

# Membangun Aplikasi Web Menggunakan Entity Framework dan MVC 5: Bagian 3

## Mengambil dan Menampilkan Data

Untuk contoh ini, saya akan membuat tampilan parsial untuk menampilkan daftar pengguna dari database. Tampilan parsial memungkinkan Anda untuk menentukan tampilan yang akan diberikan di dalam tampilan utama. Jika Anda menggunakan WebForms sebelumnya maka Anda bisa memikirkan tampilan parsial sebagai user-kontrol (ascx).

### LANGKAH 1: Menambah View Models

Hal pertama yang dibutuhkan adalah untuk membuat tampilan model untuk view. Tambahkan kode berikut di bawah ini dalam "UserModel.cs" class:

```
public class UserProfileView
{
    [Key]
    public int SYSUserID { get; set; }
    public int LOOKUPRoleID { get; set; }
    public string RoleName { get; set; }
    public bool? IsRoleActive { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Login ID")]
    public string LoginName { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Password")]
    public string Password { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "First Name")]
    public string FirstName { get; set; }
    [Required(ErrorMessage = "*")]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }
    public string Gender { get; set; }
}
```

```

public class LOOKUPAvailableRole
{
    [Key]
    public int LOOKUPRoleID { get; set; }
    public string RoleName { get; set; }
    public string RoleDescription { get; set; }
}

public class Gender
{
    public string Text { get; set; }
    public string Value { get; set; }
}

public class UserRoles
{
    public int? SelectedRoleID { get; set; }
    public IEnumerable<LOOKUPAvailableRole> UserRoleList { get; set; }
}

public class UserGender
{
    public string SelectedGender { get; set; }
    public IEnumerable<Gender> Gender { get; set; }
}

public class UserDataView
{
    public IEnumerable<UserProfileView> UserProfile { get; set; }
    public UserRoles UserRoles { get; set; }
    public UserGender UserGender { get; set; }
}

```

Jika Anda masih ingat, class view model adalah rumah dari properti yang hanya perlu untuk tampilan atau halaman.

**LANGKAH 2: Menambah ManageUserPartial view**

Buka "userManager" kelas dan menyatakan namespace di bawah:

```
using System.Collections.Generic;
```

namespace diatas interface dan kelas yang mendefinisikan koleksi generik, yang memungkinkan kita untuk membuat koleksi strongly-typed. Sekarang tambahkan kode berikut di bawah ini dalam kelas "userManager" Anda:

```
public List<LOOKUPAvailableRole> GetAllRoles() {
    using (DemoDBEntities db = new DemoDBEntities()) {
        var roles = db.LOOKUPRoles.Select(o => new LOOKUPAvailableRole {
            LOOKUPRoleID = o.LOOKUPRoleID,
            RoleName = o.RoleName,
            RoleDescription = o.RoleDescription
        }).ToList();

        return roles;
    }
}

public int GetUserID(string loginName) {
    using (DemoDBEntities db = new DemoDBEntities()) {
        var user = db.SYSUsers.Where(o => o.LoginName.Equals(loginName));
        if (user.Any())
            return user.FirstOrDefault().SYSUserID;
    }
    return 0;
}

public List<UserProfileView> GetAllUserProfiles() {
    List<UserProfileView> profiles = new List<UserProfileView>();
    using (DemoDBEntities db = new DemoDBEntities()) {
        UserProfileView UPV;
        var users = db.SYSUsers.ToList();

        foreach (SYSUser u in db.SYSUsers) {
            UPV = new UserProfileView();
        }
    }
}
```

```

        UPV.SYSUserID = u.SYSUserID;
        UPV.LoginName = u.LoginName;
        UPV.Password = u.PasswordEncryptedText;

        var SUP = db.SYSUserProfiles.Find(u.SYSUserID);
        if (SUP != null) {
            UPV.FirstName = SUP.FirstName;
            UPV.LastName = SUP.LastName;
            UPV.Gender = SUP.Gender;
        }

        var SUR = db.SYSUserRoles.Where(o => o.SYSUserID.Equals(u.SYSUserID));
        if (SUR.Any()) {
            var userRole = SUR.FirstOrDefault();
            UPV.LOOKUPRoleID = userRole.LOOKUPRoleID;
            UPV.RoleName = userRole.LOOKUPRole.RoleName;
            UPV.IsRoleActive = userRole.IsActive;
        }

        profiles.Add(UPV);
    }
}

return profiles;
}

public UserDataView GetUserDataView(string loginName) {
    UserDataView UDV = new UserDataView();
    List<UserProfileView> profiles = GetAllUserProfiles();
    List<LOOKUPAvailableRole> roles = GetAllRoles();

    int? userAssignedRoleID = 0, userID = 0;
    string userGender = string.Empty;

    userID = GetUserID(loginName);
    using (DemoDBEntities db = new DemoDBEntities()) {

```

```

        userAssignedRoleID = db.SYSUserRoles.Where(o => o.SYSUserID ==
userID)?.FirstOrDefault().LOOKUPRoleID;

        userGender = db.SYSUserProfiles.Where(o => o.SYSUserID ==
userID)?.FirstOrDefault().Gender;
    }

    List<Gender> genders = new List<Gender>();
    genders.Add(new Gender { Text = "Male", Value = "M" });
    genders.Add(new Gender { Text = "Female", Value = "F" });

    UDV.UserProfile = profiles;
    UDV.UserRoles = new UserRoles { SelectedRoleID = userAssignedRoleID, UserRoleList
= roles };
    UDV.UserGender = new UserGender { SelectedGender = userGender, Gender = genders
};
    return UDV;
}

```

Metode yang ditunjukkan dari kode di atas cukup jelas sebagai nama metode mereka. Metode utama adalah GetUserDataView () dan apa yang dilakukannya itu mendapat semua profil dan role. UserRoles dan properti UserGender akan digunakan selama editing dan pemutakhiran data pengguna. Nilai-nilai ini untuk mengisi daftar dropdown untuk role dan gender.

#### LANGKAH 3: Menambah ManageUserPartial Action method

Buka class "HomeController" dan menambahkan namespace berikut ini:

```

using System.Web.Security;
using Asp_mvc_2.Models.ViewModel;
using Asp_mvc_2.Models.EntityManager;

```

Dan kemudian menambahkan action method berikut ini:

```

[AuthorizeRoles("Admin")]
public ActionResult ManageUserPartial() {
    if (User.Identity.IsAuthenticated) {
        string loginName = User.Identity.Name;
        UserManager UM = new UserManager();
    }
}

```

```

        UserDataView UDV = UM.GetUserDataView(loginName);
        return PartialView(UDV);
    }

    return View();
}

```

Kode di atas diberi atribut Authorize umum sehingga hanya pengguna admin yang dapat memanggil metode itu. Yang dilakukan adalah memanggil metode GetUserDataView () dengan memberi loginname sebagai parameter dan mengembalikan hasil dalam tampilan parsial.

#### LANGKAH 4: Menambah ManageUserPartial partial view

Sekarang mari kita membuat tampilan parsial. Klik kanan pada metode ManageUserPartial dan pilih "Add View". Ini akan memunculkan dialog berikut:

Ketika akan membuat custom view untuk mengatur user, pilih "**Empty**" template dan pastikan centang opsi "**Create as a partial view**". Click Add dan ketik HTML berikut:

```
@model asp_mvc_2.Models.ViewModel.UserDataView
```

```
<div>
```

```

<h1>List of Users</h1>
<table class="table table-striped table-condensed table-hover">
  <thead>
    <tr>
      <th>ID</th>
      <th>Login ID</th>
      <th>Password</th>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Gender</th>
      <th colspan="2">Role</th>
      <th></th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    @foreach (var i in Model.UserProfile) {
      <tr>
        <td> @Html.DisplayFor(m => i.SYSUserID)</td>
        <td> @Html.DisplayFor(m => i.LoginName)</td>
        <td> @Html.DisplayFor(m => i.Password)</td>
        <td> @Html.DisplayFor(m => i.FirstName)</td>
        <td> @Html.DisplayFor(m => i.LastName)</td>
        <td> @Html.DisplayFor(m => i.Gender)</td>
        <td> @Html.DisplayFor(m => i.RoleName)</td>
        <td> @Html.HiddenFor(m => i.LOOKUPRoleID)</td>
        <td><a href="javascript:void(0)" class="lnkEdit">Edit</a></td>
        <td><a href="javascript:void(0)"
class="lnkDelete">Delete</a></td>
      </tr>
    }
  </tbody>
</table>
</div>

```

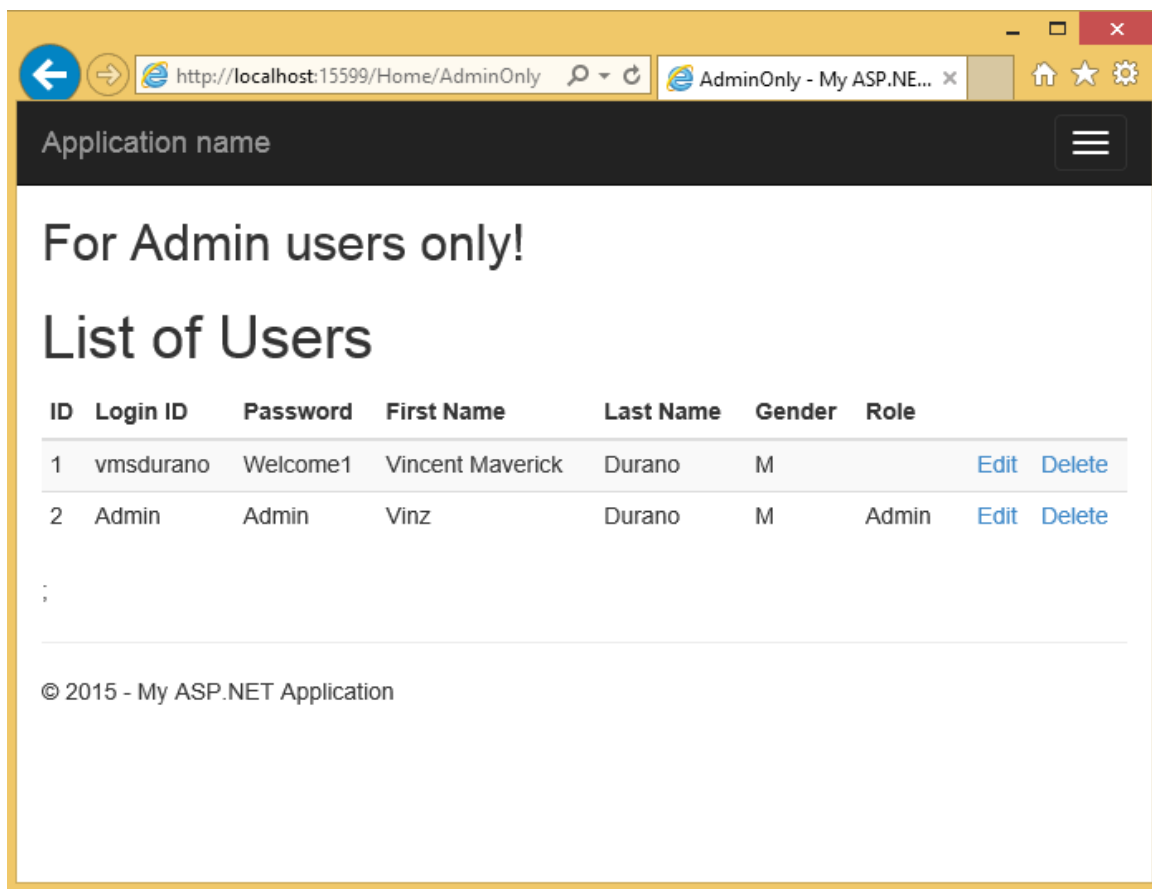
Buka "AdminOnly.cshtml" view dan tambahkan kode berikut:

```
<div id="divUserListContainer">
    @Html.Action("ManageUserPartial", "Home");
</div>
```

Kode diatas untuk menampilkan ManageUserPartial view dalam main view menggunakan Action HTML helper.

LANGKAH 5: Jalankan halaman

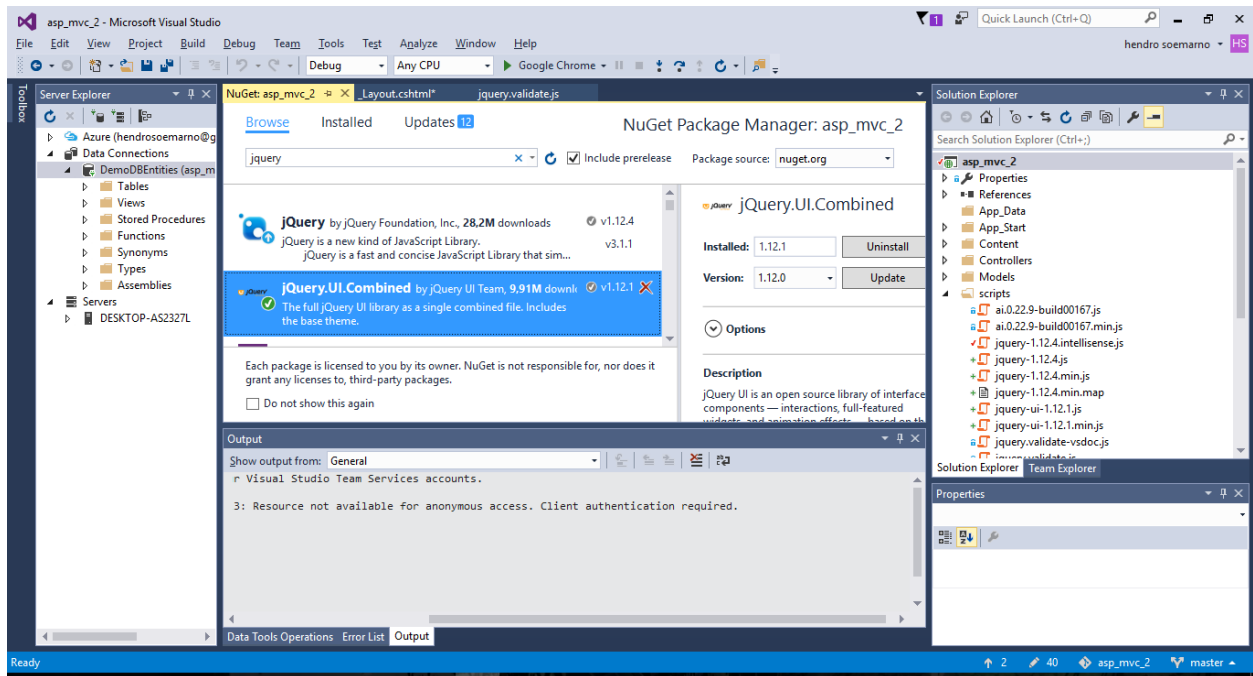
Coba login ke halaman web dan arahkan ke: <http://localhost:15599/Home/AdminOnly> . Keluaran seharusnya seperti ini:



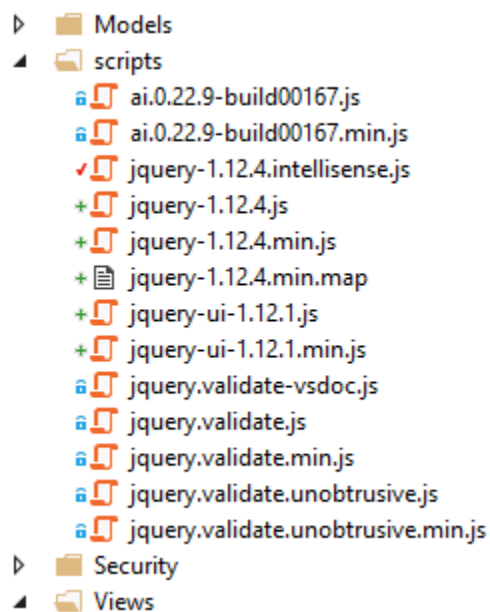
### Edit and Update Data

Untuk menghasilkan dialog box, digunakan **jQueryUI** untuk user mengubah data lalu ditambahkan reference terlebih dahulu. Klik kanan pada project dan pilih "**Manage Nuget Packages**". Dalam search box ketik "jquery" dan pilih "**jQuery.UI.Combined**" seperti gambar dibawah:





Instal jQueryUI library seharusnya ditambahkan langsung pada project didalam folder "Script":



#### LANGKAH 1: Referencing jQueryUI

Pilih Views > Shared > \_Layout.cshtml tambah jQueryUI reference sebagai berikut:

```
<script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
```

jQueryUI harus ditambahkan pada jQuery setelah jQueryUI menggunakan core jQuery library.

Tambahkan jQueryUI CSS reference:

```
<link href="~/Content/themes/base/all.css" rel="stylesheet" />
```

#### LANGKAH 2: Menambah UpdateUserAccount() method

Perlu diingat bahwa contoh ini dimaksudkan untuk membuat sebuah aplikasi sesederhana mungkin. Dalam skenario nyata yang kompleks, sangat disarankan Anda untuk menggunakan pola Repositori dan Unit kerja untuk lapisan akses data.

Tambahkan kode berikut di bawah ini dalam kelas "UserManager":

```
public void UpdateUserAccount(UserProfileView user) {

    using (DemoDBEntities db = new DemoDBEntities()) {
        using (var dbContextTransaction = db.Database.BeginTransaction()) {
            try {

                SYSUser SU = db.SYSUsers.Find(user.SYSUserID);
                SU.LoginName = user.LoginName;
                SU.PasswordEncryptedText = user.Password;
                SU.RowCreatedSYSUserID = user.SYSUserID;
                SU.RowModifiedSYSUserID = user.SYSUserID;
                SU.RowCreatedDateTime = DateTime.Now;
                SU.RowModifiedDateTime = DateTime.Now;

                db.SaveChanges();

                var userProfile = db.SYSUserProfiles.Where(o => o.SYSUserID ==
user.SYSUserID);
                if (userProfile.Any()) {
                    SYSUserProfile SUP = userProfile.FirstOrDefault();
                    SUP.SYSUserID = SU.SYSUserID;
                }
            }
        }
    }
}
```

```

        SUP.FirstName = user.FirstName;
        SUP.LastName = user.LastName;
        SUP.Gender = user.Gender;
        SUP.RowCreatedSYSUserID = user.SYSUserID;
        SUP.RowModifiedSYSUserID = user.SYSUserID;
        SUP.RowCreatedDateTime = DateTime.Now;
        SUP.RowModifiedDateTime = DateTime.Now;

        db.SaveChanges();
    }

    if (user.LOOKUPRoleID > 0) {
        var userRole = db.SYSUserRoles.Where(o => o.SYSUserID ==
user.SYSUserID);
        SYSUserRole SUR = null;
        if (userRole.Any()) {
            SUR = userRole.FirstOrDefault();
            SUR.LOOKUPRoleID = user.LOOKUPRoleID;
            SUR.SYSUserID = user.SYSUserID;
            SUR.IsActive = true;
            SUR.RowCreatedSYSUserID = user.SYSUserID;
            SUR.RowModifiedSYSUserID = user.SYSUserID;
            SUR.RowCreatedDateTime = DateTime.Now;
            SUR.RowModifiedDateTime = DateTime.Now;
        }
        else {
            SUR = new SYSUserRole();
            SUR.LOOKUPRoleID = user.LOOKUPRoleID;
            SUR.SYSUserID = user.SYSUserID;
            SUR.IsActive = true;
            SUR.RowCreatedSYSUserID = user.SYSUserID;
            SUR.RowModifiedSYSUserID = user.SYSUserID;
            SUR.RowCreatedDateTime = DateTime.Now;
            SUR.RowModifiedDateTime = DateTime.Now;
            db.SYSUserRoles.Add(SUR);
        }
    }

```

```
        db.SaveChanges();
    }
    dbContextTransaction.Commit();
}
catch {
    dbContextTransaction.Rollback();
}
}
}
```

Metode di atas mengambil objek UserProfileView sebagai parameter. Yang dilakukannya itu pertama mengeluarkan permintaan ke database menggunakan sintaks LINQ untuk mendapatkan data pengguna tertentu dengan melewati SYSUserID. Kemudian update objek SYSUser dengan data yang sesuai dari objek UserProfileView. Query kedua mendapat data SYSUserProfiles terkait dan kemudian meng-update nilai-nilai yang sesuai. Setelah itu kemudian terlihat untuk LOOKUPRoleID terkait untuk pengguna tertentu. Jika pengguna tidak memiliki peran yang ditugaskan untuk itu kemudian ia menambahkan data baru untuk database, hanya memperbarui tabel.

Transaksi sederhana dalam metode tersebut. Hal ini karena meja SYSUser, SYSUserProfile dan SYSUserRole cukup banyak terkait satu sama lain dan memastikan bahwa hanya melakukan perubahan ke database jika operasi untuk setiap tabel berhasil. Database.BeginTransaction () hanya tersedia di EF 6 dan seterusnya.

### LANGKAH 3: Adding the UpdateUserData Action method

Tambahkan kode berikut dalam kelas "HomeController":

```
[AuthorizeRoles("Admin")]

public ActionResult UpdateUserData(int userID, string loginName, string password,
string firstName, string lastName, string gender, int roleID = 0) {
    UserProfileView UPV = new UserProfileView();
    UPV.SYSUserID = userID;
    UPV.LoginName = loginName;
    UPV.Password = password;
    UPV.FirstName = firstName;
    UPV.LastName = lastName;
```

```

UPV.Gender = gender;

if (roleId > 0)
    UPV.LOOKUPRoleId = roleId;

userManager UM = new userManager();
UM.UpdateUserAccount(UPV);

return Json(new { success = true });
}

```

Metode di atas bertanggung jawab untuk mengumpulkan data yang dikirim dari tampilan untuk pembaruan. Ini kemudian memanggil metode UpdateUserAccount () dan melawatkan model tampilan UserProfileView sebagai parameter. UpdateUserData() metode akan dipanggil melalui permintaan AJAX.

#### LANGKAH 4: Mengubah ManageUserPartial view

Sekarang tambahkan HTML berikut tampilan "ManageUserPartial.cshtml":

```

<div id="divEdit" style="display:none">
    <input type="hidden" id="hidID" />
    <table>
        <tr>
            <td>Login Name</td>
            <td><input type="text" id="txtLoginName" class="form-control" /></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="text" id="txtPassword" class="form-control" /></td>
        </tr>
        <tr>
            <td>First Name</td>
            <td><input type="text" id="txtFirstName" class="form-control" /></td>
        </tr>
        <tr>
            <td>Last Name</td>

```

```

        <td><input type="text" id="txtLastName" class="form-control" /></td>
    </tr>
    <tr>
        <td>Gender</td>
        <td>@Html.DropDownListFor(o => o.UserGender.SelectedGender,
            new SelectList(Model.UserGender.Gender, "Value", "Text"),
            "",
            new { id = "ddlGender", @class="form-control" })
        </td>
    </tr>
    <tr>
        <td>Role</td>
        <td>@Html.DropDownListFor(o => o.UserRoles.SelectedRoleID,
            new SelectList(Model.UserRoles.UserRoleList, "LOOKUPRoleID",
"RoleName"),
            "",
            new { id = "ddlRoles", @class="form-control" })
        </td>
    </tr>
</table>
</div>

```

#### LANGKAH 5: Menyatukan jQuery dan jQuery AJAX

Sebelum kita pergi ke pelaksanaan itu penting untuk mengetahui apa teknologi ini:

- jQuery adalah ringan dan kaya fitur library JavaScript yang memungkinkan manipulasi DOM, bahkan penanganan, animasi dan Ajax jauh lebih sederhana dengan API yang kuat yang bekerja di semua browser utama.
- jQueryUI menyediakan satu set UI interaksi, efek, widget dan tema dibangun di atas jQuery library.
- jQuery AJAX memungkinkan Anda untuk menggunakan fungsi dan metode untuk berkomunikasi dengan data dari server dan beban data ke klien / Browser.

Sekarang beralih kembali ke "ManageUserPartial" melihat dan menambahkan blok script berikut di bagian paling bawah:

```

<script type="text/javascript">
    $(function () {

```

```

var initDialog = function (type) {
    var title = type;
    $("#divEdit").dialog({
        autoOpen: false,
        modal: true,
        title: type + ' User',
        width: 360,
        buttons: {
            Save: function () {
                var id = $("#hidID").val();
                var role = $("#ddlRoles").val();
                var loginName = $("#txtLoginName").val();
                var loginPass = $("#txtPassword").val();
                var fName = $("#txtFirstName").val();
                var lName = $("#txtLastName").val();
                var gender = $("#ddlGender").val();

                UpdateUser(id, loginName, loginPass, fName, lName, gender,
role);

                $(this).dialog("destroy");
            },
            Cancel: function () { $(this).dialog("destroy"); }
        }
    });
}

function UpdateUser(id, logName, logPass, fName, lName, gender, role) {
    $.ajax({
        type: "POST",
        url: "@(Url.Action("UpdateUserData","Home"))",
        data: { userID: id, loginName: logName, password: logPass, firstName:
fName, lastName: lName, gender: gender, roleID: role },
        success: function (data) {

$("#divUserListContainer").load("@(Url.Action("ManageUserPartial","Home", new {
status ="update" })))");

```

```

        },
        error: function (error) {
            //to do:
        }
    });
}

$("a.lnkEdit").on("click", function () {
    initDialog("Edit");
    $(".alert-success").empty();
    var row = $(this).closest('tr');

    $("#hidID").val(row.find("td:eq(0)").html().trim());
    $("#txtLoginName").val(row.find("td:eq(1)").html().trim());
    $("#txtPassword").val(row.find("td:eq(2)").html().trim());
    $("#txtFirstName").val(row.find("td:eq(3)").html().trim());
    $("#txtLastName").val(row.find("td:eq(4)").html().trim());
    $("#ddlGender").val(row.find("td:eq(5)").html().trim());
    $("#ddlRoles").val(row.find("td:eq(7) > input").val().trim());

    $("#divEdit").dialog("open");
    return false;
});
});
</script>

```

Variabel `initDialog` menginisialisasi dialog jQueryUI dengan menyesuaikan dialog. Disesuaikan dengan menambahkan tombol SAVE dan Cancel untuk menulis implementasi kode yang dapat diatur untuk setiap kebutuhan. Dalam fungsi Simpan kami ekstrak setiap nilai dari form edit dan memberikan nilai-nilai ini ke fungsi `UpdateUser()`.

Fungsi `UpdateUser ()` mengeluarkan permintaan AJAX menggunakan jQuery AJAX. "Type" parameter menunjukkan metode apa bentuk permintaan membutuhkan, dalam hal ini kita menetapkan jenisnya sebagai "POST". "url" adalah jalan untuk metode controller yang kita buat di LANGKAH 3. Perhatikan bahwa nilai url dapat menjadi layanan web, web API atau apapun yang menjadi tuan rumah data Anda. "Data" adalah di mana kita memberikan nilai pada metode yang memerlukan parameter. Jika metode di server tidak memerlukan parameter apapun maka Anda dapat meninggalkan ini sebagai kosong "{}". "Keberhasilan" fungsi biasanya digunakan



ketika Anda melakukan proses tertentu jika permintaan di server berhasil. Dalam hal ini kami memuat pandangan sebagian untuk mencerminkan perubahan pada tampilan setelah kami memperbarui data. Perlu diingat bahwa kita mengirimkan parameter baru untuk aksi "ManageUserPartial" yang menunjukkan status permintaan.

Fungsi terakhir adalah di mana kita membuka dialog ketika pengguna mengklik pada link "edit" dari grid. Ini juga di mana kita mengekstrak data dari grid menggunakan penyeleksi jQuery dan mengisi bidang dialog dengan data yang diambil.

#### LANGKAH 6: Modifying the ManageUserPartial Action method

Jika Anda ingat, kami menambahkan parameter "status" baru dengan metode ManageUserPartial dalam permintaan AJAX kami, jadi kami perlu memperbarui metode tanda tangan untuk menerima parameter. Metode baru sekarang harus terlihat seperti ini:

```
[AuthorizeRoles("Admin")]
public ActionResult ManageUserPartial(string status = "") {
    if (User.Identity.IsAuthenticated) {
        string loginName = User.Identity.Name;
        UserManager UM = new UserManager();
        UserDataView UDV = UM.GetUserDataView(loginName);

        string message = string.Empty;
        if (status.Equals("update"))
            message = "Update Successful";
        else if (status.Equals("delete"))
            message = "Delete Successful";

        ViewBag.Message = message;

        return PartialView(UDV);
    }

    return RedirectToAction("Index", "Home");
}
```

#### LANGKAH 7: Displaying the Status result

Jika Anda perhatikan, kita menciptakan string pesan berdasarkan operasi tertentu dan menyimpan hasilnya dalam ViewBag. Hal ini untuk membiarkan pengguna melihat apakah

operasi tertentu berhasil. Sekarang tambahkan markup berikut di bawah ini dalam view "ManageUserPartial":

```
<span class="alert-success">@ViewBag.Message</span>
```

LANGKAH 8: Running the page

Berikut adalah output di bawah ini:

Setelah mengklik dialog mengedit

The screenshot shows a web browser window with the URL `http://localhost:15599/Home/AdminOnly`. The page displays an 'Admin Only' interface with a 'List of Users' table. An 'Edit User' dialog box is open, allowing modification of user details.

ID	Login ID	Password
1	vmsdurano	Welcome1
2	Admin	Admin

© 2015 - My ASP.NET A

**Edit User**

Login Name: vmsdurano

Password: Welcome1

First Name: Vincent Maverick

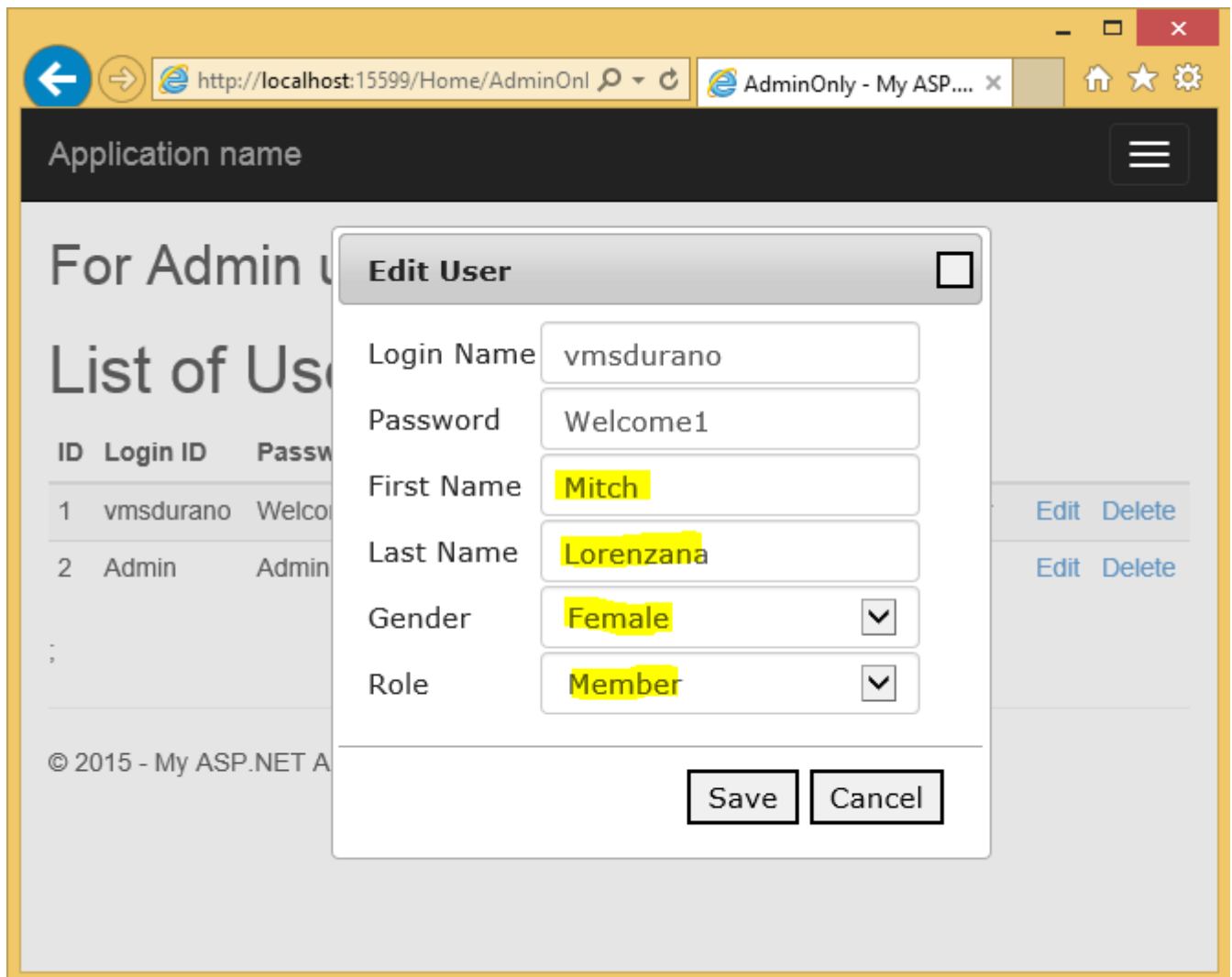
Last Name: Durano

Gender: Male

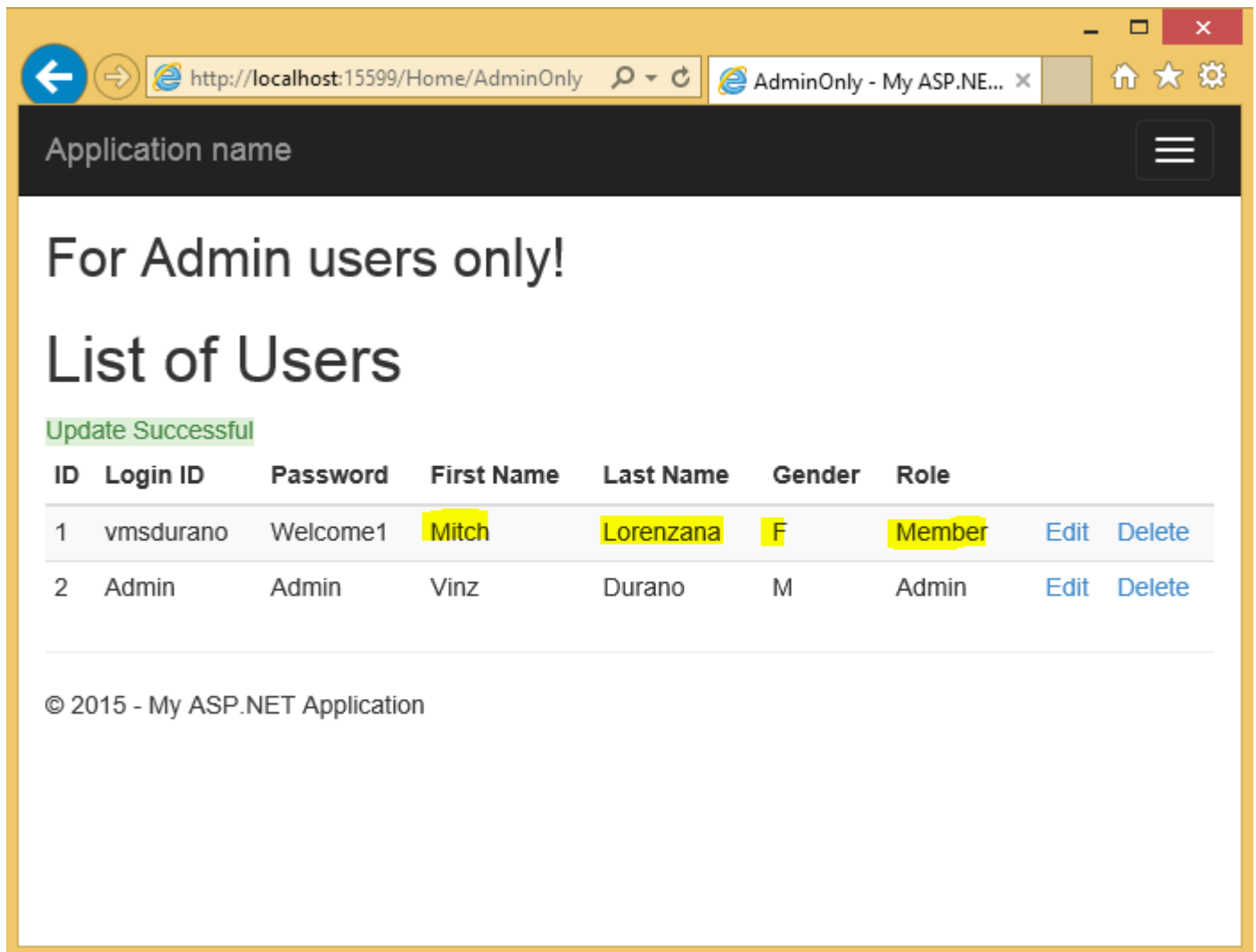
Role:

Save Cancel

Mengubah data



Setelah mengubah data



The screenshot shows a web browser window with the address bar displaying `http://localhost:15599/Home/AdminOnly`. The page has a dark header with the text "Application name" and a hamburger menu icon. Below the header, the main content area displays "For Admin users only!" and "List of Users". A green message "Update Successful" is visible above a table of users. The table has columns for ID, Login ID, Password, First Name, Last Name, Gender, and Role. There are two rows of user data. The first row has values: 1, vmsdurano, Welcome1, Mitch, Lorenzana, F, Member. The second row has values: 2, Admin, Admin, Vinz, Durano, M, Admin. Each row has "Edit" and "Delete" links. At the bottom, there is a copyright notice: "© 2015 - My ASP.NET Application".

ID	Login ID	Password	First Name	Last Name	Gender	Role		
1	vmsdurano	Welcome1	Mitch	Lorenzana	F	Member	Edit	Delete
2	Admin	Admin	Vinz	Durano	M	Admin	Edit	Delete

Jika Anda telah sampai sejauh ini maka selamat, Anda sekarang siap untuk langkah berikutnya. Sekarang turun ke bagian terakhir dari seri ini. :)

## Deleting Data

LANGKAH 1: Adding the `DeleteUser()` method

Menambahkan metode berikut di kelas "userManager":

```
public void DeleteUser(int userID) {  
    using (DemoDBEntities db = new DemoDBEntities()) {  
        using (var dbContextTransaction = db.Database.BeginTransaction()) {  
            try {
```

```

        var SUR = db.SYSUserRoles.Where(o => o.SYSUserID == userID);
        if (SUR.Any()) {
            db.SYSUserRoles.Remove(SUR.FirstOrDefault());
            db.SaveChanges();
        }

        var SUP = db.SYSUserProfiles.Where(o => o.SYSUserID == userID);
        if (SUP.Any()) {
            db.SYSUserProfiles.Remove(SUP.FirstOrDefault());
            db.SaveChanges();
        }

        var SU = db.SYSUsers.Where(o => o.SYSUserID == userID);
        if (SU.Any()) {
            db.SYSUsers.Remove(SU.FirstOrDefault());
            db.SaveChanges();
        }

        dbContextTransaction.Commit();
    }
    catch {
        dbContextTransaction.Rollback();
    }
}
}
}

```

metode di atas menghapus catatan untuk pengguna tertentu di tabel SYSUserRole, SYSUserProfile dan SYSUser.

**LANGKAH 2:** Adding the DeleteUser Action method

Tambahkan kode berikut dalam kelas "HomeController":

```

[AuthorizeRoles("Admin")]
public ActionResult DeleteUser(int userID) {
    UserManager UM = new UserManager();

```

```
UM.DeleteUser(userID);  
return Json(new { success = true });  
}
```

#### LANGKAH 3: Integrating jQuery and jQuery AJAX

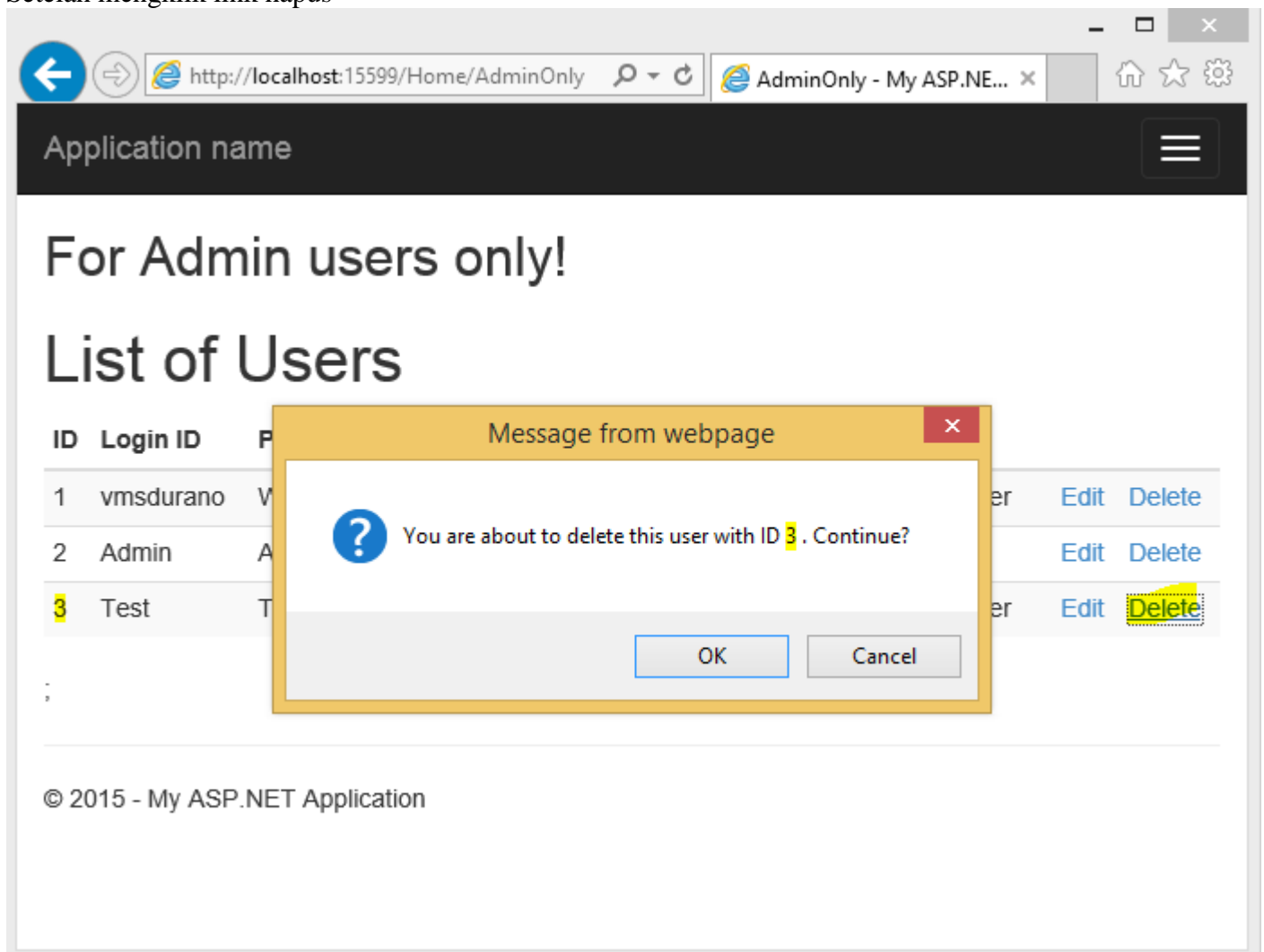
Tambahkan script berikut dalam tag script di "ManageUserPartial" lihat:

```
function DeleteUser(id) {  
    $.ajax({  
        type: "POST",  
        url: "@(Url.Action("DeleteUser","Home"))",  
        data: { userID: id },  
        success: function (data) {  
            $("#divUserListContainer").load("@(Url.Action("ManageUserPartial","Home", new  
{ status = "delete" })))");  
        },  
        error: function (error) { }  
    });  
}  
  
$("a.lnkDelete").on("click", function () {  
    var row = $(this).closest('tr');  
    var id = row.find("td:eq(0)").html().trim();  
    var answer = confirm("You are about to delete this user with ID " + id + " .  
Continue?");  
    if (answer)  
        DeleteUser(id);  
    return false;  
});
```

#### LANGKAH 4: Running the page

Berikut adalah output di bawah ini:

Setelah mengklik link hapus



Setelah menghapus

The screenshot shows a web browser window with the address bar displaying `http://localhost:15599/Home/AdminOnly`. The page has a dark header with the text "Application name" and a hamburger menu icon. The main content area displays the message "For Admin users only!" followed by the heading "List of Users". A green message "Delete successful" is shown above a table of users. The table has columns for ID, Login ID, Password, First Name, Last Name, Gender, and Role. There are two rows of user data. Each row has "Edit" and "Delete" links. Below the table is a semicolon ";" and a footer that reads "© 2015 - My ASP.NET Application".

Application name

For Admin users only!

## List of Users

Delete successful

ID	Login ID	Password	First Name	Last Name	Gender	Role		
1	vmsdurano	Welcome1	Mitch	Lorenzana	F	Member	Edit	Delete
2	Admin	Admin	Vincent Maverick	Durano	M	Admin	Edit	Delete

;

© 2015 - My ASP.NET Application