut. Misal aplikasi web yang dibangun terdiri atas puluhan halaman web maka seluruh halaman tersebut harus dibuat seragam. Jika terjadi perubahan layout maka seluruh halaman web tersebut harus diubah satu per satu agar seragam.

Seperti framework pemrograman web pada umumnya telah memiliki solusi dari permasalahan di atas. Sebagai contoh pada framework ASP.NET Web Form terdapat Master Page untuk memudahkan membuat layout. Begitu juga Razor telah memiliki solusi untuk hal tersebut. Pada bagian ini akan diperlihatkan implementasi layout dan beberapa hal dasar yang dalam membangun antarmuka. Layout dan antarmuka yang dibangun akan menggunakan theme metro yang telah dibahas pada Bab 3 bagian Theme.

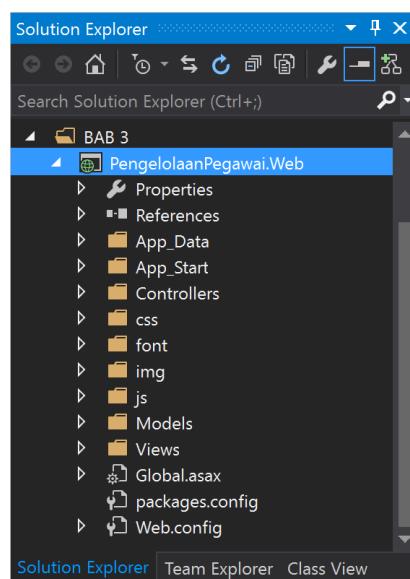
Persiapan

Langkah pertama yang harus dilakukan adalah mengunduh arsip Bootstrap Metro Dashboard pada alamat berikut ini https://github.com/jiji262/Bootstrap_Metro_Dashboard. Setelah arsip dalam bentuk file *.zip diekstrak maka dapat dilihat struktur folder dan file dari theme seperti berikut ini.

Name	Date modified	Type	Size
css	8/2/2015 1:39 PM	File folder	
font	8/2/2015 1:39 PM	File folder	
img	8/2/2015 1:39 PM	File folder	
js	8/2/2015 1:39 PM	File folder	
calendar.html	8/2/2015 1:39 PM	HTML File	20 KB
chart.html	8/2/2015 1:39 PM	HTML File	24 KB
file-manager.html	8/2/2015 1:39 PM	HTML File	20 KB
form.html	8/2/2015 1:39 PM	HTML File	32 KB
gallery.html	8/2/2015 1:39 PM	HTML File	23 KB
icon.html	8/2/2015 1:39 PM	HTML File	81 KB
index.html	8/2/2015 1:39 PM	HTML File	36 KB
login.html	8/2/2015 1:39 PM	HTML File	5 KB
messages.html	8/2/2015 1:39 PM	HTML File	29 KB
README.md	8/2/2015 1:39 PM	MD File	2 KB
submenu.html	8/2/2015 1:39 PM	HTML File	36 KB
submenu2.html	8/2/2015 1:39 PM	HTML File	36 KB
submenu3.html	8/2/2015 1:39 PM	HTML File	36 KB
table.html	8/2/2015 1:39 PM	HTML File	54 KB
tasks.html	8/2/2015 1:39 PM	HTML File	28 KB
typography.html	8/2/2015 1:39 PM	HTML File	26 KB
ui.html	8/2/2015 1:39 PM	HTML File	43 KB
widgets.html	8/2/2015 1:39 PM	HTML File	47 KB

Gambar 72. Struktur folder dan file theme Bootstrap Metro Dashboard.

Dari gambar di atas dapat dilihat terdapat 4 folder yaitu css yang berisi file-file style, font yang berisi file font, img yang berisi gambar-gambar dan js yang berisi script JavaScript. Salin keempat folder terebut beserta isinya ke dalam project PengelolaanPegawai.Web dengan cara copy keempat folder tersebut dari Windows Explorer, kemudian klik kanan pada project kemudian pilih paste. Sehingga hasilnya dapat dilihat pada area Solution Explorer seperti berikut.

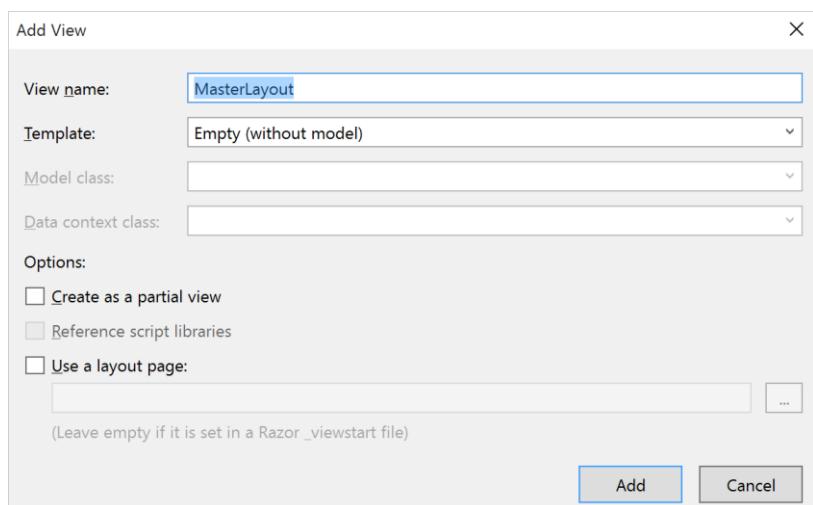


Gambar 73. Folder css, font, img dan js pada project PengelolaanPegawai.Web.

Layout

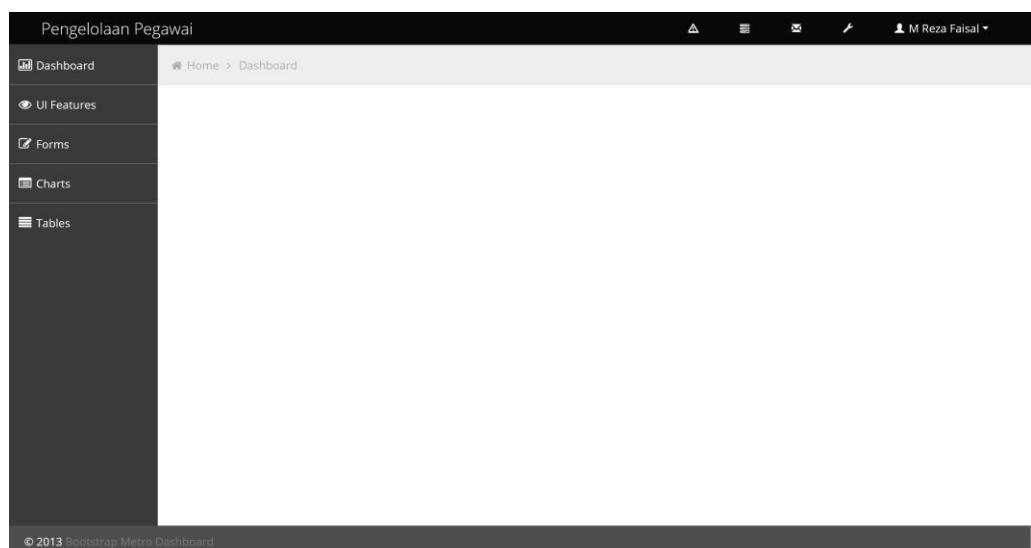
Pada bagian ini akan dijelaskan untuk membuat file layout. File layout ini akan disimpan pada folder Views. Karena ada kemungkinan file layout ini bisa lebih dari satu maka disarankan untuk membuat folder khusus untuk menyimpan file-file tersebut. Pada framework ASP.NET MVC, nama umum untuk folder yang mempunyai fungsi di atas adalah Shared.

Sehingga langkah yang dilakukan untuk membuat folder tersebut dengan cara klik kanan pada folder Views kemudian pilih New Folder dan berikan nama Shared. Kemudian untuk membuat file layout dengan cara klik kanan pada folder Views\Shared dan pilih Add > View. Pada window Add View isikan nilai MasterLayout pada kolom isian View name, kemudian klik tombol Add.



Gambar 74. Membuat file layout MasterLayout.cshtml.

Selanjutnya ubah isi file MasterLayout.cshtml dengan cara menghapus seluruh baris yang ada file tersebut, dan isinya diganti dengan isi file index.html yang dimiliki oleh theme Bootstrap Metro Dashboard. Antarmuka dari file index.html seperti pada Gambar 68 di atas akan diubah menjadi seperti berikut ini.



Gambar 75. Antarmuka MasterLayout.

Karena pada kode yang disalin dari file index.html tersebut masih menggunakan cara akses file CSS, JavaScript, gambar dan link yang belum absolut maka nilai atribut href dan src perlu diubah agar mengakses file secara absolut seperti yang telah diperlakukan caranya pada bagian Sintaks Dasar > Akses File di atas. Kemudian hilangkan bagian konten yang tidak diperlukan sehingga didapat hasil seperti pada Gambar 75.

```

MasterLayout.cshtml
<!DOCTYPE html>
<html lang="en">
<head>

    <!-- start: Meta -->
    <meta charset="utf-8">
    <title>@ViewBag.Title - Pengelolaan Pegawai</title>
    <meta name="description" content="Pengelolaan Pegawai">
    <meta name="author" content="Dennis Ji">
    <meta name="keyword" content="pengelolaan pegawai, aspnet mvc">
    <!-- end: Meta -->
    <!-- start: Mobile Specific -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- end: Mobile Specific -->
    <!-- start: CSS -->
    <link id="bootstrap-style" href("~/css/bootstrap.min.css"
rel="stylesheet">
    <link href("~/css/bootstrap-responsive.min.css" rel="stylesheet">
    <link id="base-style" href "~/css/style.css" rel="stylesheet">
    <link id="base-style-responsive" href "~/css/style-responsive.css"
rel="stylesheet">
    <link
href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,6
00italic,700italic,800italic,400,300,600,700,800&subset=latin,cyrillic-
ext,latin-ext' rel='stylesheet' type='text/css'>
    <!-- end: CSS -->
    <!-- The HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
        <script
src="~/http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
        <link id="ie-style" href "~/css/ie.css" rel="stylesheet">
    <![endif]-->
    <!--[if IE 9]>
        <link id="ie9style" href "~/css/ie9.css" rel="stylesheet">
    <![endif]-->
    <!-- start: Favicon -->
    <link rel="shortcut icon" href "~/img/favicon.ico">
    <!-- end: Favicon -->

</head>
<body>
    <!-- start: Header -->
    <div class="navbar">
        <div class="navbar-inner">
            <div class="container-fluid">
                <a class="btn btn-navbar" data-toggle="collapse" data-
target=".top-nav.nav-collapse,.sidebar-nav.nav-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </a>
                <a class="brand" href "~/Metro/Index"><span>Pengelolaan
Pegawai</span></a>

```

```

<!-- start: Header Menu -->
<div class="nav-no-collapse header-nav">
    <ul class="nav pull-right">
        <li class="dropdown hidden-phone">
            <a class="btn dropdown-toggle" data-
toggle="dropdown" href="#">
                <i class="halflings-icon white warning-
sign"></i>
            </a>
            <ul class="dropdown-menu notifications">
                <li class="dropdown-menu-title">
                    <span>You have 11 notifications</span>
                    <a href="#refresh"><i class="icon-
repeat"></i></a>
                </li>
                <li>
                    <a href="#">
                        <span class="icon blue"><i
class="icon-user"></i></span>
                        <span class="message">New user
registration</span>
                        <span class="time">1 min</span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="icon green"><i
class="icon-comment-alt"></i></span>
                        <span class="message">New
comment</span>
                        <span class="time">7 min</span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="icon green"><i
class="icon-comment-alt"></i></span>
                        <span class="message">New
comment</span>
                        <span class="time">8 min</span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="icon green"><i
class="icon-comment-alt"></i></span>
                        <span class="message">New
comment</span>
                        <span class="time">16 min</span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="icon blue"><i
class="icon-user"></i></span>
                        <span class="message">New user
registration</span>
                        <span class="time">36 min</span>
                    </a>
                </li>
                <li>

```

```

                <a href="#">
                    <span class="icon yellow"><i
class="icon-shopping-cart"></i></span>
                    <span class="message">2 items
sold</span>
                    <span class="time">1 hour</span>
                </a>
            </li>
            <li class="warning">
                <a href="#">
                    <span class="icon red"><i
class="icon-user"></i></span>
                    <span class="message">User deleted
account</span>
                    <span class="time">2 hour</span>
                </a>
            </li>
            <li class="warning">
                <a href="#">
                    <span class="icon red"><i
class="icon-shopping-cart"></i></span>
                    <span class="message">New
comment</span>
                    <span class="time">6 hour</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <span class="icon green"><i
class="icon-comment-alt"></i></span>
                    <span class="message">New
comment</span>
                    <span class="time">yesterday</span>
                </a>
            </li>
            <li>
                <a href="#">
                    <span class="icon blue"><i
class="icon-user"></i></span>
                    <span class="message">New user
registration</span>
                    <span class="time">yesterday</span>
                </a>
            </li>
            <li class="dropdown-menu-sub-footer">
                <a>View all notifications</a>
            </li>
        </ul>
    </li>
    <!-- start: Notifications Dropdown -->
    <li class="dropdown hidden-phone">
        <a class="btn dropdown-toggle" data-
toggle="dropdown" href="#">
            <i class="halflings-icon white tasks"></i>
        </a>
        <ul class="dropdown-menu tasks">
            <li class="dropdown-menu-title">
                <span>You have 17 tasks in
progress</span>
                <a href="#refresh"><i class="icon-
repeat"></i></a>

```

```

        </li>
        <li>
            <a href="#">
                <span class="header">
                    <span class="title">iOS
                </span>
                <div class="taskProgress
Development</span>
progressSlim red">80</div>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="header">
                    <span class="title">Android
Development</span>
                <span class="percent"></span>
            </span>
            <div class="taskProgress
progressSlim green">47</div>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="header">
                    <span class="title">ARM
Development</span>
                <span class="percent"></span>
            </span>
            <div class="taskProgress
progressSlim yellow">32</div>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="header">
                    <span class="title">ARM
Development</span>
                <span class="percent"></span>
            </span>
            <div class="taskProgress
progressSlim greenLight">63</div>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="header">
                    <span class="title">ARM
Development</span>
                <span class="percent"></span>
            </span>
            <div class="taskProgress
progressSlim orange">80</div>
            </a>
        </li>
        <li>
            <a class="dropdown-menu-sub-footer">View
all tasks</a>
        </li>
    </ul>

```

```

        </li>
        <!-- end: Notifications Dropdown -->
        <!-- start: Message Dropdown -->
        <li class="dropdown hidden-phone">
            <a class="btn dropdown-toggle" data-
            toggle="dropdown" href="#">
                <i class="halflings-icon white
envelope"></i>
            </a>
            <ul class="dropdown-menu messages">
                <li class="dropdown-menu-title">
                    <span>You have 9 messages</span>
                    <a href="#refresh"><i class="icon-
repeat"></i></a>
                </li>
                <li>
                    <a href="#">
                        <span class="avatar"><img
src "~/img/avatar.jpg" alt="Avatar"></span>
                        <span class="header">
                            <span class="from">
                                Dennis Ji
                            </span>
                            <span class="time">
                                6 min
                            </span>
                        </span>
                        <span class="message">
                            Lorem ipsum dolor sit amet
                            consectetur adipiscing elit, et al commore
                        </span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="avatar"><img
src "~/img/avatar.jpg" alt="Avatar"></span>
                        <span class="header">
                            <span class="from">
                                Dennis Ji
                            </span>
                            <span class="time">
                                56 min
                            </span>
                        </span>
                        <span class="message">
                            Lorem ipsum dolor sit amet
                            consectetur adipiscing elit, et al commore
                        </span>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <span class="avatar"><img
src "~/img/avatar.jpg" alt="Avatar"></span>
                        <span class="header">
                            <span class="from">
                                Dennis Ji
                            </span>
                            <span class="time">
                                3 hours
                            </span>
                        </span>
                    </a>
                </li>
            </ul>
        </li>
    </ul>

```

```

                </span>
            </span>
            <span class="message">
                Lorem ipsum dolor sit amet
            consectetur adipiscing elit, et al commore
                </span>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="avatar"><img
src "~/img/avatar.jpg" alt="Avatar"></span>
                <span class="header">
                    <span class="from">
                        Dennis Ji
                    </span>
                    <span class="time">
                        yesterday
                    </span>
                </span>
                <span class="message">
                    Lorem ipsum dolor sit amet
            consectetur adipiscing elit, et al commore
                </span>
            </a>
        </li>
        <li>
            <a href="#">
                <span class="avatar"><img
src "~/img/avatar.jpg" alt="Avatar"></span>
                <span class="header">
                    <span class="from">
                        Dennis Ji
                    </span>
                    <span class="time">
                        Jul 25, 2012
                    </span>
                </span>
                <span class="message">
                    Lorem ipsum dolor sit amet
            consectetur adipiscing elit, et al commore
                </span>
            </a>
        </li>
        <li>
            <a class="dropdown-menu-sub-footer">View
all messages</a>
                </li>
            </ul>
        </li>
        <!-- end: Message Dropdown -->
        <li>
            <a class="btn" href="#">
                <i class="halflings-icon white wrench"></i>
            </a>
        </li>
        <!-- start: User Dropdown -->
        <li class="dropdown">
            <a class="btn dropdown-toggle" data-
toggle="dropdown" href="#">

```

```

 M
Reza Faisal
    <span class="caret"></span>
</a>
<ul class="dropdown-menu">
- Account Settings
- <a href="#"><i class="halflings-icon user"></i> Profile</a></li>
    <li><a href="~/login.html"><i class="halflings-icon off"></i> Logout</a></li>
    </ul>
</li>
<!-- end: User Dropdown --&gt;
&lt;/ul&gt;
&lt;/div&gt;
<!-- end: Header Menu --&gt;

&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
<!-- start: Header --&gt;

&lt;div class="container-fluid-full"&gt;
&lt;div class="row-fluid"&gt;

<!-- start: Main Menu --&gt;
&lt;div id="sidebar-left" class="span2"&gt;
    &lt;div class="nav-collapse sidebar-nav"&gt;
        &lt;ul class="nav nav-tabs nav-stacked main-menu"&gt;
            &lt;li&gt;&lt;a href="~/Metro/Index"&gt;&lt;i class="icon-bar-chart"&gt;&lt;/i&gt;&lt;span class="hidden-tablet"&gt; Dashboard&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;
            &lt;li&gt;&lt;a href="~/ui.html"&gt;&lt;i class="icon-eye-open"&gt;&lt;/i&gt;&lt;span class="hidden-tablet"&gt; UI Features&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;
            &lt;li&gt;&lt;a href="~/form.html"&gt;&lt;i class="icon-edit"&gt;&lt;/i&gt;&lt;span class="hidden-tablet"&gt; Forms&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;
            &lt;li&gt;&lt;a href="~/chart.html"&gt;&lt;i class="icon-list-alt"&gt;&lt;/i&gt;&lt;span class="hidden-tablet"&gt; Charts&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;
            &lt;li&gt;&lt;a href="~/table.html"&gt;&lt;i class="icon-align-justify"&gt;&lt;/i&gt;&lt;span class="hidden-tablet"&gt; Tables&lt;/span&gt;&lt;/a&gt;&lt;/li&gt;
        &lt;/ul&gt;
    &lt;/div&gt;
&lt;/div&gt;
<!-- end: Main Menu --&gt;

&lt;noscript&gt;
    &lt;div class="alert alert-block span10"&gt;
        &lt;h4 class="alert-heading"&gt;Warning!&lt;/h4&gt;
        &lt;p&gt;You need to have &lt;a href="~/http://en.wikipedia.org/wiki/JavaScript" target="_blank"&gt;JavaScript&lt;/a&gt; enabled to use this site.&lt;/p&gt;
    &lt;/div&gt;
&lt;/noscript&gt;

<!-- start: Content Container --&gt;
&lt;div id="content" class="span10"&gt;

    &lt;ul class="breadcrumb"&gt;
        &lt;li&gt;</pre>

```

```

        <i class="icon-home"></i>
        <a href("~/Metro/Index">Home</a>
        <i class="icon-angle-right"></i>
    </li>
    <li><a href="#">@ViewBag.Title</a></li>
</ul>

<!-- start: Page Content -->
@RenderBody()
<!-- end: Page Content -->

</div><!--/.fluid-container-->
<!-- end: Content Container -->
</div><!--/#content.span10-->
</div><!--/fluid-row-->

<div class="modal hide fade" id="myModal">
    <div class="modal-header">
        <button type="button" class="close" data-
dismiss="modal">x</button>
        <h3>Settings</h3>
    </div>
    <div class="modal-body">
        <p>Here settings can be configured...</p>
    </div>
    <div class="modal-footer">
        <a href="#" class="btn" data-dismiss="modal">Close</a>
        <a href="#" class="btn btn-primary">Save changes</a>
    </div>
</div>

<div class="clearfix"></div>

<footer>
    <p>
        <span style="text-align:left;float:left">&copy; 2013 <a
        href="~/http://jiji262.github.io/Bootstrap_Metro_Dashboard/">
        Bootstrap Metro Dashboard</a></span>
    </p>
</footer>

<!-- start: JavaScript-->
<script src "~/js/jquery-1.9.1.min.js"></script>
<script src "~/js/jquery-migrate-1.0.0.min.js"></script>
<script src "~/js/jquery-ui-1.10.0.custom.min.js"></script>
<script src "~/js/jquery.ui.touch-punch.js"></script>
<script src "~/js/modernizr.js"></script>
<script src "~/js/bootstrap.min.js"></script>
<script src "~/js/jquery.cookie.js"></script>
<script src "~/js/fullcalendar.min.js"></script>
<script src "~/js/jquery.dataTables.min.js"></script>
<script src "~/js/excanvas.js"></script>
<script src "~/js/jquery.flot.js"></script>
<script src "~/js/jquery.flot.pie.js"></script>
<script src "~/js/jquery.flot.stack.js"></script>
<script src "~/js/jquery.flot.resize.min.js"></script>
<script src "~/js/jquery.chosen.min.js"></script>
<script src "~/js/jquery.uniform.min.js"></script>
<script src "~/js/jquery.cleditor.min.js"></script>
<script src "~/js/jquery.noty.js"></script>

```

```

<script src="~/js/jquery.elfinder.min.js"></script>
<script src="~/js/jquery.raty.min.js"></script>
<script src="~/js/jquery.iphone.toggle.js"></script>
<script src="~/js/jquery.uploadify-3.1.min.js"></script>
<script src="~/js/jquery.gritter.min.js"></script>
<script src="~/js/jquery.imagesloaded.js"></script>
<script src="~/js/jquery.masonry.min.js"></script>
<script src="~/js/jquery.knob.modified.js"></script>
<script src="~/js/jquery.sparkline.min.js"></script>
<script src="~/js/counter.js"></script>
<script src="~/js/retina.js"></script>
<script src="~/js/custom.js"></script>
<!-- end: JavaScript-->
</body>
</html>

```

Contoh-contoh baris yang diubah pada file di atas adalah sebagai berikut di bawah ini. Untuk akses file CSS di bawah ini:

```
<link href="css/bootstrap-responsive.min.css" rel="stylesheet">
```

Menjadi seperti berikut:

```
<link href="~/css/bootstrap-responsive.min.css" rel="stylesheet">
```

Untuk akses file JavaScript di bawah ini:

```
<script src="js/jquery-1.9.1.min.js"></script>
```

Menjadi seperti berikut:

```
<script src="~/js/jquery-1.9.1.min.js"></script>
```

Untuk akses file halaman web di bawah ini:

```

<li><a href="index.html"><i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span></a></li>
<li><a href="messages.html"><i class="icon-envelope"></i><span class="hidden-tablet"> Messages</span></a></li>

```

Menjadi berikut ini:

```

<li><a href="~/Metro/Index"><i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span></a></li>
<li><a href="~/messages.html"><i class="icon-envelope"></i><span class="hidden-tablet"> Messages</span></a></li>

```

Untuk akses file gambar di bawah ini:

```

```

Menjadi berikut ini:

```

```

Selain mengubah cara akses lokasi file menjadi absolut, juga perlu ditambahkan baris berikut ini pada kode di atas. Lokasi baris kode ini bisa dilihat pada cetak tebal pada kode di atas.

```

<!-- start: Page Content -->
@RenderBody()
<!-- end: Page Content -->

```

Baris ini merupakan perintah yang berfungsi untuk merender isi dari view yang menggunakan layout ini, sehingga file view dan layout ini akan digabung menjadi satu dan antarmukanya dapat dilihat secara utuh pada web browser.

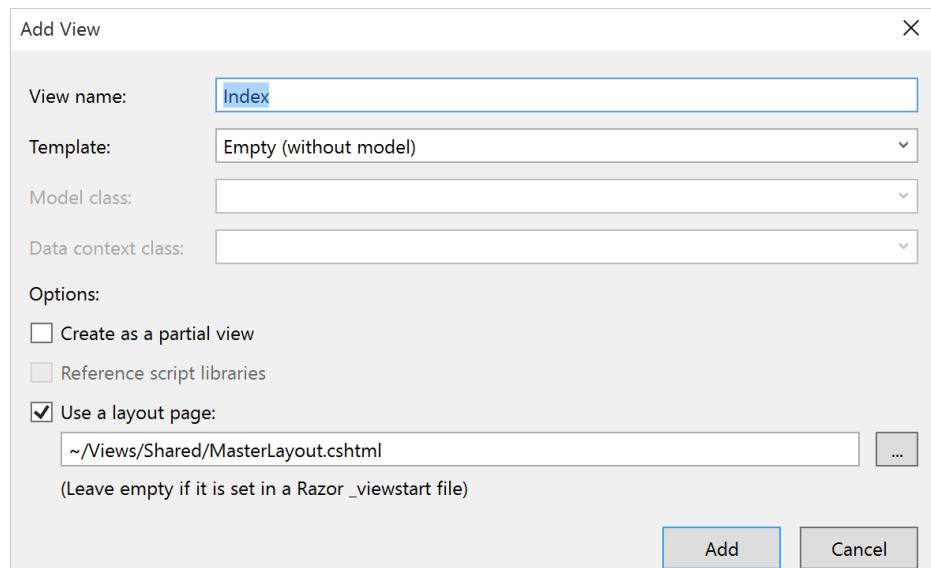
Membuat Controller dan View

Contoh penggunaan layout di atas akan dicontohkan pada bagian ini. Pertama perlu dibuat komponen controller dan view. Sebagai contoh akan dibuat controller dengan nama MetroController dengan isi seperti berikut.

```
MetroController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PengelolaanPegawai.Web.Controllers
{
    public class MetroController : Controller
    {
        // GET: Metro
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Selanjutnya membuat view pada folder Views\Metro. Yang berbeda dari cara membuat view dari tahap sebelumnya adalah pada cara ini dilakukan centang checkbox Use a layout page dan mengisi nilai ~/Views/Shared/MasterLayout.cshtml kemudian klik tombol Add.



Gambar 76.

Berikut ini adalah isi dari view Index.cshtml.

```
Index.cshtml
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/MasterLayout.cshtml";
}

<h2>Index</h2>
```

Berbeda dibandingkan isi file view yang telah dibuat sebelumnya, pada file di atas tidak diperlukan tag HTML lengkap, karena kode HTML lengkapnya telah diwakili pada file MasterLayout.cshtml. Selain itu dapat dilihat pula nilai variable Layout juga memiliki nilai, tidak seperti isi file view sebelumnya dimana variabel Layout bernilai null.

Isi file Index.cshtml di atas dapat diubah dengan memanfaatkan konten dari file index.html berikut ini.

```
Index.cshtml

@{
    ViewBag.Title = "Dashboard";
    Layout = "~/Views/Shared/MasterLayout.cshtml";
}

<div class="row-fluid">

    <div class="span3 statbox purple" onTablet="span6" onDesktop="span3">
        <div class="boxchart">5,6,7,2,0,4,2,4,8,2,3,3,2</div>
        <div class="number">854<i class="icon-arrow-up"></i></div>
        <div class="title">visits</div>
        <div class="footer">
            <a href="#"> read full report</a>
        </div>
    </div>
    <div class="span3 statbox green" onTablet="span6" onDesktop="span3">
        <div class="boxchart">1,2,6,4,0,8,2,4,5,3,1,7,5</div>
        <div class="number">123<i class="icon-arrow-up"></i></div>
        <div class="title">sales</div>
        <div class="footer">
            <a href="#"> read full report</a>
        </div>
    </div>
    <div class="span3 statbox blue noMargin" onTablet="span6" onDesktop="span3">
        <div class="boxchart">5,6,7,2,0,-4,-2,4,8,2,3,3,2</div>
        <div class="number">982<i class="icon-arrow-up"></i></div>
        <div class="title">orders</div>
        <div class="footer">
            <a href="#"> read full report</a>
        </div>
    </div>
    <div class="span3 statbox yellow" onTablet="span6" onDesktop="span3">
        <div class="boxchart">7,2,2,2,1,-4,-2,4,8,,0,3,3,5</div>
        <div class="number">678<i class="icon-arrow-down"></i></div>
        <div class="title">visits</div>
        <div class="footer">
            <a href="#"> read full report</a>
        </div>
    </div>
</div>
<div class="row-fluid">

    <div class="span8 widget blue" onTablet="span7" onDesktop="span8">
        <div id="stats-chart2" style="height:282px;"></div>
    </div>

    <div class="sparkLineStats span4 widget green" onTablet="span5" onDesktop="span4">
        <ul class="unstyled">
```

```

<li>
    <span class="sparkLineStats3"></span>
    Pageviews:
    <span class="number">781</span>
</li>
<li>
    <span class="sparkLineStats4"></span>
    Pages / Visit:
    <span class="number">2,19</span>
</li>
<li>
    <span class="sparkLineStats5"></span>
    Avg. Visit Duration:
    <span class="number">00:02:58</span>
</li>
<li>
    <span class="sparkLineStats6"></span>
    Bounce Rate: <span class="number">59,83%</span>
</li>
<li>
    <span class="sparkLineStats7"></span>
    % New Visits:
    <span class="number">70,79%</span>
</li>
<li>
    <span class="sparkLineStats8"></span>
    % Returning Visitor:
    <span class="number">29,21%</span>
</li>
</ul>

    <div class="clearfix"></div>
</div><!-- End .sparkStats -->
</div>

<div class="row-fluid hideInIE8 circleStats">

    <div class="span2" onTablet="span4" onDesktop="span2">
        <div class="circleStatsItemBox yellow">
            <div class="header">Disk Space Usage</div>
            <span class="percent">percent</span>
            <div class="circleStat">
                <input type="text" value="58" class="whiteCircle" />
            </div>
            <div class="footer">
                <span class="count">
                    <span class="number">20000</span>
                    <span class="unit">MB</span>
                </span>
                <span class="sep"> / </span>
                <span class="value">
                    <span class="number">50000</span>
                    <span class="unit">MB</span>
                </span>
            </div>
        </div>
    </div>
    <div class="span2" onTablet="span4" onDesktop="span2">
        <div class="circleStatsItemBox green">
            <div class="header">Bandwidth</div>
            <span class="percent">percent</span>

```

```

        <div class="circleStat">
            <input type="text" value="78" class="whiteCircle" />
        </div>
        <div class="footer">
            <span class="count">
                <span class="number">5000</span>
                <span class="unit">GB</span>
            </span>
            <span class="sep"> / </span>
            <span class="value">
                <span class="number">5000</span>
                <span class="unit">GB</span>
            </span>
        </div>
    </div>
    <div class="span2" onTablet="span4" onDesktop="span2">
        <div class="circleStatsItemBox greenDark">
            <div class="header">Memory</div>
            <span class="percent">percent</span>
            <div class="circleStat">
                <input type="text" value="100" class="whiteCircle" />
            </div>
            <div class="footer">
                <span class="count">
                    <span class="number">64</span>
                    <span class="unit">GB</span>
                </span>
                <span class="sep"> / </span>
                <span class="value">
                    <span class="number">64</span>
                    <span class="unit">GB</span>
                </span>
            </div>
        </div>
    </div>
    <div class="span2 noMargin" onTablet="span4" onDesktop="span2">
        <div class="circleStatsItemBox pink">
            <div class="header">CPU</div>
            <span class="percent">percent</span>
            <div class="circleStat">
                <input type="text" value="83" class="whiteCircle" />
            </div>
            <div class="footer">
                <span class="count">
                    <span class="number">64</span>
                    <span class="unit">GHz</span>
                </span>
                <span class="sep"> / </span>
                <span class="value">
                    <span class="number">3.2</span>
                    <span class="unit">GHz</span>
                </span>
            </div>
        </div>
    </div>
    <div class="span2" onTablet="span4" onDesktop="span2">
        <div class="circleStatsItemBox orange">
            <div class="header">Memory</div>
            <span class="percent">percent</span>
            <div class="circleStat">

```

```

        <input type="text" value="100" class="whiteCircle" />
    </div>
    <div class="footer">
        <span class="count">
            <span class="number">64</span>
            <span class="unit">GB</span>
        </span>
        <span class="sep"> / </span>
        <span class="value">
            <span class="number">64</span>
            <span class="unit">GB</span>
        </span>
    </div>
</div>
<div class="span2" onTablet="span4" onDesktop="span2">
    <div class="circleStatsItemBox greenLight">
        <div class="header">Memory</div>
        <span class="percent">percent</span>
        <div class="circleStat">
            <input type="text" value="100" class="whiteCircle" />
        </div>
        <div class="footer">
            <span class="count">
                <span class="number">64</span>
                <span class="unit">GB</span>
            </span>
            <span class="sep"> / </span>
            <span class="value">
                <span class="number">64</span>
                <span class="unit">GB</span>
            </span>
        </div>
    </div>
</div>
</div>

<div class="row-fluid">
    <div class="widget blue span5" onTablet="span6" onDesktop="span5">
        <h2><span class="glyphicon globe"><i></i></span> Demographics</h2>
        <hr>
        <div class="content">
            <div class="verticalChart">
                <div class="singleBar">
                    <div class="bar">
                        <div class="value">
                            <span>37%</span>
                        </div>
                    </div>
                    <div class="title">US</div>
                </div>
                <div class="singleBar">
                    <div class="bar">
                        <div class="value">
                            <span>16%</span>
                        </div>
                    </div>
                    <div class="title">PL</div>
                </div>
            </div>
        </div>
    </div>
</div>

```

```
<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>12%</span>
    </div>
    <div class="title">GB</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>9%</span>
    </div>
    <div class="title">DE</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>7%</span>
    </div>
    <div class="title">NL</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>6%</span>
    </div>
    <div class="title">CA</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>5%</span>
    </div>
    <div class="title">FI</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>4%</span>
    </div>
    <div class="title">RU</div>
  </div>

<div class="singleBar">
  <div class="bar">
    <div class="value">
      <span>3%</span>
    </div>
    <div class="title">AU</div>
  </div>
```

```

        </div>

        <div class="singleBar">
            <div class="bar">
                <div class="value">
                    <span>1%</span>
                </div>
            </div>
            <div class="title">N/A</div>
        </div>

        <div class="clearfix"></div>
    </div>
</div><!--span-->

<div class="widget span3 red" onTablet="span6" onDesktop="span3">
    <h2><span class="glyphicon pie_chart"><i></i></span> Browsers</h2>
    <hr>
    <div class="content">
        <div class="browserStat big">
            <img src "~/img/browser-chrome-big.png" alt="Chrome">
            <span>34%</span>
        </div>
        <div class="browserStat big">
            <img src "~/img/browser-firefox-big.png" alt="Firefox">
            <span>34%</span>
        </div>
        <div class="browserStat">
            <img src "~/img/browser-ie.png" alt="Internet Explorer">
            <span>34%</span>
        </div>
        <div class="browserStat">
            <img src "~/img/browser-safari.png" alt="Safari">
            <span>34%</span>
        </div>
        <div class="browserStat">
            <img src "~/img/browser-opera.png" alt="Opera">
            <span>34%</span>
        </div>
    </div>
</div>

<div class="widget yellow span4 noMargin" onTablet="span12" onDesktop="span4">
    <h2><span class="glyphicon fire"><i></i></span> Server Load</h2>
    <hr>
    <div class="content">
        <div id="serverLoad2" style="height:224px;"></div>
    </div>
</div>

<div class="row-fluid">

    <div class="box black span4" onTablet="span6" onDesktop="span4">
        <div class="box-header">
            <h2><i class="halflings-icon white list"></i><span class="break"></span>Weekly Stat</h2>
            <div class="box-icon">

```

```

        <a href="#" class="btn-minimize"><i class="halflings-icon white chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon white remove"></i></a>
    </div>
<div class="box-content">
    <ul class="dashboard-list metro">
        <li>
            <a href="#">
                <i class="icon-arrow-up green"></i>
                <strong>92</strong>
                New Comments
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-arrow-down red"></i>
                <strong>15</strong>
                New Registrations
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-minus blue"></i>
                <strong>36</strong>
                New Articles
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-comment yellow"></i>
                <strong>45</strong>
                User reviews
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-arrow-up green"></i>
                <strong>112</strong>
                New Comments
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-arrow-down red"></i>
                <strong>31</strong>
                New Registrations
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-minus blue"></i>
                <strong>93</strong>
                New Articles
            </a>
        </li>
        <li>
            <a href="#">
                <i class="icon-comment yellow"></i>
                <strong>256</strong>

```

```

        User reviews
            </a>
        </li>
    </ul>
</div><!--/span-->

<div class="box black span4" onTablet="span6" onDesktop="span4">
    <div class="box-header">
        <h2><i class="halflings-icon white user"></i><span
class="break"></span>Last Users</h2>
        <div class="box-icon">
            <a href="#" class="btn-minimize"><i class="halflings-icon
white chevron-up"></i></a>
            <a href="#" class="btn-close"><i class="halflings-icon white
remove"></i></a>
        </div>
    <div class="box-content">
        <ul class="dashboard-list metro">
            <li class="green">
                <a href="#">
                    <img class="avatar" alt="Dennis Ji"
src "~/img/avatar.jpg">
                </a>
                <strong>Name:</strong> Dennis Ji<br>
                <strong>Since:</strong> Jul 25, 2012 11:09<br>
                <strong>Status:</strong> Approved
            </li>
            <li class="yellow">
                <a href="#">
                    <img class="avatar" alt="Dennis Ji"
src "~/img/avatar.jpg">
                </a>
                <strong>Name:</strong> Dennis Ji<br>
                <strong>Since:</strong> Jul 25, 2012 11:09<br>
                <strong>Status:</strong> Pending
            </li>
            <li class="red">
                <a href="#">
                    <img class="avatar" alt="Dennis Ji"
src "~/img/avatar.jpg">
                </a>
                <strong>Name:</strong> Dennis Ji<br>
                <strong>Since:</strong> Jul 25, 2012 11:09<br>
                <strong>Status:</strong> Banned
            </li>
            <li class="blue">
                <a href="#">
                    <img class="avatar" alt="Dennis Ji"
src "~/img/avatar.jpg">
                </a>
                <strong>Name:</strong> Dennis Ji<br>
                <strong>Since:</strong> Jul 25, 2012 11:09<br>
                <strong>Status:</strong> Updated
            </li>
        </ul>
    </div><!--/span-->

```

```

<div class="box black span4 noMargin" onTablet="span12"
onDesktop="span4">
    <div class="box-header">
        <h2><i class="halflings-icon white check"></i><span
class="break"></span>To Do List</h2>
        <div class="box-icon">
            <a href="#" class="btn-setting"><i class="halflings-icon
white wrench"></i></a>
            <a href="#" class="btn-minimize"><i class="halflings-icon
white chevron-up"></i></a>
            <a href="#" class="btn-close"><i class="halflings-icon white
remove"></i></a>
        </div>
    </div>
    <div class="box-content">
        <div class="todo metro">
            <ul class="todo-list">
                <li class="red">
                    <a class="action icon-check-empty" href="#"></a>
                    Windows Phone 8 App
                    <strong>today</strong>
                </li>
                <li class="red">
                    <a class="action icon-check-empty" href="#"></a>
                    New frontend layout
                    <strong>today</strong>
                </li>
                <li class="yellow">
                    <a class="action icon-check-empty" href="#"></a>
                    Hire developers
                    <strong>tommorow</strong>
                </li>
                <li class="yellow">
                    <a class="action icon-check-empty" href="#"></a>
                    Windows Phone 8 App
                    <strong>tommorow</strong>
                </li>
                <li class="green">
                    <a class="action icon-check-empty" href="#"></a>
                    New frontend layout
                    <strong>this week</strong>
                </li>
                <li class="green">
                    <a class="action icon-check-empty" href="#"></a>
                    Hire developers
                    <strong>this week</strong>
                </li>
                <li class="blue">
                    <a class="action icon-check-empty" href="#"></a>
                    New frontend layout
                    <strong>this month</strong>
                </li>
                <li class="blue">
                    <a class="action icon-check-empty" href="#"></a>
                    Hire developers
                    <strong>this month</strong>
                </li>
            </ul>
        </div>
    </div>
</div>

```

```

</div>

<div class="row-fluid">
    <a class="quick-button metro yellow span2">
        <i class="icon-group"></i>
        <p>Users</p>
        <span class="badge">237</span>
    </a>
    <a class="quick-button metro red span2">
        <i class="icon-comments-alt"></i>
        <p>Comments</p>
        <span class="badge">46</span>
    </a>
    <a class="quick-button metro blue span2">
        <i class="icon-shopping-cart"></i>
        <p>Orders</p>
        <span class="badge">13</span>
    </a>
    <a class="quick-button metro green span2">
        <i class="icon-barcode"></i>
        <p>Products</p>
    </a>
    <a class="quick-button metro pink span2">
        <i class="icon-envelope"></i>
        <p>Messages</p>
        <span class="badge">88</span>
    </a>
    <a class="quick-button metro black span2">
        <i class="icon-calendar"></i>
        <p>Calendar</p>
    </a>

    <div class="clearfix"></div>
</div><!--row-->

```

Pengujian

Pada bagian ini akan dilakukan pengujian dari proses pembuatan layout yang telah dilakukan di atas dengan bantuan controller dan view yang telah dibuat. Pengujian yang dilakukan untuk melihat beberapa hal berikut ini :

1. Melihat apakah layout dan view telah terintegrasi dengan benar.
2. Keberhasilan integrasi file CSS dan JavaScript dengan melihat apakah antarmuka telah sesuai dengan antarmuka theme sebelum dikonversi menjadi halaman ASP.NET MVC.

Pengujian dilakukan dengan cara menjalankan atau mengeksekusi project sehingga ditampilkan pada web browser. Untuk mempermudah akses controller Metro yang telah dibuat di atas secara default, maka dapat dilakukan modifikasi pada file App_Start\RouteConfig.cs, dengan mengubah nilai default controller menjadi Metro.

RouteConfig.cs
<pre> using System; using System.Collections.Generic; using System.Linq; using System.Web; using System.Web.Mvc; using System.Web.Routing; </pre>

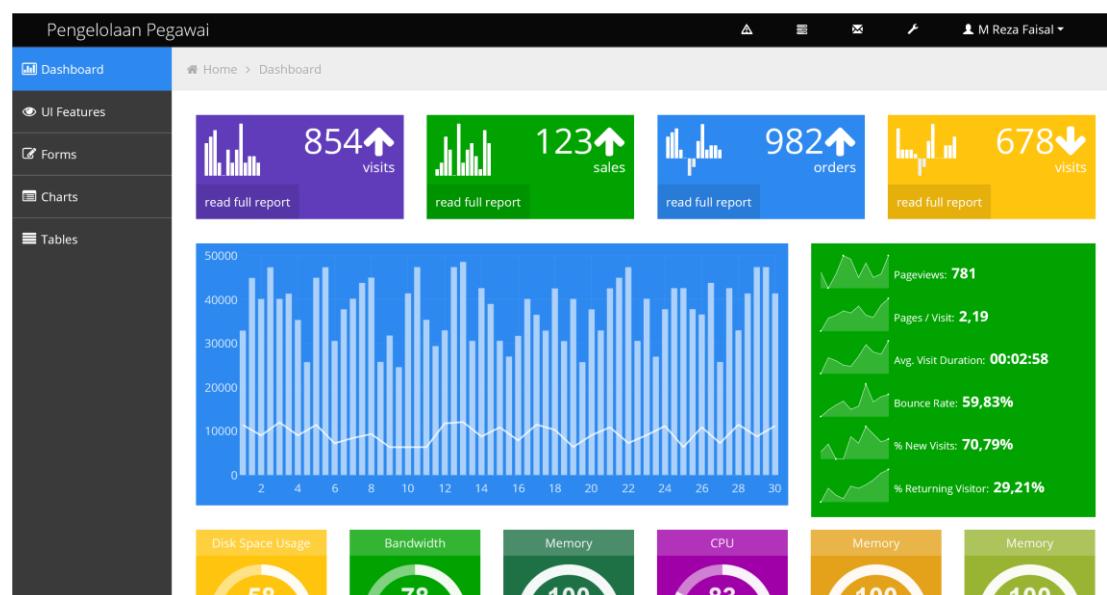
```

namespace PengelolaanPegawai.Web
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Metro", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}

```

Dengan cara di atas maka ketika project dieksekusi secara otomatis akan diakses aksi Index pada controller Metro. Hasilnya dapat dilihat pada gambar di bawah ini.



Gambar 77. Halaman ASP.NET dengan Theme Bootstrap Metro Dashboard.

Jika hasil pembuatan layout telah berhasil maka dapat dilanjutnya untuk membuat menambahkan beberapa aksi pada controller untuk menampilkan view sesuai dengan menu yang ditampilkan yaitu UI Features, Forms, Charts dan Tables. Untuk merealisasikan hal tersebut maka class MetroController.cs akan diubah menjadi berikut ini.

```

MetroController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PengelolaanPegawai.Web.Controllers
{
    public class MetroController : Controller
    {
        // GET: Metro
        public ActionResult Index()

```

```

    {
        return View();
    }

    public ActionResult UI()
    {
        return View();
    }

    public ActionResult Forms()
    {
        return View();
    }

    public ActionResult Charts()
    {
        return View();
    }

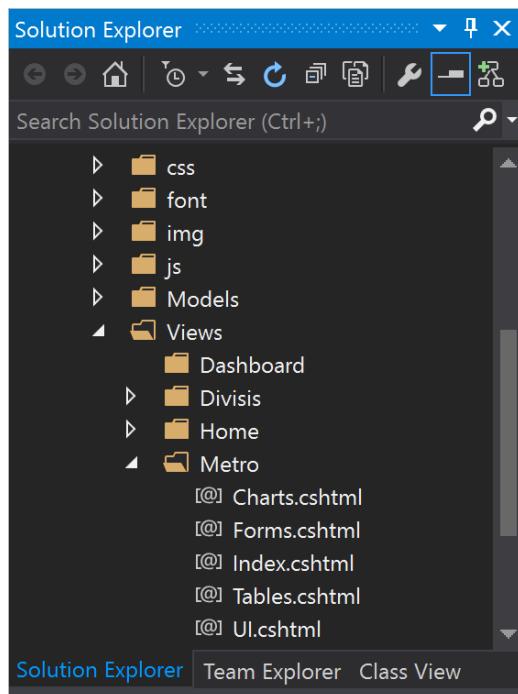
    public ActionResult Tables()
    {
        return View();
    }
}

```

Kemudian disiapkan view untuk masing-masing aksi dengan nama file yaitu :

1. UI.cshtml.
2. Forms.cshtml.
3. Charts.cshtml.
4. Tables.cshtml.

Keempat file tersebut disimpan pada folder Views\Metro.



Gambar 78. File-file view untuk controller Metro.

Agar seluruh view tersebut dapat diakses dari menu yang ada maka perlu modifikasi pada file MasterLayout.cshtml. Berikut adalah kode

```
<!-- start: Main Menu -->
<div id="sidebar-left" class="span2">
    <div class="nav-collapse sidebar-nav">
        <ul class="nav nav-tabs nav-stacked main-menu">
            <li><a href("~/Metro/Index")><i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span></a></li>
            <li><a href "~/ui.html"><i class="icon-eye-open"></i><span class="hidden-tablet"> UI Features</span></a></li>
            <li><a href "~/form.html"><i class="icon-edit"></i><span class="hidden-tablet"> Forms</span></a></li>
            <li><a href "~/chart.html"><i class="icon-list-alt"></i><span class="hidden-tablet"> Charts</span></a></li>
            <li><a href "~/table.html"><i class="icon-align-justify"></i><span class="hidden-tablet"> Tables</span></a></li>
        </ul>
    </div>
</div>

<!-- end: Main Menu -->
```

Menjadi seperti berikut ini.

```
<!-- start: Main Menu -->
<div id="sidebar-left" class="span2">
    <div class="nav-collapse sidebar-nav">
        <ul class="nav nav-tabs nav-stacked main-menu">
            <li><a href "~/Metro/UI"><i class="icon-eye-open"></i><span class="hidden-tablet"> UI Features</span></a></li>
            <li><a href "~/Metro/Forms"><i class="icon-edit"></i><span class="hidden-tablet"> Forms</span></a></li>
            <li><a href "~/Metro/Charts"><i class="icon-list-alt"></i><span class="hidden-tablet"> Charts</span></a></li>
            <li><a href "~/Metro/Tables"><i class="icon-align-justify"></i><span class="hidden-tablet"> Tables</span></a></li>
        </ul>
    </div>
</div>

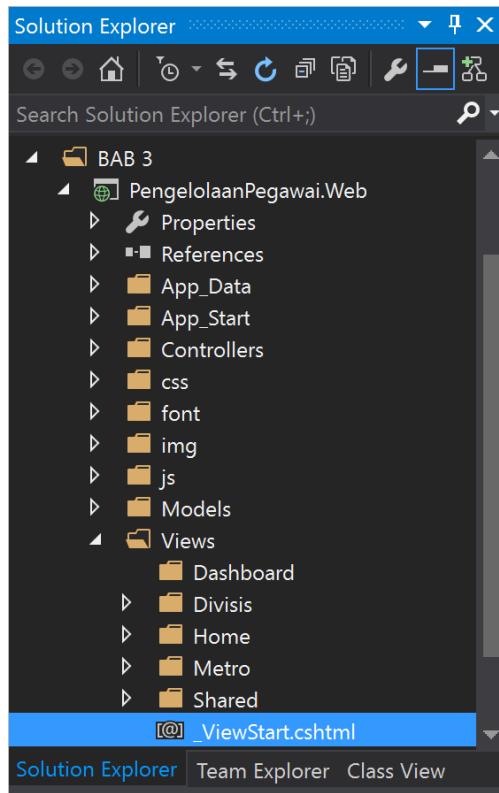
<!-- end: Main Menu -->
```

File _ViewStart.cshtml

Saat membuat file view yang menggunakan layout, maka perlu dicentang pada checkbox Use a layout page dan kemudian menentukan file layout yang akan digunakan. Sehingga pada view bisa dilihat baris berikut ini, dimana nilai Layout diisi dengan file layout.

```
@{
    ViewBag.Title = "Charts";
    Layout = "~/Views/Shared/MasterLayout.cshtml";
}
```

Dengan cara di atas mengharuskan baris tersebut ditulis pada seluruh file view yang menggunakan layout tersebut. Penentuan layout yang digunakan oleh setiap file view dapat dilakukan secara terpusat dengan cara menggunakan file _ViewStart.cshtml di dalam folder Views seperti yang terlihat pada gambar di bawah ini.

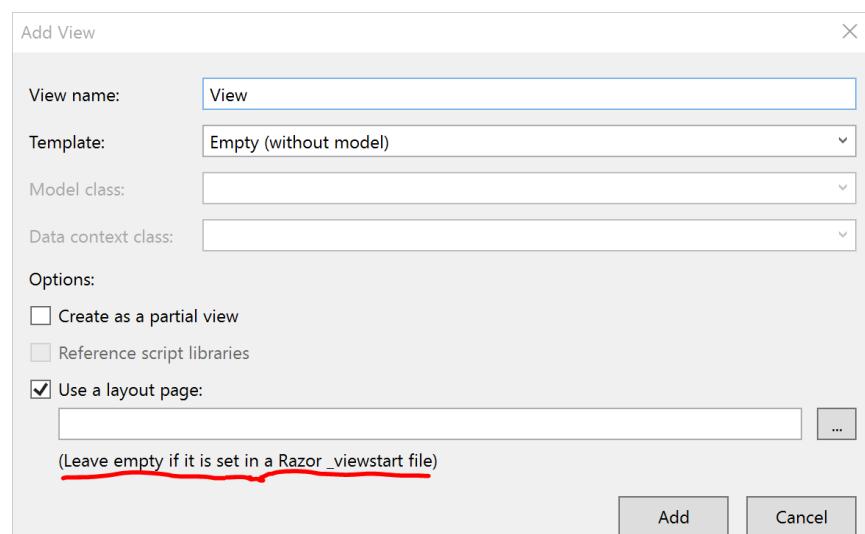


Gambar 79. File _ViewStart.cshtml.

Dan berikut ini adalah isi file tersebut:

```
@{
    Layout = "~/Views/Shared/MasterLayout.cshtml";
}
```

Dengan adanya file tersebut maka secara otomatis setiap file view yang tidak ada penentuan layout akan menggunakan file layout yang telah ditentukan pada file di atas. Sedangkan untuk membuat file view baru maka cukup melakukan pemberian nilai pada View name dan mencentang checkbox Use a layout page tanpa memberikan nilai layout yang digunakan seperti pada window di bawah ini.



Gambar 80. Window Add View.

HTML Helper

Pada bagian ini akan dibahas tentang HTML Helper yang salah satunya berfungsi untuk mempermudah membuat form HTML. HTML Helper merupakan method dengan parameter-parameter dan nilai-nilai parameter tertentu yang dapat digunakan untuk merender tag HTML dengan atribut sesuai dengan parameter dan nilai yang diberikan pada method tersebut. Sebagai contoh bisa dilihat pada HTML Helper yang ditulis pada halaman view seperti berikut:

```
@Html.Password("UserPassword")
```

Dan setelah halaman dijalankan akan dapat dilihat hasil render kode HTML seperti berikut:

```
<input id="UserPassword" name="UserPassword" type="password" value="" />
```

Contoh lain adalah, jika ditulis fungsi HTML Helper seperti berikut:

```
@Html.Label("GenreId")
```

Yang akan dirender menjadi kode HTML seperti berikut:

```
<label for="GenreId">Genre</label>
```

HTML Helper terbagi atas dua kelompok yaitu HTML Helper yang dibentuk oleh method bertipe void dan kelompok yang mengembalikan (return) nilai. Contoh di atas adalah kelompok method yang bukan bertipe void. Sedangkan method yang bertipe void tidak dapat ditulis dengan cara di atas, tetapi harus ditulis dengan cara di bawah ini:

```
@{  
    Html.RenderAction("Forms");  
}
```

Berikut ini adalah daftar HTML Helper yang terdapat pada framework ASP.NET MVC, yaitu:

No	Sintaks	Tipe
1	@Html.Action	
2	@Html.ActionLink	
3	@Html.AntiForgeryToken	
4	@Html.AttributeEncode	
5	@Html.BeginForm	
6	@Html.BeginRouteForm	
7	@Html.CheckBox	
8	@Html.CheckBoxFor<>	
9	@Html.Display	
10	@Html.DisplayFor<>	
11	@Html.DisplayForModel	
12	@Html.DisplayName	
13	@Html.DisplayNameFor<>	
14	@Html.DisplayNameForModel	
15	@Html.DisplayText	
16	@Html.DisplayTextFor<>	

No	Sintaks	Type
17	<code>@Html.DropDownList</code>	
18	<code>@Html.DropDownListFor<></code>	
19	<code>@Html.Editor</code>	
20	<code>@Html.EditorFor<></code>	
21	<code>@Html.EditorForModel</code>	
22	<code>@Html.EnableClientValidation</code>	void
23	<code>@Html.EnableUnobtrusiveJavaScript</code>	void
24	<code>@Html.Encode</code>	
25	<code>@Html.EndForm</code>	void
26	<code>@Html.EnumDropDownListFor<></code>	
27	<code>@Html.Equals</code>	
28	<code>@Html.FormatValue</code>	
29	<code>@Html.GetHashCode</code>	
30	<code>@Html.GetType</code>	
31	<code>@Html.GetUnobtrusiveValidationAttributes</code>	
32	<code>@Html.Hidden</code>	
33	<code>@Html.HiddenFor<></code>	
34	<code>@Html.Html5DateRenderingMode</code>	
35	<code>@Html.HttpMethodOverride</code>	
36	<code>@Html.Id</code>	
37	<code>@Html.IdFor<></code>	
38	<code>@Html.IdForModel</code>	
39	<code>@Html.Label</code>	
40	<code>@Html.LabelFor<></code>	
41	<code>@Html.LabelForModel</code>	
42	<code>@Html.ListBox</code>	
43	<code>@Html.ListBoxFor<></code>	
44	<code>@Html.Name</code>	
45	<code>@Html.NameFor<></code>	
46	<code>@Html.NameForModel</code>	
47	<code>@Html.Partial</code>	
48	<code>@Html.Password</code>	
49	<code>@Html.PasswordFor<></code>	
50	<code>@Html.RadioButton</code>	
51	<code>@Html.RadioButtonFor<></code>	
52	<code>@Html.Raw</code>	
53	<code>@Html.RenderAction</code>	void
54	<code>@Html.RenderPartial</code>	void
55	<code>@Html.RouteLink</code>	
56	<code>@Html.SetValidationMessageElement</code>	void
57	<code>@Html.SetValidationSummaryMessageElement</code>	void
58	<code>@Html.TextArea</code>	
59	<code>@Html.TextAreaFor<></code>	
60	<code>@Html.TextBox</code>	
61	<code>@Html.TextBoxFor<></code>	
62	<code>@Html.ToString</code>	
63	<code>@Html.Validate</code>	void
64	<code>@Html.ValidateFor<></code>	void

No	Sintaks	Tipe
65	<code>@Html.ValidationMessage</code>	
66	<code>@Html.ValidationMessageFor<></code>	
67	<code>@Html.ValidationSummary</code>	
68	<code>@Html.Value</code>	
69	<code>@Html.ValueFor<></code>	
70	<code>@Html.ValueForModel</code>	

Dari seluruh HTML Helper di atas hanya beberapa saja yang akan dibahas pada ebook ini yaitu HTML Helper yang digunakan untuk pembangunan aplikasi pengelolaan pegawai seperti untuk membuat hyperlink, tombol dan form berserta kontrol-kontrol input seperti textbox, dropdownlist dan lain-lain.

Selanjutnya akan dijelaskan satu per satu HTML Helper yang telah tersedia pada framework ASP.NET MVC. Selain itu juga akan diperlihatkan bagaimana implementasinya agar antarmukanya sesuai dengan style sesuai dengan theme yang digunakan di atas.

Akses Method Aksi

Html.Action berfungsi untuk mengeksekusi method aksi yang terdapat pada suatu controller. Karena setiap aksi akan menampilkan view, jika ada halaman view yang memanggil method ini, maka view dari aksi yang dipanggil akan ditampilkan di dalam view yang berisi method ini. Sintaks dari method ini adalah sebagai berikut:

```
@Html.Action("namaAksi")
```

Method di atas dapat digunakan jika aksi yang dipanggil berada di dalam class controller yang sama. Jika memanggil aksi yang dimiliki oleh controller yang berbeda dapat digunakan sintaks seperti berikut:

```
@Html.Action("namaAksi", "namaController")
```

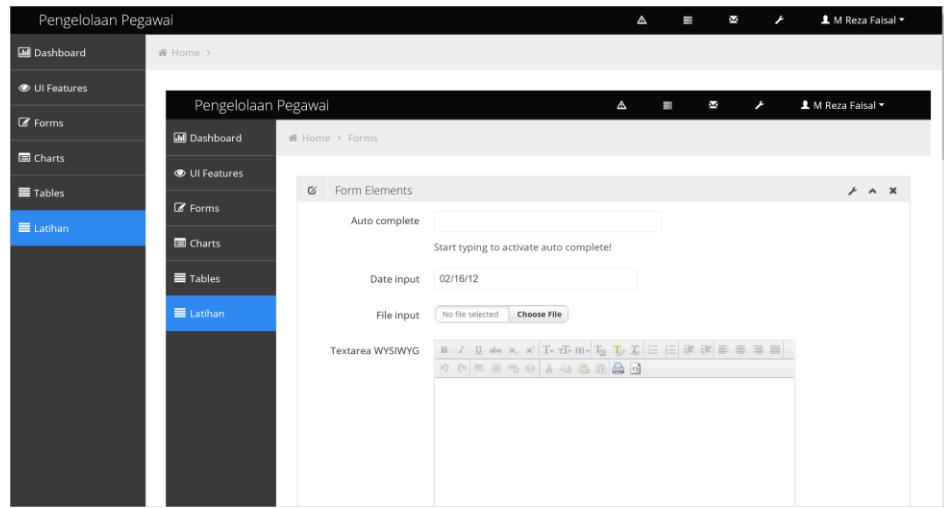
Untuk mencoba method ini, maka dapat dibuat method aksi tambahan pada untuk latihan yaitu method Latihan pada class MetroController seperti berikut ini:

```
public ActionResult Latihan()
{
    return View();
}
```

Dan buat file kosong view Latihan.cshtml, sehingga isi dari file view ini berisi sebaris seperti berikut:

```
@Html.Action("Forms")
```

Dan hasilnya dapat dilihat seperti gambar berikut ini.



Gambar 81. Hasil render halaman view Forms didalam view Latihan.

Dari hasil di atas dapat dilihat pada view Latihan mempunyai template sesuai dengan layout dari MasterLayout.cshtml dan didalamnya berisi halaman view Form.

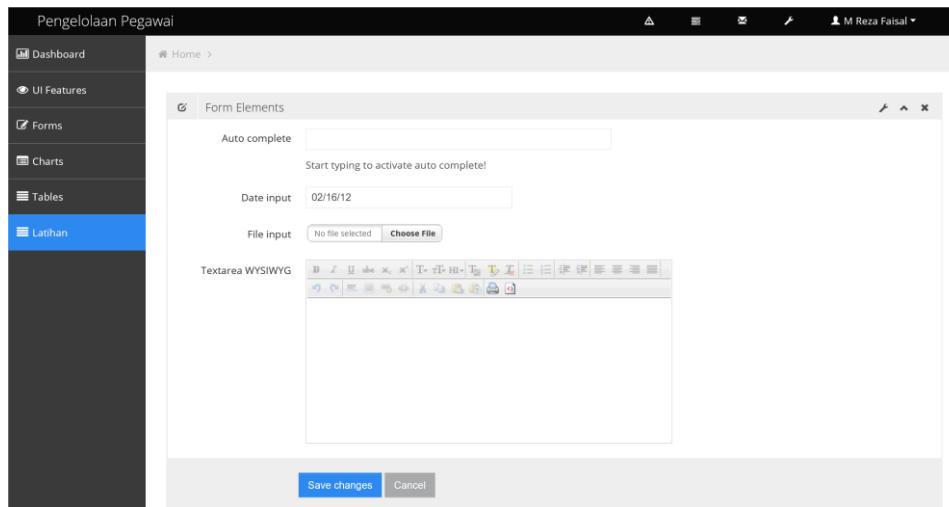
Selain menggunakan `HTML.Action`, untuk mendapatkan hasil yang sama juga dapat dilakukan dengan `HTML.RenderAction` yang bertipe void dengan cara seperti kode di bawah ini:

```
@{
    Html.RenderAction("Forms");
}
```

Tetapi seperti yang terlihat pada gambar di atas, hasilnya berupa antarmuka yang terlihat tumpang tindih. Jika ingin menampilkan konten dari view saja pada view lain dapat menggunakan method `HTML.Partial` dengan contoh kode di bawah ini:

```
@Html.Partial("Forms")
```

Sehingga didapatkan hasil seperti gambar berikut:



Gambar 82. Menampilkan konten view Forms pada halaman view Latihan.

Hyperlink

Html.ActionLink ini berfungsi untuk merender hyperlink yang dapat digunakan untuk mengakses aksi pada suatu controller. Berikut ini adalah sintaks dasar dari helper ini:

```
@Html.ActionLink("linkText", "namaAksi", "namaController")
```

Berikut ini adalah contoh penggunaannya:

```
@Html.ActionLink("Dashboard", "Index", "Metro")
```

Hasil dari helper tersebut akan dirender menjadi kode HTML seperti berikut:

```
<a href="/Metro/Index">Dashboard</a>
```

Saat ini hyperlink tidak sesederhana seperti contoh kode HTML di atas, saat ini hyperlink dapat ditulis dengan lebih rumit seperti contoh pada menu yang terdapat pada theme yang digunakan di atas. Berikut ini adalah contoh hyperlink tersebut.

```
<a href="/Metro/Index">
<i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span>
</a>
```

Hyperlink seperti itu tidak dapat dengan mudah dibuat dengan method Html.ActionLink, sehingga diperlukan cara lain untuk membuat hyperlink dengan menggunakan method Url.Action seperti itu seperti contoh berikut ini:

```
<a href="@Url.Action("Index", "Metro")">
<i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span>
</a>
```

Form

Untuk mempermudah pembuatan form HTML maka telah tersedia beberapa HTML Helper yang dapat dipergunakan untuk keperluan tersebut. Pada bagian ini akan diperlihatkan beberapa cara dan contoh kode yang dapat digunakan untuk membuat form HTML dan bagaimana implementasinya untuk menggunakan style CSS yang telah disediakan pada theme yang digunakan di atas.

Sebagai pendahuluan dan pengenalan, untuk membuat form digunakan kode tag HTML seperti berikut ini. Tag HTML untuk input diletakkan di antara tag HTML form seperti contoh kode di bawah ini.

```
<form action="url" method="post">
    /* tag HTML untuk input ditulis di sini */
</form>
```

Untuk membuat tag HTML itu dapat digunakan HTML Helper dengan contoh sintaks kode seperti berikut ini:

```
@using (Html.BeginForm("namaAksi", "namaController", FormMethod.Post))
{
    /* tag HTML untuk input ditulis di sini */
}
```

Sehingga jika ingin menggunakan method aksi Latihan pada controller Metro dengan method Post dengan class CSS adalah form-horizontal maka kode di atas ditulis dengan kode berikut:

```
@using (Html.BeginForm("Latihan", "Metro", FormMethod.Post, new { @class = "form-horizontal" }))  
{  
    /* tag HTML untuk input ditulis di sini */@  
}
```

Hasil dari kode di atas akan membuat kode HTML seperti berikut ini:

```
<form action="/Metro/Latihan" class="form-horizontal" method="post">  
  
</form>
```

Input Text

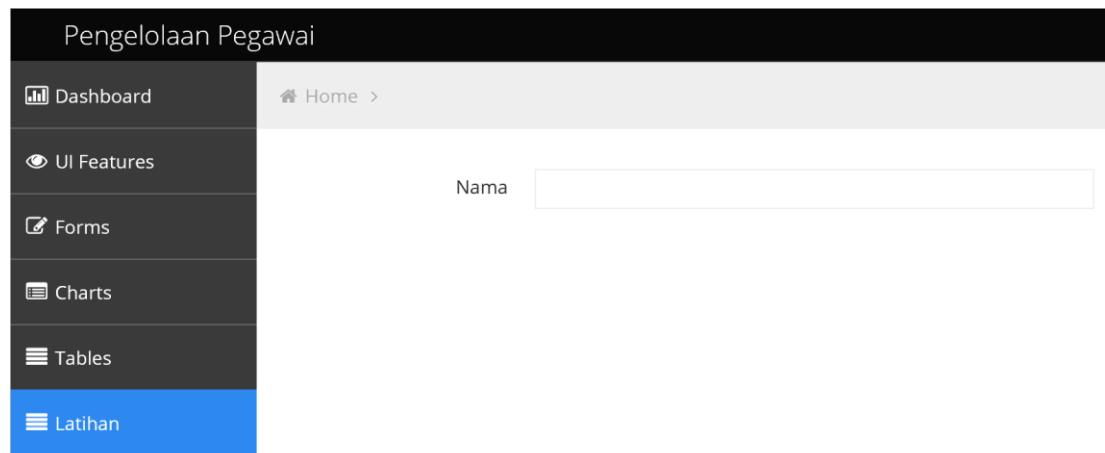
Selanjutnya dapat ditambahkan input isian text di dalam form tersebut. Untuk membuat textbox dapat digunakan beberapa method yang salah satunya adalah Html.TextBox. berikut ini adalah contoh kode dari penggunaan method tersebut.

```
@using (Html.BeginForm("Latihan", "Metro", FormMethod.Post, new { @class = "form-horizontal" }))  
{  
    @Html.Label("TextBoxNama", "Nama", new { @class = "control-label" })  
    @Html.TextBox("TextBoxNama", "", new { @class = "span6 typeahead" })  
}
```

Di bawah ini adalah kode HTML hasil dari kode di atas.

```
<form action="/Metro/Latihan" class="form-horizontal" method="post">  
    <label class="control-label" for="TextBoxNama">Nama</label>  
    <input class="span6 typeahead" id="TextBoxNama" name="TextBoxNama" type="text" value="" />  
</form>
```

Dengan tampilan antarmuka seperti pada gambar di bawah ini.



Gambar 83. Contoh form – textbox.

Dari contoh tersebut menghasilkan sebuah label dengan text bernilai Nama, dan ketika diklik akan membuat cursor berada pada pada textbox. Hal ini bisa terjadi karena pada label diberikan nilai TextBoxNama pada parameter expression, hal ini akan dirender menjadi atribut for dengan nilai TextBoxNama pada kode HTML. Sedangkan untuk menambahkan atribut pada tag HTML dapat dilakukan dengan cara membuat objek dengan property yang sesuai dengan nama atribut HTML yang diinginkan seperti berikut.

```
new { @class = "control-label" }
```

Dengan contoh kode di atas artinya akan ditambahkan atribut class dengan nilai control-label. Sedangkan jika dibuat kode seperti berikut:

```
new { @class = "control-label" , @accesskey = "N" }
```

Maka akan dirender tag label dengan atribut seperti berikut:

```
<label accesskey="N" class="control-label" for="TextBoxNama">Nama</label>
```

Jika ingin membuat antarmuka form yang lebih menarik seperti yang dicontohkan pada halaman Forms maka dapat digunakan kode HTML tambahan seperti berikut ini.

```
<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon edit"></i><span
class="break"></span>Form Latihan</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Latihan", "Metro", FormMethod.Post, new
{ @class = "form-horizontal" }))
            {
                <fieldset>
                    <div class="control-group">
                        @Html.Label("TextBoxNama", "Nama", new { @class =
"control-label" })
                        <div class="controls">
                            @Html.TextBox("TextBoxNama", String.Empty, new {
@class = "span6 typeahead" })
                        </div>
                    </div>
                </fieldset>
            }
        </div>
    </div>
</div>
```

Sehingga didapat antarmuka seperti gambar berikut ini.



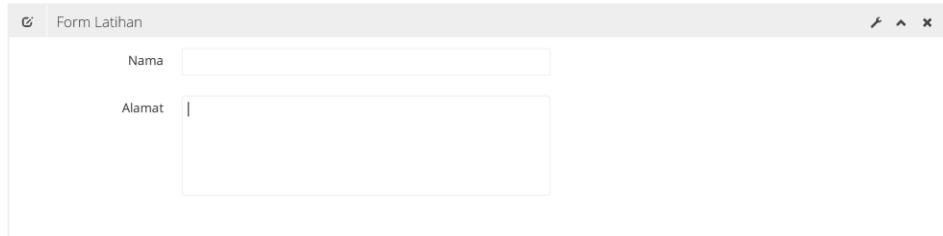
Gambar 84. Contoh form – textbox dengan perbaikan antarmuka.

HTML Helper lain yang dapat digunakan sebagai input text adalah HTML.TextArea. Contoh kode yang dapat digunakan adalah seperti berikut ini.

```
@Html.Label("TextAreaAlamat", "Alamat", new { @class = "control-label" })
```

```
@Html.TextArea("TextAreaAlamat", String.Empty, new { @rows="5", @class = "span6 typeahead" })
```

Dengan parameter di atas, maka akan dibuat textarea dengan atribut id dan name yang bernilai TextAreaAlamat dan memiliki atribut rows dengan nilai 5. Maka dapat dilihat textarea dengan jumlah baris sebanyak 5 seperti yang telihat pada gambar.

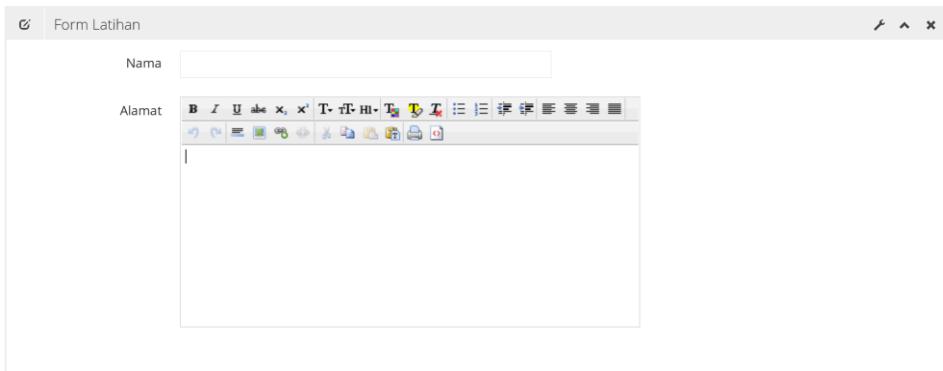


Gambar 85. Contoh form – textarea.

Karena pada ebook ini menggunakan theme Bootstrap Metro Dashboard, maka dapat dibuat textarea sebagai editor WYSIWYG (What You See Is What You Get) dengan kode sebagai berikut.

```
@Html.TextArea("TextAreaAlamat", String.Empty, new { @class = "cleditor" })
```

Sehingga dihasilkan antarmuka seperti pada gambar berikut ini.



Gambar 86. Contoh form – textarea untuk editor WYSIWYG.

Input Satu Pilihan

Salah satu input satu pilihan adalah ketika pengguna memilih propinsi tempat tinggal atau jenis kelamin. Salah satu HTML Helper yang dapat digunakan untuk kebutuhan tersebut adalah HTML.DropDownList dengan sintaks penggunaan sebagai berikut.

```
@Html.DropDownList("name_atau_id_kontrol", data, "label")
```

Sebagai contoh kode yang bisa digunakan adalah sebagai berikut:

```
@Html.DropDownList("DropDownListPropinsi",
    new SelectList(new List<Object> {
        new { value = "KALSEL", text = "Kalimantan Selatan" },
        new { value = "JABAR", text = "Jawa Barat" }
    },
    "value", "text", 2), "Pilih Propinsi")
```

Kode di atas akan menghasilkan kode HTML sebagai berikut:

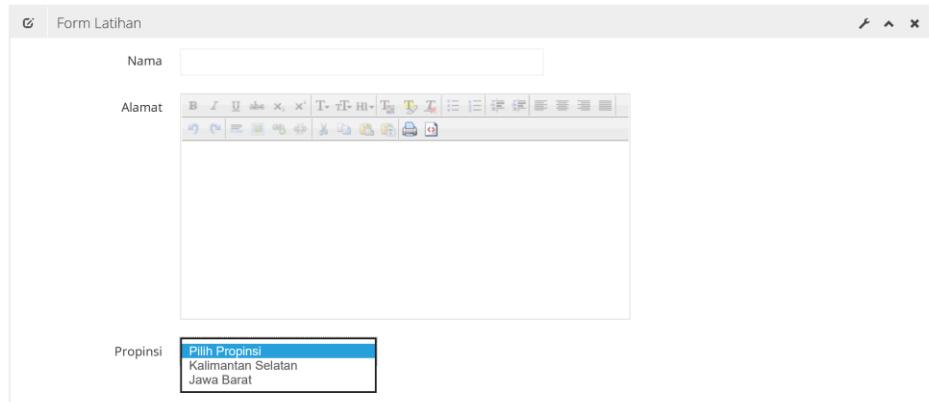
```
<select id="DropDownListPropinsi" name="DropDownListPropinsi">
    <option value="">Pilih Propinsi</option>
    <option value="KALSEL">Kalimantan Selatan</option>
```

```

<option value="JABAR">Jawa Barat</option>
</select>

```

Dengan antarmuka seperti gambar berikut ini.



Gambar 87. Contoh form – dropdownlist.

Cara di atas hanya untuk contoh untuk memperlihatkan penggunaan method `HTML.DropDownList` secara langsung di view, pada praktiknya nanti untuk mengisi data pada kontrol pada view dapat dilakukan melalui controller atau menggunakan model. Cara tersebut akan dibahas lebih lanjut pada bagian selanjutnya.

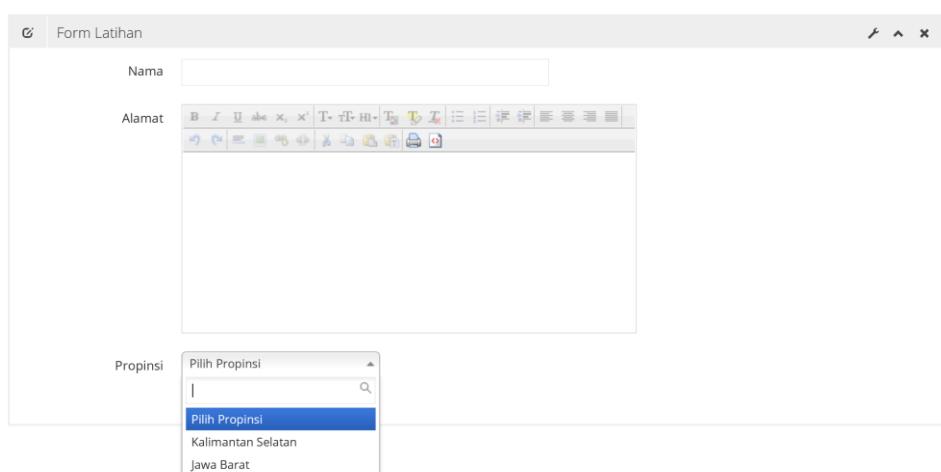
Jika tampilan kontrol dropdownlist ini sesuai dengan theme Bootstrap Metro Dashboard maka dapat ditambahkan atribut data-rel dengan nilai chosen seperti kode berikut ini.

```

@Html.DropDownList("DropDownListPropinsi",
    new SelectList(new List<Object> {
        new { value = "KALSEL", text ="Kalimantan Selatan" },
        new { value = "JABAR", text ="Jawa Barat" }
    }, "value", "text", 2), "Pilih Propinsi", new { @data_rel = "chosen" })

```

Perlu diperhatikan karena pada variabel tidak dapat menggunakan tanda – sehingga pada penulisan dapat diganti dengan tanda _, dan nanti secara otomatis framework ASP.NET MVC akan mengubahnya saat dirender menjadi kode HTML.



Gambar 88. Contoh form – dropdownlist dengan theme Bootstrap Metro Dashboard.

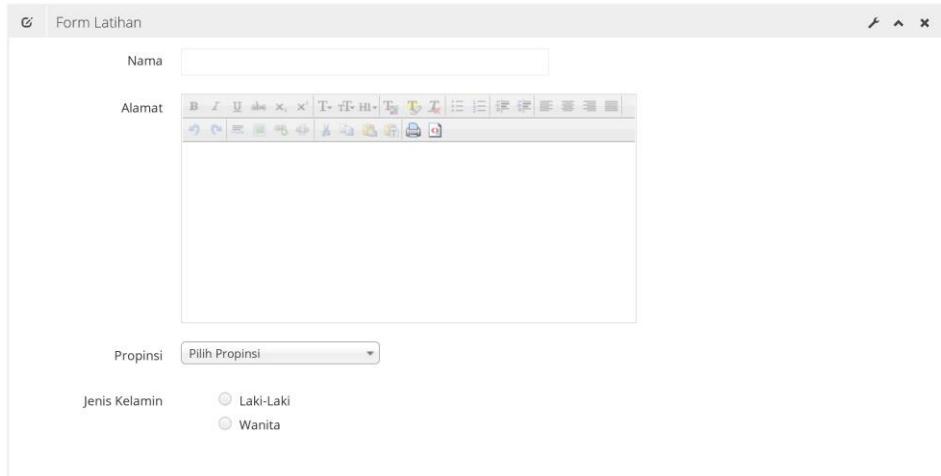
Kontrol yang dapat digunakan sebagai input satu pilihan yang lain adalah radiobutton. HTML Helper yang dapat digunakan adalah HTML.RadioButton yang dapat digunakan dengan sintaks seperti berikut.

```
@Html.RadioButton("name", "value") text_keterangan
```

Sebagai contoh implementasinya dapat dilihat pada kode di bawah ini.

```
@Html.RadioButton("RadioButtonJenisKelamin", "Laki-laki") Laki-Laki  
@Html.RadioButton("RadioButtonJenisKelamin", "Wanita") Wanita
```

Sehingga akan dihasilkan antarmuka seperti pada gambar berikut.



Gambar 89. Contoh form – radiobutton.

Input Banyak Pilihan

Salah satu contoh input banyak pilihan adalah input pemilihan bahasa pemrograman yang dikuasai. Pengguna dapat memilih lebih dari satu bahasa pemrograman yang dikuasainya. Untuk membuat input banyak pilihan seperti itu dapat digunakan HTML.Checkbox dengan sintaks penggunaan seperti berikut:

```
@Html.CheckBox("name") text_keterangan
```

Sebagai contoh dapat dilihat pada kode di bawah ini:

```
@Html.CheckBox("CheckBoxBahasaCSharp") C#  
@Html.CheckBox("CheckBoxBahasaJava") Java  
@Html.CheckBox("CheckBoxBahasaPhyton") Python
```

Hasil dari contoh kode di atas menghasilkan kode HTML seperti di bawah ini:

```
<input id="CheckBoxBahasaC_" name="CheckBoxBahasaC#" type="checkbox"  
value="true" />  
<input name="CheckBoxBahasaC#" type="hidden" value="false" /> C#  
  
<input id="CheckBoxBahasaJava" name="CheckBoxBahasaJava" type="checkbox"  
value="true" />  
<input name="CheckBoxBahasaJava" type="hidden" value="false" /> Java  
  
<input id="CheckBoxBahasaPhyton" name="CheckBoxBahasaPhyton" type="checkbox"  
value="true" />
```

```
<input name="CheckBoxBahasaPhyton" type="hidden" value="false" /> Python
```

Dari hasil di atas dapat diketahui bahwa kode selain dirender tag input type checkbox juga ditambahkan tag input type hidden yang mendampingi.

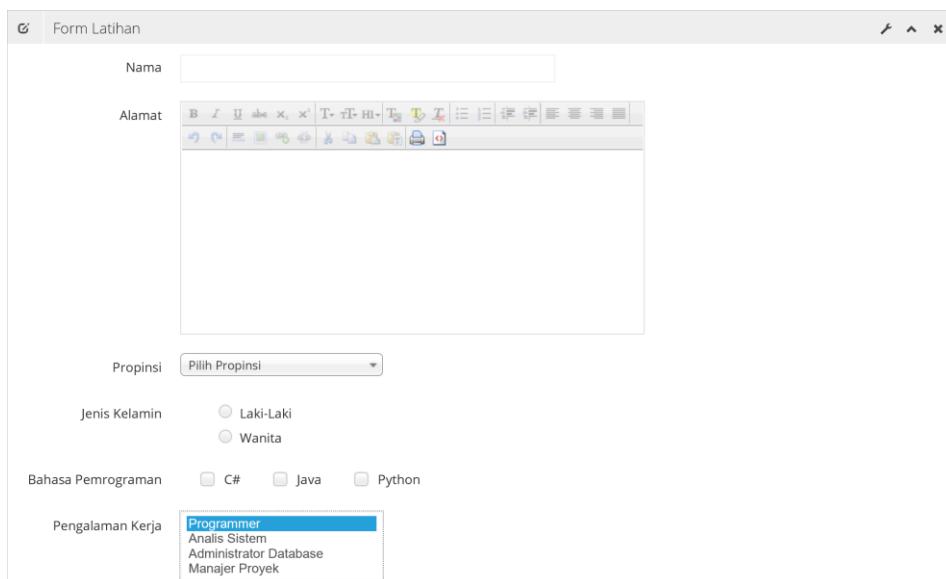
HTML Helper yang lain yang dapat digunakan untuk kebutuhan ini adalah HTML.ListBox dengan sintaks seperti berikut ini.

```
@Html.ListBox("name", data)
```

Berikut adalah contoh dapat digunakan kode sebagai berikut:

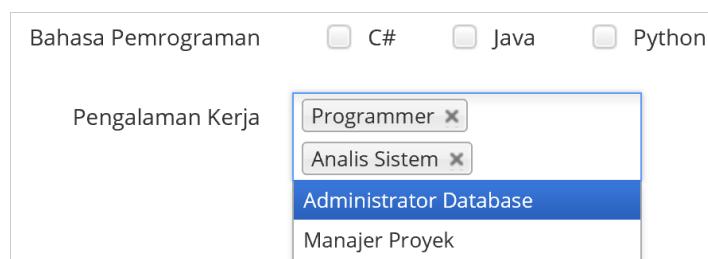
```
@Html.ListBox("ListBoxPekerjaan", new SelectList(new List<Object> {
    new { value = "1", text = "Programmer" },
    new { value = "2", text = "Analisis Sistem" },
    new { value = "3", text = "Administrator Database" },
    new { value = "4", text = "Manajer Proyek" },
}, "value", "text", 2))
```

Dan hasilnya dapat dilihat pada gambar di bawah ini.



Gambar 90. Contoh form – listbox.

Dengan sedikit penambahan atribut data-rel dengan nilai chosen maka dapat dilihat perubahan antarmuka kontrol listbox yang memanfaatkan style dari theme Bootstrap Metro Dashboard.



Gambar 91. Contoh form – listbox dengan theme Bootstrap Metro Dashboard.

Tombol

Untuk membuat tombol submit data tidak tersedia HTML Helper, sehingga cukup digunakan tag HTML untuk membuat tombol seperti contoh berikut ini.

```
<button type="submit" class="btn btn-primary">Simpan</button>
<button type="reset" class="btn">Batal</button>
```

Sehingga dapat dilihat antarmuka form secara lengkap seperti gambar berikut ini.

The screenshot shows a window titled 'Form Latihan'. It contains several input fields: a text input for 'Nama', a rich text editor for 'Alamat' with a toolbar, a dropdown for 'Propinsi' with the placeholder 'Pilih Propinsi', a radio button group for 'Jenis Kelamin' (Laki-Laki, Wanita), a checkbox group for 'Bahasa Pemrograman' (C#, Java, Python), and a text input for 'Pengalaman Kerja' with the placeholder 'Analis Sistem'. At the bottom, there are two buttons: a blue 'Simpan' button and a grey 'Batal' button.

Gambar 92. Contoh form – tombol.

ViewData dan ViewBag

Pada ASP.NET MVC terdapat sarana yang dapat digunakan untuk melakukan pertukaran data antara controller dan view yaitu ViewData dan ViewBag. Keduanya sangat membantu untuk melakukan komunikasi pada sisi server antara controller dan view. Umur dari keduanya adalah singkat, sehingga valuenya akan bernilai null jika terjadi proses redirect halaman.

Sedangkan perbedaan keduanya dapat dilihat dari bagaimana implementasinya pada kode yang ditulis. Pada bagian sebelumnya sudah diperlihatkan beberapa contoh kode pada controller dan view yang menggunakan ViewBag.

Pada bagian ini akan diberikan contoh yang terkait form yang telah dibuat di atas. Form yang telah dibuat di atas menambah method aksi Latihan pada class MetroController.cs sehingga dapat dilihat isi class ini sebagai berikut :

```

MetroController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace PengelolaanPegawai.Web.Controllers
{
    public class MetroController : Controller
    {
        // GET: Metro
        public ActionResult Index()
        {
            ViewBag.Title = "Dashboard";
            return View();
        }

        public ActionResult UI()
        {
            return View();
        }

        public ActionResult Forms()
        {
            return View();
        }

        public ActionResult Charts()
        {
            return View();
        }

        public ActionResult Tables()
        {
            return View();
        }

        public ActionResult Latihan()
        {
            return View();
        }
    }
}

```

Untuk method aksi Latihan diperlukan file view Latihan.cshtml yang merupakan implementasi HTML Helper untuk membangun form yang telah diterangkan di atas. Berikut ini adalah isi secara lengkap file Latihan.cshtml.

```

Latihan.cshtml
@{
    ViewBag.Title = "Latihan";
}
<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon edit"></i><span
class="break"></span>Form Latihan</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>

```

```

        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
<div class="box-content">
    @using (Html.BeginForm("Latihan", "Metro", FormMethod.Post, new { @class = "form-horizontal" }))
    {
        <fieldset>
            <div class="control-group">
                @Html.Label("TextBoxNama", "Nama", new { @class = "control-label", @accesskey = "N" })
                <div class="controls">
                    @Html.TextBox("TextBoxNama", String.Empty, new { @class = "span6 typeahead" })
                </div>
            </div>
            <div class="control-group">
                @Html.Label("TextAreaAlamat", "Alamat", new { @class = "control-label" })
                <div class="controls">
                    @Html.TextArea("TextAreaAlamat", String.Empty, new { @class = "cleditor" })
                </div>
            </div>
            <div class="control-group">
                @Html.Label("DropDownListPropinsi", "Propinsi", new { @class = "control-label" })
                <div class="controls">
                    @Html.DropDownList("DropDownListPropinsi",
                        new SelectList(new List<Object> {
                            new { value = "KALSEL", text = "Kalimantan Selatan" },
                            new { value = "JABAR", text = "Jawa Barat" }
                        }, "value", "text", 2), "Pilih Propinsi",
                        new { @data_rel = "chosen" })
                </div>
            </div>
            <div class="control-group">
                @Html.Label("RadioButtonJenisKelamin", "Jenis Kelamin", new { @class = "control-label" })
                <div class="controls">
                    <label class="radio">
                        @Html.RadioButton("RadioButtonJenisKelamin", "Laki-Laki")
                    </label>
                    <div style="clear:both"></div>
                    <label class="radio">
                        @Html.RadioButton("RadioButtonJenisKelamin", "Wanita")
                    </label>
                </div>
            </div>
            <div class="control-group">
                @Html.Label("CheckBoxBahasaC#", "Bahasa Pemrograman", new { @class = "control-label" })
                <div class="controls">
                    <label class="checkbox inline">

```

```

    @Html.CheckBox("CheckBoxBahasaCSharp") C#
</label>
<label class="checkbox inline">
    @Html.CheckBox("CheckBoxBahasaJava") Java
</label>
<label class="checkbox inline">
    @Html.CheckBox("CheckBoxBahasaPhyton")

```

Python

```

        </label>
    </div>
</div>
<div class="control-group">
    @Html.Label("ListBoxPekerjaan", "Pengalaman Kerja",
new { @class = "control-label" })
    <div class="controls">
        @Html.ListBox("ListBoxPekerjaan",
        new SelectList(new List<Object> {
            new { value = "1", text = "Programmer" },
            new { value = "2", text = "Analis Sistem"
},
            new { value = "3", text = "Administrator
Database" },
            new { value = "4", text = "Manajer
Proyek" },
        }, "value", "text", 2), new { @data_rel =
"chosen" })
        </div>
    </div>
    <div class="form-actions">
        <button type="submit" class="btn btn-
primary">Simpan</button>
        <button type="reset" class="btn">Batal</button>
    </div>
</fieldset>
}
</div>
</div>
</div>

```

Karena pada method yang digunakan pada form di atas adalah POST maka diperlukan method aksi tambahan untuk menangani ketika tombol submit diklik. Dari view di atas dapat dilihat tidak melibatkan model sehingga untuk menangani nilai-nilai setiap input yang ada pada form di atas. Sehingga perlu dibuat method aksi seperti berikut:

```

[HttpPost]
public ActionResult Latihan(string TextBoxNama, string TextAreaAlamat,
string DropDownListPropinsi, string RadioButtonJenisKelamin, bool
CheckBoxBahasaCSharp, bool CheckBoxBahasaJava, bool CheckBoxBahasaPhyton,
IEnumerable<string> ListBoxPekerjaan)
{
    string languages = String.Empty;
    string jobs = String.Empty;

    if (CheckBoxBahasaCSharp)
    {
        languages = "C#" + " " + languages;
    }

    if (CheckBoxBahasaJava)
    {
        languages = "Java" + " " + languages;
    }
}

```

```

if (CheckBoxBahasaPhyton)
{
    languages = "Phyton" + " " + languages;
}

foreach(string job in ListBoxPekerjaan)
{
    if (job.Equals("1"))
    {
        jobs = "Programmer " + jobs;
    }
    else if (job.Equals("2"))
    {
        jobs = "Analis Sistem " + jobs;
    }
    else if (job.Equals("3"))
    {
        jobs = "Administrator Database " + jobs;
    }
    else
    {
        jobs = "Manajer Proyek " + jobs;
    }
}

ViewBag.Hasil = TextBoxNama + " " + TextAreaAlamat + " " +
DropDownListPropinsi + " " + RadioButtonJenisKelamin + " " + languages + " "
+ jobs;
return View();
}

```

Dari contoh di atas dapat dilihat parameter dari method aksi di atas berisi name dari setiap input yang digunakan di dalam form. Untuk setiap tipe data untuk setiap parameter dapat disesuaikan dengan nilai dari setiap input.

```

public ActionResult Latihan(string TextBoxNama, string TextAreaAlamat, string
DropDownListPropinsi, string RadioButtonJenisKelamin, bool
CheckBoxBahasaCSharp, bool CheckBoxBahasaJava, bool CheckBoxBahasaPhyton,
IEnumerable<string> ListBoxPekerjaan)
{
}

```

Sebagai contoh input text untuk Nama digunakan tipe data string. Input text checkbox untuk Bahasa Pemrograman digunakan tipe data boolean. Sedangkan untuk input banyak pilihan seperti listbox untuk Pengalaman Kerja dapat digunakan tipe IEnumerable<string> sehingga dapat menampung lebih dari satu nilai yang dipilih.

Selanjutnya nilai-nilai parameter akan diolah sesuai dengan keinginan. Hasilnya kemudian akan ditampung pada ViewBag.Hasil. Sedangkan untuk menampilkannya pada view cukup gunakan baris berikut ini.

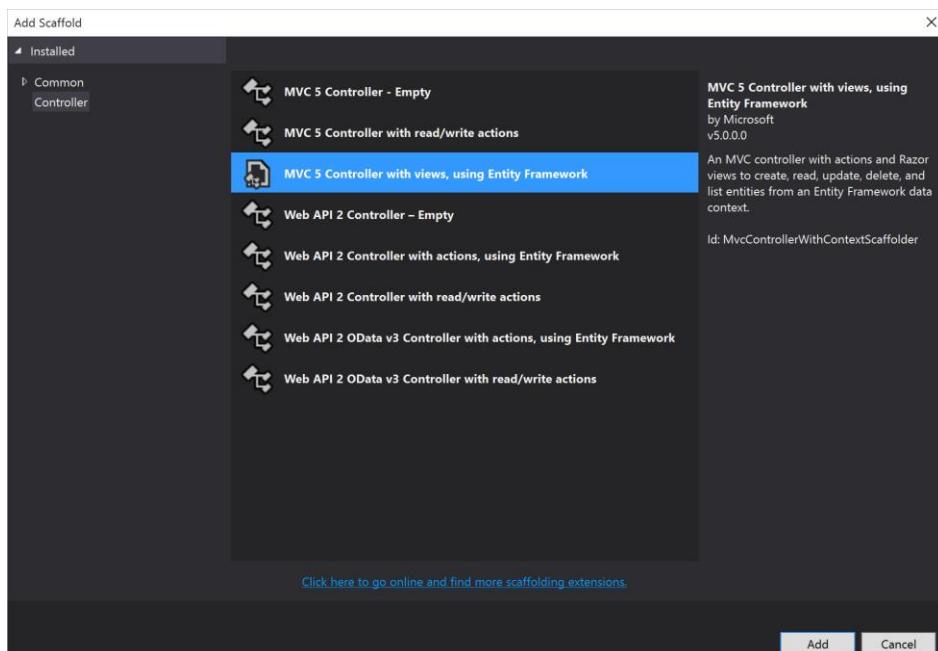
```
@ViewBag.Hasil
```

Dari contoh di atas, dapat dilihat jumlah parameter yang digunakan pada method aksi terlalu banyak. Oleh sebab itu akan lebih baik digunakan model. Contoh penggunaan model untuk penanganan form di atas akan dijelaskan pada bagian selanjutnya yaitu pada bagian Implementasi.

Implementasi

Pada bagian ini akan dibuat dua fitur untuk mengelola data divisi dan data pegawai. Pembangunan fitur ini dengan bantuan fitur template controller scaffold. Sehingga mempermudah untuk membuat sebuah halaman yang lengkap berdasarkan model dari Entity Framework untuk membuat controller yang telah lengkap berisi method aksi dan halaman view untuk setiap method aksi tersebut.

Untuk membuat controller dengan cara ini, langkah-langkah yang umum dilakukan adalah dengan cara klik kanan pada folder Controllers kemudian pilih Add > Controller makan akan ditampilkan window Add Scaffold seperti berikut ini.

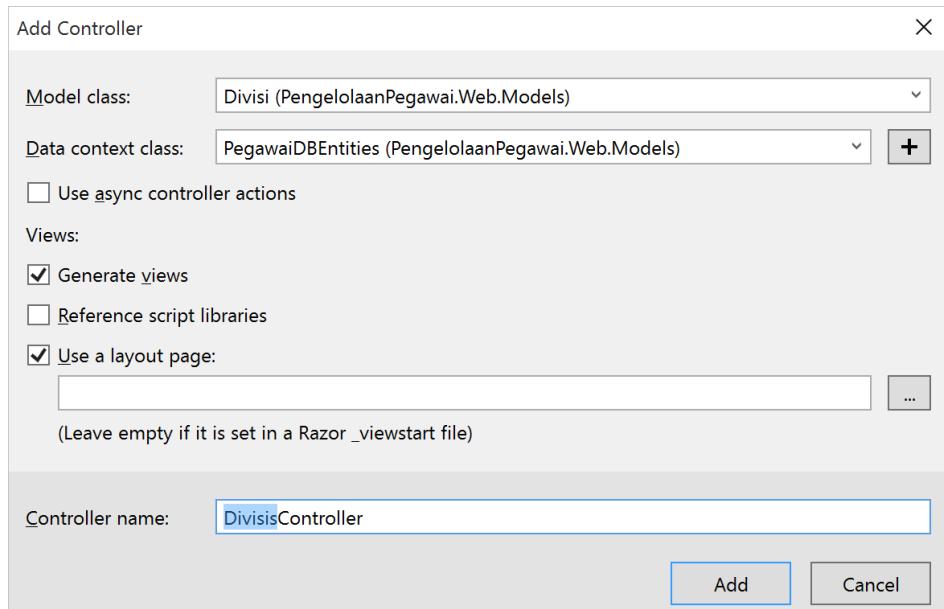


Gambar 93. Menambah MVC 5 Controller with view, using Entity Framework.

Kemudian pilih MVC 5 Controller views, using Entity Framework dan kemudian klik tombol Add. Selanjutnya akan dilanjutkan pada sub bab berikut ini.

Mengelola Divisi

Setelah mengklik tombol Add di atas, maka akan ditampilkan window Add Controller. Pada kolom input Model class dipilih model yang sesuai untuk mengelola data Divisi yaitu Divisi (PengelolaanPegawai.Web.Models). Kemudian pada input Controller name diisi nilai DivisisController.



Gambar 94. Add Controller – Divisi.

Centang pada checkbox Generate views dan Use a layout page kemudian klik tombol Add. Setelah tombol Add diklik maka dilakukan proses generasi kode yang menghasilkan file-file dan folder berikut :

1. DivisisController.cs.
2. Folder Divisis di dalam folder Views dengan file-file sebagai berikut :
 - a. Create.cshtml.
 - b. Delete.cshtml.
 - c. Detail.cshtml.
 - d. Edit.cshtml.
 - e. Index.cshtml.

Berikut ini adalah isi dari file DivisisController.cs.

```
DivisisController.cs
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using PengelolaanPegawai.Web.Models;

namespace PengelolaanPegawai.Web.Controllers
{
    public class DivisisController : Controller
    {
        private PegawaiDBEntities db = new PegawaiDBEntities();

        // GET: Divisis
        public ActionResult Index()
        {
            return View(db.Divisis.ToList());
        }

        // GET: Divisis/Details/5
        public ActionResult Details(int? id)
```

```

{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Divisi divisi = db.Divisis.Find(id);
    if (divisi == null)
    {
        return HttpNotFound();
    }
    return View(divisi);
}

// GET: Divisis/Create
public ActionResult Create()
{
    return View();
}

// POST: Divisis/Create
// To protect from overposting attacks, please enable the specific
properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "DivisiID,NamaDivisi")]
Divisi divisi)
{
    if (ModelState.IsValid)
    {
        db.Divisis.Add(divisi);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(divisi);
}

// GET: Divisis/Edit/5
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Divisi divisi = db.Divisis.Find(id);
    if (divisi == null)
    {
        return HttpNotFound();
    }
    return View(divisi);
}

// POST: Divisis/Edit/5
// To protect from overposting attacks, please enable the specific
properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "DivisiID,NamaDivisi")]
Divisi divisi)

```

```

    {
        if (ModelState.IsValid)
        {
            db.Entry(divisi).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        return View(divisi);
    }

    // GET: Divisis/Delete/5
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Divisi divisi = db.Divisis.Find(id);
        if (divisi == null)
        {
            return HttpNotFound();
        }
        return View(divisi);
    }

    // POST: Divisis/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        Divisi divisi = db.Divisis.Find(id);
        db.Divisis.Remove(divisi);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }
}

```

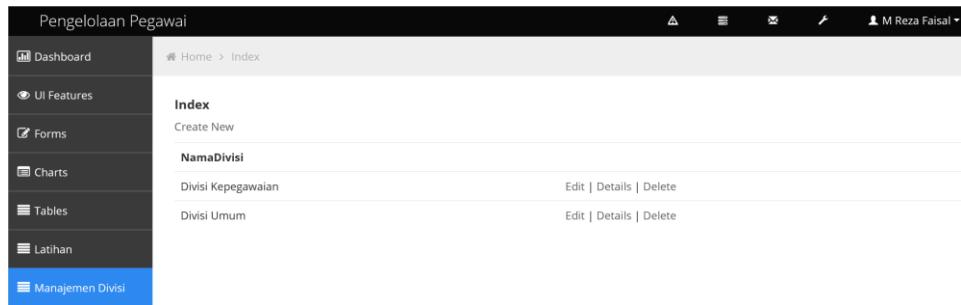
Dari class di atas dapat dilihat terdapat method aksi yaitu :

1. Index/GET.
2. Detail/GET.
3. Create/GET.
4. Create/POST.
5. Edit/GET.
6. Edit/POST.
7. Delete/GET.
8. DeleteConfirmed/POST.

Untuk setiap aksi tersebut akan menggunakan file view seperti yang telah disebutkan di atas.

Index

Method aksi ini bertujuan untuk menampilkan daftar data Divisi yang dapat dilihat pada gambar berikut ini.



Gambar 95. Daftar divisi.

Tampilan dari halaman ini merupakan hasil dari view Index.cshtml yang merupakan hasil dari generasi oleh Visual Studio. Tampilan di atas masih belum menggunakan theme Bootstrap Dashboard Metro yang telah digunakan di atas. Hasil generasi kode secara default dapat dilihat di bawah ini.

```
Index.cshtml
@model IEnumerable<PengelolaanPegawai.Web.Models.Divisi>

@{
    ViewBag.Title = "Index";
}



## Index



Create New



| <a href="#">Divisi Kepegawaian</a> | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
|------------------------------------|-------------------------------------------------------------------------|
| <a href="#">Divisi Umum</a>        | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
|                                    |                                                                         |


```

Untuk mengubah tampilan antarmuka di atas sesuai dengan theme yang digunakan maka kode di atas dimodifikasi dengan kode di bawah ini.

```
Index.cshtml
@model IEnumerable<PengelolaanPegawai.Web.Models.Divisi>
```

```

@{
    ViewBag.Title = "Index";
}



<div class="box span12" style="margin-left:0px">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon user"></i><span
class="break"></span><i class="halflings-icon
plus"></i><@Html.ActionLink("Tambah data", "Create")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        <div class="box-content">
            <table class="table table-striped table-bordered bootstrap-
datatable datatable">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.NamaDivisi)
                    </th>
                    <th>Actions</th>
                </tr>

                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.NamaDivisi)
                        </td>
                        <td>
                            <a class="btn btn-success"
href="@Url.Action("Details", new { id = item.DivisiID })">
                                <i class="halflings-icon white zoom-in"></i>
                            </a>
                            <a class="btn btn-info"
href="@Url.Action("Edit", new { id = item.DivisiID })">
                                <i class="halflings-icon white edit"></i>
                            </a>
                            <a class="btn btn-danger"
href="@Url.Action("Delete", new { id = item.DivisiID })">
                                <i class="halflings-icon white trash"></i>
                            </a>
                        </td>
                    </tr>
                }
            </table>
        </div>
    </div>


```

Sehingga dapat dilihat perubahan antarmuka seperti berikut ini.



Gambar 96. Daftar divisi dengan theme Bootstrap Metro Dashboard.

Penjelasan bagaimana controller mengambil data divisi sampai menampilkannya pada view akan dijelaskan sebagai berikut. Untuk mengambil data divisi dari database dapat dilihat pada class controller DivisisController.cs dimulai dari baris berikut ini.

```
private PegawaiDBEntities db = new PegawaiDBEntities();
```

Baris di atas melakukan instansiasi class entities yang dibuat saat menggenerasi class Entity Framework. Dengan melakukan instansiasi maka secara sederhana telah dilakukan koneksi ke database yang menyimpan data Divisi dan dapat melakukan operasi database dengan memanfaatkan method-method yang dimiliki oleh class tersebut.

Setelah objek db dibuat maka untuk mengambil data Divisi dapat dilakukan dengan baris berikut. Data yang telah diambil disimpan dalam objek collection.

```
db.Divisis.ToList()
```

Agar data tersebut dapat ditampilkan pada bagian view maka data tersebut dapat dikirim ke view, seperti yang dilihat pada method aksi Index berikut ini.

```
public ActionResult Index()
{
    return View(db.Divisis.ToList());
}
```

Objek collection dapat berfungsi sebagai model, sehingga pada view Index.cshtml dapat dilihat deklasi model tersebut seperti baris berikut ini.

```
@model IEnumerable<PengelolaanPegawai.Web.Models.Divisi>
```

Kemudian untuk menampilkan data dari model tersebut digunakan HTML Helper seperti berikut ini.

```
@Html.DisplayFor(modelItem => item.NamaDivisi)
```

Detail

Method aksi Detail berfungsi untuk menampilkan detail record yang dipilih pada tabel daftar data divisi. Untuk mengakses method aksi ini dari tabel, maka caranya adalah dengan mengklik tombol yang berwarna hijau. Ketika tombol tersebut diklik maka method aksi Details akan dieksekusi, berikut ini adalah isi dari method tersebut.

```
public ActionResult Details(int? id)
{
```

```

if (id == null)
{
    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
}
Divisi divisi = db.Divisis.Find(id);
if (divisi == null)
{
    return HttpNotFound();
}
return View(divisi);
}

```

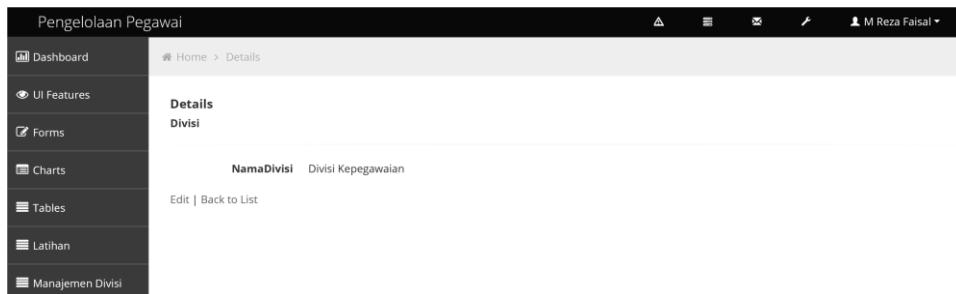
Dari kode method aksi di atas dapat dilihat memiliki parameter id yang merupakan nilai untuk memilih record devisi yang dipilih. Untuk mengambil record sesuai nilai id tersebut digunakan baris berikut ini.

```
Divisi divisi = db.Divisis.Find(id);
```

Kemudian untuk menampilkannya pada halaman view Details.cshtml maka data dikirimkan ke view dengan cara seperti berikut:

```
return View(divisi);
```

Sehingga akan ditampilkan dengan antarmuka seperti berikut ini.



Gambar 97. Detail data devisi.

Dengan kode halaman view Detail.cshtml seperti berikut.

```

Details.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

@{
    ViewBag.Title = "Details";
}



## Details



#### Divisi



---



@Html.DisplayNameFor(model => model.NamaDivisi)


@Html.DisplayFor(model => model.NamaDivisi)

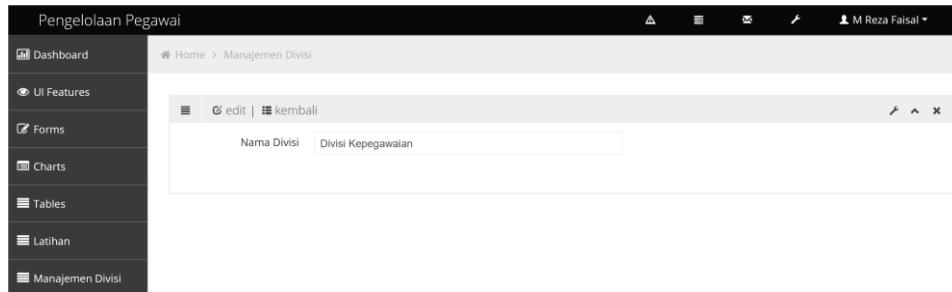

```

```

        </dl>
    </div>
    <p>
        @Html.ActionLink("Edit", "Edit", new { id = Model.DivisiID }) |
        @Html.ActionLink("Back to List", "Index")
    </p>

```

Dengan mengimplementasikan theme yang telah digunakan sebelumnya, maka tampilan halaman detail menjadi seperti berikut.



Gambar 98.Detail divisi dengan theme Bootstrap Metro Dashboard.

Untuk membuat antarmuka seperti itu maka halaman view Detail.cshtml diubah menjadi berikut ini.

```

Details.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

 @{
    ViewBag.Title = "Manajemen Divisi";
}



<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
edit"></i>@Html.ActionLink("edit", "Edit", new { id = Model.DivisiID }) | <i
class="halflings-icon th-list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            <form class="form-horizontal">
                <fieldset>
                    <div class="control-group">
                        <label class="control-label">Nama Divisi</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.NamaDivisi)</span>
                        </div>
                    </div>
                </fieldset>
            </form>
        </div>
    </div>


```

Create

Untuk menambah data memerlukan dua method aksi yaitu method Create dengan method GET dan POST. Ketika link tambah yang terdapat pada daftar data divisi maka akan dipanggil method Create dengan method GET dengan isi method aksi seperti berikut ini.

```
public ActionResult Create()
{
    return View();
}
```

Fungsi method aksi di atas bertujuan untuk menampilkan view Create.cshtml dengan tampilan seperti berikut.



Gambar 99. Form input data divisi.

Dengan isi file Create.cshtml seperti berikut ini.

```
Create.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

 @{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Divisi</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.DivisiID, htmlAttributes: new {
                @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DivisiID, new {
                    htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DivisiID, "", new {
                    @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.NamaDivisi, htmlAttributes: new {
                @class = "control-label col-md-2" })
            <div class="col-md-10">
```

```

        @Html.EditorFor(model => model.NamaDivisi, new {
    htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.NamaDivisi, "", new { @class = "text-danger" })
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" value="Create" class="btn btn-default" />
    </div>
</div>
}
<div>
    @Html.ActionLink("Back to List", "Index")
</div>

```

Setelah tombol Create diklik maka method aksi Create yang menggunakan method POST akan dieksekusi. Berikut ini adalah isi dari method tersebut.

```

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "DivisiID,NamaDivisi")] Divisi
divisi)
{
    if (ModelState.IsValid)
    {
        db.Divisis.Add(divisi);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(divisi);
}

```

Parameter input dari method ini adalah objek divisi yang telah berisi data. Kemudian data tersebut disimpan ke dalam database dengan cara berikut ini.

```

db.Divisis.Add(divisi);
db.SaveChanges();

```

Setelah data berhasil disimpan maka akan dijalankan baris berikut ini, yang akan mengantarkan kembali ke halaman daftar data divisi.

```

return RedirectToAction("Index");

```

Agar tampilan form sesuai dengan theme Bootstrap Metro Dashboard maka Create.cshtml diubah menjadi berikut ini.

```

Create.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

 @{
    ViewBag.Title = "Manajemen Divisi";
}

```

```

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i><@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        <div class="box-content">
            @using (Html.BeginForm("Create", "Divisis", FormMethod.Post, new
{ @class = "form-horizontal" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    @Html.ValidationSummary(true, "", new { @class = "text-
danger" })

                    <div class="control-group">
                        @Html.LabelFor(model => model.DivisiID,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            @Html.EditorFor(model => model.DivisiID, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                            @Html.ValidationMessageFor(model =>
model.DivisiID, "", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model>NamaDivisi,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            @Html.EditorFor(model => model>NamaDivisi, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                            @Html.ValidationMessageFor(model =>
model>NamaDivisi, "", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="form-actions">
                        <button type="submit" class="btn btn-
primary">Simpan</button>
                    </div>
                </fieldset>
            }
        </div>
    </div>
</div>

```

Sehingga ditampilkan antarmuka seperti berikut ini.

Gambar 100. Form create dengan theme Bootstrap Metro Dashboard.

Edit

Proses edit data dimulai dengan memilih record yang akan diedit pada daftar data divisi dengan mengklik tombol warna biru. Kemudian akan ditampilkan form input dengan nilai sesuai dengan record yang dipilih.

Gambar 101. Form edit.

Setelah tombol tersebut diklik maka akan dieksekusi method aksi Edit dengan method GET berikut ini.

```
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Divisi divisi = db.Divisis.Find(id);
    if (divisi == null)
    {
        return HttpNotFound();
    }
    return View(divisi);
}
```

Pada method di atas memiliki parameter id sebagai nilai untuk memilih record yang dipilih, kemudian untuk mengambil record tersebut dari database digunakan baris berikut ini.

```
Divisi divisi = db.Divisis.Find(id);
```

Kemudian objek yang berisi data record yang dipilih dikirimkan ke view dengan cara berikut ini.

```
return View(divisi);
```

Berikut adalah kode dari file Edit.cshtml yang digunakan oleh method aksi Edit untuk menampilkan form edit di atas.

```
>Edit.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

@{
    ViewBag.Title = "Edit";
}



## Edit



@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Divisi</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.DivisiID)

        <div class="form-group">
            @Html.LabelFor(model => model.NamaDivisi, htmlAttributes: new {
                @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NamaDivisi, new {
                    htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NamaDivisi, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Kemudian setelah tombol untuk menyimpan data diklik maka akan dieksekusi method aksi Edit yang menggunakan method POST berikut ini.

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "DivisiID,NamaDivisi")] Divisi
divisi)
{
    if (ModelState.IsValid)
    {
        db.Entry(divisi).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(divisi);
}
```

Method di atas memiliki parameter input untuk menampung objek yang berisi data record yang telah diedit nilainya. Kemudian untuk menyimpan objek tersebut ke dalam database digunakan baris berikut ini.

```
db.Entry(divisi).State = EntityState.Modified;  
db.SaveChanges();
```

Dan jika sukses maka akan di-redirect ke halaman daftar data divisi. Jika ingin mengimplementasikan theme ke halaman ini maka file Edit.cshtml diubah menjadi berikut.

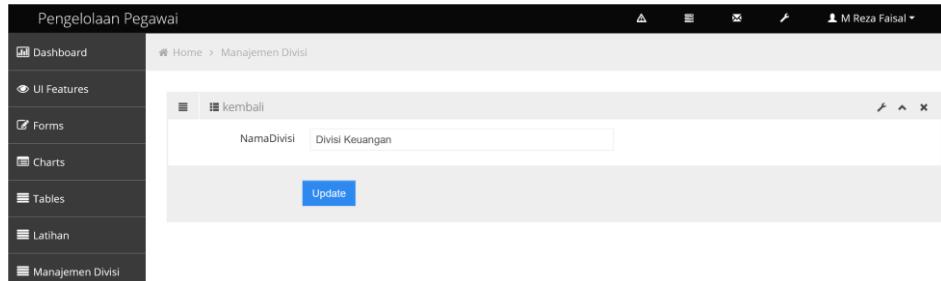
```
Edit.cshtml  
@model PengelolaanPegawai.Web.Models.Divisi  
  
{@  
    ViewBag.Title = "Manajemen Divisi";  
}  
  
<div class="row-fluid sortable">  
    <div class="box span12">  
        <div class="box-header" data-original-title>  
            <h2><i class="halflings-icon align-justify"></i><span  
            class="break"></span><i class="halflings-icon th-  
            list"></i>@Html.ActionLink("kembali", "Index")</h2>  
            <div class="box-icon">  
                <a href="#" class="btn-setting"><i class="halflings-icon  
                wrench"></i></a>  
                <a href="#" class="btn-minimize"><i class="halflings-icon  
                chevron-up"></i></a>  
                <a href="#" class="btn-close"><i class="halflings-icon  
                remove"></i></a>  
            </div>  
            <div class="box-content">  
                @using (Html.BeginForm("Edit", "Divisis", FormMethod.Post, new {  
                    @class = "form-horizontal" }))  
                {  
                    <fieldset>  
                        @Html.AntiForgeryToken()  
                        @Html.ValidationSummary(true, "", new { @class = "text-  
                        danger" })  
                        @Html.HiddenFor(model => model.DivisiID)  
  
                        <div class="control-group">  
                            @Html.LabelFor(model => model.NamaDivisi,  
                            htmlAttributes: new { @class = "control-label" })  
                            <div class="controls">  
                                @Html.EditorFor(model => model.NamaDivisi, new {  
                                    htmlAttributes = new { @class = "span6 typeahead" } })  
                                @Html.ValidationMessageFor(model =>  
                                model.NamaDivisi, "", new { @class = "text-danger" })  
                            </div>  
                        </div>  
  
                        <div class="form-actions">  
                            <button type="submit" class="btn btn-primary">Update</button>  
                        </div>  
                    </fieldset>  
                }  
            </div>
```

```

</div>
</div>

```

Dan berikut adalah tampilan form setelah menggunakan theme tersebut.



Gambar 102. Form edit create dengan theme Bootstrap Metro Dashboard.

Delete

Proses ini untuk menghapus record data yang dipilih, dengan mengklik tombol hapus berwarna merah pada daftar data divisi. Ketika tombol tersebut diklik maka akan dieksekusi method aksi Delete dengan method GET di bawah ini.

```

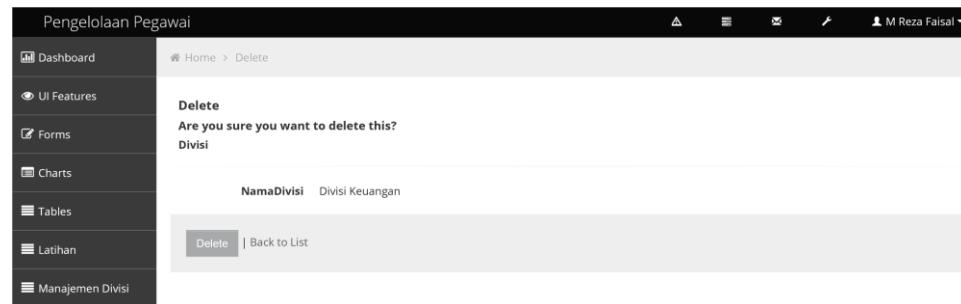
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Divisi divisi = db.Divisis.Find(id);
    if (divisi == null)
    {
        return HttpNotFound();
    }
    return View(divisi);
}

```

Method aksi di atas bertujuan untuk menentukan objek yang akan dihapus dengan baris kode berikut ini.

```
Divisi divisi = db.Divisis.Find(id);
```

Kemudian akan ditampilkan halaman konfirmasi data yang akan dihapus dengan antarmuka seperti berikut.



Gambar 103. Halaman konfirmasi sebelum record yang dipilih dihapus.

Berikut adalah kode view Delete.cshtml dari antarmuka di atas.

```
Delete.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

@{
    ViewBag.Title = "Delete";
}



## Delete



### Are you sure you want to delete this?



#### Divisi



---



@Html.DisplayNameFor(model => model.NamaDivisi)


@Html.DisplayFor(model => model.NamaDivisi)


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-actions no-color">
        <input type="submit" value="Delete" class="btn btn-default" /> |
        @Html.ActionLink("Back to List", "Index")
    </div>
}
</div>
```

Setelah tombol Delete diklik maka akan dieksekusi method aksi DeleteConfirmed berikut ini.

```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Divisi divisi = db.Divisis.Find(id);
    db.Divisis.Remove(divisi);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Untuk menghapus record yang diinginkan maka langkah pertama yang dilakukan adalah memilih objek yang akan dihapus dengan perintah berikut.

```
Divisi divisi = db.Divisis.Find(id);
```

Setelah objek ditemukan, maka perintah untuk menghapus data digunakan baris berikut ini.

```
db.Divisis.Remove(divisi);
db.SaveChanges();
```

Dan kemudian halaman akan diredirect ke halaman daftar data divisi. Untuk mengubah halaman konfirmasi di atas agar menggunakan theme Bootstrap Metro Dashboard, maka dapat melakukan modifikasi halaman di atas dengan kode berikut.

```

Delete.cshtml
@model PengelolaanPegawai.Web.Models.Divisi

@{
    ViewBag.Title = "Manajemen Divisi";
}

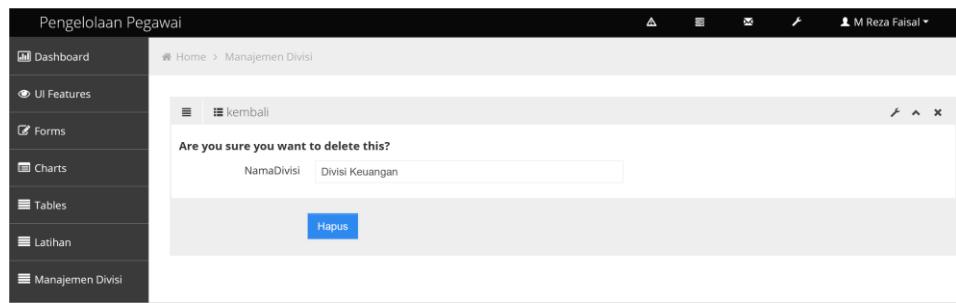


<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("Kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Delete", "Divisis", FormMethod.Post, new
{ @class = "form-horizontal" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    <h3>Are you sure you want to delete this?</h3>
                    <div class="control-group">
                        @Html.LabelFor(model => model.NamaDivisi,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.NamaDivisi)</span>
                        </div>
                    </div>

                    <div class="form-actions">
                        <button type="submit" class="btn btn-
primary">Hapus</button>
                    </div>
                </fieldset>
            }
        </div>
    </div>
</div>


```

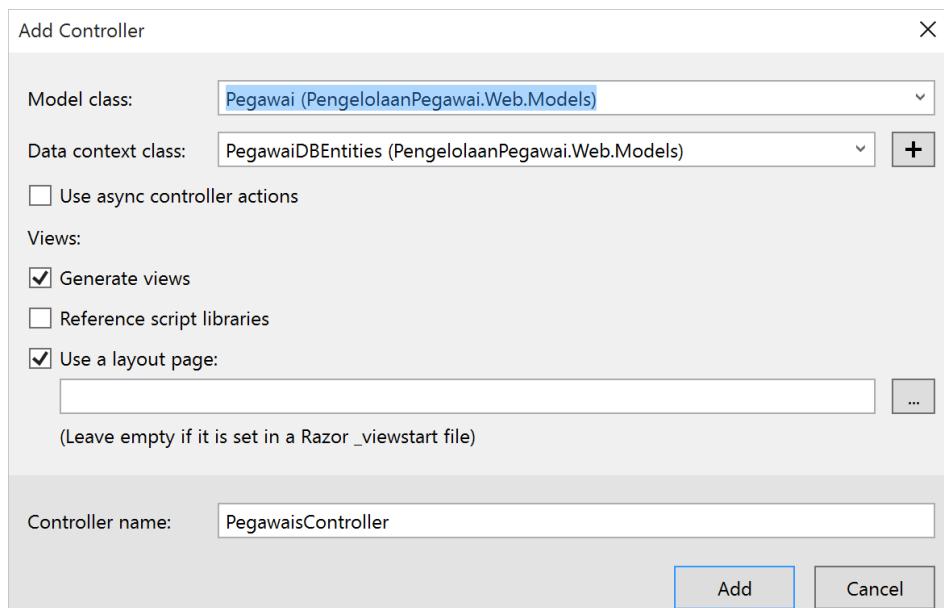
Dan berikut ini adalah antarmuka dari halaman view tersebut.



Gambar 104. Halaman konfirmasi dengan menggunakan theme Bootstrap Metro Dashboard.

Mengelola Pegawai

Untuk membuat fitur manajemen pegawai dapat dibuat dengan langkah-langkah yang sama seperti pada contoh pembuatan manajemen divisi. Dimulai dengan membuat class controller dengan memasukan nilai-nilai berikut pada window Add Controller di bawah ini.



Gambar 105. Window Add Controller – Pegawai.

Maka secara otomatis akan dibuat class controller PegawaiController.cs dengan isi seperti berikut ini.

```
PegawaiController.cs
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using PengelolaanPegawai.Web.Models;

namespace PengelolaanPegawai.Web.Controllers
{
    public class PegawaiController : Controller
    {
        private PegawaiDBEntities db = new PegawaiDBEntities();
```

```

// GET: Pegawais
public ActionResult Index()
{
    var pegawais = db.Pegawais.Include(p => p.Divisi);
    return View(pegawais.ToList());
}

// GET: Pegawais/Details/5
public ActionResult Details(string id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Pegawai pegawai = db.Pegawais.Find(id);
    if (pegawai == null)
    {
        return HttpNotFound();
    }
    return View(pegawai);
}

// GET: Pegawais/Create
public ActionResult Create()
{
    ViewBag.DivisiID = new SelectList(db.Divisis, "DivisiID",
"NamaDivisi");
    return View();
}

// POST: Pegawais/Create
// To protect from overposting attacks, please enable the specific
properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include =
"NIP,DivisiID,Nama,Alamat,TanggalLahir")] Pegawai pegawai)
{
    if (ModelState.IsValid)
    {
        db.Pegawais.Add(pegawai);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    ViewBag.DivisiID = new SelectList(db.Divisis, "DivisiID",
"NamaDivisi", pegawai.DivisiID);
    return View(pegawai);
}

// GET: Pegawais/Edit/5
public ActionResult Edit(string id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Pegawai pegawai = db.Pegawais.Find(id);
    if (pegawai == null)

```

```

        {
            return HttpNotFound();
        }
        ViewBag.DivisiID = new SelectList(db.Divisis, "DivisiID",
"NamaDivisi", pegawai.DivisiID);
        return View(pegawai);
    }

    // POST: Pegawais/Edit/5
    // To protect from overposting attacks, please enable the specific
properties you want to bind to, for
    // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include =
"NIP,DivisiID,Nama,Alamat,TanggalLahir")] Pegawai pegawai)
{
    if (ModelState.IsValid)
    {
        db.Entry(pegawai).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    ViewBag.DivisiID = new SelectList(db.Divisis, "DivisiID",
"NamaDivisi", pegawai.DivisiID);
    return View(pegawai);
}

// GET: Pegawais/Delete/5
public ActionResult Delete(string id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Pegawai pegawai = db.Pegawais.Find(id);
    if (pegawai == null)
    {
        return HttpNotFound();
    }
    return View(pegawai);
}

// POST: Pegawais/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(string id)
{
    Pegawai pegawai = db.Pegawais.Find(id);
    db.Pegawais.Remove(pegawai);
    db.SaveChanges();
    return RedirectToAction("Index");
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}

```

```

        }
    }
}

```

Dan secara otomatis juga akan dibuat file-file view untuk method-method aksi dalam class controller. File-file tersebut disimpan di dalam folder /Views/Pegawai. Berikut adalah file-file view yang dibuat, yaitu :

1. Create.cshtml.
2. Delete.cshtml.
3. Detail.cshtml.
4. Edit.cshtml.
5. Index.cshtml.

Pada bagian hanya akan dilakukan perubahan pada bagian view saja, seperti yang telah dilakukan pada bagian pembahasan Mengelola Divisi.

Index

Berikut adalah antarmuka dari halaman yang menampilkan daftar data pegawai.

Nama	Alamat	TanggalLahir	NamaDivisi	
Mohammad	Banjarmasin	12/12/1980 12:00:00 AM	Divisi Kepegawaian	Edit Details Delete
Faisal	Banjarbaru	7/7/1988 12:00:00 AM	Divisi Kepegawaian	Edit Details Delete

Gambar 106. Daftar pegawai.

Dengan isi file Index.cshtml seperti berikut.

```

Index.cshtml
@model IEnumerable<PengelolaanPegawai.Web.Models.Pegawai>

@{
    ViewBag.Title = "Index";
}



## Index



Create New



| <a href="#">Nama</a> | <a href="#">Alamat</a> | <a href="#">TanggalLahir</a> | <a href="#">NamaDivisi</a> |                                                                         |
|----------------------|------------------------|------------------------------|----------------------------|-------------------------------------------------------------------------|
| Mohammad             | Banjarmasin            | 12/12/1980 12:00:00 AM       | Divisi Kepegawaian         | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |
| Faisal               | Banjarbaru             | 7/7/1988 12:00:00 AM         | Divisi Kepegawaian         | <a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a> |


```

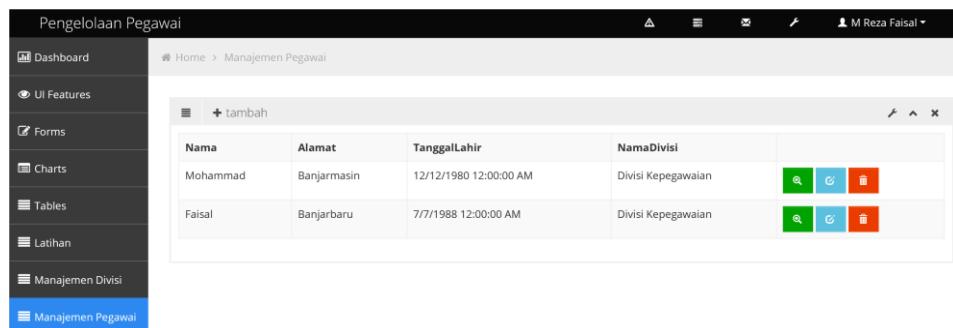
```

        @Html.DisplayNameFor(model => model.Divisi.NamaDivisi)
    </th>
    <th></th>
</tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Nama)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Alamat)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TanggalLahir)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Divisi.NamaDivisi)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.NIP }) |
            @Html.ActionLink("Details", "Details", new { id=item.NIP }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.NIP })
        </td>
    </tr>
}
</table>

```

Setelah kode di atas diubah dengan mengikuti aturan theme maka tampilannya akan menjadi seperti berikut ini.



Gambar 107. Antarmuka daftar pegawai dengan menggunakan theme Bootstrap Metro Dashboard.

Dan berikut ini adalah kode file Index.cshtml yang telah menggunakan theme.

```

Index.cshtml
@model IEnumerable<PengelolaanPegawai.Web.Models.Pegawai>

@{
    ViewBag.Title = "Manajemen Pegawai";
}



<div class="box span12" style="margin-left:0px">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span class="break"></span><i class="halflings-icon plus"></i>@Html.ActionLink("tambah", "Create")</h2>
            <div class="box-icon">


```

```

        <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
<div class="box-content">
    <table class="table table-striped table-bordered bootstrap-datatable">
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Nama)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Alamat)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.TanggalLahir)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Divisi>NamaDivisi)
            </th>
            <th></th>
        </tr>

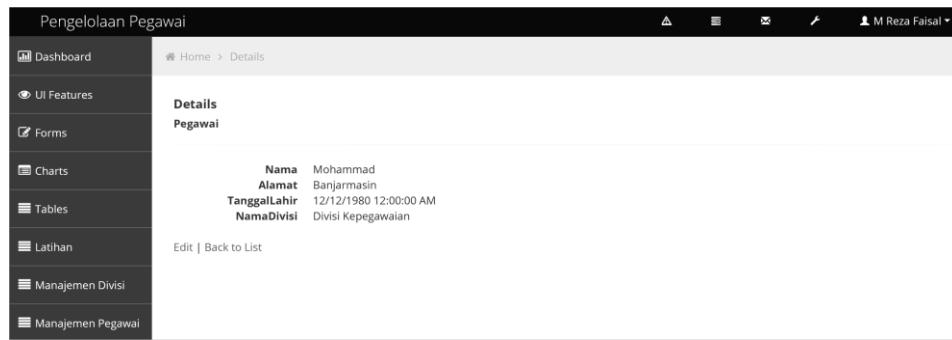
        @foreach (var item in Model)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Nama)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Alamat)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.TanggalLahir)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Divisi>NamaDivisi)
                </td>
                <td>
                    <a class="btn btn-success"
                    href="@Url.Action("Details", new { id = item.NIP })">
                        <i class="halflings-icon white zoom-in"></i>
                    </a>
                    <a class="btn btn-info"
                    href="@Url.Action("Edit", new { id = item.NIP })">
                        <i class="halflings-icon white edit"></i>
                    </a>
                    <a class="btn btn-danger"
                    href="@Url.Action("Delete", new { id = item.NIP })">
                        <i class="halflings-icon white trash"></i>
                    </a>
                </td>
            </tr>
        }
    </table>

```

```
</div>
</div>
</div>
```

Detail

Berikut ini adalah antarmuka untuk menampilkan detail dari record data yang dipilih.



Gambar 108. Antarmuka detail pegawai.

Berikut ini adalah isi file view dari antarmuka di atas.

```
Detail.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Details";
}



## Details



#### Pegawai



---



@Html.DisplayNameFor(model => model.Nama)
:   @Html.DisplayFor(model => model.Nama)


@Html.DisplayNameFor(model => model.Alamat)
:   @Html.DisplayFor(model => model.Alamat)


@Html.DisplayNameFor(model => model.TanggalLahir)
:   @Html.DisplayFor(model => model.TanggalLahir)


```

```

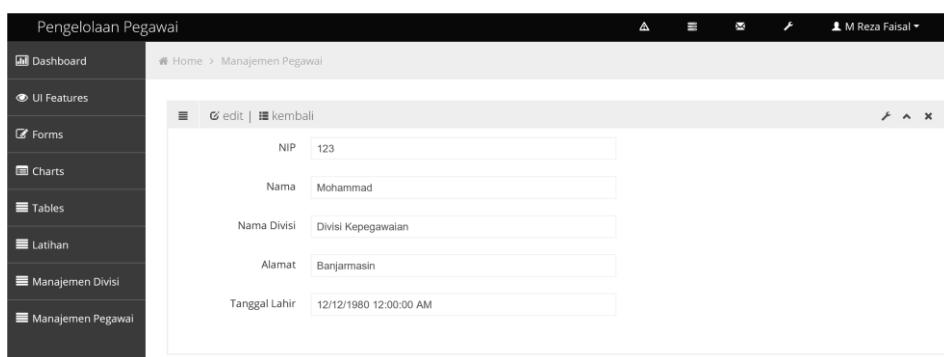
        @Html.DisplayNameFor(model => model.Divisi.NamaDivisi)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Divisi.NamaDivisi)
    </dd>

</dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.NIP }) |
    @Html.ActionLink("Back to List", "Index")
</p>

```

Kemudian setelah kode di atas disesuaikan dengan aturan theme yang digunakan pada ebook ini maka tampilannya akan menjadi seperti berikut ini.



Gambar 109. Antarmuka detail pegawai dengan theme Bootstrap Metro Dashboard.

Dan berikut ini adalah kode file Detail.cshtml yang telah dimodifikasi sesuai aturan theme yang digunakan.

```

Detail.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

 @{
    ViewBag.Title = "Manajemen Pegawai";
}

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
edit"></i>@Html.ActionLink("edit", "Edit", new { id = Model.NIP }) | <i
class="halflings-icon th-list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            <form class="form-horizontal">
                <fieldset>
                    <div class="control-group">
                        <label class="control-label">NIP</label>
                        <div class="controls">

```

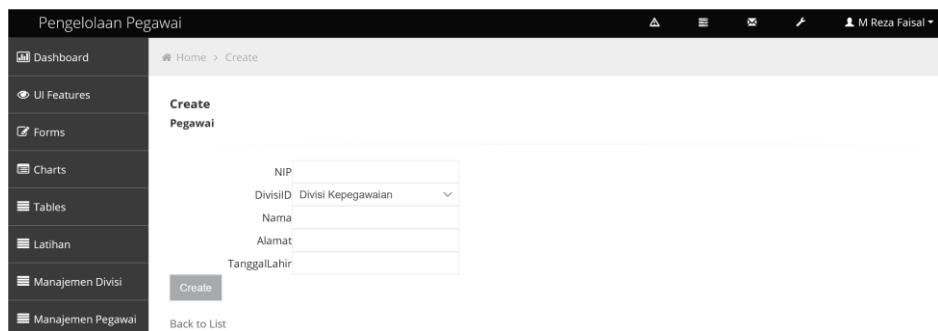
```

<span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.NIP)</span>
</div>
</div>
<div class="control-group">
<label class="control-label">Nama</label>
<div class="controls">
<span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model>Nama)</span>
</div>
</div>
<div class="control-group">
<label class="control-label">Nama Divisi</label>
<div class="controls">
<span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Divisi>NamaDivisi)</span>
</div>
</div>
<div class="control-group">
<label class="control-label">Alamat</label>
<div class="controls">
<span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Alamat)</span>
</div>
</div>
<div class="control-group">
<label class="control-label">Tanggal Lahir</label>
<div class="controls">
<span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.TanggalLahir)</span>
</div>
</div>
</fieldset>
</form>
</div>
</div>
</div>

```

Create

Berikut ini adalah antarmuka form untuk menambahkan data pegawai.



Gambar 110. Antarmuka form input data pegawai.

File view untuk antarmuka di atas adalah file Create.cshtml dengan isi file sebagai berikut.

```
Create.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Pegawai</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.NIP, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NIP, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NIP, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DivisiID, "DivisiID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("DivisiID", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.DivisiID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model>Nama, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model>Nama, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model>Nama, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Alamat, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Alamat, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Alamat, "", new { @class = "text-danger" })
            </div>
        </div>
    </div>
}
```

```

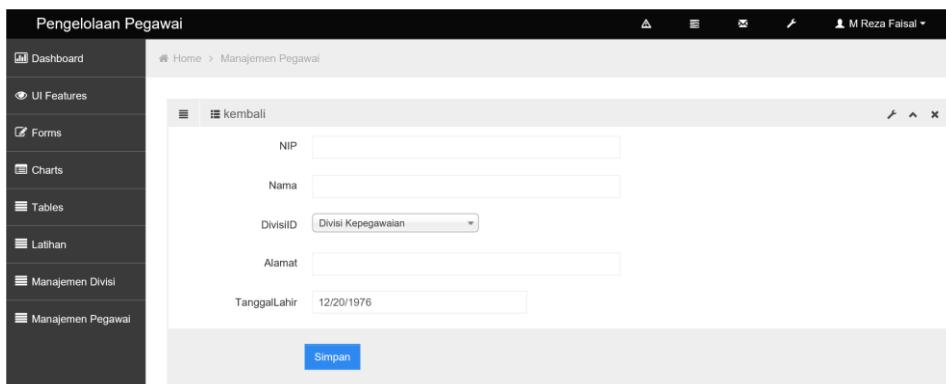
        <div class="form-group">
            @Html.LabelFor(model => model.TanggalLahir, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TanggalLahir, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TanggalLahir, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

```

Setelah isi file Create.cshtml diubah dan disesuaikan dengan theme yang digunakan, maka tampilannya akan menjadi seperti gambar di bawah ini.



Gambar 111. Antarmuka form input dengan theme Bootstrap Metro Dashboard.

Dan berikut isi dari file Create.cshtml dari view ini.

```

Create.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Manajemen Pegawai";
}

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
            </div>
        </div>
        <div class="box-content">
            <form>
                <div class="form-group">
                    @Html.LabelFor(model => model.TanggalLahir, htmlAttributes: new
{ @class = "control-label col-md-2" })
                    <div class="col-md-10">
                        @Html.EditorFor(model => model.TanggalLahir, new {
htmlAttributes = new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.TanggalLahir, "", new { @class = "text-danger" })
                    </div>
                </div>

                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Create" class="btn btn-default" />
                    </div>
                </div>
            </form>
        </div>
    </div>

```

```

                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Create", "Pegawais", FormMethod.Post,
new { @class = "form-horizontal" }))
{
    <fieldset>
        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true, "", new { @class = "text-
danger" })

        <div class="control-group">
            @Html.LabelFor(model => model.NIP, htmlAttributes:
new { @class = "control-label" })
            <div class="controls">
                @Html.EditorFor(model => model.NIP, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                @Html.ValidationMessageFor(model => model.NIP,
"", new { @class = "text-danger" })
            </div>
        </div>

        <div class="control-group">
            @Html.LabelFor(model => model>Nama, htmlAttributes:
new { @class = "control-label" })
            <div class="controls">
                @Html.EditorFor(model => model>Nama, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                @Html.ValidationMessageFor(model => model>Nama,
"", new { @class = "text-danger" })
            </div>
        </div>

        <div class="control-group">
            @Html.LabelFor(model => model.DivisiID,
htmlAttributes: new { @class = "control-label" })
            <div class="controls">
                @Html.DropDownList("DivisiID", null,
htmlAttributes: new { @data_rel = "chosen" })
                @Html.ValidationMessageFor(model =>
model.DivisiID, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="control-group">
            @Html.LabelFor(model => model.Alamat,
htmlAttributes: new { @class = "control-label" })
            <div class="controls">
                @Html.EditorFor(model => model.Alamat, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                @Html.ValidationMessageFor(model =>
model.Alamat, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="control-group">
            @Html.LabelFor(model => model.TanggalLahir,
htmlAttributes: new { @class = "control-label" })
            <div class="controls">

```

```

@Html.EditorFor(model => model.TanggalLahir, new
{ htmlAttributes = new { @class = "input-xlarge datepicker" } })
@Html.ValidationMessageFor(model =>
model.TanggalLahir, "", new { @class = "text-danger" })
</div>
</div>

<div class="form-actions">
<button type="submit" class="btn btn-primary">Simpan</button>
</div>
</fieldset>
}
</div>
</div>
</div>

```

Pada form input ini dapat dilihat kontrol lain selain input text, tetapi juga ada berupa dropdownlist dan juga input text untuk pilihan tanggal. Berikut adalah antarmuka input berupa dropdownlist dengan menggunakan style dari theme ini.

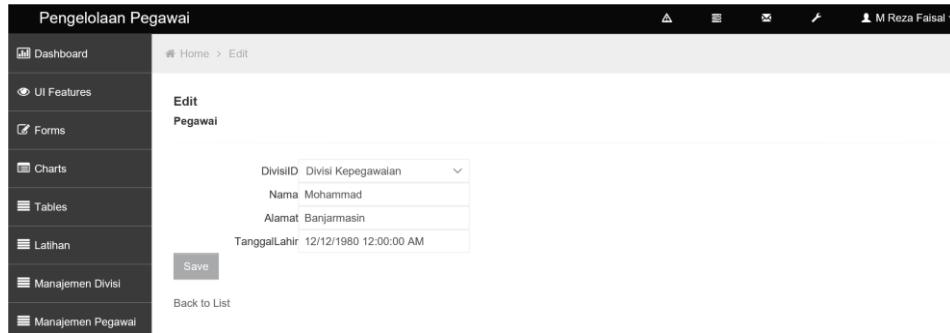
Gambar 112. Form input pegawai – dropdownlist pilihan divisi.

Sedangkan antarmuanya untuk input pilihan tanggal dapat dilihat seperti berikut ini.

Gambar 113. Form input pegawai – pilihan tanggal.

Edit

Berikut ini adalah antarmuka untuk edit data pegawai yang dipilih.



Gambar 114. Form edit pegawai.

Dan berikut ini adalah isi file Edit.cshtml.

```
>Edit.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Edit";
}



## Edit



@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Pegawai</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.NIP)

        <div class="form-group">
            @Html.LabelFor(model => model.DivisiID, "DivisiID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("DivisiID", null, htmlAttributes: new {
                    @class = "form-control" })
                @Html.ValidationMessageFor(model => model.DivisiID, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model>Nama, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model>Nama, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model>Nama, "", new {
                    @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Alamat, htmlAttributes: new {
                @class = "control-label col-md-2" })

```

```

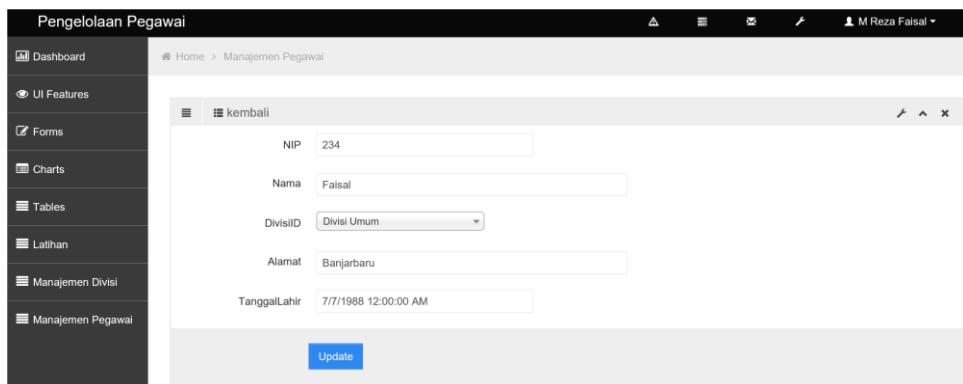
        <div class="col-md-10">
            @Html.EditorFor(model => model.Alamat, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.Alamat, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(model => model.TanggalLahir, htmlAttributes: new { @class = "control-label col-md-2" })
        <div class="col-md-10">
            @Html.EditorFor(model => model.TanggalLahir, new { htmlAttributes = new { @class = "form-control" } })
            @Html.ValidationMessageFor(model => model.TanggalLahir, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Save" class="btn btn-default" />
        </div>
    </div>
</div>
<div>
    @Html.ActionLink("Back to List", "Index")
</div>

```

Untuk menggunakan theme yang digunakan pada ebook ini, maka perlu ditambahkan beberapa kode pada file di atas sehingga antarmukanya menjadi seperti berikut ini.



Gambar 115. Form edit pegawai dengan theme Bootstrap Metro Dashboard.

Dan berikut ini adalah kode dari file Edit.cshtml yang telah diperbarui.

```

Edit.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Manajemen Pegawai";
}

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>

```

```

        <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("kembali", "Index")</h2>
        <div class="box-icon">
            <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
            <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
            <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Edit", "Pegawai", FormMethod.Post, new
{ @class = "form-horizontal" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    @Html.ValidationSummary(true, "", new { @class = "text-
danger" })

                    <div class="control-group">
                        @Html.LabelFor(model => model.NIP, htmlAttributes:
new { @class = "control-label" })
                        <div class="controls">
                            @Html.EditorFor(model => model.NIP, new {
htmlAttributes = new { @class = "input-xlarge uneditable-input" } })
                            @Html.ValidationMessageFor(model => model.NIP,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model>Nama, htmlAttributes:
new { @class = "control-label" })
                        <div class="controls">
                            @Html.EditorFor(model => model>Nama, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                            @Html.ValidationMessageFor(model => model>Nama,
"", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model.DivisiID,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            @Html.DropDownList("DivisiID", null,
htmlAttributes: new { @data_rel = "chosen" })
                            @Html.ValidationMessageFor(model =>
model.DivisiID, "", new { @class = "text-danger" })
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model.Alamat,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            @Html.EditorFor(model => model.Alamat, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                        </div>
                    </div>
            }
        </div>
    
```

```

        @Html.ValidationMessageFor(model =>
model.Alamat, "", new { @class = "text-danger" })
    </div>
</div>

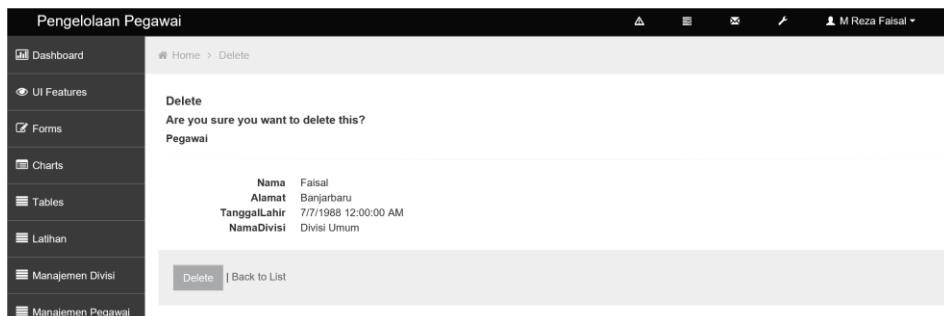
    <div class="control-group">
        @Html.LabelFor(model => model.TanggalLahir,
htmlAttributes: new { @class = "control-label" })
        <div class="controls">
            @Html.EditorFor(model => model.TanggalLahir, new
{ htmlAttributes = new { @class = "input-xlarge datepicker" } })
            @Html.ValidationMessageFor(model =>
model.TanggalLahir, "", new { @class = "text-danger" })
        </div>
    </div>

    <div class="form-actions">
        <button type="submit" class="btn btn-primary">Simpan</button>
    </div>
</fieldset>
}
</div>
</div>
</div>

```

Delete

Berikut adalah antarmuka untuk konfirmasi sebelum data dihapus.



Gambar 116. Antarmuka konfirmasi sebelum data dihapus.

Dan berikut ini adalah isi dari file Delete.cshtml.

```

Delete.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Pegawai</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Nama)

```

```

</dt>

<dd>
    @Html.DisplayFor(model => model.Nama)
</dd>

<dt>
    @Html.DisplayNameFor(model => model.Alamat)
</dt>

<dd>
    @Html.DisplayFor(model => model.Alamat)
</dd>

<dt>
    @Html.DisplayNameFor(model => model.TanggalLahir)
</dt>

<dd>
    @Html.DisplayFor(model => model.TanggalLahir)
</dd>

<dt>
    @Html.DisplayNameFor(model => model.Divisi>NamaDivisi)
</dt>

<dd>
    @Html.DisplayFor(model => model.Divisi>NamaDivisi)
</dd>

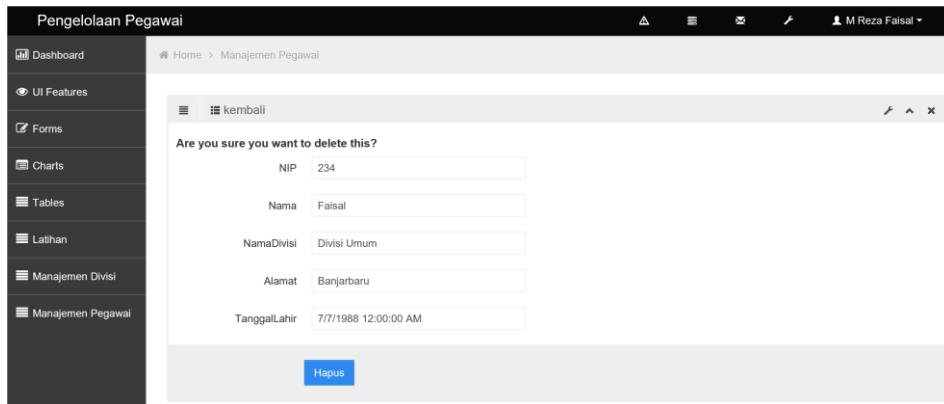
</dl>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()

    <div class="form-actions no-color">
        <input type="submit" value="Delete" class="btn btn-default" /> |
        @Html.ActionLink("Back to List", "Index")
    </div>
}
</div>

```

Setelah file di atas dimodifikasi untuk menyesuaikan dengan theme yang digunakan, maka antarmuka akan menjadi seperti berikut ini.



Gambar 117. Antarmuka konfirmasi penghapusan yang telah memanfaatkan theme.

Dan berikut ini adalah isi file Delete.cshtml yang telah dimodifikasi.

```
>Delete.cshtml
@model PengelolaanPegawai.Web.Models.Pegawai

@{
    ViewBag.Title = "Manajemen Pegawai";
}



<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Delete", "Pegawai", FormMethod.Post,
new { @class = "form-horizontal" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    <h3>Are you sure you want to delete this?</h3>
                    <div class="control-group">
                        @Html.LabelFor(model => model.NIP, htmlAttributes:
new { @class = "control-label" })
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.NIP)</span>
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model>Nama, htmlAttributes:
new { @class = "control-label" })
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model>Nama)</span>
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model.Divisi>NamaDivisi,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.Divisi>NamaDivisi)</span>
                        </div>
                    </div>

                    <div class="control-group">
                        @Html.LabelFor(model => model.Alamat,
htmlAttributes: new { @class = "control-label" })
                        <div class="controls">


```

```

        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Alamat)</span>
            </div>
        </div>

        <div class="control-group">
            @Html.LabelFor(model => model.TanggalLahir,
htmlAttributes: new { @class = "control-label" })
            <div class="controls">
                <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.TanggalLahir)</span>
                </div>
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-primary">Hapus</button>
            </div>
        </fieldset>
    }
</div>
</div>

```

Kesimpulan

Dari penjelasan di atas telah diterangkan beberapa hal yaitu :

1. Dasar-dasar dan cara kerja komponen view dan controller.
2. Pengenalan sintaks Razor.
3. Implementasi penggunaan komponen model dari Entity Framework, view dan controller serta sintaks Razor untuk membuat fitur pengelolaan data divisi dan pegawai.
4. Implementasi theme Bootstrap pada aplikasi.

Dengan penjelasan materi di atas diharapkan pembaca telah dapat membuat web aplikasi sederhana untuk mengelola data pada database.

Referensi

- [https://msdn.microsoft.com/en-us/library/dd381412\(v=vs.108\).aspx](https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx)
- [http://www.asp.net/web-pages/overview/getting-started/introducing-razor-syntax-\(c\)](http://www.asp.net/web-pages/overview/getting-started/introducing-razor-syntax-(c))
- <http://haacked.com/archive/2011/01/06/razor-syntax-quick-reference.aspx/>
- <http://www.codeproject.com/Articles/476967/What-is-ViewData-ViewBag-and-TempData-MVC-Option>

6

Keamanan

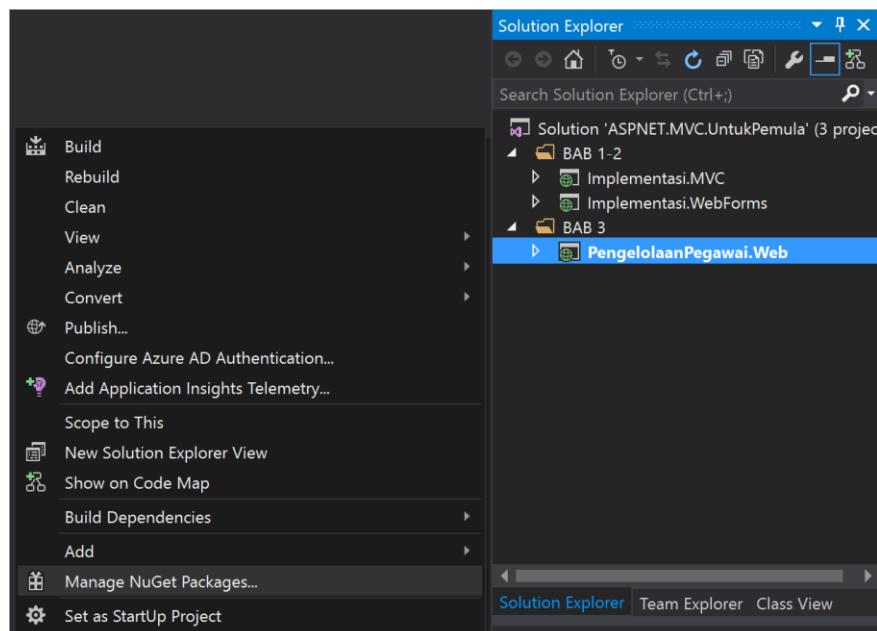
ASP.NET Identity

Untuk keamanan fitur-fitur pada suatu aplikasi web dapat mempergunakan pemeriksaan pengguna yang mengakses fitur tersebut. Cara yang lazim digunakan adalah dengan membuat form login, untuk mengidentifikasi apakah user tersebut terdaftar untuk mengakses fitur tersebut.

Pada tahun 2005 telah dikenal ASP.NET Membership yang berfungsi untuk mengelola proses otentikasi dan otorisasi pada aplikasi web. Tetapi secara default, database yang harus digunakan adalah SQL Server. Kemudian dikenal pula ASP.NET Universal Provider yang telah didukung oleh Entity Framework sebagai akses datanya. Tetapi masih memiliki keterbatasan dalam penggunaan database yaitu hanya dapat menggunakan SQL Server saja.

Sekarang ini telah dikenal ASP.NET Identity, yang memungkinkan pengelolaan proses otentikasi yang lebih bebas. Termasuk kemampuan untuk melakukan otentikasi atau login dengan memanfaatkan pengguna pada social media seperti facebook, twitter atau yang lainnya. ASP.NET Identity dapat digunakan pada semua keluarga ASP.NET Framework seperti ASP.NET MVC, Web Forms, Web Pages, Web API dan SignalR selain itu juga dapat digunakan untuk berbagai platform aplikasi seperti web, phone, store atau aplikasi hybrid.

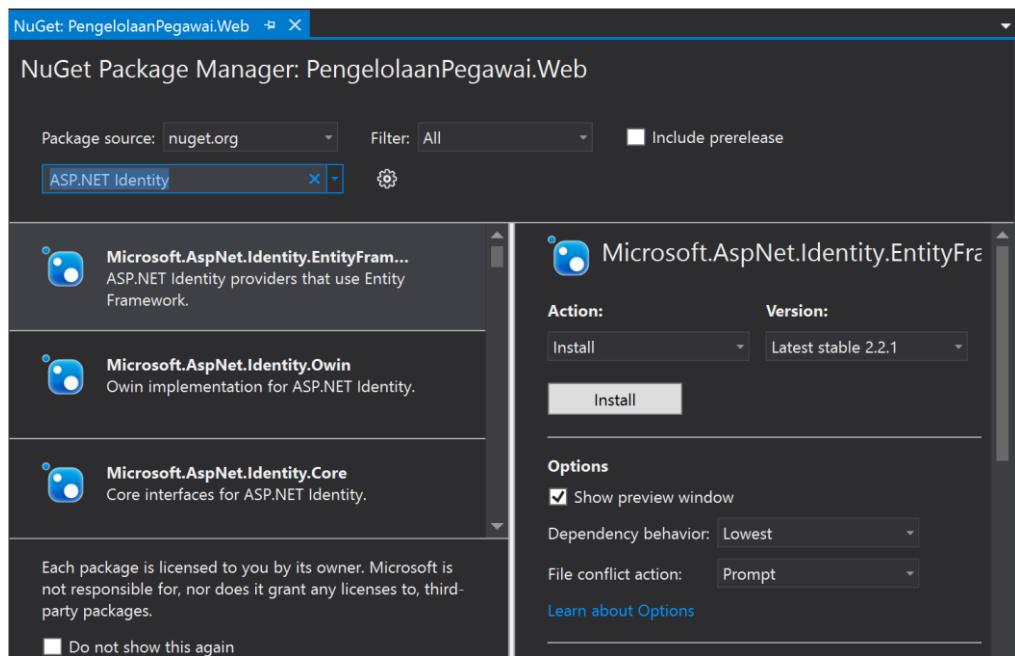
Untuk menggunakan ASP.NET Identity pada project yang telah dibuat di atas dapat dilakukan langkah-langkah berikut ini.



Gambar 118. Menambahkan ASP.NET Identity pada project via NuGet.

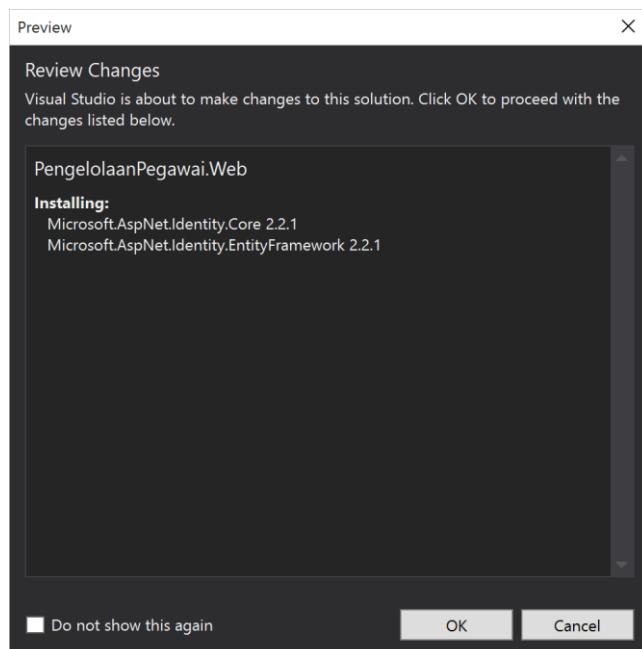
Klik kanan pada project PengelolaanPegawai.Web kemudian pilih Manage NuGet Packages. Pada window NuGet Package Manager ketika kata kunci ASP.NET Identity maka dapat dilihat tiga komponen utama ASP.NET Identity yaitu :

1. Microsoft.AspNet.Identity.EntityFramework.
2. Microsoft.AspNet.Identity.Core .
3. Microsoft.AspNet.Identity.OWIN.



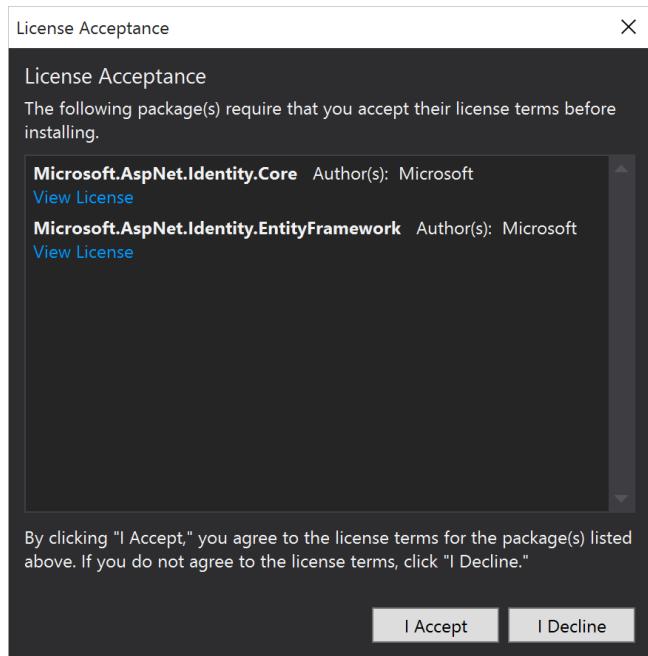
Gambar 119. Tiga komponen utama ASP.NET Identity.

Install ketiga komponen tersebut dengan memilih satu-satu dari ketiga komponen tersebut dan kemudian klik tombol Install. Sebagai contoh klik Microsoft.AspNet.Identity.EntityFramework kemudian klik Install, maka akan ditampilkan window dialog seperti berikut, kemudian klik tombol OK.



Gambar 120. Window preview.

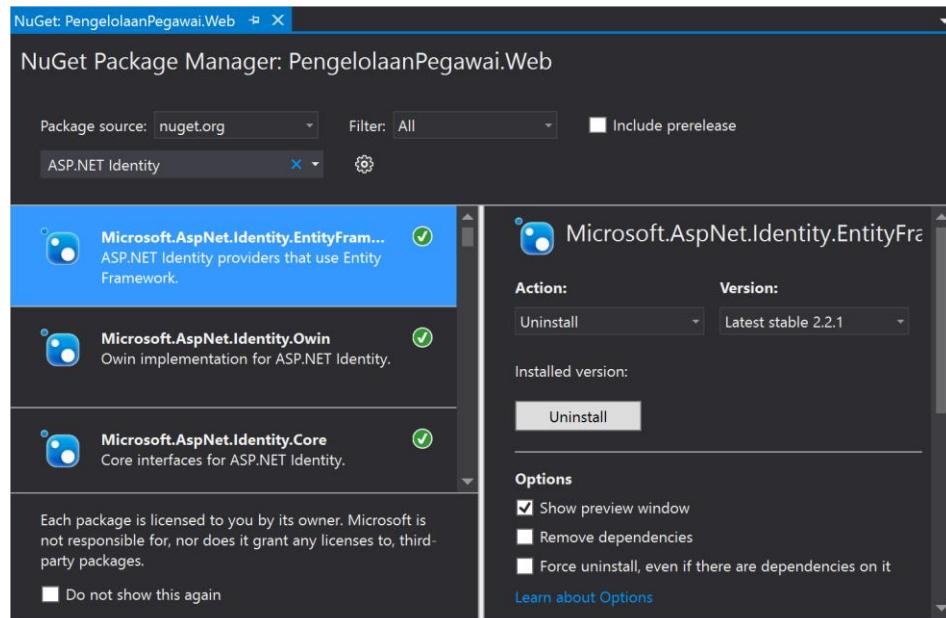
Kemudian akan ditampilkan window seperti berikut ini.



Gambar 121. Window License Acceptance.

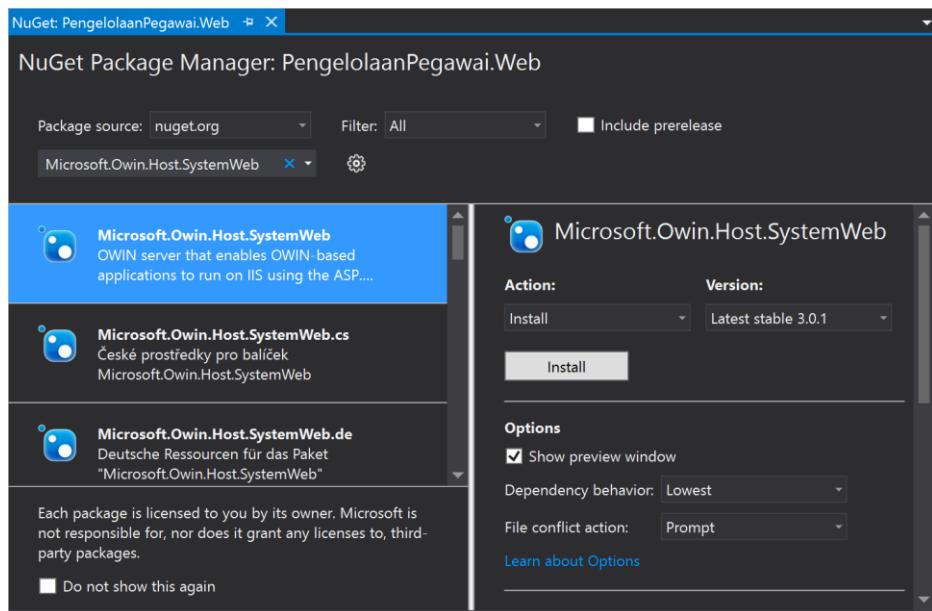
Selanjutnya komponen yang diperlukan akan diunduh, sehingga harus dipastikan tersedia koneksi internet saat melakukan proses ini.

Jika proses installasi ketiga komponen tersebut telah selesai dilakukan maka setiap komponen akan diberikan status berupa gambar centang seperti gambar di bawah ini.



Gambar 122. Status instalasi komponen ASP.NET Identity.

Komponen selanjutnya yang perlu diinstall adalah Microsoft.Owin.Host.SystemWeb seperti yang dapat dilihat pada gambar di bawah ini.



Gambar 123. Komponen Microsoft.Owin.Host.SystemWeb.

Otentikasi

Salah satu pengamanan yang umum digunakan adalah dengan melindungi halaman dengan pemeriksaan otentikasi dengan terlebih dahulu dengan melewati halaman login. Halaman login biasanya mengharuskan pengguna untuk memasukkan username dan password. Kemudian proses selanjutnya adalah mencocokan nilai username dan password yang dimasukkan dengan data yang terdapat di dalam database.

Dari penjelasan singkat di atas maka pada bagian ini akan dilakukan tahap per tahap untuk membuat halaman login.

Persiapan Database

Persiapan database diperlukan karena data user akan disimpan di sini. Langkah yang harus dilakukan adalah menentukan database yang akan digunakan dengan membuat connection string untuk mengakses database tersebut. Pada project PengelolaanPegawai.Web, database yang digunakan adalah PegawaiDB.mdf yang terdapat pada folder App_Data. Jika ingin menggunakan database tersebut sebagai tempat penyimpanan data user maka perlu ditambahkan baris berikut pada bagian `<connectionStrings>`.

```
<add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\PegawaiDB.mdf
;Initial Catalog=PegawaiDB;Integrated Security=True"
providerName="System.Data.SqlClient" />
```

Sehingga secara lengkap akan dilihat kode pada bagian `<connectionStrings>` secara lengkap seperti berikut ini.

```
<connectionStrings>
<add name="PegawaiDBEntities"
connectionString="metadata=res://*/Models.PegawaiModel.csdl|res://*/Models.P
egawaiModel.ssdl|res://*/Models.PegawaiModel.msl;provider=System.Data.SqlClient" />
```

```

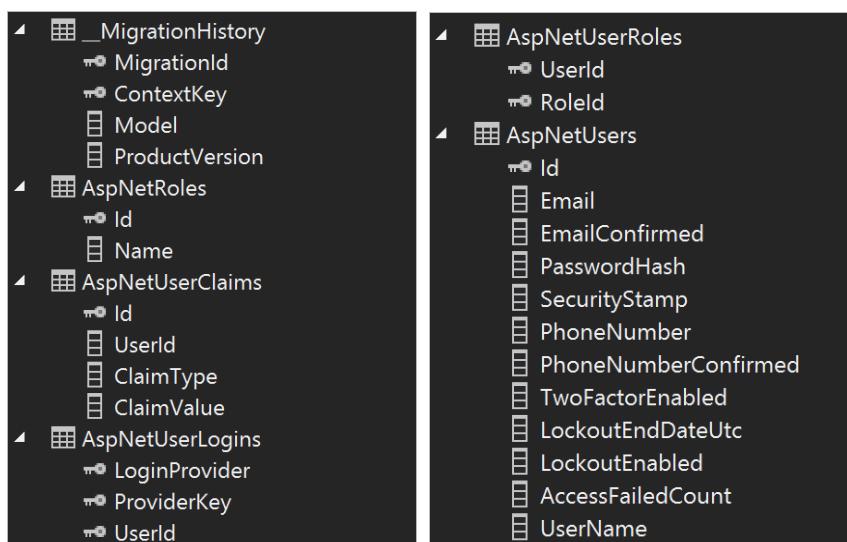
<connectionStrings>
  <add name="DefaultConnection" connectionString="data
source=(LocalDB)\MSSQLLocalDB;attachdbfilename=|DataDirectory|\PegawaiDB.mdf
;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework";
providerName="System.Data.EntityClient" />

  <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\PegawaiDB.mdf
;Initial Catalog=PegawaiDB;Integrated Security=True"
providerName="System.Data.SqlClient" />
</connectionStrings>

```

Pada persiapan ini tidak perlu dilakukan pembuatan tabel untuk menyimpan data. Tabel-tabel yang diperlukan untuk menyimpan data pengguna akan dilakukan secara otomatis. Hal ini mungkin terjadi karena memanfaatkan Entity Framework Code First.

Berikut ini adalah tabel-tabel default yang akan dibuat untuk penyimpanan data pengguna.



Gambar 124. Tabel-tabel hasil generasi otomatis dari Entity Framework Code First.

Dari gambar tersebut dapat dilihat terdapat beberapa tabel yaitu :

1. `_MigrationHistory`.
2. `AspNetRoles`.
3. `AspNetUserClaims`.
4. `AspNetUserLogins`.
5. `AspNetUserRoles`.
6. `AspNetUsers`.

Model

Pembuatan tabel secara otomatis pada implementasi Entity Framework Code First dimungkinkan karena terlebih dahulu dibuat class model. Berikut ini ada beberapa model yang dibuat, salah satunya adalah membuat class `IdentityModel.cs` dengan isi seperti berikut ini.

IdentityModels.cs

```

IdentityModels.cs
using System.Data.Entity;
using System.Security.Claims;
using System.Threading.Tasks;

```

```

using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace PengelolaanPegawai.Web.Models
{
    public class ApplicationUser : IdentityUser
    {
        // Profile data tambahan dapat dilakukan dengan menambahkan property pada class ApplicationUser
        public string NIP { get; set; }
        public string Alamat { get; set; }

        public async Task<ClaimsIdentity>
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            // Note the authenticationType must match the one defined in
CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            return userIdentity;
        }
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }
}

```

Pada file IdentityModels.cs di atas berisi dua class yaitu :

1. ApplicationUser, dapat berfungsi untuk menentukan data profile user. Secara default profile user memiliki property seperti yang dapat dilihat pada tabel AspNetUser pada Gambar 124.
2. ApplicationDbContext, berfungsi untuk menyatakan cara koneksi ke database dengan cara memilih name dari connection string yang digunakan yaitu DefaultConnection.

AccountViewModel.cs

Selanjutnya perlu ditambah model pendukung untuk kebutuhan pembuatan form atau sebagai model untuk pertukaran data antara komponen controller dan view. Berikut ini adalah model-model yang berfungsi untuk register user dan login.

```

AccountViewModels.cs
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace PengelolaanPegawai.Web.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
    }
}

```

```

        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }

        [Display(Name = "Remember this browser?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }

    public class LoginViewModel
    {
        [Required]
        [Display(Name = "Username")]
        public string Username { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [Display(Name = "User Name")]
        public string UserName { get; set; }
    }

```

```

[Required]
[Display(Name = "NIP")]
public string NIP { get; set; }

[Required]
[EmailAddress]
[Display(Name = "Email")]
public string Email { get; set; }

[Display(Name = "Alamat")]
public string Alamat { get; set; }

[Required]
[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
[DataType(DataType.Password)]
[Display(Name = "Password")]
public string Password { get; set; }

[DataType(DataType.Password)]
[Display(Name = "Confirm password")]
[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
public string ConfirmPassword { get; set; }
}

public class UserViewModel
{
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Display(Name = "NIP")]
    public string NIP { get; set; }

    [Display(Name = "Email")]
    public string Email { get; set; }

    [Display(Name = "Alamat")]
    public string Alamat { get; set; }

    [Display(Name = "Role")]
    public string Roles { get; set; }
}

public class UserFormViewModel
{
    [Required]
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Required]
    [Display(Name = "NIP")]
    public string NIP { get; set; }

    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }
}

```

```

[Display(Name = "Alamat")]
public string Alamat { get; set; }

[Required]
[Display(Name = "Role")]
public string Role { get; set; }

[Required]
[StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
[DataType(DataType.Password)]
[Display(Name = "Password")]
public string Password { get; set; }

[DataType(DataType.Password)]
[Display(Name = "Confirm password")]
[Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
public string ConfirmPassword { get; set; }

public IEnumerable<System.Web.Mvc.SelectListItem> Roles { get; set;
}

public class ResetPasswordViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
    public string ConfirmPassword { get; set; }

    public string Code { get; set; }
}

public class ForgotPasswordViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }
}

```

ManageViewModel.cs

Selanjutnya perlu dibuat model yang berfungsi untuk membuat fitur yg dapat digunakan pengguna yang telah terotentikasi untuk mengelola akunnya sendiri, seperti mengganti password milik pengguna.

```

ManageViewModels.cs
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;

namespace PengelolaanPegawai.Web.Models
{
    public class IndexViewModel
    {
        public bool HasPassword { get; set; }
        public IList<UserLoginInfo> Logins { get; set; }
        public string PhoneNumber { get; set; }
        public bool TwoFactor { get; set; }
        public bool BrowserRemembered { get; set; }
    }

    public class ManageLoginsViewModel
    {
        public IList<UserLoginInfo> CurrentLogins { get; set; }
        public IList<AuthenticationDescription> OtherLogins { get; set; }
    }

    public class FactorViewModel
    {
        public string Purpose { get; set; }
    }

    public class SetPasswordViewModel
    {
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ChangePasswordViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
    }
}

```

```

        [Compare("NewPassword", ErrorMessage = "The new password and
confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class AddPhoneNumberViewModel
    {
        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string Number { get; set; }
    }

    public class VerifyPhoneNumberViewModel
    {
        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }

        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string PhoneNumber { get; set; }
    }

    public class ConfigureTwoFactorViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get;
set; }
    }
}

```

Controller

Pada bagian ini akan dibuat beberapa class controller yang mendukung pengelolaan pengguna/user dan juga pengaturan keamanan untuk login.

AccountController.cs

Selanjutnya akan dibuat controller yang berisi method-method aksi untuk mengelola akun pengguna yang disimpan pada class AccountController.cs. Berikut ini adalah isi dari class tersebut.

```

AccountController.cs
using System;
using System.Globalization;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;

using PengelolaanPegawai.Web.Models;
using System.Collections.Generic;
using System.Net;
using Microsoft.AspNet.Identity.EntityFramework;
using System.Data.Entity;

```

```

namespace PengelolaanPegawai.Web.Controllers
{
    [Authorize]
    public class AccountController : Controller
    {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;

        public AccountController()
        {

        }

        public AccountController(ApplicationUserManager userManager,
ApplicationSignInManager signInManager)
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationSignInManager SignInManager
        {
            get
            {
                return _signInManager ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        // GET: Roles
        public ActionResult Index()
        {
            var users = UserManager.Users.Select(p => p).ToList();
            List<UserViewModel> items = new List<UserViewModel>();
            foreach (var user in users)
            {
                UserViewModel item = new UserViewModel();
                item.UserName = user.UserName;
                item.Alamat = user.Alamat;
                item.NIP = user.NIP;
                item.Email = user.Email;
            }
        }
    }
}

```

```

        var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
        if (String.IsNullOrEmpty(roleForUser))
{
    roleForUser = "-";
}
item.Roles = roleForUser;

        items.Add(item);
    }
    return View(items);
}

// GET: Divisis/Details/5
public ActionResult Details(string id)
{
    if (String.IsNullOrEmpty(id))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    var user = UserManager.FindByName(id);
    var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
    if (String.IsNullOrEmpty(roleForUser))
    {
        roleForUser = "-";
    }

    UserViewModel model = new UserViewModel();
    model.UserName = user.UserName;
    model.NIP = user.NIP;
    model.Email = user.Email;
    model.Alamat = user.Alamat;
    model.Roles = roleForUser;

    return View(model);
}

// GET: Divisis/Create
public ActionResult Create()
{
    ApplicationDbContext context = new ApplicationDbContext();

    var model = new UserFormViewModel
    {
        Roles = context.Roles.ToList().Select(x => new
SelectListItem
        {
            Value = x.Name,
            Text = x.Name
        })
    };

    return View(model);
}

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Create(UserFormViewModel model)
{

```

```

        ApplicationDbContext context = new ApplicationDbContext();

        model.Roles = context.Roles.ToList().Select(x => new
SelectListItem
{
    Value = x.Name,
    Text = x.Name
});

        if (ModelState.IsValid)
        {
            var user = new ApplicationUser { UserName = model.UserName,
Email = model.Email, NIP = model.NIP, Alamat = model.Alamat };
            var result = await UserManager.CreateAsync(user,
model.Password);

            if (result.Succeeded)
            {
                await UserManager.AddToRoleAsync(user.Id, model.Role);
                return RedirectToAction("Index");
            }
        }

        return View(model);
    }

// GET: Account/Edit/5
public ActionResult Edit(string id)
{
    if (String.IsNullOrEmpty(id))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    ApplicationDbContext context = new ApplicationDbContext();
    var user = UserManager.FindByName(id);
    var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
    if (String.IsNullOrEmpty(roleForUser))
    {
        roleForUser = "-";
    }

    UserFormViewModel model = new UserFormViewModel();
    model.UserName = user.UserName;
    model.NIP = user.NIP;
    model.Email = user.Email;
    model.Alamat = user.Alamat;
    model.Role = roleForUser;
    model.Password = user.PasswordHash;
    model.ConfirmPassword = user.PasswordHash;

    model.Roles = context.Roles.ToList().Select(x => new
SelectListItem
{
    Value = x.Name,
    Text = x.Name,
    Selected = x.Name == model.Role
});
}

```

```

        return View(model);
    }

    // POST: Account/Edit/5
    // To protect from overposting attacks, please enable the specific
    // properties you want to bind to, for
    // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Edit(UserFormViewModel model)
    {
        ApplicationDbContext context = new ApplicationDbContext();

        model.Roles = context.Roles.ToList().Select(x => new
SelectListItem
        {
            Value = x.Name,
            Text = x.Name
        });

        if (ModelState.IsValid)
        {
            var user = UserManager.FindByName(model.UserName);

            var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
            if (String.IsNullOrEmpty(roleForUser))
            {
                roleForUser = "-";
            }

            user.NIP = model.NIP;
            user.UserName = model.UserName;
            user.Alamat = model.Alamat;
            user.Email = model.Email;

            if (!String.IsNullOrEmpty(model.Password) &&
!String.IsNullOrEmpty(model.ConfirmPassword))
            {

            }
            else
            {
                user.PasswordHash = model.Password;
            }

            var result = UserManager.Update(user);
            if (result.Succeeded)
            {
                await UserManager.RemoveFromRoleAsync(user.Id,
roleForUser);
                await UserManager.AddToRoleAsync(user.Id, model.Role);
                return RedirectToAction("Index");
            }
        }

        return View(model);
    }

    // GET: Divisis/Delete/5
    public ActionResult Delete(string id)

```

```

    {
        if (String.IsNullOrEmpty(id))
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }

        //var user = context.Users.FirstOrDefault(u =>
        u.UserName.Equals(id));
        var user = UserManager.FindByName(id);
        var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
        if (String.IsNullOrEmpty(roleForUser))
        {
            roleForUser = "-";
        }

        UserViewModel model = new UserViewModel();
        model.UserName = user.UserName;
        model.NIP = user.NIP;
        model.Email = user.Email;
        model.Alamat = user.Alamat;
        model.Roles = roleForUser;

        return View(model);
    }

    // POST: Divisis/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(string id)
    {
        var user = UserManager.FindByName(id);
        if (user != null)
        {
            UserManager.Delete(user);
        }

        return RedirectToAction("Index");
    }

    //
    // GET: /Account/Login
    [AllowAnonymous]
    public ActionResult Login(string returnUrl)
    {
        ViewBag.ReturnUrl = returnUrl;
        return View();
    }

    //
    // POST: /Account/Login
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Login(LoginViewModel model, string
returnUrl)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }
    }
}

```

```

        // This doesn't count login failures towards account lockout
        // To enable password failures to trigger account lockout,
change to shouldLockout: true
        var result = await
SignInManager.PasswordSignInAsync(model.Username, model.Password,
model.RememberMe, shouldLockout: false);
        switch (result)
        {
            case SignInStatus.Success:
                return RedirectToAction(returnUrl);
            case SignInStatus.LockedOut:
                return View("Lockout");
            case SignInStatus.RequiresVerification:
                return RedirectToAction("SendCode", new { ReturnUrl =
returnUrl, RememberMe = model.RememberMe });
            case SignInStatus.Failure:
            default:
                ModelState.AddModelError("", "Invalid login attempt.");
                return View(model);
        }
    }

    //
    // GET: /Account/VerifyCode
    [AllowAnonymous]
    public async Task<ActionResult> VerifyCode(string provider, string
returnUrl, bool rememberMe)
{
    // Require that the user has already logged in via
username/password or external login
    if (!await SignInManager.Has Been Verified Async())
    {
        return View("Error");
    }
    return View(new VerifyCodeViewModel { Provider = provider,
ReturnUrl = returnUrl, RememberMe = rememberMe });
}

    //
    // POST: /Account/VerifyCode
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> VerifyCode(VerifyCodeViewModel
model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

        // The following code protects for brute force attacks against
the two factor codes.
        // If a user enters incorrect codes for a specified amount of
time then the user account
        // will be locked out for a specified amount of time.
        // You can configure the account lockout settings in
IdentityConfig
}

```

```

        var result = await
SignInManager.TwoFactorSignInAsync(model.Provider, model.Code, isPersistent:
model.RememberMe, rememberBrowser: model.RememberBrowser);
        switch (result)
    {
        case SignInStatus.Success:
            return RedirectToLocal(model.ReturnUrl);
        case SignInStatus.LockedOut:
            return View("Lockout");
        case SignInStatus.Failure:
        default:
            ModelState.AddModelError("", "Invalid code.");
            return View(model);
    }
}

// 
// GET: /Account/Register
[AllowAnonymous]
public ActionResult Register()
{
    return View();
}

// 
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.UserName,
Email = model.Email, NIP = model.NIP, Alamat = model.Alamat };
        var result = await UserManager.CreateAsync(user,
model.Password);
        if (result.Succeeded)
        {
            await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);

            // For more information on how to enable account
confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkId=320771
                // Send an email with this link
                // string code = await
UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
                // var callbackUrl = Url.Action("ConfirmEmail",
"Account", new { userId = user.Id, code = code }, protocol:
Request.Url.Scheme);
                // await UserManager.SendEmailAsync(user.Id, "Confirm
your account", "Please confirm your account by clicking <a href=\"" +
callbackUrl + "\">here</a>");

            return RedirectToAction("Register", "Account");
        }
        AddErrors(result);
    }
}

// If we got this far, something failed, redisplay form

```

```

        return View(model);
    }

    //
    // GET: /Account/ConfirmEmail
    [AllowAnonymous]
    public async Task<ActionResult> ConfirmEmail(string userId, string
code)
    {
        if (userId == null || code == null)
        {
            return View("Error");
        }
        var result = await UserManager.ConfirmEmailAsync(userId, code);
        return View(result.Succeeded ? "ConfirmEmail" : "Error");
    }

    //
    // GET: /Account/ForgotPassword
    [AllowAnonymous]
    public ActionResult ForgotPassword()
    {
        return View();
    }

    //
    // POST: /Account/ForgotPassword
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult>
ForgotPassword(ForgotPasswordViewModel model)
    {
        if (ModelState.IsValid)
        {
            var user = await UserManager.FindByNameAsync(model.Email);
            if (user == null || !(await
UserManager.IsEmailConfirmedAsync(user.Id)))
            {
                // Don't reveal that the user does not exist or is not
confirmed
                return View("ForgotPasswordConfirmation");
            }

            // For more information on how to enable account
confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkId=320771
                // Send an email with this link
                // string code = await
UserManager.GeneratePasswordResetTokenAsync(user.Id);
                // var callbackUrl = Url.Action("ResetPassword", "Account",
new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

                // await UserManager.SendEmailAsync(user.Id, "Reset
Password", "Please reset your password by clicking <a href=\"" + callbackUrl
+ "\">here</a>");
                // return RedirectToAction("ForgotPasswordConfirmation",
"Account");
            }
        }

        // If we got this far, something failed, redisplay form
    }
}

```

```

        return View(model);
    }

    //
    // GET: /Account/ForgotPasswordConfirmation
    [AllowAnonymous]
    public ActionResult ForgotPasswordConfirmation()
    {
        return View();
    }

    //
    // GET: /Account/ResetPassword
    [AllowAnonymous]
    public ActionResult ResetPassword(string code)
    {
        return code == null ? View("Error") : View();
    }

    //
    // POST: /Account/ResetPassword
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> ResetPassword(ResetPasswordViewModel
model)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }
        var user = await UserManager.FindByNameAsync(model.Email);
        if (user == null)
        {
            // Don't reveal that the user does not exist
            return RedirectToAction("ResetPasswordConfirmation",
"Account");
        }
        var result = await UserManager.ResetPasswordAsync(user.Id,
model.Code, model.Password);
        if (result.Succeeded)
        {
            return RedirectToAction("ResetPasswordConfirmation",
"Account");
        }
        AddErrors(result);
        return View();
    }

    //
    // GET: /Account/ResetPasswordConfirmation
    [AllowAnonymous]
    public ActionResult ResetPasswordConfirmation()
    {
        return View();
    }

    //
    // POST: /Account/ExternalLogin
    [HttpPost]
    [AllowAnonymous]

```

```

[ValidateAntiForgeryToken]
public ActionResult ExternalLogin(string provider, string returnUrl)
{
    // Request a redirect to the external login provider
    return new ChallengeResult(provider,
Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl }));
}

//
// GET: /Account/SendCode
[AllowAnonymous]
public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)
{
    var userId = await SignInManager.GetVerifiedUserIdAsync();
    if (userId == null)
    {
        return View("Error");
    }
    var userFactors = await
UserManager.GetValidTwoFactorProvidersAsync(userId);
    var factorOptions = userFactors.Select(purpose => new
SelectListItem { Text = purpose, Value = purpose }).ToList();
    return View(new SendCodeViewModel { Providers = factorOptions,
ReturnUrl = returnUrl, RememberMe = rememberMe });
}

//
// POST: /Account/SendCode
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> SendCode(SendCodeViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View();
    }

    // Generate the token and send it
    if (!await
SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider))
    {
        return View("Error");
    }
    return RedirectToAction("VerifyCode", new { Provider =
model.SelectedProvider, ReturnUrl = model.ReturnUrl, RememberMe =
model.RememberMe });
}

//
// GET: /Account/ExternalLoginCallback
[AllowAnonymous]
public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
{
    var loginInfo = await
AuthenticationManager.GetExternalLoginInfoAsync();
    if (loginInfo == null)
    {
}

```

```

        return RedirectToAction("Login");
    }

    // Sign in the user with this external login provider if the
    user already has a login
    var result = await SignInManager.ExternalSignInAsync(loginInfo,
isPersistent: false);
    switch (result)
    {
        case SignInStatus.Success:
            return RedirectToLocal(returnUrl);
        case SignInStatus.LockedOut:
            return View("Lockout");
        case SignInStatus.RequiresVerification:
            return RedirectToAction("SendCode", new { ReturnUrl =
returnUrl, RememberMe = false });
        case SignInStatus.Failure:
        default:
            // If the user does not have an account, then prompt the
            user to create an account
            ViewBag.ReturnUrl = returnUrl;
            ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
            return View("ExternalLoginConfirmation", new
ExternalLoginConfirmationViewModel { Email = loginInfo.Email });
    }
}

// 
// POST: /Account/ExternalLoginConfirmation
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult>
ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string
returnUrl)
{
    if (User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Index", "Manage");
    }

    if (ModelState.IsValid)
    {
        // Get the information about the user from the external
        login provider
        var info = await
AuthenticationManager.GetExternalLoginInfoAsync();
        if (info == null)
        {
            return View("ExternalLoginFailure");
        }
        var user = new ApplicationUser { UserName = model.Email,
Email = model.Email };
        var result = await UserManager.CreateAsync(user);
        if (result.Succeeded)
        {
            result = await UserManager.AddLoginAsync(user.Id,
info.Login);
            if (result.Succeeded)
            {

```

```

        await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                return RedirectToLocal(returnUrl);
            }
        }
        AddErrors(result);
    }

    ViewBag.ReturnUrl = returnUrl;
    return View(model);
}

// GET: /Account/LogOff
[HttpGet]
public ActionResult LogOff()
{
    AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);
    return RedirectToAction("Login", "Account");
}

// GET: /Account/ExternalLoginFailure
[AllowAnonymous]
public ActionResult ExternalLoginFailure()
{
    return View();
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        if (_userManager != null)
        {
            _userManager.Dispose();
            _userManager = null;
        }

        if (_signInManager != null)
        {
            _signInManager.Dispose();
            _signInManager = null;
        }
    }

    base.Dispose(disposing);
}

#region Helpers
// Used for XSRF protection when adding external logins
private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager
{
    get
    {
        return HttpContext.GetOwinContext().Authentication;
    }
}

```

```

private void AddErrors(IdentityResult result)
{
    foreach (var error in result.Errors)
    {
        ModelState.AddModelError("", error);
    }
}

private ActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    return RedirectToAction("Index", "Metro");
}

internal class ChallengeResult : HttpUnauthorizedResult
{
    public ChallengeResult(string provider, string redirectUri)
        : this(provider, redirectUri, null)
    {
    }

    public ChallengeResult(string provider, string redirectUri,
string userId)
    {
        LoginProvider = provider;
        RedirectUri = redirectUri;
        UserId = userId;
    }

    public string LoginProvider { get; set; }
    public string RedirectUri { get; set; }
    public string UserId { get; set; }

    public override void ExecuteResult(ControllerContext context)
    {
        var properties = new AuthenticationProperties { RedirectUri
= RedirectUri };
        if (UserId != null)
        {
            properties.Dictionary[XsrfKey] = UserId;
        }
    }
}

context.HttpContext.GetOwinContext().Authentication.Challenge(properties,
LoginProvider);
}
}
#endif
}
}

```

ManageController.cs

Controller yang lain berfungsi untuk menangani fitur pengelolaan akun oleh pengguna sendiri yaitu ManageController.cs dengan isi sebagai berikut.

ManageController.cs
<pre> using System; using System.Linq; using System.Threading.Tasks; </pre>

```

using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;

using PengelolaanPegawai.Web.Models;

namespace PengelolaanPegawai.Web.Controllers
{
    [Authorize]
    public class ManageController : Controller
    {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;

        public ManageController()
        {
        }

        public ManageController(ApplicationUserManager userManager,
ApplicationSignInManager signInManager)
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationSignInManager SignInManager
        {
            get
            {
                return _signInManager ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        //
        // GET: /Manage/Index
        public async Task<ActionResult> Index(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Your
password has been changed."

```

```

        : message == ManageMessageId.SetPasswordSuccess ? "Your
password has been set."
        : message == ManageMessageId.SetTwoFactorSuccess ? "Your
two-factor authentication provider has been set."
        : message == ManageMessageId.Error ? "An error has
occurred."
        : message == ManageMessageId.AddPhoneSuccess ? "Your phone
number was added."
        : message == ManageMessageId.RemovePhoneSuccess ? "Your
phone number was removed."
        : "";

        var userId = User.Identity.GetUserId();
        var model = new IndexViewModel
        {
            HasPassword = HasPassword(),
            PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),
            TwoFactor = await
UserManager.GetTwoFactorEnabledAsync(userId),
            Logins = await UserManager.GetLoginsAsync(userId),
            BrowserRemembered = await
AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)
        };
        return View(model);
    }

    //
    // POST: /Manage/RemoveLogin
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> RemoveLogin(string loginProvider,
string providerKey)
    {
        ManageMessageId? message;
        var result = await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new
UserLoginInfo(loginProvider, providerKey));
        if (result.Succeeded)
        {
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
            }
            message = ManageMessageId.RemoveLoginSuccess;
        }
        else
        {
            message = ManageMessageId.Error;
        }
        return RedirectToAction("ManageLogins", new { Message = message
});
    }

    //
    // GET: /Manage/AddPhoneNumber
    public ActionResult AddPhoneNumber()
    {
        return View();
    }
}

```

```

        }

        //
        // POST: /Manage/AddPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
AddPhoneNumber(AddPhoneNumberViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    // Generate the token and send it
    var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
model.Number);
    if (UserManager.SmsService != null)
    {
        var message = new IdentityMessage
        {
            Destination = model.Number,
            Body = "Your security code is: " + code
        };
        await UserManager.SmsService.SendAsync(message);
    }
    return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber =
model.Number });
}

//
// POST: /Manage/EnableTwoFactorAuthentication
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> EnableTwoFactorAuthentication()
{
    await
UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);
    var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
    if (user != null)
    {
        await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
    }
    return RedirectToAction("Index", "Manage");
}

//
// POST: /Manage/DisableTwoFactorAuthentication
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> DisableTwoFactorAuthentication()
{
    await
UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);
    var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
    if (user != null)
    {
}

```

```

        await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
    }
    return RedirectToAction("Index", "Manage");
}

// GET: /Manage/VerifyPhoneNumber
public async Task<ActionResult> VerifyPhoneNumber(string
phoneNumber)
{
    var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
phoneNumber);
    // Send an SMS through the SMS provider to verify the phone
number
    return phoneNumber == null ? View("Error") : View(new
VerifyPhoneNumberViewModel { PhoneNumber = phoneNumber });
}

// POST: /Manage/VerifyPhoneNumber
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult>
VerifyPhoneNumber(VerifyPhoneNumberViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    var result = await
UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(),
model.PhoneNumber, model.Code);
    if (result.Succeeded)
    {
        var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
        if (user != null)
        {
            await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
        }
        return RedirectToAction("Index", new { Message =
ManageMessageId.AddPhoneSuccess });
    }
    // If we got this far, something failed, redisplay form
    ModelState.AddModelError("", "Failed to verify phone");
    return View(model);
}

// GET: /Manage/RemovePhoneNumber
public async Task<ActionResult> RemovePhoneNumber()
{
    var result = await
UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);
    if (!result.Succeeded)
    {
        return RedirectToAction("Index", new { Message =
ManageMessageId.Error });
    }
}

```

```

        }
        var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
        if (user != null)
        {
            await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
        }
        return RedirectToAction("Index", new { Message =
ManageMessageId.RemovePhoneSuccess });
    }

    //
    // GET: /Manage/ChangePassword
    public ActionResult ChangePassword()
    {
        return View();
    }

    //
    // POST: /Manage/ChangePassword
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult>
ChangePassword(ChangePasswordViewModel model)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }
        var result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(),
model.OldPassword, model.NewPassword);
        if (result.Succeeded)
        {
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
            }
            return RedirectToAction("Index", new { Message =
ManageMessageId.ChangePasswordSuccess });
        }
        AddErrors(result);
        return View(model);
    }

    //
    // GET: /Manage/SetPassword
    public ActionResult SetPassword()
    {
        return View();
    }

    //
    // POST: /Manage/SetPassword
    [HttpPost]
    [ValidateAntiForgeryToken]

```

```

        public async Task<ActionResult> SetPassword(SetPasswordViewModel
model)
    {
        if (ModelState.IsValid)
        {
            var result = await
UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);
            if (result.Succeeded)
            {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message =
ManageMessageId.SetPasswordSuccess });
            }
            AddErrors(result);
        }

        // If we got this far, something failed, redisplay form
        return View(model);
    }

    //
    // GET: /Manage/ManageLogins
    public async Task<ActionResult> ManageLogins(ManageMessageId?
message)
    {
        ViewBag.StatusMessage =
            message == ManageMessageId.RemoveLoginSuccess ? "The
external login was removed."
            : message == ManageMessageId.Error ? "An error has
occurred."
            : "";
        var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
        if (user == null)
        {
            return View("Error");
        }
        var userLogins = await
UserManager.GetLoginsAsync(User.Identity.GetUserId());
        var otherLogins =
AuthenticationManager.GetExternalAuthenticationTypes().Where(auth =>
userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();
        ViewBag.ShowRemoveButton = user.PasswordHash != null ||
userLogins.Count > 1;
        return View(new ManageLoginsViewModel
{
    CurrentLogins = userLogins,
    OtherLogins = otherLogins
});
    }

    //
    // POST: /Manage/LinkLogin
    [HttpPost]
    [ValidateAntiForgeryToken]

```

```

        public ActionResult LinkLogin(string provider)
    {
        // Request a redirect to the external login provider to link a
        // login for the current user
        return new AccountController.ChallengeResult(provider,
Url.Action("LinkLoginCallback", "Manage"), User.Identity.GetUserId());
    }

    //
    // GET: /Manage/LinkLoginCallback
    public async Task<ActionResult> LinkLoginCallback()
    {
        var loginInfo = await
AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
        if (loginInfo == null)
        {
            return RedirectToAction("ManageLogins", new { Message =
ManageMessageId.Error });
        }
        var result = await
UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
        return result.Succeeded ? RedirectToAction("ManageLogins") :
RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing && _userManager != null)
        {
            _userManager.Dispose();
            _userManager = null;
        }

        base.Dispose(disposing);
    }

    #region Helpers
    // Used for XSRF protection when adding external logins
    private const string XsrfKey = "XsrfId";

    private IAuthenticationManager AuthenticationManager
    {
        get
        {
            return HttpContext.GetOwinContext().Authentication;
        }
    }

    private void AddErrors(IdentityResult result)
    {
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError("", error);
        }
    }

    private bool HasPassword()
    {
        var user = UserManager.FindById(User.Identity.GetUserId());
        if (user != null)

```

```

        {
            return user.PasswordHash != null;
        }
        return false;
    }

    private bool HasPhoneNumber()
    {
        var user = UserManager.FindById(User.Identity.GetUserId());
        if (user != null)
        {
            return user.PhoneNumber != null;
        }
        return false;
    }

    public enum ManageMessageId
    {
        AddPhoneSuccess,
        ChangePasswordSuccess,
        SetTwoFactorSuccess,
        SetPasswordSuccess,
        RemoveLoginSuccess,
        RemovePhoneSuccess,
        Error
    }

    #endregion
}
}

```

View

Selanjutnya dibuat view yang digunakan oleh AccountController. Jika dilihat dari method aksi yang terdapat pada controller tersebut maka dapat diketahui perlu dibuat beberapa file view sebagai berikut :

1. ConfirmEmail.cshtml.
2. ExternalLoginConfirmation.cshtml.
3. ExternalLoginFailure.cshtml.
4. ForgotPassword.cshtml.
5. ForgotPasswordConfirmation.cshtml.
6. Login.cshtml.
7. Register.cshtml.
8. ResetPassword.cshtml.
9. ResetPasswordConfirmation.cshtml.
10. SendCode.cshtml.
11. VerifyCode.cshtml.

Register.cshtml

Untuk percobaan pada ebook ini akan dijelaskan view Register.cshtml yang berfungsi untuk membuat user dengan form seperti pada gambar di bawah ini.

Gambar 125. Form registrasi pengguna/user.

Form ini mungkin dibuat karena menggunakan model dari class RegisterViewModel yang terdapat pada file AccountViewModel.cs. Berikut ini adalah kutipan dari class RegisterViewModel.

```
public class RegisterViewModel
{
    [Required]
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Required]
    [Display(Name = "NIP")]
    public string NIP { get; set; }

    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Display(Name = "Alamat")]
    public string Alamat { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2}
    characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation
    password do not match.")]
    public string ConfirmPassword { get; set; }
}
```

Dan berikut ini adalah isi dari file view Register.cshtml yang hasilnya telah dilihat pada gambar di atas.

```
Register.cshtml
@model PengelolaanPegawai.Web.Models.RegisterViewModel

@{
    ViewBag.Title = "Manajemen User";
```

```

}

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i><@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        <div class="box-content">
            @using (Html.BeginForm("Register", "Account", FormMethod.Post,
new { @class = "form-horizontal", role = "form" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    @Html.ValidationSummary("", new { @class = "text-danger"
})
                <div class="control-group">
                    @Html.LabelFor(m => m.UserName, htmlAttributes: new
{ @class = "control-label" })
                    <div class="controls">
                        @Html.TextBoxFor(m => m.UserName, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
                </div>
                <div class="control-group">
                    @Html.LabelFor(m => m.NIP, htmlAttributes: new {
@class = "control-label" })
                    <div class="controls">
                        @Html.TextBoxFor(m => m.NIP, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
                </div>
                <div class="control-group">
                    @Html.LabelFor(m => m.Email, htmlAttributes: new {
@class = "control-label" })
                    <div class="controls">
                        @Html.TextBoxFor(m => m.Email, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
                </div>
                <div class="control-group">
                    @Html.LabelFor(m => m.Alamat, htmlAttributes: new {
@class = "control-label" })
                    <div class="controls">
                        @Html.TextAreaFor(m => m.Alamat, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
                </div>
                <div class="control-group">
                    @Html.LabelFor(m => m.Password, htmlAttributes: new
{ @class = "control-label" })

```

```

        <div class="controls">
            @Html.PasswordFor(m => m.Password, new {
                htmlAttributes = new { @class = "span6 typeahead" } })
        </div>
    </div>
    <div class="control-group">
        @Html.LabelFor(m => m.ConfirmPassword,
        htmlAttributes: new { @class = "control-label" })
        <div class="controls">
            @Html.PasswordFor(m => m.ConfirmPassword, new {
                htmlAttributes = new { @class = "span6 typeahead" } })
        </div>
    </div>

    <div class="form-actions">
        <button type="submit" class="btn btn-primary">Register</button>
    </div>
    </fieldset>
}
</div>
</div>

```

Pada halaman view tersebut dapat dilihat yang menangani aksi setelah tombol Register diklik adalah method aksi Register yang terdapat pada controller Account. Berikut ini adalah kode dari method aksi tersebut.

```

// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.UserName, Email =
model.Email, NIP = model.NIP, Alamat = model.Alamat };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);

            return RedirectToAction("Register", "Account");
        }
        AddErrors(result);
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}

```

Pada kode di atas dapat dilihat isian dari form disiapkan untuk disimpan dengan cara baris seperti berikut.

```

var user = new ApplicationUser { UserName = model.UserName, Email =
model.Email, NIP = model.NIP, Alamat = model.Alamat };

```

Kemudian persiapan penyimpanan data user dan password dengan cara seperti berikut.

```

var result = await UserManager.CreateAsync(user, model.Password);

```

Login.cshtml

Sedangkan untuk form login dibuat file Login.cshtml dengan isi sebagai berikut.

```
 Login.cshtml
 @using PengelolaanPegawai.Web.Models
 @model LoginViewModel
 @{
     Layout = null;
 }

 <!DOCTYPE html>
 <html lang="en">
 <head>

     <!-- start: Meta -->
     <meta charset="utf-8">
     <title>Login - Pengelolaan Pegawai</title>
     <meta name="description" content="Login - Pengelolaan Pegawai">
     <meta name="author" content="Dennis Ji">
     <meta name="keyword" content="Pengelolaan Pegawai">
     <!-- end: Meta -->
     <!-- start: Mobile Specific -->
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <!-- end: Mobile Specific -->
     <!-- start: CSS -->
     <link id="bootstrap-style" href("~/css/bootstrap.min.css"
rel="stylesheet">
     <link href("~/css/bootstrap-responsive.min.css" rel="stylesheet">
     <link id="base-style" href "~/css/style.css" rel="stylesheet">
     <link id="base-style-responsive" href "~/css/style-responsive.css"
rel="stylesheet">
     <link
href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,6
00italic,700italic,800italic,400,300,600,700,800&subset=latin,cyrillic-
ext,latin-ext' rel='stylesheet' type='text/css'>
     <!-- end: CSS -->
     <!-- The HTML5 shim, for IE6-8 support of HTML5 elements -->
     <!--[if lt IE 9]>
         <script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
         <link id="ie-style" href "~/css/ie.css" rel="stylesheet">
     <![endif]-->
     <!--[if IE 9]>
         <link id="ie9style" href "~/css/ie9.css" rel="stylesheet">
     <![endif]-->
     <!-- start: Favicon -->
     <link rel="shortcut icon" href "~/img/favicon.ico">
     <!-- end: Favicon -->

     <style type="text/css">
         body {
             background: url(../../../img/bg-login.jpg) !important;
         }
     </style>

 </head>
 <body>
     <div class="container-fluid-full">
         <div class="row-fluid">
```

```

        <div class="row-fluid">
            <div class="login-box">
                <div class="icons">
                </div>
                <h2>Sistem Pengelolaan Pegawai</h2>
                @using (Html.BeginForm("Login", "Account", new {
                    ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
                {
                    @Html.AntiForgeryToken()
                    <hr />
                    @Html.ValidationSummary(true, "", new { @class =
                    "text-danger" })
                    <fieldset>
                        <div class="input-prepend" title="Username">
                            <span class="add-on"><i class="halflings-icon user"></i></span>
                            @Html.TextBoxFor(m => m.Username, new {
                                @class = "input-large span10" })
                            @Html.ValidationMessageFor(m => m.Username,
                            "", new { @class = "text-danger" })
                        </div>
                        <div class="clearfix"></div>
                        <div class="input-prepend" title="Password">
                            <span class="add-on"><i class="halflings-icon lock"></i></span>
                            @Html.PasswordFor(m => m.Password, new {
                                @class = "input-large span10" })
                            @Html.ValidationMessageFor(m => m.Password,
                            "", new { @class = "text-danger" })
                        </div>
                        <div class="clearfix"></div>
                        <label class="remember" for="remember">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </label>
                        <div class="button-login">
                            <button type="submit" class="btn btn-primary">Login</button>
                        </div>
                        <div class="clearfix"></div>
                    </fieldset>
                }
                <hr>
            </div><!--/span-->
        </div><!--/row-->

    </div><!--/.fluid-container-->

</div><!--/fluid-row-->

<script src="~/js/jquery-1.9.1.min.js"></script>
<script src="~/js/jquery-migrate-1.0.0.min.js"></script>

<script src="~/js/jquery-ui-1.10.0.custom.min.js"></script>

<script src="~/js/jquery.ui.touch-punch.js"></script>

<script src="~/js/modernizr.js"></script>

```

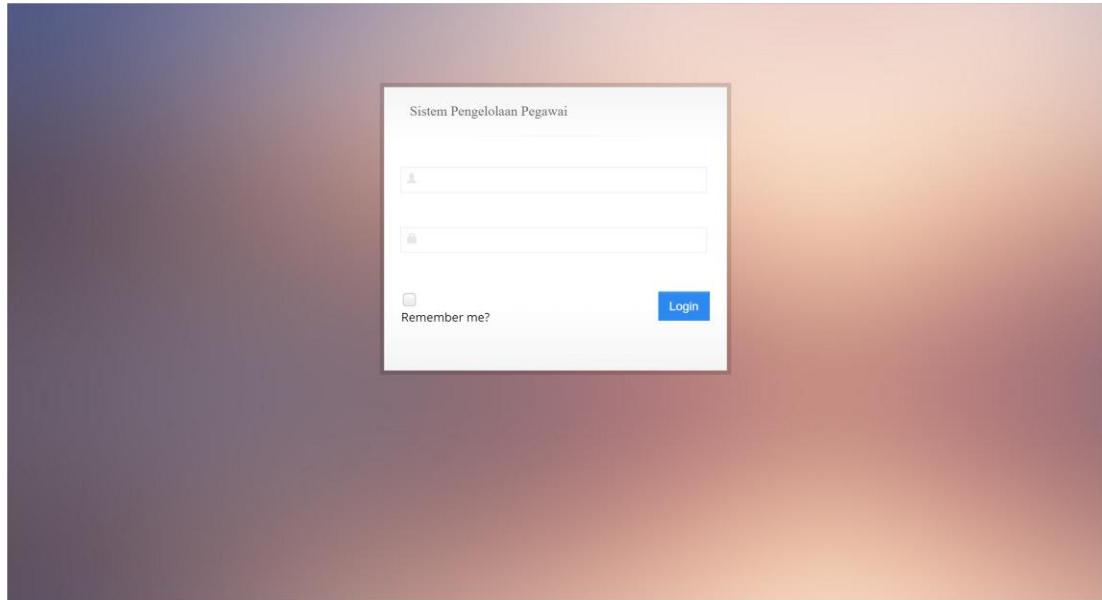
```

<script src="~/js/bootstrap.min.js"></script>
<script src="~/js/jquery.cookie.js"></script>
<script src='~/js/fullcalendar.min.js'></script>
<script src='~/js/jquery.dataTables.min.js'></script>
<script src="~/js/excanvas.js"></script>
<script src="~/js/jquery.flot.js"></script>
<script src="~/js/jquery.flot.pie.js"></script>
<script src="~/js/jquery.flot.stack.js"></script>
<script src="~/js/jquery.flot.resize.min.js"></script>
<script src="~/js/jquery.chosen.min.js"></script>
<script src="~/js/jquery.uniform.min.js"></script>
<script src="~/js/jquery.cleditor.min.js"></script>
<script src="~/js/jquery.noty.js"></script>
<script src="~/js/jquery.elfinder.min.js"></script>
<script src="~/js/jquery.raty.min.js"></script>
<script src="~/js/jquery.iphone.toggle.js"></script>
<script src="~/js/jquery.uploadify-3.1.min.js"></script>
<script src="~/js/jquery.gritter.min.js"></script>
<script src="~/js/jquery.imagesloaded.js"></script>
<script src="~/js/jquery.masonry.min.js"></script>
<script src="~/js/jquery.knob.modified.js"></script>
<script src="~/js/jquery.sparkline.min.js"></script>
<script src="~/js/counter.js"></script>
<script src="~/js/retina.js"></script>
<script src="~/js/custom.js"></script>
<!-- end: JavaScript-->

</body>
</html>

```

Dengan tampilan seperti berikut.



Gambar 126. Form login.

MasterLayout.cshtml

Selanjutnya perlu dilakukan penambahan sedikit kode untuk mengamankan halaman-halaman yang menggunakan layout dari file MasterLayout.cshtml agar hanya bisa diakses oleh user yang telah login. Berikut ini adalah kode yang ditambahkan pada halaman ini.

```
@if (!Request.IsAuthenticated)
{
    Response.Redirect("~/Account/Login");
}
```

Selain itu juga dilakukan modifikasi untuk membuat tombol logout. Tombol ini dapat digunakan oleh user untuk keluar dari halaman member. Berikut ini adalah kode untuk melakukan logout.

```
<!-- start: User Dropdown -->
<li class="dropdown">
    <a class="btn dropdown-toggle" data-toggle="dropdown" href="#">
        <i class="halflings-icon white user"></i> @User.Identity.Name
        <span class="caret"></span>
    </a>
    <ul class="dropdown-menu">
        <li class="dropdown-menu-title">
            <span>Account Settings</span>
        </li>
        <li><a href="#"><i class="halflings-icon user"></i> Profile</a></li>
        <li><a href="@Url.Action("LogOff", "Account")"><i class="halflings-icon off"></i> Logout</a></li>
    </ul>
</li>
<!-- end: User Dropdown -->
```

Pada kode di atas dapat dilihat untuk melakukan logout dari sistem dapat dilakukan dengan mengakses method aksi LogOff yang ada pada controller Account. Pada kode di atas juga dapat dilihat cara menampilkan username yang sedang login dengan cara menggunakan kode berikut ini.

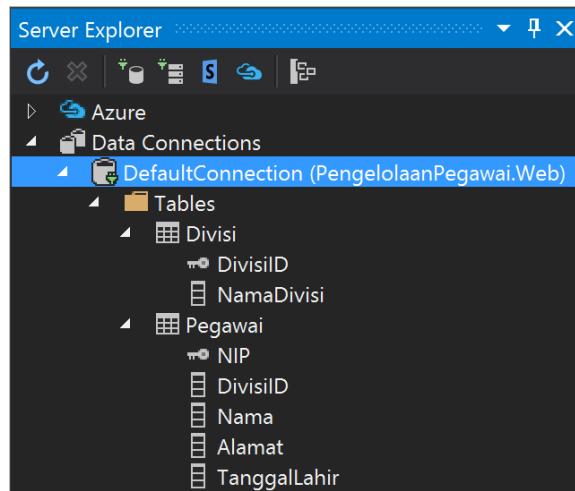
```
@User.Identity.Name
```

Uji Coba

Setelah semua persiapan di atas dilakukan maka pada bagian ini akan dilakukan uji coba untuk mengetahui hal-hal berikut ini, yaitu :

1. Fungsi Entity Framework Code First yang akan membuat tabel-tabel yang diperlukan untuk penyimpanan data user secara otomatis ketika aplikasi pertama kali dijalankan.
2. Fungsi untuk registrasi user.
3. Fungsi untuk proses otentifikasi dengan menggunakan form login.

Pengujian fungsi pertama dapat dilakukan cukup dengan cara mengeksekusi project. Jika sebelum mengimplementasikan ASP.NET Identity hanya dimiliki dua tabel pada database seperti yang terlihat pada gambar.



Gambar 127. Kondisi awal pada database.

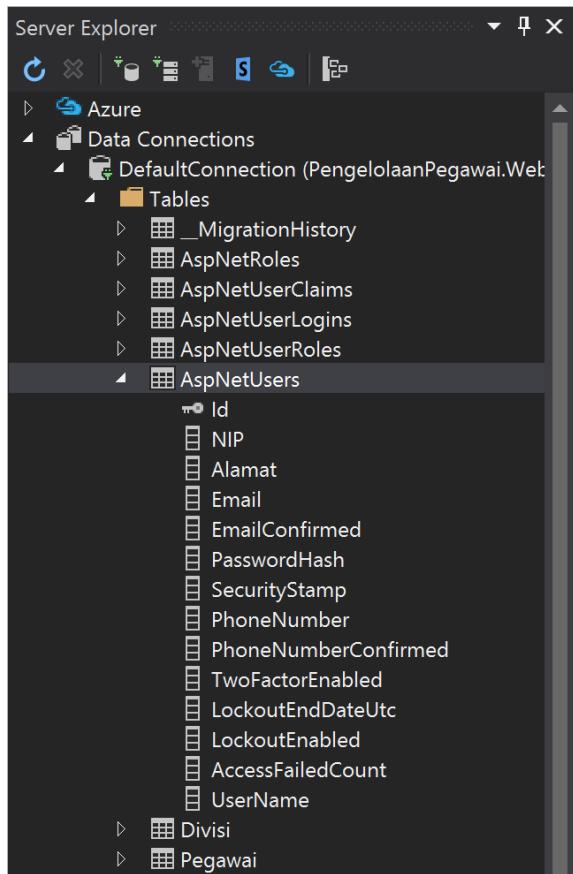
Registrasi User

Pada bagian ini akan diuji melakukan registrasi user dengan cara melakukan pengisian nilai-nilai pada form seperti berikut ini.

The image shows a registration form titled 'kembali'. The form has six input fields: 'User Name' (rezafaisal), 'NIP' (123321), 'Email' (rezafaisal@rezafaisal.net), 'Alamat' (Jl. Kayu Tangi II No. 666), 'Password' (*****), and 'Confirm password' (*****). Below the form is a blue 'Register' button.

Gambar 128. Form register pengguna/user yang telah diisi.

Kemudian setelah tombol Register diklik maka akan diperlihatkan perubahan pada database dan isi dari tabel yang bersangkutan. Perubahan pada database dapat dilihat penambahan tabel seperti pada gambar di bawah ini.



Gambar 129. Tabel tambahan untuk pengelolaan data pengguna/user.

Pada gambar dapat dilihat tambahan tabel-tabel berikut ini:

1. AspNetRoles.
2. AspNetUserClaims.
3. AspNetUserLogins.
4. AspNetUserRoles.
5. AspNetUsers.

Kemudian pada tabel AspNetUsers dapat dilihat secara default memiliki field-field sebagai berikut:

1. Id.
2. Email.
3. EmailConfirmed.
4. PasswordHash.
5. SecurityStamp.
6. PhoneNumber.
7. PhoneNumberConfirmed
8. TwoFactorEnabled.
9. LockoutEndDateUtc.
10. LockoutEnabled.
11. AccessFailedCount.
12. Username.

Sedang field NIP dan Alamat merupakan field tambahan yang dihasilkan karena penambahan property pada class ApplicationUser dengan kode seperti di bawah ini.

```
public class ApplicationUser : IdentityUser
{
    // Profile data tambahan dapat dilakukan dengan menambahkan property
    // pada class ApplicationUser
    public string NIP { get; set; }
    public string Alamat { get; set; }

    public async Task<ClaimsIdentity>
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
    {
        // Note the authenticationType must match the one defined in
        CookieAuthenticationOptions.AuthenticationType
        var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
        // Add custom user claims here
        return userIdentity;
    }
}
```

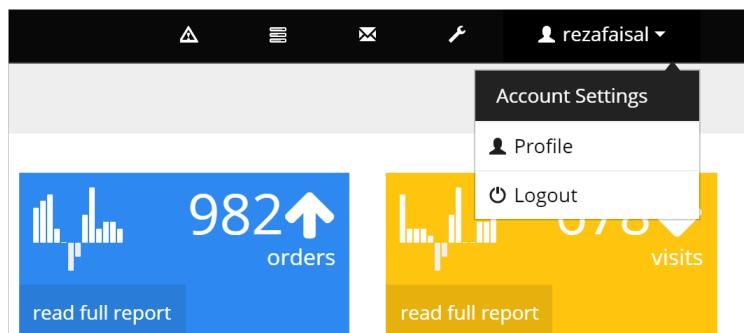
Pengujian halaman registrasi user ini dapat dilakukan tanpa melakukan modifikasi pada halaman MasterLayout.cshtml yang diperlihatkan pada bagian sebelumnya.

Login

Selanjutnya akan dicoba untuk menguji proses otentikasi. Untuk pengujian fungsi ini maka harus dilakukan modifikasi pada halaman MasterLayout.cshtml seperti yang telah dijelaskan pada bagian sebelumnya. Yang pertama dilakukan melakukan akses ke alamat berikut dari browser <http://localhost:51165/Account/Register>. Karena halaman ini menggunakan layout MasterLayout.cshtml yang telah dilindungi agar hanya bisa diakses oleh user yang telah terotentikasi, maka ketika user langsung mengakses alamat tersebut maka user akan langsung dikembalikan ke halaman login di <http://localhost:51165/Account/Login>.

Logout

Setelah user berhasil login, maka akan ditampilkan username seperti pada gambar di bawah ini.



Gambar 130. Informasi user yang aktif dan tombol logout.

Pada bagian ini akan dicoba untuk menguji fungsi logout, yaitu cukup dengan mengklik tombol Logout seperti yang terlihat pada gambar. Hasilnya secara otomatis hasilnya akan menghapus informasi login user pada cookies dan user akan diantarkan kembali ke halaman login.

Pengelolaan Role

Pada bagian di atas sudah diterangkan implementasi otentikasi user. Pada bagian ini akan dilakukan penambahan fungsi pengelolaan role sehingga user dapat dikelompokkan berdasarkan role. Selain itu akan diterangkan implementasi role dalam pengelolaan menu dan aksi berdasarkan role dari user yang sedang login. Hal ini dibutuhkan untuk mengelola menu apa saja yang dapat diakses oleh user dengan role tertentu, atau aksi apa saja yang diperbolehkan diakses oleh user dengan role tertentu.

Berikut ini akan dijelaskan langkah demi langkah untuk melakukan implementasi pengelolaan role dengan menggunakan ASP.NET Identity.

Untuk mengelola role tidak diperlukan pembuatan class model.

Controller

Berikut ini adalah class controller yang digunakan untuk mengelola role.

```
RolesController.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

using PengelolaanPegawai.Web.Models;

namespace PengelolaanPegawai.Web.Controllers
{
    public class RolesController : Controller
    {
        private ApplicationDbContext context = new ApplicationDbContext();

        // GET: Roles
        public ActionResult Index()
        {
            var roles = context.Roles.ToList();
            return View(roles);
        }

        // GET: /Roles/Create
        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Roles/Create
        [HttpPost]
        public ActionResult Create(FormCollection collection)
        {
            try
            {
                context.Roles.Add(new
Microsoft.AspNet.Identity.EntityFramework.IdentityRole()
                {
                    Name = collection["RoleName"]
                });
                context.SaveChanges();
                ViewBag.ResultMessage = "Role created successfully !";
                return RedirectToAction("Index");
            }
        }
    }
}
```

```

        }
        catch
        {
            return View();
        }
    }

    // GET: /Roles/Edit/4
    public ActionResult Edit(string roleName)
    {
        var thisRole = context.Roles.Where(r => r.Name.Equals(roleName,
StringComparison.CurrentCultureIgnoreCase)).FirstOrDefault();

        return View(thisRole);
    }

    //
    // POST: /Roles/Edit/4
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult
Edit(Microsoft.AspNet.Identity.EntityFramework.IdentityRole role)
    {
        try
        {
            context.Entry(role).State =
System.Data.Entity.EntityState.Modified;
            context.SaveChanges();

            return RedirectToAction("Index");
        }
        catch
        {
            return View();
        }
    }

    // GET: /Roles/Get/4
    public ActionResult Delete(string RoleName)
    {
        var thisRole = context.Roles.Where(r => r.Name.Equals(RoleName,
StringComparison.CurrentCultureIgnoreCase)).FirstOrDefault();
        context.Roles.Remove(thisRole);
        context.SaveChanges();
        return RedirectToAction("Index");
    }
}
}

```

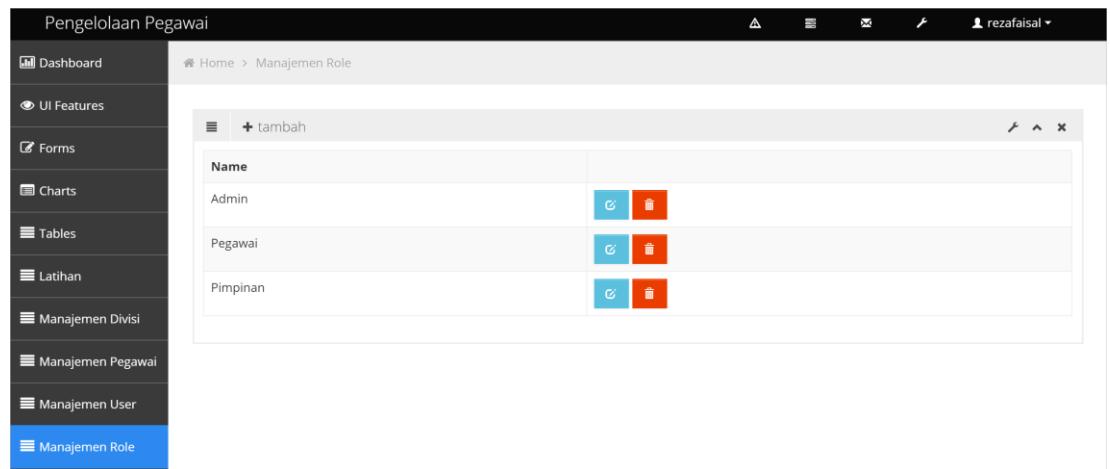
View

Untuk mengelola role akan dibuat 3 view saja yaitu:

1. Index.cshtml, untuk menampilkan daftar role yang telah dibuat.
2. Create.cshtml, untuk menampilkan form input role.
3. Edit.cshtml, untuk menampilkan form edit role.

Index

Berikut ini adalah tampilan dari halaman view Index.cshtml.



Gambar 131. Daftar role.

Dan berikut ini adalah isi dari file Index.cshtml.

```

Index.cshtml
@model IEnumerable<Microsoft.AspNet.Identity.EntityFramework.IdentityRole>
@{
    ViewBag.Title = "Manajemen Role";
}



<div class="box span12" style="margin-left:0px">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
plus"></i>@Html.ActionLink("tambah", "Create")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        <div class="box-content">
            <table class="table table-striped table-bordered bootstrap-
datatable">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.Name)
                    </th>
                    <th></th>
                </tr>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.Name)
                        </td>
                        <td>
                            <a class="btn btn-info"
                            href="@Url.Action("Edit", new { roleName = item.Name })">
                                <i class="halflings-icon white edit"></i>
                            </a>
                            <a class="btn btn-danger"
                            href="~/Roles/Delete?RoleName=@item.Name">


```

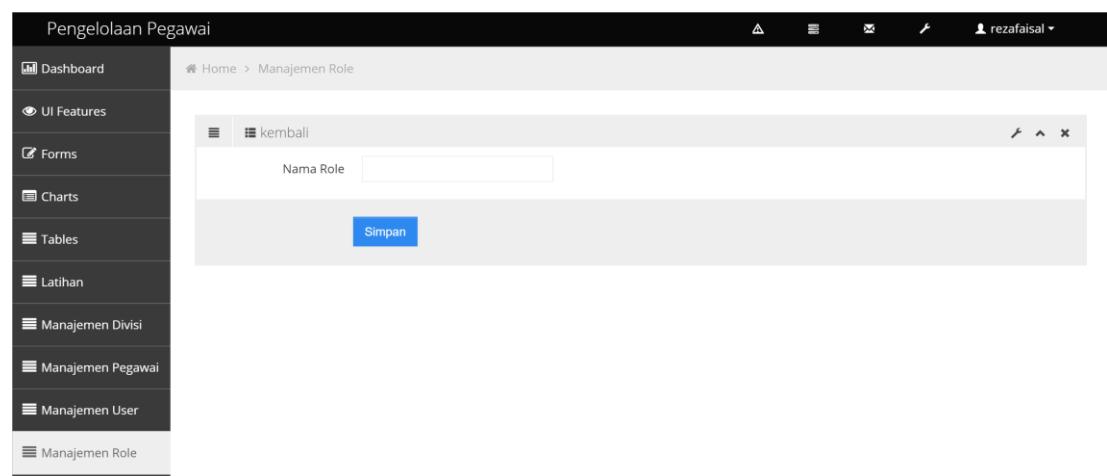
```

                <i class="halflings-icon white trash"></i>
            </a>
        </td>
    </tr>
}
</table>
</div>
</div>
</div>

```

Create

Dan berikut ini adalah tampilan dari form untuk menambah data.



Gambar 132. Form menambah role.

Dan berikut adalah isi dari file Create.cshtml.

```

Create.cshtml

@{
    ViewBag.Title = "Manajemen Role";
}



## <i class="halflings-icon align-justify"></i><span class="break"></span><i class="halflings-icon th-list"></i>@Html.ActionLink("kembali", "Index")</h2> <i class="halflings-icon wrench"></i></a> <i class="halflings-icon chevron-up"></i></a> <i class="halflings-icon remove"></i></a>



@using (Html.BeginForm("Create", "Roles", FormMethod.Post, new {
@class = "form-horizontal" }))
{
    <fieldset>


```

```

        @Html.AntiForgeryToken()
        @Html.ValidationSummary(true, "", new { @class = "text-
danger" })

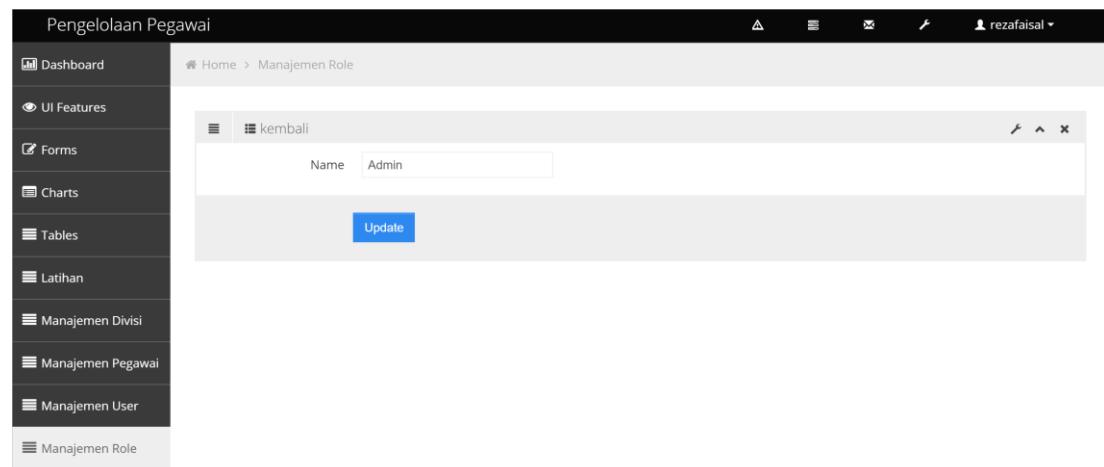
            <div class="control-group">
                @Html.Label("Nama Role", htmlAttributes: new {
                    @class = "control-label" })
                <div class="controls">
                    @Html.TextBox("RoleName", "", new {
                        htmlAttributes = new { @class = "span6 typeahead" } })
                </div>
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-
primary">Simpan</button>
            </div>
        </div>
    </div>
</div>

```

Edit

Dari berikut ini adalah tampilan dari form untuk mengedit role yang dipilih dari daftar role.



Gambar 133. Form untuk mengedit role.

Dan berikut adalah isi dari file Edit.cshtml.

```

Edit.cshtml
@model Microsoft.AspNet.Identity.EntityFramework.IdentityRole
 @{
    ViewBag.Title = "Manajemen Role";
}

<div class="row-fluid sortable">
    <div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i><@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>

```

```

        <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
        <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
    </div>
<div class="box-content">
    @using (Html.BeginForm("Edit", "Roles", FormMethod.Post, new {
@class = "form-horizontal" }))
    {
        <fieldset>
            @Html.AntiForgeryToken()
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            @Html.HiddenFor(model=>model.Id)

            <div class="control-group">
                @Html.LabelFor(model => model.Name, htmlAttributes:
new { @class = "control-label" })
                <div class="controls">
                    @Html.TextBoxFor(model => model.Name, "", new {
htmlAttributes = new { @class = "span6 typeahead" } })
                </div>
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-primary">Update</button>
            </div>
        </fieldset>
    }
</div>
</div>

```

Pengelolan User Aplikasi

Untuk menggunakan role yang telah dibuat di atas, maka pada bagian ini dibuat pengelolaan user aplikasi yang telah menggunakan role yang telah dibuat sebelumnya.

Model

Untuk mendukung fitur ini digunakan beberapa class yang telah dibuat pada AccountViewModel.cs di atas. Class-class yang digunakan pada fitur ini adalah :

1. UserViewModel.
2. UserFormViewModel.

Berikut ini adalah kode dari kedua class tersebut.

```

public class UserViewModel
{
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Display(Name = "NIP")]
    public string NIP { get; set; }

    [Display(Name = "Email")]
    public string Email { get; set; }

```

```

[Display(Name = "Alamat")]
public string Alamat { get; set; }

[Display(Name = "Role")]
public string Roles { get; set; }

}

public class UserFormViewModel
{
    [Required]
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Required]
    [Display(Name = "NIP")]
    public string NIP { get; set; }

    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Display(Name = "Alamat")]
    public string Alamat { get; set; }

    [Required]
    [Display(Name = "Role")]
    public string Role { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    public IEnumerable<System.Web.Mvc.SelectListItem> Roles { get; set; }
}

```

Class UserViewModel akan digunakan untuk menampilkan data dalam bentuk tabel grid pada halaman view Index dan data detail user yang dipilih pada view Detail dan Delete. Sedangkan class UserFormViewModel akan digunakan pada halaman yang memiliki form seperti pada view Create dan Edit.

Controller

Pada pengelolaan user aplikasi memiliki beberapa aksi yaitu:

1. Index.
2. Detail.
3. Create.

4. Edit.
5. Delete.

Kelima method aksi tersebut telah dibuat di dalam class AccountController.cs. berikut adalah isi dari kelima method tersebut.

Index

Method aksi ini berfungsi untuk mengkoleksi data pada objek items yang nantinya akan ditampilkan dalam bentuk tabel grid pada view Index.

```
// GET: Roles
public ActionResult Index()
{
    var users = UserManager.Users.Select(p => p).ToList();
    List<UserViewModel> items = new List<UserViewModel>();
    foreach (var user in users)
    {
        UserViewModel item = new UserViewModel();
        item.UserName = user.UserName;
        item.Alamat = user.Alamat;
        item.NIP = user.NIP;
        item.Email = user.Email;

        var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
        if (String.IsNullOrEmpty(roleForUser))
        {
            roleForUser = "-";
        }
        item.Roles = roleForUser;

        items.Add(item);
    }
    return View(items);
}
```

Dapat dilihat pada kode di atas digunakan model UserViewModel.

Detail

Method aksi ini berfungsi untuk mengambil data user yang dipilih.

```
// GET: Divisis/Details/nama_user
public ActionResult Details(string id)
{
    if (String.IsNullOrEmpty(id))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    var user = UserManager.FindByName(id);
    var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
    if (String.IsNullOrEmpty(roleForUser))
    {
        roleForUser = "-";
    }

    UserViewModel model = new UserViewModel();
    model.UserName = user.UserName;
    model.NIP = user.NIP;
```

```

    model.Email = user.Email;
    model.Alamat = user.Alamat;
    model.Roles = roleForUser;

    return View(model);
}

```

Dapat dilihat pula penggunaan class model UserViewModel pada method aksi ini.

Create Get & Create Post

Kedua method aksi ini secara kesatuan berfungsi untuk membuat data user baru. Terdiri atas method Create dengan method Get yang berfungsi untuk menyiapkan form input user. Berikut ini adalah kode dari method aksi Create tersebut.

```

// GET: Divisis/Create
public ActionResult Create()
{
    ApplicationDbContext context = new ApplicationDbContext();

    var model = new UserFormViewModel
    {
        Roles = context.Roles.ToList().Select(x => new SelectListItem
        {
            Value = x.Name,
            Text = x.Name
        })
    };

    return View(model);
}

```

Pada kode method di atas dapat dilihat terdapat baris untuk mengkoleksi data role yang akan ditampilkan pada kontrol Html.DropDownListFor.

Sedangkan method aksi Create yang menggunakan method Post berfungsi untuk menangani data yang telah diisi pada halaman view Create untuk disimpan ke dalam database. Berikut adalah kode dari method tersebut.

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Create(UserFormViewModel model)
{
    ApplicationDbContext context = new ApplicationDbContext();

    model.Roles = context.Roles.ToList().Select(x => new SelectListItem
    {
        Value = x.Name,
        Text = x.Name
    });

    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.UserName, Email = model.Email, NIP = model.NIP, Alamat = model.Alamat };
        var result = await UserManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            await UserManager.AddToRoleAsync(user.Id, model.Role);
            return RedirectToAction("Index");
        }
    }
}

```

```

        }

    }

    return View(model);
}

```

Dari kode di atas dapat dilihat menggunakan UserManager untuk melakukan pembuatan user dengan memanfaatkan method CreateAsync. Selain itu juga digunakan SignInManager untuk membuat identitas user, sehingga data tambahan untuk user dapat disimpan.

Harap diperhatikan, jika pada view terdapat kontrol yang datanya dikoleksi dari method aksi maka baik method aksi yang menggunakan method Get atau Post harus tetap melakukan koleksi data tersebut agar tidak terjadi error. Baris berikut adalah kode yang digunakan untuk memberikan data ke view Create.

```

ApplicationDbContext context = new ApplicationDbContext();

model.Roles = context.Roles.ToList().Select(x => new SelectListItem
{
    Value = x.Name,
    Text = x.Name
});

```

Jika kode tersebut hanya terdapat pada method aksi yang menggunakan Get saja, maka akan terjadi kesalahan jika dilakukan proses yang dilakukan oleh method aksi yang menggunakan Post yang tidak menggunakan baris kode di atas.

Edit Get & Edit Post

Seperti halnya pada method aksi Create, pada method aksi Edit juga terdapat dua method aksi yang menjadi satu kesatuan. Yang pertama adalah method Edit yang menggunakan method Get dengan kode di bawah ini.

```

// GET: Account/Edit/nama_user
public ActionResult Edit(string id)
{
    if (String.IsNullOrEmpty(id))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    ApplicationDbContext context = new ApplicationDbContext();
    var user = UserManager.FindByName(id);
    var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
    if (String.IsNullOrEmpty(roleForUser))
    {
        roleForUser = "-";
    }

    UserFormViewModel model = new UserFormViewModel();
    model.UserName = user.UserName;
    model.NIP = user.NIP;
    model.Email = user.Email;
    model.Alamat = user.Alamat;
    model.Role = roleForUser;
    model.Password = user.PasswordHash;
    model.ConfirmPassword = user.PasswordHash;
}

```

```

model.Roles = context.Roles.ToList().Select(x => new SelectListItem
{
    Value = x.Name,
    Text = x.Name,
    Selected = x.Name == model.Role
});

return View(model);
}

```

Pada kode di atas dapat dilihat kode untuk mendapatkan data user yang dipilih untuk ditampilkan pada view Edit.

Setelah data diubah maka selanjutnya akan dilakukan proses update data dengan menggunakan method aksi Edit yang menggunakan method Post dengan kode di bawah ini.

```

// POST: Account/Edit/nama_user
// To protect from overposting attacks, please enable the specific
// properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Edit(UserFormViewModel model)
{
    ApplicationDbContext context = new ApplicationDbContext();

    model.Roles = context.Roles.ToList().Select(x => new SelectListItem
    {
        Value = x.Name,
        Text = x.Name
    });

    if (ModelState.IsValid)
    {
        var user = UserManager.FindByName(model.UserName);

        var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
        if (String.IsNullOrEmpty(roleForUser))
        {
            roleForUser = "-";
        }

        user.NIP = model.NIP;
        user.UserName = model.UserName;
        user.Alamat = model.Alamat;
        user.Email = model.Email;

        if (!String.IsNullOrEmpty(model.Password) &&
!String.IsNullOrEmpty(model.ConfirmPassword))
        {

        }
        else
        {
            user.PasswordHash = model.Password;
        }

        var result = UserManager.Update(user);
        if (result.Succeeded)
        {
            await UserManager.RemoveFromRoleAsync(user.Id, roleForUser);
        }
    }
}

```

```

        await UserManager.AddToRoleAsync(user.Id, model.Role);
        return RedirectToAction("Index");
    }

    return View(model);
}

```

Delete Get & Delete Post

Method aksi Delete berfungsi untuk menghapus data user yang dipilih. Berikut adalah kode dari kedua method aksi ini.

```

// GET: Divisis/Delete/nama_user
public ActionResult Delete(string id)
{
    if (String.IsNullOrEmpty(id))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    //var user = context.Users.FirstOrDefault(u => u.UserName.Equals(id));
    var user = UserManager.FindByName(id);
    var roleForUser = UserManager.GetRoles(user.Id).Select(x =>
x).FirstOrDefault();
    if (String.IsNullOrEmpty(roleForUser))
    {
        roleForUser = "-";
    }

    UserViewModel model = new UserViewModel();
    model.UserName = user.UserName;
    model.NIP = user.NIP;
    model.Email = user.Email;
    model.Alamat = user.Alamat;
    model.Roles = roleForUser;

    return View(model);
}

// POST: Divisis/Delete/nama_user
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(string id)
{
    var user = UserManager.FindByName(id);
    if (user != null)
    {
        UserManager.Delete(user);
    }

    return RedirectToAction("Index");
}

```

View

Setiap method aksi yang telah dibuat pada class controller akan menggunakan komponen view yang sesuai dengan nama method aksi tersebut.

Index

Berikut ini adalah kode dari Index.cshtml.

```
Index.cshtml
@model IEnumerable<PengelolaanPegawai.Web.Models.UserViewModel>
@if (ViewBag.Title = "Manajemen User");
}

<div class="row-fluid sortable">
    <div class="box span12" style="margin-left:0px">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
plus"></i>@Html.ActionLink("tambah", "Create")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            <table class="table table-striped table-bordered bootstrap-
datatable">
                <tr>
                    <th>
                        @Html.DisplayNameFor(model => model.UserName)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.NIP)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.Email)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.Roles)
                    </th>
                    <th>
                        @Html.DisplayNameFor(model => model.Alamat)
                    </th>
                    <th></th>
                </tr>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>
                            @Html.DisplayFor(modelItem => item.UserName)
                        </td>
                        <td>
                            @Html.DisplayFor(modelItem => item.NIP)
                        </td>
                        <td>
```

```

                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Roles)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Alamat)
            </td>
            <td>
                <a class="btn btn-success"
                href="@Url.Action("Details", new { id = item.UserName })">
                    <i class="halflings-icon white zoom-in"></i>
                </a>
                <a class="btn btn-info"
                href="@Url.Action("Edit", new { id = item.UserName })">
                    <i class="halflings-icon white edit"></i>
                </a>
                <a class="btn btn-danger"
                href="@Url.Action("Delete", new { id = item.UserName })">
                    <i class="halflings-icon white trash"></i>
                </a>
            </td>
        </tr>
    }
</table>
</div>
</div>
</div>

```

Dan berikut adalah antarmuka dari view Index ini.

User Name	NIP	Email	Role	Alamat
rezafaisal	123321	rezafaisal@rezafaisal.net	Admin	Jl. Kayu Tangi II No. 666

Gambar 134. Daftar user.

Detail

Berikut ini adalah kode dari file Detail.cshtml.

```

Detail.cshtml
@model PengelolaanPegawai.Web.Models.UserViewModel

@{
    ViewBag.Title = "Manajemen User";
}



<div class="box span12">
        <div class="box-header" data-original-title>

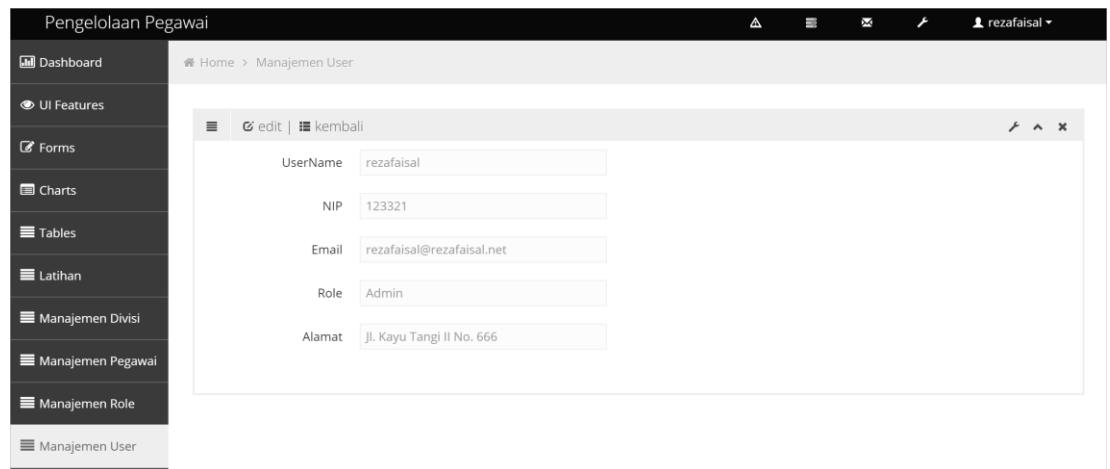

```

```

        <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
edit"></i>@Html.ActionLink("edit", "Edit", new { id = Model.UserName }) | <i
class="halflings-icon th-list"></i>@Html.ActionLink("kembali", "Index")</h2>
        <div class="box-icon">
            <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
            <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
            <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
        </div>
        <div class="box-content">
            <form class="form-horizontal">
                <fieldset>
                    <div class="control-group">
                        <label class="control-label">UserName</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.UserName)</span>
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">NIP</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.NIP)</span>
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">Email</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.Email)</span>
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">Role</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.Roles)</span>
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">Alamat</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.Alamat)</span>
                        </div>
                    </div>
                </fieldset>
            </form>
        </div>
    </div>

```

Dari antarmuka dari view ini dapat dilihat pada gambar di bawah ini.



Gambar 135. Detail user yang dipilih.

Create

Berikut ini adalah kode dari file Create.cshtml.

```
Create.cshtml
@model PengelolaanPegawai.Web.Models.UserFormViewModel

@{
    ViewBag.Title = "Manajemen User";
}
@* @ViewBag.NamaUser<br />
@ViewBag.Error*@


<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Create", "Account", FormMethod.Post, new
{ @class = "form-horizontal", role = "form" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    @Html.ValidationSummary("", new { @class = "text-danger" })
                </fieldset>
                <div class="control-group">
                    @Html.LabelFor(m => m.UserName, htmlAttributes: new
{ @class = "control-label" })
                    <div class="controls">
                        @Html.TextBoxFor(m => m.UserName, new {
                            htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
                </div>
                <div class="control-group">


```

```

@Html.LabelFor(m => m.NIP, htmlAttributes: new {
@class = "control-label" })
<div class="controls">
    @Html.TextBoxFor(m => m.NIP, new {
htmlAttributes = new { @class = "span6 typeahead" } })
        </div>
    </div>
    <div class="control-group">
        @Html.LabelFor(m => m.Email, htmlAttributes: new {
@class = "control-label" })
            <div class="controls">
                @Html.TextBoxFor(m => m.Email, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                    </div>
            </div>
            <div class="control-group">
                @Html.LabelFor(m => m.Roles, htmlAttributes: new {
@class = "control-label" })
                    <div class="controls">
                        @Html.DropDownListFor(m => m.Role, Model.Roles,
"Pilih Role", htmlAttributes: new { @data_rel = "chosen" })
                    </div>
                </div>
                <div class="control-group">
                    @Html.LabelFor(m => m.Alamat, htmlAttributes: new {
@class = "control-label" })
                        <div class="controls">
                            @Html.TextAreaFor(m => m.Alamat, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                                </div>
                        </div>
                        <div class="control-group">
                            @Html.LabelFor(m => m.Password, htmlAttributes: new {
{ @class = "control-label" })
                                <div class="controls">
                                    @Html.PasswordFor(m => m.Password, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                                </div>
                            </div>
                            <div class="control-group">
                                @Html.LabelFor(m => m.ConfirmPassword,
htmlAttributes: new { @class = "control-label" })
                                    <div class="controls">
                                        @Html.PasswordFor(m => m.ConfirmPassword, new {
htmlAttributes = new { @class = "span6 typeahead" } })
                                    </div>
                                </div>
                            <div class="form-actions">
                                <button type="submit" class="btn btn-primary">Register</button>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

Dan antarmuka dari view ini dapat dilihat pada gambar di bawah ini.

Gambar 136. Form input data user.

Edit

Berikut adalah kode dari view Edit.cshtml.

```
Edit.cshtml
@model PengelolaanPegawai.Web.Models.UserFormViewModel

@{
    ViewBag.Title = "Manajemen User";
}



<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon th-
list"></i>@Html.ActionLink("kembali", "Index")</h2>
            <div class="box-icon">
                <a href="#" class="btn-setting"><i class="halflings-icon wrench"></i></a>
                <a href="#" class="btn-minimize"><i class="halflings-icon chevron-up"></i></a>
                <a href="#" class="btn-close"><i class="halflings-icon remove"></i></a>
            </div>
        </div>
        <div class="box-content">
            @using (Html.BeginForm("Edit", "Account", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
            {
                <fieldset>
                    @Html.AntiForgeryToken()
                    @Html.ValidationSummary("", new { @class = "text-danger" })
                </fieldset>
                <div class="control-group">
                    @Html.LabelFor(m => m.UserName, htmlAttributes: new
{ @class = "control-label" })
                    <div class="controls">


```

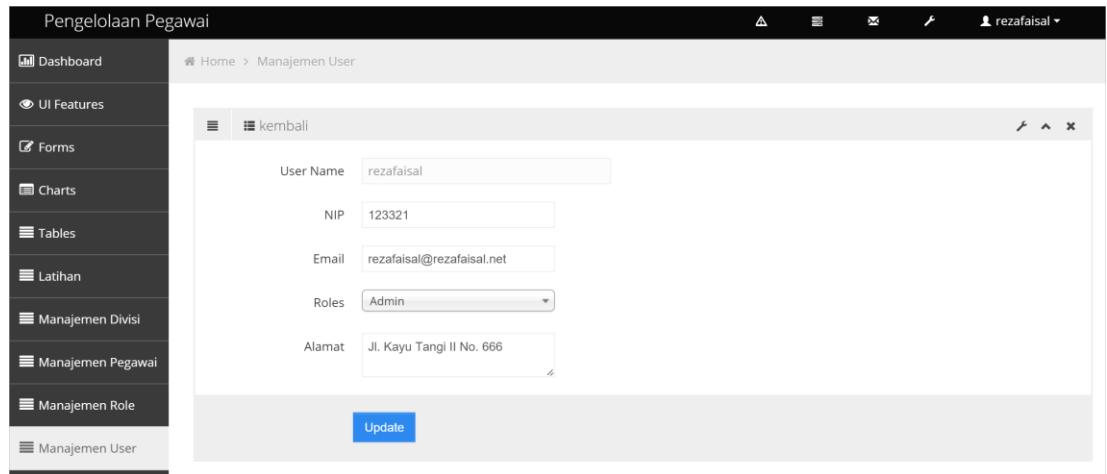
```

        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.UserName)</span>
                @Html.HiddenFor(m=> m.UserName)
            </div>
        </div>
        <div class="control-group">
            @Html.LabelFor(m => m.NIP, htmlAttributes: new {
@class = "control-label" })
            <div class="controls">
                @Html.TextBoxFor(m => m.NIP, new {
htmlAttributes = new { @class = "span6 typeahead" } })
            </div>
        </div>
        <div class="control-group">
            @Html.LabelFor(m => m.Email, htmlAttributes: new {
@class = "control-label" })
            <div class="controls">
                @Html.TextBoxFor(m => m.Email, new {
htmlAttributes = new { @class = "span6 typeahead" } })
            </div>
        </div>
        <div class="control-group">
            @Html.LabelFor(m => m.Roles, htmlAttributes: new {
@class = "control-label" })
            <div class="controls">
                @Html.DropDownListFor(m => m.Role, Model.Roles,
"Pilih Role", htmlAttributes: new { @data_rel = "chosen" })
            </div>
        </div>
        <div class="control-group">
            @Html.LabelFor(m => m.Alamat, htmlAttributes: new {
@class = "control-label" })
            <div class="controls">
                @Html.TextAreaFor(m => m.Alamat, new {
htmlAttributes = new { @class = "span6 typeahead" } })
            </div>
        </div>
        @Html.HiddenFor(m=>m.Password)
        @Html.HiddenFor(m => m.ConfirmPassword)

        <div class="form-actions">
            <button type="submit" class="btn btn-primary">Update</button>
        </div>
    </fieldset>
}
</div>
</div>
</div>

```

Dan gambar di bawah ini adalah antarmuka dari view ini.



Gambar 137. Form update user yang dipilih.

Delete

Sedangkan view untuk menghapus user ditangani ole file Delete.cshtml.

```
Delete.cshtml
@model PengelolaanPegawai.Web.Models.UserViewModel

@{
    ViewBag.Title = "Manajemen User";
}



<div class="box span12">
        <div class="box-header" data-original-title>
            <h2><i class="halflings-icon align-justify"></i><span
class="break"></span><i class="halflings-icon
edit"></i>@Html.ActionLink("edit", "Edit", new { id = Model.UserName }) | <i
class="halflings-icon th-list"></i>@Html.ActionLink("kembali", "Index")</h2>
        <div class="box-icon">
            <a href="#" class="btn-setting"><i class="halflings-icon
wrench"></i></a>
            <a href="#" class="btn-minimize"><i class="halflings-icon
chevron-up"></i></a>
            <a href="#" class="btn-close"><i class="halflings-icon
remove"></i></a>
        </div>
        @ViewBag.Username
        <div class="box-content">
            @using (Html.BeginForm("Delete", "Account", FormMethod.Post, new
{ @class = "form-horizontal" }))
            {
                @Html.AntiForgeryToken()
                <fieldset>
                    <div class="control-group">
                        <label class="control-label">UserName</label>
                        <div class="controls">
                            <span class="input-xlarge uneditable-
input">@Html.DisplayFor(model => model.UserName)</span>
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">NIP</label>
                        <div class="controls">


```

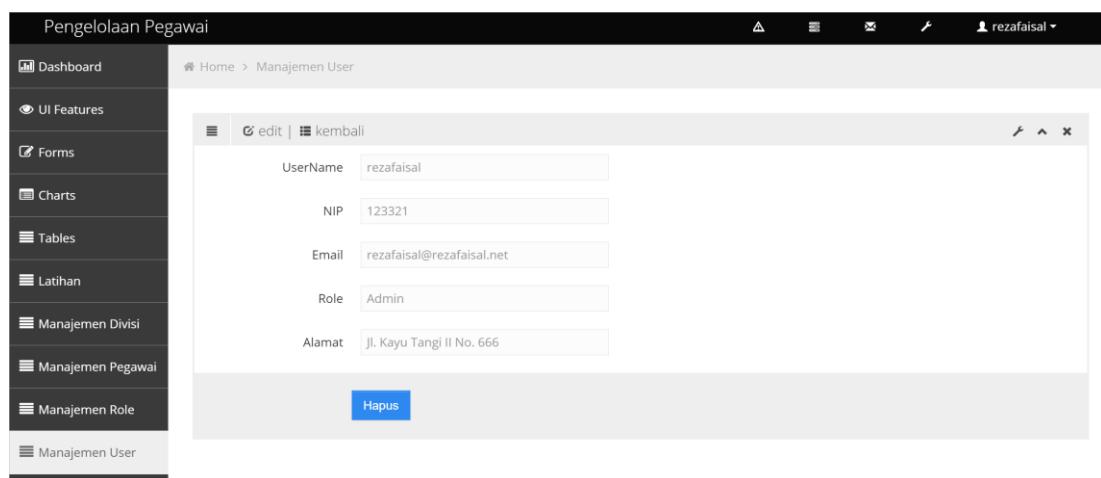
```

        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.NIP)</span>
    </div>
</div>
<div class="control-group">
    <label class="control-label">Email</label>
    <div class="controls">
        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Email)</span>
    </div>
</div>
<div class="control-group">
    <label class="control-label">Role</label>
    <div class="controls">
        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Roles)</span>
    </div>
</div>
<div class="control-group">
    <label class="control-label">Alamat</label>
    <div class="controls">
        <span class="input-xlarge uneditable-input">@Html.DisplayFor(model => model.Alamat)</span>
    </div>
</div>

        <div class="form-actions">
            <button type="submit" class="btn btn-primary">Hapus</button>
        </div>
    </div>
</div>

```

Dan berikut ini adalah screenshot dari antarmuka view ini.



Gambar 138. Konfirmasi untuk menghapus user.

Otorisasi

Otorisasi berfungsi untuk membatasi akses suatu halaman agar hanya dapat diakses oleh user tertentu saja atau user-user dari role tertentu saja. Sebagai contoh fitur untuk mengelola user

hanya bisa digunakan oleh user dari role admin saja. Atau fitur laporan hanya bisa diakses oleh user dari role eksekutif saja.

Cara sederhana yang bisa dilakukan untuk implementasi proses otorisasi pada ASP.NET MVC akan dibahas pada bagian ini.

Otorisasi Menu

Cara pertama adalah dengan menampilkan menu sesuai dengan role dari user yang mengakses aplikasi web. Berikut ini adalah tampilan menu dari aplikasi web.

The screenshot shows a web application interface titled 'Pengelolaan Pegawai'. On the left, there is a sidebar with a navigation menu containing items like 'Dashboard', 'UI Features', 'Forms', 'Charts', 'Tables', 'Latihan', 'Manajemen Divisi', 'Manajemen Pegawai', 'Manajemen Role', and 'Manajemen User'. The 'Manajemen User' item is highlighted with a blue background. The main content area displays a table with three rows of user data:

User Name	NIP	Email	Role	Alamat	Action
rezafaisal	123321	rezafaisal@rezafulan.net	Admin	Jl. Kayu Tangi II No. 666	
pegawai	123444	pegawai@esevens.net	Pegawai	Alamat Pegawai123#	
pimpinan	1001444	pimpinan@esevens.net	Pimpinan	Alamat Pimpinan123#	

Gambar 139. Menu aplikasi Pengelolaan Pegawai.

Sebelumnya telah dibuat tiga role yaitu:

1. Admin.
2. Pegawai.
3. Pimpinan.

Dan kemudian dibuat tiga user yaitu:

1. rezafaisal dengan role admin. Role admin dapat mengakses semua menu.
2. pegawai dengan role pegawai. Role pegawai hanya dapat mengakses menu Dashboard, UI Feature, Forms, Charts, Tables dan Latihan saja.
3. pimpinan dengan role pimpinan. Role ini dapat mengakses menu yang dapat diakses oleh role pegawai ditambah dengan menu Manajemen Divisi dan Manajemen Pegawai.

Maka langkah yang harus dilakukan adalah mengedit menu yang ada pada halaman MasterLayout.cshtml menjadi seperti berikut ini.

```
<div id="sidebar-left" class="span2">
    <div class="nav-collapse sidebar-nav">
        <ul class="nav nav-tabs nav-stacked main-menu">
            <li><a href="@Url.Action("Index", "Metro")"><i class="icon-bar-chart"></i><span class="hidden-tablet"> Dashboard</span></a></li>
                @if (User.IsInRole("Pegawai") || User.IsInRole("Pimpinan") || User.IsInRole("Admin"))
                {
                    <li><a href="@Url.Action("UI", "Metro")"><i class="icon-eye-open"></i><span class="hidden-tablet"> UI Features</span></a></li>
                    <li><a href="@Url.Action("Forms", "Metro")"><i class="icon-edit"></i><span class="hidden-tablet"> Forms</span></a></li>
                    <li><a href="@Url.Action("Charts", "Metro")"><i class="icon-list-alt"></i><span class="hidden-tablet"> Charts</span></a></li>
                }
            </ul>
        </div>
    </div>
</div>
```

```

        <li><a href="@Url.Action("Tables", "Metro")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Tables</span></a></li>
        <li><a href="@Url.Action("Latihan", "Metro")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Latihan</span></a></li>
    }
    @if (User.IsInRole("Pimpinan") || User.IsInRole("Admin"))
    {
        <li><a href="@Url.Action("Index", "Divisis")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Manajemen Divisi</span></a></li>
        <li><a href="@Url.Action("Index", "Pegawai")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Manajemen Pegawai</span></a></li>
    }
    @if (User.IsInRole("Admin"))
    {
        <li><a href="@Url.Action("Index", "Roles")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Manajemen Role</span></a></li>
        <li><a href="@Url.Action("Index", "Account")"><i class="icon-align-justify"></i><span class="hidden-tablet"> Manajemen User</span></a></li>
    }
</ul>
</div>
</div>

```

Dari kode di atas dapat dilihat cara sederhana dengan memanfaatkan baris kode dengan sintaks seperti contoh berikut ini.

```

@if (User.IsInRole("Admin"))
{
}

```

Dengan cara ini user nakal masih bisa mengakses halaman yang seharusnya bukan miliknya dengan cara langsung menulis alamat menuju halaman tersebut pada address bar.

Otorisasi Controller

Untuk mencegah user dapat langsung mengakses halaman yang seharusnya tidak bisa user tersebut akses maka dapat diberikan penentuan otorisasi pada method aksi yang ada pada controller.

Dengan cara ini maka pada suatu fitur dapat ditentukan hak akses untuk setiap method-method aksi untuk fitur tersebut. Sebagai contoh misalnya untuk fitur Pengelolaan Divisi, user pada role Pimpinan hanya dapat melihat daftar divisi, menambah dan mengedit data divisi tetapi tidak bisa menghapus data divisi. Tetapi user pada role Admin dapat melakukan semua aksi termasuk menghapus data divisi.

Untuk menentukan otorisasi pada method aksi digunakan atribut berikut ini.

```
[Authorize(Roles = "nama_role")]
```

Atau dengan sintaks berikut ini jika ingin memberikan otorisasi untuk multiple role

```
[Authorize(Roles = "nama_role1, nama_role2, ..., nama_roleN")]
```

Maka untuk kasus di atas dapat dilihat kode DivisisController.cs yang telah dimodifikasi.

```
DivisisController.cs
```

```
...
```

```

namespace PengelolaanPegawai.Web.Controllers
{
    public class DivisisController : Controller
    {
        private PegawaiDBEntities db = new PegawaiDBEntities();

        // GET: Divisis
        [Authorize(Roles ="Admin, Pimpinan")]
        public ActionResult Index()
        {
            ...
        }

        // GET: Divisis/Details/5
        [Authorize(Roles = "Admin, Pimpinan")]
        public ActionResult Details(int? id)
        {
            ...
        }

        // GET: Divisis/Create
        [Authorize(Roles = "Admin, Pimpinan")]
        public ActionResult Create()
        {
            ...
        }

        // POST: Divisis/Create
        // To protect from overposting attacks, please enable the specific
        properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin, Pimpinan")]
        public ActionResult Create([Bind(Include = "DivisiID,NamaDivisi")]
        Divisi divisi)
        {
            ...
        }

        // GET: Divisis/Edit/5
        [Authorize(Roles = "Admin, Pimpinan")]
        public ActionResult Edit(int? id)
        {
            ...
        }

        // POST: Divisis/Edit/5
        // To protect from overposting attacks, please enable the specific
        properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin, Pimpinan")]
        public ActionResult Edit([Bind(Include = "DivisiID,NamaDivisi")]
        Divisi divisi)
        {
            ...
        }

        // GET: Divisis/Delete/5
    }
}

```

```
[Authorize(Roles = "Admin")]
public ActionResult Delete(int? id)
{
    ...
}

// POST: Divisis/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
public ActionResult DeleteConfirmed(int id)
{
    ...
}

...
}
```

Jika user dengan role pimpinan mengakses method aksi Delete baik yang menggunakan method Get atau Post, maka user tersebut akan diredirect ke halaman login sebagai halaman default ketika ada user yang melanggar aturan hak akses.

Referensi

- <http://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity>
- <http://www.asp.net/identity/overview/getting-started/adding-aspnet-identity-to-an-empty-or-existing-web-forms-project>
- <http://www.dotnetfunda.com/articles/show/2898/working-with-roles-in-aspnet-identity-for-mvc>
- <http://stackoverflow.com/questions/19689183/add-user-to-role-asp-net-identity>
- <http://stackoverflow.com/questions/23241204/populate-mvc-dropdownlist-based-on-user-role>

7

Penutup

Sebagai penutup, ebook ini dibuat untuk para developer yang ingin berkenalan dengan ASP.NET MVC dengan mencoba langsung implementasinya dari contoh-contoh yang diberikan.

Masih banyak kemampuan dari ASP.NET MVC yang belum dibahas pada ebook ini seperti implementasi AJAX atau integrasi dengan Web API. Harapannya pada ebook selanjutnya materi-materi tersebut akan dibahas.

Akhir kalimat, jika ada kritik dan saran dapat ditujukan langsung via email ke alamat reza.faisal[at]gmail[dot]com.