AJAX dengan ASP.NET 2.0

Soetam Rizky

PRESTASI PUSTAKA
PUBLISHER

AJAK dengan ASP.Net 2.0

Penulis: Soetam Rizky

Editor: **Alfa**

Desain Cover: **Abdullah Syafik Noer**

Setting: **Tim Prestasi**

Jakarta – Indonesia

Email Penerbit:

pprsby@plasa.com

AJAX dengan ASP .Net 2.0

ISBN : **978-602-8117-25-8**

Cetakan Pertama: Juni 2008

AJAX Dengan ASP .NET 2.0

Untuk istri dan putri kecilku, pelipur jiwa di kala suka dan duka

Kekayaan adalah merasa bahagia dan bersyukur dengan apa yang telah kita miliki
Kemiskinan adalah merasa sedih dan mengeluh dengan apa yang telah kita miliki

KATA PENGANTAR

Alhamdulillah, akhirnya buku AJAX dengan ASP .NET 2.0 dapat terselesaikan juga. Buku yang merupakan lanjutan dari buku Panduan Belajar ASP .NET 2.0 memang masih belum membahas secara tuntas mengenai implementasi AJAX dalam ASP .NET 2.0. Ambil saja conoth implementasi dengan framework lain seperti Anthem .NET maupun Magic AJAX yang belum terliput dalam buku ini. Kendala terbatasnya waktu, jumlah halaman serta semakin cepatnya perkembangan ASP .NET (yang pada saat buku ini ditulis telah resmi meluncurkan versi 3.5) menjadi alibi klasik dari segala kekurangan yang ada dalam buku ini.

Terima kasih kepada sesama rekan penulis buku di STIKOM Surabaya yang secara tidak langsung memberi "dorongan panas" agar terus berkarya : Erwin Sutomo, Tan Amelia, Slamet Ar Rokhim, Pak Sholiq dan Wido Nugroho dan juga penulis "pendatang baru", rekan Heri Pratikno. Tak lupa pula terima kasih pada Pak Yuswanto masih tetap setia untuk "membakar" semangat dalam setiap proses penulisan buku.

Seperti buku-buku dari penulis sebelumnya, buku ini memang sengaja dibuat dengan gaya classroom activity, sehingga seluruh latihan soal dan studi kasusnya dapat

dipraktekkan langsung di dalam kelas atau lab, baik oleh pengajar training, dosen maupun mahasiswa. Begitu pula tiap pengembangan dari studi kasus yang dapat menjadi tugas eksplorasi bagi para pembacanya.

Sedangkan studi kasus tambahan yang ada, meski kecil dan masih banyak kekurangan di banyak tempat, tetap diharapkan menjadi sebuah pembuka wawasan baru dalam melakukan implementasi AJAX di dalam sebuah situs yang lengkap.

Semoga buku ini dapat memberi manfaat bagi seluruh insan TI yang berminat mempelajari ASP .NET 2.0 dan juga implementasi AJAX secara umum. Selamat berkarya !!

Candi, Sidoarjo
Rabiul Akhir 1429 H – Awal Mei 2008

Pengantar

Apakah yang terbayang saat Anda mendengar kata AJAX ? Sebuah klub sepakbola ? Atau sebuah teknik pemrograman baru yang terkesan *high class* dan rumit ? Bagi Anda yang berkecimpung di dalam dunia pemrograman, khususnya di pemrograman web, tentu jawaban kedua yang dipilih.

AJAX atau Asynchronous Javascript and XML merupakan sebuah teknik pemrograman baru tetapi *lama* di dunia pemrograman web. AJAX sendiri pertama kali dicetuskan oleh Jesse James Garret dari Adaptive Path pada bulan Februari 200*5* dan lebih mengarah kepada sebuah cara pendekatan para web master ke pihak pelanggan, sehingga pelanggan merasa memiliki *pengalaman baru* dalam berinternet ria.

AJAX sendiri saat ini menjadi booming tersendiri di kalangan para programmer, baik para profesional programer, kalangan mahasiswa maupun para hobiis web. Implementasi dari AJAX sendiri juga sangat beragam, baik dengan menggunakan PHP, ASP .NET 2.0, JSP, Ruby serta framework independen lainnya yang berjumlah puluhan dan dapat ditemukan dengan mudah di dunia maya.

Penggunaan AJAX secara sederhana lebih mengarah kepada refresh halaman web secara parsial sehingga pengunjung situs tidak merasa meninggalkan situs secara utuh saat terjadi proses *postback* atau submit

sebuah data ke web server. Dengan teknik yang sama pula, maka saat ini banyak situs baru yang memanfaatkan teknik AJAX menjadi sebuah situs yang mengarah ke konsep Web 2.0 dengan berbagai fitur yang membuat *wow effect* bagi para pengunjungnya.

Terlepas dari segala jenis teori dan praktek tentang AJAX, masih dapat dimaklumi bahwa masih sedikit sekali programmer web yang berminat ataupun sudah berminat tapi ragu untuk mempelajari teknik baru ini. Beberapa programmer (mungkin juga termasuk Anda) merasa ragu untuk mempelajarinya karena secara konseptual, pembelajaran AJAX membutuhkan kemampuan tingkat intermediate untuk Javascript serta XML itu sendiri. Belum lagi minimnya buku berbahasa Indonesia yang membahas tentang hal ini. Beberapa yang lain mempelajari AJAX hanya dari sisi visual effect yang dianggap *wow* dengan menggunakan framework yang telah banyak tersedia di internet.

Tapi apakah benar belajar AJAX itu sulit ? Jawabannya mungkin saja ya, mungkin juga tidak. Dan yang pasti, dengan mencoba mempelajari dan (wajib) mempraktekkan apa yang ada dalam buku ini, nantinya Anda akan menjawab tidak. Karena dengan menggunakan bahasa yang *awam* dan teknik sederhana, diharapkan

setelah Anda *menghabiskan* buku ini, Anda tidak lagi fobia terhadap kata AJAX.

Dengan konsep *step by step*, teknik AJAX dalam buku ini akan diimplementasikan dengan menggunakan ASP .NET 2.0. Mengapa ASP .NET 2.0 ? Dengan banyak keunggulan dan kemudahan (baca buku "Panduan Belajar ASP .NET 2.0" oleh penulis yang sama) serta tool developer gratisnya yaitu Visual Web Developer Express Edition, maka programmer web tidak lagi risau dengan masalah lisensi serta lebih cepat mengimplementasikan sebuah situs.

Tetapi, seperti yang telah jamak diketahui, bahwa kebanyakan para praktisi *malas* jika diharuskan terlebih dulu membaca teori yang sifatnya bertele-tele dan hingga akhir sebuah pembelajaran buku, hanya mendapatkan keterangan tentang sintaks tertentu. Selain itu, kebanyakan buku yang bersifat **syntax oriented** lebih digunakan sebagai bahan referensi dibandingkan sebagai sebuah sarana membuat aplikasi sederhana. Untuk buku ASP .NET 2.0 yang bersifat syntax oriented Anda dapat memiliki buku lain dari penulis dan penerbit yang sama dengan judul "Panduan Belajar ASP .NET 2.0".

Sekali lagi, tetap disarankan, setelah Anda menyelesaikan pembuatan proyek dalam buku ini, tetaplah untuk mempelajari lebih lanjut tentang konsep AJAX

secara utuh dan juga jangan lupa untuk memperdalam kemampuan Anda di bahasa Javascipt serta konsep XML itu sendiri, sehingga jika Anda menemui kesulitan atau studi kasus yang lain, Anda jauh merasa lebih siap dan lebih mumpuni untuk memecahkan masalah tersebut.

Akhir kata, selamat berkarya dan jangan menyerah jika terjadi kegagalan !

Pendahuluan

Persiapan Awal

Buku ini khususnya ditujukan para praktisi TI, baik dari kalangan profesional ataupun pelajar dan hobiis yang ingin mempelajari lebih lanjut tentang pembuatan situs dengan menggunakan ASP .NET 2.0, terutama yang berbasis AJAX. Buku ini akan sangat cepat diaplikasikan untuk para programmer web yang sebelumnya telah mengenal ASP .NET 1.x dengan menggunakan Visual Studio .NET 2003. Tapi juga tidak menutup kemungkinan bagi para programmer web yang sebelumnya telah familiar dengan bahasa pemrograman web lainnya seperti PHP ataupun ASP klasik.

Bagaimana dengan para pemula ? Apa yang harus diketahui sebelum mengaplikasikan buku ini ? Berikut adalah daftar pengetahuan yang seharusnya Anda ketahui terlebih dulu :

1. Logika pemrograman

   Jika Anda sama sekali belum pernah melakukan aktifitas pemrograman, disarankan untuk terlebih dulu memahami logika pemrograman dasar, seperti percabangan dan perulangan. Meski dalam studi kasus, hal tersebut tidak disebutkan secara signifikan.

2. Pengetahuan tentang konsep pemrograman web

   Jika sebelumnya Anda telah melakukan aktifitas pemrograman, tetapi untuk aplikasi jenis desktop, seperti Visual Basic 6.0, C++ dan yang lain, maka disarankan agar Anda (minimal) mengetahui tentang konsep dari sebuah aplikasi basis web. Didalamnya Anda wajib memahami tentang *sitemap*, konsep navigasi link dan jenis tampilan serta jenis browser dalam sebuah aplikasi basis web.

3. HTML dan CSS

   Pengetahuan tentang HTML saat ini seringkali dilupakan para programmer web dikarenakan banyaknya tool desain yang telah mempermudah implementasi dari HTML. Bahkan beberapa tool seperti halnya Dreamweaver, telah mampu menyulap para pemula yang tidak memiliki kemampuan HTML sama sekali, menjadi seorang desainer web yang mumpuni. Tetapi di dalam proses pemrograman web dengan menggunakan ASP .NET 2.0, pengetahuan tentang HTML tetap diperlukan karena dalam banyak olah tampilan halaman web, Anda tetap memerlukan pengetahuan HTML untuk merombak sebuah komponen menjadi lebih *eye catching*.

   Sedangkan untuk CSS (Cascading Style Sheet) lebih diperlukan sebagai alat bantu dalam melakukan desain

halaman web yang lebih fleksibel dan seragam. Dengan bantuan CSS, seluruh halaman web dapat didesain ataupun dimodifikasi dengan sangat mudah dalam satu langkah saja. Di studi kasus buku ini, desain dilakukan dengan menggunakan CSS secara total. Bagi Anda yang sama sekali buta dengan CSS, tidak perlu khawatir, karena Visual Web Developer Express (VWD Express) telah menyediakan alat bantu, jika Anda ingin melakukan modifikasi desain.

4. Konsep .NET framework

   Pengetahuan tentang konsep .NET framework sebenarnya merupakan komponen terpenting sebelum Anda memulai mengerjakan studi kasus di dalam buku ini. Tetapi, secara naluriah, hampir semua praktisi ataupun pemula, seringkali mengabaikan bagian buku yang mengenalkan tentang sebuah konsep. Konsep .NET framework sendiri, secara umum telah dimuat dalam MSDN Library, baik versi online (msdn. microsoft.com) ataupun versi offline. Disarankan untuk tetap membaca (sedikit banyak) tentang konsep .NET framework, karena dengan konsep yang matang, Anda akan mampu mengembangkan studi kasus ini menjadi sebuah situs berita yang tidak lagi dalam kategori sederhana.

5. Bahasa Visual Basic

   Dalam studi kasus ini, digunakan ASP .NET 2.0 dengan menggunakan bahasa pemrograman Visual Basic .NET. ASP .NET 2.0 sendiri merupakan sebuah CLR yang dapat dibangun dengan beberapa bahasa pemrograman di lingkup .NET framework, seperti Visual Basic .NET, Visual C# dan Visual J#. Jika Anda sebelumnya telah mengenal ASP klasik, ataupun Visual Basic 6.0, terlebih lagi jika Anda telah menggunakan ASP .NET 1.x sebelumnya, Anda tidak akan mengalami kesulitan untuk memahami studi kasus di buku ini. Tetapi jika Anda sebelumnya tidak mengenal Visual Basic, tidak perlu khawatir, karena listing program yang terdapat dalam studi kasus ini, tidak akan mencapai 50 baris listing program, sehingga dipastikan Anda juga tidak akan lama dalam beradaptasi dengan Visual Basic .NET. Dan juga tidak menutup kemungkinan, jika Anda ingin melakukan konversi ke dalam bahasa pemrograman lain di lingkup .NET framework, misal Visual C# (dibaca : Visual C Sharp)

6. Javascipt

   Meski Javascript yang digunakan dalam studi kasus bukanlah termasuk level intermediate, tetapi tetap disarankan untuk mempelajari konsep dari Javascript secara utuh. Dan disarankan pula untuk lebih

mempelajari Javascript sebagai sarana manipulasi dari halaman web, bukan hanya memandang Javascript sebagai alat bantu visual effect dalam sebuah situs.

Pembuatan situs dengan ASP .NET 2.0 sendiri, selain bisa menggunakan tool profesional yaitu Visual Studio 2005 (versi komersil), juga bisa menggunakan tool *freeware* Visual Web Developer Express Edition yang dapat didownload secara gratis dari situs Microsoft

Visual Web Developer Express Edition

Sejak kemunculan Visual Studio Express Edition (Visual Basic Express, Visual C# Express dan Visual Web Developer), banyak praktisi pemrograman yang menyambut gembira. Hal ini dikarenakan, versi Express dari Visual Studio yang telah menggratiskan penggunaan dan lisensi dari pembuatan tiap aplikasi dengan menggunakan tool tersebut. Itu berarti bahwa para praktisi yang berstatus sebagai hobiis, ataupun para pelajar dan mahasiswa, tidak lagi terbentur dengan masalah lisensi, dan tidak perlu lagi melakukan instalasi ilegal untuk membuat dan mempelajari aplikasi dengan menggunakan bahasa pemrograman di lingkup .NET framework.

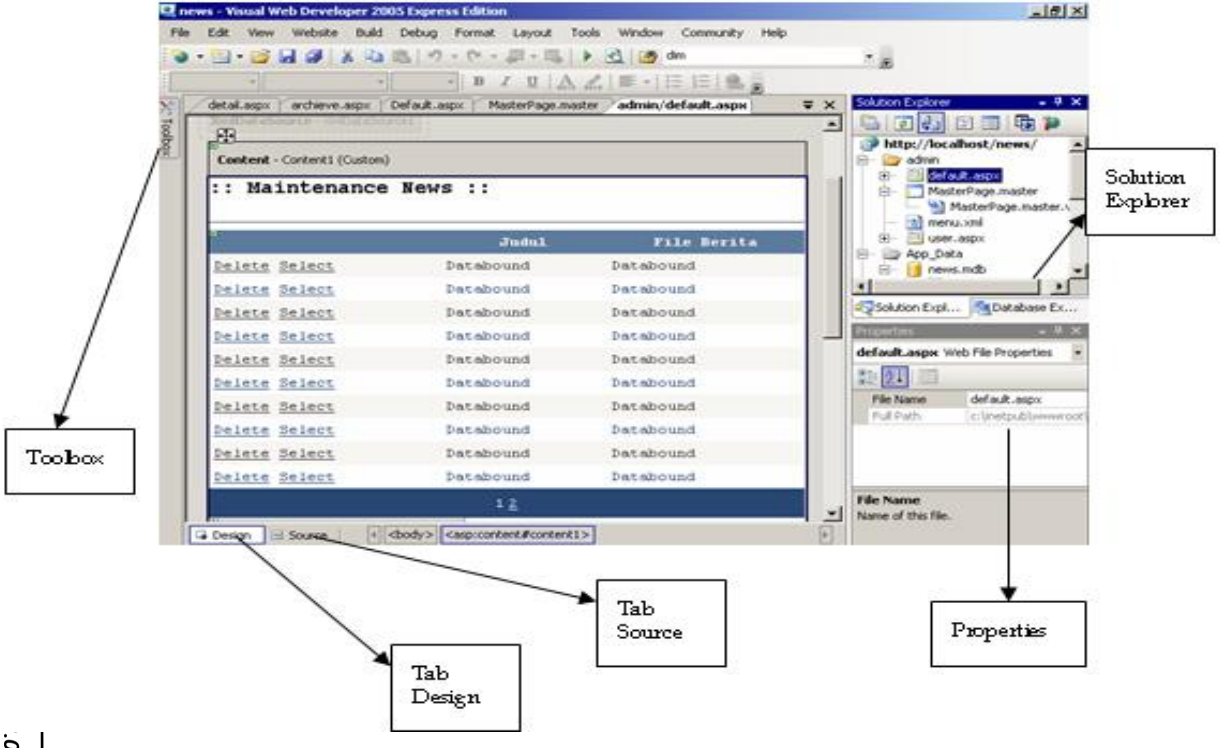
VWD Express Edition sendiri dikhususkan untuk pembuatan aplikasi berbasis web dengan menggunakan ASP .NET 2.0. Meski memiliki banyak keterbatasan dibandingkan dengan versi komersilnya (Visual Studio 2005), tetapi terbukti bahwa dengan menggunakan VWD Express, Anda akan tetap mampu membuat sebuah aplikasi web yang mampu bernilai komersil.

Instalasi VWD Express sendiri sangatlah sederhana, yang perlu Anda lakukan hanyalah mengeksekusi file *setup* dan kemudian tinggal melakukan klik tombol *next* hingga

proses instalasi selesai dilakukan. Sedangkan spesifikasi umum dari VWD Express adalah sebagai berikut :

- Prosesor sekelas Pentium 3 atau kecepatan minimal 600 MHz
- RAM minimal 192 Mb, disarankan 256 Mb
- Sisa ruang harddisk minimal 115 Mb
- Sistem operasi Windows XP atau 2000 (Server atau Pro)

Sedangkan skema IDE dari VWD Express adalah sebagai berikut :


news - Visual Web Developer 2005 Express Edition
File Edit View Website Build Debug Format Layout Tools Window Community Help
detail.aspx archieve.aspx Default.aspx MasterPage.master admin/default.aspx
Content - Content1 (Custom)
:: Maintenance News ::
Judul
File Berita
Delete Select
Databound
Design
Source
<body>
<asp:content#content1>
Solution Explorer
http://localhost/news/
admin
default.aspx
MasterPage.master
menu.xml
user.aspx
App_Data
news.mdb
Solution Expl...
Database Ex...
Properties
default.aspx Web File Properties
File Name
default.aspx
Full Path
File Name
Name of this file.
Toolbox
Tab Source
Tab Design
Properties
Solution Explorer

Sekilas ASP .NET 2.0

Apakah ASP .NET merupakan sebuah bahasa pemrograman ? Jawabannya adalah tidak ! Secara sederhana, ASP .NET merupakan sebuah bagian dari .NET framework yang dikhususkan untuk pembuatan aplikasi web. ASP .NET sendiri dapat dikembangkan dengan berbagai bahasa pemrograman yang terdapat dalam lingkup .NET framework seperti Visual Basic .NET, Visual C# ataupun Visual J#.

ASP .NET 2.0 sendiri secara revolusioner telah mengembangkan banyak fitur baru dan fasilitas yang sangat memudahkan para programmer dalam membuat sebuah situs atau aplikasi berbasis web. Scott Guthrie, team leader dari pengembang ASP .NET 2.0, saat peluncuran ASP .NET 2.0, mengatakan bahwa dengan menggunakan ASP .NET 2.0, maka 70% listing program dapat direduksi dalam proses pembuatan sebuah situs.

Banyak komponen baru yang sangat membantu dan mengubah paradigma dari sebuah pembuatan aplikasi basis web. Misalnya penggunaan Master, fitur Web Parts, koneksi database dengan komponen berbasis Data Source, navigasi situs terintegrasi dengan site map, ataupun pengembangan terbaru lainnya seperti ASP .NET AJAX.

Konsep dari ASP .NET 2.0 (dan juga .NET framework) tidak akan dijelaskan secara panjang lebar, karena akan memakan ruang dan waktu yang banyak (dan juga tidak sesuai dengan konsep dan tujuan awal dari buku ini, tetapi jika Anda ingin mempelajarinya dapat membeli buku "Panduan Belajar ASP .NET 2.0" dari penerbit dan penulis yang sama). Tapi yang jauh lebih penting adalah bagaimana mempelajari secara mudah dan cepat ASP .NET 2.0, dengan asumsi bahwa hal ini akan memicu Anda untuk lebih memperdalam dan membuka wawasan tentang ASP .NET 2.0.

Sekilas AJAX

Saat beragam implementasi teknik AJAX bermunculan, maka muncul pertanyaan : kenapa harus repot belajar AJAX ?

Tentu saja banyak versi dari para AJAX mania untuk pertanyaan ini, ada yang mengatakan bahwa dengan adanya AJAX maka proses *roundtrip* ke web server dapat dikurangi sehingga menghemat bandwidth. Ada lagi yang mengatakan bahwa sifat AJAX yang *asynchronous* dapat meningkatkan kinerja situs dengan melakukan beberapa proses dalam satu waktu. Dan ada juga yang berkata bahwa AJAX dapat meningkatkan *response time* sebuah situs serta menimbulkan *pengalaman baru* bagi pengunjung situs. Tapi, tentu saja, AJAX juga memiliki beberapa kelemahan. Misalnya, jika terjadi sebuah error (terutama di sisi Javascript) akan lebih sulit melakukan proses debugging dibanding teknik biasa. Juga pada koneksi internet yang tidak stabil, penggunaan AJAX seringkali menimbulkan kesan lambat dalam situs tersebut.

AJAX (Asynchronous Javascript and XML) pertama kali dicetuskan pada 18 Februari 2005 oleh Jesse James Garret dari Adaptive Path dalam artikel berjudul "AJAX, A new approach to web applications". Artikel tersebut

menyebutkan bahwa untuk lebih menarik perhatian pengunjung situs dibutuhkan teknik baru yang dapat mengatasi keterbatasan aplikasi web, sehingga kemampuan aplikasi desktop dapat dimasukkan ke dalam aplikasi web. Meski tergolong baru, teknologi AJAX saat ini mengalami booming yang sangat menakjubkan di dunia internet, sehingga seakan menjadi standar baru bagi semua situs web baik komersial ataupun non komersial.

AJAX sendiri lebih dikategorikan sebagai sebuah teknik pemrograman tanpa memandang bahasa pemrograman yang mengimplementasikannya. Banyak teori dan pendapat tentang definisi sesungguhnya dari AJAX, tetapi dari sekian banyak definisi tersebut, dapat disimpulkan bahwa AJAX dapat terdiri dari beberapa komponen penting antara lain :

1. Presentasi web yang menggunakan HTML dan CSS
2. Data yang ditransportasikan dalam bentuk XML dan adanya penggunaan JSON (Javascript Object Notation)
3. Adanya proses XMLHTTPRequest
4. Penggunaan Javascript

Pada dasarnya pengembangan sebuah aplikasi web yang menggunakan teknik AJAX harus memiliki konsep sebagai berikut :

1. Hanya melakukan update terhadap bagian halaman web yang mengalami proses transfer data

2. Bersifat asynchronous sehingga proses dapat dijalankan secara majemuk dalam waktu yang bersamaan
3. Memiliki solusi yang dapat memecahkan masalah yang umumnya dihadapi di aplikasi desktop, misal : drag and drop
4. Lebih sedikit terjadi proses postback

Dari teknologi yang digunakan tersebut, AJAX dapat diimplementasikan dengan menggunakan berbagai macam bahasa pemrograman web seperti PHP, JSP, Coldfusion ataupun seluruh bahasa pemrograman yang mendukung ASP .NET 2.0 seperti Visual Basic .NET atau Visual C#. Dengan adanya beragam AJAX Framework (baik yang gratis ataupun komersil), maka programmer web tidak lagi harus menguasai secara expert tentang XML, Javascript dan bahasa pemrograman yang digunakan untuk dapat mengaplikasi teknik AJAX.

AJAX sendiri telah mengalami perkembangan yang sangat pesat, sehingga kian banyak situs besar yang telah mengaplikasikannya seperti Google, Yahoo dan Microsoft sendiri. Tentu saja dengan teknik baru tersebut, banyak hal yang makin bisa diimplementasikan dalam sebuah situs, bahkan mulai banyak aplikasi desktop yang diusung sebagai aplikasi web, misal : aplikasi perkantoran (word processing, spreadsheet, presentation) sejenis Microsoft Office, aplikasi pengolah image sejenis Photoshop dan

masih banyak lagi. Meski memiliki berbagai kelebihan yang membuka sudut pandang baru terhadap pemrograman web, tetapi AJAX sendiri memiliki berbagai kelemahan, diantaranya :

1. Kesulitan dalam mengimplementasikan Javascript terutama dalam hal debugging
2. Tidak seluruh browser support AJAX
3. Masih sulitnya diimplementasikan di multi device
4. Sulitnya menangani implementasi jika pengunjung situs melakukan penekanan tombol Back di dalam browser
5. Proses loading awal yang membutuhkan waktu lebih lama dibanding situs biasa.

Karenanya, tidak semua solusi situs sesuai dengan penerapan teknik AJAX. Meski demikian, saat ini hampir semua situs "modern" berlomba – lomba mengaplikasikan teknik AJAX dengan berbagai macam tema baru. Situs – situs tersebut umumnya dimasukkan ke kategori Web 2.0 yang kini menjadi jargon marketing baru bagi situs yang mengimplementasikan teknik AJAX.

Di lingkup AJAX Framework yang semakin lama semakin menjamur di dunia maya, perlu diperhatikan beberapa hal dalam pemilihannya :

1. Lebih baik memilih framework yang dikhususkan untuk bahasa pemrograman yang digunakan, misal :

untuk ASP .NET 2.0 lebih baik menggunakan framework yang memang dikhususkan untuk ASP .NET 2.0 seperti ASP .NET AJAX, Anthem .NET, Magic AJAX atau AJAX pro.

2. Pilih framework sesuai kebutuhan, terutama jika framework tersebut berbasis Javascript murni seperti scriptaculous atau Dojo. Karena terkadang efek visual atau teknik lain yang ditawarkan tidak dibutuhkan semuanya dalam situs yang akan dibuat.
3. Pilih framework (jika bisa) yang terkecil ukurannya, atau dalam ukuran yang bisa ditoleransi, sehingga akses terhadap situs tersebut tidak berat.
4. Jika memang harus menggabungkan antara satu framework dengan framework yang lain, perhatikan terlebih dulu, apakah kedua framework tersebut tidak menimbulkan masalah di kemudian hari.
5. Dan jika memang bisa mendapatkan framework yang gratis dengan kualitas tinggi, tidak usah berusaha mencari framework yang komersial !!!

Secara umum, meski AJAX saat ini menjadi sebuah tren yang tidak mungkin terelakkan di dunia pemrograman web, perlu diperhatikan apakah memang AJAX dibutuhkan dalam situs tersebut atau tidak. Karena penggunaan AJAX sebenarnya lebih ditekankan terhadap penghematan traffic internet serta kemudahan dan pengalaman baru bagi

pengunjung situs, bukan sekedar mengikuti tren atau agar situs terlihat lebih "modern".

ASP .NET AJAX

ASP .NET AJAX merupakan nama resmi dari framework AJAX dari Microsoft yang sebelumnya memiliki nama ATLAS. Pemilihan framework ini dalam buku dengan alasan utama bahwa framework ini memang dikhususkan untuk ASP .NET 2.0 dan juga telah dilengkapi berbagai komponen tambahan yang bersifat open source dan terpaket dalam AJAX toolkit (dapat didownload di www.codeplex.com dan akan dibahas pada bab tersendiri). Selain itu, ASP .NET AJAX tidak menyertakan file runtime, melainkan terinstalasi dalam GAC (Global Assembly Cache) sehingga tidak terjadi pembengkakan ukuran file hasil pemrogaman. Dan juga, ASP .NET AJAX juga tersedia untuk bahasa pemrograman web lain seperti PHP dan JSP di Microsoft AJAX Library.

Meski demikian, tidak berarti bahwa ASP .NET AJAX merupakan framework AJAX yang terbaik dan satu-satunya untuk ASP .NET 2.0. Karena masih banyak framework lain yang dapat dipilih sesuai kebutuhan dan juga sama – sama gratis seperti Anthem .NET, Magic AJAX atau AJAX Pro.

Langkah pertama yang harus dilakukan dalam implementasi ASP .NET AJAX adalah melakukan instalasi framework tersebut (dapat didownload di ajax.asp.net) sehingga di dalam VWD Express muncul template baru untuk ASP .NET AJAX, kemudian buat situs dengan

menggunakan template baru tersebut. Dalam sebuah situs ASP .NET AJAX yang perlu diperhatikan pertama kali adalah file web.config yang telah berisi konfigurasi untuk mendefinisikan Microsoft Web Extension (dapat dicari file runtime-nya di dalam folder Program Files).

Terdapat tiga komponen penting dalam ASP .NET AJAX yaitu :

1. Script Manager

   Tiap halaman web yang akan ditempati oleh teknik AJAX (perlu diingat, bahwa dalam satu situs, tidak semua halaman web ditempati oleh teknik AJAX), harus ditempati oleh komponen ini. Karena dengan adanya komponen ini, proses update parsial dalam sebuah halaman web dapat dilakukan. Dalam satu halaman web, cukup terdapat satu komponen Script Manager.

2. Update Panel

   Jika komponen Script Manager merupakan otak dari implementasi teknik AJAX, maka dapat diibaratkan bahwa Update Panel adalah "badan" yang berfungsi untuk mengeksekusi dari teknik AJAX itu sendiri. Tiap komponen yang akan diberi sifat AJAX dan masuk ke proses update parsial harus "dibalut" dengan komponen Update Panel. Dalam sebuah halaman web dapat terdapat lebih dari satu Update Panel. Tiap

Update Panel juga dapat mempengaruhi Update Panel yang lain dengan melakukan setting di sub tag Trigger. Secara default, Update Panel hanya akan melakukan update dengan kondisi tertentu (conditional), tetapi dapat juga diberikan setting agar melakukan update tanpa kondisi, jika komponen yang terletak dalam Update Panel tersebut secara otomatis terpengaruh dengan proses yang terjadi dalam satu halaman web.

3. Update Progress

   Tiap gambar atau tulisan yang terjadi dalam komponen ini akan ditampilkan jika terjadi proses update parsial dalam halaman web yang mengimplementasikan teknik AJAX. Komponen ini umumnya dibutuhkan untuk penanda bagi pengunjung situs agar tahu bahwa sedang terjadi proses dalam halaman web. Karena dalam sebuah halaman web yang mengimplementasikan AJAX, progress bar browser umumnya tidak menunjukkan aktifitas, sehingga seringkali pengunjung situs berasumsi bahwa koneksi internet tidak berjalan.

Di sub bab berikutnya akan diterangkan contoh-contoh penggunaan teknik AJAX dengan menggunakan ASP .NET AJAX.

Contoh 1 : Partial Postback

Contoh partial postback sederhana adalah dengan memanfaatkan komponen timer yang diletakkan di dalam suatu halaman web. Contoh pertama ini terdiri dari dua bagian, bagian pertama akan menunjukkan efek timer tanpa penggunaan AJAX, sedangkan bagian kedua akan menunjukkan penggunaan timer dengan menggunakan AJAX.

Efek terpenting yang akan dilihat dalam bagian pertama adalah adanya proses *full page postback* yang berarti seluruh isi dari halaman web akan diproses ulang dengan memanggil isi program langsung dari server. Proses ini sebenarnya merupakan proses dalam sebuah halaman web yang "normal" tanpa menggunakan teknik AJAX didalamnya.

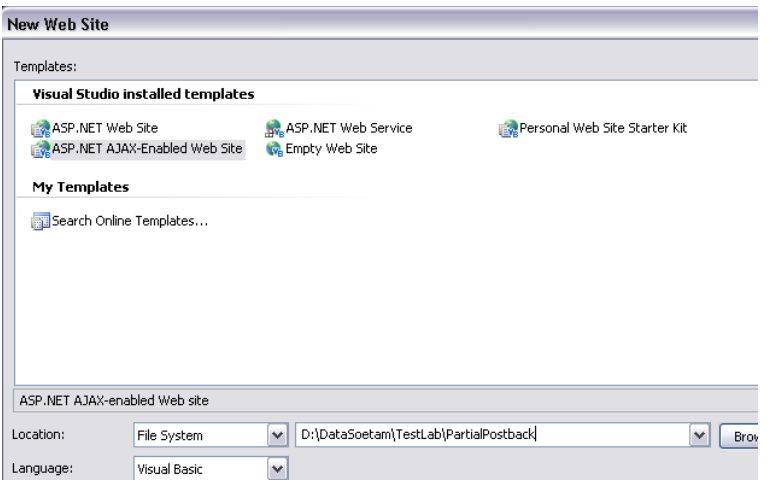
Sedangkan di bagian kedua proses postback hanya akan dilakukan sebagian dalam sebuah halaman web, sehingga hanya bagian yang diperlukan saja yang akan mengalami proses postback atau seringkali ditandai dengan adanya proses refresh. Proses seperti ini disebut sebagai *partial postback* atau proses postback sebagian dari halaman web.

Sesungguhnya, proses partial postback telah sering dilakukan oleh para programmer web, tetapi dengan

memanfaatkan bantuan dari pemrograman *client side* seperti Javascript. Sedangkan untuk pemrograman server side, masih sangat sulit dilakukan. Tetapi dengan menggunakan bantuan dari ASP .NET AJAX, proses semacam itu sangat mudah untuk dilakukan.

Untuk membuat contoh partial postback, ikuti langkah berikut ini :

Buat sebuah web site baru dalam Visual Web Developer Express dengan tipe ASP .NET AJAX Web Site .

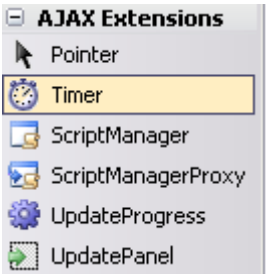


Dengan asumsi bahwa framework ASP .NET AJAX telah terinstalasi sebelumnya.

Kemudian, beralihlah ke halaman *Default.aspx* yang telah tersedia. Tempatkan sebuah komponen *Timer* dan sebuah komponen *Label* di dalam halaman web tersebut. Untuk memperlihatkan efek full page postback, tempatkan pula sebuah gambar di dalam halaman web tersebut. Gambar yang ditempatkan bisa berupa gambar apa saja (ekstensi jpg, bmp maupun gif) dengan ukuran sedang yang ditempatkan di bagian paling atas halaman web.

Penempatan gambar dapat langsung menggunakan teknik *drag and drop* dari Windows Explorer yang langsung ditempatkan di dalam Solution Explorer, atau dengan menggunakan pilihan menu *Website → Add Existing Item* lalu pilih file gambar yang Anda miliki di dalam komputer.

Perlu diperhatikan, bahwa ukuran file gambar sebaiknya tidak terlalu kecil, agar pada saat halaman web dijalankan, efek *flicker* atau berkedip dari gambar dapat sangat jelas terlihat. Usahakan pula untuk tidak menempatkan gambar jenis *gif animator* sehingga tidak mempengaruhi pengamatan saat proses postback terjadi.

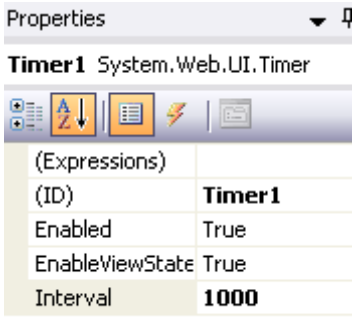


Jika ASP .NET AJAX terinstalasi dengan benar, maka dalam Toolbox akan terdapat tambahan grup item *AJAX Extensions* yang berisi komponen utama dari ASP .NET AJAX.

Setiap halaman web default yang dibentuk dari jenis situs ASP .NET AJAX akan ditempati komponen *ScriptManager* secara default. Selain itu, perhatikan pula isi dari file web.config yang ada dalam Solution Explorer. Untuk sementara, isi dari file web.config jangan dimodifikasi terlebih dulu.

Selanjutnya, atur property dari komponen *Timer*

a. Interval : 1000
b. Enabled : True



Interval Timer memiliki satuan miliseconds (milidetik), sehingga jika interval diset menjadi 1000, berarti interval akan berjalan tiap satu detik.

Lalu, dobel klik di komponen Timer dan ketikkan listing berikut di dalam *Timer1_Tick*

```
Label1.Text = Format(Now, "hh:mm:ss")
```

Eksekusi situs dengan menekan kombinasi tombol Ctrl+F5, dan lihat hasil dalam browser

Perhatikan hasil yang terlihat dalam browser dari situs tersebut. Akan terlihat bahwa gambar akan mengalami *flicker* atau berkedip sebagai efek dari proses full page postback. Perhatikan pula pada icon di pojok kanan browser, pada tiap detik berjalan, icon tersebut akan "berputar" sebagai tanda bahwa halaman web sedang menjalani proses pemanggilan ke server (dalam hal ini server adalah web server lokal dalam komputer kita sendiri).

Sekarang, untuk bagian kedua yang mendemonstrasikan teknik AJAX agar terjadi proses partial postback, ikuti langkah berikut :

1. Tambahkan komponen *UpdatePanel* dalam halaman tersebut, dan drag komponen *Timer* serta komponen *Label* ke dalam komponen *UpdatePanel* yang ada.

2. Kemudian, eksekusi ulang situs dengan menekan tombol Ctrl+F5.

Sekarang perhatikan perbedaan antara contoh di bagian pertama dengan bagian kedua. Pada bagian kedua, gambar yang ada tidak akan berkedip, meski "jam digital" berubah pada tiap satuan detik.

Komponen *UpdatePanel* merupakan "otak" dari implementasi teknik AJAX dengan berbagai "improvisasi" didalamnya yang akan dijelaskan pada contoh berikutnya.

Hal ini disebabkan karena adanya komponen *UpdatePanel* yang mengimplementasikan proses partial postback dalam halaman web tersebut. Ini berarti, sebuah halaman web "biasa" secara teoritis dapat diubah menjadi sebuah halaman web berbasis AJAX dengan sangat mudah dan sangat cepat !!

Contoh 2 : DropDownList AutoPostBack

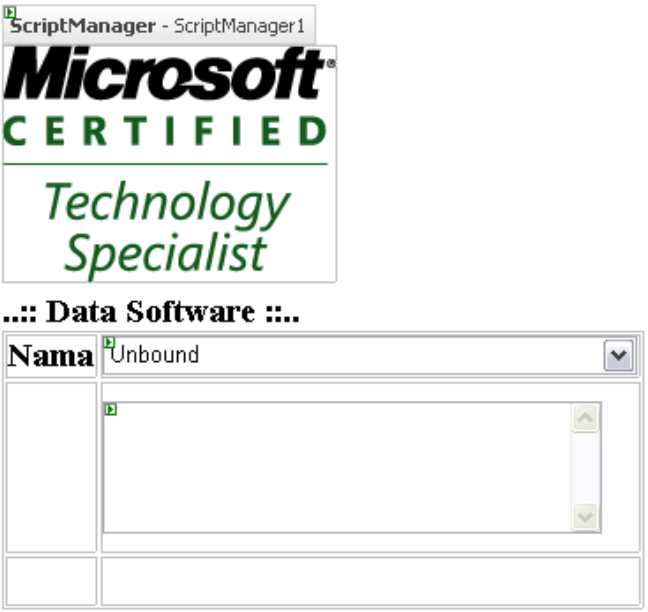
Contoh kedua masih mendemonstrasikan penggunaan UpdatePanel sebagai inti dari proses partial postback. Tetapi kali ini akan diimplementasikan dalam teknik AutoPostBack dengan memanfaatkan komponen DropDownList didalamnya.

Penggunaan teknik AutoPostBack merupakan teknik yang langsung melakukan reaksi terhadap perubahan yang terjadi dalam sebuah komponen tanpa menunggu penekanan sebuah tombol atau proses refresh. Dengan kata lain yang lebih sederhana, proses AutoPostBack merupakan proses refresh instant yang terjadi saat pengguna situs melakukan reaksi terhadap sebuah komponen non Button, misal : DropDownList yang sedang dilakukan pemilihan item, RadioButtonList yang diklik, pengetikan TextBox dan lainnya.

Dalam contoh ini, masih sama dengan contoh sebelumnya, akan dibandingkan antara penggunaan teknik AutoPostBack dalam sebuah situs biasa dengan penggunaan teknik AutoPostBack dalam ASP .NET AJAX. Untuk bagian pertama tanpa menggunakan teknik AJAX, lakukan langkah berikut :

1. Buat sebuah situs baru bertipe *ASP .NET AJAX Enabled Website*

2. Letakkan sebuah gambar, DropDownList dan sebuah TextBox dengan property MultiLine yang diset menjadi *true* dalam halaman web *Default.aspx*.



Sama seperti pada contoh sebelumnya, gambar yang disertakan dalam halaman web ini bebas, karena lebih ditekankan pada perbedaan efek flicker pada saat implementasi teknik AJAX.

3. Kemudian buat dua buah file teks, masing-masing dengan nama *data.txt* dan *detail.txt*, dengan isi data sebagai berikut :

   data.txt

   ```
   Microsoft Office,Open Office, Windows Server 2008
   ```

   detail.txt

   ```
   Program perkantoran paling banyak digunakan saat ini
   Pesaing dari Microsoft OFfice berbasis open source
   Server generasi terbaru dari Microsoft
   ```

Anda bisa bebas mengisikan data dalam kedua file teks tersebut, dengan syarat bahwa jumlah baris dalam file *detail.txt* sama dengan jumlah frase yang terpisah dengan tanda koma di file *data.txt*.Isi file dari *data.txt* akan diisikan ke dalam DropDownList1, sedangkan isi dari file *detail.txt* akan diisikan ke dalam TextBox1 sesuai dengan indeks DropDownList yang dipilih.

4. Set property *AutoPostBack* dari DropDownList menjadi *true* dengan melakukan klik pada smart tag dan pilih opsi *Enable AutoPostBack*

5. Berikutnya, dobel klik di dalam DropDownList1 dan isikan listing berikut di dalam *DropDownList1_SelectedIndexChanged*

```
Dim xFile As String = _
    Server.MapPath("detail.txt")
Dim xdata() As String
If System.IO.File.Exists(xFile) Then
    xdata = Split(System.IO.File. _
        ReadAllText(xFile), Chr(13))
End If
TextBox1.Text = xdata _
    (DropDownList1.SelectedIndex)
```

Pembacaan file teks secara utuh dapat menggunakan prosedur *ReadAllText* dari class *File* dalam namespace *System.IO*. Untuk melakukan pengecekan file tersebut berada dalam web server, digunakan fungsi *FileExist*. Fungsi *Server.MapPath* akan mengembalikan nilai string yang berupa url path dari sebuah file yang ada dalam web server. Sedangkan fungsi *Split* yang digunakan dalam listing tersebut, berfungsi sebagai pemecah sebuah string ke dalam suatu array berdasarkan tanda

pemisah tertentu. Dalam file *detail.txt*, tanda pemisah yang digunakan adalah tanda *Enter* (ganti baris), dengan kata lain, tiap baris dianggap sebagai sebuah anggota array baru.

6. Lalu dobel klik di dalam halaman web untuk menuju ke prosedur *Page_Load* dan ketikkan listing berikut :

```
Dim xFile As String = _
    Server.MapPath("data.txt")
Dim xdata() As String
If Not IsPostBack Then
    If System.IO.File. _
        Exists(xFile) Then
        xdata = Split(System.IO.File. _
            ReadAllText(xFile), ",")
    End If
    With DropDownList1
        .DataSource = xdata
        .DataBind()
    End With
End If
```

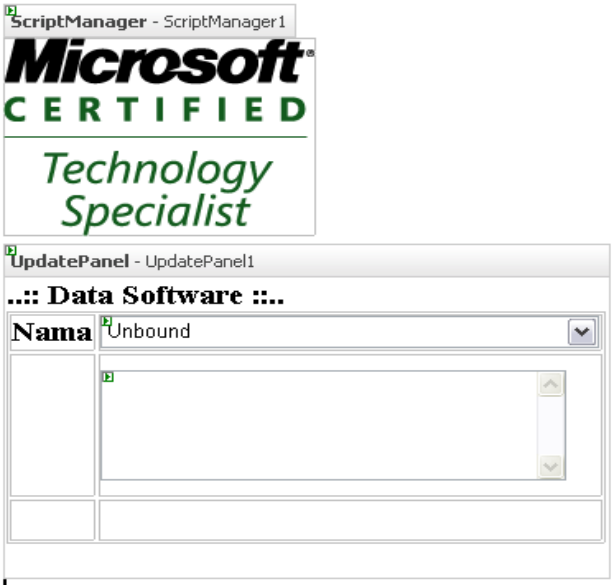
Perhatikan cara penempatan array ke dalam sebuah komponen DropDownList yang ada dalam prosedur tersebut. Pengecekan *IsPostBack* berarti bahwa pengisian data dalam

DropDownList hanya diisikan pada saat halaman web diasumsikan pertama kali melakukan proses loading, sehingga menghemat waktu *roundtrip* ke server saat pengguna web menekan tombol *Refresh* pada browser.

7. Eksekusi situs dengan menekan kombinasi tombol Ctrl+F5

Perhatikan efek yang terjadi saat Anda melakukan pemilihan data di dalam DropDownList. Efek flicker kembali terjadi, khususnya di gambar yang telah ditempatkan dalam halaman web. Kini, untuk menghentikan efek flicker tersebut, ikuti langkah berikut untuk mengimplementasikan teknik AJAX didalamnya.

1. Tempatkan sebuah komponen UpdatePanel dalam halaman web sebelumnya.
2. Drag seluruh komponen dalam halaman web tersebut, kecuali gambar, ke dalam komponen UpdatePanel tersebut.



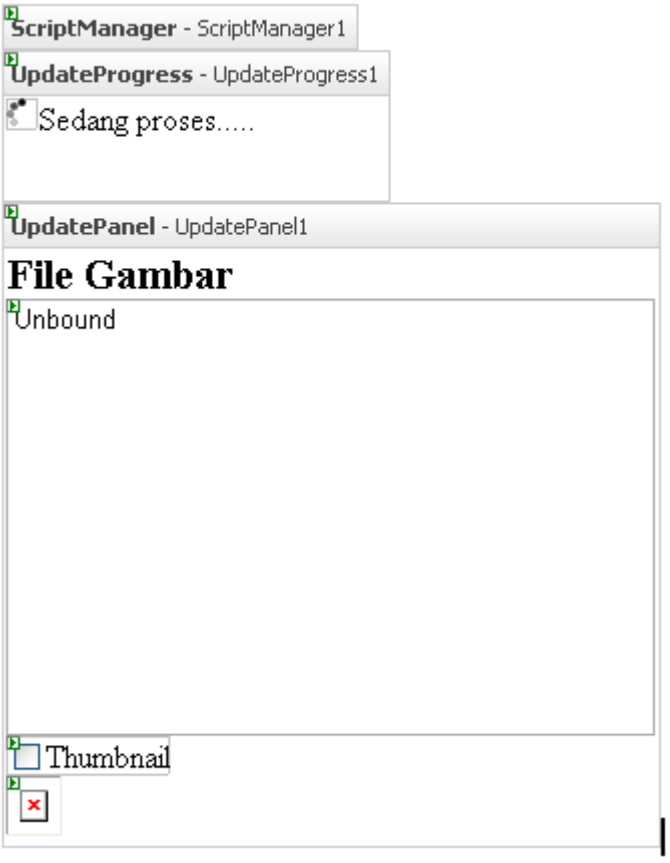
3. Kini, eksekusi ulang halaman web yang ada, dan perhatikan bahwa efek flicker tidak lagi terjadi.

Sekali lagi, teknik AJAX bukan lagi menjadi sebuah teknik yang "menakutkan" untuk diimplementasikan ke dalam sebuah halaman web. ASP .NET AJAX telah sangat banyak membantu proses implementasi teknik AJAX dengan sangat mudah. Untuk contoh-contoh berikutnya, akan langsung diimplementasikan penggunaan teknik AJAX tanpa membandingkan dengan teknik konvensional dalam sebuah halaman web.

Contoh 3 : View Thumbnail dan Tampilan UpdateProgress

Dalam contoh ini akan dijelaskan mengenai penggunaan komponen UpdateProgress, sekaligus mendemonstrasikan penggunaan GDI+ (Graphics Drawing Interface) untuk membuat sebuah file gambar thumbnail *on the fly* atau saat proses berlangsung di dalam web server. Penggunaan UpdateProgress pada server lokal tidak akan terlalu terlihat karena proses berlangsung dengan sangat cepat. Karenanya, di dalam contoh ini akan disertakan proses delay agar tampilan dalam UpdateProgress dapat terlihat.

Layout dari halaman web yang akan dibuat, terdiri dari sebuah komponen UpdateProgress, dan di dalam komponen tersebut ditempati tulisan *Sedang Proses…* serta sebuah file gambar berjenis gif animasi untuk menampilkan status proses (file gambar sejenis dapat juga didownload secara gratis di www.ajaxload.info). Berikutnya ditempatkan komponen UpdatePanel yang didalamnya ditempatkan sebuah ListBox dengan property *AutoPostBack* diset menjadi *true*, dan sebuah Checkbox serta sebuah komponen *Image*. Layout akan tampak seperti pada gambar berikut :



Untuk pembuatan dari halaman web tersebut, ikuti langkah berikut ini :

1. Dobel klik di bagian halaman web yang kosong, lalu di dalam *Page_Load* ketikkan listing berikut :

```
If Not IsPostBack Then
    Dim di As New IO.DirectoryInfo _
            (Server.MapPath("~\"))
```

```
    Dim fi As IO.FileInfo() = _
        di.GetFiles("*.jpg")
    Dim xtemp As String
    For i As Integer = 0 To UBound(fi)
        xtemp &= fi(i).Name & _
            IIf(i = UBound(fi), "", ",")
    Next
    Dim xtemp2() As String = _
        Split(xtemp, ",")
    ListBox1.DataSource = xtemp2
    ListBox1.DataBind()
End If
```

Langkah pertama yang dilakukan di dalam proses ini adalah mengambil informasi dari folder, yaitu nama file, khususnya file yang memiliki ekstensi .jpg. Selanjutnya nama file dalam folder root di web server dimasukkan ke dalam array, dan dimasukkan ke dalam ListBox.

Pengambilan informasi folder dilakukan menggunakan class *DirectoryInfo* sedangkan pengambilan informasi nama file menggunakan class *FileInfo*. Untuk kepentingan testing, copykan beberapa file gambar berekstensi .jpg dengan ukuran yang bervariasi.

2. Kemudian dobel klik di dalam Listbox, dan ketikkan listing berikut ini di dalam *ListBox1_SelectedIndexChanged* :

```
If CheckBox1.Checked Then
    Dim xbitmap As New _
        Drawing.Bitmap _
            (Server.MapPath(ListBox1.Text))
    Dim xthumb As New _
        Drawing.Bitmap(100, 100)
    xthumb = _
        xbitmap.GetThumbnailImage(100, 100, _
            Nothing, IntPtr.Zero)
    Dim thumbImage As String = _
        Server.MapPath("~\thumb.gif")
    xthumb.Save(thumbImage, _
        Drawing.Imaging.ImageFormat.Gif)
    System.Threading.Thread.Sleep(1000)
    Image1.ImageUrl = thumbImage
    xbitmap.Dispose()
Else
    System.Threading.Thread.Sleep(1000)
    Image1.ImageUrl = Server.MapPath("~\") & _
        ListBox1.Text
End If
```

3. Sedangkan untuk proses delay digunakan metode *thread.sleep* yang akan melakukan delay dengan satuan milidetik, sehingga dengan delay 1000 milidetik

diharapkan dapat memperlihatkan kinerja UpdateProgress.

Untuk pembuatan thumbnail sebuah file gambar, digunakan GDI+ yaitu dari namespace *System.Drawing* dengan menggunakan class *Bitmap*. Pada awal, diinisialisasi obyek Bitmap dari nama file yang dipilih di dalam ListBox. Selanjutnya, dibuat obyek Bitmap baru dengan ukuran 100 x 100 untuk menampung file thumbnail baru yang dibuat dengan metode *GetThumbnailImage*.
Kemudian file thumbnail tersebut disimpan ulang dengan format .gif dengan nama *thumb.gif* (untuk kepentingan di keadaan nyata, nama thumbnail bisa disesuaikan sehingga *unique* dan tidak bentrok jika terjadi proses yang lain).
Sedangkan untuk proses delay digunakan metode *thread.sleep* yang akan melakukan delay dengan satuan milidetik, sehingga dengan delay 1000 milidetik diharapkan dapat memperlihatkan kinerja UpdateProgress.

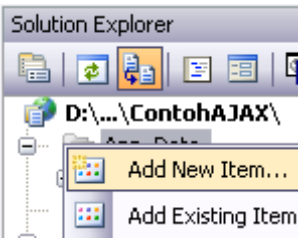
Contoh 4 : Akses Database dan Multiple UpdatePanel

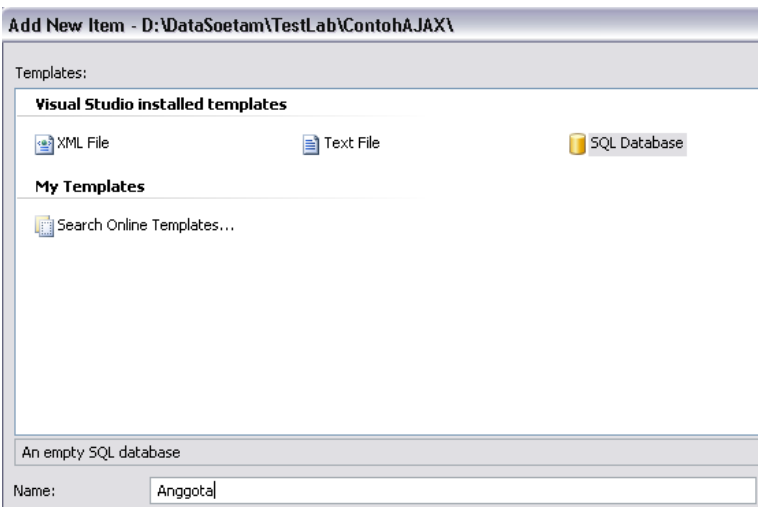
Contoh berikutnya merupakan contoh aplikasi *zero code* atau situs yang dalam proses pembuatannya tidak menggunakan pengetikan listing sama sekali. Contoh aplikasi ini, selain mendemonstrasikan penggunaan teknik AJAX dalam akses database serta penggunaan lebih dari satu komponen UpdatePanel di dalam satu halaman web.

Dalam contoh ini, akan dibuat sebuah proses sederhana untuk melakukan entri sebuah tabel biasa, kemudian ditampilkan dalam sebuah GridView di halaman yang sama. Jika situs ini dibuat tanpa menggunakan teknik AJAX, dipastikan efek flicker akan terjadi (jika di dalam web server lokal), dan akan juga terjadi proses full postback (jika di webhosting). Akibatnya, form untuk entri data yang seharusnya tidak ikut direfresh, akan ikut terefresh juga. Sedangkan untuk penggunaan teknik AJAX, yang akan dilakukan proses refresh hanya di dalam GridView, tidak secara keseluruhan.

Sebelum membuat situs, terlebih dulu dibuat sebuah database bernama *Anggota* di dalam SQL Server Express 2005, yang didalamnya akan berisi sebuah tabel dengan nama *Anggota* juga. Untuk pembuatan selengkapnya, ikuti langkah berikut ini :

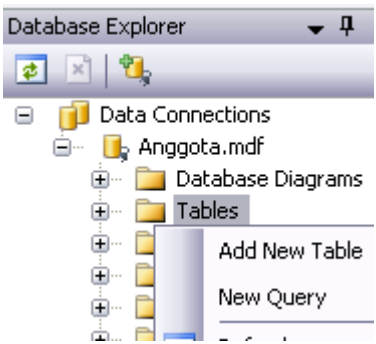
4. Buat sebuah solution baru bertipe *ASP .NET AJAX Enabled Website*.
5. Klik kanan di folder *App_Data* dan pilih sub menu *Add New Item*. Kemudian di dalam kotak dialog yang tersedia, pilih tipe SQL Database dan beri nama *Anggota.*





Secara default, SQL Server 2005 Express Edition terinstalasi bersamaan dengan instalasi Visual Web Developer Express Edition. SQL Server 2005 Express Edition sebenarnya memiliki utilitas sejenis *Enterprise Manager* (utilitas manajemen database di SQL Server 2000), bernama *SQL Server Studio Management Express* yang dapat didownloa secara terpisah dan gratis.
SQL Server 2005 Express sendiri juga memiliki lisensi gratis (meski tidak open source), dengan beberapa keterbatasan yang dapat dilihat di dalam lampiran mengenai perbandingan SQL Server 2005 secara lengkap per edisi.

6. Kemudian, jendela Database Explorer klik kanan pada sub tree *Table* dan pilih sub menu *Add New Table* lalu buat tabel dengan struktur seperti pada gambar berikut :



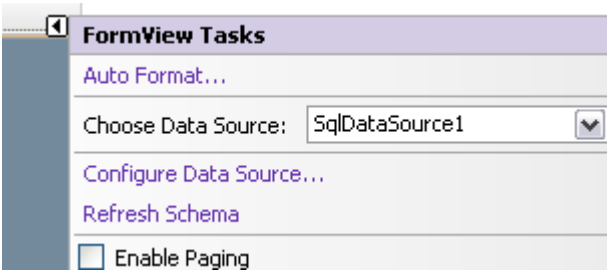
Column Name	Data Type	Allow Nulls
Nama	varchar(50)	☐
Alamat	varchar(50)	☑
Kota	varchar(50)	☑
Email	varchar(50)	☑

7. Setelah tabel selesai disimpan, di dalam halaman web *Default.aspx*, tempatkan dua buah komponen UpdatePanel didalamnya.
8. Kemudian, drag tabel *Anggota* dari Database Explorer ke dalam UpdatePanel pertama di halaman web.

Pada saat sebuah tabel dari SQL Server 2005 Express Edition didrag ke dalam sebuah halaman web, maka proses yang terjadi adalah pembentukan komponen *SQLDataSource*, komponen *GridView* dan modifikasi bagian *connection string* di dalam file konfigurasi *web.config* yang secara otomatis akan mengarah ke koneksi database yang tabelnya telah didrag sebelumnya.
Cara melakukan koneksi dengan menggunakan teknik drag and drop seperti ini seringkali disebut sebagai cara koneksi *zero code* karena tidak membutuhkan proses pengetikan listing sama sekali. Meski relatif sangat mudah, tetapi untuk beberapa studi kasus yang membutuhkan koneksi dengan pengaturan

yang kompleks, cara seperti ini tidak sesuai untuk dilakukan.

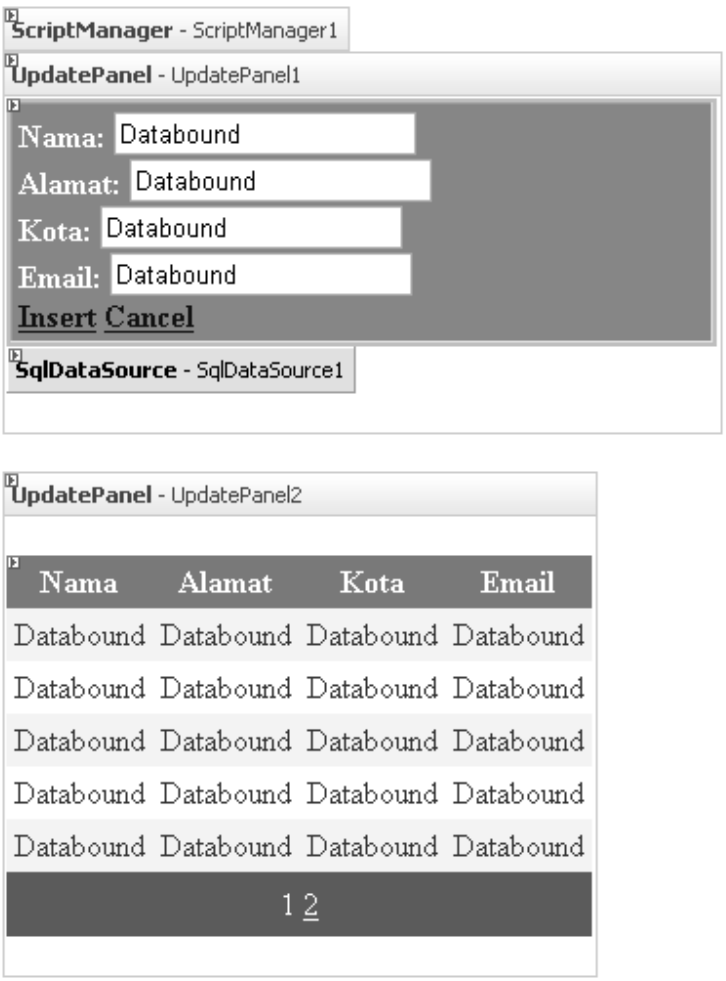
9. Setelah itu tambahkan komponen *FormView* di dalam UpdatePanel1 dan di dalam *smart tag*, arahkan koneksi ke *SQLDataSource1*. Kemudian atur property *Default Mode* menjadi bernilai *Insert*.



Komponen FormView lebih banyak digunakan untuk proses manipulasi data dalam sebuah tabel atau lebih. Secara default, komponen FormView akan memiliki nilai property *DefaultMode* yang berisi *ReadOnly* dan mampu melakukan operasi insert, edit dan juga delete. Tetapi dalam contoh kasus ini hanya dibatasi untuk operasi Insert.

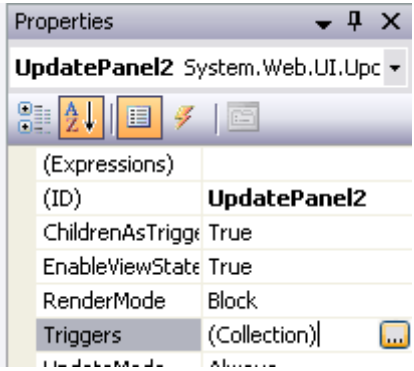
10. Kemudian pindahkan komponen *GridView* dengan cara drag and drop ke dalam komponen UpdatePanel2. Atur

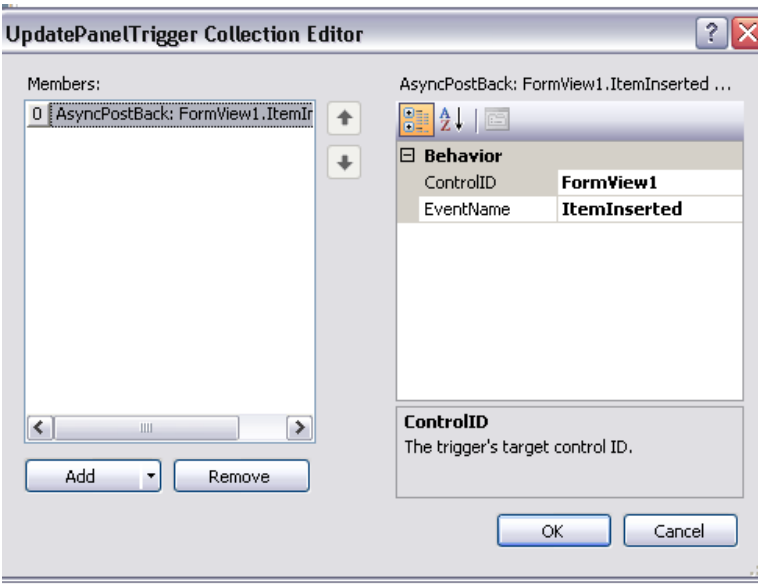
pula property *AllowPaging* pada GridView agar bernilai *true* dan property *PageSize* agar bernilai 5.



Pemindahan GridVIew ke dalam UpdatePanel2 dimaksudkan agar pada saat data selesai diinputkan di dalam FormView, GridView akan ikut terupdate pula dengan proses partial postback, tanpa harus melakukan refresh ulang ke seluruh bagian halaman web.

11. Berikutnya, klik di komponen UpdatePanel2 dan atur property *Triggers*, dengan menambahkan trigger dari behavior *AsyncPostBack* pada saat komponen FormView selesai melakukan inputan data atau pada saat event *ItemInserted*





12. Kemudian cobalah untuk melakukan eksekusi terhadap situs tersebut dengan menekan kombinasi tombol Ctrl+F5.

Penambahan trigger pada sebuah UpdatePanel akan menyebabkan sebuah UpdatePanel "dipaksa" melakukan proses refresh seluruh komponen yang ada didalamnya secara asynchronous berdasarkan event yang didefinisikan di dalam trigger tersebut.

Untuk testing, tambahkan sekitar 10 data dalam FormView sehingga proses partial paging juga dapat terlihat di GridView.

Contoh 5 : Cascading DropDownList

Contoh berikutnya untuk penggunaan UpdatePanel adalah membuat sebuah halaman web yang didalamnya terdapat *Cascading DropDownList* atau DropDownList majemuk yang antara satu sama lain saling berhubungan. Penggunaan jenis ini sangat sering terjadi di sebuah halaman web, khususnya di dalam proses pengisian sebuah form, baik form untuk registrasi anggota ataupun pengisian data dalam sebuah content management system.

Di dalam pembuatan Cascading DropDownList tanpa menggunakan teknik AJAX, umumnya akan menemui kendala pada saat sebuah DropDownList (terutama yang pertama), diganti itemnya. Karena pada saat proses tersebut dilakukan, halaman web akan melakukan refresh dan mengambil nilai dari tabel yang bersesuaian untuk DropDownList berikutnya. Tentu saja hal ini akan menyebabkan halaman web berulangkali melakukan loading ke server, sehingga jika halaman web tersebut berada di sebuah web hosting, akan menyebabkan halaman web berulangkali flickr dan pengguna situs secara otomatis seperti merasa "kehilangan" jejak pengisian data.

Tetapi, dengan menggunakan teknik AJAX, diharapkan proses tunggu tidak akan menyebabkan

pengguna situs merasa frustrasi akibat halaman web yang flickr ataupun blank pada saat sebuah DropDownList memanggil data yang bersesuaian dari server.

Pada contoh kasus ini akan dibuat sebuah halaman web untuk mendemonstrasikan penggunaan Cascading DropDownList dengan menggunakan pengisian berhirarkis untuk mengisi data sebuah desa. Sebuah desa, dalam kasus ini, merupakan bagian dari sebuah kecamatan dan kecamatan merupakan sebuah bagian dari kabupaten. Sedangkan hirarki tertinggi merupakan propinsi yang memiliki banyak kabupaten didalamnya.

Untuk kepentingan tersebut, nantinya akan dibuat empat buah tabel dalam sebuah tabel yang masing-masing akan diisi dengan data contoh (kecuali tabel desa yang diisi dari halaman web). Dan berikut ini adalah langkah pembuatan dari studi kasus ini :

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Enabled Web Site
2. Kemudian buat sebuah database baru dengan nama *DataPenduduk* yang didalamnya memiliki empat tabel dengan struktur masing-masing tabel tampak seperti pada gambar berikut ini :

Tabel Propinsi

Column Name	Data Type	Allow Nulls
NamaPropinsi	varchar(50)	☐
StatusPropinsi	bit	☐

Tabel Kabupaten

Column Name	Data Type	Allow Nulls
NamaPropinsi	varchar(50)	☐
NamaKabupaten	varchar(50)	☐
StatusKabupaten	bit	☐

Tabel Kecamatan

Column Name	Data Type	Allow Nulls
NamaPropinsi	varchar(50)	☐
NamaKabupaten	varchar(50)	☐
NamaKecamatan	varchar(50)	☐
StatusKecamatan	bit	☐

Tabel Desa

Column Name	Data Type	Allow Nulls
NamaPropinsi	varchar(50)	☐
NamaKabupaten	varchar(50)	☐
NamaKecamatan	varchar(50)	☐
NamaDesa	varchar(50)	☐
StatusDesa	bit	☐

3. Selanjutnya, isikan data contoh dari tiap tabel seperti pada gambar berikut (isi data bisa ditambah sesuai selera, karena hanya untuk kepentingan testing)

Tabel Propinsi

NamaPropinsi	StatusPropinsi
Jawa Tengah	True
Jawa Timur	True

Tabel Kabupaten

NamaPropinsi	NamaKabupaten	StatusKabupaten
Jawa Tengah	Temanggung	True
Jawa Timur	Gresik	True
Jawa Timur	Sidoarjo	True
Jawa Timur	Surabaya	True

Tabel Kecamatan

NamaPropinsi	NamaKabupaten	NamaKecamatan	StatusKecamatan
Jawa Timur	Sidoarjo	Candi	True
Jawa Timur	Surabaya	Rungkut	True
Jawa Timur	Surabaya	Sukolilo	True

Perhatikan keempat tabel yang telah dibuat, dan primary key dari masing-masing tabel yang saling berelasi satu sama lain. Tiap tabel akan mengacu ke tabel *parent* atau tabel induknya untuk mendapatkan relasi yang tepat.

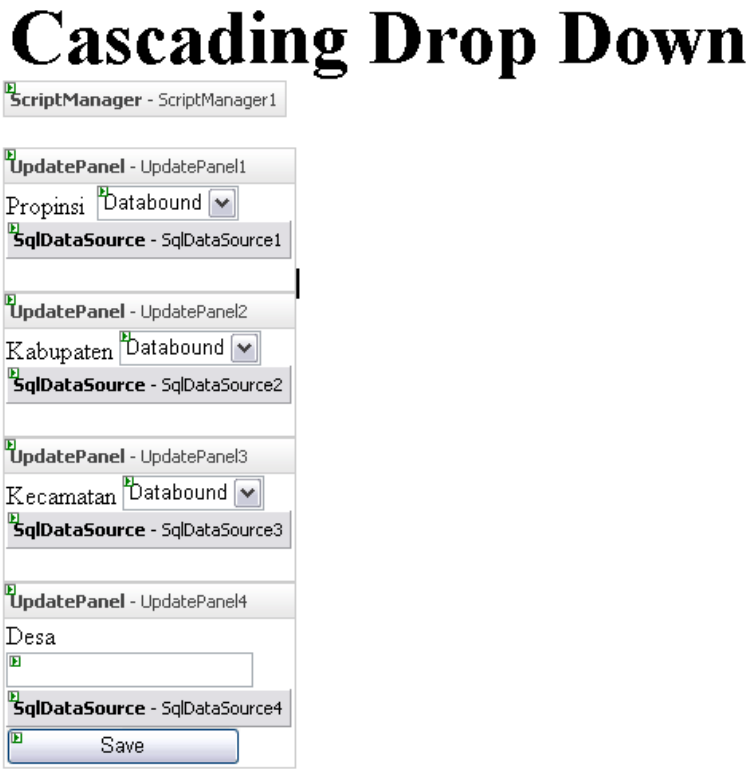
4. Kini, untuk halaman web yang akan ditempati *cascading dropdownlist*, ikuti langkath berikut :

a. Tempatkan empat buah komponen UpdatePanel
b. Set property *UpdateMode* pada tiap UpdatePanel menjadi *Conditional*

Kunci utama dari implementasi Cascading DropDownList adalah adanya setting property UpdateMode menjadi Conditional. Sehingga pada saat terjadi pemilihan data di tiap DropDownList, maka hanya DropDownList dengan level dibawahnya saja yang akan mengalami proses update.

c. Di tiap UpdatePanel, drag tiap tabel yang tersedia secara berurutan yaitu tabel Propinsi, Kabupaten, Kecamatan dan Desa.
d. Dari tiap hasil drag tabel tersebut, hapus tiap GridView yang terbentuk.
e. Selanjutnya tempatkan DropDownList di tiap UpdatePanel, kecuali di dalam UpdatePanel yang ditempati oleh tabel Desa.
f. Kemudian arahkan tiap DropDownList ke komponen SQLDataSource di tiap UpdatePanel yang bersesuaian.

g. Khusus untuk UpdatePanel yang keempat (yang ditempati oleh tabel Desa), tempatkan sebuah Textbox dan sebuah Button didalamnya.



h. Dobel klik di Button dalam UpdatePanel yang terakhir, dan ketikkan listing berikut :

```
With SqlDataSource4
    .InsertParameters("NamaPropinsi"). _
```

```
            DefaultValue = _
                DropDownList1.SelectedValue
        .InsertParameters("NamaKabupaten"). _
            DefaultValue = _
                DropDownList2.SelectedValue
        .InsertParameters("NamaKecamatan"). _
            DefaultValue = _
                DropDownList3.SelectedValue
        .InsertParameters("NamaDesa"). _
            DefaultValue = TextBox1.Text
        .InsertParameters("StatusDesa"). _
            DefaultValue = "True"
        .Insert()
    End With
```

Jika contoh Cascading DropDownList ini diaplikasikan secara online di sebuah web hosting, maka diperlukan penambahan komponen UpdateProgress untuk menandakan bahwa satu atau lebih DropDownList sedang dalam keadaan melakukan refresh data berdasarkan data dari DropDownList yang memliki hirarki diatasnya.

Contoh 6 : Image Rotator

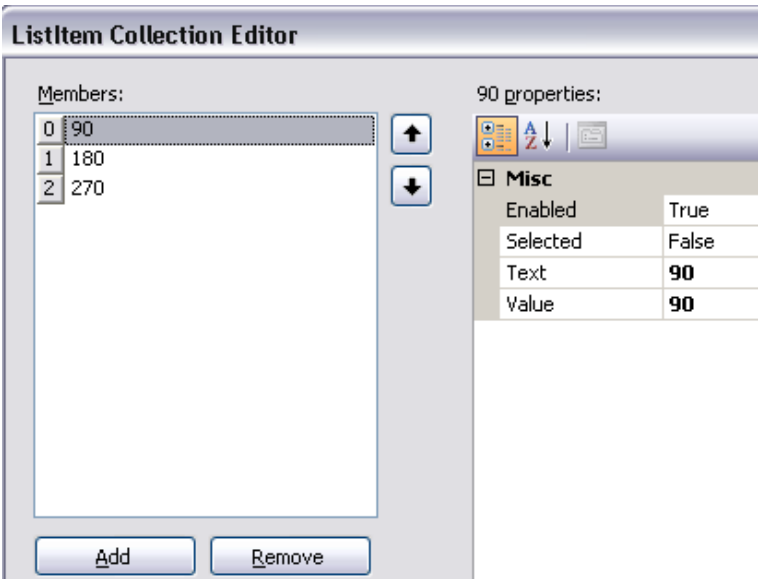
Contoh berikut merupakan contoh sederhana untuk menggabungkan penggunaan GDI+ dengan teknik yang sedikit berbeda (perhatikan contoh sebelumnya mengenai galeri gambar sederhana) dengan proses partial postback.

Dengan menggunakan teknik sederhana, dan tentu saja menggunakan teknik AJAX, maka bisa dibuat sebuah halaman web yang dapat berfungsi menjadi image manipulator sederhana. Dalam contoh ini teknik manipulasi yang diambil hanya terbatas pada rotasi sebuah gambar.

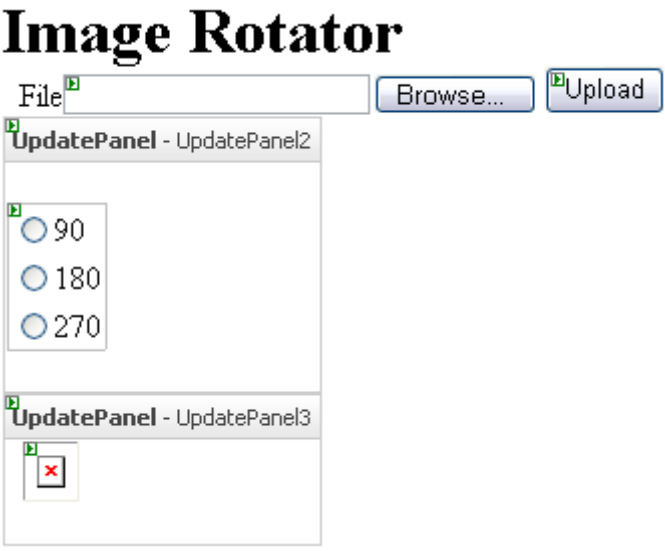
1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Website.
2. Tempatkan sebuah komponen FileUpload dan sebuah Button di dalam halaman web tersebut.

Proses upload membutuhkan proses postback, karenanya hindari penempatan komponen FileUpload di dalam sebuah komponen UpdatePanel, karena jika hal tersebut dilakukan, proses upload tidak akan berjalan sebagaimana mestinya.

3. Setelah itu, tempatkan dua buah komponen UpdatePanel tepat di bawah komponen FileUpload tersebut.
4. Khusus untuk UpdatePanel yang pertama, set property *UpdateMode* menjadi *Conditional*
5. Berikutnya, letakkan sebuah Listbox dalam UpdatePanel yang pertama dan set property *AutoPostBack* dari Listbox tersebut menjadi *true*. Masih di dalam Listbox, isikan item pada kotak dialog untuk property *Items* untuk sudut perputaran dari gambar nantinya.



6. Kemudian di dalam UpdatePanel yang kedua, drag sebuah komponen *Image* didalamnya.



7. Kini, dobel klik di dalam Button untuk proses upload dan isikan listing berikut :

```
Dim xfile As String = _
    Server.MapPath _
    ("~/" & FileUpload1.FileName)
FileUpload1.SaveAs(xfile)
Image1.ImageUrl = xfile
```

8. Langkah terakhir adalah melakukan dobel klik di dalam Listbox dan mengetikkan listing berikut :

```
Dim xGmbr As New _
    Bitmap(Image1.ImageUrl)
With xGmbr
    .RotateFlip(RadioButtonList1. _
        SelectedIndex + 1)
    Dim xformat As String = _
```

```
        Server.MapPath("temp." & _
        .RawFormat.GetType.ToString)
    .Save(xformat, .RawFormat)
    Image1.ImageUrl = xformat
```

Proses rotasi gambar sesungguhnya adalah proses yang sangat sederhana, karena hanya membutuhkan presedur *RotateFlip* yang telah ada didalam namespace Drawing.
Dalam contoh tersebut, penamaan file gambar hasil rotasi diseragamkan menjadi satu nama yaitu *temp*.
Dalam sebuah situs yang nantinya menjadi sebuah situs online dengan kemampuan menangani pengguna secara majemuk, penamaan file dapat diganti dengan sesuatu yang unique, misalnya dari session unique hasil login tiap user ataupun dari sebuah bilangan atau kata acak yang telah digenerate sebelumnya.

Image Rotator

File Browse... Button

- 90
- 180
- 270

Studi Kasus I : Simple Address Book

Pengantar

Studi kasus berikut ini merupakan studi kasus sederhana yang akan menggambarkan penggunaan AJAX dalam sebuah aplikasi database sederhana. Dikatakan sederhana karena didalamnya hanya memiliki dua tabel dalam sebuah database. Tetapi cukup layak dijadikan lebih dari sekedar contoh biasa, karena dalam situs sederhana berikut telah mencakup demonstrasi penggunaan proses login dan beberapa proses partial postback yang lebih kompleks dibandingkan contoh sebelumnya.

Studi kasus yang akan dikerjakan merupakan sebuah buku alamat atau address book sederhana yang menempatkan sebuah komponen Repeater, Gridview serta Formview untuk proses manipulasi tabel utama. Serta pemanfaatan Textbox biasa untuk proses login serta penggantian password. Didalamnya juga akan didemonstrasikan penggunaan Session untuk menampilkan maupun menyembunyikan sebuah komponen ataupun bagian dari sebuah komponen dalam sebuah halaman web.

Rancangan dari situs address book ini merupakan rancangan dengan menggunakan *single page solution* atau hanya satu halaman web untuk membuat sebuah situs. Rancangan jenis ini, selain untuk mempermudah

pembuatan situs, juga untuk mendemonstrasikan "kekuatan" dari penggunaan teknik AJAX dalam situs ini.

Tidak semua situs dapat menerapkan rancangan dengan jenis *single page solution*. Beberapa situs yang terbilang "kompleks" akan menjadi lebih sulit implementasinya jika menerapkan rancangan jenis ini.
Umumnya, rancangan jenis ini diimplementasikan lebih banyak pada halaman utama dari sebuah situs yang menggunakan teknik AJAX. Hal tersebut selain memudahkan perancangan, juga akan membuat situs menjadi lebih menarik bagi pengguna. Sebab pengguna seakan-akan tidak pernah "meninggalkan" situs meskti terjadi proses postback dalam halaman web yang sedang dikunjungi.

Penggambaran kerangka halaman web dalam situs ini, terlihat pada gambar berikut :

..:: Simple Address Book ::..
Password
*****
Login
Pass Lama
*****
Pass Baru
*****
Ganti Pass
Logout
Repeater yang menampilkan rangkaian alfabet dari huruf depan nama di data address book
A B C D E .....
GridView untuk menampilkan detail data dari address book saat huruf depan dalam Repeater diklik oleh pengguna Dan pada saat proses login selesai dilakukan, GridView ini memiliki kemampuan tambahan untuk melakukan proses hapus data.
Nama : .....
Alamat : .....
Kota : .....
Telp : .....
HP : .....
FormView untuk melakukan input data baru di address book.
FormView ini hanya bisa dilihat setelah proses login dilakukan
Nama : .....
Alamat : .....
Kota : .....
Telp : .....
HP : .....

Dari kerangka rancangan tersebut, terdapat tiga bagian yang akan berpengaruh pada saat proses login dilakukan, yaitu bagian untuk mengganti password, FormView untuk melakukan input data baru serta bagian dari GridView yang nantinya digunakan untuk menghapus data.

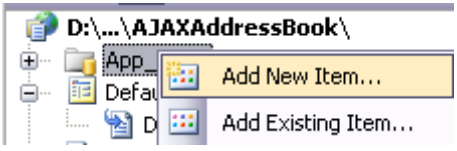
Proses login sendiri hanya menetapkan satu jenis user yaitu untuk administrator dari situs itu sendiri. Sehingga pada saat proses login, tidak terdapat isian untuk melakukan entri user id, tetapi langsung menuju ke isian password.

Untuk pengembangan lebih lanjut, Anda dapat menambahkan user majemuk dalam situs tersebut. Sehingga nantinya akan terdapat isian user id didalamnya.

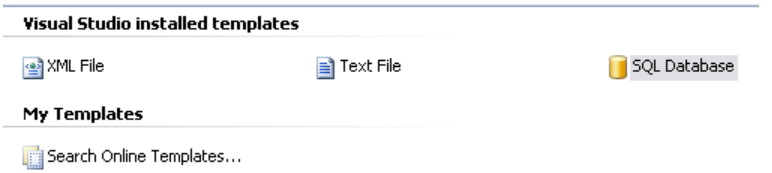
Khusus untuk bagian penggantian password dan bagian login, nantinya akan menggunakan komponen MultiView dengan bantuan dua buah komponen View didalamnya. Sehingga dapat mempermudah implementasi dari teknik AJAX yang akan dilakukan.

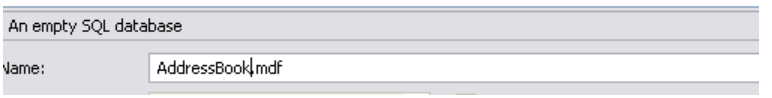
Sebelum memulai pembuatan studi kasus ini, disarankan untuk menyediakan waktu minimal dua jam, untuk membaca sekaligus mencoba langkah demi langkah yang ada di bagian studi kasus ini.
Sebab, jika studi kasus ini dikerjakan secara terpotong-potong, kemungkinan besar pembaca akan lupa bagian terakhir yang telah dikerjakan. Selain itu, disarankan pula untuk membaca terlebih dahulu hingga tuntas langkah demi langkah yang akan dilakukan sebelum mencoba implementasi situs.

Langkah pertama yang dilakukan dalam pembuatan studi kasus ini adalah membuat sebuah solution baru di dalam VWD Express dengan memilih tipe ASP .NET AJAX Website. Selanjutnya, di bagian Solution Explorer, klik kanan pada folder *App_Data* dan pilih sub menu *Add New Item*.



Lalu dalam kotak dialog yang tersedia buat sebuah database dengan nama *AddressBook*. Langkah berikutnya dalam pembuatan database akan dijelaskan di sub bab berikutnya.


Visual Studio installed templates
XML File
Text File
SQL Database
My Templates
Search Online Templates...


An empty SQL database
Name:
AddressBook.mdf

Pembuatan Database

Setelah database selesai dibuat, maka dalam database baru tersebut, akan dibuat dua buah tabel sederhana dengan struktur tabel sebagai berikut :

Tabel Password

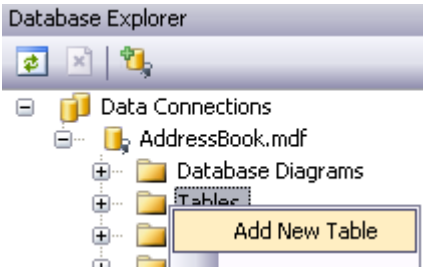
Column Name	Data Type	Allow Nulls
password	nchar(10)	☐

Tabel Address

Column Name	Data Type	Allow Nulls
Nama	varchar(50)	☐
Alamat	varchar(50)	☐
Kota	varchar(50)	☐
Telp	varchar(50)	☐
HP	varchar(50)	☐

Di dalam tabel *Address* terdapat dua field yang menjadi primary key, yaitu field *Nama* dan field *HP*. Untuk membuat dua field yang terpisah menjadi primary key, klik salah satu field, kemudian tekan tombol Ctrl (tanpa dilepas) dan klik field satunya lagi. Setelah itu, klik icon  untuk membuat kedua field tersebut menjadi primary key.

Pembuatan tabel dilakukan dengan melakukan klik kanan pada bagian Database Explorer dan memilih sub menu *Add New Table*.



Selanjutnya, untuk kepentingan testing situs, entrikan data contoh berikut ini :

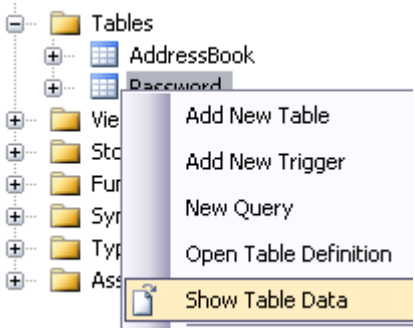
Tabel Password

password
admin

Tabel Address

Nama	Alamat	Kota	Telp	HP
Ali	Jl. Kota 1	Surabaya	18273662	080000212
Baba	Jl. Desa 2	Surabaya	29382827	080292873
Caca	Jl. Koki 3	Sidoarjo	2342342	039383222

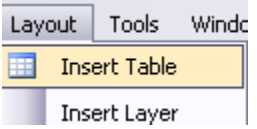
Untuk melakukan proses entri data contoh dapat melakukan klik kanan pada tabel yang akan diisi dan memilih sub menu *Show Table Data*.



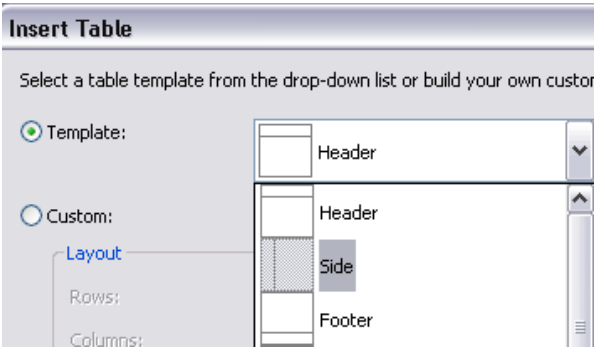
Anda dapat mengubah isi tabel sesuai selera. Tapi pastikan bahwa isi data di tabel *Password* hanya terdiri dari satu record.

Desain Halaman Web

Langkah berikutnya, setelah menyelesaikan pembuatan database, adalah melakukan desain dari satu-satunya halaman web yang akan dibuat dalam studi kasus ini. Pertama, berpindahlah terlebih dulu ke mode Design dari halaman web *Default.aspx* yang tersedia. Kemudian tempatkan sebuah tabel didalamnya dengan menggunakan template *Side*. Pembuatan tabel ini bisa dilakukan dengan memilih menu *Layout → Insert Table*.



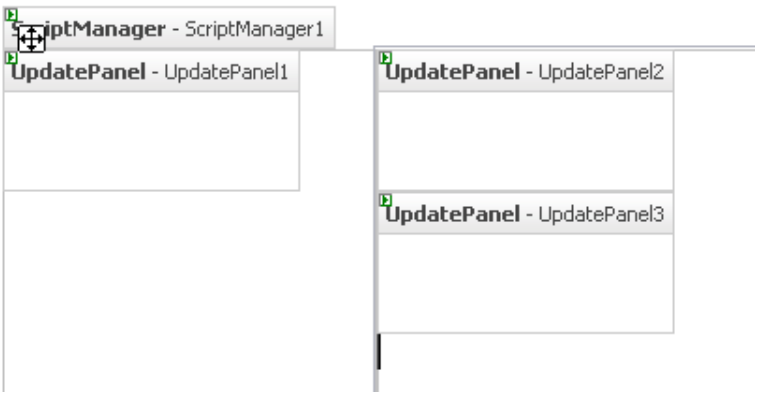
Selanjutnya, di kotak dialog yang tersedia, pilih opsi template *Side* yang terdiri dari dua kolom dan satu baris untuk satu halaman web penuh.



Kemudian, blok kedua kolom tersebut dan lihat di bagian window Properties. Set property *valign* dari dua

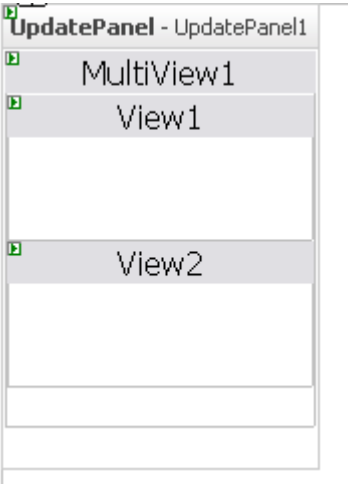
kolom tersebut menjadi *top*. Property ini diset agar desain yang terjadi akan langsung merapat ke bagian atas dari tabel.

Berikutnya, drag tiga buah komponen Update Panel, masing-masing di bagian kolom kiri satu, dan dua lainnya di bagian kolom sebelah kanan. Hasil dari proses drag akan tampak seperti pada gambar berikut ini.

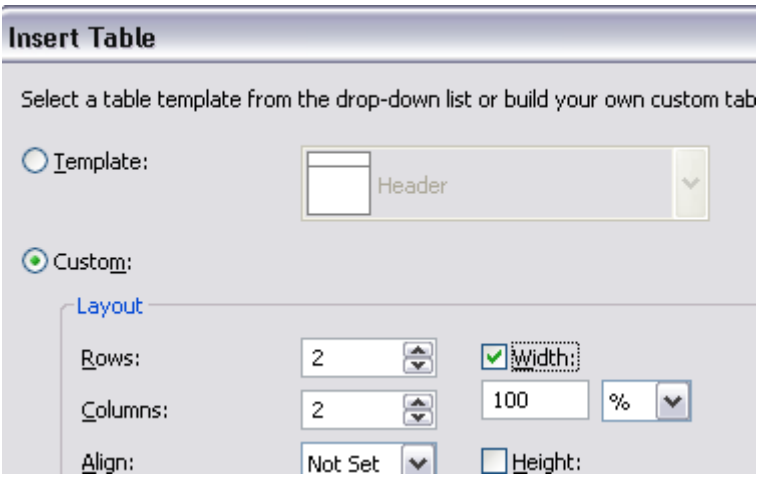


Penempatan Update Panel berdasarkan pada proses yang akan dilakukan di dalam halaman web tersebut.

Kemudian, di komponen UpdatePanel yang pertama di kolom sebelah kiri, drag sebuah komponen MultiView, dan juga dua buah komponen View di dalam MultiView tersebut.



Lalu, di View yang pertama tempatkan sebuah tabel dengan jumlah kolom dan baris sebanyak dua. Cara menempatkan tabel tersebut dengan meletakkan cursor di area View1, kemudian pilih menu *Layout → Insert Table*. Selanjutnya di kotak dialog yang tersedia, pilih opsi *Row* dan isi dengan nilai 2, dan begitu pula dengan opsi *Column* yang ada. Jangan lupa untuk memberi nilai property *Width* dengan nilai 100%.

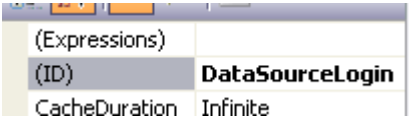


Kemudian, drag sebuah Textbox di kolom kedua baris pertama dan sebuah Button di kolom pertama baris kedua. Tambahkan juga sebuah Label di kolom kedua baris kedua. Label ini nantinya akan menjadi tempat penampung pesan kesalahan jika proses login gagal. Karenanya, set property *Text* dari Label ini menjadi kosong.

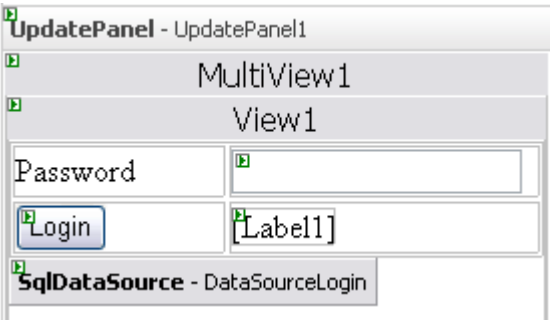
Dalam Textbox yang tersedia, set property *TextMode* menjadi mode *Password*. Hal ini agar pada saat pengisian password oleh pengguna, akan timbul *character mask* (biasanya berupa tanda * atau tanda bullet) yang umumnya ada di pengisian password.

Text	
TextMode	**Password**
ToolTip	

Langkah terakhir di View1 adalah melakukan drag tabel *Password* di bawah tabel yang tersedia. Dari hasil drag tersebut, hapus GridView yang terbentuk. Lalu, ganti property *ID* dari SqlDataSource menjadi *DataSourceLogin.*

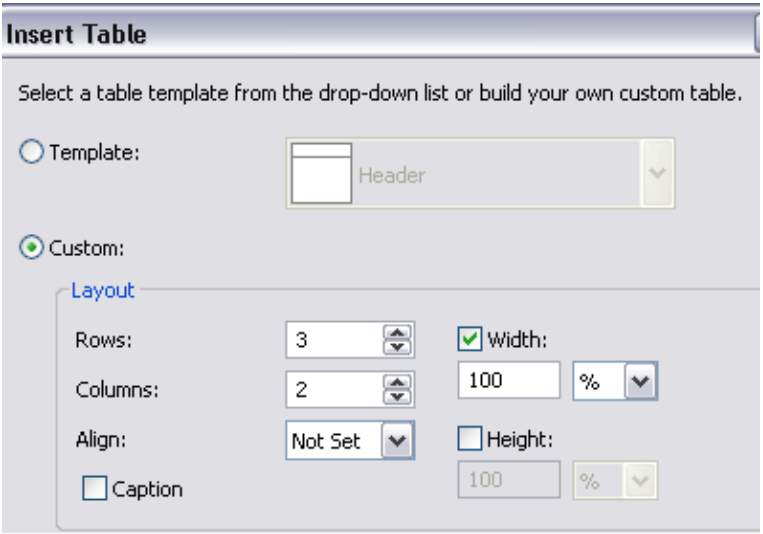


Hasil akhir dari desain untuk View1 terlihat seperti pada gambar berikut ini.



Isi dari proses Button Login dan modifikasi dari komponen *DataSourceLogin* akan dibahas lebih lanjut di sub bab Proses Login

Kini untuk View yang kedua, tempatkan lagi sebuah tabel dengan jumlah baris sebanyak 3 dan dan jumlah kolom sebanyak 2. Cara pembuatan tabel ini sama dengan langkah yang ada di bagian View1.



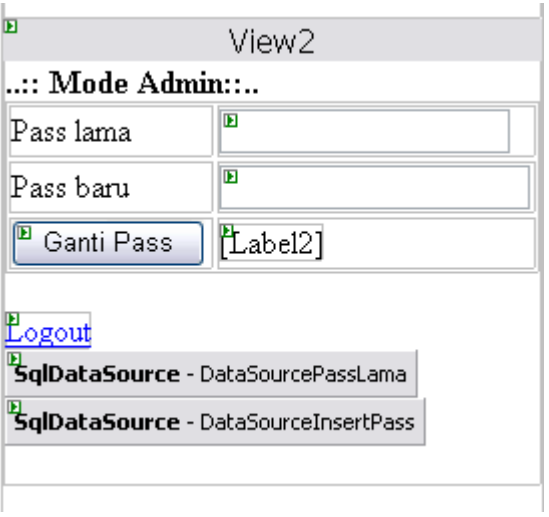
Lalu, drag dua buah Textbox di kolom kedua, masing-masing di baris pertama dan kedua. Textbox yang pertama akan digunakan untuk mengisikan password yang lama, sedangkan Textbox yang kedua untuk mengisikan password yang baru dalam proses ganti password. Gantilah property *Textmode* dari Textbox yang pertama menjadi *Password*.

Selanjutnya, di baris yang ketiga, drag sebuah Button di kolom yang pertama dan sebuah Label di kolom kedua. Untuk Button yang ada, ganti property *Text*

menjadi *Ganti Pass*, sedangkan untuk Label ganti property *Text* menjadi kosong.

Lalu drag sebuah LinkButton tepat di bawah tabel yang telah dibuat. Ganti property *Text* dari LinkButton tersebut menjadi *Logout*.

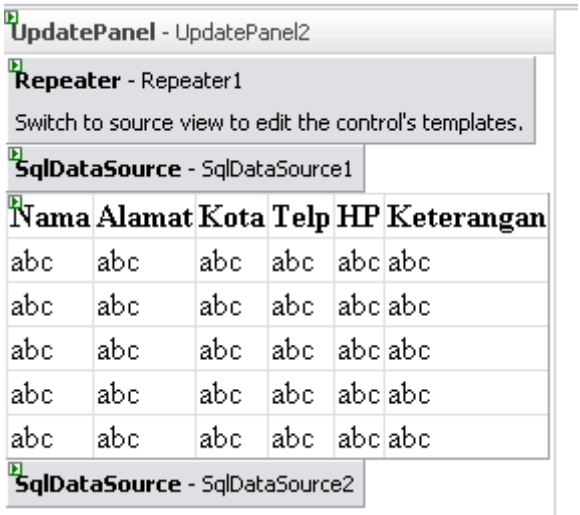
Langkah terakhir di View2 adalah melakukan drag tabel *Password* ke bawah Linkbutton, sebanyak dua kali. Dari hasil drag dua kali tersebut, hapus tiap Gridview yang terbentuk secara hati-hati. Kemudian, ganti property *ID* dari tiap SqlDataSource yang ada, masing-masing menjadi *DataSourcePassLama* dan *DataSourceInsertPass*.



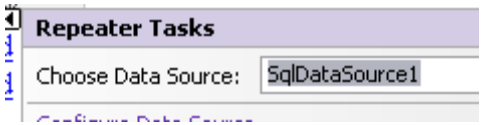
Komponen SqlDataSource dengan nama *DataSourcePassLama* akan digunakan untuk melakukan pengecekan terhadap password yang lama sebelum diganti. Sedangkan komponen SqlDataSource dengan nama *DataSourceInsertPass* akan digunakan untuk melakukan penggantian password.
Modifikasi dari kedua komponen SqlDataSource tersebut akan dibahas lebih lanjut di sub bab Ganti Password.

Kini untuk kolom sebelah kanan dari halaman web, akan dimulai dari Update Panel yang kedua. Dalam komponen UpdatePanel2, tempatkan sebuah komponen Repeater yang nanti akan berfungsi untuk menampilkan huruf depan dari data Address Book yang ada. Kemudian drag dua kali tabel *Address* dari Database Explorer, ke bagian bawah dari Repeater. Lalu hapus GridView yang terbentuk dari hasil drag yang pertama (dari komponen SqlDataSource1).

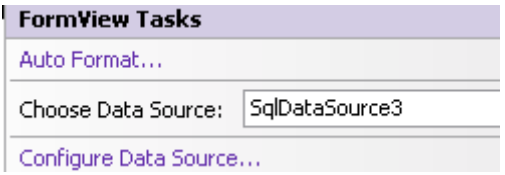
Modifikasi dari tiap SqlDataSource akan dibahas di sub bab Tampilan Utama.



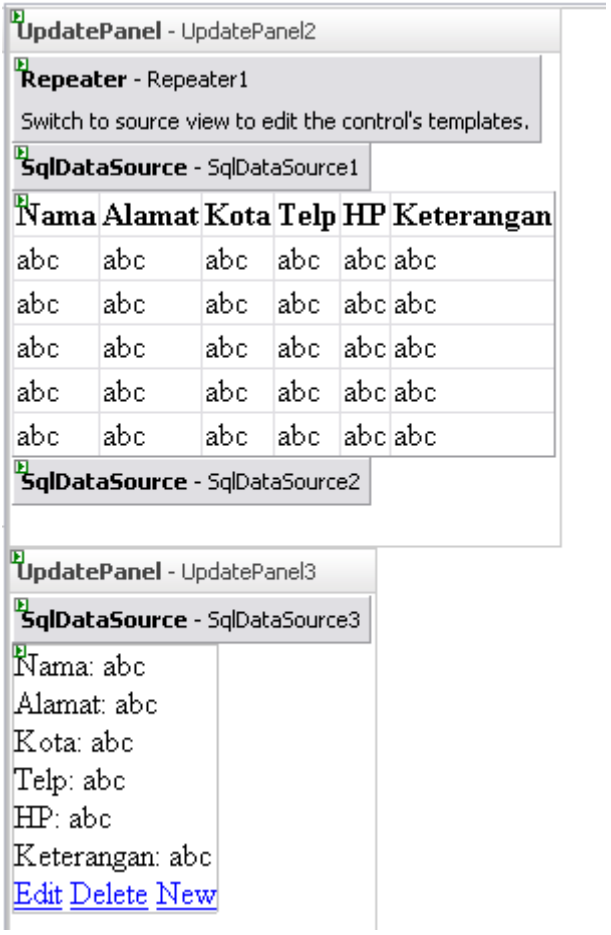
Selanjutnya, klik pada Smart Tag yang ada di Repeater, dan pada opsi *Data Source* arahkan ke komponen *SqlDataSource1*.



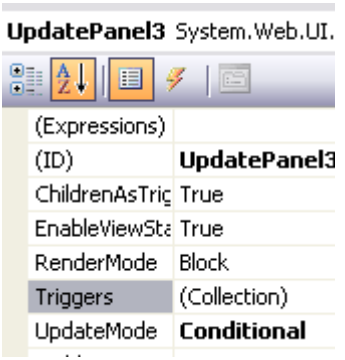
Sekarang untuk Update Panel yang terakhir, yaitu UpdatePanel3, drag (sekali lagi) tabel *Address* dari Database Explorer. Lalu hapus GridView yang terbentuk dari hasil drag tersebut. Kemudian drag sebuah komponen FormView dibawahnya, dan klik pada Smart Tag dari FormView tersebut. Isi opsi *Choose Data Source* ke komponen SqlDataSource3 yang baru terbentuk tersebut.



Hasil dari desain untuk kolom sebelah kanan terlihat pada gambar berikut.



Berikutnya, klik pada UpdatePanel3 dan set property *UpdateMode* menjadi *Conditional*. Setting ini nantinya akan dibutuhkan pada saat proses login dilakukan.

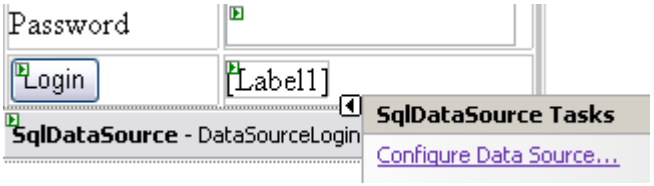


Modifikasi dari tiap FormView akan dibahas kemudian pada sub bab Input Data Baru.

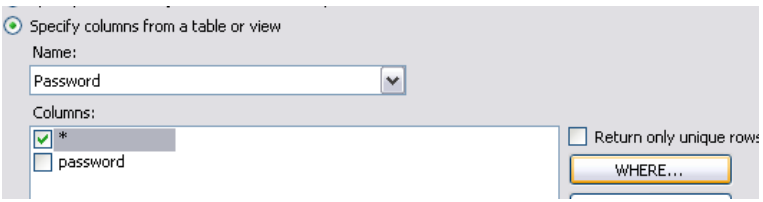
Proses Login

Setelah proses desain selesai dilakukan, maka proses pertama yang dilakukan adalah melakukan penyelesaian proses login. Pada proses login, langkah pertama yang harus dilakukan adalah memodifikasi komponen SqlDatasource dengan id *DataSourceLogin*.

Klik pada Smart Tag yang pada komponen *DataSourceLogin* dan pilih sub menu *Configure Data Source* untuk memodifikasi perintah SQL yang tersedia agar nantinya dapat membaca parameter dari Textbox.

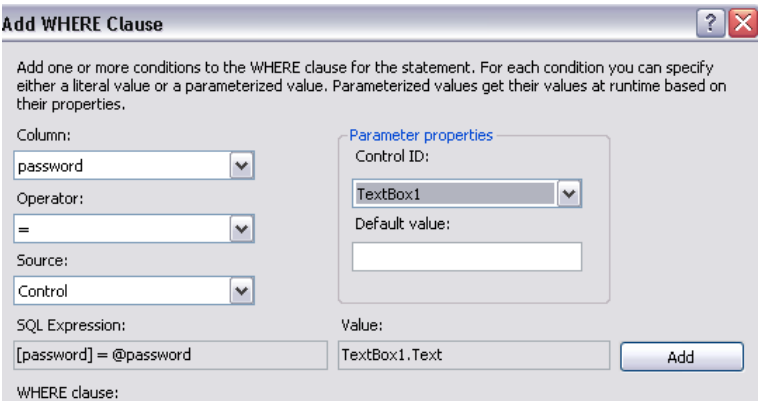


Kemudian pada kotak dialog yang tersedia, klik pada tombol *Where* untuk mengentrikan parameter yang akan diambil.



Di kotak dialog berikutnya, pilih field *Password* dan komponen *Textbox1* sebagai parameter yang akan

dijadikan acuan dalam perintah SQL nantinya. Setelah itu klik tombol *Add* untuk menambahkan parameter tersebut.



Penempatan parameter melalui sebuah komponen dalam halaman web akan mempermudah proses eksekusi perintah SQL, dan juga akan menghemat pengetikan listing program.
Dalam proses ini, parameter perintah SQL diambil dari Textbox1 sebagai pengecekan terhadap password yang diketikkan oleh pengguna.

Berikutnya, lakukan dobel klik pada Button Login dan mengetikkan listing berikut ini didalamnya.

```
If cekData(DataSourceLogin) Then
    MultiView1.ActiveViewIndex = 1
    FormView1.Visible = True
```

```
        Session("Login_OK") = True
        UpdatePanel2.DataBind()
        UpdatePanel3.Update()
        Label1.Text = ""
        Label2.Text = ""
    Else
        Label1.Text = "Login gagal !"
    End If
```

Pada saat proses login berhasil dilakukan, maka akan dibentuk sebuah Session baru sebagai penanda untuk mode dalam GridView tampilan data sekaligus tanda dalam menampilkan FormView. Selain itu, MultiView juga diarahkan ke View yang kedua.

Sedangkan jika proses login gagal, maka akan ditampilkan pesan kesalahan dengan menggunakan bantuan Label2 sebagai tempat penampung pesan.

Karena Gridview dan Formview terletak dalam Update Panel yang berbeda, maka kedua Update Panel tersebut harus "dipaksa" untuk melakukan proses *postback*.

Untuk UpdatePanel2, karena yang akan dilakukan proses refresh data dalam Gridview, maka proses refresh ditempatkan dengan menggunakan method *DataBind*. Sedangkan untuk UpdatePanel3, karena yang akan dilakukan proses refresh nantinya adalah property *Visible* dari Formview, maka menggunakan method *Update*.

Langkah terakhir dalam proses login ini adalah melakukan pembuatan fungsi *cekData* yang digunakan sebagai proses validasi pengecekan data. Fungsi ini nantinya juga akan digunakan dalam proses penggantian password. Untuk melakukan langkah ini, ketikkan listing berikut di dalam mode pengetikan listing program.

```
Function cekData _
    (ByVal xSource As SqlDataSource) _
    As Boolean
    xSource.DataBind()
    Dim xtabel As New Data.DataView
    xtabel = xSource.Select _
        (DataSourceSelectArguments.Empty)
    If xtabel.Count > 0 Then
        cekData = True
    Else
        cekData = False
    End If
End Function
```

Fungsi *cekData* digunakan untuk melakukan validasi apakah perintah SQL yang ada dalam sebuah komponen SqlDataSource memiliki data didalamnya. Proses pengecekan yang terjadi sangat sederhana, karena dalam perintah SQL yang terdapat dalam SqlDataSource telah diisikan parameter yang

bersesuaian, sehingga dalam pemanggilannya, perintah Select tidak lagi diberi parameter (DataSourceSelectArguments diberi property *Empty*).

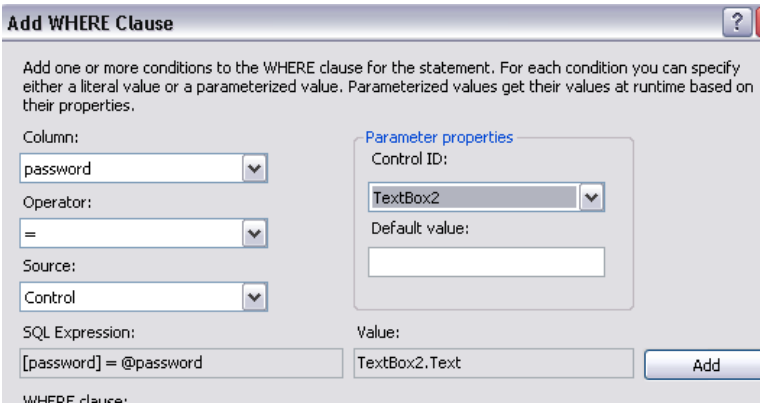
Ganti Password

Setelah proses login selesai dikerjakan, maka langkah berikutnya adalah membuat proses penggantian password. Pada proses penggantian password ini dilakukan dalam dua langkah, yaitu melakukan pengecekan pada password yang lama, sekaligus melakukan penggantian password yang baru. Karena password yang diganti merupakan primary key dari tabel,dan hanya terdiri dari satu record, maka proses penggantian tidak bisa dilakukan dengan proses update biasa. Tetapi dengan menggunakan teknik penghapusan dan penginputan data baru didalamnya.

Langkah pertama dalam proses ganti password adalah melakukan modifikasi terhadap kedua komponen SqlDataSource yang ada di View2. Modifikasi pertama dilakukan pada komponen SqlDataSource yang memiliki nama *DataSourcePassLama*.

Klik pada Smart Tag di komponen tersebut, kemudian pilih sub menu *Configure Data Source*. Selanjutnya, lakukan langkah yang sama persis dengan modifikasi pada komponen SqlDataSource sebelumnya (di proses login), tetapi parameter yang diambil bukan lagi dari Textbox1, melainkan dari Textbox2. Proses

selengkapnya dapat dilihat pada rangkaian gambar berikut ini.



Kemudian, berpindahlah ke mode Source dan carilah tag untuk komponen *DataSourceInsertPass*. Setelah itu, modifikasilah bagian dari tag tersebut seperti pada listing berikut ini.

```
<asp:SqlDataSource ID="DataSourceInsertPass"
    runat="server"
    ConnectionString=
    "<%$
ConnectionStrings:AddressBookConnectionString1 %>"
    DeleteCommand="DELETE FROM [Password] "
    InsertCommand=
    "INSERT INTO [Password] ([password]) VALUES
(@password)"
    ProviderName=
    "<%$
ConnectionStrings:AddressBookConnectionString1.Pro
viderName %>"
```

```
    SelectCommand=
   "SELECT [password] FROM [Password]">
    <InsertParameters>
        <asp:ControlParameter ControlID="TextBox3"
        Name="password" Type="String" />
    </InsertParameters>
</asp:SqlDataSource>
```

Fokus dari modifikasi komponen *DataSourceInsertPass* terletak pada perintah SQL *Delete* dan *Insert*.
Pada perintah SQL untuk penghapusan, parameter di perintah SQL Delete dibuang agar pada saat eksekusi seluruh isi tabel langsung dihapus.
Sedangkan untuk perintah SQL *Insert*, ditambahkan definisi parameter agar parameter diambil dari *Textbox3* secara langsung.

Selanjutnya, dobel klik pada Button *Ganti Pass* lalu ketikkan listing berikut ini.

```
If cekData(DataSourcePassLama) Then
    Label2.Text = ""
    With DataSourceInsertPass
        .Delete()
        .Insert()
    End With
    TextBox2.Text = ""
```

```
    TextBox3.Text = ""
    Label2.Text = "Password telah diganti !"
Else
    Label2.Text = "Password lama salah !"
End If
```

Pada proses penggantian password, yang dilakukan pertama kali adalah melakukan pengecekan terhadap entrian password lama. Pengecekan ini mirip seperti pengecekan yang dilakukan pada proses login, sehingga tetap menggunakan fungsi yang sama yaitu fungsi *cekData*.
Setelah proses pengecekan selesai, dilakukan penghapusan terhadap seluruh password, dan password yang baru diinputkan ulang ke dalam tabel yang sama. Hal ini dilakukan karena dalam tabel *Password* hanya memiliki satu record dan satu field, dan juga field yang akan diedit merupakan field primary key.

Langkah terakhir dari proses ganti password adalah melakukan dobel klik pada LinkButton *Logout*, dan mengetikkan listing ini.

```
MultiView1.ActiveViewIndex = 0
FormView1.Visible = False
Session("Login_OK") = False
Label2.Text = ""
UpdatePanel2.DataBind()
UpdatePanel3.Update()
```

Pada proses logout, yang dilakukan pertama kali adalah mengembalikan keadaan MultiView ke View yang pertama, begitu pula dengan keadaan Formview agar tidak terlihat lagi. Dan juga dilakukan pengisian ulang nilai Session, agar tampilan di kolom sebelah kanan dapat terpicu untuk melakukan refresh ke keadaan awal sebelum login.
Selanjutnya dilakukan proses yang sama dengan proses login, yaitu "memaksa" kedua Update Panel untuk melakukan proses postback.

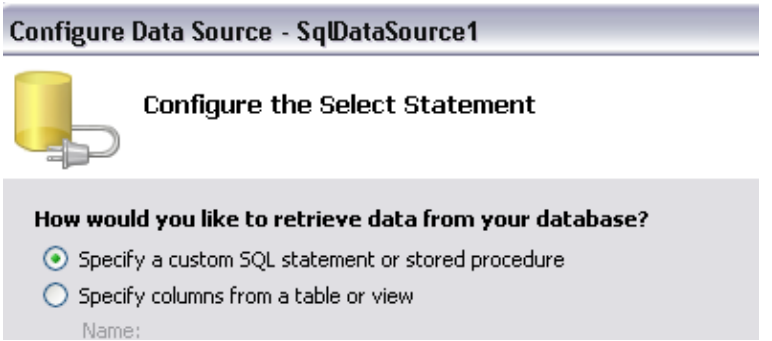
Tampilan Grid

Proses berikutnya adalah proses untuk memodifikasi tampilan Grid sekaligus memodifikasi tampilan Repeater. Dari hasil proses ini nantinya akan terlihat lebih jelas penggunaan teknik AJAX dalam sebuah halaman web, terutama yang menggunakan rancangan jenis *single page solution*.

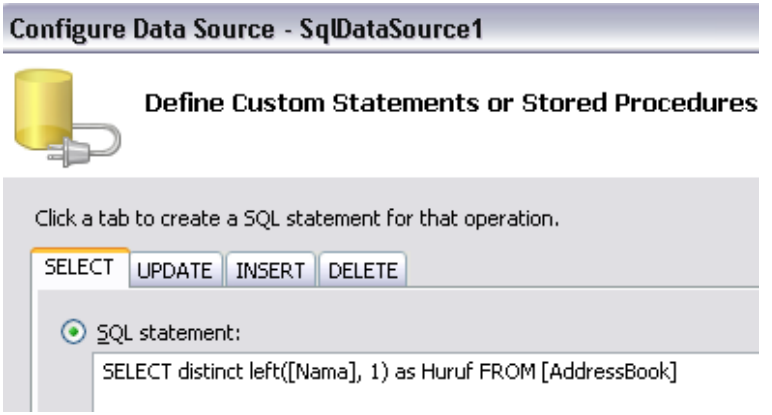
Proses modifikasi tampilan grid sesungguhnya merupakan proses modifikasi seluruh komponen yang terletak dalam komponen UpdatePanel2. Seluruh komponen yang terletak dalam UpdatePanel2 dianggap sebagai sebuah proses postback yang sama. Hal ini bisa dianggap sebagai acuan dalam menempatkan kumpulan komponen dalam sebuah Update Panel, terutama untuk kasus yang lainnya.

Langkah pertama yang dilakukan dalam proses ini adalah melakukan modifikasi pada dua komponen SqlDataSource di dalam UpdatePanel2. Untuk komponen *SqlDataSource1*, klik pada Smart Tag dan pilih sub menu *Configure Data Source*.

Selanjutnya, pada kotak dialog yang tersedia, pilih opsi *Specify a custom SQL Statement* untuk mengetikkan secara manual perintah SQL tanpa melalui kotak dialog wizard.



Kemudian, ketikkan perintah SQL untuk menampilkan huruf depan dari tiap data buku alamat di kotak dialog berikutnya.



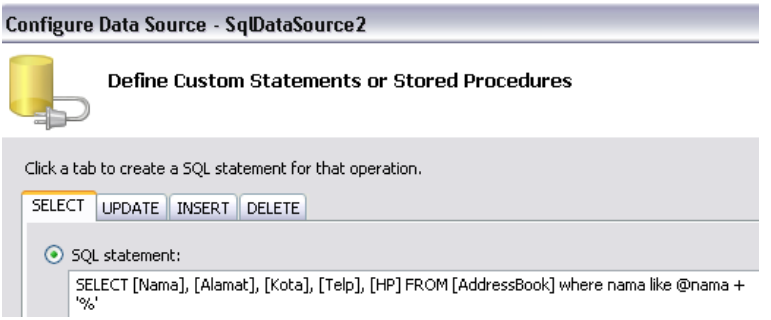
Pemberian parameter *distinct* pada perintah SQL *Select* akan menyebabkan hasil query yang *unique*. Sehingga jika terjadi dua

atau lebih huruf yang sama, hanya akan ditampilkan sekali.
Sedangkan pemberian fungsi *left* digunakan untuk mengambil huruf pertama pada field *Nama*, sehingga nantinya akan terbentuk semacam index karakter dari hasil query ini. Field yang diberi fungsi, sebisa mungkin diberi nama alias (dalam perintah query tersebut diberi nama alias *Huruf*) agar nantinya lebih mudah diakses di halaman web.

Setelah selesai memodifikasi komponen *SqlDataSource1*, kini klik pada Smart Tag di komponen *SqlDataSource2* dan pilih lagi sub menu *Configure Data Source*.

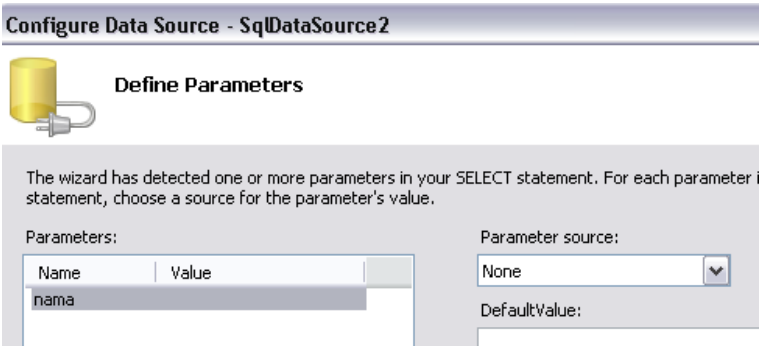

Komponen *SqlDataSource1* akan digunakan untuk mengisi data pada Repeater, sedangkan komponen *SqlDataSource2* akan digunakan untuk GridView.

Selanjutnya di kotak dialog yang tersedia, lakukan langkah yang sama seperti pada komponen *SqlDataSource1*, dan untuk perintah SQL yang ada, ketikkan seperti pada gambar berikut.



Penggunaan parameter *like* untuk menampilkan seluruh data yang diawali dengan karakter tertentu. Karakter tersebut, nantinya akan diambil dari hasil klik di dalam Repeater, dan akan dibahas pada bagian selanjutnya.

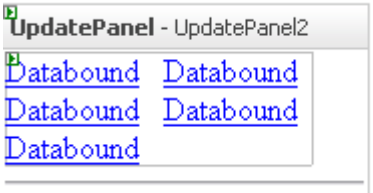
Lalu, pada kotak dialog berikutnya kosongkan parameter *nama* dengan memilih opsi *none* di dalam pilihan parameter yang tersedia. Hal ini dilakukan karena parameter nama akan diisikan melalui listing program.



Langkah berikutnya adalah memodifikasi tampilan Repeater. Untuk memodifikasi Repeater, berpindahlah ke dalam mode Source dan ketikkan listing berikut ini.

```
<asp:Repeater ID="Repeater1" runat="server"
    DataSourceID="SqlDataSource1">
<ItemTemplate>
    <asp:LinkButton ID="LinkButton1"
    runat="server" OnClick="refreshGrid"
    Text='<%#ucase(Eval("Huruf"))%>' >
    </asp:LinkButton>  
</ItemTemplate>
</asp:Repeater>
<hr />
```

Hasil dari modifikasi Repeater, akan terlihat seperti pada gambar berikut di dalam mode Design.



Pada modifikasi Repeater tersebut, ditempatkan sebuah *ItemTemplate* baru yang didalamnya berisi sebuah komponen LinkButton.
Dalam komponen LinkButton tersebut nantinya akan mengeksekusi sebuah prosedur bernama *RefreshGrid* serta menampilkan hasil query dari komponen *SqlDataSource1* yang berupa sebuah huruf unique dari tiap data di dalam tabel *Address*.

Masih tetap di dalam mode Source, kini carilah tag untuk GridView2, dan modifikasilah seperti pada listing berikut ini.

```
<asp:GridView ID="GridView2" runat="server"
    AllowPaging="True" AutoGenerateColumns="False"
    BorderStyle="None" BorderWidth="0px"
    DataKeyNames="Nama,HP"
    PageSize="3"
    DataSourceID="SqlDataSource2"
    ShowHeader="False" Width="100%"
    ShowFooter="True">
    <Columns>
        <asp:TemplateField HeaderText="Nama"
        SortExpression="Nama">
            <ItemTemplate>
               <table>
                    <tr><td>Nama</td>
                    <td><div style="color:Fuchsia;
```

```
    font-weight:bold ">
<%#Eval("Nama")%></div>
</td></tr>
<tr><td>Alamat</td>
<td>
    <div style="color:Fuchsia;
    font-weight:bold ">
<%#Eval("Alamat")%></div>
 </td></tr>
<tr><td>Kota</td>
<td>
 <div style="color:Fuchsia;
    font-weight:bold ">
<%#Eval("Kota")%></div>
</td></tr>
<tr><td>Telp</td>
<td><div style="color:Fuchsia;
    font-weight:bold ">
<%#Eval("Telp")%></div>
 </td></tr>
<tr><td>No. HP</td>
<td><div style="color:Fuchsia;
    font-weight:bold ">
<%#Eval("HP")%></div>
 </td></tr>
 <tr>
<td colspan="2">
<asp:LinkButton
    ID="LinkButton4"
    runat="server"
```

```
                    CommandArgument=
                <%#eval("Nama") & "," & eval("HP") %>
                    Visible="<%#visibleLink() %>"
                   OnClick="DeleteMe">
                   Delete</asp:LinkButton>
                   </td>
                   </tr>
               </table>
                <hr style="color:Black;
                      height:2px" />
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

Hal pertama yang dimodifikasi dalam GridView tersebut adalah melakukan pembuangan terhadap seluruh field dan menggantinya dengan sebuah field bertipe *template field*. Dengan mengganti menjadi jenis *template*, maka isi dari GridView dapat lebih bebas untuk dimodifikasi.

Selanjutnya, untuk pengisian field dengan model *columnar* (turun ke bawah), maka dibuat sebuah tabel dengan jumlah kolom sebanyak dua dan jumlah baris sebanyak enam.

Lalu dibuat sebuah LinkButton baru untuk proses penghapusan data yang hanya akan dapat tampil saat proses login selesai dilakukan. LinkButton *Delete* ini juga akan

mempertegas penggunaan teknik AJAX di dalam GridView dengan memanfaatkan penggunaan Session.
Dan jika diperhatikan lebih seksama, banyak hal yang bisa dipersingkat dalam proses pengetikan listing tersebut. Yaitu pada proses pembuatan baris-baris yang ada, dapat dilakukan dengan menggunakan teknik *copy paste* karena banyak hal yang sama didalamnya.
Selain itu, perhatikan pula property *CommandArgument* yang terletak pada LinkButton *Delete* yang mengambil dua field dari tabel *Address*. Isi dari property ini nantinya akan dimanfaatkan oleh proses penghapusan di bagian berikutnya.

Kini, berpindahlah ke mode pengetikan listing program untuk kemudian melakukan pengetikan listing berikut.

```
Public Function visibleLink() _
    As Boolean
    Return CBool _
    (Session("Login_OK"))
End Function

Public Sub DeleteMe _
    (ByVal sender As Object, _
    ByVal e As EventArgs)
    Dim xParam As String() = _
    Split(sender.commandargument, ",")
    With SqlDataSource3
```

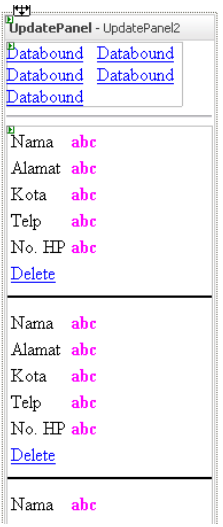
```
            .DeleteParameters(0). _
                DefaultValue = xParam(0)
            .DeleteParameters(1). _
                DefaultValue = xParam(1)
            .Delete()
            UpdatePanel2.DataBind()
        End With
    End Sub

Public Sub refreshGrid _
       (ByVal sender As Object, _
       ByVal e As EventArgs)
       SqlDataSource2. _
           SelectParameters(0).DefaultValue = _
           sender.text
       SqlDataSource2.DataBind()
    End Sub
```

Fungsi yang pertama yaitu *visibleLink* digunakan untuk melakukan proses *toggle* pada LinkButton yang terletak pada GridView. Sehingga jika proses login berhasil dilakukan, maka property *visible* dari LinkButton *Delete* akan mengikuti isi dari Session.
Sedangkan prosedur *DeleteMe* digunakan untuk melakukan proses penghapusan dari data yang ada di dalam tabel *Address*. Karena primary key dari tabel *Address* terdiri dari dua field, maka parameter penghapusan diambil dari dua field pula. Pengambilan parameter ini

menggunakan isi property *CommandArgument* yang telah ada sebelumnya (lihat lagi di catatan sebelumnya). Dan karena terdiri dari dua field, maka kedua field tersebut harus dipecah dengan menggunakan fungsi *Split*. Kemudian dari hasil split tersebut, masing-masing dimasukkan ke parameter dari perintah SQL Delete dari komponen *SqlDataSource2*.
Untuk prosedur terakhir, digunakan untuk melakukan refresh terhadap GridView jika sebuah proses manipulasi selesai dilakukan, baik pada saat proses penghapusan data maupun pada proses input data baru yang akan diterangkan pada sub bab berikutnya.

Hasil dari modifikasi di UpdatePanel2 akan terlihat seperti pada gambar berikut.



Sedangkan, pada saat proses eksekusi halaman web akan tampak seperti pada gambar selanjutnya.

..:: Simple Address Book ::..

Password

Login

A B C

Nama Ali
Alamat Jl. Kota 1
Kota Surabaya
Telp 18273662
No. HP 080000212

Dan pada saat proses login selesai dilakukan, maka akan tampak tampilan berikut.

..:: Simple Address Book ::..

..:: Mode Admin::..

Pass lama

Pass baru

Ganti Pass

Logout

A B C

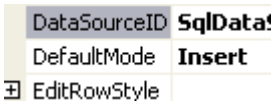
Nama Ali
Alamat Jl. Kota 1
Kota Surabaya
Telp 18273662
No. HP 080000212
Delete

Perhatikan peralihan dari saat proses login selesai dilakukan, maka tidak akan terjadi efek *flicker* atau berkedip yang menandakan bahwa teknik AJAX berhasil diimplementasikan.

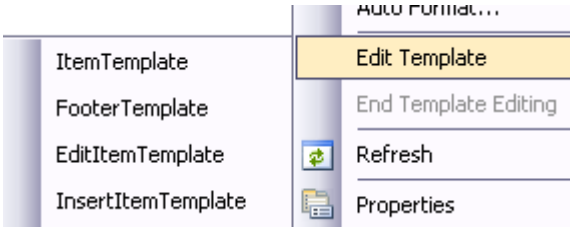
Input Data Baru

Langkah terakhir dari studi kasus address book ini adalah melakukan implementasi proses input data baru. Proses ini merupakan modifikasi pada komponen FormView yang terletak pada UpdatePanel3.

Yang pertama kali harus dilakukan adalah memodifikasi tampilan dari FormView itu sendiri. Set property *DefaultMode* dari FormView menjadi *Insert* agar pada saat muncul pertama kali akan langsung menuju ke proses input data.

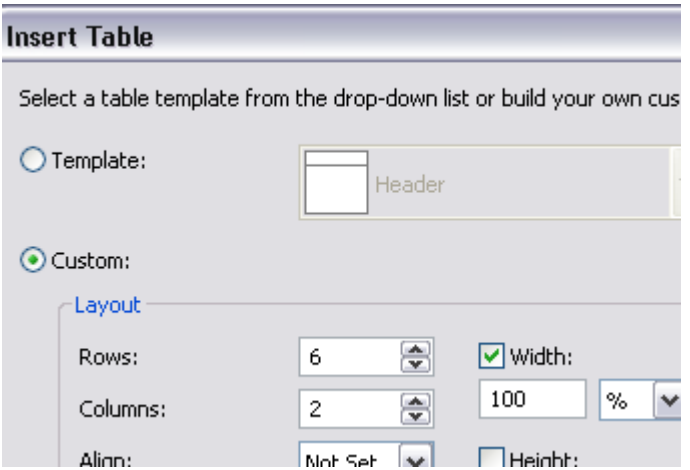


Kemudian, modifikasi tampilan proses inputan tersebut, dengan melakukan klik kanan pada FormView dan memilih sub menu *Edit Template → Insert Item Template*



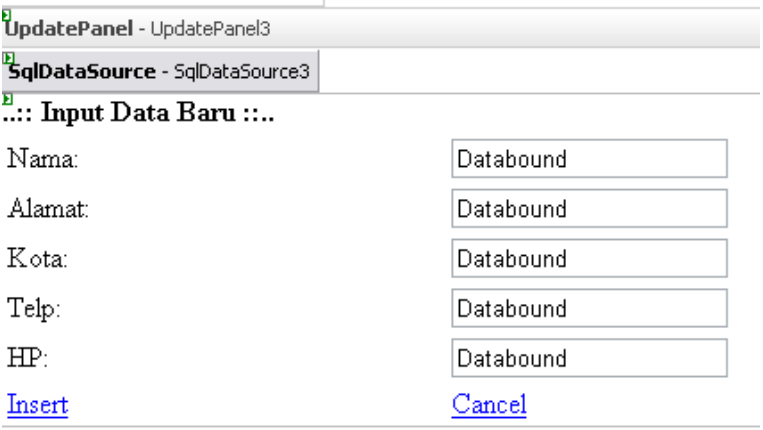
Pada tampilan mode edit *Insert Item Template* tersebut, sisipkan sebuah tabel dengan memilih menu *Layout → Insert Table*. Di dalam kotak dialog yang

tersedia, ketikkan jumlah baris sebanyak 6 dan jumlah kolom sebanyak 2.

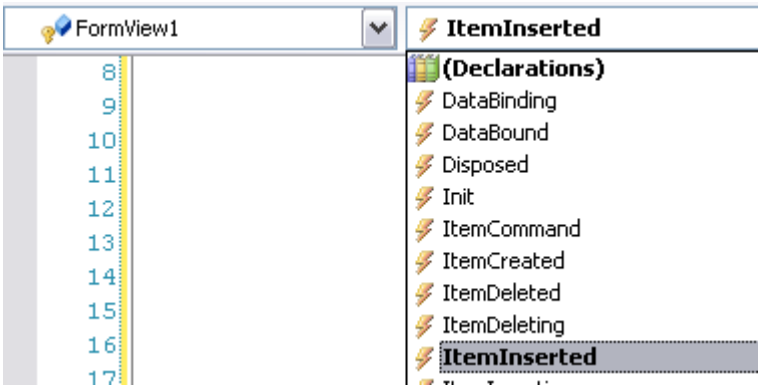


Perhitungan jumlah baris disesuaikan dengan jumlah field yang tersedia.

Kemudian (dengan hati-hati) pindahkan tiap teks dan Textbox pada tampilan ke dalam tabel yang baru saja dibuat, sehingga tampilan akan menjadi seperti pada gambar berikut.



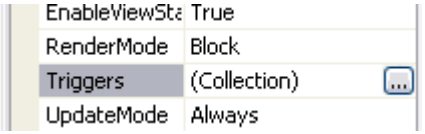
Langkah berikutnya adalah melakukan dobel klik pada FormView tersebut, kemudian di dalam prosedur *FormView1_PageIndex Changing* pindahkan ke prosedur *FormView1_ItemInserted* dengan mengganti nama method dari FormView ke *ItemInserted*.



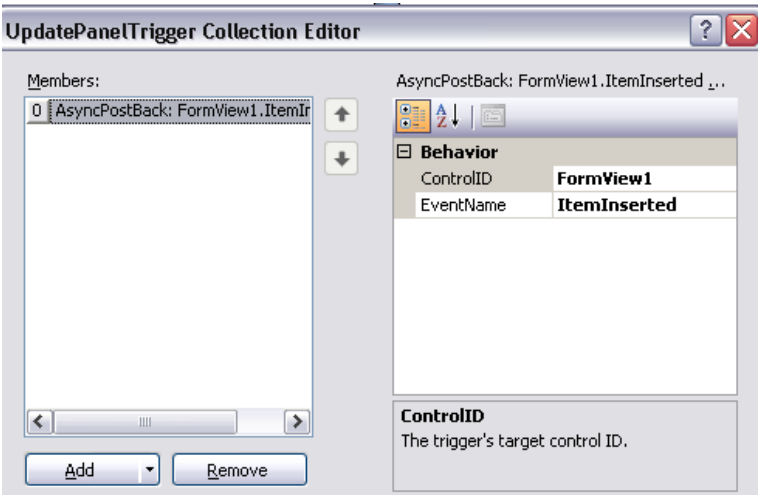
Kemudian di dalam prosedur *FormView1_ItemInserted* ketikkan listing berikut :

```
UpdatePanel2.DataBind()
```

Kini, langkah terakhir dari pembuatan studi kasus ini adalah melakukan setting pada property *Triggers* di dalam UpdatePanel2. Klik pada UpdatePanel2, dan pada property *Triggers,* klik tombol kecil di sebelahnya untuk menuju ke kotak dialog berikutnya.



Pada kotak dialog yang tersedia, tambahkan trigger baru dengan menekan tombol *Add.* Kemudian pada sub property *ControlID* isikan dengan komponen *FormView1*, serta pada sub property *EventName* isikan dengan *ItemInserted*.



Pengisian property *Trigger* dilakukan agar pada saat terjadi proses entri data baru, maka Gridview maupun Repeater yang berada di dalam UpdatePanel2 akan secara otomatis melakukan proses refresh.
Hal ini dilakukan karena proses entri data baru di dalam FormView tidak akan mengakibatkan proses refresh otomatis, karena dilakukan di dalam komponen Update Panel. Sehingga hanya akan terjadi proses partial postback.

Kemudian cobalah untuk melakukan eksekusi pada situs yang telah dibuat. Cobalah beberapa kali untuk melakukan entri data buku alamat baru maupun melakukan proses penggantian password. Jika ingin melihat lebih jelas hasil dari implementasi AJAX, maka disarankan untuk menempatkan sebuah gambar (di sembarang tempat dalam halaman web) agar terlihat apakah efek flicker masih terjadi atau tidak.

Hasil dari eksekusi studi kasus yang telah dikerjakan akan tampak pada rangkaian gambar berikut ini.

..:: Simple Address Book ::..

Password

Login

A B C

Nama Ali
Alamat Jl. Kota 1
Kota Surabaya
Telp 18273662
No. HP 080000212

..:: Simple Address Book ::..

..:: Mode Admin::..

Pass lama

Pass baru

Ganti Pass

Logout

A B C

Nama Ali
Alamat Jl. Kota 1
Kota Surabaya
Telp 18273662
No. HP 080000212
Delete

..:: Input Data Baru ::..

Nama:
Alamat:
Kota:
Telp:
HP:

Insert Cancel

AJAX Control Toolkit

Pengantar

AJAX Control Toolkit merupakan sekumpulan komponen gratis (dan open source) yang dikeluarkan secara resmi oleh Microsoft untuk membantu para programmer web dalam melakukan pemrograman berbasis web dengan menggunakan teknik AJAX.

Kumpulan komponen ini dapat secara gratis didownload di situs ajax.asp.net. Dan yang perlu diperhatikan bahwa kumpulan komponen ini akan terus berkembang, karena selalu didonasi oleh banyak programmer yang secara sengaja membuat komponen-komponen baru di dalam AJAX Control Toolkit.

Salah satu keunggulan utama dari AJAX Control Toolkit adalah kemudahan penggunaan serta adanya dokumentasi dan contoh yang dapat dilihat di situs ajax.asp.net. Akibatnya, penggunaan AJAX Control Toolkit jauh lebih mudah dilakukan, dibandingkan dengan komponen-komponen lain (baik yang gratis maupun komersil) untuk AJAX.

Dalam bab ini, akan dibahas beberapa contoh kecil mengenai penggunaan AJAX Control Toolkit, yang tentu saja contoh-contoh yang akan ditampilkan berbeda dengan contoh yang telah ada di dokumentasi resmi AJAX Control Toolkit. Beberapa perbedaan yang sangat mendasar dalam

implementasi contoh di buku ini adalah pemakaian database hampir diseluruh contoh mengenai AJAX Control Toolkit.

Hal tersebut memang sengaja dilakukan, pasalnya di contoh yang resmi mengenai AJAX Control Toolkit kebanyakan hanya membahas mengenai penggunaan komponen dengan menggunakan variabel, sehingga banyak programmer (khususnya pemula) yang merasa kesulitan untuk mengimplementasikan AJAX Control Toolkit dengan menggunakan database.

Selain contoh-contoh kecil, di akhir bab ini juga terdapat studi kasus yang sedikit kompleks mengenai penggunaan AJAX Control Toolkit, yaitu mengenai pembuatan mini blog yang sepenuhnya merupakan contoh penggunaan AJAX Control Toolkit secara utuh dalam sebuah situs.

Selamat mencoba !!!

Contoh 1 : Calendar

Contoh pertama dari penggunaan AJAX Control Toolkit dalam buku ini adalah komponen Calendar Extender. Penggunaan komponen Calendar sesungguhnya mirip dengan komponen DateTimePicker di pemrograman berbasis desktop. Sayangnya, di dalam ASP .NET 2.0, secara default, komponen tersebut tidak tersedia. Yang tersedia hanyalah komponen Calendar biasa yang selalu menimbulkan proses postback pada saat terjadi pemilihan tanggal tertentu didalamnya.

Tetapi dengan menggunakan komponen Calendar dari AJAX Control Toolkit, maka proses postback tidak lagi terjadi. Meski komponen yang sejenis banyak tersedia di dunia maya, tetapi kebanyakan berupa script dari Javascript yang seringkali membuat para programmer berbasis web sedikit kesulitan dalam melakukan implementasi.

Dalam contoh kasus ini, akan dibuat sebuah form isian sederhana untuk mengentrikan data dalam sebuah tabel *HariLibur* yang didalamnya akan mendefinisikan hari libur nasional dalam satu tahun.

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Web Site
2. Buat folder *bin* dalam solution

Pembuatan folder *bin* dilakukan dengan memilih menu *Add ASP .NET Folder → bin* saat klik kanan di solution explorer.

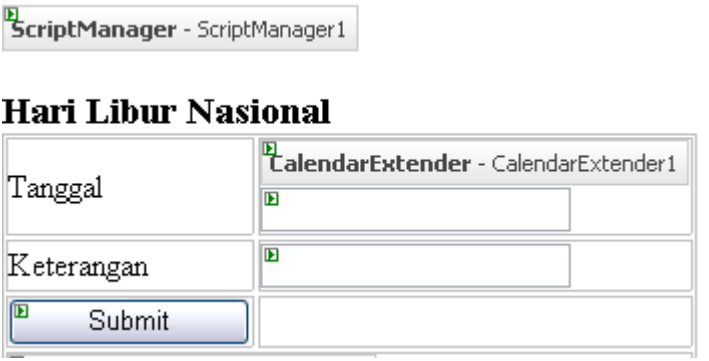
3. Tambahkan file dll dari AJAX Control Toolkit ke dalam folder *bin*

File *AJAXControlToolkit.dll* dapat diambil dari folder *SampleWebSite/Bin* setelah AJAX Control Toolkit diekstrasi dari paket hasil download. Penambahan tersebut dapat melalui menu *Project → Add References* atau dengan melakukan klik kanan di folder bin dan pilih menu *Add Existing Item* kemudian arahkan ke file *AJAXControlToolkit.dll*.
Secara otomatis akan diikutsertakan pula file-file resources yang mendefinisikan bahasa yang didukung oleh AJAX Control Toolkit, serta di dalam Toolbox akan ditambahkan satu tab baru yang berisikan komponen-komponen dari AJAX Control Toolkit.

4. Buat sebuah database dengan nama *Calendar*, dan didalamnya buat sebuah tabel bernama *HariLibur* dengan struktur sebagai berikut :

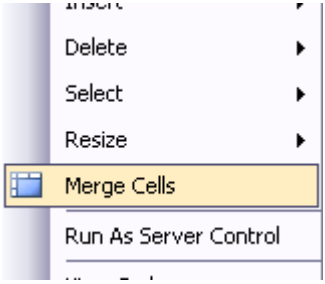
	Column Name	Data Type	Allow Nulls
🔑	TglLibur	datetime	☐
	Keterangan	varchar(50)	☑

5. Selanjutnya, di dalam halaman web yang tersedia buat sebuah tabel dengan rincian empat baris dan dua kolom melalui menu *Layout → Insert Table*.
6. Di kolom pertama, untuk baris pertama ketikkan *Tanggal* dan pada baris kedua ketikkan *Keterangan*.
7. Tempatkan sebuah Button di baris ketiga dan ganti teks dalam Button tersebut menjadi *Simpan*.
8. Berikutnya, drag dua buah Textbox masing-masing untuk kolom kedua di baris pertama dan kedua.
9. Untuk Textbox pertama, drag komponen *Calendar* dari tab AJAX Control Toolkit tepat disebelahnya.

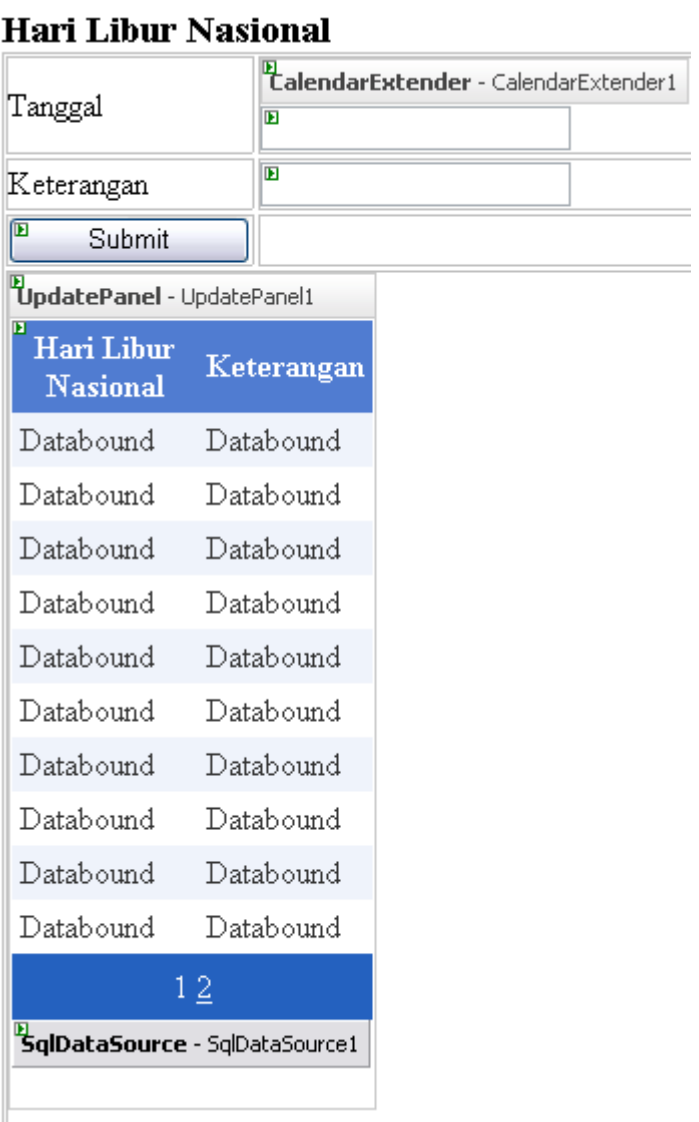


Penempatan komponen Calendar dari AJAX Control Toolkit membutuhkan sebuah komponen lain yang akan menjadi penampung dari hasil pemilihan komponen Calendar. Umumnya komponen penampung tersebut adalah komponen yang bersifat sebagai penerima inputan seperti Textbox.

10. Kini, di baris keempat lakukan *Merge Cell* dengan cara memilih kedua kolom di baris yang sama lalu klik kanan dan pilih menu *Merge Cell*.



11. Drag sebuah komponen UpdatePanel di dalam baris yang telah digabungkan tersebut, lalu didalamnya drag tabel *HariLibur* .

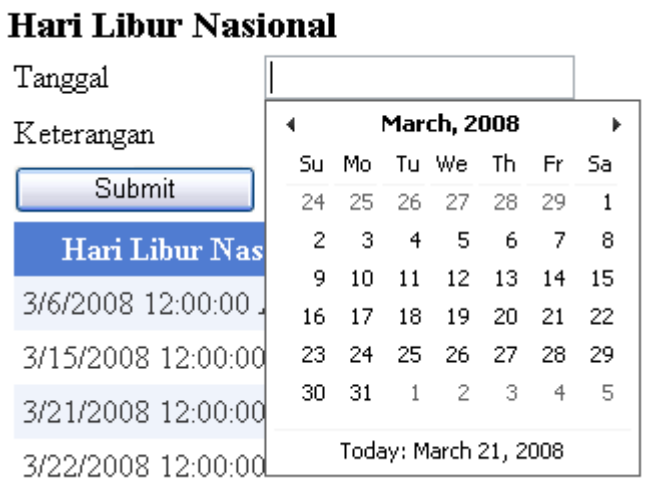


12. Klik di komponen Calendar, dan set property *TargetControlID* ke *Textbox1*

13. Selanjutnya dobel klik di Button *Simpan* dan ketikkan listing berikut :

```
With SqlDataSource1
    .InsertParameters(0). _
        DefaultValue = TextBox1.Text
    .InsertParameters(1). _
        DefaultValue = TextBox2.Text
    .Insert()
End With
```

Pengambilan parameter untuk isian tabel *HariLibur* berdasarkan dari isian di dalam Textbox. Hal ini juga berlaku untuk isian Textbox yang berasal dari pemilihan tanggal di komponen Calendar.
Terdapat kelemahan dalam penggunaan Calendar dengan cara seperti ini, yaitu isi Textbox masih bisa diganti dengan format lain selain format tanggal. Hal ini tentu saja sangat berbahaya pada saat terjadi pengisian data. Untuk mengatasi hal tersebut, property *ReadOnly* pada Textbox dapat diganti dengan menggunakan setting nilai *True* agar isi dari Textbox hanya dapat diisi dari hasil pemilihan tanggal di komponen Calendar.



Perhatikan bahwa di tiap halaman web yang ditempati oleh AJAX Control Tookit selalu membutuhkan komponen *ScriptManager* didalamnya.

Contoh 2 : Pop Up Control

Di contoh yang kedua dibahas mengenai komponen Pop Up dari AJAX Control Tookit. Komponen ini sangat sering digunakan dalam sebuah situs, mengingat fleksibilitas dari komponen ini yang mampu menampung berbagai jenis komponen lain sebagai target pop up yang akan muncul saat sebuah komponen diklik.

Konsep dari komponen Pop Up sesungguhnya sebagai sebuah jembatan dari dua buah komponen yang akan ditautkan prosesnya. Sebagai contoh, jika saat sebuah Textbox diklik oleh pengguna, maka dapat muncul pilihan dari sebuah RadioButtonList atau dari sebuah ListBox. Atau pada saat sebuah Textbox diklik, diharapkan akan muncul sebuah panel yang berisi keterangan mengenai isian dari Textbox tersebut.

Dalam contoh ini akan diterapkan Pop Up dalam sebuah halaman web untuk kepentingan login. Pop Up Control akan dijadikan sebagai jembatan untuk sebuah RadioButtonList yang berisikan UserID agar pengunjung yang akan login tidak salah memasukkan UserID saat proses login dilakukan.

1. Buat sebuah website baru dengan tipe ASP .NET AJAX Website

2. Pastikan bahwa file runtime dari AJAX Control Toolkit telah ada di folder bin (lihat lagi contoh sebelumnya untuk cara menambahkan).
3. Buat sebuah database dengan nama *UserProfile* yang didalamnya memiliki sebuah tabel dengan nama *UserProfile* juga dengan struktur sebagai berikut :

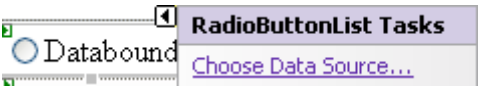
Column Name	Data Type	Allow Nulls
UserID	nchar(10)	☐
UserName	varchar(50)	☑
UserPassword	nchar(10)	☑
UserLevel	nchar(1)	☑

4. Setelah itu, isikan contoh data sebagai berikut (contoh data dapat diisi sesuai keinginan) :

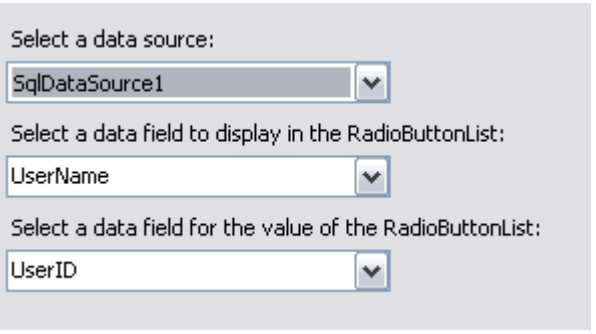
UserID	UserName	UserPassw...	UserLevel
admin	Administrator	admin	A
user1	User 1	user1	A
user2	User 2	user2	B

5. Kemudian di halaman web yang tersedia, buat sebuah tabel dari menu *Layout → Insert Table* dengan ketentuan empat baris dan dua kolom.
6. Tempatkan teks *User ID* dan *Password* masing-masing di baris kedua dan ketiga di kolom pertama.
7. Berikutnya di kolom kedua, tempatkan dua buah UpdatePanel, masing-masing di baris kedua dan keempat.

8. Kemudian drag tabel *UserProfile* sebanyak dua kali di tiap UpdatePanel yang telah tersedia. Lalu hapus tiap GridView yang terbentuk didalamnya.
9. Untuk UpdatePanel yang pertama, letakkan sebuah Textbox dan sebuah RadioButtonList. Lalu klik pada *Smart Tag* ( ) pada RadioButtonList dan pilih menu *Choose Data Source*.



10. Pada kotak dialog yang tersedia, arahkan property *Display Member* dan *Value Member* seperti pada gambar berikut :



11. Masih di UpdatePanel yang pertama, drag sebuah komponen PopUpControl Extender didalamnya. Kini, berpindahlah ke mode *Source* dan modifikasilah PopUp Control menjadi seperti listing berikut :

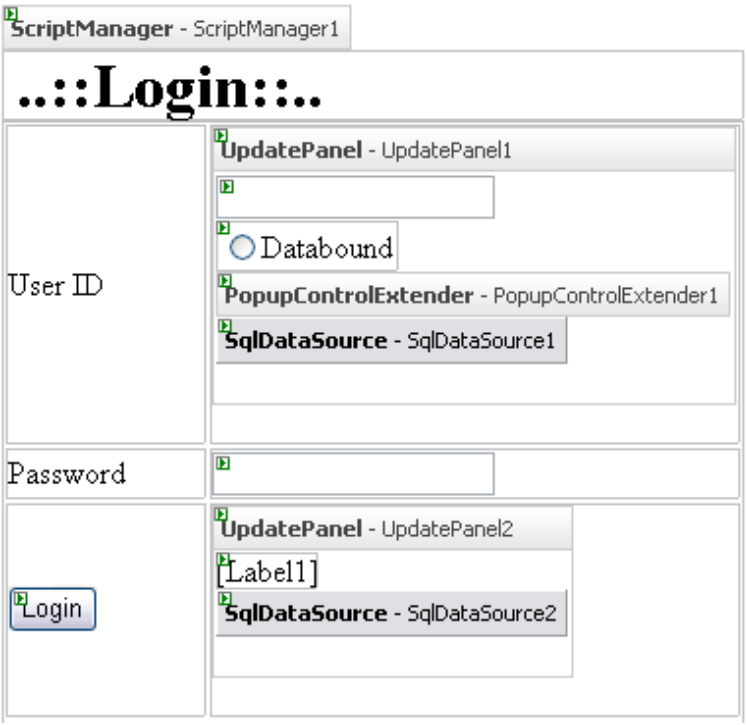
```
<cc1:PopupControlExtender
    ID="PopupControlExtender1"
```

```
        runat="server"
        TargetControlID="TextBox1"
        PopupControlID="RadioButtonList1"
        Position="Right"  >
    </cc1:PopupControlExtender>
```

Modifikasi pada komponen Pop Up terletak pada pengisian property *PopUpControlID* yang merupakan komponen yang akan dijadikan sebagai komponen Pop Up (yaitu RadioButtonList) dan pengisian property *TargetControlID* yang nantinya merupakan komponen pemicu dari munculnya komponen yang dijadikan sebagai komponen Pop Up.
Sehingga pada saat halaman web dijalankan, maka pada saat Textbox diklik (atau dalam keadaan focus), komponen RadioButtonList secara otomatis muncul di sebelah kanan Textbox.

12. Letakkan sebuah Textbox untuk pengisian password di baris ketiga kolom kedua, dan set property *TextMode* menjadi *password*.
13. Untuk UpdatePanel yang kedua, drag sebuah Label didalamnya dan set property *Text* dari Label tersebut menjadi kosong. Dan set property *UpdateMode* dari UpdatePanel menjadi *Conditional*.

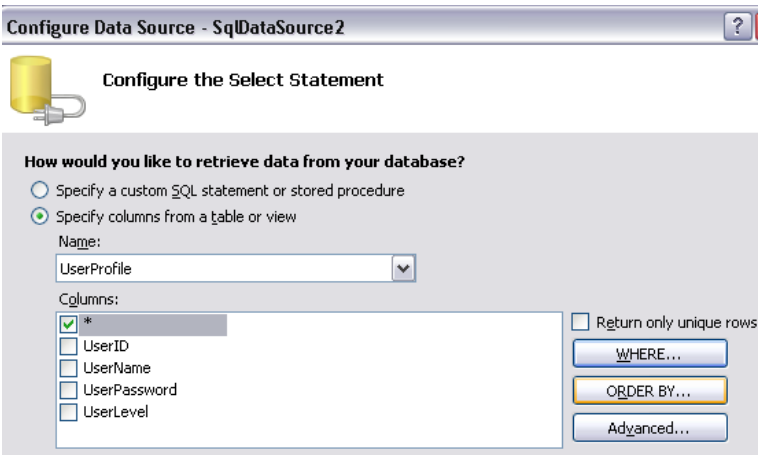
UpdatePanel kedua hanya akan diupdate isinya setelah penekanan tombol. Karenanya property *UpdateMode* diset menjadi *Conditional* agar pada saat Pop Up muncul tidak terjadi efek flickr.



14. Lalu set property *DataSourceMode* di komponen SqlDataSource2 menjadi *DataReader*. Hal ini dilakukan

untuk kepentingan proses pengecekan data pada saat login.

15. Klik pada Smart Tag di SqlDataSource2, dan pilih menu *Configure Data Source*. Kemudian pada kotak dialog yang tersedia, klik tombol *Where* untuk menentukan kondisi perintah *Select*



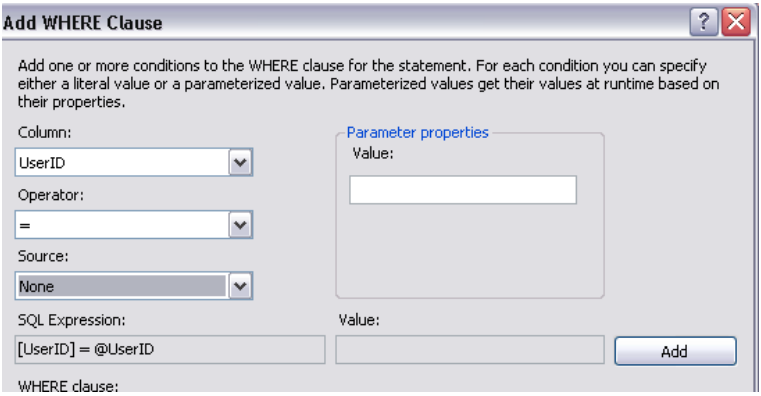
16. Kemudian, pilih field *UserID* sebagai parameter dan pilih sumber kondisi menjadi *None*, dan klik tombol *Add* untuk menambahkan kondisi tersebut. Lakukan langkah yang sama dengan memilih field *UserPassword* sebagai parameter. Setelah selesai, klik tombol *Next* hingga kotak dialog selesai dieksekusi.

Langkah tersebut juga dapat dilakukan dengan langsung mengganti perintah SQL di komponen SqlDataSource2, tepatnya di property *SelectCommand* dan mengetikkan perintah SQL berikut :

```
SELECT [UserID], [UserName], [UserPassword], [UserLevel] FROM [UserProfile] WHERE (([UserID] = @UserID) AND ([UserPassword] = @UserPassword))
```

Kemudian memodifikasi bagian tag *SelectParameters* menjadi sebagai berikut :

```
<SelectParameters>
    <asp:Parameter Name="UserID" Type="String" />
    <asp:Parameter Name="UserPassword" Type="String" />
</SelectParameters>
```



17. Berikutnya, set property *AutoPostBack* pada RadioButtonList menjadi *True*, kemudian dobel klik

pada RadioButtonList tersebut dan ketikkan listing berikut :

```
With Me.RadioButtonList1
    If .SelectedIndex >= 0 Then
        Me.PopupControlExtender1. _
            Commit(.SelectedValue)
    Else
        Me.PopupControlExtender1. _
        Cancel()
    End If
End With
```

Pada saat RadioButtonList1 diklik, maka komponen Pop Up akan melakukan proses Commit yang berarti mengumpanbalikkan nilai yang dipilih dari RadioButtonList sebagai komponen Pop Up ke dalam komponen Textbox sebagai komponen target dari Pop Up Control tersebut, sekaligus pula memicu RadioButtonList dalam keadaan *hidden* secara otomatis. Sedangkan, jika tidak ada yang dipilih dari RadioButtonList tersebut (pengguna melakukan klik di bagian lain dari halaman web), maka RadioButton akan kembali ke kondisi semula (hidden) tanpa mengumpanbalikkan nilai apapun ke dalam Textbox.

18. Langkah terakhir adalah melakukan dobel klik pada Button *Login* dan ketikkan listing berikut ini :

```
With Me.SqlDataSource2
     .SelectParameters(0). _
         DefaultValue = TextBox1.Text
     .SelectParameters(1). _
         DefaultValue = TextBox2.Text
     Dim xreader As  _
    Data.IDataReader = _
      .Select(DataSourceSelectArguments.Empty)
     If xreader.Read Then
         label1.text = "Sukses !"
     Else
         label1.text = "Login gagal !"
     End If
End With
```

Pembacaan SqlDataSource dalam mode *DataReader* akan mempercepat proses, khususnya pada saat situs tersebut benar-benar berada di sebuah web hosting yang online. Sehingga, mode pembacaan hanya akan berada dalam mode *Read Only* sesuai dengan kebutuhan dalam sebuah proses login. Perhatikan pula cara penempatan parameter dalam sebuah komponen SqlDataSource yang langsung dapat diumpanbalikkan dalam mode runtime.

..::Login::..

User ID admin

Password

Login

Administrator

User 1

User 2

Pada saat Textbox1 diklik oleh pengguna, maka secara otomatis RadioButtonList akan memunculkan isi dari field *UserName*. Kemudian pada saat item di RadioButtonList selesai dipilih dan isi field *UserID* yang bersesuaian tampil di dalam Textbox, maka RadioButtonList secara otomatis akan hilang.

Contoh 3 : Modal Pop Up

Penggunaan AJAX Control Toolkit lainnya yang menarik adalah penggunaan komponen Modal Pop Up. Modal Pop Up mirip dengan komponen Pop Up, dengan perbedaan utama adalah cara menampilkan kotak Pop Up yang lebih dominan dan "menguasai" browser, sehingga perhatian dari pengunjung benar-benar tertuju ke dalam isi dari kotak Pop Up tersebut.

Pada implementasi teknik AJAX tanpa menggunakan AJAX Control Toolkit, cara menampilkan kotak Pop Up, baik secara biasa (seperti halnya Pop Up Control) atau secara modal (seperti Modal Pop Up) sangatlah beragam. Terdapat belasan dan bahkan (mungkin) puluhan komponen (baik yang gratis maupun komersial) yang menawarkan hal yang sama denan kelebihan dan kekurangan masing-masing.

Sebut saja seperti *Lightbox* yang mampu menampilkan kotak Pop Up jenis modal baik untuk teks maupun galeri gambar, dengan menggunakan animasi yang begitu menarik. Begitu pula dengan komponen *Overlib* yang mampu menampilkan kotak Pop Up dengan berbagai jenis template kotak yang telah tersedia secara *built-in*. Dan masih sangat banyak komponen lain yang

bertebaran di dunia maya, khususnya untuk menampilkan kotak jenis Pop Up tersebut.

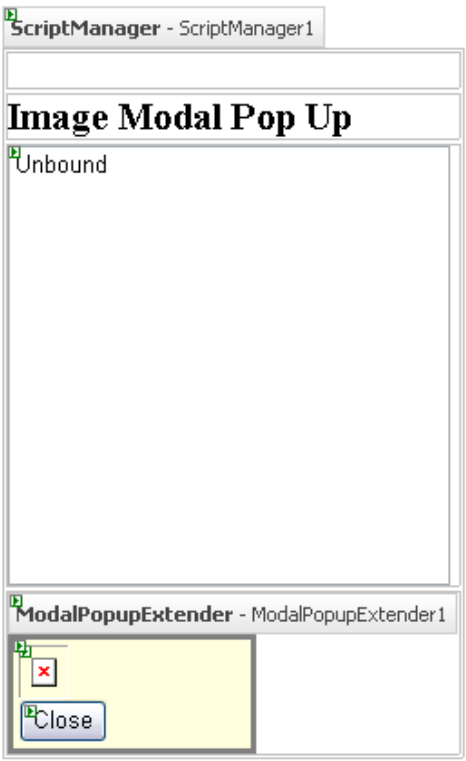
Tetapi secara umum, kebanyakan dari komponen Pop Up tersebut, seringkali sulit untuk diintegrasikan dengan *server side*, khususnya di dalam ASP .NET 2.0. Dan tentu saja hal ini sangat berbeda dengan penggunaan AJAX Control Toolkit yang memang menyatu dan khusus dirancang untuk kepentingan pemrograman ASP .NET 2.0.

Dalam contoh kasus ini akan diimplementasikan penggunaan Modal Pop Up pada sebuah halaman web yang didalamnya akan menampilkan sebuah gambar berdasarkan daftar file yang diisi dari sebuah folder.

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Website
2. Tempatkan beberapa file gambar dalam solution tersebut.

Usahakan untuk menempatkan lebih dari satu file gambar dalam folder default di dalam solution. Usahakan pula agar file-file gambar tersebut memiliki ekstensi yang sama (misal : jpg semua atau bmp semua) dan juga memiliki ukuran yang hampir sama (agar pada saat kotak Pop Up muncul, tidak terlalu ekstrem perbedaan ukurannya).

3. Pastikan bahwa komponen AJAX Control Toolkit telah tersedia di dalam Toolbox (lihat lagi contoh sebelumnya).
4. Kini, di halaman web yang tersedia, tempatkan sebuah Listbox, sebuah komponen ModalPopUpExtender dan sebuah Panel yang didalamnya terdiri dari satu komponen Image dan komponen Button dengan isi teks *Close*.



5. Set property *CSSClass* dalam panel menjadi *modalPopup*

6. Buat sebuah file CSS baru dengan memilih menu *Website → Add New Item* dan pilih tipe file *StyleSheet*, lalu beri nama sesuai selera.
7. Di dalam file CSS tersebut, ketikkan class CSS berikut ini :

```
.modalBackground {
    background-color:Gray;
    filter:alpha(opacity=50);
    opacity:0.9;

}

.modalPopup {
    background-color:#ffffdd;
    border-width:3px;
    border-style:solid;
    border-color:Gray;
    padding:3px;
    width:150px;
    margin:0;
    position:static;
    float:left ;
}
```

Class *modalBackground* digunakan sebagai style latar belakang dari halaman web yang "ditumpuki" oleh kotak Modal Pop Up, sedangkan class *modalPopup* digunakan untuk style dari komponen Panel yang sebenarnya merupakan kotak Pop Up itu sendiri.
Penggunaan style *filter* dan *opacity* merupakan trik agar halaman web yang "ditumpuki" oleh kotak Pop Up terlihat samar dan seakan-akan "tenggelam" oleh fokus yang diperlihatkan oleh kotak Pop Up.
Sedangkan style untuk kotak Pop Up sendiri usahakan agar memiliki kesan "tegas" untuk memperjelas fokus dan keberadaan kotak Pop Up.

Lalu berpindahlah ke mode *Source* dan modifikasilah tag dari tiap komponen sebagai berikut :

a. Di antara tag *head*, ketikkan perintah untuk menempatkan file CSS sebagai berikut :

```
<link href="StyleSheet.css"
    rel="stylesheet"
type="text/css" />
```

Asumsi nama file CSS dalam kasus ini adalah *StyleSheet.css*, nama file ini tentu saja bergantung pada saat pemberian nama di langkah pembuatan file CSS.

b. ModalPopUp

```
<cc1:ModalPopupExtender
ID="ModalPopupExtender1"
runat="server"
PopupControlID="Panel1"
TargetControlID="ListBox1"
DropShadow="true"
RepositionMode="None"
X="0" Y="0"
BackgroundCssClass="modalBackground">
</cc1:ModalPopupExtender>
```

Pengisian property *PopUpControlID* menyatakan komponen yang akan dijadikan kotak Pop Up itu sendiri, sedangkan property *TargetControlID* merupakan komponen yang akan menjadi pemicu munculnya kotak Pop Up.
Penggunaan property *X* dan property *Y* ditujukan untuk menentukan posisi dari kotak Pop Up itu sendiri di dalam browser. Dalam kasus ini, kotak Pop Up akan muncul di pojok kiri atas browser, karena koordinat X dan Y keduanya diset ke nilai nol. Jika kedua property tersebut tidak diisi oleh nilai tertentu, maka secara default, kotak Pop Up akan muncul tepat di tengah browser.

8. Kemudian dobel klik di Listbox dan ketikkan listing berikut :

```
Image1.ImageUrl = _
 Server.MapPath("~\") & _
 ListBox1.SelectedValue
```

Perhatikan bahwa property *AutoPostBack* dalam Listbox tidak perlu diberi nilai *true* agar Listbox dapat langsung bereaksi. Karena dengan adanya komponen ModalPopUp, Listbox akan "dipaksa" untuk bereaksi saat terjadi aksi tertentu.

9. Selanjutnya, dobel klik di bagian kosong dari halaman web, dan di dalam prosedur *Page_Load* ketikkan listing berikut :

```
     If Not IsPostBack Then
Dim di As New IO.DirectoryInfo _
          (Server.MapPath("~\"))
Dim fi As IO.FileInfo() = _
    di.GetFiles("*.jpg")
Dim xtemp As String
For i As Integer = 0 To UBound(fi)
    xtemp &= fi(i).Name & _
        IIf(i = UBound(fi), "", ",")
Next
Dim xtemp2() As String = _
    Split(xtemp, ",")
ListBox1.DataSource = xtemp2
ListBox1.DataBind()
End If
```

Pada saat proses loading halaman web, maka Listbox akan diisi oleh seluruh file gambar yang ada dalam folder di solution.

10. Langkah terakhir adalah melakukan dobel klik di Button untuk *Close* dan ketikkan listing berikut :

```
Me.ModalPopupExtender1.Hide()
```

Dalam sebuah komponen yang akan dijadikan sebagai komponen Pop Up (umumnya Panel), harus diletakkan pemicu untuk menutup kotak Pop Up. Umumnya pemicu tersebut adalah Button atau LinkButton (dan seringkali juga berupa ImageButton dengan icon yang menandakan fungsi close) yang harus diisikan dengan perintah untuk menutup kotak Pop Up tersebut.

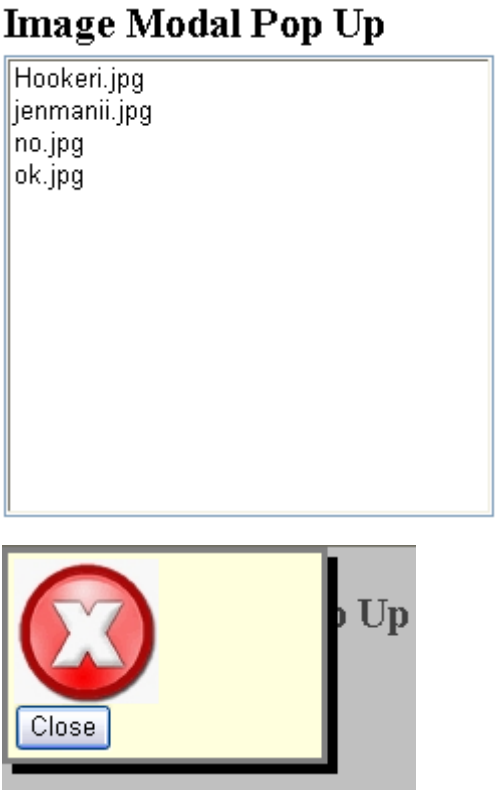


Image Modal Pop Up
Hookeri.jpg
jenmanii.jpg
no.jpg
ok.jpg
Up
Close

Contoh 4 : Rating

Penggunaan komponen Rating, umumnya digunakan dalam polling yang bersifat sederhana, dalam artian bahwa polling atau survei yang diadakan tidak bersifat majemuk. Salah satu penggunaan Rating yang paling sering dilakukan adalah penggunaan dalam polling status tingkat favorit dari sebuah situs.

Komponen Rating dalam AJAX Control Toolkit sebenarnya adalah kumpulan dari image berukuran kecil (sejenis icon) yang disusun sejajar (sesuai keinginan, mendatar atau menurun). Dari kumpulan image tersebut nantinya akan "ditangkap" saat pengguna melakukan klik didalamnya dan kemudian diterjemahkan menjadi sebuah angka berdasarkan skala yang ada dalam property di komponen Rating.

Hal yang perlu diperhatikan dalam penggunaan komponen Rating adalah adanya kebutuhan file gambar kecil (usahakan tidak terlalu besar dan memiliki ekstensi .jpg) sebanyak tiga buah dengan tampilan yang sejenis. Gambar pertama akan digunakan sebagai gambar default pada saat komponen Rating pertama kali muncul, sedangkan gambar kedua digunakan pada saat proses pengisian nilai angka hasil klik dari Rating memasuki masa tunggu. Umumnya proses masa tunggu diwakili oleh

komponen UpdateProgress, tetapi dalam hal ini komponen Rating memiliki cara tersendiri dengan menampilkan file gambar jenis tunggu. Gambar terakhir merupakan gambar yang berfungsi sebagai gambar dalam komponen Rating yang bersifat "off".

Sebagai contoh, gambar yang digunakan dalam contoh dokumentasi AJAX Control Toolkit yang terdapat di folder *images* pada folder *SampleWebSite* dari AJAX Control Toolkit (dan juga dalam contoh kasus berikutnya) adalah sebagai berikut :

Nama File	Gambar	Keterangan
EmptyStar.jpg		Digunakan untuk property empty Star
FilledStar.jpg		Digunakan untuk property filled Star
SavedStar.jpg		Digunakan untuk property waiting Star

Selain membutuhkan tiga buah file gambar yang sejenis, komponen Rating juga membutuhkan tiga class CSS untuk tiap implementasi dari file gambar tersebut. Dari tiap class CSS, digunakan property *background-image* secara *tiled* yang nantinya akan secara otomatis

diimplementasikan gambar tersebut berderet sebanyak nilai yang diberikan di dalam property *MaxRating*.

Contoh implementasi dari komponen Rating dalam studi kasus berikut adalah untuk mendapatkan opini pengunjung sebuah situs mengenai tingkat kesukaan dari desain yang ada.

1. Buat sebuah website baru dengan tipe ASP .NET AJAX Website
2. Buat sebuah database baru didalamnya dengan nama *Polling* yang didalamnya terdapat sebuah tabel bernama *Pertanyaan* dengan struktur sebagai berikut :

Column Name	Data Type	Allow Nulls
LevelSuka	int	☐
Jumlah	int	☑

3. Isikan data contoh dalam tabel tersebut seperti contoh berikut ini (isi data contoh dapat diganti sesuai keinginan, tetapi harus disesuaikan dengan isi dari property *MaxRating* di dalam komponen Rating yang secara default berisikan nilai 5) :

LevelSuka	Jumlah
1	0
2	0
3	0
4	0
5	0

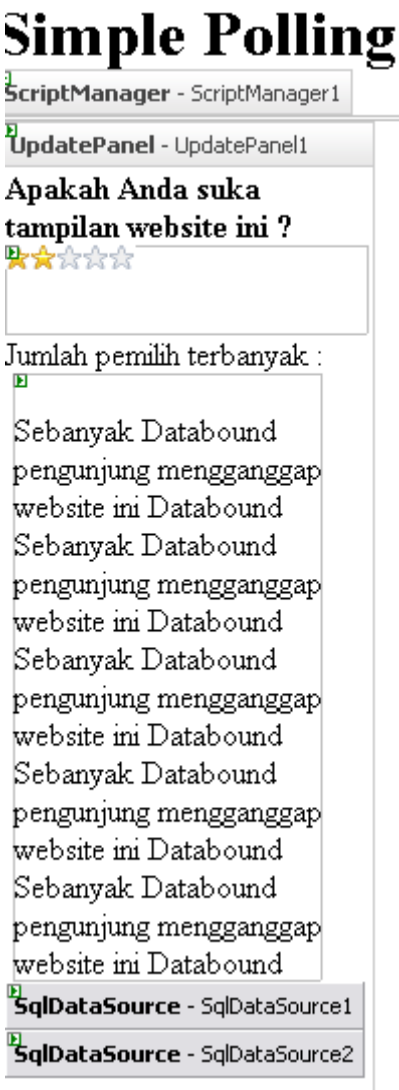
4. Kemudian, buat sebuah file CSS baru dengan isi class sebagai berikut :

```
.ratingStar {
    font-size: 0pt;
    width: 13px;
    height: 12px;
    margin: 0px;
    padding: 0px;
    cursor: pointer;
    display: block;
    background-repeat: no-repeat;
}
.filledRatingStar {
    background-image: url(FilledStar.png);
}
.emptyRatingStar {
    background-image: url(EmptyStar.png);}
.savedRatingStar {
    background-image: url(SavedStar.png);}
```

Class yang digunakan dalam komponen Rating minimal terdiri dari empat macam, yaitu untuk kotak dari komponen Rating (ratingStar), saat Rating dipilih (filledRatingStar), saat Rating tidak dipilih (emptyRatingStar) dan saat menunggu proses pemilihan (savedRatingStar).

5. Selanjutnya, di halaman web yang tersedia, drag sebuah komponen UpdatePanel. Di dalam komponen Update Panel tersebut, secara berturut-turut tempatkan komponen Rating dan drag dua kali tabel *Pertanyaan* dari Database Explorer.
6. Dari dua kali hasil drag tabel *Pertanyaan* tersebut, hapus kedua GridView yang secara otomatis terbentuk. Dan sebagai gantinya, drag sebuah komponen *Repeater* ke dalam UpdatePanel tersebut.

GridView yang terbentuk secara otomatis tidak digunakan dalam contoh ini. Tetapi dalam proses penghapusan, hendaknya berhati-hati agar pada saat proses penghapusan GridView, komponen SqlDataSource juga tidak ikut terhapus. Salah satu trik termudah untuk mengatasi hal tersebut adalah dengan melakukan klik terlebih dulu ke bagian kosong halaman web, kemudian klik pada GridView yang akan dihapus, dan setelah itu tekan tombol Delete untuk penghapusan.



7. Pada komponen SqlDataSource2, isikan property *DataSourceMode* menjadi *DataReader*.

8. Pada komponen Repeater, klik Smart Tag dan arahkan property Data Source ke SqlDataSource1.

9. Selanjutnya, berpindahlah ke mode *Source* dan modifikasi tiap komponen seperti pada petunjuk berikut ini :

   a. Ketik listing berikut di antara tag *head* di bagian paling atas halaman web, untuk menghubungkan file CSS ke dalam halaman web :

```
<link href="StyleSheet.css"
rel="stylesheet" type="text/css" />
```

Perlu diingat bahwa penamaan dan peletakan file CSS adalah sesuai selera tiap webmaster. Dalam contoh kasus di AJAX Control Toolkit, hampir seluruh file CSS yang digunakan diberi nama default yaitu *StyleSheet.css* dan diletakkan di dalam folder yang sama dengan halaman web.

   b. Rating

```
<cc1:rating id="Rating1" runat="server"
```

```
currentrating="0"
emptystarcssclass="emptyRatingStar"
filledstarcssclass="filledRatingStar"
starcssclass="ratingStar"
OnChanged="Pilih"
Tag=<%#eval("kodePertanyaan") %>
waitingstarcssclass="savedRatingStar"
Height="45px" Width="181px"
Direction="LeftToRight"
AutoPostBack="True"></cc1:rating>
```

Modifikasi komponen Rating difokuskan pada pengisian property untuk CSS class (filledStarCSSClass, watingStarCSSClass dan starCSSClass serta emptyStarCSSClass) yang diisi dari class yang sebelumnya telah dibuat di file CSS.

Modifikasi lainnya adalah pada pengisian property *AutoPostBack* yang diisi dengan nilai *true*, sehingga pada saat Rating dipilih maka akan langsung melakukan eksekusi prosedur *Pilih* (yang diisikan di property *OnChanged*). Prosedur ini nantinya akan dibuat di dalam listing program.

c. SqlDataSource1

Carilah property *SelectCommand* dan ganti perintah SQL yang ada dengan perintah SQL berikut ini :

```
SELECT LevelSuka, Jumlah FROM Pertanyaan
ORDER BY Jumlah DESC
```

d. SqlDataSource2

Sama dengan langkah sebelumnya, carilah property *SelectCommand* dan ganti perintah SQL dengan perintah berikut :

```
SELECT [LevelSuka], [Jumlah] FROM
[Pertanyaan] WHERE ([LevelSuka] =
@LevelSuka)
```

Penggantian perintah SQL, khususnya untuk perintah Select pada SqlDataSource1, digunakan pada saat tampilan data di Repeater agar urut dari jumlah pemilih terbanyak. Sedangkan pada SqlDataSource2, digunakan untuk mencari data rating yang dipilih, sehingga pada saat proses update terjadi, nilai yang lama dapat secara otomatis ditambahkan.

e. Repeater1

```
<asp:Repeater ID="Repeater1" runat="server"
```

```
DataSourceID="SqlDataSource1" >
<ItemTemplate >
<br /> Sebanyak <%#Eval("Jumlah")%>
pengunjung mengganggap website ini
<%#IIf(Eval("LevelSuka") = 5, "terbaik", _
     IIf(Eval("LevelSuka") = 4, "bagus", _
     IIf(Eval("LevelSuka") = 3, "biasa", _
     IIf(Eval("LevelSuka") = 2, "kurang ", _
     "jelek"))))%>
</ItemTemplate>
</asp:Repeater>
```

Modifikasi komponen Repeater terletak pada pengisian tag *ItemTemplate* yang diisikan dengan data dari tabel *Pertanyaan* dan menerjemahkan level suka dari pengunjung ke dalam kata-kata dengan menggunakan perintah *Iif*.

10. Kini berpindahlah lagi ke mode listing program dan bagian paling atas dari listing program, ketikkan listing berikut untuk melakukan impor namespace dari AJAX Control Toolkit.

```
Imports AjaxControlToolkit
```

11. Selanjutnya ketikkan listing prosedur *Pilih* yang digunakan pada saat komponen Rating diklik oleh pengguna.

```
Sub Pilih(ByVal sender As Object, _
    ByVal e As RatingEventArgs)
        SqlDataSource2.SelectParameters(0). _
            DefaultValue = _
            Rating1.CurrentRating
        Dim xreader As Data.IDataReader = _
        SqlDataSource2.Select _
            (DataSourceSelectArguments.Empty)
        xreader.Read()
        Dim xjumlah As Integer = _
            xreader.Item(1)
        SqlDataSource1.UpdateParameters(0). _
            DefaultValue = xjumlah + 1
        SqlDataSource1.UpdateParameters(1). _
            DefaultValue = Rating1.CurrentRating
        SqlDataSource1.Update()
        Rating1.CurrentRating = 0
End Sub
```

Pada saat komponen Rating dipilih oleh pengguna, maka langkah pertama yang dilakukan adalah membaca data dari tabel *Pertanyaan* dan kemudian dari SqlDataSource2 diumpanbalikkan parameter dari level rating yang dipilih.
Kemudian dari data yang telah dibaca, dilakukan proses update data dengan menambahkan field *Jumlah* dengan angka 1 sebagai tanda bahwa komponen Rating telah dipilih.

Simple Polling

Apakah Anda suka tampilan website ini ?

3

Jumlah pemilih terbanyak :

Sebanyak 3 pengunjung mengganggap website ini bagus
Sebanyak 2 pengunjung mengganggap website ini terbaik
Sebanyak 2 pengunjung mengganggap website ini kurang
Sebanyak 1 pengunjung mengganggap website ini biasa
Sebanyak 1 pengunjung mengganggap website ini jelek

Contoh 5 : TextBoxWatermark, DropDown dan ConfirmButton

Contoh selanjutnya akan menggunakan tiga jenis komponen AJAX Control Toolkit. Ketiga jenis komponen ini yaitu : TextBoxWatermark, DropDown dan ConfirmButton, meski memiliki beragam fungsi, tetapi seringkali digunakan dalam satu jenis halaman web, yaitu halaman web untuk melakukan manipulasi data.

TextBoxWatermark merupakan modifikasi dari komponen Textbox biasa yang didalamnya akan diberi sebuah pesan dalam bentuk Watermark atau samar pada saat halaman web pertama kali dilihat oleh pengguna. Sehingga pengguna akan mendapatkan petunjuk mengenai cara mengisi sebuah Textbox secara langsung. Petunjuk yang muncul, nantinya akan secara otomatis hilang pada saat Textbox yang dimaksud diklik oleh pengguna atau dalam keadaan fokus.

Komponen ini pada implementasinya membutuhkan sebuah Textbox biasa sebagai target dari TextBoxWatermark yang akan menampilkan petunjuk di dalam Textbox biasa. Tiap Textbox biasa membutuhkan sebuah TextBoxWatermark tersendiri untuk menampilkan petunjuk yang berbeda.

Sedangkan komponen DropDown, memiliki fungsi yang berbeda dengan komponen DropDownList biasa. Komponen DropDown, pada dasarnya merupakan sebuah Textbox yang jika diklik oleh pengguna dalam suatu halaman web akan menampilkan sebuah DropDownList atau juga RadioButtonList didalamnya. Dan jika terjadi pemilihan di dalam DropDownList atau RadioButtonList, maka nilai yang dipilih akan diumpanbalikkan ke dalam Textbox.

Salah satu kelebihan dari penggunaan komponen DropDown adalah penghematan ruang dalam sebuah halaman web serta interaksi yang lebih baik antara pengguna dengan halaman web. Penghematan ruang terjadi dikarenakan RadioButtonList (jika menggunakan RadioButtonList) tidak akan terlihat kecuali pada saat pengguna berusaha mengisi Textbox yang ada. Tetapi, di sisi yang lain, webmaster juga harus memikirkan CSS yang akan diterapkan di dalam RadioButtonList, agar kesan RadioButtonList menjadi lebih "tegas" dibandingkan warna dari halaman web itu sendiri, sehingga komponen DropDown dapat menarik fokus perhatian dari pengguna.

Komponen terakhir yang akan digunakan dalam contoh ini adalah komponen ConfirmButton. Seperti halnya komponen di dalam AJAX Control Toolkit yang lain, komponen ConfirmButton bukanlah tiruan dari komponen

Button. Tetapi merupakan penghubung antara komponen Button yang sudah ada sehingga dapat menampilkan pesan konfirmasi pada saat Button tersebut diklik oleh pengguna.

Umumnya komponen ConfirmButton digunakan untuk menjembatani proses penghapusan data. Karena pengguna seringkali melakukan kesalahan fatal yang tidak disengaja dalam proses penghapusan. Tapi juga tidak menutup kemungkinan bahwa komponen tersebut digunakan sebagai jembatan sebelum proses penyimpanan dilakukan, baik dalam prose edit maupun proses input data.

Dalam contoh yang akan dijelaskan, ketiga komponen tersebut yaitu TextBoxWatermark, DropDown dan ConfirmButton akan digunakan secara bersama-sama dalam sebuah halaman web yang berfungsi untuk melakukan entri data dalam sebuah tabel. Bagi sebuah halaman web "biasa", penggunaan ketiga komponen tersebut memang tidak akan membuat perbedaan yang sangat signifikan. Tetapi, bagi pengguna "awam", penggunaan ketiga komponen tersebut akan sangat membantu dalam proses interaksi dengan halaman web, khususnya halaman web yang berbasis AJAX.

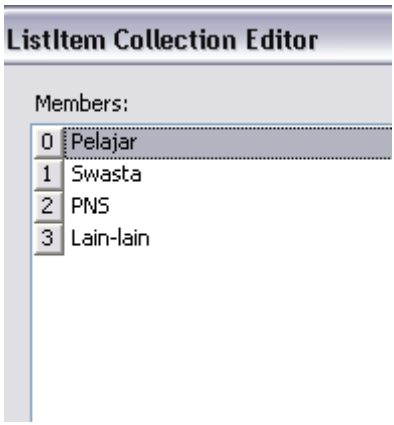
Untuk lebih lanjut membahas mengenai penggunaan ketiga komponen tersebut, ikuti langkah-langkah berikut ini :

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Website
2. Didalamnya, buat sebuah database dengan nama *Anggota* yang didalamnya terdapat sebuah tabel yang juga bernama *Anggota* dengan struktur tabel sebagai berikut :

Column Name	Data Type	Allow Nulls
Nama	varchar(50)	☐
Pekerjaan	varchar(50)	☐
Status	varchar(50)	☐

3. Kemudian di dalam halaman web yang tersedia, buat tabel dari menu *Layout → Insert Table* dengan ketentuan empat baris dan dua kolom.
4. Di kolom pertama, ketikkan di tiap baris masing-masing keterangan untuk *Nama, Pekerjaan* dan *Status*
5. Sedangkan di kolom kedua baris pertama, tempatkan UpdatePanel yang didalamnya diletakkan sebuah Textbox dan sebuah TextBoxWatermark didalamnya (pastikan bahwa AJAX Control Toolkit telah ada sebelumnya).
6. Set property *TargetControlID* dari TextBoxWatermark mengarah ke *Textbox1*

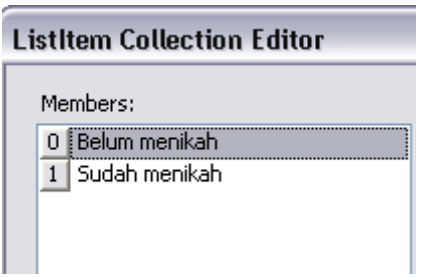
7. Kemudian, masih di kolom kedua tetapi di baris kedua, tempatkan lagi sebuah UpdatePanel dan didalamnya letakkan sebuah Textbox, sebuah komponen DropDown dan sebuah RadioButtonList.
8. Untuk RadioButtonList, set property yang ada menjadi sebagai berikut :
    a. AutoPostBack : True
    b. BorderStyle : Solid
    c. BackColor : #E0E0E0
    d. Item (tampak seperti pada gambar)



9. Kini berpindahlah ke mode *Source*, dan ganti definisi tag dari komponen DropDown menjadi seperti pada listing berikut :

```
<cc1:DropDownExtender ID="DropDownExtender1"
runat="server" TargetControlID="Textbox2"
DropDownControlID ="RadioButtonList1" >
</cc1:DropDownExtender>
```

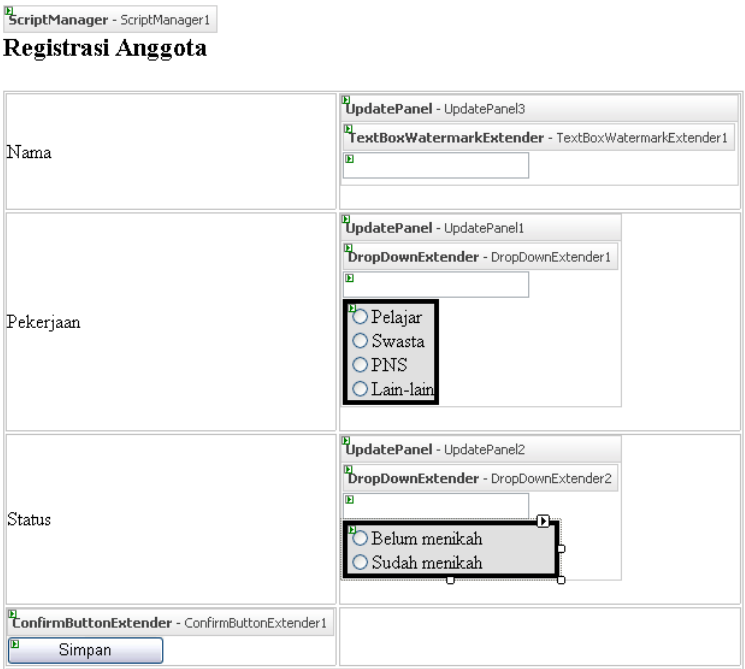
10. Kemudian, di baris ketiga (masih tetap di kolom kedua), tempatkan lagi sebuah UpdatePanel yang didalamnya juga terdapat sebuah Textbox dan sebuah komponen DropDown beserta satu RadioButtonList.
11. Set property RadioButtonList seperti pada langkah nomor delapan, tetapi untuk pengisian property item, set seperti pada gambar berikut :



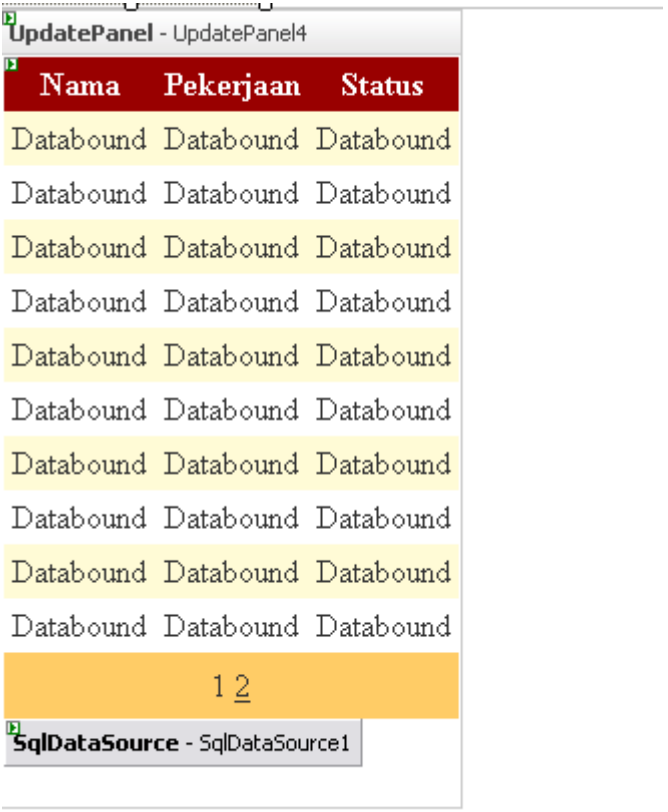
Setting property Item secara manual di dalam RadioButtonList, seringkali dilakukan jika pilihan yang akan ditampilkan bersifat statis dan memiliki kemungkinan sangat jarang untuk berubah. Misalnya : Status Menikah, Status Pekerjaan dan lainnya.

12. Lakukan langkah yang sama seperti pada nomor sembilan dan ganti listing dari DropDown seperti pada listing berikut :

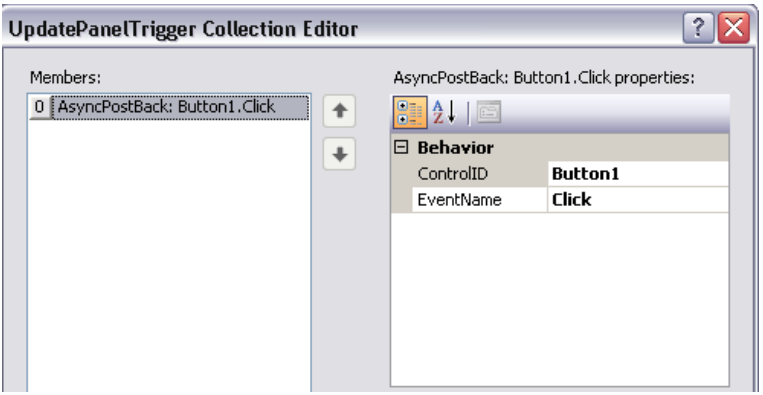
13. Kini, di baris terakhir dari tabel yang ada, tempatkan sebuah komponen Button dan sebuah komponen ConfirmButton. Lalu set property dari ConfirmButton mengarah ke Button1.



14. Kini, tambahkan UpdatePanel yang terakhir di bagian bawah dari Button, dan untuk akses ke database, drag tabel *Anggota* dari Database Explorer ke dalam bagian paling bawah halaman web.



15. Di UpdatePanel yang terakhir, set property *UpdateMode* menjadi *Conditional*, lalu klik di property *Triggers* dan pada kotak dialog yang tersedia, set menjadi seperti pada gambar berikut ini :



Pengisian parameter *Trigger* dalam UpdatePanel menandakan bahwa UpdatePanel akan "dipaksa" untuk melakukan postback pada saat sebuah komponen diberi sebuah reaksi tertentu. Trigger atau pemicu dalam Update Panel yang terakhir ini adalah penekanan Button *Simpan* yang berarti bahwa dalam halaman web akan dilakukan penyimpanan data, sehingga GridView diharapkan juga akan secara otomatis melakukan refresh.

16. Lalu, dobel klik masing-masing pada RadioButtonList yang pertama dan kedua serta pada Button, dan untuk tiap komponen ketikkan listing berikut :

    a. RadioButtonList1

```
TextBox2.Text = _
    RadioButtonList1.SelectedValue
```

b. RadioButtonList2

```
TextBox3.Text = _
    RadioButtonList2.SelectedValue
```

c. Button1

```
With SqlDataSource1
    .InsertParameters(0). _
        DefaultValue = TextBox1.Text
    .InsertParameters(1). _
        DefaultValue = TextBox2.Text
    .InsertParameters(2). _
        DefaultValue = TextBox3.Text
    .Insert()
End With
```

Pada saat halaman web dieksekusi, maka tiap RadioButtonList secara otomatis akan dalam keadaan *hidden* tanpa harus dilakukan setting property *Visible*. Perhatikan pula pada Textbox pertama yang langsung secara otomatis memiliki petunjuk *Watermark* didalamnya.

Contoh 6 : Accordion

Komponen AJAX Control Toolkit yang cukup menarik perhatian dari sisi efek visual adalah komponen Accordion. Komponen Accordion yang menimbulkan efek sliding dari beberapa panel secara bertumpuk, biasanya digunakan para webmaster untuk menimbulkan efek "wow" dalam sebuah halaman web.

Komponen Accordion saat diimplementasikan membutuhkan dua buah komponen didalamnya yaitu komponen Accordion sendiri sebagai "wadah" utama serta Accordion Panes sebagai tempat dari tiap panel yang akan diimplementasikan. Di dalam contoh implementasi komponen Accordion yang terdapat dalam dokumentasi AJAX Control Toolkit, komponen Accordion hanya diimplementasikan sebagai "dekorasi" dari sebuah halaman web untuk menimbulkan efek "wow" tersebut.

Tetapi dalam contoh kali ini, komponen Accordion akan diimplementasikan dengan menggunakan koneksi database, sehingga dapat dilihat utilitas yang baik dari komponen Accordion itu sendiri.

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Website
2. Untuk koneksi database, akan digunakan database serta tabel yang sama dengan contoh sebelumnya

(lihat lagi mengenai contoh kelima), yaitu database *Anggota.*

Untuk menambahkan database yang sudah ada ke dalam solution yang baru, gunakan menu *Website → Add Existing Item* kemudian arahkan ke folder yang didalamnya terdapat file *Anggota.mdf*. Kemudian pada saat file tersebut terpilih, maka akan diberikan konfirmasi agar dibuat folder *App_Data* (jika belum ada). Dan juga file *Anggota.ldf* (sebagai log file database) akan langsung ditambahkan secara otomatis.
Pada saat penambahan database yang sudah ada ke dalam sebuah solution yang lain, usahakan database tersebut dalam keadaan tidak terkoneksi. Atau dengan kata lain bahwa solution atau situs yang menampung database lama tidak dalam keadaan terbuka atau tereksekusi. Karena jika hal tersebut terjadi, akan mengakibatkan kemungkinan database yang ditambahkan mengalami *file corrupt*.

3. Selanjutnya, di halaman web yang tersedia, drag sebuah komponen Accordion di dalamnya.
4. Berikutnya drag tabel *Anggota* ke dalam halaman web sehingga muncul komponen *SqlDataSource* dan sebuah GridView.

5. Hapus GridView yang muncul akibat hasil drag tabel *Anggota*
6. Lalu, berpindahlah ke mode Source, dan carilah tag *<Accordion ….>*, kemudian modifikasi tag didalamnya menjadi seperti listing berikut ini :

```
<cc1:Accordion ID="Accordion1"
       runat="server"
       DataSourceID="SqlDataSource1">
<HeaderTemplate>
    <div style="background-color:Silver;
       border:dashed 1px black;
       padding:2px;cursor:pointer ">
    Nama : <%#Eval("Nama")%>
    </div>
</HeaderTemplate>
<ContentTemplate >
    <div style="background-color:Aqua ;
    border:dashed 1px gray;
    padding:2px; border-top:none;">
     Pekerjaan :
     <%#Eval("Pekerjaan")%>  <br />
     Status : <%#Eval("Status")%>
     </div>
</ContentTemplate>
</cc1:Accordion>
```

Modifikasi pokok dari komponen Accordion yang dikoneksikan dengan sebuah tabel dari database adalah pada pengisian property *DataSourceID* dengan komponen SqlDataSource yang berkaitan di dalam halaman web tersebut. Selain itu, juga pada pengisian bagian *HeaderTemplate* serta pengisian *ContentTemplate*. Pada kedua bagian tersebut, didalamnya masing-masing diisikan tampilan field dari tabel yang dikoneksikan kepada Accordion.

Tiap bagian (HeaderTemplate dan ContentTemplate) kemudian diberikan style CSS (dalam contoh langsung diisikan melalui tag HTML *div* sebagai layer, tapi juga bisa diisikan dari class CSS dari file style sheet terpisah) yang masing-masing berbeda (dan diusahakan diberi border agar pengguna dapat memahami bagian mana yang harus diklik untuk melakukan proses ekspansi Accordion).

Nama : Ali

Pekerjaan : Swasta
Status : Belum menikah

Nama : Baba

Nama : Caca

Nama : test

Pada hasil dari implementasi komponen Accordion, field *Nama* menjadi bar utama, sedangkan field *Pekerjaan* dan *Status* menjadi bar yang akan muncul pada saat ekspansi dilakukan. Bandingkan penggunaan komponen ini dengan penggunaan GridView biasa !!

Contoh 7 : Hover Menu

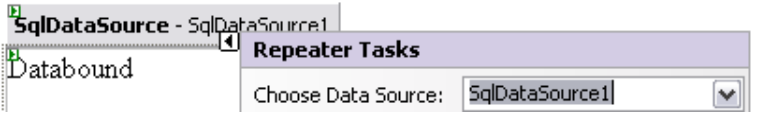
Hover menu merupakan komponen dari AJAX Control Toolkit yang berfungsi sebagai *floating menu* pada saat sebuah komponen yang dijadikan target dilewati cursor mouse oleh pengguna. Penggunaan Hover Menu, sebenarnya tidak harus berupa menu, tetapi bisa juga berupa keterangan yang akan berupa *temporary pop up* pada saat sebuah komponen dalam halaman web dilewati oleh cursor mouse.

Dengan kata lain, Hover Menu (yang dalam contoh resmi AJAX Control Toolkit direpresentasikan dalam sebuah GridView) juga bisa menjadi semacam implementasi tag HTML *acronym*. Tetapi, perbedaan utamanya adalah komponen Hover Menu juga bisa diberikan koneksi database didalamnya, sehingga informasi pop up sementara yang dihasilkan dapat berisikan field dari sebuah tabel.

Dalam contoh yang akan dibahas, Hover Menu diimplementasikan dalam dua buah komponen Panel dan dikoneksikan dengan sebuah tabel dari database dengan menggunakan sebuah komponen Repeater sebagai jembatan didalamnya. Tabel yang digunakan dalam contoh ini adalah tabel yang sama dengan tabel yang digunakan dalam contoh kasus implementasi *Pop Up Control*.

Untuk penambahan database yang sudah ada ke dalam sebuah solution baru, lihat lagi keterangan di contoh sebelumnya.

1. Buat sebuah solution baru dengan tipe ASP .NET AJAX Website, dan tambahkan database *UserProfile* didalamnya.
2. Drag tabel *UserProfile* dari Database Explorer ke dalam halaman web yang tersedia.
3. Kemudian hapus GridView yang terbentuk dari hasil drag tabel tersebut.
4. Selanjutnya drag sebuah komponen Repeater didalamnya dan klik pada Smart Tag untuk mengkoneksikan dengan komponen *SqlDataSource* dari hasil drag tabel.



5. Berikutnya, berpindahlah ke mode Source, dan carilah tag *<Repeater>….</Repeater>*
6. Di dalam tag tersebut, modifikasilah tag seperti pada listing berikut :

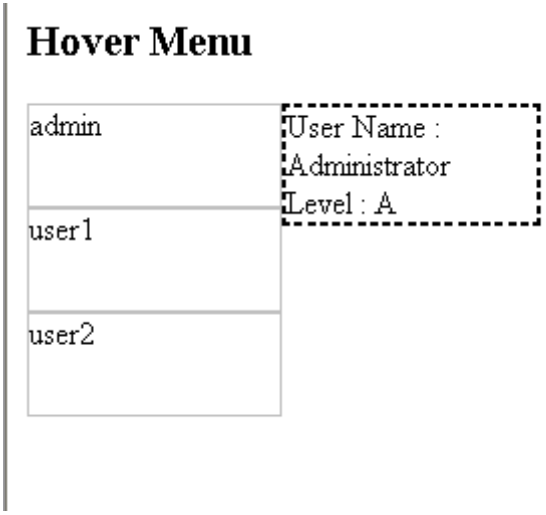
```
<asp:Repeater ID="Repeater1" runat="server"
DataSourceID="SqlDataSource1">
<ItemTemplate>
<asp:Panel ID="Panel1"
    runat="server"
    Height="50px" Width="125px"
    BorderWidth="1px"
    BorderStyle="Solid"
    BorderColor="silver" >
    <div style="cursor:help ">
        <%#Eval("UserID") %>
    </div>
</asp:Panel>
<asp:Panel ID="Panel2"
    runat="server" Height="50px"
    Width="125px"
    BorderColor="black"
    BorderStyle="Dashed"
    BorderWidth="2px">
        User Name :
        <%#Eval("UserName") %>
        <br />
        Level :
        <%#Eval("UserLevel")%>
 </asp:Panel>
<cc1:HoverMenuExtender
    ID="HoverMenuExtender1"
    runat="server"
    PopupControlID="Panel2"
    TargetControlID="Panel1"
```

```
    PopupPosition="Right">
</cc1:HoverMenuExtender>
</ItemTemplate>
</asp:Repeater>
```

Perhatikan penambahan dua buah komponen Panel dan sebuah komponen Hover Menu di bagian *ItemTemplate* dari Repeater tersebut. Komponen Panel pertama (Panel1) berfungsi sebagai penampil field *UserID* dan akan muncul pertama kali pada saat halaman web dieksekusi. Sedangkan komponen Panel yang kedua (Panel2) berfungsi sebagai komponen target dari Hover Menu itu sendiri dengan menampilkan field *UserName* dan *UserLevel* dari tabel *UserProfile*.

Untuk setting dari komponen Hover Menu sendiri membutuhkan dua property utama yaitu *TargetControlID* yang menampung komponen yang akan dilewati oleh cursor mouse (dalam contoh ini adalah Panel yang pertama), serta property *PopUpControlID* yang menampung komponen yang akan dijadikan sebagai komponen pop up sementara (yaitu Panel yang kedua).

Sedangkan property *PopUpPosition* dalam komponen Hover Menu bertujuan untuk menentukan letak pop up sementara yang akan muncul, apakah akan ditampilkan di sebelah kanan, kiri ataupun bawah dari komponen yang menjadi target.


Hover Menu
admin
user1
user2
User Name :
Administrator
Level : A

Studi Kasus II : Mini Blog

Pengantar

Studi kasus berikut ini akan mensimulasikan pembuatan sebuah blog dengan cara yang sangat sederhana. Untuk kepentingan penulisan buku, banyak fitur dan fasilitas dari studi kasus blog ini yang dihilangkan. Penghilangan fitur tersebut tidak akan mengubah esensi dari blog tersebut, tetapi hal yang terpenting dari studi kasus ini adalah penggunaan ASP .NET AJAX serta pemakaian AJAX Control Toolkit itu sendiri.

Blog yang akan dibuat merupakan jenis single blog sederhana, tanpa ada fasilitas untuk mengiputkan komentar di tiap blog yang diisi. Selain itu, fasilitas entri blog hanya berupa teks, tidak menangani entri dengan menggunakan tag HTML.

Karenanya, setelah pembuatan studi kasus ini diharapkan Anda dapat mengembangkan beberapa fitur antara lain :

1. Pemberian fasilitas komentar di tiap berita
2. Pemberian fasilitas archieve bulanan
3. Entri isi blog yang mendukung isi dengan tag HTML dengan menggunakan komponen editor teks yang free seperti FreeTextBox.
4. Pemberian fasilitas untuk mengganti theme dengan menggunakan CSS.

Blog yang akan dibuat sendiri hanya terdiri dari dua halaman web yaitu halaman web utama yang menampilkan lima isi blog terbaru, sekaligus arsip blog yang dibagi berdasarkan kategori. Halaman web utama juga menampilkan form untuk melakukan login bagi administrator. Sedangkan halaman kedua adalah halaman untuk pihak administrator yang melakukan maintenance kategori serta maintenance dari blog itu sendiri.

Dalam studi kasus ini akan digunakan teknik *Master Pages* didalamnya. Penggunaan Master Pages tidak dijelaskan secara rinci dalam buku ini, untuk mengetahui lebih lanjut mengenai pembahasan Master Pages dapat dibaca di buku *Panduan Belajar ASP .NET 2.0* oleh pengarang dan penerbit yang sama.

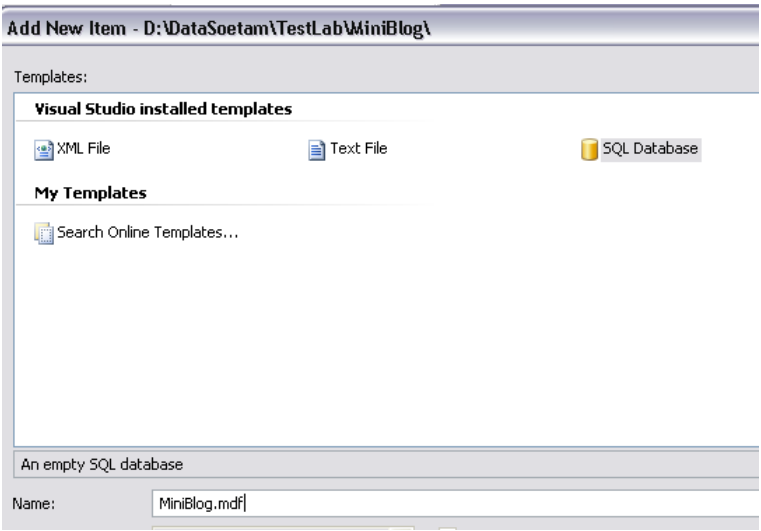
Langkah pertama dalam pembuatan studi kasus ini sama dengan contoh sebelumnya yaitu membuat sebuah solution dengan tipe ASP .NET AJAX didalamnya. Kemudian dilanjutkan dengan pembuatan database yang akan dijelaskan dalam sub bab selanjutnya.

Disarankan untuk mengikuti langkah pembuatan secara berurutan dalam pembuatan studi kasus ini.

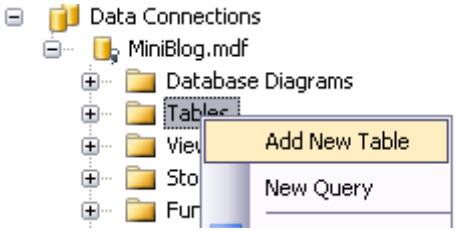
Pembuatan Database

Database dalam studi kasus ini tergolong sederhana, hanya terdiri dari tiga buah tabel yang masing-masing berfungsi untuk menampung data untuk kepentingan login, kategori blog serta isi dari blog itu sendiri.

Dari solution yang telah dibuat sebelumnya, klik kanan pada folder *App_Data* dan pilih sub menu *Add New Item* dan pada kotak dialog yang tersedia, tambahkan sebuah database baru dengan nama *MiniBlog*



Kemudian, klik pada *Database Explorer*, dan pada sub tree *Table*, klik kanan dan pilih sub menu *Add New Table*.



Lalu, buat tiga buah tabel dengan struktur sebagai berikut :

Tabel UserProfile

Column Name	Data Type	Allow Nulls
UserID	varchar(10)	☐
UserName	varchar(50)	☐
UserPassword	varchar(10)	☑

Tabel Category

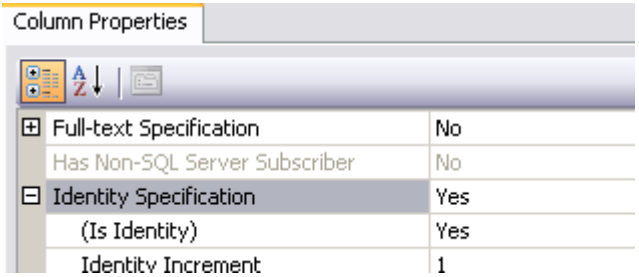
Column Name	Data Type	Allow Nulls
Category	varchar(50)	☐
StatusCategory	bit	☐

Tabel IsiBlog

Column Name	Data Type	Allow Nulls
idBlog	int	☐
TglBlog	datetime	☐
JudulBlog	varchar(MAX)	☐
Category	varchar(50)	☐
IsiBlog	varchar(MAX)	☐
StatusBlog	bit	☐

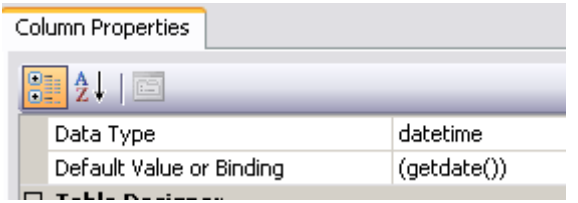
Khusus untuk tabel *IsiBlog*, di field *idBlog* diset agar menjadi field yang memiliki karakteristik *autonumber* atau

melakukan counter otomatis pada saat sebuah record ditambahkan. Hal ini bisa dilakukan dengan mengubah setting *Identity Specification* menjadi *Yes* pada field tersebut. Untuk melakukannya, klik pada field *idBlog* dan lihat pada kotak property dibawahnya lalu set property *IsIdentity* menjadi *Yes*.



Penggunaan field tipe *Identity* lebih banyak digunakan untuk field dengan tipe numerik (int, double dan sejenisnya). Perhatikan pula, resiko dari penggunaan field jenis *Identity* yang pada saat sebuah proses penghapusan record yang berada di tengah, maka nomor urut akan meloncat selamanya. Misal : jika terdapat 10 buah record, dan kemudian record ke-5 dihapus, maka nomor 5 akan hilang selamanya, sehingga urutan nomor akan meloncat dari 1 hingga 4 lalu dilanjutkan nomor 6 hingga 10.

Sedangkan untuk field *TglBlog* diset secara otomatis agar pada saat pengisian record baru langsung diisi dengan tanggal dan jam sistem yang bersangkutan, sehingga tidak diperlukan lagi proses entri tanggal dalam pengisian blog nantinya. Untuk melakukan hal tersebut, bisa dengan melakukan setting *Column Properties* pada field *TglBlog* dengan menempatkan fungsi *GetDate()* didalamnya. Sehingga, pada saat record diisikan, field tersebut akan langsung berisikan tanggal pada saat blog disimpan.



Berikutnya, isikan data contoh pada tabel *UserProfile* dan tabel *Category* untuk kepentingan testing program. Data contoh dapat diganti sesuai dengan keinginan Anda, tetapi harus tetap diperhatikan pada saat testing dilakukan.

Cara pengisian data contoh dengan melakukan klik kanan pada tabel yang bersangkutan, dan pilih sub menu *Show Table Data*. Kemudian isikan data contoh seperti pada gambar berikut :

Tabel UserProfile

UserID	UserName	UserPassword
admin	Administrator	4dm1n

Tabel Category

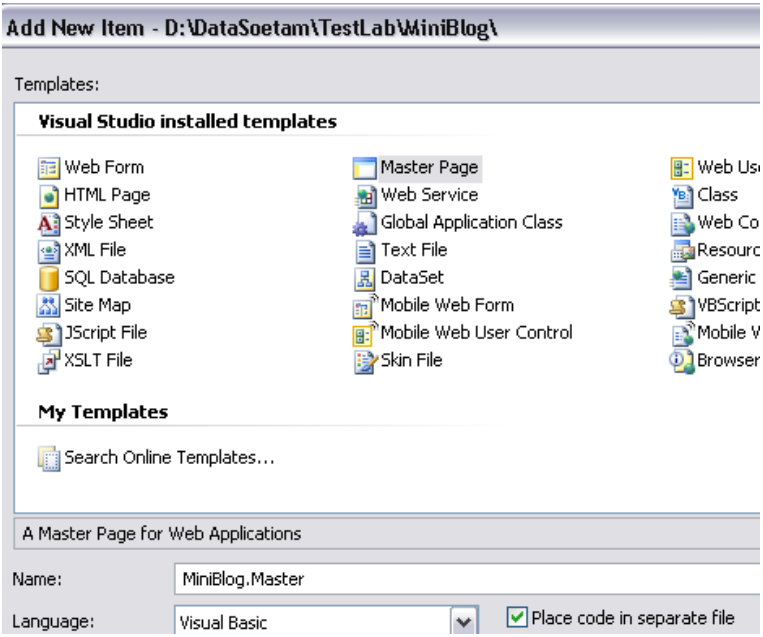
Category	StatusCategory
Campus Life	True
Knowledge	True
Link	True
Personal Life	True

Master Page

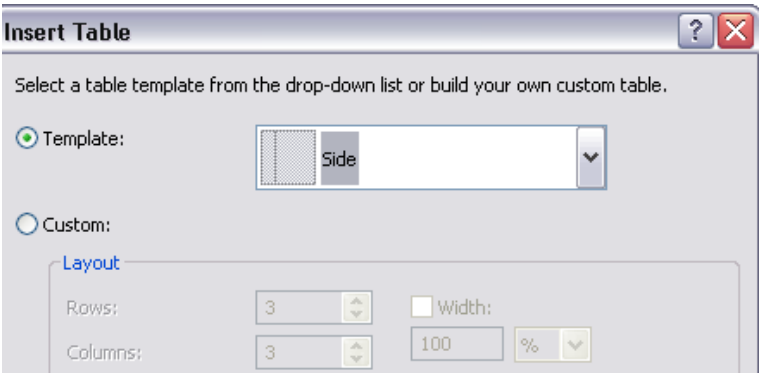
Pembuatan Master Page dilakukan agar terjadi konsistensi desain pada sebuah situs yang memiliki lebih dari satu halaman web. Selain itu juga mempermudah desainer untuk berkolaborasi lebih lanjut dengan programmer dalam menyatukan visi antara desain dan program salam sebuah situs (untuk mempelajari lebih lanjut mengenai Master Page, bisa dibaca buku dengan judul *Panduan Belajar ASP .NET 2.0* oleh pengarang dan penerbit yang sama).

Pembuatan Master Page dalam sebuah solution dapat dilakukan dengan cara melakukan klik kanan pada solution, kemudian pilih sub menu *Add New Item* lalu dalam kotak dialog yang tersedia pilih tipe *Master Page*. Untuk studi kasus ini, Master Page akan diberi nama *MiniBlog*.

Di dalam Master Page tersebut, secara otomatis terdapat sebuah komponen *ContentPlaceHolder* yang nantinya menjadi tempat dari halaman web yang akan dibuat. Perlu diingat, bahwa sebuah Master Page tidak dapat berdiri sendiri, melainkan harus minimal memiliki sebuah halaman web yang didalamnya mengacu kepada Master Page tersebut.

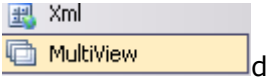


Berikutnya tempatkan sebuah tabel dalam Master Page tersebut, dengan menggunakan template *side* dari menu *Layout→ Insert Table* dan set template seperti pada gambar berikut :

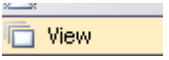


Kemudian, set property *valign* tiap kolom dari tabel menjadi *top*. Cara melakukan setting property kolom dari tabel cukup dengan melakukan klik di tiap kolom yang dimaksud dan membuka window property. Hal ini dilakukan agar pada langkah desain berikutnya, komponen yang diletakkan akan secara langsung mengarah ke bagian paling atas dari halaman web.

Selanjutnya, tempatkan sebuah komponen MultiView



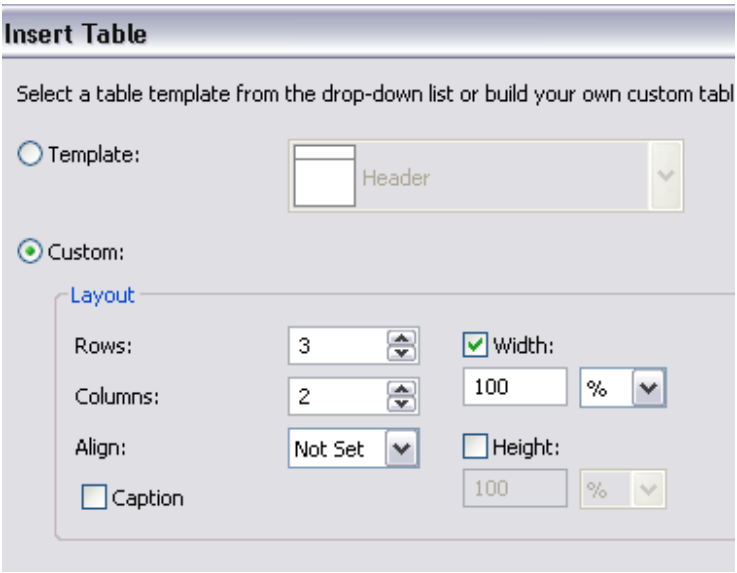
di kolom pertama. Di dalam MultiView tersebut, tempatkan dua buah komponen View



untuk kepentingan halaman utama dan halaman administrator.

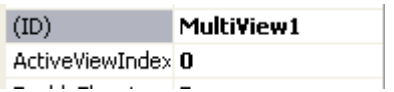
Sebuah komponen MultiView tidak bisa berdiri sendiri, didalamnya minimal terdapat sebuah komponen View.

Di View yang pertama, tempatkan judul situs serta sebuah tabel (dengan menggunakan menu *Layout* → *Insert Table*) yang memiliki dua kolom dan tiga baris serta lebar 100% secara relatif dari kolom itu sendiri. Tabel ini nantinya ditempati oleh dua buah Textbox dan sebuah Button untuk kepentingan proses login.



Kemudian masih di View yang pertama, drag tabel *UserProfile* dari Database Explorer di dalam View tersebut. Hapus GridView yang terbentuk dari hasil drag tersebut, karena yang dibutuhkan dalam proses nantinya hanya komponen SqlDataSource. Sedangkan di View yang kedua, tempatkan sebuah LinkButton untuk kepentingan Logout.

Selanjutnya set property *ActiveViewIndex* menjadi *0* pada MultiView agar secara default MultiView akan mengarah langsung ke View yang pertama.

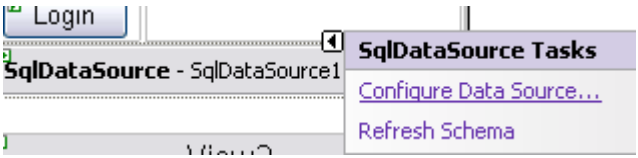


Pada Textbox yang akan digunakan untuk pengisian Password, set property *TextMode* menjadi *Password* agar Textbox tersebut memiliki fasilitas *masking* pada saat pengisian password.

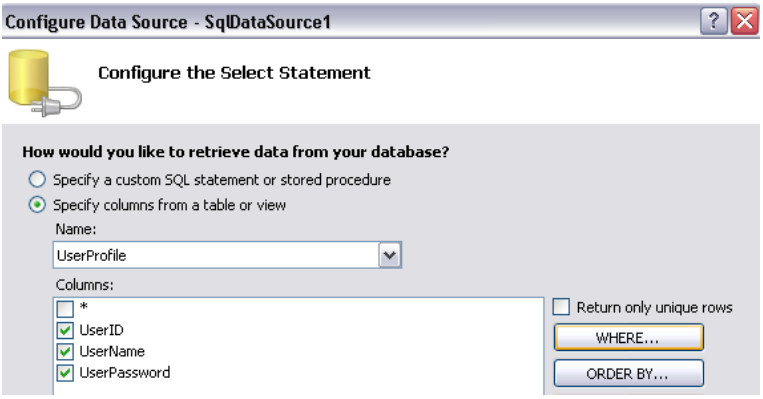
Lalu set property *DataSourceMode* SqlDataSource tersebut menjadi *DataReader.* Hal ini dilakukan karena

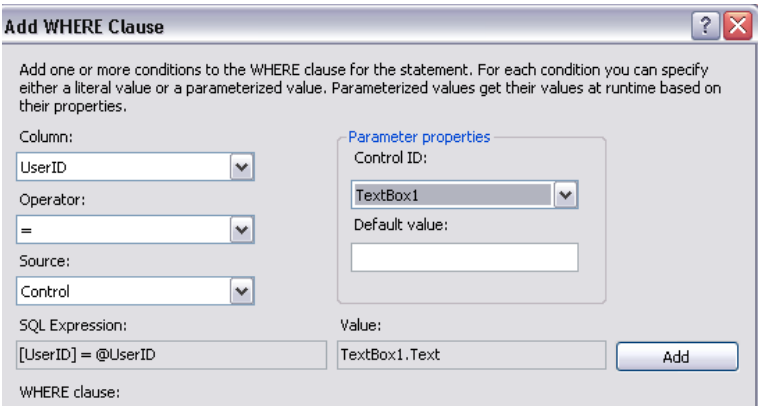
koneksi yang diperlukan hanya untuk pembacaan data secara sederhana, bukan untuk manipulasi data.

Kini, klik pada Smart Tag di komponen SqlDataSource yang tersedia, dan pilih sub menu *Configure Data Source* untuk memodifikasi perintah SQL yang ada didalamnya.

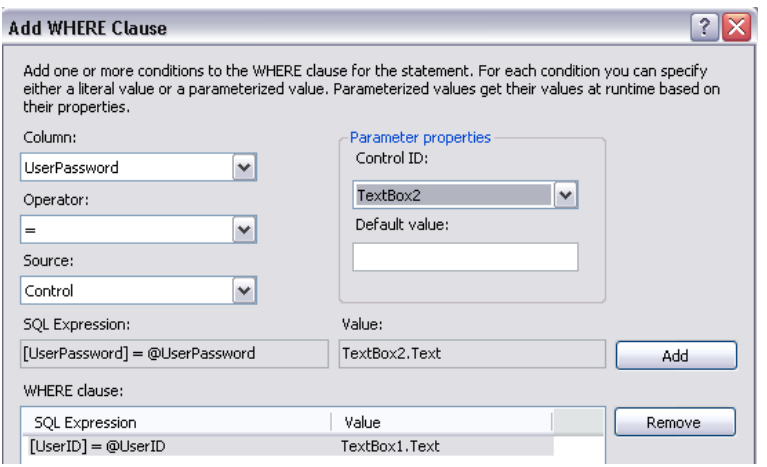


Di dalam kotak dialog yang tersedia, klik pada tombol *Where* untuk mengubah kondisi parameter yang ada di dalam SqlDataSource tersebut. Di kotak dialog berikutnya, arahkan field *UserID* sebagai field yang akan dikondisikan dan pilih tipe *Control* untuk tipe parameter dan Textbox1 sebagai parameter.





Untuk parameter yang kedua, field yang menjadi kondisi adalah *UserPassword* dan sebagai tipe parameternya tetap *Control* sedangkan komponen yang menjadi parameter adalah Textbox2.



Langkah berikutnya adalah melakukan dobel klik pada Button *Login* kemudian didalamnya ketikkan listing berikut :

```
Dim xreader As Data.IDataReader
With SqlDataSource1
    xreader = .Select _
        (DataSourceSelectArguments.Empty)
    If xreader.Read Then
        Session("Login") = _
            xreader.Item("UserName")
        Response.Redirect("admin.aspx")
    Else
        Page.RegisterStartupScript _
        ("pesan", _
        "<script>alert" & _
        "('Login Failed !')" & _
        "</script>")
    End If
End With
```

Pembacaan data login, dilakukan dengan memanfaatkan perintah SQL didalam komponen SqlDataSource yang telah dimodifikasi sebelumnya (dengan parameter UserID dan UserPassword yang diambilkan dari dua buah Textbox yang tersedia). Pembacaan data menggunakan mode DataReader yang bersifat *fast forward* dan *ReadOnly* agar pembacaan lebih cepat.

Sedangkan pada saat record yang dimaksud tidak ditemukan (berarti bahwa proses login gagal), akan ditampilkan messagebox yang

memanfaatkan Javascript yang diinjeksikan ke dalam listing.

Setelah itu, dobel klik pada LinkButton *Logout* di View yang kedua, kemudian ketikkan listing berikut :

```
Session.Clear()
Response.Redirect("default.aspx")
```

Penggunaan Session, selain sebagai penanda setelah proses login dilakukan, juga sebagai proteksi nantinya bagi halaman web administrator. Session yang digunakan dalam studi kasus ini sangat sederhana, karena selain terdiri dari satu jenis Session, juga hanya menyimpan field *UserName* dari hasil proses login.
Penggunaan Session lebih lanjut dapat dibaca di buku *Panduan Belajar ASP .NET 2.0*.

Halaman Admin

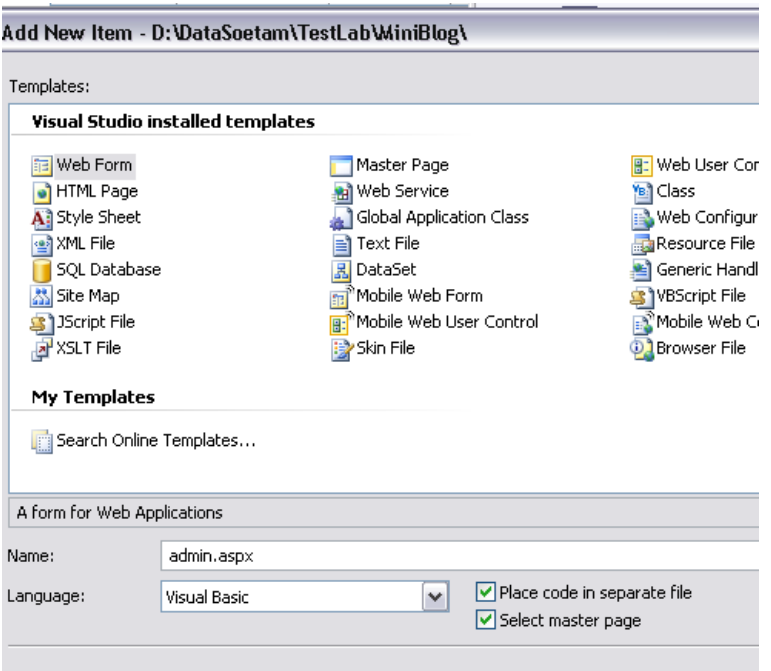
Hingga proses pembuatan Master Page selesai, situs yang dibuat masih belum bisa dieksekusi untuk melihat hasilnya. Karena sebuah Master Page membutuhkan minimal satu halaman web yang mengacu kepadanya agar bisa dieksekusi.

Halaman admin merupakan halaman yang berisi dua fitur yaitu maintenance kategori dan maintenance dari blog itu sendiri. Di dalam halaman ini akan diterapkan teknik AJAX tanpa menggunakan AJAX Control Toolkit didalamnya. Penggunaan AJAX Control Toolkit baru digunakan di halaman utama.

Halaman web untuk admin ini juga akan terproteksi sehingga tidak bisa dieksekusi langsung tanpa harus melalui proses login. Hal ini dimungkinkan dengan adanya proteksi melalui Session yang telah diterapkan sebelumnya di dalam Master Page.

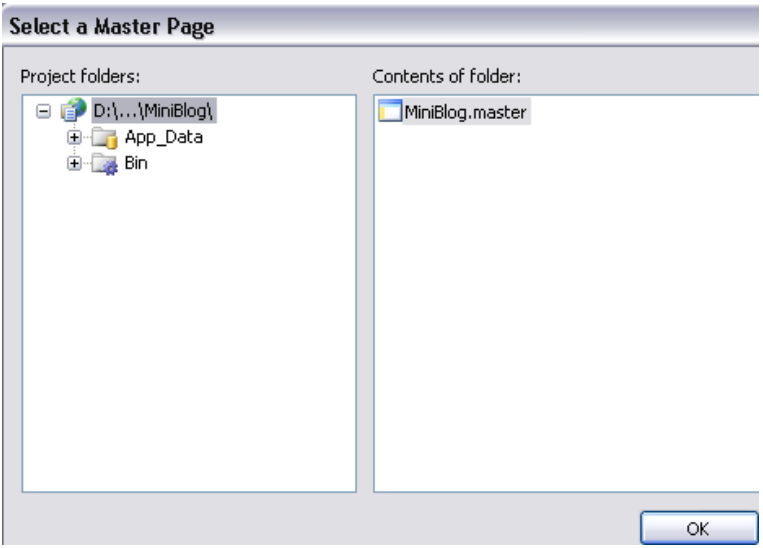
Jenis proteksi lain untuk sebuah halaman web ataupun untuk sebuah folder dapat dilakukan dengan memodifikasi file web.config dan menambahkan tag tertentu didalamnya. Hal tersebut dibahas lebih detail di buku *Panduan Belajar ASP .NET 2.0*

Langkah pertama dari pembuatan halaman admin ini adalah menambahkan halaman web baru yang mengacu ke dalam Master Page yang sudah dibuat. Klik kanan pada solution dan pilih sub menu *Add New Item*. Kemudian pada kotak dialog yang tersedia, pilihlah opsi *Select Master Page* untuk mereferensikan halaman web tersebut ke dalam sebuah Master Page.



Kemudian, di kotak dialog berikutnya, pastikan bahwa pemilihan Master Page telah mengacu ke Master yang sebelumnya telah dibuat. Dalam studi kasus ini, kebetulan hanya terdapat sebuah Master Page, sehingga

pemilihan dapat lebih mudah dilakukan. Tetapi jika pada kasus lain terdapat lebih dari satu Master Page, usahakan agar lebih berhati-hati saat melakukan pemilihan Master Page.



Setelah memberi nama halaman web tersebut dengan nama *admin.aspx*, maka didalamnya akan terlihat sebuah komponen ContentPlaceHolder yang bisa diedit, dan bagian lain yang berwarna lebih gelap, menandakan bahwa bagian tersebut adalah bagian dari Master Page yang tidak boleh diedit.

Dengan kondisi demikian, maka apabila terjadi perubahan ataupun modifikasi desain awal, perubahan atau modifikasi tersebut hanya perlu dilakukan di dalam Master Page, bukan di tiap halaman yang ada.

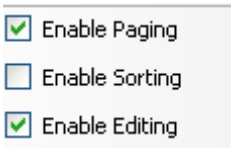
Komponen pertama yang ditempatkan di dalam ContentPlaceHolder yang tersedia adalah komponen *ScriptManager* yang nantinya dibutuhkan pada saat teknik AJAX diimplementasikan. Selanjutnya ditempatkan sebuah komponen UpdatePanel untuk mengimplementasikan proses *partial postback* di halaman tersebut.

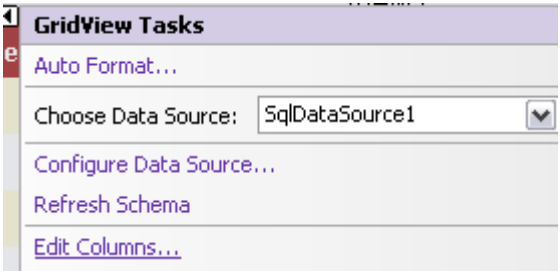
Seperti halnya pada contoh-contoh di bab sebelumnya, pastikan bahwa ScriptManager terletak di bagian paling atas dari halaman web yang akan mengimplementasikan teknik AJAX.

Di dalam UpdatePanel yang tersedia, tempatkan lagi sebuah komponen MultiView, lalu didalamnya tempatkan tiga buah komponen View yang nantinya akan digunakan untuk proses maintenance data kategori, maintenance isi blog serta menu kecil untuk berpindah antar maintenance tersebut.

Maintenance Data Kategori

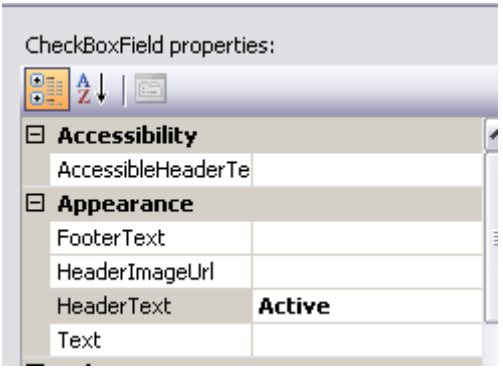
Dimulai dari View yang pertama, drag tabel *Category* dari Database Explorer ke dalamnya. Kemudian, di GridView yang tersedia, klik pada Smart Tag dan pilih sub menu *Enable Editing* agar GridView tersebut mampu melakukan edit untuk filed *StatusCategory* dan pilih juga sub menu opsi *Enable Paging*. Lalu pilih sub menu *Edit Column* untuk mengedit tampilan dari GridView tersebut.





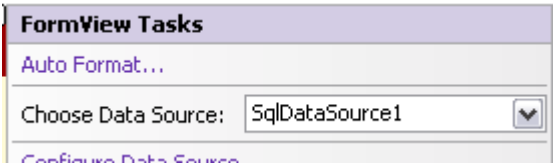
Di dalam kotak dialog yang tersedia, edit property *HeaderText* kolom *StatusCategory* menjadi *Active*. Hal ini ditujukan agar pengguna dapat lebih paham mengenai fungsi dari kolom tersebut.

Penggantian nama pada property HeaderText sangat berpengaruh dalam sebuah halaman web. Hal ini dikarenakan pengguna pada dasarnya sangat bergantung pada keterangan yang ada dalam halaman web, sedangkan pihak programmer seringkali memiliki kebiasaan untuk menganalogikan pengguna telah memiliki pengertian yang sama dengan dirinya dalam hal pemahaman desain halaman web.



Setting terakhir pada GridView adalah mengubah property *PageSize* menjadi 5. Setting property dilakukan agar pada saat tampilan data kategori lebih dari 5 record, akan menampilkan fitur *paging*, sehingga tampilan GridView tidak terlalu memanjang ke bawah.

Kemudian, tambahkan komponen FormView di di bawah GridView yang sudah diset propertynya. Klik pada Smart Tag di FormView dan arahkan ke komponen SqlDataSource yang tersedia.

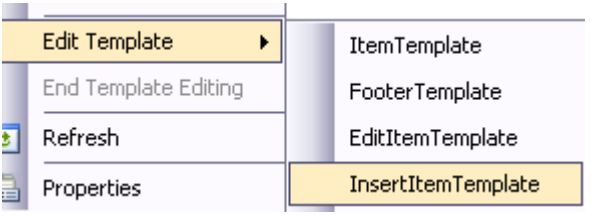


Langkah berikutnya adalah mengubah setting property *DefaultMode* dari FormView tersebut menjadi *Insert*. Hal ini dilakukan karena FormView nantinya hanya menjadi alat bantu dalam pengentrian kategori baru, sedangkan untuk proses edit menggunakan alat bantu GridView.

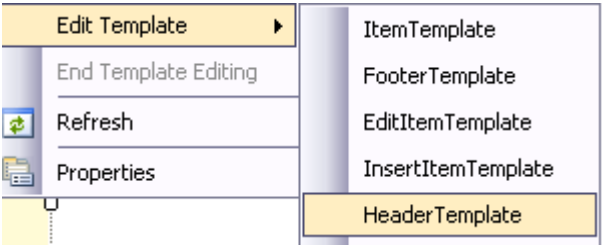
Dalam proses maintenance data kategori tidak terdapat proses penghapusan data kategori. Hal ini dilakukan untuk menghindari anomali data jika pada satu saat terjadi penghapusan data kategori, maka isi blog dalam kategori tersebut tidak memiliki *parent* kategori yang bersesuaian. Untuk "menghapus" sebuah kategori, dilakukan dengan mengedit field *StatusCategory*

menjadi *false* dan nantinya akan difilter di dalam halaman utama.

Kemudian, klik kanan pada FormView dan pilih sub menu *Edit Template → InsertItemTemplate* untuk mengedit tampilan dalam FormView. Pada mode edit template tersebut, hapus Checkbox untuk field *StatusCategory.*



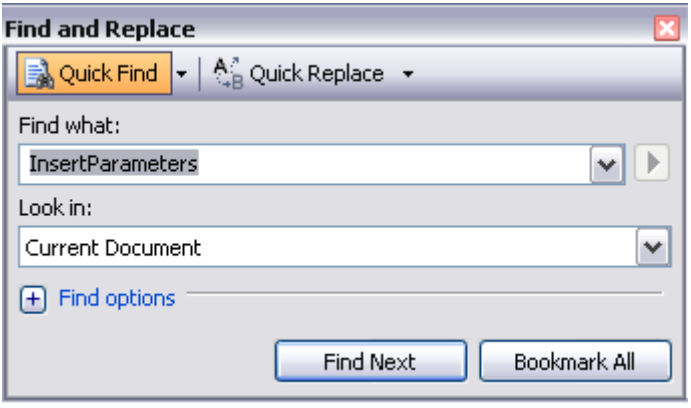
Setelah itu, klik kanan lagi pada FormView dan pilih sub menu *Edit Template → HeaderTemplate* dan beri judul pada FormView tersebut. Setelah selesai, klik pada Smart Tag untuk mengatur property *AutoFormat* sesuai dengan selera.



Berikutnya, berpindahlah ke mode Source dari halaman web admin, dan carilah tag *SelectParameter* (bisa

dengan menggunakan fasilitas *Find* atau dengan menekan kombinasi tombol Ctrl + F), lalu edit tag dalam *InsertParameters* untuk SqlDataSource1 menjadi seperti listing berikut ini :

```
<InsertParameters>
    <asp:Parameter Name="Category"
    Type="String">
    </asp:Parameter>
    <asp:Parameter Name="StatusCategory"
        Type="Boolean" DefaultValue="True">
    </asp:Parameter>
</InsertParameters>
```

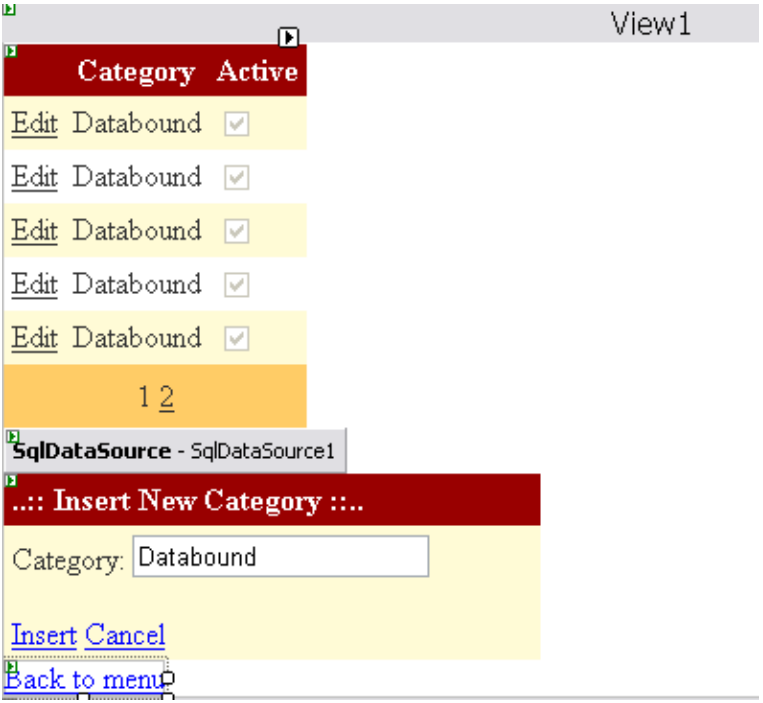


Penggantian parameter pada field *StatusCategory* dengan mengasumsikan bahwa pada saat pertama kali data kategori diinputkan, maka kategori tersebut akan langsung digunakan oleh pengguna. Sehingga status dari kategori tersebut langsung dianggap *true* atau aktif. Hal ini akan mengurangi interaksi pengguna pada saat proses maintenance data kategori, sehingga pengguna merasa lebih cepat dalam melakukan proses.

Langkah terakhir untuk View yang pertama adalah menambahkan sebuah Linkbutton di bawah dari FormView. Ganti property *text* dari Linkbutton menjadi *Back to menu*. Kemudian dobel klik pada Linkbutton tersebut untuk mengetikkan listing berikut :

```
MultiView1.ActiveViewIndex = 2
```

Linkbutton *Back to menu* berfungsi untuk mengembalikan keadaan halaman web ke kondisi awal halaman web admin. Yaitu menuju ke View yang ketiga atau yang terakhir. Pembuatan View yang terakhir akan dijelaskan pada bagian lanjutan dari studi kasus ini.



Setelah View yang pertama selesai dimodifikasi, maka langkah berikutnya adalah melakukan modifikasi

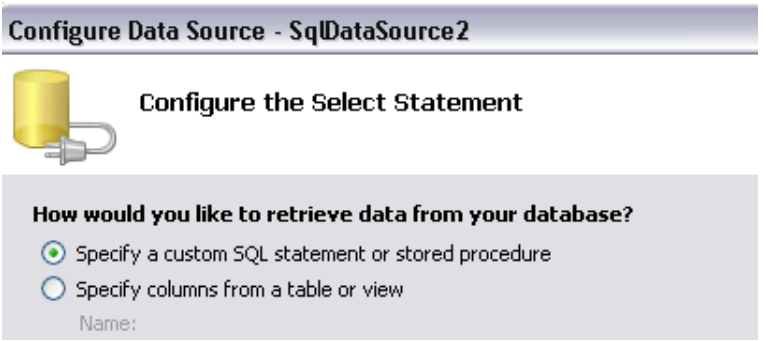
untuk View yang kedua, yaitu View untuk maintenance isi dari blog itu sendiri.

Maintenance Data Blog

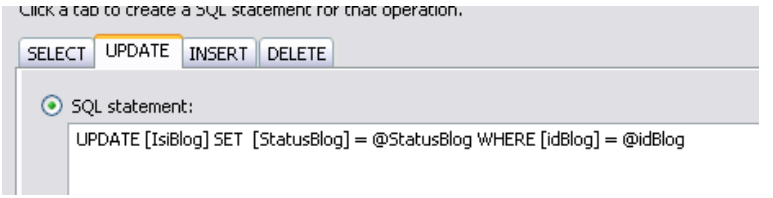
Langkah pertama untuk modifikasi View yang kedua adalah melakukan drag tabel *IsiBlog* pada View yang kedua. Dari hasil operasi drag tersebut, maka akan terdapat sebuah komponen SqlDataSource yang kedua serta GridView baru diatasnya.

Pada komponen SqlDataSource yang kedua, klik pada Smart Tag dan piilh sub menu *Configure Data Source* untuk melakukan edit pada perintah *Update* yang terdapat dalam komponen tersebut. Edit perintah Update dilakukan untuk menyesuaikan dengan modifikasi GridView yang akan dilakukan pada langkah selanjutnya.

Pada kotak dialog yang tersedia, pilih opsi *Specify a custom SQL Statement ….* Untuk melakukan editing perintah Update secara manual. Opsi ini umumnya dilakukan untuk melakukan editing perintah SQL non *Select* secara lebih interaktif.



Kemudian pada kotak dialog berikutnya, klik pada tab untuk perintah Update, dan edit perintah tersebut seperti pada gambar berikut ini.

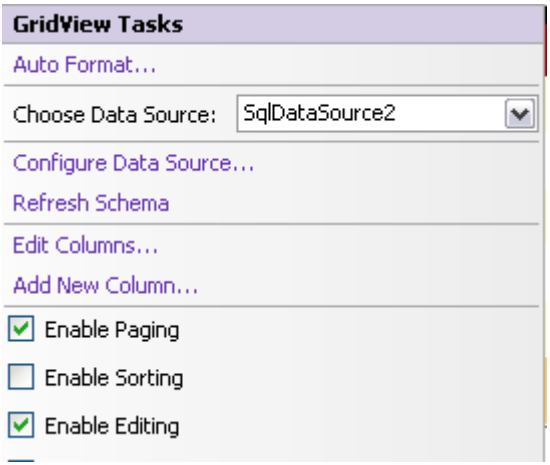


Setelah selesai, berpindahlah ke mode Source, dan cari tag *<SqlDataSource2>…</SqlDataSource2>*. Di dalam lingkup tag tersebut, cari sub tag *<UpdateParameters>..</UpdateParameters>* dan edit seperti pada listing berikut :

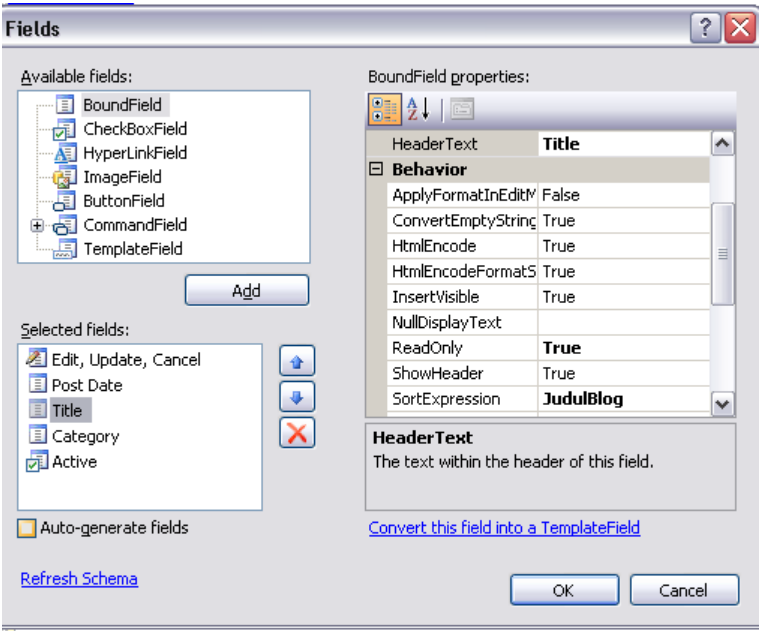
```
<UpdateParameters>
<asp:Parameter Name="StatusBlog"
   Type="Boolean"></asp:Parameter>
<asp:Parameter Name="idBlog"
   Type="Int32"></asp:Parameter>
</UpdateParameters>
```

Editing perintah SQL juga bisa langsung dilakukan pada mode Source jika Anda sudah memiliki pemahaman yang lebih baik dalam proses editing tag. Jika tidak, maka lebih baik menggunakan bantuan kotak dialog interaktif yang telah disediakan oleh Visual Web Developer Express.
Hal tersebut dikarenakan proses editing langsung di mode Source sangat rentan dan sangat beresiko untuk terjadinya sebuah kesalahan. Dan dalam sebuah halaman web yang kompleks, pencarian kesalahan di dalam mode Source seringkali menjadi sebuah proses yang menjengkelkan sekaligus melelahkan.
Disarankan juga untuk tetap memperhatikan pasangan tag dalam mode Source, karena dari pasangan tag tersebut (<…> …</…>) dapat dilacak lebih lanjut, proses modifikasi yang dilakukan oleh kotak dialog interaktif yang telah dilakukan sebelumnya.

Langkah berikutnya adalah melakukan klik pada Smart Tag yang terdapat dalam GridView, kemudian pilih opsi *Enable Paging* dan opsi *Enable Editing,* lalu klik pada sub menu *Edit Column*.



Selanjutnya, di kotak dialog hapus field *idBlog* dan *IsiBlog* dari daftar *Selected Fields.* Lalu lakukan pengeditan kolom dengan mengganti property *HeaderText* untuk kolom *TglBlog* menjadi *Post Date*, *JudulBlog* menjadi *Title* dan kolom atau field *StatusBlog* menjadi *Active.*

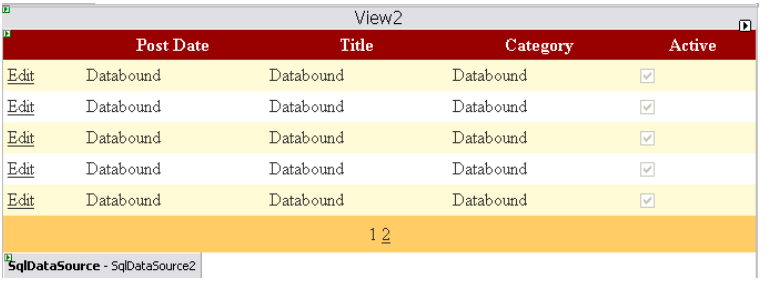


Selain itu, set juga property *ReadOnly* dari tiap field tersebut menjadi *True*, kecuali untuk field *StatusBlog* (di bagian *Selected Fields* sudah berubah menjadi *Active*).

Sama halnya dengan proses maintenance data kategori, tidak terdapat proses penghapusan pada proses maintenance isi blog. Karenanya, jika pengguna menginginkan isi blog tidak muncul lagi di dalam halaman utama, cukup dengan melakukan edit field *StatusBlog* menjadi *False* di dalam tabel *IsiBlog*.

Kemudian set property *Width* dari GridView menjadi *100%* agar GridView terlihat *stretch* atau melebar sesuai dengan lebar kolom pada tabel. Ganti juga property *PageSize* menjadi 5 agar GridView tidak terlihat terlalu scroll ke bawah jika isi blog sudah banyak.

Selanjutnya, GridView dapat diatur formatnya dengan melakukan klik pada Smart Tag dan memilih menu *AutoFormat*, lalu atur berdasarkan template sesuai selera.



Langkah berikutnya adalah menempatkan komponen FormView di bawah GridView. Sama juga dengan proses di maintenance data kategori, FormView hanya akan digunakan untuk melakukan entri data blog.

Pada FormView tersebut, nantinya akan ditempatkan sebuah komponen editor HTML yaitu FCKEditor. Komponen ini bisa didownload secara gratis (dan open source) dari www.FCKEditor.net. Pada saat proses download dilakukan, pilih komponen FCKEditor yang memang dikhususkan untuk ASP .NET (hal ini dikarenakan komponen FCKEditor

memiliki berbagai jenis, misal : FCKEditor yang dikhususkan untuk PHP).

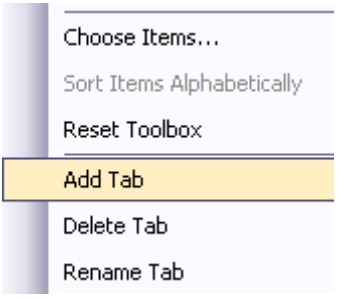
Terdapat beragam jenis komponen editor HTML yang tersedia di internet dan siap untuk didownload, baik yang gratis maupun yang komersial. Selain FCKEditor, juga terdapat komponen lain seperti FreeTextBox, TinyMCE, RTE, WYSWYG open ataupun MCE.
Dari berbagai jenis komponen tersebut, usahakan pilih komponen yang gratis dan memiliki jenis khusus untuk bahasa pemrograman yang kita gunakan. Karena dalam konteks buku ini digunakan ASP .NET 2.0, maka lebih baik jika memilih komponen yang memang memiliki kapabilitas khusus untuk digunakan dalam lingkup ASP .NET 2.0, seperti FCKEditor, FreeTextBox atau TinyMCE. Hal tersebut akan sangat memudahkan pada saat proses desain maupun proses pemrograman dilakukan.
Dalam studi kasus ini digunakan FCKEditor dengan alasan bahwa komponen ini gratis dan open source dan juga telah terbukti handal digunakan oleh portal berbasis ASP .NET lainnya yaitu DotNetNuke.

Dari hasil download komponen, maka copy file *FredCK.FCKeditorV2.dll* (umumnya terletak di dalam folder Bin di hasil download komponen) ke dalam folder *bin* di

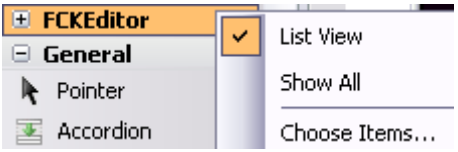
dalam solution yang kita miliki (bisa dengan cara drag & drop dari Windows Explorer, atau dengan melakukan klik kanan pada folder *Bin* dan memilih sub menu *Add Existing Item*).

Kemudian, tambahkan pula satu folder *FCKeditor* ke dalam solution (dengan menggunakan cara yang sama seperti menambahkan file dll). Di dalam folder ini nantinya akan terdapat script serta setting dari FCKEditor yang akan digunakan.

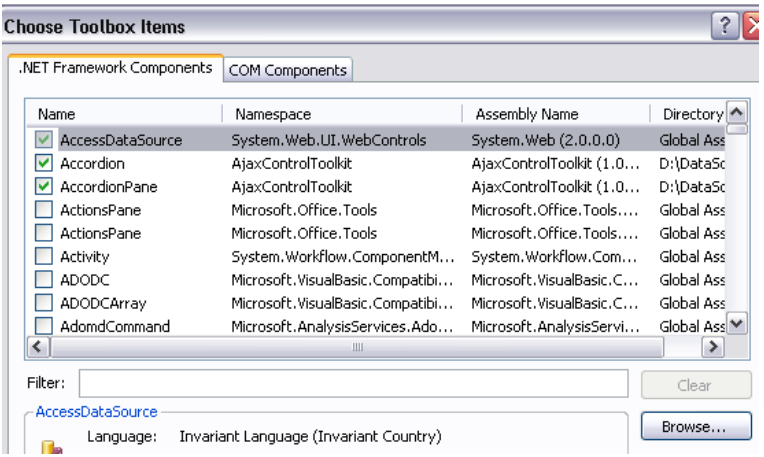
Langkah terakhir sebelum menggunakan FCKEditor adalah melakukan penambahan komponen ke dalam Toolbox yang kita miliki. Pada Toolbox, klik kanan di tab yang paling bawah (dan kosong), lalu pilih sub menu *Add Tab* dan beri nama tab tersebut *FCKEditor.*



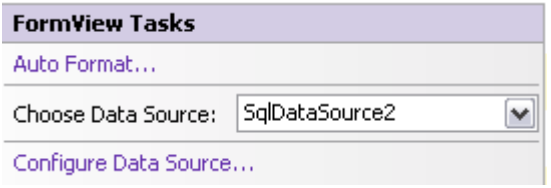
Selanjutnya, klik kanan di tab baru tersebut dan kini pilih sub menu *Choose Item* untuk menambahkan komponen baru di dalam tab tersebut.



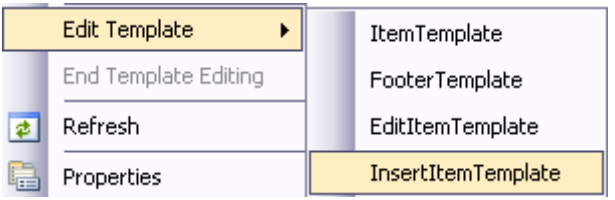
Pada kotak dialog yang tersedia, klik pada button *Browse* dan arahkan pada folder tempat *Bin* di solution lalu pilih file *FredCK.FCKeditorV2.dll*. Hasil dari penambahan ini akan menyebabkan Toolbox terisi komponen baru yaitu *FCKEditor.*



Setelah FormView ditempatkan, klik pada Smart Tag dan set property *DataSource* ke komponen SqlDataSource yang kedua. Lalu set property *DefaultMode* dari komponen FormView tersebut menjadi *Insert*.



Kemudian, sama dengan langkah pada maintenance data kategori, klik kanan pada FormView dan pilih sub menu *Edit Template → Insert Item Template*. Dan kini saatnya untuk melakukan editing dari template FormView.



Dalam template yang tersedia, edit template, dengan menghilangkan field *TglBlog, Category, IsiBlog* dan *StatusBlog*, serta menghilangkan keterangan untuk *JudulBlog*. Selanjutnya, drag komponen FCKEditor ke bawah bagian *JudulBlog*.

Lakukan penghapusan field dengan hati-hati, agar komponen yang lain tidak ikut terhapus.

Berikutnya, klik pada komponen FCKEditor dan set property *BasePath* seperti pada gambar berikut ini.

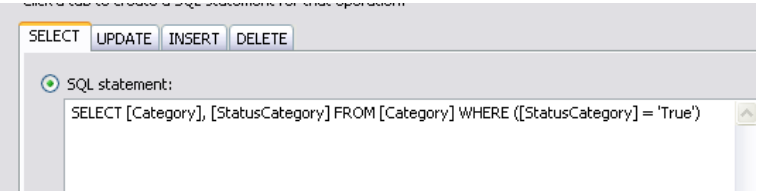
(Expressions)	
(ID)	**FCKeditor1**
BasePath	~/**FCKeditor**/

Setting property ini untuk memastikan bahwa komponen FCKEditor akan mengacu pada folder yang telah dicopykan sebelumnya.

Kemudian, di bagian atas dari FormView tersebut, tambahkan keterangan untuk field *Category*, dengan cara melakukan drag tabel *Category* ke dalam FormView. Setela itu, GridView yang terbentuk dari hasil drag tersebut bisa dihapus, sehingga menyisakan komponen SqlDataSource yang ketiga dalam halaman web admin.

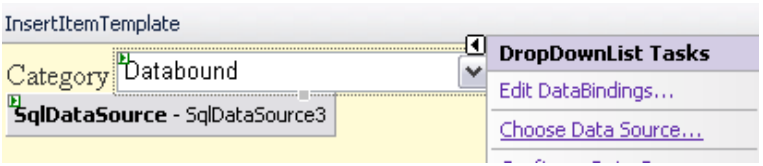
Penempatan SqlDataSource hasil drag tabel *Category* bertujuan untuk mengambil data kategori ke dalam FormView dalam bentuk DropDownList.

Kini, klik pada Smart Tag dari SqlDataSource dan pilih sub menu *Configure Data Source* untuk melakukan edit perintah SQL. Di dalam kotak dialog yang tersedia, klik opsi *Specify a custom SQL…* dan modifikasi perintah *Select* seperti pada rangkaian gambar berikut.

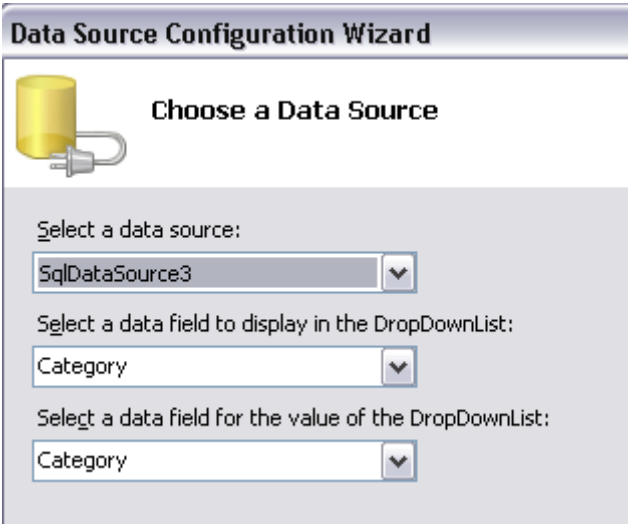


Kemudian, tempatkan sebuah DropDownList di atas SqlDataSource yang telah tersedia untuk menampung data

kategori tersebut. Di dalam DropDownList yang ada, klik pada Smart Tag dan pilih sub menu *Choose Data Source* untuk melakukan setting koneksi data dalam DropDownList tersebut.



Di kotak dialog yang tersedia, set property *DataSource* ke komponen SqlDataSource3, dan set property dibawahnya untuk mengacu ke field *Category*.



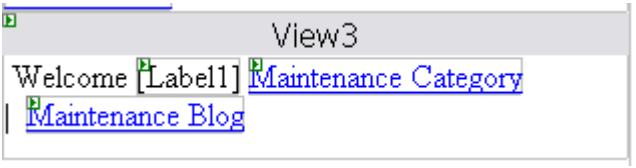
Setelah proses editing template selesai dilakukan, dobel klik pada FormView, dan beralihlah ke method *FormView_ItemInserting* lalu ketikkan listing berikut ini :

```
With SqlDataSource2
     .InsertParameters("Category").DefaultValue = _
   CType(FormView2.FindControl("DropDownList1"), _
    DropDownList).SelectedValue
     .InsertParameters("IsiBlog").DefaultValue = _
     CType(FormView2.FindControl("FCKeditor1"), _
     FredCK.FCKeditorV2.FCKeditor).Value
End With
```

Setting parameter *Insert* di prosedur tersebut ditujukan untuk mengalihkan parameter asli yang sebelumnya telah dihapus. Jika Anda tidak menempatkan FCKEditor, dan tetap menggunakan Textbox default sebagai isian dari FormView, maka parameter yang kedua yaitu FCKEditor bisa diabaikan.

Menu Halaman Web Admin

View yang terakhir atau View yang ketiga merupakan yang paling sederhana dibandingkan kedua View sebelumnya. Pada View ini hanya terdapat sebuah komponen Label dan dua buah komponen Linkbutton yang berfungsi sebagai navigator dalam halaman web admin.



Komponen Label dalam View berfungsi sebagai penanda session hasil login yang telah dilakukan. Hal ini umumnya dilakukan sebagai salam pembuka dari sebuah halaman web yang baru saja melalui proses login, sekaligus sebagai proses cek dalam halaman tersebut, apakah login telah sesuai dengan yang diharapkan.

Dobel klik pada Linkbutton *Maintenance Category* dan ketikkan listing berikut agar menuju ke dalam View yang pertama :

```
MultiView1.ActiveViewIndex = 0
```

Kemudian klik pada Linkbutton *Maintenance Blog* untuk mengetikkan listing berikut :

```
MultiView1.ActiveViewIndex = 1
```

Langkah terakhir dari View ketiga ini adalah melakukan dobel klik pada bagian halaman web yang kosong untuk menuju ke prosedur *Page_Load* dan mengetikkan listing berikut kedalamnya :

```
If Session("Login") = "" Then
    Response.Redirect("default.aspx")
Else
    CType(Master.FindControl("MultiView1"), _
        MultiView).ActiveViewIndex = 1
    Label1.Text = Session("Login")
End If
```

Pengecekan login dilakukan pada saat halaman web admin pertama kali dieksekusi, hal ini agar halaman web admin tidak dapat diakses secara langsung tanpa melalui proses login.
Sedangkan saat proses login berhasil dilakukan, maka MultiView yang terdapat di Master Page akan dialihkan ke View yang kedua untuk mengganti menu sebelumnya.

Halaman Default

Halaman web default, merupakan halaman web utama dalam studi kasus mini blog ini. Halaman web default, sudah muncul pertama kali pada saat solution dibuat (lihat lagi di langkah pertama pada studi kasus ini). Tetapi, halaman web utama tersebut belum terhubung ke Master Page yang telah dibuat sebelumnya.

Untuk menghubungkan sebuah halaman web ke dalam Master Page, maka perlu dilakukan modifikasi di dalam mode Source. Modifikasi pertama (setelah berada dalam mode Source) adalah menghapus seluruh bagian di dalam tag *<html>…</html>*.

Penghapusan seluruh isi bagian dari tag HTML tersebut dilakukan karena, dalam koneksi antara halaman web biasa dengan Master Page, tag HTML akan ditangani oleh Master Page, bukan oleh halaman web.

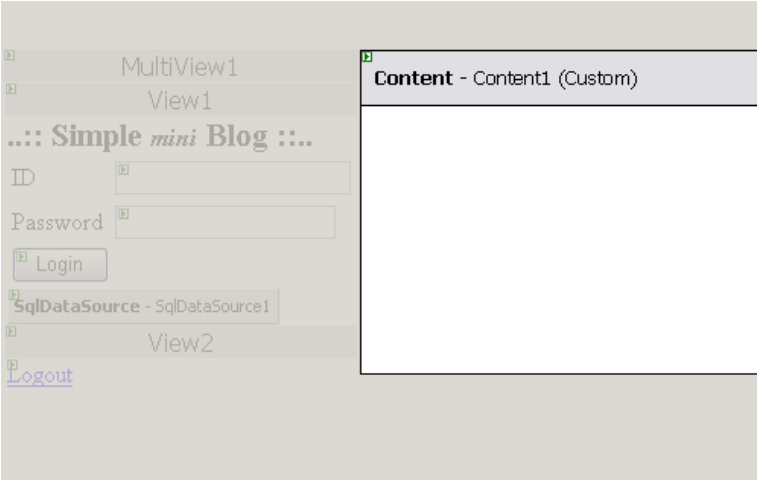
Setelah bagian tag HTML dihapus, modifikasilah bagian tag paling atas dalam mode Source di halaman web tersebut. Modifikasi yang dilakukan adalah menambahkan atribut *MasterPageFile* dan mengisinya dengan nama Master Page yang akan diacu (referensi). Modifikasi dalam halaman web tersebut akan terlihat pada listing berikut :

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb" Inherits="Default"
MasterPageFile="~/MiniBlog.master"  %>
```

Berikutnya, tempatkan sebuah komponen *Content* di dalam halaman web tersebut, sebagai penampung dari halaman web tersebut ke dalam Master Page. Penambahan tersebut dilakukan dengan cara mengetikkan listing berikut ini di bagian bawah dari mode Source :

```
<asp:Content
ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
</asp:Content>
```

Jika modifikasi sudah dilakukan dengan benar, maka saat berpindah lagi ke mode Design, halaman web akan tampak seperti pada gambar ini.



Di dalam halaman web utama atau halaman default ini nantinya akan diletakkan dua buah komponen *Collapsible Panel* dari AJAX Control Toolkit sebagai navigasi utama yang menangani data blog terbaru serta data arsip dari blog berdasarkan kategori.

Di tiap isi panel dari Collapsible Panel, akan ditempatkan lagi komponen Accordion (juga dari AJAX Control Toolkit), sebagai ganti dari GridView untuk menampilkan detail dari isi blog.

Khusus untuk panel yang kedua, juga akan melibatkan komponen *DropDown Extender* (yang didalamnya terdiri dari sebuah Textbox dan Listbox) dari AJAX Control Toolkit sebagai ganti dari DropDownList biasa, dalam rangka pencarian arsip blog berdasarkan kategori yang ada.

Langkah pertama yang dilakukan dalam halaman default ini adalah melakukan drag komponen ScriptManager ke bagian paling atas dari ContentPlaceHolder yang tersedia. Seperti contoh sebelumnya, komponen ini wajib ditempatkan di sebuah halaman web yang didalamnya akan menerapkan teknik AJAX.

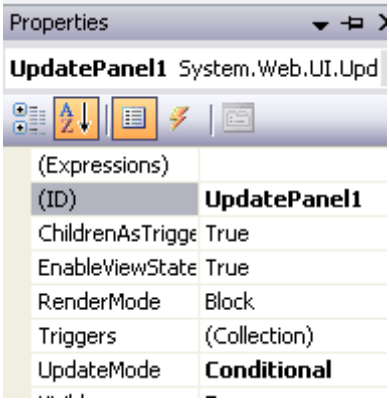
Kemudian, dobel klik di bagian kosong dari ContentPlaceHolder yang tersedia, dan pada prosedur *Page_Load* ketikkan listing berikut ini :

```
CType(Master.FindControl("MultiView1"), _
    MultiView).ActiveViewIndex = 0
```

Listing tersebut berfungsi sebagai validasi saat pengguna beralih ke halaman utama, terutama jika pengguna baru saja beralih dari halaman web admin. Validasi yang dilakukan adalah mengganti posisi View dari MultiView yang terdapat dalam Master Page agar berpindah ke View yang pertama (posisi awal).

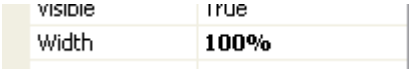
Bagian Newest Blog

Langkah pertama dalam bagian Newest Blog di halaman default adalah menempatkan komponen UpdatePanel dalam ContentPlaceHolder yang tersedia. Lalu set property *UpdateMode* dari UpdatePanel ini menjadi *Conditional* agar tidak selalu melakukan refresh, terutama pada saat bagian archieve nanti selesai dimodifikasi.



Berikutnya, drag sebuah komponen Collapsible Panel Extender dari tab AJAX Control Toolkit. Dalam komponen ini nantinya akan diset lebih lanjut propertynya di dalam mode Source.

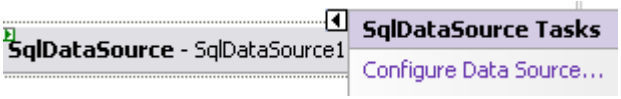
Tepat, di bawah komponen Collapsible Panel yang baru ditempatkan, drag sebuah komponen Panel. Lalu set property *Width* dari komponen Panel tersebut menjadi *100%* agar komponen Panel nantinya *stretch* (melebar ke samping sesuai dengan resolusi browser) pada saat dieksekusi.



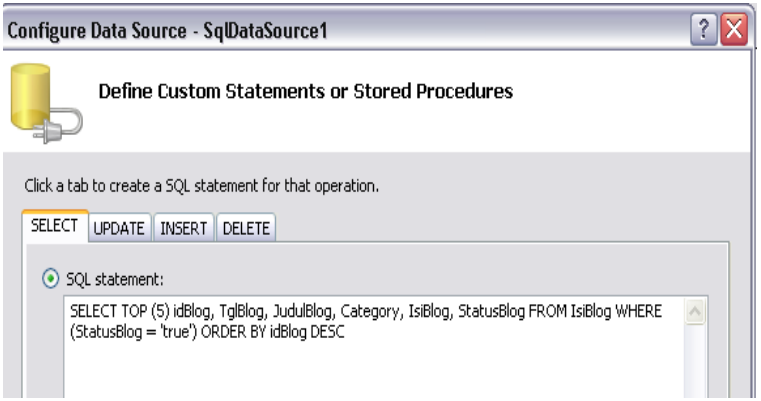
Sangat disarankan untuk melakukan setting property *Width* dalam satuan persen, dibandingkan satuan pixel. Hal ini dimaksudkan agar komponen yang akan di*stretch* tidak terpengaruh oleh resolusi layar dari pengguna.
Kecuali jika komponen yang dimaksud, memang akan diberikan sebuah ukuran lebar yang statis, maka property Width harus diset dengan menggunakan satuan pixel.
Selain itu, juga harus dipertimbangkan konsep dari desain halaman web itu sendiri, apakah akan menggunakan ukuran relatif (persentase) atau menggunakan ukuran absolut (pixel). Karenanya, jika dalam sebuah pembuatan situs, melibatkan dua orang atau lebih yang masing-masing bertugas sebagai programmer dan desainer, diperlukan komunikasi yang baik agar situs dapat terlihat sesuai rencana awal.

Kemudian, drag tabel *IsiBlog* ke dalam Panel tersebut, dan jangan lupa untuk menghapus GridView hasil dari proses drag dari Panel. Penghapusan ini dilakukan karena hasil koneksi database akan diarahkan ke dalam komponen Accordion.

Klik pada Smart Tag dari komponen SqlDataSource yang tersedia, dan pilih menu *Configure Data Source* untuk mengganti perintah SQL yang akan menampilkan lima data blog terbaru pada komponen Accordion.



Di dalam kotak dialog yang tersedia, pilih opsi *Specify a custom SQL…..*, lalu ketikkan perintah SQL berikut ini :



Parameter *Top 5* dalam perintah SQL tersebut berarti bahwa hanya akan ada lima record teratas yang akan ditampilkan dari hasil perintah Select tersebut. Lima record teratas tersebut adalah record yang memenuhi kriteria field *StatusBlog* –nya true dan telah diurutkan secara terbalik (descending) dari field *idBlog*.
Sehingga perintah Select yang telah dimodifikasi tersebut nantinya akan menampilkan lima isian blog terbaru dari tabel *IsiBlog*, dan dianggap dalam kategori *Newest Blog*.

Setelah kotak dialog selesai prosesnya, maka berpindahlah ke dalam mode Source untuk memodifikasi lebih lanjut bagian dari Update Panel yang pertama. Pada mode Source ini, bagian yang pertama kali adalah tag dari komponen Collapsible Panel. Modifikasi dari tag Collapsible Panel terlihat pada listing berikut :

```
<cc1:CollapsiblePanelExtender
ID="CollapsiblePanelExtender1"
runat="server" TargetControlID="Panel1"
ExpandControlID="LinkButton1"
CollapseControlID="LinkButton1">
</cc1:CollapsiblePanelExtender>
```

Sebuah Collapsible Panel membutuhkan minimal sebuah Panel sebagai target dan sebuah komponen pemicu (umumnya Linkbutton atau Button) dalam implementasinya.
Di dalam Collapsible Panel tersebut, yang akan menjadi target komponen yang dituju adalah Panel1, sedangkan pemicu dari proses ekspansi Panel adalah komponen Linkbutton1 (yang akan dibuat dan diterangkan selanjutnya). Karena property *ExpandControlID* dan *CollapseControlID* diset ke komponen yang sama, maka Linkbutton1 nantinya akan berfungsi ganda, sebagai pemicu untuk ekspansi Panel sekaligus sebagai pemicu untuk penutupan Panel itu sendiri.

Selanjutnya, masih dalam mode Source, ketikkan listing berikut tepat di bawah bagian Collapsible Panel :

```
<div style="background-color:Yellow;
border:solid 1px red;
font-weight:bold;
font-size:14pt;
cursor:pointer ">
    <asp:LinkButton ID="LinkButton1"
        runat="server">
        Newest Blog
    </asp:LinkButton>
</div>
```

Listing tag tersebut merupakan sebuah layer dalam HTML yang didalamnya terdapat sebuah komponen Linkbutton yang merupakan pemicu untuk proses ekspansi dan penutupan dari Panel di dalam Collapsible Panel.
Di dalam layer tersebut, diset stylenya dengan menggunakan property CSS border (warna dan style bisa diganti sesuai dengan selera). Dan juga diset property cursor menjadi pointer, sehingga pengguna dapat mengetahui, bahwa pada saat kursor moruse melintas di bagian layer tersebut, berarti bagian layer tersebut merupakan bagian *clickable* (dapat diklik).
Proses pembuatan layer tersebut juga dapat menggunakan file CSS (Style Sheet) yang terpisah dan dibuat class CSS tersendiri didalamnya.

Di dalam Collapsible Panel tersebut, yang akan menjadi target komponen yang dituju adalah Panel1, sedangkan pemicu dari proses ekspansi Panel adalah komponen Linkbutton1 (yang akan dibuat dan diterangkan selanjutnya). Karena property *ExpandControlID* dan *CollapseControlID* diset ke komponen yang sama, maka Linkbutton1 nantinya akan berfungsi ganda, sebagai pemicu untuk ekspansi Panel sekaligus sebagai pemicu untuk penutupan Panel itu sendiri.

Berikutnya, adalah pembuatan bagian yang didalamnya menggunakan komponen Accordion. Seperti telah dicontohkan pada bagian sebelumnya mengenai penggunaan Accordion dengan melakukan koneksi ke sebuah database, maka penggunaan Accordion dalam konteks ini akan melibatkan bagian tag *HeaderTemplate* dan *ContentTemplate* untuk menempatkan field-field yang akan ditampilkan di bagian *Newest Blog*.

Listing lengkap berikut ini untuk penggunaan Accordion diketikkan tepat di bawah bagian layer yang telah diketikkan sebelumnya.

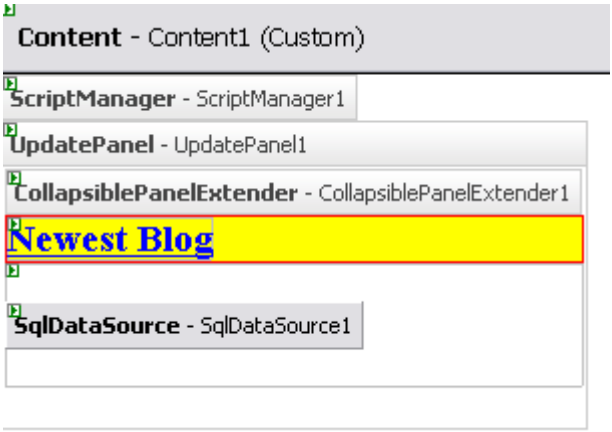
```
<cc1:Accordion ID="Accordion1" runat="server"
DataSourceID="SqlDataSource1">
<HeaderTemplate>
<h2><b>
    <div
    style="border-bottom:dashed 1px black;
    cursor:pointer; background-color:Aqua">
        <%#eval("JudulBlog") %>
     </div>
 </b></h2>
</HeaderTemplate>
<ContentTemplate>
<b>
Posted at :
    <%#format(Eval("TglBlog"),"dd-MM-yyyy")%>
    in <%#eval("Category") %>
    </b><br />
```

```
    <%#Eval("IsiBlog")%>
</ContentTemplate>
</cc1:Accordion>
```

Di dalam koneksi database ke dalam komponen Accordion, modifikasi utama terletak pada property *DataSourceID* yang diisi dengan komponen SqlDataSource hasil dari drag and drop tabel yang akan dikoneksikan dari Database Explorer.

Sedangkan bagian tag Header Template diisikan dengan field yang
akan dijadikan judul. Dalam kasus ini, field yang akan dijadikan judul adalah field *JudulBlog*. Di bagian tag ini juga ditempatkan sebuah layer baru yang berfungsi sebagai penanda header. Style dari layer tersebut bisa diganti sesuai dengan selera, dengan asumsi property *cursor* tetap dinyatakan dengan nilai *pointer* agar pada saat pengguna mengarahkan mouse di atas judul dapat langsung mengetahui bahwa bagian tersebut dapat diklik untuk mengetahui isi dari blog.

Untuk bagian Content Template, diisikan dengan isi dari blog itu sendiri, yaitu field *TglBlog* yang digabungkan dengan field *Category* dan kemudian dibawahnya (dipisah dengan tag HTML *
* untuk ganti baris) diisikan dengan field *IsiBlog*.


Content - Content1 (Custom)
ScriptManager - ScriptManager1
UpdatePanel - UpdatePanel1
CollapsiblePanelExtender - CollapsiblePanelExtender1
Newest Blog
SqlDataSource - SqlDataSource1

Bagian Archieve Blog

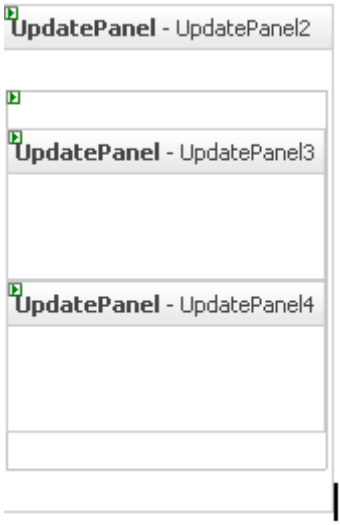
Bagian Archieve Blog merupakan bagian terakhir dari pembuatan situs Mini Blog. Bagian ini masih merupakan satu kesatuan halaman web default, yang salah satu bagiannya telah selesai dibahas yaitu bagian *Newest Blog*.

Bagian archive blog mirip dengan bagian *Newest Blog* yaitu memanfaatkan komponen Collapsible Panel dari AJAX Control Toolkit, tetapi didalamnya terdiri dari sebuah komponen DropDown Extender dan Grid View yang juga memanfaatkan Collapsible Panel didalamnya.

Umumnya penempatan tiap komponen AJAX Control Toolkit membutuhkan sebuah Update Panel sebagai "wadah" dari komponen tersebut untuk melakukan proses partial postback.

Untuk pembuatan bagian ini, langkah pertama yang harus dilakukan adalah menempatkan Update Panel tambahan yang pertama di bagian bawah dari Newest Blog. Di dalam Update Panel tersebut, letakkan sebuah Panel yang didalamnya ditempatkan lagi dua buah Update

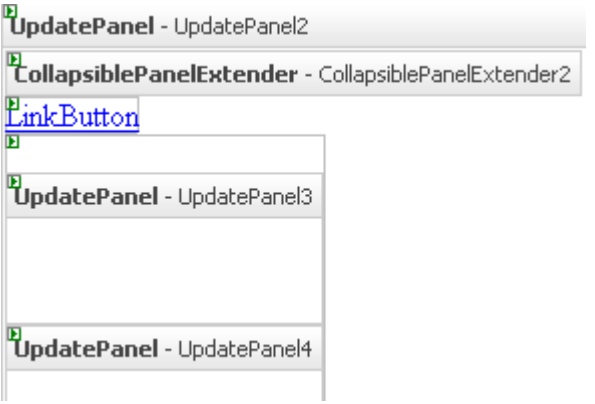
Panel tambahan, yang masing-masing akan ditempati Drop Down Extender dan Grid View.



Perhatikan penempatan UpdatePanel3 dan UpdatePanel4 yang keduanya berada dalam sebuah Panel baru. Panel ini nantinya akan menjadi pemicu bagi komponen Collapsible Panel berikutnya.

Selanjutnya, tempatkan satu komponen Collapsible Panel di dalam UpdatePanel2 tepat di atas bagian panel

yang sudah ditempatkan sebelumnya. Dan drag sebuah Linkbutton di bawah Collapsible Panel tersebut.

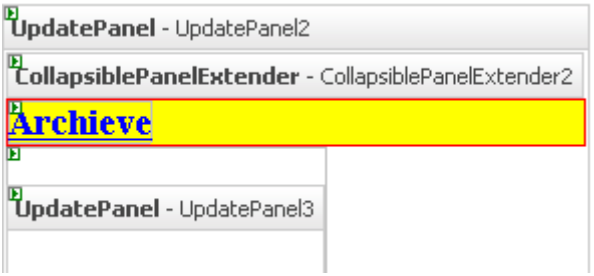


Lalu, berpindahlah ke mode Source dan edit bagian dari Collapsibel Panel dan Linkbutton menjadi seperti pada listing berikut ini :

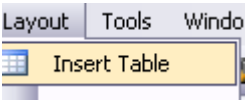
```
<cc1:CollapsiblePanelExtender
    ID="CollapsiblePanelExtender2"
    runat="server"
    TargetControlID="Panel2"
    ExpandControlID="LinkButton2"
    CollapseControlID="LinkButton2"
     Collapsed="true" SuppressPostBack="true">
</cc1:CollapsiblePanelExtender>
<div style="background-color:Yellow;
     border:solid 1px red; font-weight:bold;
     font-size:14pt; cursor:pointer ">
    <asp:LinkButton ID="LinkButton2"
        runat="server" >
        Archieve
```

```
        </asp:LinkButton>
    </div>
```

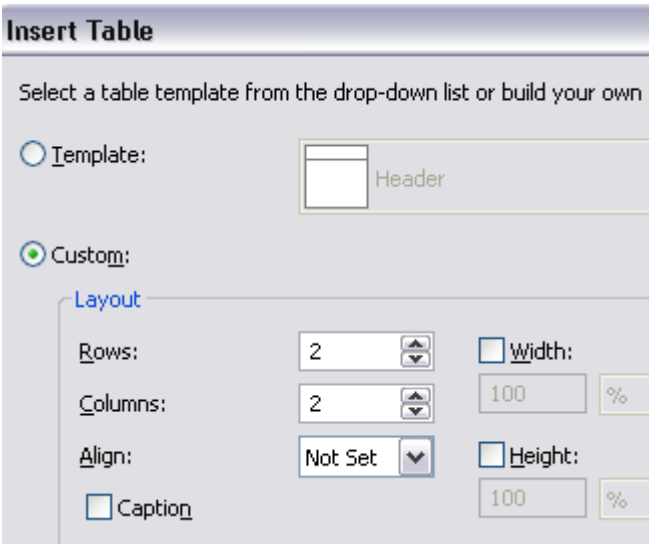
Modifikasi dari komponen Collapsible Panel untuk mengarahkan Panel target ke Panel2 dan pemicu ke arah Linkbutton yang telah dimodifikasi stylenya dengan menggunakan CSS.



Langkah selanjutnya adalah membuat sebuah tabel di dalam UpdatePanel3 yang terdiri dari dua baris dan dua kolom. Untuk pembuatan tabel tersebut, letakkan cursor di dalam UpdatePanel3 dan pilih menu *Layout→ Insert Table*.

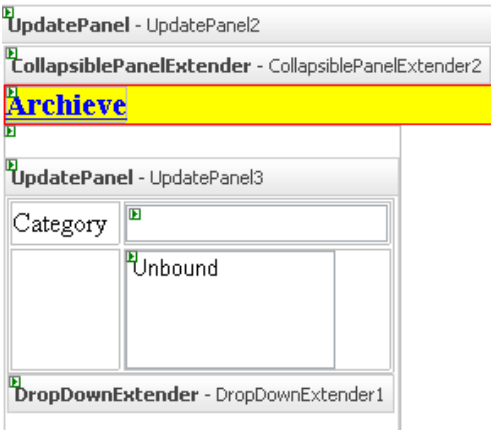


Kemudian atur jumlah *Rows* dan *Columns* masing-masing menjadi dua.

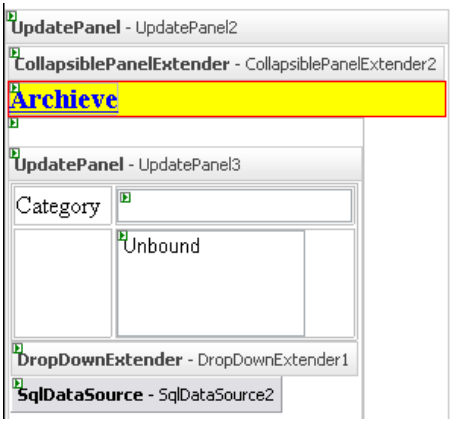


Berikutnya, tempatkan sebuah Textbox di baris pertama kolom kedua. Dan sebuah komponen Panel di baris kedua kolom pertama. Lalu, di dalam Panel tersebut ditempatkan sebuah Listbox. Textbox dan Listbox ini nantinya akan menjadi bagian dari Drop Down Extender yang ditempatkan tepat dibawah bagian tabel tersebut.

Lihat lagi contoh penggunaan Drop Down Extender di sub bab sebelumnya. Sebuah Drop Down Extender membutuhkan sebuah Textbox dan komponen berbentuk list dalam implementasinya.

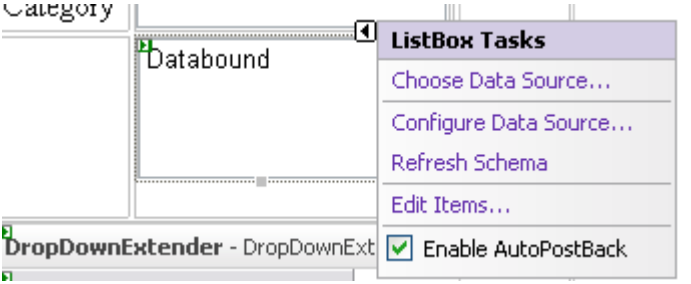


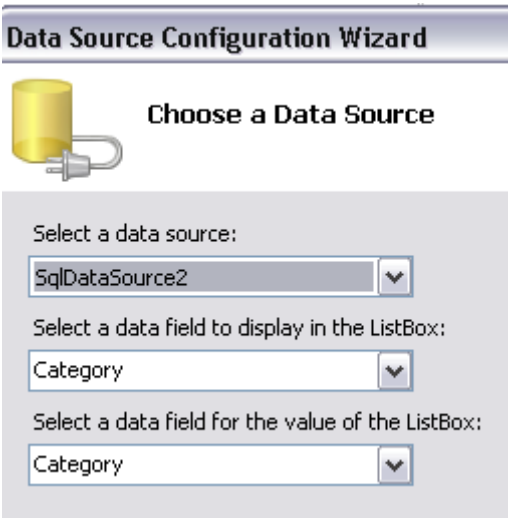
Lalu, drag tabel *Category* di bawah Drop Down Extender. Dari hasil operasi drag tersebut, hapus Grid View yang terbentuk, sehingga hanya tersisa komponen SqlDataSource yang akan digunakan sebagai data source bagi Listbox yang ada.



Berikutnya adalah melakukan setting terhadap Listbox yang sudah diletakkan. Klik pada Smart Tag di

Listbox tersebut dan pilih opsi *Enable Auto Postback* didalamnya. Setelah selesai, klik pada sub menu *Choose Data Source* untuk mengarahkan data ke dalam Listbox tersebut.



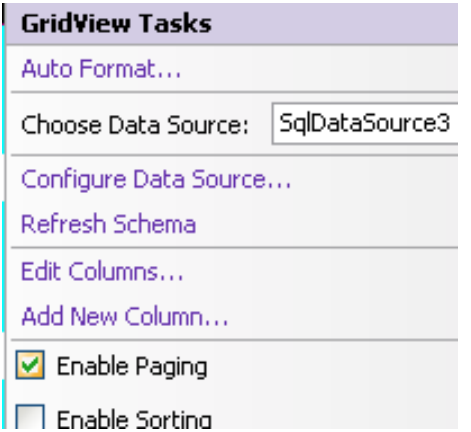


Kemudian, berpindahlah ke mode Source lagi dan modifikasi listing dari Drop Down Extender tersebut menjadi seperti pada listing berikut ini :

```
<cc1:DropDownExtender
        ID="DropDownExtender1" runat="server"
       TargetControlID="TextBox1"
       DropDownControlID ="Panel5">
    </cc1:DropDownExtender>
```

Modifikasi utama dari bagian ini adalah pada di property *TargetControlId* dan property *DropDownControlID* yang masing-masing diarahkan ke dalam Textbox dan Panel.
Asumsi dari Panel tersebut adalah panel yang terdapat tabel baris kedua kolom pertama memiliki property id *Panel5*. Jika ternyata Panel tersebut memiliki id yang berbeda, maka isi property dari *DropDownControlID* juga ikut menyesuaikan.

Kemudian drag tabel *IsiBlog* ke dalam UpdatePanel4, dan dari hasil drag tersebut, klik pada Smart Tag di Gridview yang terbentuk, lalu pilih opsi *Enable Paging* untuk Gridview tersebut. Lalu, set property *PageSize* dari Gridview tersebut menjadi *5* dan property *Width* menjadi *100%*.



Berikutnya, dobel klik pada Listbox (di dalam UpdatePanel3) , dan ketikkan listing berikut ini pada prosedur *ListBox1_SelectedIndexChanged*:

```
TextBox1.Text = ListBox1.SelectedValue
SqlDataSource3.DataBind()
```

Listing di dalam prosedur *ListBox1_SelectedIndexChanged* merupakan listing yang akan berfungsi sebagai pemicu aksi refresh di dalam Gridview saat Drop Down Extender dieksekusi.
Didalamnya memerintahkan agar isi dari Listbox yang diklik oleh pengguna diumpankan ke Textbox sekaligus melakukan refresh terhadap komponen SqlDataSource3 yang nantinya juga akan melakukan refresh ke komponen Gridview secara otomatis.

Selanjutnya, berpindahlah ke mode Source dan cari tag yang mewakili SqlDataSource3, untuk kemudian dimodifikasi seperti pada listing berikut ini :

```
<asp:SqlDataSource
    ID="SqlDataSource3" runat="server"
     ConnectionString=
 "<%$ ConnectionStrings:MiniBlogConnectionString1 %>"
     ProviderName=
"<%$
ConnectionStrings:MiniBlogConnectionString1.ProviderNam
e %>"
     SelectCommand=
     "SELECT * FROM [IsiBlog] WHERE (([Category] =
@Category) AND ([StatusBlog] = @StatusBlog)) ORDER
BY [idBlog] DESC">
```

```
    <SelectParameters>
        <asp:ControlParameter
        ControlID="TextBox1"
        Name="Category" PropertyName="Text"
            Type="String" />
        <asp:Parameter DefaultValue="True"
        Name="StatusBlog" Type="Boolean" />
    </SelectParameters>
</asp:SqlDataSource>
```

Modifikasi utama dari bagian tag SqlDataSource3 terletak pada perubahan perintah Select. Karena komponen SqlDatassource3 terletak di dalam sebuah Update Panel, maka modifikasi secara visual (dengan menggunakan wizard atauk kotak dialog) akan lebih sulit dilakukan, sebab komponen yang berada dalam sebuah Update Panel akan menjadi "terisolasi". Karenanya, untuk melakukan modifikasi terhadap komponen SqlDataSource3 tersebut, dilakukan secara manual dari mode Source.

Modifikasi lainnya terletak pada pembuatan parameter baru untuk field *Category* yang diisi dari Textbox (hasil refresh dari komponen Drop Down extender, dan field *StatusBlog* yang secara otomatis diisi dengan nilai *True*. Pengisian parameter di field *StatusBlog* dimaksudkan agar hanya isi blog yang diberi status *True* yang akan ditampilkan di bagian ini.

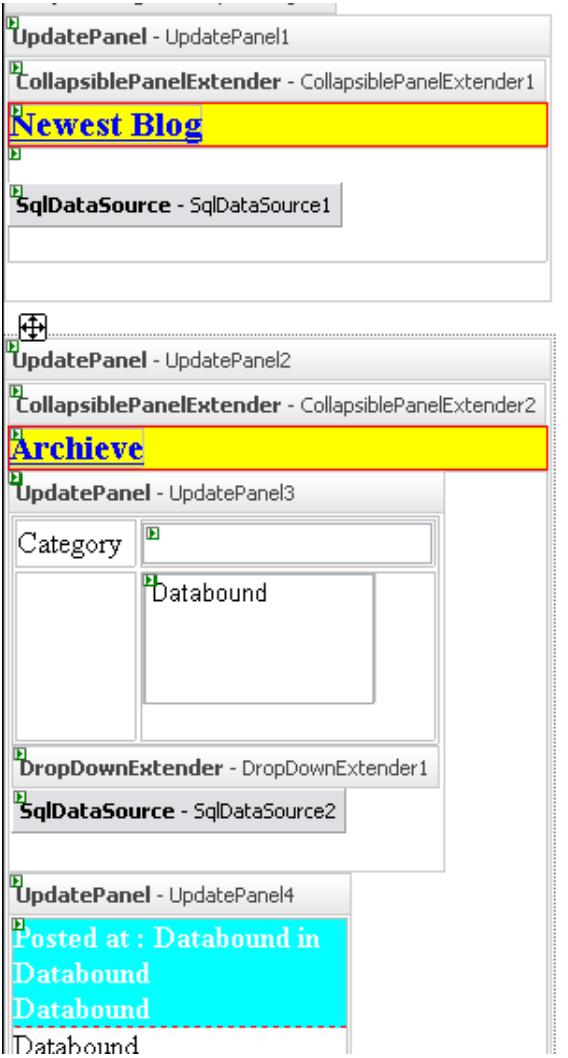
Langkah terakhir (masih tetap dalam mode Source) adalah melakukan modifikasi pada bagian tag Gridview menjadi listing berikut ini :

```
<asp:GridView ID="GridView1" runat="server"
AllowPaging="True"
    AutoGenerateColumns="False"
    BorderStyle="None" BorderWidth="0px"
    DataKeyNames="idBlog"
    DataSourceID="SqlDataSource3"
    EmptyDataText="There are no data records to
display."
    ShowFooter="True" ShowHeader="False"
    Width="100%" PageSize="5">
    <Columns>
        <asp:TemplateField
        HeaderText="JudulBlog"
        SortExpression="JudulBlog">
            <ItemTemplate>
                <cc1:CollapsiblePanelExtender
                    ID="CollapsiblePanelExtender3"
                    runat="server"
                    ExpandControlID="Panel3"
                    CollapseControlID="Panel3"
                    TargetControlID="Panel4"
                    Collapsed="true"
                    SuppressPostBack="true"
                    ExpandedText="Hide Detail"
                    CollapsedText="Show Detail"
                    TextLabelID="Label1"  >
```

```
                    </cc1:CollapsiblePanelExtender>
                    <asp:Panel
                    ID="Panel3" runat="server" >
                    <div style="background-color:aqua;
                    border-bottom :dashed 1px red;
                    font-weight:bold; font-size:12pt;
                    cursor:pointer; color:White  ">
                        <b>Posted at :
            <%#format(Eval("TglBlog"),"dd-MM-yyyy")%>
                  in <%#eval("Category") %>
                    </b><br />
                        <%#eval("JudulBlog") %>
                    </div>
                   </asp:Panel>
                    <asp:Panel
                           ID="Panel4" runat="server"
                        Height="0" Width="100%"  >
                        <%#Eval("IsiBlog")%>
                    </asp:Panel>
                </ItemTemplate>
            </asp:TemplateField>
        </Columns>
    </asp:GridView>
```

Modifikasi terakhir dari bagian Archive Blog dilakukan di dalam Gridview. Modifikasi ini merupakan modifikasi yang paling panjang di dalam mode Source, sebab di dalam modifikasi ini, selain mengubah jenis kolom dalam Gridview menjadi mode template (sehingga dapat ditempati oleh komponen lain), juga menyisipkan sebuah komponen Collapsible Panel serta sebuah Panel di dalam Gridview tersebut.
Penambahan kedua komponen tersebut, ditujukan agar terjadi efek partial postback saat pengguna melakukan klik pada judul sebuah blog, dan akan langsung menuju ke isi blog itu sendiri secara otomatis.

Pada saat seluruh modifikasi selesai dilakukan, maka dalam mode Design akan terlihat seperti pada gambar berikut ini :



Dan jika situs dijalankan maka contoh hasil terlihat seperti pada gambar ini :

..:: Simple *mini* Blog ::..

ID

Password

Login

Newest Blog

Test again

Posted at : 08-04-2008 in Campus Life
Last test ?

Once more

Finally

Test again

ASP .NET 3.x

Archieve

Category Campus Life

Posted at : 08-04-2008 in Campus Life
Test again

Posted at : 08-04-2008 in Campus Life
Once more

Posted at : 08-04-2008 in Campus Life