CS708 Lecture Notes
Visual Basic.NET Programming
Object-Oriented Programming
Web Technologies and ASP.NET

(Part I)
(Lecture Notes 5B)

Prof. Abel Angel Rodriguez
SECTION I. INTRODUCTION TO WEB APPLICATIONS & TECHNOLOGIES

1.0 Overview

1.1 Introduction

1.2 Web-Base Client/Server Applications

2.0 Web Application Architectures

2.1 Overview

2.2 Standard Web Client/Server Applications Architecture

2.3 Object-Oriented Client/Server Web Applications Architecture

3.0 Introduction to ASP.NET

3.1 ASP.NET

3.2 Important Facts About ASP.NET

3.3 Server

3.4 Client

3.5 Static Web Pages

3.6 Important Facts About ASP.NET

3.7 Server-Side Dynamic Web Pages

3.8 Client-Side Dynamic Web Pages

3.9 Designing a Web Page

4.0 Creating a Web Site Using Visual Studio IDE

4.1 Type of Web Sites available in Visual Studio 2005

4.2 Creating FILE SYSTEM WEB SITE using the IDE

4.3 Visual Studio 2005 IDE Web Components

5.0 Web Form Life-Cycle

5.1 Important Facts about ASP.NET

5.2 ASP.NET Web Page (Web Form) Processing Life-cycle

6.0 Sample Program

6.1 Sample Program #1 – Customer Management Retail Application /MS Access Database

   Example #1 – ASP.NET Small Business Customer Retail Management Application (MS ACCESS version)

3  11  16  25  32
Section I. Introduction to Web Applications & Technologies

1.0 Overview

1.1 Introduction
- Due to the World-wide web, client/server web applications are becoming the norms. With the demands of E-Commerce technologies have been developed to make it easier to create complex web applications.
- In this section we examine these technologies and learn the basics required to create web applications using ASP.NET.

1.2 Web-Base Client/Server Applications
- Let’s review the components for a Web-based Client/Server application:
- Client machines don’t run a client program, but put as little as possible on the client machine.
- Instead of a client program, the client runs an HTML Browser Application such as Microsoft Internet Explorer and Netscape Navigator which handle presenting the data to the user or User-Interface (UI).

Client or Browser:
- There is only one client program, or program which runs on the client machine, which is the Browser application such as Microsoft Internet Explorer and Netscape Navigator.
- This means, programmers don’t have to worry about writing a complex client program and all it’s interaction with the operating system etc. The programmer only concern is writing HTML or other server side code to solve the problem at hand, not other Operating System related programming.
- Programming in HTML is easier; although for complex processing scripting languages are used such as VBScript, JavaScript, ASP.NET, CGI scripting etc (More on this in later). But writing complex combinations of HTML & scripting language is still easier than having to write an entire client application.
- Also, there is not installation of client application involved since there is only one client application, the Browser!

Web Server:
- The client code, actually resides on a Web Server which stores the actual HTML code that are requested by the Browser.
- The job of the web server is to provide the browsers with the requested HTML pages. It is simply an HTML page distributor.
- Writing the basic client program means writing HTML code which will then reside on a Web Server and the server distributes the pages as requested by the client browsers.

Database Server:
- In the web server architecture, database servers are also used to store data requested by the clients.
Characteristics of Web Architecture

- We don’t need Business Objects or Object-Oriented Programming to write an application on the Web Architecture. A matter of fact, originally, web applications were simply an HTML page with text/graphics that is loaded from the server to the client browser. The browser simply translated and displayed the text/graphics.
- But this limits how powerful a Web App can be, since there is no processing on the client, applications cannot perform any processing functionalities and are not powerful. To resolve this problem computer scientist developed the following technologies:

  - **Client-Side Scripting:** Browser built-in *Interpreters* (Translators) for scripting languages such as VBScript and JavaScript for client side processing.
    - **Advantage:**
      - Processing on the client
      - Web application can do business logic and programs can be powerful
    - **Disadvantage:**
      - Learn new language, VBScript or JavaScript
      - Powerful workstation to handle the processing

  - **Server-Side Scripting:** Web Server Built-in *Interpreters* (Translators) and Technology such as *Active-Server Pages* (uses VBScript or JavaScript) and *CGI Script* (can use VBScript, JavaScript, C/C++, Pearl & others).
    - **Advantage:**
      - No Processing on the Client
      - Less powerful Client computer
    - **Disadvantage:**
      - Learn new language, VBScript or JavaScript
      - And, learn a Active Server Pages or worst CGI Script

- These days the Server-Side technology has been in the fore front of the Web development industry. The advantages of being able to have as little processing in the client and allowing the important processing to be done at the server where is closer to the database has made it appealing to write most web applications using Server-Side Scripting.
- Technologies like Active Server Pages & Java Server Side Scripting has grown in popularity

Internet/Intranet

- The Web-Based Client/Server configuration is the standard for the Internet.
- The Internet is a combination of thousands or Web-Based Client/Server architecture all connected together.
- Every time you log on to the Web, you are using a Web Browser to connect to a Web Server that has an address such as Microsoft.com that you enter in the Web Browser.
- The Web-Based architecture has become so popular that industry & governments are now adopting this architecture to develop their internal business applications.
- This lead to the terminology Intranet.
- **Intranet** refers to a Web-Based Client/Server architecture that is private or internal to an organization. It is not public like the Internet.
2.0 Web Application Architectures

2.1 Overview

- In this section I will briefly provide an overview of the types of Web applications usually created.
- We will discuss applications that use where all the processing is done on the Web Forms and Object-Oriented architectures such as multitier object-oriented architecture we have been implementing with our Windows Applications.

2.2 Standard Web Client/Server Applications Architecture

- From the beginning of CS608 I have been preaching the OOP model for creating applications.
- I stated that creating application s the conventional way of putting PROCESSING code on the FORMS is not what we want to do.
- This architecture & diagram looks as follows:

With ASP.NET this approach is actually quite popular. Most books and courses on ASP.NET will simply focus on this time of Web Development.

Does this mean that we don’t have OOP?

Not really, since ASP.NET is a Object-Oriented environment, the WEB FORM, is an instance of an ASP.NET CLASS name PAGE. This PAGE CLASS contains all the properties and methods to create Web Pages. Some of these properties are ASP.NET OBJECTS.

With that said, we have Object-Oriented Programming already embedded into the PAGE.

With that said, we will approach learning ASP.NET and Web Development using a standard Web Form using this architecture at first.

We will understand the Web Form, Web Controls, database connectivity etc., as follow:
2.3 Object-Oriented Client/Server Web Applications Architecture

- Although the previous architecture is suitable for situations where you need to create a Web Application fast and easy, large scale application required a better more scalable architecture.
- After we understand how ASP.NET web application work, then we will apply our SCALABLE, Client/Server application architecture model that we used for our WINDOWS APPLICATIONS:

![Diagram of Architecture](image)

- Using this architecture we will encapsulate ALL PROCESSING, VALIDATION & DATA ACCESS into BUSINESS OBJECTS.
- This will allow us to create the following WEB application implementations:
Business Objects Placement in STANDARD Web Based Client/Server

A standard Web Client/Server architecture has the 3 basic Web components, Browser, Web Server & Database:

1. Write the Business Objects in VB.NET, C#, C++, or Java.
2. Package the Business Objects in a Class Library or DLL COMPONENT, enable Remoting & Serialization.
3. Have ASP.NET Server Side Technologies in the Web Server interact with the Business Objects in the DLL.
4. Place the Business Objects as follows:
   - In a Web-Based configuration, the User-Interface (UI) is the Web Browser on the client, and the UI-Business Objects and the Data-Service Objects that handle database access resided in the Web Server. As usual, the Data Service layer or database itself resides on the Database Server.
Business Objects Placement in N-TIER Web Based Client/Server
- As the demand for faster processing and the eCommerce growth, the Web-Based Client/Server architecture has evolved into a combination of the Three-Tier & Web-Based Client/Server to form the n-tier Web Based Client/Server Architecture.
- In this architecture, the Application Server Tier is added, thus yielding all the benefits of managing faster connection, business object management etc.

Application Server:
- The application server is used to manage the connections and data access to the database servers.
- This server has all the advantages listed for the Three-Tier Client/Server configuration.
- You can add several Application Servers or Server Farms for load balancing.
In this configuration, the **User-Interface (UI)** is the Web Browser on the client, and the User Interface Related Business Objects or **UI-Centric Business Objects** are kept at the Web Server closer to the HTML code.

On the other hand, the **Data-Centric Business Objects** are placed and managed in the Application Server.

As usual, the **Data Service layer** or database itself resides on the Database Server.
3.0 Introduction to ASP.NET

3.1 ASP.NET
- In the infancy of web technology web pages were static
- Static web pages refer to HTML pages in which the content of the page is determined before the request is made. The output is always the same and never changes.
- The diagram below illustrates communication between browser and web server to generate static pages

3.2 Important Facts About ASP.NET
- In the infancy of web technology web pages were static
- Static web pages refer to HTML pages in which the content of the page is determined before the request is made. The output is always the same and never changes.

The diagram below illustrates communication

ASP.NET is Object-Oriented
- Development work can be done in terms of OBJECTS & EVENTS
- (Create object & Use them (Set & get proper…))
3.1 Static Web Pages

- In the infancy of web technology web pages were static.
- Static web pages refer to HTML pages in which the content of the page is determined before the request is made. The output is always the same and never changes.
- The diagram below illustrates communication between browser and web server to generate static pages.

**Advantages/Characteristics**
- Easy to program (HTML only)
- Fast, quick to copy small file over network.

**Disadvantages**
- Output always the same
- Content is completely determined before the page is requested.
- Cannot personalize the web page to the user profile
- No security
- Code is available to anyone who wishes to copy and use it for themselves
- In short, not dynamic!

---

**Diagram Description**

1. **Browser request web page from Web Server**
2. **Web Server locates web page**
3. **Web Server sends HTML stream from web page to Browser**
4. **Browser Process HTML and displays page to user**

0. **Web Developer creates web page using HTML**
3.2 Client-Side Dynamic Web Pages

- To work around the limitations of static web pages, technology was provided to the browser in the form of plug-ins, such as Translators for scripting languages such as **VBScript** and **JavaScript** for client-side processing.
- This technology is known as Client-Side scripting and provided programmers with the ability to make the static web pages into dynamic web pages.
- The diagram below illustrates communication between browser and web server to generate client-side dynamic web pages:

<table>
<thead>
<tr>
<th>Advantages/Characteristics</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Generated dynamically upon request</td>
<td>• Requires plug-in or Interpreter to be installed, version control etc.</td>
</tr>
<tr>
<td>• Output not the same, can be personalized, security etc.</td>
<td>• Fat-client</td>
</tr>
<tr>
<td></td>
<td>• Difficult to program (Must learn VBScript or JavaScript)</td>
</tr>
<tr>
<td></td>
<td>• HTML &amp; Instruction code sent to browser</td>
</tr>
<tr>
<td></td>
<td>• Network bandwidth limitations, take too long to download</td>
</tr>
<tr>
<td></td>
<td>• Each browser type interpret code differently, so you may need to program</td>
</tr>
<tr>
<td></td>
<td>for different versions of browsers.</td>
</tr>
<tr>
<td></td>
<td>• Since interpreted in client, issues communication with</td>
</tr>
<tr>
<td></td>
<td>using database server etc.</td>
</tr>
<tr>
<td></td>
<td>• Code not secured, can be seen in the client by anyone.</td>
</tr>
</tbody>
</table>
3.3 Server-Side Dynamic Web Pages

- Server side scripting provides the flexibility and dynamic requirements of the web-base client/server architecture.
- The diagram below illustrates communication between browser and web server to generate Server-side dynamic web pages:

**Programmer** creates web page using **HTML** code and **Instructions (code)** written in programming Language (VB.NET, C#.NET, etc.)

---

**Advantages/Characteristics**

- Generated dynamically upon request
- Server-side processing
- Output not the same, can be personalized, security etc.
- Only HTML sent to browser, browser type no longer an issue
- Fast, quick to copy small file over network.
- Original code is safe and secure in server; user cannot see the source code.
- Easier to communicate with database servers etc.

**Disadvantages**

- Difficult to program (Must learn VBScript or JavaScript or VB.NET, or C#)
- Must understand ASP.NET, and .NET Framework.

---

**Diagram Description**

1. **Browser** request web page from **Web Server** via **URL**
2. **Web Server** locates HTML web page & **Instructions (code)** file
3. **Web Server** process HTML & **Instructions** and converts to pure HTML
4. **Web Server** send pure HTML stream to **Browser**
5. **Browser** process HTML and displays page to user
3.x Designing a Web Page

- In this section we provide a brief overview of the process required to create and use WEB FORM.

Thinking Object-Oriented

- If you have been keeping up with my method of teaching programming, you should have the following in mind:

  - ASP.NET Web Forms are Objects, therefore our 3 step rule for Object-Oriented program still holds:
    1. Create Class
    2. Create Object
    3. User Object (Get/Set Properties, call method, program event-handlers etc.)

  - We can modify these rules to fit Web Development as follows;
    1. Create Web Page Class – When you create a WEB FORM, you are actually working with an inherited class from the Microsoft’s PAGE CLASS, which is located in the System.Web.UI.Page Namespace
    2. Create Object – When the web application is running, an OBJECT of the WEB PAGE is created by ASP.NET.
    3. User Object – As you web page is running, you CODE WILL USE THE WEB FORM PAGE OBJECT by GETTING/SETTING PROPERTIES, calling METHODS, executing EVENT-HANDLERS etc.

User-Interface and Code (Graphical & Visual)

- As with WINDOWS APPLICATIONS, WEB APPLICATION have a GRAPHICAL aspect or what the user sees (Web Form), and there is the CODING portion which performs the processing and works behind the scenes.
- In WINDOWS APPLICATION, we only dealt with TWO aspects, GRAPHICAL AND CODE, in WEB FORMS, there are actually THREE:
  1. GRAPHICAL – What the user sees (Web Form, Controls etc.)
  2. HTML SOURCE CODE – What is being processed by the BROWSER
  3. VB.NET or C# Code – PROCESSING CODE EXECUTED ON THE SERVER. This code can be kept in a Code-behind file or embedded into the HTML code)

- To create a Web Page, you simply perform the following:
  1. Create a Web Application using Visual Studio 2005
  2. The application will contain a DEFAULT WEB FORM
  3. Design your interface by ADDING user-interface CONTROLS to the WEB FORM
  4. Write VB.NET or C# CODE to support the WEB FORM.
  5. Run the Web Application
4.0 Creating a Web Site Using Visual Studio IDE

4.1 Type of Web Sites available in Visual Studio 2005

- As previously discuss, a web application is composed of three major components:
  1. Browser
  2. Web Server
  3. Database Server

- The second component is the WEB SERVER or the host which stores the web pages

- Visual Studio 2005 provides 3 types of Web Sites to HOST your WEB PAGES, these described as follows:

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Description</th>
</tr>
</thead>
</table>
| **File System** | - WEB SITE & WEB PAGES are managed and executed under the ASP.NET Development WEB SERVER  
- This Dev Web Server stores the entire site in a FILE  
- Not managed by Windows Operating System  
- This server is simple, flexible and ideal for class room and lab environment  
- Since is in a file, it is portable. |
| **HTTP** | - This method uses an actual Production Web Server, such as Microsoft Internet Information Server (IIS)  
- This is a professional-quality WEB SERVER, which requires security management, settings, permissions etc.  
- This type of server is more complex to manage.  
- Nevertheless, it already available in the Windows Operating System and ready to use  
- This is NOT A FILE, IT IS NOT EASILY PORTABLE. A web site you create on your home computer would have to be recreated in another computer, say school lab etc. |
| **FTP** | - WEB SITE & WEB PAGES are stored in a REMOTE SERVER or REMOTE WEB SITE using FTP (File Transfer Protocol)  
- Requires the username/password, server & port of the target machine.  
- Visual Studio will actually create the Web Site in the remote server once the required credential and information is provided |
4.2 Creating FILE SYSTEM WEB SITE using the IDE

- You should be familiar with using the IDE to create Windows Application. Now we follow the same road-map in order to create our web applications
- The steps to using the IDE to program our applications as follows:

**Step 1: Open the Visual Studio IDE and invoke the Start Page:**

- The screen looks as follows:

![Visual Studio IDE Start Page](image)

- The “New Web Site screen will display as follows:

![New Web Site Screen](image)
Step 2: In the New Web Site Dialog Create the Web Site & Location

To create your Web Site follow these steps:

a). In the Templates box select: “ASP.NET Web Site”
b). In the Template box select: “File System”
c). Click “Browse” button and select the path
d). Select the DEFAULT Language you will use for the CODE. You can set a default, but can still use other language

clicking the Browse button will allow you to navigate and create a folder for your project

e). Browse to the desired location
f). Click the “Create New Folder ICON” on the top right-hand side of the ‘Choose Location” screen.
g). Name the folder
h). Click Open button to return to the “New Web Site” screen
i). Click OK

Step a – Web Site type

Step b – Select Location or Type of Web Site. In this case File System

Step c – Browse to Project Path or target file Location

Step d – Select default Programming language

Step e – Click to create NEW FOLDER

Step f – Name the NEW FOLDER

Step g – Click Open button
Step 3: The IDE Main Screen.

- This screen is where you will create all your Web Form and develop your site.

Web Site Local File Structure

- Our Website looks as follows in the file system:
4.3 Visual Studio 2005 IDE Web Components

- In this section we take a look at the components available via Visual Studio 2005 to create Web Applications.
- You will notice that some of these components are similar to the Windows Applications you created before.

Main Web Application Components

- This screen is composed some basic Window items such as **Title bar, Menu bar, Menu Toolbars** and a **Status Bar**, like most windows application.
- In addition the same application structure that are important for WINDOWS APPLICATIONS ARE USED FOR WEB APPLICATIONS such as:
  - **Document Window** – Area where Web Forms is designed
  - **Solution Explorer Window** – List the FILES, FOLDERS, WEB FORMS, CLASSES etc., in the Web Application
  - **Properties Window** – Allows you to SET properties for the Objects (elements) in the Solution Explorer of a Control on a Web Form in the document window.
  - **Toolbox** – Contains the graphical controls to design and build your web pages
  - **Server Explorer** – Allows you to manage other services on the local computer or network, such as SQL Databases, Event logs, Services, etc.

- The major components are shown below:
### Solution Explorer & Main Application Files

- The Solution Explore contains a listing of all the files, web forms or page and objects which make up our applications:

#### Web Application Solution

- Default Web Page
- Code behind File for
- Web Configuration File

#### Main Application Files

- There are several main application files we need to discuss:

<table>
<thead>
<tr>
<th>File</th>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default.aspx</td>
<td>.aspx</td>
<td>This is the DEFAULT WEB PAGE AUTOMATICALLY ADDED TO YOUR APPLICATION</td>
</tr>
<tr>
<td>Default.aspx.vb</td>
<td>.aspx.vb</td>
<td>This is the CODE-BEHIND file automatically created for the Default.aspx page</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Every Web Form will contain an associated code-behind file with .vb extension</td>
</tr>
<tr>
<td>Web.config</td>
<td>.config</td>
<td>XML-configuration file for the entire application. Here you configure settings for the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Database connectivity (connection string)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Customizing security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- State management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Memory management etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- This file may not be available when you create a web application, if so, created manually</td>
</tr>
<tr>
<td>Global.asax</td>
<td>.asax</td>
<td>Global application file. Here you can declare global variables, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This file includes GLOBAL EVENT-HANDLERS, which you can use to execute code for global events such as:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Application_Start() – Executes when application starts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Application_End() – Executes just before application ends</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Application_Error() – Executes when an unhandled exceptions occurs.</td>
</tr>
</tbody>
</table>
Document Window & Available Views

- The document window is the location where you will design and create your Web Form.
- Visual Studio provides server designers and views to work with the various FILES in the SOLUTION EXPLORER.
- We will now discuss three important views:

1. **Design View** – Visually design and create your WEB FORM using CONTROLS from the Toolbox.
   - There are three available methods to display Design View:
     1. **[Option 1]** Click Design tab for DESIGN View
     2. **[Option 2]** Double-Click.aspx File in Solution Explorer for DESIGN View
     3. **[Option 3]** Click View Designer button to display DESIGN View
2. **Source or HTML View** – Directly edit your WEB FORM HTML code:

   ![Image of HTML View]

   - Click Source tab for SOURCE

3. **Code Editor View (Code-behind View)** – Editor where you will WRITE CODE (VB.NET, C#, etc.) to support the user-interface WEB FORM. This code is places in the CODE-BEHIND FILE with .aspx.vb extension.

   ![Image of Code Editor View]

   - [Option 1] Double-Click .aspx.vb File in Solution Explorer for CODE View
   - [Option 2] Click View Designer button to display CODE View
Toolbox

- The Toolbox contains the various types of controls for your application.
- Toolbox is divided into several sections, examples:
  - Standard: Contains all ASP.NET CONTROLS (Run on Server)
  - HTML: Contains the classic HTML controls (Run on client only)
  - Data: Contains CONTROLS to perform data access or connect to database.

Server Explorer

- Server Explorer Windows provides an interface for your application to connect to SERVICES in your local or network computer such as DATABASES, FILES, Services, etc.
- Server Explorer not only lets you have access to these services but can allow you to CONFIGURE & MODIFY the services.
- For example, you can do the following with an SQL Server database just as if you were doing it with the SQL Server Management Console Enterprise Manager:
  - Create database
  - Connect to a database
  - Create tables, relationships etc
  - Create & Execute queries, stored procedures etc.

- In summary, you can work from your Visual Studio 2005 IDE to create your application and create & manage the DATABASE.
5.0 Web Form Life-Cycle

5.1 Important Facts about ASP.NET

- Now we will analyze one the most important aspects of ASP.NET
- Before we look into how to process a WEB PAGE, we need to understand some important properties of the ASP.NET technology and the communication rules between the BROWSER & the SERVER.

WEB FORM EXECUTE ON SERVER

- Web Form execute on Server only
- User performs the task in the BROWSER, but the CODE EXECUTES ON THE SERVER
- User actions happen on the client or BROWSER, PROCESSING of these actions happen on the WEB SERVER
- WEB FORMS ARE RE-CREATED WITH EVERY ROUND TRIP.

EVENT-DRIVEN & Server Side Execution

- ASP.NET is event-driven, everything in your web-page is executed in RESPONSE TO AN EVENT.
- Because EVENTS are initially requested in the BROWSER, but executed on the SERVER, there is a delay or lag in the process of triggering events and reacting to them.
- WEB APPLICATIONS are not as ROBUST AS WINDOWS APPLICATIONS when it comes to EVENT-DRIVEN PROGRAMMING.
- WEB APPLICATIONS actually SIMULATE EVENTS-DRIVEN PROGRAMMING due to the limitations of the WEB PAGE LIFE-CYCLE and a process called POSTBACK (More on this below)

WEB APPLICATIONS ARE STATELESS (VIEW STATE)

- Web Page lifetime only exist for ONE PAGE REQUEST
- WEB SERVER DISCARDS ALL WEB PAGE INFORMATION after rendering and sending to the BROWSER THE HTML PAGE
- DISCARDING SERVER RESOURCES after each request ensures that the WEB SERVER can support hundreds or thousands of REQUESTS WITHOUT BOGGING DOWN.

VIEW STATE & WEB CONTROLS

- The state or content of all Web Controls are stored or kept via VIEW STATE mechanism on the SERVER
- Form Data is stuffed into a HIDDEN PROPERTY, ENCRYPTED and SENT TO SERVER.
- View State enabled by default
- View State can be granular (Form-level or Control-level)
- VIEW STATE only works with WEB PAGE & WEB CONTROL PROPERTIES

Final Comments on VIEW STATE

- Using VIEW STATE mechanism is a great solution, because server resources are discarded and server is freed to handle other requests.
- Nevertheless, there is a price:
- Because form data or VIEW STATE is stored on the WEB FORM, it results in a LARGER WEB PAGE SIZE
- BROWSER receives a LARGE PAGE.
- BROWSER has to package & SEND THE HIDDEN-VIEW STATE BACK to SERVER with the NEXT POSTBACK
- Therefore with LARGE PAGES, it TAKES LONGER TO SEND/RECEIVE
POSTBACK

- PAGE and ALL USER-SUPPLIED INFORMATION IS SENT BACK TO SERVER when certain action in the BROWSER is performed
- Round trip from BROWSER to SERVER to BROWSER in response to a WEB FORM being submitted to Web Server (after the page has been displayed in browser for the first time.)
- All Data in controls are sent to server via View State
- Initiates when user clicks a BUTTON in the Client, or trigger a WEB CONTROL, example check box etc.

POSTBACK & PAGE OBJECT PROPERTY

- The PAGE CLASS, contains a BOOLEAN PROPERTY named IsPostBack.
- This is a very important PROPERTY for programming in ASP.NET.
- The value of this property is as follows:
  - FALSE: NEW PAGE REQUEST (First time the Page is Loaded? First time page is displayed by browser
  - TRUE: Postback has occurred, Web Form has been sent to server, server has perform the required operation, Web Form sent back to Browser.

POSTBACK & WEB CONTROLS

- All Web Controls have a PROPERTY named AUTOPOSTBACK. Set to FALSE by Default
- If this AutoPostBack is set to TRUE on a CONTROL, the control will initiate a POSTBACK.
- With this PROPERTY SET TO TRUE, changes to controls such as CheckBox, ListBox, TextBox, etc, can generate POSTBACK which will allow the WEB SERVER to execute EVENT-HANDLERS FOR THE CONTROLS.
5.2 ASP.NET Web Page (Web Form) Processing Life-cycle

- Now we will analyze one of the most important aspects of ASP.NET.

Processing Life-Cycle for First Request for a Web Form

- Let’s analyze in detail what happens when the USER makes the first request for a WEB FORM.
Detailed explanation of steps 1 – 5:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action by Web Server or Browser</th>
<th>Reaction by Web Server (ASP) or Browser</th>
</tr>
</thead>
</table>
| 0    | Web Developer Creates Web Form, with Server Side Controls & Code behind-file                    | • Developer copies ASP.NET Web Form to WEB SERVER.  
• WEB FORM is now stored in Web Server for retrieval by BROWSERS                                      |
| 1    | Browser or Client makes request for Web Form                                                    | • Request is sent to server via **URL**                                                                 |
| 2    | Web Server locates and initializes the Web Form                                                 | • PAGE RUNS FOR THE FIRST TIME  
1. Server Receives Request via **URL**  
2. Server locates the Web Form  
3. Web Server CREATES OBJECTS of Web Form  
   (instantiates Web Form object & associated objects)                                           |
| 3    | Web Server Process Web Form                                                                     | 1. Page Events are fired:  
   - **Page_Load**  
   - Other initialization events  
2. RENDERS or GENERATES an **XHTML** PAGE from WEB PAGE OBJECT  
3. SENDS pure **XHTML** to Browser  
4. DESTROYS WEB PAGE OBJECTS                                                                     |
| 4    |                                                                                                 | • PURE HTML in route to Browser                                                                        |
| 5    | BROWSER displays WEB FORM to user                                                                | • HTML is processed by BROWSER and displayed to User                                                    |
Let’s analyze in detail what happens when the USER triggers a POSTBACK and makes the second request for an UPDATED WEB FORM:
### Detailed explanation of steps 6 – 12:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action by Web Server or Browser</th>
<th>Reaction by Web Server (ASP) or Browser</th>
</tr>
</thead>
</table>
| **6** | User clicks a button or does something to trigger POSTBACK | • BROWSER INITIATES A POSTBACK  
1. User information is packed into VIEW STATE hidden property  
2. Web Page is sent to WEB-SERVER including VIEW STATE DATA |
| **7** |  | • Web Form is in route to server  
• Includes FORM DATA from all WEB CONTROLS |
| **8** | Web Server receives the MODIFIED WEB FORM DATA from BROWSER . | • PAGE RUNS FOR THE SECOND TIME  
• PAGE INITIALIZED AND CREATED TO SAME STATE THE LAST TIME SENT TO CLIENT:  
1. Locates the ORIGINAL Web Form on the server  
2. Web Server CREATES OBJECTS of Web Form (instantiates Web Form object & associated objects)  
3. Web Form is in state when FIRST REQUESTED  
4. LOADS VIEW STATE information from memory to a COLLECTION  
5. DESerializes VIEW STATE Collection data & UPDATES ALL CONTROLS with VIEW STATE data.  
6. WEB PAGE is now in same STATE as it was when SENT TO BROWSER the LAST TIME (before POSTBACK) |
| **9** | Server executes or begins POSTBACK process | • POSTBACK PROCESS IS EXECUTED IN SERVER:  
1. MODIFIES WEB PAGE with POSTBACK DATA from BROWSER  
2. WEB PAGE reflects the CURRENT STATE AS IT APPEARS TO USER in BROWSER  
3. EVENTS for all WEB CONTROLS are fired and processed  
4. ASP.NET TRIGGERS THE APPROPRIATE EVENT HANDLERS fired:  
   - Page_Load Event is triggered  
   - Other WEB CONTROL EVENTS  
5. Server Side operations can take place (updating database, reading data from file, move to new page, etc.)  
6. POPULATE WEB CONTROL OBJECTS with NEW DATA from server side operations (database, file etc.)  
7. ASP.NET examines all PROPERTIES of all WEB CONTROLS and verifies if it has changed from its INITIAL STATE.  
8. SAVES VIEW STATE information (Web Control’s data) to a Key/Value COLLECTION  
9. ENCRYPTS & SERIALIZES the data from Collection to a STRING and APPENDS to VIEW STATE HIDDEN FIELD |
<table>
<thead>
<tr>
<th>Step</th>
<th>Action by Web Server or Browser</th>
<th>Reaction by Web Server (ASP) or Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.</td>
<td>Renders or generates a pure HTML page from the web page</td>
<td>10. Renders or generates a pure HTML page from the web page</td>
</tr>
<tr>
<td>11.</td>
<td>Sends pure XHTML to browser</td>
<td>11. Sends pure XHTML to browser</td>
</tr>
<tr>
<td>12.</td>
<td>Destroys web page objects (including web form, controls etc.)</td>
<td>12. Destroys web page objects (includes web form, controls etc.)</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>NEW MODIFIED PURE HTML in route to browser</td>
</tr>
<tr>
<td>11</td>
<td>BROWSER displays NEW MODIFIED WEB FORM to user</td>
<td>NEW HTML is processed by BROWSER and displayed to User</td>
</tr>
<tr>
<td>12</td>
<td>User clicks a button or does something to trigger POSTBACK</td>
<td>BACK TO STEP 6, POSTBACK PROCESS REPEATS</td>
</tr>
</tbody>
</table>
6.0 Sample Program

6.1 Sample Program #1 – Customer Management Retail Application /MS Access Database

- We now implement our Single-Tier Client/Server example of a Small Business Customer Management Application as a WEB-BASED CLIENT/SERVER APPLICATION.

**Single-Tier Client/Server**

- The requirements for Sample program #1. are as follows:

**Example #1 – ASP.NET Small Business Customer Retail Management Application (MS ACCESS version)**

**Problem statement:**
- Implement the Small Business Customer Management application as a WEB APPLICATION using ASP.NET
- The requirements are as follows:

**Application Architecture – Programming Methodology**
- We will NOW implement the 4-tiered layer Web-based Application Architecture:

**Client/Server Architecture – Web-based Client/Server**
- This architecture looks as follows:
Presentation Layer – Client Process req

- NO NEED TO CREATE A CLIENT PROGRAM, THE CLIENT IS THE BROWSER
- No coding or configuration required at this time

User-Interface Layer (THIS IS WHERE MOST OF THE PROGRAMMING WILL TAKE PLACE)

- The user-interface code is the XHTML & ASP.NET CODE needed to create our WEB PAGE
- We will create XHTML and ASP.NET code using VB.NET to implement our WEB FORM

New requirement for User-Interface Web Form

- In the previous Windows Application, we used a LISTBOX to display all customers
- In this WEB version, our WEB FORM will use GRIDVIEW control to display the customer records from DATABASE

Business Object Layer Requirements

- This code HAS ALREADY BEEN IMPLEMENTED IN THE PREVIOUS EXAMPLE
- Keep the DLL COMPONENT, Object Model, Business Rules & DATA ACCESS CODE

NEW Requirements for Business Object Layer

- We will be implementing DATABINDING in order to POPULATE THE GRIDVIEW CONTROL.
- Here is the issue:
  
  - Databinding is usually done by BINDING DATASETS, DATAREADERS or DATATABLES to a GRIDVIEW
  - Nevertheless, you can also BIND COLLECTIONS as well.
  - SINCE NONE OF OUR DATA ACCESS CODE RETURN DATASETS, DATAREADERS OR DATATABLES, WE WILL USE OUR COLLECTION (clsCustomerList) TO BIND TO THE GRIDVIEW CONTROL
  - UNFORTUNATELY, the .NET DATABINDING MECHANISM DOES NOT SUPPORT BINDING OF DICTIONARY TYPE COLLECTIONS!
  - BINDING ONLY WORKS FOR ILIST TYPE COLLECTION OR ARRAYS
  - THEREFORE, WE CANNOT USE clsCustomerList TO DIRECTLY BIND TO THE GRIDVIEW CONTROL

- Solution or workaround:
  
  - We will implement A METHOD INSIDE THE COLLECTION CLASS, this method will RETURN A CONVERTED VERSION OF OUR DICTIONARY COLLECTION AS AN ARRAY!!
  - So we will convert the collection to an ARRAY and return this ARRAY so we can BIND IT TO THE GRIDVIEW
  - The code looks as follows:

```vbnet
'****************************************
'Function returns array version of the Dictionary collection
'Note this function returns an ARRAY OF CUSTOMER [clsCustomer()]
Public Function ToArray() As clsCustomer()
  'create empty array of class objects. Size is obtained from collection
  Dim arrCustomerList(MyBase.Dictionary.Count - 1) As clsCustomer
  'Copy Dictionary Collection to Empty array
  MyBase.Dictionary.Values.CopyTo(arrCustomerList, 0)
  'Return populated array
  Return arrCustomerList
End Function
```

Data Service Layer – Database Requirements

- Continue to use the smallbusinessapp.mdb database
HOW IT’S DONE:

Part I – Create The Web Application:

Web User-Interface Layer

Implementing Application as a Web Application using ASP.NET & XHTML

- We are now ready to add the a WEB FORM and add our ASP.NET CODE.
- At this point, the Customer Retail Management Application has been upgraded to contain all the Business Logic, Validation & Data Access methods with ADO.NET code.
- Our application is a full Single-tier Client/Server application managing an MS ACCESS DATABASE.
- Using Visual Studio 2005, we will upgrade our application by doing the following:
  - Create Web Site
  - Create Web Form
  - Design and implement using ASP.NET
  - Test the application

Step 1: Open the Customer Management Application with BO & Data Access Code

- Open the last version of the Customer Management Retail Application.
- This version contains all the Business Rules, Validation, Data Access methods, and Data Access Code to continue to save and load from an Access Database.
- When you open the application, the application will contain the DLL & Windows Client Projects as follows:
Another view with all Forms closed:

**Step 2: Add a FILE Web Site to Existing Project**

- We add a web project to our existing application. No need to start from scratch, simply add the Web Project and our solution contains both type of clients.
  
a) In the main menu, select File|Add|New Web Site... To invoke the Add New Web Site screen
b) In the Templates box select: “ASP.NET Web Site”, in the Template box select: “File System”
c) Select Visual Basic as the DEFAULT Language
d) Click “Browse” and browse to the desired location

e) Click the “Create New Folder ICON” on the top right-hand side of the ‘Choose Location’ screen

f) Name the folder “WebAppClient”

g) Click Open button to return to the “New Web Site” screen

h) Click OK in Add New Web Site screen to create the web site

- Click “Browse” and browse to the desired location
- Click the “Create New Folder ICON” on the top right-hand side of the ‘Choose Location’ screen
- Name the folder “WebAppClient”
- Click Open button to return to the “New Web Site” screen
- Click OK in Add New Web Site screen to create the web site

a) Click OK in Add New Web Site screen to create the web site & the IDE will display as follows:

- Note we now have three project: DLL, Windows Client & new WEB CLIENT
- Project will automatically include the DEFAULT.ASPX WEB FORM, Code-behind file default.aspx.vb & Web.config

Web Project
Step 3: Set Web Project as Startup Project

- Currently it is the Windows Client or WinAppClient that is the STARTUP PROJECT, thus running or controlling this application.
- Since we are now managing this application as a web application we need to set it as the STARTUP PROJECT as follows:

  a) In the Solution Explore, right-click on the Web Project WebAppClient
  b) In the menu select "
  c) The Web Project will now be IN BOLD
Step 4: [Optional] Set Web Project for Debugging

- When you try to compile, you may get a dialog box when executing the project that will ask you if you want to turn on debugging:

![Debugging Not Enabled dialog](image)

- This will enable the debugging mechanism in Visual Studio for this project, that is you can SET BREAKPOINTS, etc..
- It is in the WEB.CONFIG file in a TAG name `<compilation debug>` where debugging is turned ON/OFF. This dialog will actually modify the Web.config file.
- Initially, the TAG looks as follows in the Web.config file:

```xml
<compilation debug="false" strict="false" explicit="true" />
```

- After this dialog executes it looks as follows:

```xml
<compilation debug="true" strict="false" explicit="true"/>
```

- NOTE THAT YOU CAN MAKE THIS MODIFICATION YOURSELF AND YOU WON’T GET THE DIALOG BOX!
Step 5: IMPORTANT! Add Reference to DLL Component

- Now we need to tell the Web Project where the classes it will need are located.
- We need to REFERENCE OR POINT to the DLL PROJECT:

  a) In the Solution Explore, right-click on the Web Project **WebAppClient**
  b) In the menu select “Add Reference…”
  c) This will invoke the “Add Reference” screen
  d) Click the “Project” tab and select the **BusinessObjects** DLL
  e) Click OK

![Add Reference Screen]

Step 6: IMPORTANT! Copy Database File & Printer File to the Root of the Solution

- The Web Client will automatically search for any Database File or text file in the ROOT DIRECTORY IT WAS CREATED IN.
- Since our program requires the ACCESS MDB file & the PRINTER file for Customer Printing, we need to COPY/PASTE these files to the root location
- Currently these files exist in the WINDOWS CLIENT PROJECT WinAppClient\BIN\DEBUG\ folder
- Do the following:

  a) Navigate to the folder **WinAppClient\Bin\Debug\**
  b) And SELECT COPY the database file **smallbusinessapp.mdb** & **CustomerPrinter.txt** files:

  ![Navigate to Folder]

  c) PASTE these file in the Root directory of the SOLUTION \ASPAccessDBSmallBusinessApp:

  ![Paste Files]

- **SOLUTION ENVIRONMENT IS CONFIGURED, WE ARE NOW READY TO BEGIN TO DESIGN OUR WEB FORMS!**
Step 7: RENAME Default.aspx, to create the Main Web Form

- Instead of adding a new form and deleting the Default.aspx, we will rename this form, to see how that process works.
- We need to rename three items: 1) The Web Form file name (Solution Explorer), 2) HTML Source Code & 3) the Class Name in the aspx.vb Code-behind file:

a) In the Solution Explore, right-click on the Web Project WebAppClient
b) In the menu select “Rename” and rename your Default.aspx to Main.aspx

c) Double-click the file, to open the newly renamed Web Form
d) Click the Source button to switch to SOURCE VIEW, in the XHTML code the first line looks as follows:

```html
@ Page Language="VB" AutoEventWireup="false" CodeFile="Main.aspx.vb"
Inherits="Main" %>
```

e) MODIFY by renaming the value that indicates the name of the code-behind file: CodeFile = Default.aspx.vb & the value that indicates the class name Inherits="_Default" as follows:

```html
@ Page Language="VB" AutoEventWireup="false" CodeFile="Main.aspx.vb"
Inherits="Main" %>
```

f) Finally we need to open the code-behind file (double-click on Main.aspx.vb in Solution Explorer):

```vbnet
Partial Class _Default
    Inherits System.Web.UI.Page
End Class
```

g) Rename from _Default to Main:

```vbnet
Partial Class Main
    Inherits System.Web.UI.Page
End Class
```
Step 8: Design the Main Page

- Begin to design what your main page will look like.
- Since we are trying to recreate the WINDOWS APPLICATION AS A WEB APPLICATION AS CLOSE AS POSSIBLE, currently the Windows version looks as follows:

![Main Page Screenshot]

- I will use the following WEB controls:
  - HTML table – To align controls
  - One ASP control BUTTONS – for Customer Management
  - One ASP control BUTTONS – for Retail Management

- Note that we don’t need an EXIT button because this is a web form in a BROWSER, we just close the browser.
- Design your WEB FORM as follows:

  a) Open the Main.aspx Form and at the top type “Small Business Application”, highlight the text and select Heading 1 in the style drop-down listbox in the toolbar:

![Form Design Screenshot]
b) ADD A TABLE: From the Toolbox, drag an HTML table or in the Menu Bar select Layout|Insert table… to invoke the following dialog, SELECT a 3X3 table:

![Insert Table dialog](image)

The form now looks as follows:

![Small Business Application Layout](image)
d) ADD TWO BUTTONS from the toolbox and SET the PROPERTIES for ID & TEXT:

**Small Business Application**

- Customer Management
- Retail Management

---

**Step 9: Design the Customer Management Form**

- Begin to design what you main page will look like.
- Since we are trying to recreate the WINDOWS APPLICATION AS A WEB APPLICATION AS CLOSE AS POSSIBLE, currently the Windows version looks as follows:
I will use the following WEB controls:

- 1 HTML TABLE – To align controls for Customer Information
- 1 HTML TABLE – To align controls for GRIDVIEW CONTROL
- 6 ASP TEXTBOX CONTROLS – for imputing Customer ID, Name, Social Security, Birth date, Address, & Phone
- 7 ASP control BUTTONS – for various operations, Search, Add, Edit, Remove, Print, PrintAll & Exit
- 1 LABEL – For displaying error messages or other messages (more on this later)
- 1 GRIDVIEW CONTROL – For displaying records of all customers from the database

Note that we don’t need LABELS FOR IDENTIFYING the TEXTBOXES as we did in the Windows Application, we can type these directly in to the TABLE

Design your WEB FORM as follows:

a) Add a Web Form and name it Customer Management as follows:

b) Design the Form using the controls indicated as follows: