

## Hands On Labs

# ASP.NET MVC

Pada HOL kali ini akan dibuat sebuah program sederhana untuk menampilkan daftar katalog products menggunakan ASP.NET MVC. Untuk mengakses database. Adapun software yang harus diinstal untuk menjalankan HOL ini adalah:

- Visual Studio 2008 SP1
- ASP.NET MVC 1.0
- SQL Server Express 2005
- JQuery 1.3
- JQuery UI 1.7.1

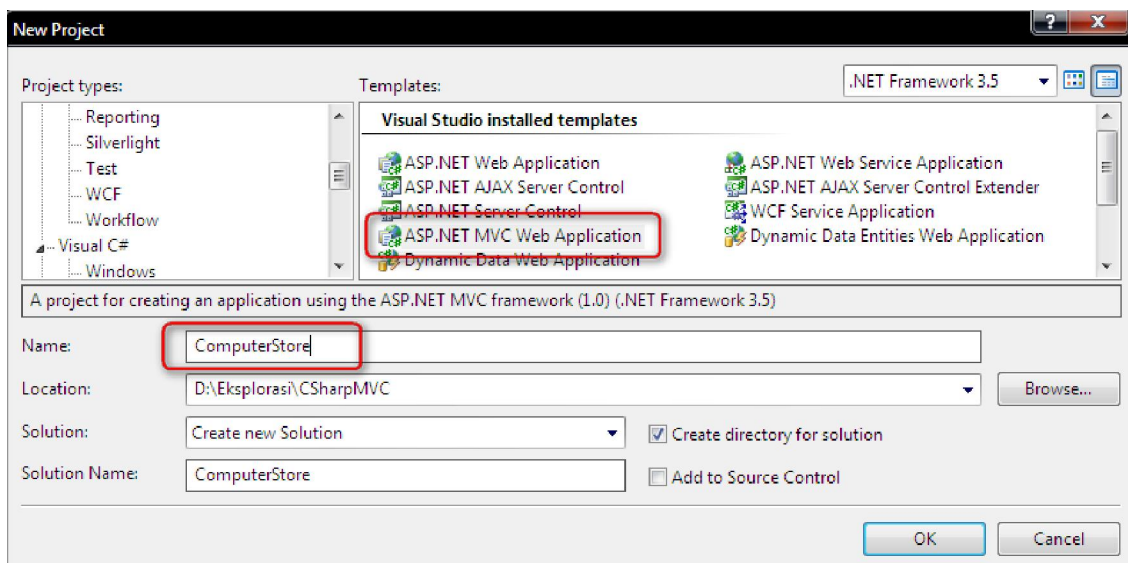
Topik yang akan dibahas pada contoh ini:

- Membuat Model (LINQ to SQL)
- Membuat Class Repository
- Membuat Controller
- Membuat View (Helper, Validation)
- Simple JQuery Integration

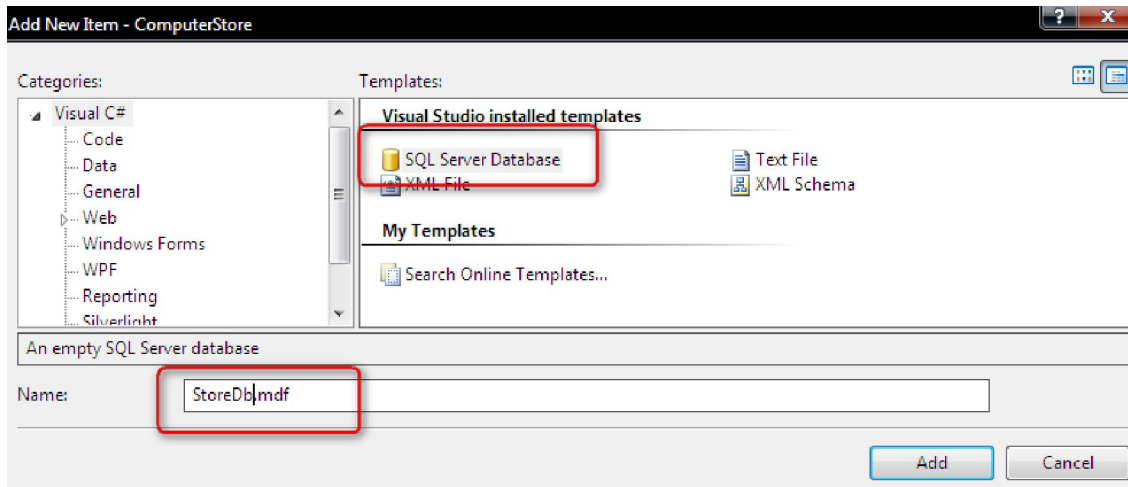
### *Membuat DataSource*

Pertama kita akan membuat database yang akan kita gunakan dalam aplikasi ini, untuk itu ikuti langkah-langkah dibawah ini:

1. Buka VS 2008, kemudian buat project ASP.NET MVC, beri nama "ComputerStore"



2. Kemudian pada Solution Explorer, klik kanan folder "App\_Data"



3. Kemudian pada database tambahkan table dengan nama "Products", dan tambahkan field-field berikut pada table "Products".

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(50)	<input type="checkbox"/>
	Description	varchar(MAX)	<input checked="" type="checkbox"/>
	Price	money	<input type="checkbox"/>
	Qty	smallint	<input type="checkbox"/>
	BuyDate	datetime	<input type="checkbox"/>

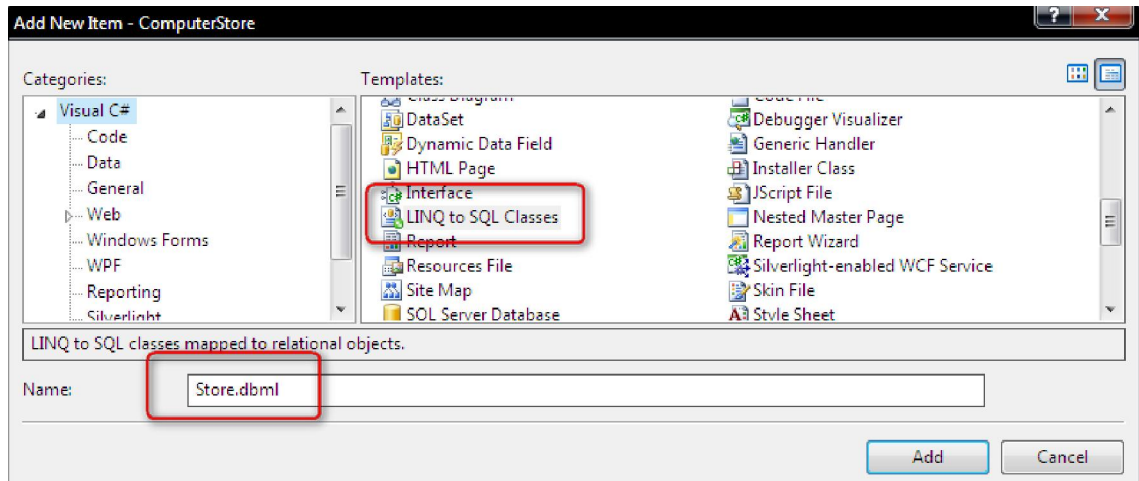
4. Pada field "Id" ubah property "Identity Specification" menjadi "Yes" untuk menambahkan auto increment pada field tersebut.

<input checked="" type="checkbox"/>	Identity Specification	Yes
	(Is Identity)	<input checked="" type="checkbox"/>
	Identity Increment	1
	Identity Seed	1

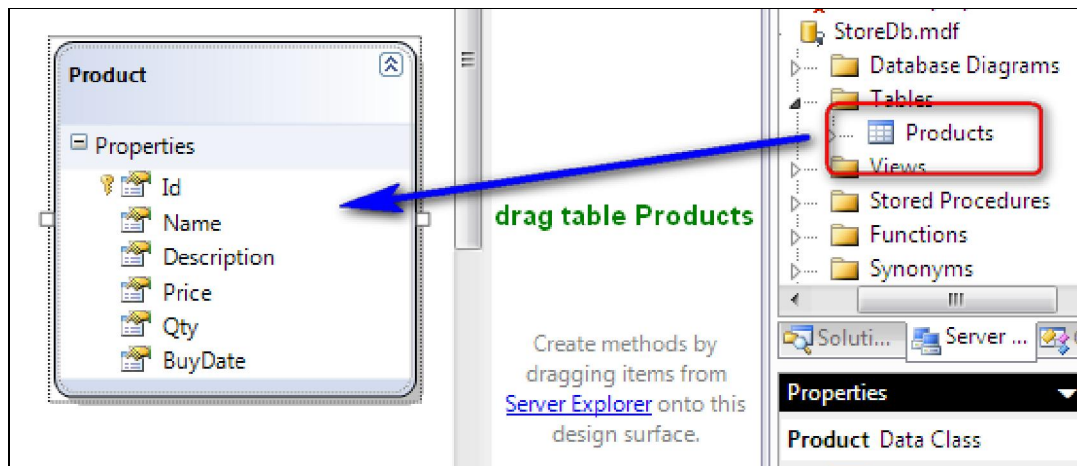
### *Membuat Data Model (LINQ to SQL)*

Setelah kita selesai membuat database, sekarang saatnya melakukan mapping table yang sudah kita buat menjadi object model dengan menggunakan LINQ to SQL. Object model ini akan kita simpan di dalam folder "Model". Pada aplikasi MVC model berisi semua application logic (business logic, data access logic).

1. Pada Solution Explorer, klik kanan folder "Model", kemudian pilih "Add" – "New Item" – pilih "LINQ to SQL Classes" untuk membuat object model (mapping) secara otomatis. Anda dapat juga menggunakan "SQL Metal" atau melakukan mapping table secara manual.



2. Beri nama "Store.dbml". Kemudian drag table "Products" dari server explorer kedalam LINQ to SQL Diagram. Kemudian save.



### *Membuat Product Repository*

Pada small application tidak menjadi masalah jika kita mengakses LINQ to SQL DataContext class langsung dari Controller, namun jika aplikasinya bertambah kompleks maka akan sulit untuk di maintain dan melakukan tes terhadap kode-kode tersebut, akan terjadi duplikasi banyak code pada aplikasi kita (berlawanan dengan prinsip MVC yaitu DRY = Don't Repeat Yourself).

Teknik yang dapat digunakan untuk membuat aplikasi kita lebih mudah di maintain adalah dengan menggunakan "repository pattern" yaitu membuat repository class yang mengenkapsulasi query dan persistence logic sehingga aplikasi kita menjadi lebih clean.

1. Tambahkan class baru kedalam folder "Model", beri nama "ProductRepository.cs". kemudian tulis kode berikut:

```
public class ProductRepository {
    StoreDataContext db = new StoreDataContext();

    //menampilkan semua product, digunakan pada View Index
    public IQueryable<Product> ShowAllProduct() {
```

```

        var query = from p in db.Products
                    orderby p.Id
                    select p;
        return query;
    }

    //menambah product baru, digunakan pada View Create
    public void CreateProduct(Product productToCreate) {
        db.Products.InsertOnSubmit(productToCreate);
    }

    //mengambil product berdasarkan id tertentu
    public Product ShowProduct(int id) {
        Product prod = db.Products.Where(p => p.Id ==
            id).Single<Product>();
        return prod;
    }

    //mengedit product
    public void EditProduct(Product productToEdit) {
        Product prod = db.Products.Where(p => p.Id ==
            productToEdit.Id).Single<Product>();
        prod.Name = productToEdit.Name;
        prod.Price = productToEdit.Price;
        prod.Qty = productToEdit.Qty;
        prod.BuyDate = productToEdit.BuyDate;
    }

    //menghapus product
    public void DeleteProduct(int id) {
        Product prod = db.Products.Where(p => p.Id ==
            id).Single<Product>();
        db.Products.DeleteOnSubmit(prod);
    }

    public void Save() {
        db.SubmitChanges();
    }
}

```

2. StoreDataContext adalah class yang digenerate oleh tools LINQ to SQL Classes, yang berisi object model dari table Products. Untuk mengakses data maka kita menggunakan standar query expression dari LINQ. Query expression tersebut akan diterjemahkan oleh LINQ to SQL menjadi T-SQL sintaks untuk dieksekusi oleh database.

### *Membuat Controller*

Sesudah membuat Model, kemudian langkah selanjutnya adalah membuat Controller. Pada aplikasi MVC, Controller Berisi Control-flow logic. Controller berinteraksi dengan Model dan View untuk mengontrol jalannya aplikasi.

1. Pada solution explorer klik kanan folder "Controller", pilih "Add" – "Controller", beri nama "Home Controller", kemudian check pada pilihan "Add action method for create, update, and details scenario" untuk menambahkan secara otomatis ActionMethod yang akan kita buat untuk handle proses CRUD (create, read, update, dan delete) .

2. Kemudian tambahkan kode berikut pada Controller:

```
public class HomeController : Controller
{
    private ProductRepository productRepo = new
        ProductRepository();

    // GET: /Home/
    public ActionResult Index() {
        return View(productRepo.ShowAllProduct());
    }

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

    // GET: /Home/Details/5
    public ActionResult Details(int id)
    {
        return View();
    }

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

    // POST: /Home/Create
    [AcceptVerbs(HttpVerbs.Post)]
    public ActionResult Create([Bind(Exclude="Id")] Product
        productToCeate)
    {
        if (ModelState.IsValid) {
            try {
                // TODO: Add insert logic here
                productRepo.CreateProduct(productToCeate);
                productRepo.Save();
                return RedirectToAction("Index");
            }
            catch {}
        }
        return View(productToCeate);
    }

    // GET: /Home/Edit/5
    public ActionResult Edit(int id)
    {
        return View(productRepo.ShowProduct(id));
    }

    // POST: /Home/Edit/5
    [AcceptVerbs(HttpVerbs.Post)]
    public ActionResult Edit(Product productToEdit)
```

```

    {
        if (ModelState.IsValid) {
            try {
                // TODO: Add update logic here
                productRepo.EditProduct(productToEdit);
                productRepo.Save();
                return RedirectToAction("Index");
            }
            catch {}
        }
        return View(productToEdit);
    }

    public ActionResult Delete(int id) {
        Product prod = productRepo.ShowProduct(id);
        return View(prod);
    }

    [AcceptVerbs(HttpVerbs.Post)]
    public ActionResult Delete(int id, string confirmButton) {
        try {
            productRepo.DeleteProduct(id);
            productRepo.Save();
        }
        catch {}
        return View("Deleted");
    }
}

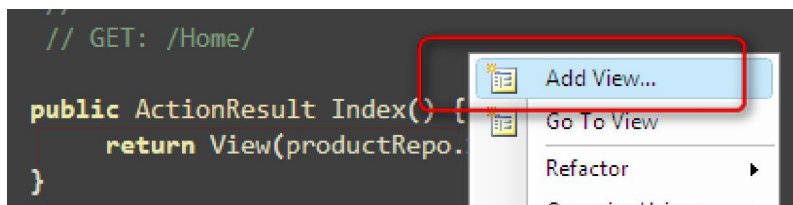
```

### *Membuat View*

View Berisi HTML Markup dan View Logic, digunakan untuk menampilkan isi dari Model.

#### **View Index**

1. Pertama kita akan membuat View yang digunakan untuk menampilkan seluruh Product kedalam table. View ini akan dijalankan pertama kali ketika kita menjalankan aplikasi
2. Anda dapat membuat View dengan cara manual tapi dengan editor VS 2008 anda dapat menggunakan fitur yang mirip "scaffold" untuk membuat View dengan cara cepat.
3. Masuk ke halaman Controller, kemudian klik kanan di bagian ActionResult "Index" – pilih "Add View"



4. Pada tampilan "Add View", View name akan secara otomatis sesuai dengan nama ActionResult di Controllernya, check pada "Create a strongly typed view" kemudian pilih "ComputerStore.Models.Product". Pada View Content pilih "Details", jangan lupa juga untuk check pada pilihan "Select Master Page" untuk menggunakan Master Page yang sudah tersedia.

The screenshot shows the 'Add View' dialog box with the following configuration:

- View name: Index
- Create a partial view (.ascx)
- Create a strongly-typed view
  - View data class: ComputerStore.Models.Product
  - View content: Details
- Select master page
  - ~/Views/Shared/Site.Master
- ContentPlaceHolder ID: MainContent

5. Maka secara otomatis Visual Studio akan mengcreate View berdasarkan Model yang anda gunakan.

```

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent"
runat="server">
  <h2>Product Details</h2>
  <table>
    <tr>
      <th></th>
      <th>Id</th>
      <th>Name</th>
      <th>Description</th>
      <th>Price</th>
      <th>Qty</th>
      <th>BuyDate</th>
    </tr>

    <% foreach (var item in Model) { %>
      <tr>
        <td>
          <%= Html.ActionLink("Edit", "Edit", new { id = item.Id })%> |
          <%= Html.ActionLink("Delete", "Delete", new { id = item.Id })%>
        </td>
        <td>
          <%= Html.Encode(item.Id) %>
        </td>
        <td>
          <%= Html.Encode(item.Name) %>
        </td>
        <td>
          <%= Html.Encode(item.Description) %>
        </td>
        <td>
          <%= Html.Encode(String.Format("{0:F}", item.Price)) %>
        </td>
      </tr>
    </td>
  </table>

```

```

        <td>
            <%= Html.Encode(item.Qty) %>
        </td>
        <td>
            <%= Html.Encode(String.Format("{0:d}", item.BuyDate)) %>
        </td>
    </tr>
<% } %>
</table>
<p>
    <%= Html.ActionLink("Create New", "Create") %>
</p>
</asp:Content>

```

6. Jika program dijalankan maka di halaman pertama "Home/Index" akan ditampilkan list Product sebagai berikut.

**Product Details**

	Id	Name	Description	Price	Qty	BuyDate
<a href="#">Edit</a> <a href="#">Delete</a>	7	Motherboard Asus TXBX	Seri terbaru motherboad asus dengan 10 lapisan karbon	20000000.00	4	4/5/2009

[Create New](#)

### View Create

1. Sekarang kita akan membuat View untuk Create / Insert data, caranya sama dengan cara yang sudah kita lakukan untuk membuat Index View diatas.
2. Masuk ke HomeController.cs, Klik kanan pada ActionResult "Create" - kemudian pilih 'Add View' – check "Create a strongly-typed view" – pada View Content pilih "Create".

3. Maka secara otomatis VS akan membuatkan View yang berisi form untuk insert data Product, beserta validasinya.

**<h2>Insert Product</h2>**

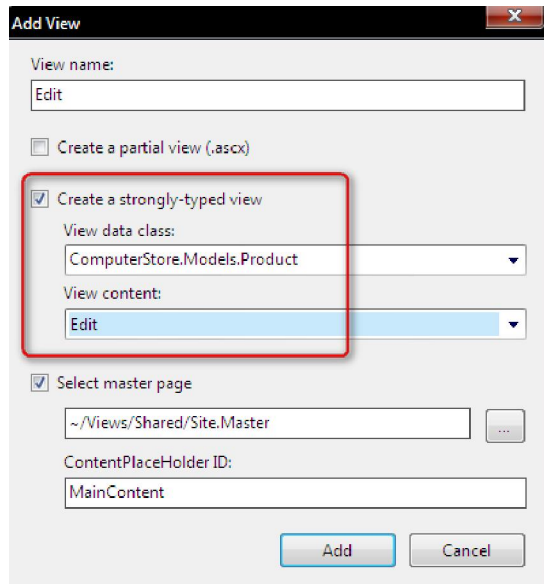
```

<%= Html.ValidationSummary("Create was unsuccessful. Please correct
the errors and try again.") %>
<% using (Html.BeginForm()) {%>
    <fieldset>
        <legend>Product Fields</legend>
        <p>
            <label for="Name">Name: </label >
            <%= Html.TextBox("Name") %>
            <%= Html.ValidationMessage("Name", "") %>
        </p>
        <p>
            <label for="Description">Description: </label >
            <%= Html.TextArea("Description") %>
            <%= Html.ValidationMessage("Description", "") %>
        </p>
        <p>
            <label for="Price">Price: </label >
            <%= Html.TextBox("Price") %>
            <%= Html.ValidationMessage("Price", "") %>
        </p>
        <p>
            <label for="Qty">Qty: </label >
            <%= Html.TextBox("Qty") %>
            <%= Html.ValidationMessage("Qty", "") %>
        </p>
        <p>
            <label for="BuyDate">BuyDate: </label >
            <%= Html.TextBox("BuyDate") %>
            <%= Html.ValidationMessage("BuyDate", "") %>
        </p>
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
<% } %>

```

### View Edit

1. Langkah selanjutnya buat View untuk menangani Edit data Product.
2. Masuk ke HomeController.cs, kemudian klik kanan pada ActionResult "Edit" – pilih "Add View" – check pada "Create a strongly-typed view" – pada View Content pilih "Edit".



3. Secara otomatis Visual Studio akan membuatkan Form untuk edit data pada View Edit.

```

<h2>Edit Product</h2>
<%= Html.ValidationSummary("Edit was unsuccessful. Please correct the
errors and try again.") %>

<% using (Html.BeginForm()) {%>
    <fieldset>
        <legend>Product Fields</legend>
        <p>
            <label for="Id">Id: </label >
            <%= Html.TextBox("Id", Model.Id, new { @readonly="true" })%>
            <%= Html.ValidationMessage("Id", "") %>
        </p>
        <p>
            <label for="Name">Name: </label >
            <%= Html.TextBox("Name", Model.Name) %>
            <%= Html.ValidationMessage("Name", "") %>
        </p>
        <p>
            <label for="Description">Description: </label >
            <%= Html.TextBox("Description", Model.Description) %>
            <%= Html.ValidationMessage("Description", "") %>
        </p>
        <p>
            <label for="Price">Price: </label >
            <%= Html.TextBox("Price", String.Format("{0:F}",
Model.Price)) %>
            <%= Html.ValidationMessage("Price", "") %>
        </p>
        <p>
            <label for="Qty">Qty: </label >
            <%= Html.TextBox("Qty", Model.Qty) %>
            <%= Html.ValidationMessage("Qty", "") %>
        </p>
    </fieldset>
} %>

```

```

        <p>
            <label for="BuyDate">BuyDate:</label >
            <%= Html.TextBox("BuyDate", String.Format("{0:d}",
Model.BuyDate)) %>
            <%= Html.ValidationMessage("BuyDate", "") %>
        </p>
        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
<% } %>

```

### View Delete

1. Terakhir buat View untuk menangani proses Delete data.
2. Masuk ke HomeController.cs, klik kanan pada ActionResult "Delete" – pilih "Add View" – kemudian check "Create a strongly-typed view" – pada View Content pilih "Empty".

3. Tuliskan kode berikut pada Delete View.

```
<h2>Delete Confirmation</h2>
```

```

<div>
  <p>Please confirm you want to deleted Product :
  <i> <%=Html.Encode(Model.Name) %>? </i> </p>
</div>
<% using (Html.BeginForm()) { %>
  <input name="confirmButton" type="submit" value="Delete" />
<% } %>

```

4. Kemudian buat juga Deleted View untuk menampilkan konfirmasi bahwa data sudah berhasil di delete.

```

<h2>Product Deleted</h2>
<div>
  <p>Your product was successfully deleted.</p>
</div>
<div>
  <p><a href="/Home">Click for view Product</a></p>
</div>

```

### Delete Confirmation

Please confirm you want to deleted Product : *Motherboard Asus TXBX?*

### Product Deleted

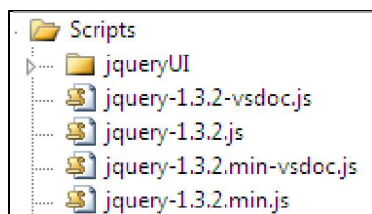
Your product was successfully deleted.

[Click for view Product](#)

### *Menambahkan JQuery Script*

JQuery adalah javascript framework yang saat ini paling populer dan banyak digunakan, karena selain penggunaannya yang relatif mudah JQuery juga menyediakan plugin yang sangat banyak dan dapat digunakan secara "free". JQuery juga disupport oleh komunitas yang sangat besar dan aktif.

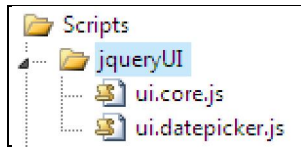
JQuery secara default sudah ditambahkan kedalam ASP.NET MVC Project, letaknya didalam folder "script"



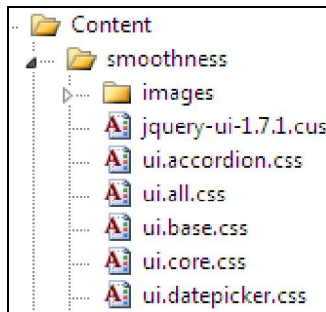
Saat ini JQuery sudah mencapai versi 1.3.2, anda dapat mendownload JQuery Plugin di <http://jquery.com/plugin/> untuk mendownload plugin-plugin yang disediakan.

Pada contoh kali ini kit akan menggunakan salah satu plugin dari JQuery yaitu "JQuery UI Calendar" untuk menambahkan date picker calendar kedalam program ASP.NET MVC diatas.

1. Untuk menambahkan JQuery UI tambahkan library berikut pada folder "script"



2. Kemudian tambahkan pula css dan image pada folder "content"



3. Kemudian buat "Partial View". Partial View adalah bagian View yang dapat digunakan pada View yang lain (mirip Web User Control pada Web Form).
4. Klik kanan pada folder "Home" didalam folder "Views". Pilih "Add View" – check "Create Partial View (.ascx)" beri nama "VUCDateTime.ascx". tulis kode berikut:

```
<link href="../../../Content/smoothness/ui.all.css" rel="stylesheet"
type="text/css" />
<script src="../../../Scripts/jquery-1.3.2.min.js"
type="text/javascript"></script>
<script src="../../../Scripts/jqueryUI/ui.core.js"
type="text/javascript"></script>
<script src="../../../Scripts/jqueryUI/ui.datepicker.js"
type="text/javascript"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $("#BuyDate").datepicker({
            showOn: 'button',
            buttonImage: '../../../Content/calendar.gif',
            buttonImageOnly: true
        });
    });
</script>
```

5. Kode diatas berisi kode JQuery yang digunakan untuk menampilkan date picker.
6. Tambahkan Partial View yang sudah dibuat kedalam View Create dan View Edit.

```
<% Html.RenderPartial("VUCDateTime"); %>
```

7. Maka ketika anda menjalankan Create atau Edit View anda dapat menggunakan date picker yang sudah anda buat.

Qty:

April 2009						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		