2. TEORI PENUNJANG

2.1. *Active Server Page.NET* (ASP.NET)

ASP.NET tidak sekedar *upgrade* dari ASP. ASP.NET menyediakan *platform* pengembangan *web* terdepan yang diciptakan dewasa ini. Terlebih lagi, ASP.NET dibangun dari dasar untuk membuat infrastruktur pengembangan *web* yang sama sekali baru.

Yang membuat ASP.NET menajdi sebuah revolusi adalah pembuatannya yang didasarkan pada *platform* baru *Microsoft.NET*, atau lebih tepatnya *.NET Framework*. Agar lebih memahami kapan dan dimana harus menggunakan ASP.NET, mari kita luangkan waktu untuk mempelajari *platform Microsoft.NET*, produk-produk yang dicakup, serta dimana ASP.NET cocok dalam *Microsoft.NET*.

2.1.1. *Microsoft.NET*

Microsoft.NET adalah istilah umum yang mencakup sejumlah teknologi yang baru dikeluarkan oleh *Microsoft*. Sebagai suatu kesatuan, teknologi ini adalah perubahan paling penting pada *platform* pengembangan *Microsoft* sejak pergeseran dari 16-*bit* ke 32-*bit*. *Microsoft.NET* mencakup teknologi berikut ini:

1. *.NET Framework.*
2. *.NET Enterprise Servers.*
3. Bahasa-bahasa dan *tools* bahasa .NET.

2.1.1.1. *.NET Framework*

.NET Framework adalah teknologi mendasar untuk pengembangan ASP.NET. Teknologi ini menyediakan layanan sistem dasar yang mendukung pengembangan *form* pada *Windows ASP.NET*, sebuah teknologi pengembangan *client* yang baru dan kaya yang disediakan oleh *.NET*. Teknologi ini kebanyakan menyerupai *Option Pack Windows NT 4.0*, sebagai *add-on* pada *Windows NT 4.0* yang menambahkan teknologi-teknologi *Internet Information Server 4.0* dan

Active Server Pages ke dalam *NT 4.0, .NET Framework* adalah *add-on* pada *Windows 2000, Windows NT 4.0, Windows 98/Me* yang menambahkan dasar layanan sistem pendukung untuk teknologi .NET. *Framework* ini juga akan dimasukkan ke dalam jajaran sistem operasi *server Windows* rilis baru, termasuk jajaran server *.NET Server*. *.NET Framework* terdiri dari dua bagian utama:

- *Runtime* bahasa umum

  Runtime bahasa umum (*Runtime)* menyediakan lingkungan *runtime* untuk eksekusi kode yang ditulis dalam bahasa .NET. *Runtime* mengelola eksekusi kode .NET, termasuk pengelolaan *runtime* dan masa hidup obyek. Selain layanan pengelolaan, *runtime* memungkinkan pengembang melakukan *debug*, penanganan eksepsi, serta pewarisan antar berbagai bahasa.

  Untuk melakukan kegiatan-kegiatan tadi dibutuhkan *compiler* bahasa yang mengikuti *Common Language Spesification* (*CLS*), yang menerangkan suatu *subset* tipe data yang didukung oleh *runtime* serta umum dipakai pada semua bahasa dalam .NET. *Compiler* bahasa mengkompilasi kode yang ditulis oleh pengembang menjadi sebuah bahasa tingkat menengah yang disebut *Microsoft Intermediate Language* (*IL* atau *MSIL*). *IL* ini kemudian dikompilasi menjadi bahasa asli oleh *runtime* pada saat instalasi, atau dikompilasi menjadi *Just-In-Time* (*JIT*) pada awal eksekusi. Kode yang dikompilasi oleh *IL* dan dikelola oleh *runtime* disebut kode terkelola. Disebut kode terkelola karena tanggungjawab pengelolaan eksekusinya dipikul oleh *runtime*, termasuk instansi obyek, elokasi memori, serta pengumpulan sampah obyek dan memori. Komponen-komponen yang dikelola oleh *runtime* disebut kode terkelola oleh .NET, atau singkatnya *assembly*. *Assembly* adalah unit dasar penerapan dalam dunia .NET serta menyerupai komponen *COM*. Perbedaanya adalah komponen COM mempunyai tipe pustaka yang menyatu untuk menerangkan bagaimana client berinteraksi dengannya. Sebuah *assembly* memiliki manifestasi, yaitu serangkaian *metadata* yang menerangkan isi *assembly*. Dari sekian banyak kelebihan *assembly* salah satunya adalah sifat menerangkan diri pada komponen .NET. Artinya untuk menjalankannya, komponen-komponen tersebut tidak perlu diregister pada komputer.

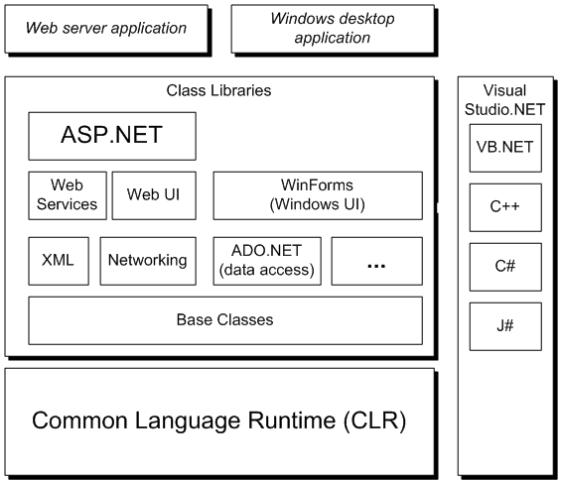
Metadata juga menerangkan dependensi dan informasi versi yang berkaitan dengan suatu *assembly*. Hal ini bukan saja mempermudah untuk memastikan bahwa semua dependensi yang diperlukan sudah terpenuhi, tetapi juga berarti berbagai versi *assembly* yang sama bisa berjalan berdampingan dalam satu komputer tanpa menimbulkan konflik. Ini adalah lompatan besar dalam penyelesaian bagi *"DLL Hell"*, petaka bagi eksistensi para pengembang. Para pengembang yang telah bekerja dengan *(ActiveX Data Objects)* ADO, mempunyai banyak keluhan tentang aplikasi-aplikasi yang rusak akibat versi ADO yang baru. Dengan memakai .NET semua dapat teratasi. Selama aplikasi yang menggunakannya tahu tentang versi mana pada *assembly* yang harus dipakai, ia akan memakai versi yang tepat dari sejumlah versi yang sama dengan cara melakukan *query* ke *metadata assembly*.

- Pustaka Kelas *.NET Framework*

  Pustaka kelas *.NET Framework* dirancang untuk mendukung usaha pengembang dalam menyediakan kelas-kelas dasar yang akan dipakai dalam pewarisan. Pusataka ini adalah kumpulan hirarki kelas-kelas .NET yang bisa dipakai pengembang dalam aplikasi mereka. Kelas-kelas ini yang disusun oleh wadah yang bernama *namspace*, menyediakan baik fungsi dasar maupun lanjut yang bisa dengan mudah dipakai ulang oleh pengembang. *Namespac*e ini menyimpan kelas-kelas yang mendukung tipe-tipe data yang umum, kelas yang menyediakan akses ke data, serta kelas yang mendukung layanan sistem seperti gambar, fungsi *network* (termasuk DNS dan *lookup* balik DNS*)*, dan masih banyak lagi. Pustaka juga berisi kelas-kelas yang membentuk dasar ASP.NET, termasuk kelas *Page* (bagian dari *namespace System.Web.UI*) tempat berasalnya halaman-halaman ASP.NET, dan banyak lagi kelas lainnya dalam *namespcae System.Web* beserta anak-anaknya.

- *Windows.NET Server*

  Hampir sama dengan *Microsoft Transaction Server, Microsoft Message Queue Server, Internet Information Server,* dan *Internet Explorer* adalah produk-produk yang diinstal secara terpisah yang selanjutnya disatukan menjadi dasar *system* operasi, *runtime* dan pusataka kelas *.NET Framework* akan menjadi bagian dari *system* operasi *Windows.*



Gambar 2.1. .*NET Framework* Arsitektur

Berdasarkan gambar diatas dapat dilihat, pengembangan .NET *platform* merupakan satu set komponen untuk membangun aplikasi *Web Server* dan aplikasi *Windows Desktop.* Aplikasi terbentuk dari *platform run* di bagian CLR, kemudian suau *software engine* mengeksekusi aplikasi dan memberikan pernyataan apakah aplikasi tersebut tidak mempunyai *error* agar dapat dilanjutkan, selain itu melakukan pengecekan permintaan *security*, menjalankan aplikasi dan membersihkan objek yang tidak diperlukan setelah eksekusi selesai. Satu *set class libraries* menyediakan *code* yang dapat membuat aplikasi membaca dan menuliskan data XML, berkomunikasi lewat *internet*, mengakses database, menampilkan *user interface* dan lainnya. Semua *class libraries* menyedian fungsi untuk mengatur data yang paling sering digunakan, seperti misalnya *numbers* atau *text strings* dan *low-level* fungsi seperti *input* dan *output*. Aplikasi *web server* secara normal berada dalam ASP.NET, dimana suatu *server-side library* memproses w*eb requests*. ASP.NET memberikan jawaban pada *web services library* untuk mengirimkan dan menerima *SOAP message*, dan *Web user interface library* (*Web Forms*) digunakan untuk menerima *input*-an user dari *browser* dan secara dinamik melakukan *response* pada *Web Page*. Aplikasi *Windows Desktop* dapat menampilkan grafik UI melalui *WinFroms Library* yang juga dikenal sebagai *Windows Forms*. Terakhir *Visual Studio.NET* menyediakan IDE (*Integrated Development Environment*) untuk membuat aplikasi pada *platform*.

2.1.1.2. *.NET Enterprise Servers*

.NET Enterprise Servers adalah langkah awal dalam evolusi *platform* pengembangan Microsoft. Meskipun tidak secara eksplisit memanfaatkan *runtime* dan pustaka kelas, *.NET Enterprise Servers* betul-betul membentuk fondasi yang kokoh. Dalam *.NET Enterprise Servers* termasuk:

- SQL Server 2000
- Exchange 2000 Server
- Commerce Server 2000
- Host Integration Server 2000
- BizTalk Server 2000
- Internet Security and Acceleration Server 2000
- Application Center 2000

Apabila disatukan, produk-produk diatas memberi banyak fungsi yang dibutuhkan oleh kebanyakan perusahaan besar, sebagai nilai tambah.

2.1.1.3. Bahasa dan Tool-Tool Bahasa

Salah satu yang terbaik pada *platform* .NET adalah, sementara ASP klasik membatasi pengembang pada bahasa *script*, ASP.NET memungkinkan kita bekerja dengan semua bahasa yang sesuai dengan .NET. Ini artinya kode yang ditulis dalam ASP.NET dikompilasi agar kinerjanya lebih baik, sambil memanfaatkan fitur-fitur bahasa tingkat lanjut. Untuk *platform* .NET, bahasa-bahasa akan menjadi topik utama yang dibahas adalah:

- *NOTEPAD* ".NET"

  Masih banyak pengembang ASP, masih bekerja memaki *Microsoft Notepad*. Meskipun dipakai dimana-mana, *tool* ini bukanlah lingkungan untuk pengembangan aplikasi yang bisa disebut besar. Artinya, jika anda berkeja dengan *.NET Framework* SDK (Sebagai pengganti *Visual Studio .NET*), tidak ada alasan kita tidak memakai *Notepad* untuk mengerjakan semua pengembangan .NET. *.NET Framework* SDK mengandung *compiler* baris perintah untuk *Visual Basic .NET*, C# (baca "*C sharp*"), dan *Jscript .NET*. Jadi kita dapat membuat kelas, halaman ASP.NET dan sebagainya dengan *Notepad*, kemudian mengkompilasinya secara eksplisit dengan *compiler* baris

perintah atau dalam ASP.NET, memungkinkan *runtime* ASP.NET untuk secara dinamis mengkompilasi halaman yang dibutuhkan pada permulaannya.

- *Visual Studio.NET*

  Untuk pengembangan yang lebih mudah dan cepat, mungkin kebanyakan pengembang lebih memilih *Visual Studio.NET*. Untuk pertama kalinya dengan *platform* pengembangan *Microsoft, Visual Studio.NET* menyediakan satu *Integrated Development Environment* (IDE) untuk semua bahasa-bahasa *Microsoft.NET*. *Visual Studio.NET* menyediakan banyak fitur baru, di antaranya:

  - Satu model pemrograman yang terpadu untuk semua bahasa-bahasa .NET, untuk, *Windows* maupun aplikasi *web*.
  - Pengembangan *drag-and-drop* untuk *server* dengan memakai *Server Explorer*.
  - Bantuan dinamis.
  - Model kustomisasi yang besar dan jangka lama untuk IDE.
  - Sangat mendukung XML.
  - *Web services* dengan integrasi aplikasi antar *platform* yang jauh lebih mudah.

  Lebih lanjut mengenai *Visual Studio.NET* akan dibahas pada sub bab berikutnya.

- *Visual Basic.NET*

  Seperti telah disebutkan, pengembang ASP.NET tidak lagi hanya pada versi terbatas *Visual Basic* seperti VB*Script*. Pengembang sekarang dapat memanfaatkan kekuatan penuh *Visual Basic.NET*. Kekuatan ini telah ditingkatkan dalam *Visual Basic.NET* dengan menambanhkan dukungan pewarisan, serta dukungan penanganan eksepsi terstruktur, ini hanya sebagian kecil perbaikan saja.

  Visual Basic.NET adalah bahasa yang teratur, artinya *runtime* mengatur eksekusi kode *Visual Basic.NET*. Kita tidak perlu lagi secara eksplisit menghancurkan referensi obyek dengan *set object = nothing,* seperti yang dilakukan dengan komponen COM pada *Visual Basic 6* dan sebelumnya.

Kolektor sampah .NET akan secara otomatis memulihkan memori yang dipakai oleh komponen tersebut, begitu tidak ada *client* yang memakainya.

Keterangan lebih lanjut mengenai *Visual Basic.NET* akan diterangkan pada sub bab berikutnya.

- C#

  Anggota baru keluarga *Visual Studio*, C# adalah turunan dari C. Lebih menyerupai C++, tetapi dirancang agar lebih mudah digunakan. Meskipun c# tidaklah semudah *Visual Basic*, tetapi jauh lebih mudah dibanding C++, lagipula semua kemampuan yang ada pada C++ juga ada pada C#. Kita tidak perlu mengelola alokasi dan dealokasi sebagaimana dalam C++. Oleh karena C# sama dengan *Visual Basic.NET,* bahasa yang teratur, semua pengelolaan memori ditangani oleh *runtime.* Ini adalah keuntungan penting karena pengelolaan memori adalah salah satu hambatan dalam pengembangan C++ yang menyebabkan banyak aplikasi *crash*.

  Tidak ada keterangan lebih lanjut mengenai C# karena *prototype* yang dibuat akan menggunakan *Visual Basic.NET.*

2.1.2. Arsitektur ASP.NET

Selain banyak fitur-fitur ASP.NET, ada juga beberapa perubahan besar pada arsitektur ASP.NET, termasuk banyak perbaikan dan fitur-fitur baru. Perlu diperhatikan bahwa banyak hal dalam ASP.NET bukanlah hal baru bagi pengembang *web* yang pernah memakai obyek-obyek aplikasi, sesi dan *server*, meskipun dengan metode *property* yang baru. Anda masih bisa memakai blok *<SCRIPT RUNAT "SERVER">* atau *delimiter script* ASP *<% %>* untuk menunjukkan *script server-side*. Pada kenyataannya, kebanyakan user dapat menulis halaman ASP.NET persis sama dengan cara menulis halaman ASP klasik. Meskipun begitu terbiasa dengan model pemrograman ASP.NET, user tidak akan kembali ke pengkodean aplikasi ASP.

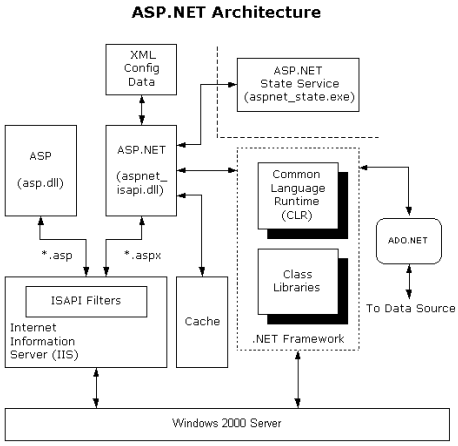
Lagipula, user tidak perlu melakukan migrasi semua aplikasi ASP sekaligus. ASP.NET memang dirancang untuk jalan berdampingan dengan ASP klasik. Jadi sambil bekerja dengan ASP.NET yang baru, user masih bisa

menjalankan aplikasi ASP yang ada sekarang secara berdampingan. Banyak hal yang baru dalam ASP.NET. Beberapa fitur baru tersebut adalah:

- *Web Forms.* Ini adalah model pemrograman yang baru pada *web form* ASP.NET yang menggabungkan aplikasi yang terbaik pada ASP dengan kemudahan pengembangan dan produktivitas *Visual Basic*. Anda bisa menarik kontrol ke dalam halaman kemudian menuliskan kode untuk menyediakan interaksinya, memanggil obyek bisnis, dan sebagainya.
- Kontrol Server. Komponen besar pada model pemrograman *Web Form*, kontrol server ASP.NET memetakan ke elemen HTML serta menyediakan kemampuan memrogram *server-side* yang handal. Kontrol server dijalankan pada server dan bisa membuat output HTML yang dirancang untuk browser tingkat tinggi, seperti *Intenet Explorer 5.x* atau yang lebih baru, atau untuk semua browser yang sesuai dengan HTML 3.2.
- *Web Services*. Ini adalah kunci ASP.NET yang memungkinkan pengembang membuat layanan program yang bisa dipakai pengembang lain melalui internet. *Web services* didasarkan pada tumbuhnya standar SOAP (*Simple Object Access Protocol*), sehingga memungkinkan hubungan yang mudah di antara operasi lewat *platform* yang berbeda.
- *Caching*. ASP.NET memiliki mesin *caching* yang baru dan handal yang memungkinkan pengembang meningkatkan kinerja aplikasi mereka dengan mengurangi beban pemrosesan *server web* dan *server* database.
- Perbaikan Konfigurasi. ASP.NET menggunakan metode baru penyimpanan infromasi konfigurasi untuk aplikasi *web*. Sebagai pengganti memerintah IIS menyimpan informasi ini dalam database yang sulit diakses, infromasi disimpan dalam file konfigurasi berbasis XML yang bisa dibaca oleh mesin maupun oleh user.
- Perbaikan Pengelolaan Status. ASP.NET mengatasi keterbatasan dengan menyediakan dukungan pendistribusian status sesi dalam *server web,* menaruh informasi status dalam database SQL Server, serta menyediakan pengelolaan status tanpa menggunakan *cookies.*
- Pengamanan. Fitur ini adalah yang terpenting dalam aplikasiweb saat ini. Model pengamanan ASP.NET telah mengalami perbaikan besar, termasuk

metode otentikasi yang baru dan lebih baik, pengamanan akses kode, serta ototrisasi berdasarkan peran.

- Perbaikan Kehandalan. ASP.NET berisi fitur baru ini untuk meningkatkan kehandalan aplikasi *web*, termasuk *restart* aplikasi yang proaktif serta proses otomatis daur ulang untuk mengatasi kondisi deadlock dan kebocoran memori.



Gambar 2.2. Arsitektur ASP.NET

Berdasarkan gambar diatas ASP.NET menggunakan ISAPI agar dapat dijalankan pada IIS di *Windows 2000 Server*. Tidak hanya IIS ASP.NET tetapi ISAPI *filter* juga mengijinkan ASP dan ASP.NET untuk berada dalam *server* IIS yang sama. Konfigurasi dari ASP.NET diatur dengan menggunakan informasi yang tersimpan dalam format XML pada konfigurasi file (*Web.config). Cache* memperbolehkan improvisasi dalam kinerja ASP.NET, sebagai contoh halaman yang paling sering diminta akan ditampilkan dari ASP.Net *cache. State management services* untuk ASP.NET menyediakan state service ASP.NET. *.NET Framework* meyediakan CLR yang mengatur dan menjalankan eksekusi kode dari ASP.NET dan class libraries yang menawarkan untuk pembangunan fungsi program, untuk *Web Forms*, XML *support* dan *exception* handling. ADO.NET menyediakan fungsi untuk berhubungan dengan database.

2.1.3. *Namespace* ASP.NET

Dalam ASP.NET kita mengenal istilah *namespace*. *Microsoft* mendefinisikan *namespace* sebagai skema penamaan logis untuk

mengelompokkan tipe-tipe yang berhubungan. *Namespace* tidak lain dari kumpulan obyek-obyek yang berhubungan. ASP.NET telah menyediakan obyek-obyek siap pakai, dan obyek-obyek yang berhubungan tersebut dikelompokkan dalam satu *namespace.*

Sebuah *namespace* bisa memiliki *subnamespace*, yaitu *namespace-namespace* yang berada di bawahnya. *Namespace System* misalnya, memiliki *subnamespace-subnamespace* seperti *System.Web* yang berisi sekumpulan obyek yang berguna untuk pengaturan *web*, *System*.XML berisi sekumpulan obyek untuk pengaturan XML, dan sebaginya.

Namespace System merupakan namespace utama dari ASP.NET. Semua *namespace-namespace* lainnya merupakan *subnamespace* dari *namespace System*. Pada prinsipnya, walaupun kita tidak menyatakannya secara eksplisit, *namespace System* telah menjadi referensi bagi halaman ASP.NET yang di buat. Pada saat dkompilasi, CLR akan mencari referensi obyek-obyek yang kita gunakan pada *namespace System*.

2.1.4. Struktur Halaman ASP.NET

Halaman ASP.NET yang kita guankan sampai saat ini berbentuk:

```
<%@ page language=”VB” %>
<script runat=”server”>
……kode VB.NET
</script>
…kode HTML
<%
…kode VB.NET
 %>
…kode HTML
```

Pada prisipnya ada 3 bagian NON-HTML disini, yaitu:

- Direktif

  Direktif dimulai dengan tanda <%@ dan diakhiri dengan %>. Ada cukup banyak direktif dalam ASP.NET. Enam direktif yang paling sering digunakan ditampilkan pada tabel 2.1.

Tabel 2.1 Direktif yang paling sering digunakan pada ASP.NET

Direktif	Keterangan
@Page	Mendeklarasikan atribut-atribut yang berkaitan dengan halaman ASP.NET
@Import	Mengimpor sebuah namespace dalam flie class halaman ASP.NET
@OutputCache	Menspesifikasikan bagaimana mengatur cache halaman ASP.NET
@Register	Mendaftarkan suatu kontrol baru yang dibuat dalam halaman ASP.NET
@Assembly	Melink kepada kode assembly
@Implements	Mengimplementasikan interface dalam halaman ASP.NET

- *Skrip in-page*

  Skrip in-page pada ASP.NET mirip dengan skrip yang biasa kita gunakan dalam ASP versi sebelumnya. *Skrip in-page* selalu diapit dengan tanda <% dan %>. Dalam *skrip in-page* inilah kita membuat kode-kode ASP.NET standar.

- *Skrip code-behind*

  Skrip code-behind merupakan skrip-skrip yang dapat ditempatkan dalam file terpisah yang bisa diselipkam ke dalam halaman ASP.NET yang sedang dikerjakan. Kode-kode dalam skirp *code-behind* merupakan kode-kode seperti pendefinisian prosedur, struktur, dan *class.* Bila skrip ini disatukan dalam halaman ASP.NET yag sedang dikerjakan, maka skirp ini harus diapit oleh *tag* <*script runat=”server”*> dan <*/script*>. Skrip *code-behind* yang digunakan selama ini selalu disatukan dalam satu halaman dengan halaman ASP.NET yang sedang dikerjakan. Bagaimana bila ingin memisahkannya? Dengan ASP.NET, pemisahan ini bisa dilakukan dengan mendefinisikannya sebagai *namesapce* baru. Hal ini bisa dilakukan dengan 2 cara:

  1. Tanpa kompilasi, skrip *code-behind* disimpan dalam file .vb (untuk VB.NET) atau .cs (untuk C#.NET).

2. Dengan kompilasi, script *code-behind* disimpan dalam file .dll yang dikompilasi dari file VB.NET ataupun C#.NET.

   Berikut contoh scriptnya:

```
<@ Page inherits=”belajar.pesan” src=”kodeku.vb” %>
```

2.1.5. Tipe-Tipe File ASP.NET

Beberapa tipe-tpe file ASP.NET:

- .aspx. Ekstensi inilah yang paling sering ditemui. Tipe file ini dipakai untuk halaman *Web Form*, bisa dianalogikan dengan ekstensi .asp dalam ASP klasik.
- .ascx. Ekstensi ini dipakai untuk kontrol pengguna *Web Form* adalah .ascx. Kontrol pengguna menyediakan salah satu dari banyak cara yang ada dalam ASP.NET untuk pemakaian ulang kode. Kontrol ini sama dengan menyisipkan file pada ASP klasik, dalam hal sama mudahnya dengan tag HTML atau bisa memasukkan logika yang kompleks dan ingin deipai ulang dalam banyak halaman. Kontrol pengguna ditambahkan ke halaman *Web Form* dengan memakai *direktif @Register*.
- .asmx. Ekstensi ini dipakai untuk file yang menerapkan layanan XML *Web*. Layanan XML *Web* bisa dikases langsung lewat file .asmx, ayau file ini bia juga menersukan permintaan ke iyang dikompilasi yang menerpakan layanan *web*.
- .vb. Ekstensi untuk modul kode *Visual Basic.NET* adalah .vb. Semua halaman *Web Form* (.aspx) yang ditambahkan ke dalam aplikasi *Web Visual Studio.NET* yang ditulis dalam *Visual Basic* akan memiliki modul code-behind .vb dengan nama yang sama seperti halaman *Web Form* yang dihubunginya.
- .cs. Ekstensi untuk modul C# adalah .cs. Sperti halnya ekstensi .vb, semua halaman *Web Form* (.aspx) yang ditambahkan ke aplikasi C# *Visual Studio.NET Web* akan memiliki modul .cs dengan nama sama seperti halaman *Web Form* yang dihubunginya.
- *Global.asax*. Sama dengan *Global.asa* pada ASP klasik, Global.asax adalah file yang dipakai untuk mendefinisikan *variable application* dan *session-level*

serta prosedur *start-up*. Kita perhatikan bahwa struktur global.asax meyerupai Global.asa, dengan prosedur-prosedur seperti *Session_OnStart (Session_Start* dalam ASP.NET) kodenya ditulis langsung dalam file *Global.asax* dalam blok *<script runat=”server”>, Visual Studio* menerpakan prosedur ini dalam modul *code-behind* ketimbang dalam file global.asax itu sendiri.

- Web.config. *Web.config* adalah tipe baru dalam ASP.NET. File ini dipakai untuk menyelesaikan masalah besar ASP klasik yaitu konfigurasi. File *Web.config* adalah file berbasis XML yang bisa dibaca, baik oleh mesin maupun manusia. File ini menyimpan semua setting konfigurasi untuk sebuah aplikasi. File *Web.config* diterjemahkan secara hierarkis, maksudnya, file *Web.config* dalam subdirektori aplikasi akan menyisihkan *setting file Web.config* dalam direktori induk. Keuntungannya adalah setting konfigurasi bisa diwariskan jika diinginkan, tetapi kita tetap bisa mengatur konfigurasinya.

2.1.6. *Web* Kontrol.

Web kontrol adalah jenis dari *server* kontrol ASP.NET yang tidak terkait dengan kode HTML secara langsung, Hal ini berbeda dengan HTML kontrol yang merupakan jenis *server* kontrol yang merupakan kontrol yang terkait langsung dengan kode HTML. Beberapa kategori *web* kontrol beserta scriptnya:

- Label kontrol.

  Label kontrol berguna untuk menampilkan teks secara terprogram, kerna merupakan obyek kontrol. Contoh *script*-nya:

  ```
  <asp:Label id=”label1” runat=”server” text=”Hallo” />
  ```

- Button kontrol.

  Button kontrol merupakan *web* kontrol yang biasanya digunakan untuk *button submit* dari *form* suatu aplikasi. Contoh *script*-nya:

  ```
  <asp:button id=”Button1” text=”Submit”
  onclick=”Button1_click” Runat=”server” />
  ```

  Button kontrol dapat digunakan untuk *submit form* yang ada pada halaman aspx atau menghasilkan *event command.*

- Image Button.

Selain *Button* kontrol yang menampilkan tombol pada *browser*. *User* dapat menggunakan *Image Button* yang menampilkan *image*, tetapi memiliki fungsi sebagai *button* atau tombol.

Berikut script yang digunakan:

<asp:ImageButton OnClick=”ImageButton_Click” ImageURL=”gambarku.gif” runat=”server” />

- Link Button.

  Link button adalah sama dengan *button* namun *button* ini menggunakan teks yang mengandung *hyperlink.* Berikut *script* yang digunakan:

  ```
  <asp:linkbutton onclick=”link_click” text=”Klik disini” runat=”server” />
  ```

- Textbox.

  Text box akan menampilkan obyek *form* yang berfungsi untuk menerima *input* teks dari *user*.

  Berikut contoh scriptnya:

  ```
  <asp:textbox id=”textbox1” runat=”server” textmode=”singleline” />
  ```

- Radio button.

  Radio button biasanya digunakan bila *users* ingin melakukan kontrol terhadap 1 atau lebih *variable* tetap. Berikut contoh *script*-nya:

  ```
  <asp:radiobutton id=”radio1” runat=”server” text=”Remove” Checked=”true” />
  ```

- Checkbox Kontrol.

  Hampir sama dengan *radio button* namun berbentuk tanda chek. Berikut *script*-nya:

  ```
  <asp:CheckBox id=”check1” text=”bayar” runat=”server” Checked=”false” />
  ```

- Drop Down List.

  Merupakan kontrol yang memiliki konsep sama dengan *radio button* namun berupa *list*, hanya saja *Drop Down List* lebih efisien dalam tampilan di layar, karena menempati ruang lebih kecil.

  Berikut contoh *script*-nya:

  ```
  <asp:dropdownlist id=”drop1” runat=”server” >
        <asp:listitem value=”Pizza” selected=”true” />
  ```

```
      <asp:listitem value="Ice" Selected="false" />
</asp:dropdownlist >
```

- ListBox

  Merupakan kontrol yang menampilkan daftar pilihan dalam satu kotak. Berikut contoh *script*-nya:

```
<asp:listbox id="list" runat="server" rows="2" >
      <asp:listitem value="satu">Satu</asp:listitem>
      <asp:listitem value="dua">Dua</asp:listitem>
</asp:listbox>
```

- Hyperlink

  Dapat berupa teks atau *image* yan merujuk ke halaman lain. Berikut contoh *script*-nya:

```
<asp:hyperlink id="hyper1" target="window"
navigateurl="login.aspx" text="Tekan sini" runat="server"/>
```

- Image kontrol.

  Berguna untuk menyisipkan *image* ke dalam *form*. Berikut contoh *srciptnya*:

```
<asp:image id="image1" runat="server"
imageurl="gambarku.gif" />
```

  Ada beberpa *web* kontrol lainnya namun untuk pengerjaan *website prototype* ini hanya menggunakan beberapa *web* kotnrol yang sudah dijelaskan diatas.

2.1.7. Validation Kontrol.

Kontrol ini berguna untuk memberikan validasi terhadap form secara praktis dan mudah. Ada beberapa *validation* kontrol yang dipakai di *prototype website* yang dibuat diantaranya adalah:

- *Required Field Validator*

  Digunakan untuk mengecek apakah kontrol memiliki nilai yang biasanya digunakan untuk kontrol *textbox.* Contoh *script*-nya:

```
<asp:requiredfieldvalidator id="re1" runat="server"
errormessage="name harus diisi" initial value="youname"
controltovalidate="textbox1">Anda harus mengisi
nama</asp:requiredfieldvalidator>
```

- *Compare Validator*

Kontrol ini digunakan untuk membandingkan antara data yang diisikan dengan nilai lain. Berikut contoh *script*-nya:

```
<asp:comparevalidator id=”compare1” runat=”Server”
errormessage=”comparevalidator” controltovalidate=”password”
valuetocompare=”confirmpassword” />
```

2.1.8. User Kontrol

Dengan *user* kontrol kita dapat melakukan hal yang sama dengan kemampuan dan fasilitas yang jauh lebih baik untuk melakukan *include* halaman statis yang biasanya digunakan untuk *header* dan *footer* sehingga tiap halaman web memiliki tampilan seragam. Biasanya *user* kontrol dibuat dengan ekstensi .ascx seperti pembuatan *web* biasa namun pemanggilan di .aspx nya berbeda. Berikut contoh *script*-nya:

```
<@ register tagprefix=”Usercontrolku” tagname”header”
src=”header.ascx” %>
```

Diatas adalah *script* registrasi pada halaman *web* yang menggunakan *user* kontrol. Berikut adalah contoh penggunaan *user* kontrol tersebut.

```
<usercontrolku:header id=”headerku” runat=”server” />
```

2.2. *Visual Studio.NET*

Dalam suatu development tools dikenal adanya IDE (*Integrated Development Environment*). IDE merupakan suatu lingkungan di mana kita membangun aplikasi. Ada beberapa kemajuan yang mendukung produktivitas pembuatan dalam IDE *Visual Studio.NET* ini:

- *Start Page.* Ini adalah halaman *default* yang ditampilkan setiap kali menjalankan *Visual Studio.NET*. Halaman ini memungkinkan membuat prefensi untuk IDE, mengakses proyek terbaru atau yang ada, serta memnuat proyek baru.
- IDE Multibahasa. Tidak seperti *Visual Studio 6.0*, yang memakai IDE berlianan untuk setiap bahasa, dalam *Visual Studio.NET*, semua bahasa memakai IDE yang sama. Ini berarti fitur-fitur standar sperti *Find and Replace, debug*, dan sebagainya bekerja secara konsisten dalam bahasa-bahasa yang berlainan. Ini saja menjadi perbaikan besar dalam produktivitas.

- *Tabbed Documents.*

  Dirancang untuk mempermudah pengelolaan berbagai file yang diedit terus menrus, anatarmuka. *Tabbed Documents* ini memungkinkan untuk melihat semua file yang sedang di *edit* sekaligus. Hal ini akan mempermudah bolak-balik antar *windows* yang sedang di edit. Akan tetai pemakai tetap bisa menggunakan *Visual Studio .NET* dalam memakai metode lama pada *Visual Studio 6.* Pilih saja *Options* dari menu *Tools*, pilih *General* dalam *folder Environment*, pindah dari *Tabbed documents* ke lingkungan MDI, kemudian pilih OK. Agar perubahan dilakukan, pemakai perlu melakukan *restart* pada *Visual Studio .NET.*

- *Auto-hide. Auto-hide* bekerja seperti fitur dengan nama yang sama pada *toolbar Windows*. Untuk mengaktifkan fitur *auto-hide* pada sebuah *window*, klik pada ikon paku payung dalam baris judul *window*. Sekarang window itu akan tersembungyi dibalik IDE yang melesak saat mouse menjauh dari *window*, yang terlihat hanya *tab* dengan nama *window* itu. Menaruh *mouse* di atas *window* akan menampilkan kembali *window* itu. Ini adalah fitur yang bagus untuk menjaga jumlah maksimum tampilan *window* kode, serta mudah mengatur berbagai *window* dalam IDE.

- Perbaikan Editor HTML.

  Seperti *Visual InterDev* yang mendahulukan editor *Visual Studio.NET,* HTML menyediakan tampilan desain dan tampilan HTML. *Visual Studio.NET* melangkah jauh dengan *window Quick View* yang disediakan *Visual InterDev*. Selain itu, pemakai dapat melihat halaman dalam *window* browser yang menyatu, tampilan yang lebih pasti bagaimana halaman sebenarnya akan ditampilkan. *Editor* yang diperbaiki ini juga mendukung untuk memperlihatkan skema yang sedang ditulis melalui *property targetSchema*. Pengaturan *targetschema* menentukan eleman mana saja yang akan ditampilkan lewat fitur pelengkap pernyataan dan memungkinkan IDE memberi masukan tentang sintaks yang salah dalam konteks skema pda target yang dipilih.

  Selain perbaikan IDE, ada juga sejumlah fitur yang sama sekali baru dalam *Visual Studio.NET* IDE:

- *Editor* XML. Ini memungkinkan pemakai meng-*edit* file data XML dan file skema (.xsd) dalam tampilan sumber, data atau skema, tergantung tipe file XML yang sedang di-*edit*.
- Dokumentasi otomatis. Fitur menarik yang saat ini hanya ada dalam C#. Fitur ini memungkinkan pemaakai membuat dokumentasi dari komentar-komentar yang ada dalam kode C# menggunakan delimiter (///) komentar dan sintaks khusus. *Visual Studio* juga dapat menghasilkan dokumentasi untuk proyek dan solusi apapun bahasa yang dipakai dalam proyek tersebut.
- *Dynamic Help.* Fitur yang menyediakan *context-sensitive help* saat pemakai bekerja dengan IDE, menyarankan topik saat pemakai menambahkan file, kontrol, dan kode ke dalam proyek.
- Dukungan untuk *window*s *Installer. Visual Studio* sekarang mendukung teknologi *set-up* yang jauh lebih maju untuk aplikasi *Window*s. Termasuk dukungan untuk instalasi mundur jika terjadi masalah dalam instalasi. Pemakai bahkan dapat menerapkan paket aplikasi w*eb* yang memungkinkan pemakai menginstal dan enjalankanaplikasi ASP.NET pada mesin yang belum terinstal ASP.NET Framework. Paket penerapan tersebut akan menginstal semua *runtime* yang penting.

  Selama bekerja dengan *Visual Studio.NET*, pemakai akan menemui berbagai variasi *window* dalam IDE, yang dipakai untuk berbagai tujuan. Sebagian memang baru, seperti *window*s *Dynamic Help*, sebagian lagi sudah tidak asing lagi bagi pengguna versi *Visual Studio* sebelumnya.

2.3. Visual Basic.NET

Sejak *Visual Basic 1.0*, yang dibangun untuk membuat aplikasi *window*, sampai *visual Basic* 4.0, yang telah membantu terbentuknya konsep COM2, bahasa *Visual Basic* telah menjadi pendukung *platform Windows.*

Visual Basic.NET menjadi pilar dalam *.NET Framework, dan* merupakan langkah berikut dari hasil evolusi bahasa VB 6.0. VB. NET merupakan bahasa pemrograman *high-level* untuk *.NET Framework,* dan menyediakan pintu masuk bagi banyak kalangan *developer* untuk platform *Microsoft.NET.* Lahirnya generasi terbaru dari VB, oleh programmer dunia

dianggap sebagai revolusi terhadap perkembangan bahasa VB. VB.NET memang diahirkan dengan paradigma pemrograman yang baru. Ada 2 paradigma baru yang didukung secara menakjubkan pada VB .NET dibandingkan dengan VB versi sebelumnya, yaitu *client server* dan OOP (*Object Oriented Programming*).

VB.NET kini mendukung secara penuh konsep *client server*. Dengan mengguanakn VB.NET, sebuah aplikasi yang berbasis *client server* bisa dikembangkan dengan cukup gampang. Aplikasi berbasis client server banyak digunakan di Internet, dan salah satu aplikasi nyatanya terlihat pada pembuatan Web.

OOP juga kini didukung secara penuh dalam VB.NET. Dibandingkan dengan versi sebelumnya, kini VB.NET memungkinkan paradigma-paradigma OOP diimplementasikan. Berbagai paradigma OOP sperti objek, c*lass, inheritance, polymophism*, kini bisa diaplikasikan dengan menggunakan VB.NET Berikut ini adalah perubahan-perubahan yang terjadi dalam VB.NET:

- Tipe data *Variant* tidak berlaku lagi. Tipe ini digantikan dengan tipe Objek, dimana tipe Objek harus secara eksplisit dikonversi menjadi tipe data primitif lain.
- Tanda kurung harus digunakan saat memanggil suatu *method*, bahkan untuk *method* yang tidak menggunakan parameter. Sebagai contoh:

```
If Flag = Flase then
   DisplayMessage()
End If
```

- Secara *default*, parameter yang digunakan by value, tidak dengan by reference seperti pada versi sebelumnya. Jika ingin menggunakan by reference, pengembang perlu menggunakan *ByRef* di awal parameter.

```
Call Myfunction(argbyvalue, ByRef argbyref)
```

- *Set* dan *Let* tidak di-support lagi. Obyek dapat di-assign denganmenggunakan operasi biasa.

```
Object1 = Object2
```

- Tipe data *integer* sekarang 32 *bits* dan tipe data *Long* adalah 64 *bits*
- Manipulasi ekspresi *Boolean* tekah diubah. Sebagai contoh, jika dibagian pertama dari multiple *AND* bernilai *False*, maka hasil dari keseluruhan

ekspresi bernilai *False*. Jika di bagian pertama dari *multiple OR* bernilai *True*, maka hasil dari keseluruhan ekspresi bernilai *True*.

- Tipe data harus dikonversi agar dapat digabung dengan tipe data lain. Sebagai contoh:

  ```
  Response.Write(“The count is “ & Cstr(count))
  ```

- Variabel-variabel yang dibuat dalam statement Dim yang sama akan mempunyai tipe data yang sama. Sebagai contoh, dalam *Visual Basic .NET*, *statement Dim:*

  ```
  Dim I, j, k As Integer.
  ```

  Akan membuat tiga buah objek (I, j, dan k) sebagai *Integer*. Di *versi Visual Basic* sebelumnya, hal ini akan membuat I dan j sebagai *Variant* dan k sebagai Integer.

- *Statement If* haus dibuat dalam struktur yang ada. Dalam *VBScript* masih dimungkinkan untuk membuat satu *statement If* dalam satu baris, seperti If x then y. Dalam *Visual basic .NET,* hal ini harus ditulis:

  ```
  If x then
     Y
  End If
  ```

- *Option Explicit.* Aktif secara *default*, artinya semua variable harus di deklarasikan sebelum digunakan.

2.4. ADO.NET (*ActiveX Data Object .NET*)

Pada saat membuat aplikasi dengan berbasis *database*, banyak teknologi yang dpat dipakai untuk menginplementasikannya. *Visual Studio .NET* menawarkan alternatif lain untuk mengakses database, yaitu menggunakan ADO. NET.

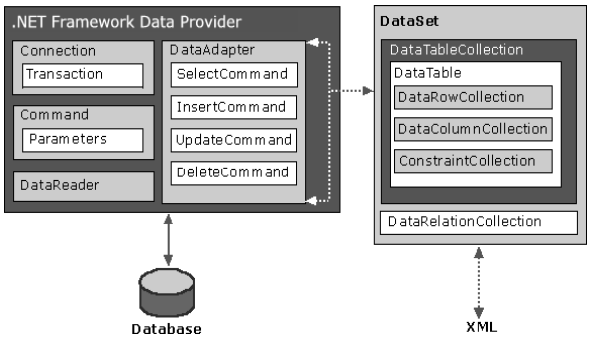
Sedangkan ADO .NET sendiri merupakan pengembangan perbaikan dari ADO yang menyediakan interoperabilitas dan skalabilitas dalam mengakses data. Pemakaian XML ADO .NET dapat menjamin efisiensi transfer data dari suatu aplikasi lain yang berbeda *platform*.

2.4.1. Arsitektur ADO .NET

Ada dua komponen utama dari *ADO. Net* yaitu Data Set dan *.Net Data Provider*. Dimana .Net Data Provider terdiri atas *Connection, Command, Data Reader*, dan *Data Adapter*.

Data Set dari *ADO. Net* adalah komponen utama dari arsitektur *disconnected ADO. Net.* Data Set secara eksplisit didesain untuk akses data yang independen terhadap sumber data. Hasilnya dapat digunakan dengan banyak sumber data yang berbeda, misalnya data *XML* atau untuk data lokal dari aplikasi. Data Set terdiri atas satu atau lebih koleksi obyek Data Table yang masing-masing terdiri atas baris dan kolom data, serta *primary key*, *foreign key*, *constraint* dan informasi relasi data dalam obyek Data Table.

Komponen utama lainnya adalah *.Net Data Provider* yang memiliki komponen-komponen yang didesain khusus untuk memanipulasi data dan cepat. Objek *Connection* menyediakan konektifitas dengan sumber data, Objek C*ommand* mengizinkan akses ke *command* database untuk mendapatkan data, memodifikasi data, menjalankan *stored procedures,* dan mengirimkan atau mendapatkan informasi parameter. Data reader menyediakan aliran data high *performance* dari sumber data*. Data Adapter* menyediakan jembatan antara Objek *Data Set* dan sumber data. *Data Adapter* menggunakan objek untuk mengisi *Data Set* dengan data dan melakukan perubahan data pada *Data Set* dengan data dan melakukan peruabahan data pada *Data Set* kembali ke sumber data.



Gambar 2.3. *ADO.NET Architecture*

2.4.2. Keuntungan ADO .NET

Pada aplikasi database ADO .NET banyak menawarkan keuntungan atau kelebihan dibandingkan metode databse yang lain. Berikut ini keuntungan-keuntungan yang diberikan oleh ADO .NET:

- *Interoperability*

  Aplikasi database *ADO. Net* memperoleh keuntungan dari kompatibilitasnya dengan *platform* lain. Hal ini disebabkan data *ADO. Net* mempunyai *format XML*. Data yang memiliki *format XML,* pola penyebarannya dapat leluasa dalam suatu jaringan maupun oleh komponen database lain.

- *Maintainability*

  Kemudahannya dalam mendesain suatu aplikasi database, baik itu *two-tier* maupun *n-tier*, karena komunikasi datanya menggunakan *format XML.*

- *Programmability*

  Oleh karena *library* dan kelas telah disediakan oleh *Visual Studio .Net,* dari segi cara pemrograman, *ADO. Net* menawarkan kecepatan dalam membuat program.

- *Performance*

  Pada aplikasi *disconnect, ADO. Net* dengan *Data Set*-nya memberikan unjuk kerja yang lebih baik dibandingkan *ADO recordset.* Disamping itu, dalam *ADO. Net* konversi tipe data bukan merupakan hal penting atau bermasalah.

- *Scalability*

  ADO. Net menyediakan skalabilitas yang tinggi dengan *resource* yang kecil sehingga memberikan akses data yang cepat. Hal ini disebabkan *ADO. Net* mengakses *disconnect* data sehingga tidak terus menerus terkoneksi dengan databse dalam waktu yang lama.

2.4.3. Perbandingan *ADO. Net* dan *ADO*

Berikut ini perbandingan antara *ADO* dan *ADO. Net:*

- Penyajian data dalam memori

  Pada *ADO* data dalam memori dalam bentuk *recordset*, sedangkan pada *ADO.Net* semua data dalam memori dalam bentuk dataset.

- Jumlah tabel

  Dalam *recordset* terlihat seperti satu tabel. Jikasebuah recordset berisi banyak tabel, harus dilakukan *join Query*, yaitu data yang mirip akan dijadikan satu dalam satu tabel. Sebaliknya sebuah *dataset* merupakan kumpulan dari beberapa tabel. Tabel dalam *dataset* disebut data tabel atau nama objeknya

adalah *Data Table.* Jika sebuah dataset berisi data dengan multiple database, *dataset* akan mempunyai banyak objek *Data Table.* Setiap objek *Data Table* secara khusus mempunyai hubungan dengan satu database atau *view.*

- Data Navigasi

  Dalam database menggunakan *ADO,* kita melakukan penelusuran data dalam *row recordset* menggunakan fungsi *ADO movenext.* Sedangkan dalam *ADO. Net, row* ditampilkan dalam *collection* sehingga kita dapat melakukan looping melalui *collection* atau *row index ordinal* atau *primary key. Object DataRelation* menyediakan informasi mengenai *master* dan *record* lengkap serta fungsi yang memungkinkan kita mengambil *record* yang berhubungan dengan data yang sedang bekerja.

- Akses Data *Disconnect*

  Pada ADO. Net objek dataset menyediakan akses data disconnect, sedangkan dalam ADO menyediakan juga akses data disconnect, tetapi ini dibuat terutama dalam akses data koneksi.

- *Sharing* Data Antaraplikasi

  Transfer data antaraplikasi dalam *ADO. Net dataset* adalah lebih mudah dibandingkan data akses *disconnect recordset.* Untuk mengirim data *transfer ADO recordset* dari satu komponen ke lainnya memakai *COM Marshalling*, sedangkan menggunakan *ADO. Net dataset* untuk pengiriman data menggunakan *XML.*

2.4.4. Tujuan *ADO .NET*

ADO.NET didesain untuk memenuhi kebutuhan model pemrograman terbaru, yaitu:

- Arsitektur untuk *disconnected* data
- Integrasi erat dengan *XML*
- Reprensentasi data umum
- Kemampuan mengkombinasikan dari banyak sumber data dan berbagai sumber data
- Mengoptimalkan fasilitas interaksi dengan *database*
- Dijalankan dalam *.Net Framework*

- Penggunaannya mirip *ADO* sehingga mudah dipelajari

2.5. *Microsoft SQL Server 2000*

SQL Server 2000 merupakan salah satu *software* RDBMS yang diperditungkan pada saat ini. Perkembangan *SQL Server* 2000 dimulai pada waktu *Microsoft* pertama kali bekerja sama dengan *Sybase* pada tahun 1989 untuk membuat *Sybase SQL Server* di *platform* OS/2. Kemudian, seiring dengan kemajuan Microsoft memasarkan system operasinya, pada versi 4.2, *Microsoft* membuat *SQL Server* untuk *platform Window*s NT. Sampai pada versi 6.0, *Microsoft* memutuskan kerjasamanya dengan *Sybase*. Pada versi 7.0, *Microsoft* memebuat dari awal *SQL Server* dan merupakan versi pertama yang dapat digunakan di *platform Window*s 9.x.

SQL Server 2000 mempunyai tujuh edisi, yaitu *Standard Edition, Enterprise Edition, Personal Edition, Developer* Edition, *Window*s CE Edition, Evaluation Edition dan Microsoft Desktop Engine. Dari kesemua versi yang ada, Enterprise Edition merupakan versi yang paling banyak digunakan dan mempunyai fasilitas terlengkap. Di dalam versi *Enterprise Edition* tidak seperti *vendor database* lain, *SQL Server* 2000 menyediakan fasilitas *Online Transaction Processing* (OLTP) dan *Online Analytical Processing* (OLAP), juga *Data Transformation Service* (DTS) yang berguna untuk mengekspor-impor data.

Banyak fitur-fitur yang baru pada *SQL Server* 2000, berikut ini properties-properties yang dimiliki oleh *SQL Server* 2000:

- *Internet Integration*

  SQL Server 2000 memungkinkan aplikasi integrasi dengan internet dan mendukung system data XML Model pemrograman *SQL Server* 2000 tergabung dalam arsitektur *Window*s DNA untuk membuat aplikasi *English Query* dan *Servis Microsoft Search* secara mudah.

- *Scalability* dan *availability*.

  Dapat digunakan untuk multi *platform* baik mulai dari *laptop* sampai *mainframe*. SQL *Server* 2000 juga dilengkapi dengan fasilitas seperti *federated server*, *indexed views*, dan kapasitas memori yang besar guna

meningkatkan *level performance* yang diperlukan baik aplikasi *desktop* amupun aplikasi *web*.

- *Enterprise-level Database Features*

  SQL Server 2000 sebagai database relational mendukung permintaan pengolahan data yang lebih cepat dan efisien.

- Kemudahan *Instalasi*, *Package* dan Penggunaan.

  Fitur ini jelas dimiliki *SQL Server* 2000, karena dengan kemudahan yang diberikan *SQL Server* 2000 diharapkan dapat mempercepat proses pembuatan aplikasi.

- *Data Warehousing.*

  Oleh karena kemampuan dalam mengolah data, *SQL Server 2000* dapat digunakan dalam menganalisa suatu aliran data yang sedang aktif.

2.5.1.*Stored Procedure*

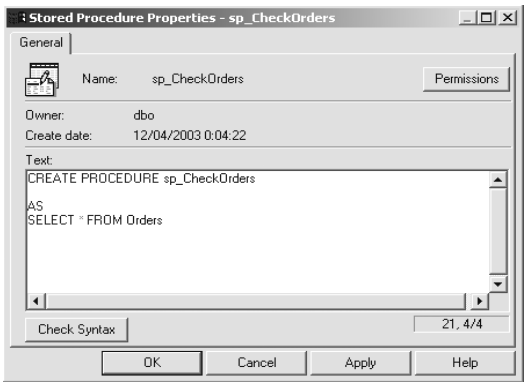
Data di dalam *database* hanya dapat diakses dengan melalui eksekusi perintah Transact-SQL. Pada dasarnya *stored procedure* adalah sebuah program yang ditulis dalam bahasa Transact-SQL dan disimpan di dalam *database* SQL Server. *Stored procedure* dibentuk dari perintah variabel, serta alur logik yang terdapat pada SQL. Sama seperti bahasa yang lain, *stored procedure* ini dapat menerima parameter input (nilai yang dikirimkan kepada prosedur) untuk memprosesnya dan menjalankan beberapa nilai sebagai parameter output kepada program yang menjalankannya. *Stored procedure* dapat memanggil *stored procedure* yang lain. Namun tidak seperti sebuah fungsi, meskipun *stored procedure* menghasilkan parameter output, *stored procedure* tidak akan menghasilkan nilai apapun untuk mengganti namanya dan tidak dapat digunakan di dalam ekspresi.

Setelah *stored procedure* dibuat, ia akan dapat digunakan oleh aplikasi apa saja yang mampu mengakses *database*. Pada saat membuat aplikasi untuk berbagai fungsi sebagai *database interface*, pengembang dapat memilih untuk membuat program SQL, yang disimpan secara lokal dan dikirimkan ke *server* untuk dieksekusi di sana, atau membuat dan memelihara program di dalam *server*

itu sendiri, yaitu berupa *stored procedure-stored procedure* yang dapat diakses oleh program di dalam komputer *client*.

Stored procedure sangat bermanfaat pada lingkungan *client-server*, baik untuk meningkatkan kinerja maupun untuk pemeliharaan. Oleh karena satu buah *stored procedure* dapat digunakan untuk beberapa program, pengelolaan sistem dapat menjadi lebih mudah karena satu perubahan akan segera direfleksikan kepada semua *user*. Setelah prosedur dijalankan di dalam *server*, akses berikutnya akan menjadi cepat karena rencana eksekusi sudah disimpan di dalam *memory*.

Stored procedure dibuat dengan perintah Transact-SQL CREATE PROCEDURE. Proses pembuatan yang sesungguhnya dapat dilakukan dengan *Enterprise Manager*, *Query Analyzer*, atau *Stored Procedures Wizard*. SQL Server 2000 juga memungkinkan kita mengisikan parameter-parameter ke dalam *stored procedure* yang kita bangun. Gambar 2.4 dan 2.5 adalah merupakan contoh pembangunan *stored procedure* dengan dan tanpa parameter.



Gambar 2.4. *Stored Procedure*

Parameter di dalam prosedur harus memiliki nama yang eksklusif dan selalu diawali dengan simbol @, juga harus disertakan tipe datanya. Gambar 2.3. menunjukkan cara membangun *stored procedure* dengan menggunakan parameter.

```
CREATE PROCEDURE sp_ViewProductsByCategory
(
  @CatalogID_1 char(1)
)

AS
SELECT * FROM Products

WHERE CatalogID = @CatalogID_1

ORDER BY ProductName
```

Gambar 2.5. *Stored Procedure dengan Parameter*

2.6. *Web Services*

Selama beberapa tahun, para pengembang telah mencari cara untuk memanfaatkan ulang produk-produk kerja mereka, atau menghadirkan bersama system-sistem terpisah yang telah dmiliki oleh organisasi mereka ke dalam bentuk yang *kohesif*. Pencarian ini telah menghadapi berbagai kesulitan sejak dari *platform* yang berbeda hingga *system* yang tidak memiliki standar untuk komunikasi, sampai ukuran keamanan yang menghalangi *client* potensial dari luar jaringan perusahaan untuk bisa menggunakan funsgionalitas yang ada.

Sebagian dari solusi atas kesulitan ini telah memasukkan pembayaran *front-end web custom* untuk fungsionalitas yang ada, atau menggunakan komunikasi *superior* dan teknologi pemaketan untuk menjembatani kesenjangan antar system. Solusi terdahulu bekerja dengan baik dalam sejumlah situasi, namun organisasi yang menampung aplikasi harus membuat dan menjaga sebuah UI untuk fungsionalitas yang telah mereka sediakan, termasuk membuat sebuah *front-end* untuk emua fungsionalitas tambahan berikutnya. Lebih jauh, UI ini mungkin atau mungkin tidak benar-benar memenuhi kebutuhan *client* mereka yang ada maupun yang akan dating. Penggunaan komunikasi *superior* atau perangkat lunak perantara berarti biasanya pengeluaran yang lebih besar, ketergantungan pada *vendor*, dan tergantung pada perawatan dan *upgrade vendor*. *Web services* berbasis XML secara langsung mengatasi masalah-masalah ini, yang memungkinkan fungsionalitas programatis diekspos tanpa membutuhkan UI berbasis *web*,dan tidak memerlukan perangkat lunak perantara.

Web service merupakan satuan diskrit dari fungsionalitas programatis yang diekspos kepada *client* via protokol komunikasi, dan format data standar bernama HTTP dan XML. Protokol-protokol ini mengatsi masalah komunikasi

lintas *internet* dan lintas firewall perusahaan tanpa beralih ke solusi *superior* yang memerlukan *port-port* komunikasi tambahan yang harus dibuka untuk akses eksternal. Oleh karena *web services* menggunakan XML untuk format permintaan dan jawaban, ia bisa diekspos dari dan digunakan oleh segala platform yang bisa memformat dan menguraikan sebuah pesan XML. Ini memungkinkan *web services* berbasis XML menghadirkan bersama potongan-potongan fungsionalitas berbeda, baik yang ada maupun yang baru, internal maupun eksternal pada sebuah organisasi, dalam suatu kekompakan yang koheren.

Web services diakses dengan memanggil ke pendengar yang bisa memberikan sebuah kontrak untuk menjelaskan *web services* tersebut, dan yang juga memproses permintaan yang masuk atau meneruskan mereka ke logika aplikasi lain dan menyalurkan kembali segala *respons* ke *client*.

2.6.1. SOAP dan *Service*

Protokol SOAP adalah sebuah kunci pengaktif layanan *web* berbasis XML sebagai sebuah startegi untuk integrasi dan layanan –layanan perangkat lunak berbasis internet SOAP, yang merupakan singkatan dari *Simple Object Access Protocol*, adalah sebuah protokol pesan berbasis XML yang distandarisasi oleh W3c (*World Wide Web*) dan dikukung luas dari berbagai *vendor*. Spesifikasi SOAP menyediakan sebuah format wajib untuk pesan-pesan SOAP (disebut dengan amplop SOAP) surat standar opsional untuk pengkodean data, memproses permintaan/tanggapan dan pengikatan pesan-pesan SOAP ke HTTP

2.6.2. Keuntungan menggunakan *Web services*

Dalam implementasi model aplikasi *N-tier*, *web services* dapat dijadikan alternatif. Karena dibangun berdasarkan *text-based document* format XMl dan *standard protocol* TCP/IP, memungkinkan *web services* berkembang pesat dan diadiopsi oleh berbagai *platform*. Salah satu keunggulan utama dari penggunaan *web services* adalah masalah platform independecy. *Web services* yang dibuat dalam platform *Mircosfot*, dapat digunakan oleh aplikasi yang berada dalam *platform* lainnya. Artinya, jika *system* yang akan dibuat perlu mengintegrasikan

beberapa *platform* yang ada, pendekatan menggunakan model *web service* ini sangat tepat.

Web services merupakan alternatif yang dapat digunakan dalam model aplikasi *N-Tier*.fungsi-fungsi yang mendukung business *logic* dari suatu *system* daapt dibuat dalam *web services* sehingga dengan demikian *system* yang dibuat nantinya akan daapt digunakan dalam *platform* manapun, melalui bahasa pemroraman apa pun, bahkan dengan menggunakan *web services* yang sma penggunda dapat menggunakan *device* apa pun untuk berinteraksi dengan system (PDA, WAP, *Smart Phone* dan lain-lain).

Ada banyak *factor* yang perlu dipertimbangkan karena pengguna tidak dapat mengubah langsung aplikasi *e-commerce* berbasiskan *web service, factor* tersebut diantaranya adalah:

- *Web service* tidak mendukung transaksi.

  Dalam *database* dikenal adanya istilah *transaction*, di mana transaksi-transaksi yang dilakukan terhadap *database* dapat dii atau di *rollback* berdasarkan kondisi tertentu. Protokol SOAP sat ini belum mendukung pengguanan transaksi.

- Mekanisme *Callback*

  Web services tidak dapat digunakan untuk melakukan *call back* terhadap *clien*t. Artinya *web service* tidak dapat mengaktifkan *method-method* yang berada di *client.*

- Isu dalam *debugging*

  Ketika terjadi *error* pada saat pemanggilan web method, web service merespons dalam bentuk infromasi kesalahan SOAP. Artinya *error* yang muncul di server tidak sama dengan pesan *error* yang muncul di client. Sebagai contoh, saat pemanggilan suatu *web method* yang berhubungan dengan *database* terjadi *time out*, di *server* kita dapat mengetahui exception yang muncul, namun di sisi *client* hanya menerima *exception Soap Exception.*

- *Performance.*

  Web service membutuhkan suatu infrastruktur jaringan yang cukup baik, dalam hal ini tentu saja *web service* tidak dapat begitu saja diimplementasikan

dalam *system* yang tidak didukung dnegan infrasturktur jaringan yang memadai.

2.6.3. Mendeklarasikan Layanan *Web*

Web service adalam ASP .NET didefinisikan dalam file-file berekstensi .asmx. Kode sebenarmya untuk layanan web ASP.NET bisa ditempatkan dalam file .asmx atau dlam sebuah kelas prakompilasi. Bisa juga, web service dideklarasikan dalam file .asmx dengan menambhakan *direktif @webservice* ke file tersebut. Sintaks untuk mendeklarasikan sebuah layanan web di tempat kelas tersebut akan didefinisikan dalam file-file .asmx adalah:

```
<%@ webService Language="VB" Class="ClassName" %>
```

ClassName adalah nama kelas yang berisi metode-metode *web service.* Sintaks untuk mendeklarasikan sebuah layanan web di tempat kode yang mendefinisikan kelas tersebut berada dalam kelas yang telah dimompilasi adalah:

```
<%@ WebService Class="Namespace.ClassName, AssemblyName" %>
```

Na*mespaceName* adalah nama *namespace* tempat kelas untuk *web service* berada, *className* adalah nama keals yang mengandung metode-metode *web serivce,* dan *AssemblyName* adalah nama *assembly* yang mengandung kode terkompilasi untuk *web service.*

2.7. XML (*eXstensible Markup Languag*e)

XML adalah bahasa markup yang dirancang khusus untuk penyampaian informasi melalui *World Wide Web*, persis seperti HTML, yang telah menjadi bahasa standar untuk membuat halaman web. Walaupun serangkaian elemen HTML terdefinisi telah ditambahkan , HTML masih tidak cocok untuk mendefinisikan sejumlah tipe dokumen. Seperti misalnya:

1. Dokumen yang tidak berisi dokumen biasa, contohnya penandaan skor matematika.
2. Database seperti menyimpan dan menampilkan informasi database yang dinamis dimana HTML hanya bisa menampilkan secara statis.
3. Dokumen dalam bentuk hierarki pohon.

Dengan adanya keterbatasan itu maka dibuatlah XML sebagai solusinya. Karena XML tidak memasukkan elemen yang telah ada, XML tampaknya

menjadi standar biasa secara relatif. XML memiliki sintaks yang didefinisikan langsung. Misalnya, tidak seperti HTML, setiap elemen XML harus memiliki sepasang *tag* awal dan *tag* akhir. Berbagai elemen yang tersarang harus sepenuhnya termasuk dalam elemen yang menutupnya.

Pada sat ini daripada menggantikan HTML dengan XML, lebih baik keduanya dipakai secara bersama-sam sehingga dapat memperluas kapabilitas halaman *web* untuk:

1. Menyampaikan secara *virtual* berbagai macam dokumen.
2. Mengurutkan, menyaring, mencari dan memanipulasi informasi dengan cara yang lain.
3. Menghadirkan informasi yang sangat terstruktur.