

INFS 3204/7204

Service-Oriented Architecture



A/Prof Heng Tao SHEN
ITEE, UQ
Semester 2, 2011

M4: .NET Advance

1

M4 Topics

- .NET advances
 - ADO.NET
 - LINQ
 - ASP.NET
 - Web Service
 - Web Form
 - MVC

2

- ADO.NET

3

Evolution of Databases

- File-based
- Hierarchical
- Network
- Relational
- Object-oriented
- XML

4

Database Connections

- ODBC (Open Database Connectivity)
 - Interoperability to a wide range of database management systems (DBMS)
 - Widely accepted API
 - Uses SQL as data access language
- DAO (Data Access Objects)
 - Programming interface for JET (Joint Engine Technology) / ISAM (Indexed Sequential Access Method) databases
 - Uses automation: ActiveX, OLE (Object Linking and Embedding) automation
- RDO (Remote Data Objects)
 - Tighter coupling to ODBC
 - Geared more to client/server databases

5

Database Connections

- OLE (Object Linking and Embedding) DB
 - Broad access to data, relational and other
 - Built on COM (Component Object Model)
 - Not restricted to SQL for retrieving data
 - Can use ODBC drivers
 - Low-level (C++) interface
- ADO (ActiveX Data Objects)
 - Simple component-based, object-oriented interface
 - Provides a programming model to OLE DB accessible outside of C++

6

ADO

- ADO was designed as a connected, tightly coupled model
 - Appropriate for client-server architectures
- Primarily relational (not hierarchical like XML)
- Object design is not well factored
 - Too many ways to do the same thing
 - Objects try to do too much
- Not originally designed for a distributed, n-tier environment

7

What Is ADO.NET?

- ADO .NET is a collection of classes, interfaces, structures, and enumerated types that manage data access from relational databases within the .NET Framework
 - Collections are organized into namespaces:
 - System.Data, System.Data.SqlClient, etc
- ADO .NET is an evolution from ADO
 - Does not share the same object model, but shares many of the same paradigms and functionality!

8

ADO.NET Goals

- Well-factored design
- Highly scalable through a robust disconnected model
- Rich XML support (hierarchical as well as relational)
- Data access over HTTP
- Maintain familiar ADO programming model
- Keep ADO available via .NET COM interoperability

9

Data Access Styles

- Connected
 - Application issues query then reads back results continuously and processes them
 - "Firehose" cursor
 - **DataReader** object
- Disconnected
 - Application issues query then retrieves and stores results for processing
 - Minimizes time connected to database
 - **DataSet** object

10

ADO.NET Classes

- IDbConnection Interface
- IDbCommand Interface
- IDataReader Interface
- System.Data.OleDb Namespace
- System.Data Namespace
- **DataSet**
- System.Data.SqlClient Namespace
- IDataAdapter Interface

11

IDbConnection Interface

- Creates a unique session with a data source
- Implemented by SqlConnection and OleDbConnection
- Functionality
 - Open, close connections
 - Begin transactions
 - IDbTransaction provide Commit and Rollback methods
- Used in conjunction with IDbCommand and IDataAdapter objects
- Additional properties, methods and collections depend on the provider

12

IDbCommand Interface

- Represents a statement to be sent to a source
 - Usually, but not necessarily SQL
- Implemented by OleDbCommand and SqlCommand
- Functionality
 - Define statement to execute
 - Execute statement
 - Pass and retrieve parameters
 - Create a prepared version of command
- ExecuteReader returns rows, ExecuteScalar returns single value
- Additional properties, methods and collections depend on the provider

13

IDataReader Interface

- Forward-only, read-only ("fire hose") access to a stream of data
- Implemented by SqlDataReader and OleDbDataReader
- Created via ExecuteReader method of IDbCommand
- Operations on associated IDbConnection object disallowed until reader is closed

14

System.Data.OleDb Namespace

- Managed provider for use with OLEDB providers
 - SQLOLEDB (SQL Server) – use System.Data.SQL
 - MSDAORA (Oracle)
 - JOLT (Jet)
 - OLEDB for ODBC providers
- OleDbConnection, OleDbCommand and OleDbDataReader classes
- Classes for error handling
- Classes for connection pooling

15

DataReader: An Example

```
string sConnString = "Provider=SQLOLEDB.1;" +
    "User ID=XX;Initial Catalog=YY;" +
    "Data Source=MYSERVER";
OleDbConnection conn = new OleDbConnection(sConnString);
conn.Open();
string sQueryString = "SELECT CompanyName FROM Customers";
OleDbCommand myCommand = new OleDbCommand(sQueryString, conn);
OleDbDataReader myReader = myCommand.ExecuteReader();
while (myReader.Read()) {
    Console.WriteLine(myReader.GetString(0));
}
myReader.Close();
conn.Close();
```

16

System.Data Namespace

- Contains the core classes of the ADO.NET architecture
- Disconnected DataSet is central
- Supports all types of applications
 - Internet based
 - ASP.NET
 - XML
 - Windows forms based
- Contains classes used by or derived from managed providers
 - IDbConnection, IDbCommand, IDbDataReader

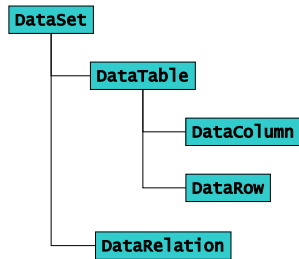
17

DataSet

- A collection of tables
- Has no knowledge of the source of the data
- Keeps track of all relationships among tables
- Rich programming model (has objects for tables, columns, relationships, and so on)
- Remembers original and current state of data
- Can dynamically modify data and metadata
- Native serialization format is XML
- Located in System.Data

18

DataSet



19

IDataAdapter Interface

- Populates or sends updates to a DataSet
- Implemented by OleDbDataAdapter and SqlDataAdapter
- Not connection based
- Represents an asynchronous approach
- A superset of a command object
- Contains four default command objects for Select, Insert, Update, and Delete

21

DataSet: An Example

```
string sConnString = "Persist Security Info=False;" +
    "User ID=XX;Initial Catalog=YY;" +
    "Data Source=MYSERVER";
SqlConnection conn = new SqlConnection(sConnString);
conn.Open();
string sQueryString = "SELECT CompanyName FROM Customers";
SqlDataAdapter myDataAdapter = new SqlDataAdapter();
DataSet myDataSet = new DataSet();
myDataAdapter.SelectCommand = new SqlCommand(sQueryString, conn);
myDataAdapter.Fill(myDataSet);
conn.Close();
```

22

Errors and Exceptions

- Error class
 - Contains information on an error or warning returned by data source
 - Created and managed by Errors class
- Errors class
 - Contains all errors generated by an adapter
 - Created by Exception class
- Exception class
 - Created whenever an unhandled error occurs
 - Always contains at least one Error instance

23

- LINQ

24

Motivation

- Most programs written today manipulate data in one way or another and often this data is stored in a relational database. Yet there is a huge divide between modern programming languages and databases in how they represent and manipulate information
- This mismatch is visible in multiple ways
 - Most notable is that programming languages access information in databases through APIs that require queries to be specified as text strings
 - These queries are significant portions of the program logic. Yet they are opaque to the language, unable to benefit from compile-time verification and design-time features

25

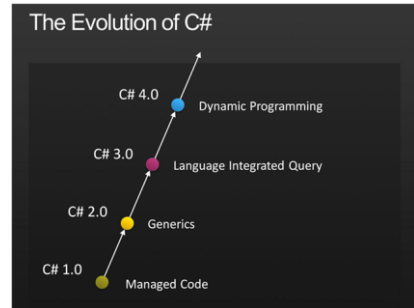
.NET solution: LINQ

- LINQ is a set of extensions to the .NET Framework that encompass Language-INtegrated Query, set, and transform operations. It extends C# and Visual Basic with native language syntax for queries and provides class libraries to take advantage of these capabilities
- LINQ to Object, SQL, XML, Wikipedia

Date	.NET Framework	CLR	C#	IDE	Introduced Features
February 13th, 2002	1.0	1.0	1.0	Visual Studio .NET 2002	Managed Code
April 3rd, 2003	1.1	1.1		Visual Studio .NET 2003	.NET Compact Framework, etc.
October 27th, 2005	2.0	2.0	2.0	Visual Studio 2005	Generics
November 21st, 2006	3.0				WCF, WPF, WF, WC
November 19th, 2007	3.5		3.0	Visual Studio 2008	LINQ
April 12th, 2010	4.0	4.0	4.0	Visual Studio 2010	Dynamic programming, etc.

26

LINQ via C#



27

LINQ to SQL

- LINQ to SQL provides a run-time infrastructure for managing relational data as objects without losing the ability to query
- It does this by translating LINQ into SQL for execution by the database, and then translating the tabular results back into objects you define
- Your application is then free to manipulate the objects while LINQ to SQL stays in the background tracking your changes automatically

28

Features of LINQ to SQL

- Non-intrusive to your application
 - It is possible to migrate current ADO.NET solutions to LINQ to SQL in a piecemeal fashion (sharing the same connections and transactions) since LINQ to SQL is simply another component in the ADO.NET family
 - LINQ to SQL also has extensive support for stored procedures, allowing reuse of the existing enterprise assets
- Easy to get started
 - Objects linked to relational data can be defined just like normal objects, only decorated with attributes to identify how properties correspond to columns
 - A design-time tool is provided to automate translating pre-existing relational database schemas into object definitions
- Language-agnostic
 - Any language built to provide LINQ can use it to enable access to information stored in relational databases

29

A Quick Tour

- Creating Entity Classes
- The **DataContext**
- Defining Relationships
- Querying Across Relationships
- Modifying and Saving Entities

Creating Entity Classes

- A class **Customer** and associate it with the Customers table

```
[Table(Name="customers")]
public class customer
{
    [Column(IsPrimaryKey=true)]
    public string CustomerID;

    [Column]
    public string city;
}
```

- The Table attribute has a **Name** property that specifies the exact name of the database table. If no Name property is supplied, LINQ to SQL will assume the database table has the same name as the class
- Instances of these types of classes are known as **entities**. The classes themselves are known as **entity classes**

The DataContext

- The **DataContext** is the main conduit by which you retrieve objects from the database and resubmit changes
- The DataContext is initialized with a connection or connection string
- The purpose of the DataContext is to translate your requests for objects into SQL queries and then assemble objects out of the results
- The DataContext enables LINQ by implementing the same operator pattern as the standard query operators such as **Where** and **Select**

The DataContext – An Example

```
// DataContext takes a connection string
DataContext db = new
DataContext("c:\\northwind\\northwnd.mdf");

// Get a typed table to run queries
Table<Customer> Customers = db.GetTable<Customer>();

// Query for customers from London
var q =
    from c in Customers
    where c.City == "London"
    select c;
foreach (var cust in q)
    Console.WriteLine("id = {0}, City = {1}",
        cust.CustomerID, cust.City);
```

Defining Relationships

- Relationships in relational databases
 - **Foreign keys** referring to primary keys in other tables
 - To navigate between them, use a relational **join**
- Objects, on the other hand
 - Refer to each other using property references, i.e. **"dot"**
- Obviously, dotting is simpler than joining, since you need not recall the explicit join condition each time you navigate
- LINQ to SQL defines an **Association** attribute you can apply to a member
 - An association relationship is one like a foreign-key to primary-key relationship that is made by matching column values between tables

Defining Relationships – An Example on Customer Class

```
[Table(Name="Customers")]
public class Customer {
    [Column(Id=true)]
    public string CustomerID; ...

    private EntitySet<Order> _Orders;
    [Association(Storage="_Orders",
        OtherKey="CustomerID")]

    public EntitySet<Order> Orders {
        get { return this._Orders; }
        set { this._Orders.Assign(value); }
    }
}
```

The **Orders** property is of type **EntitySet** because the relationship is one-to-many.
The **OtherKey** property in the **Association** attribute to describe how this association is done.

Defining Relationships – An Example on Order Class

```
[Table(Name="Orders")]
public class Order {
    [Column(Id=true)]
    public int OrderID;
    [Column] public string CustomerID;

    private EntityRef<Customer> _Customer;
    [Association(Storage="_Customer",
        ThisKey="CustomerID")]

    public Customer Customer {
        get { return this._Customer.Entity; }
        set { this._Customer.Entity = value; }
    }
}
```

The **EntityRef** type to describe the relationship back, required to support *deferred loading*.
The **Storage** property tells LINQ to SQL which private member is used to hold the value of the property.

Querying Across Relationships

```
var q =
    from c in db.Customers
    from o in c.Orders
    where c.City == "London"
    select new { c, o};
```

The **Orders** property forms the cross product between customers and orders, producing a new sequence of **Customer** and **Order** pairs.

Modifying and Saving Entities

- Few applications are built with only query in mind. Data must be created and modified, too
- LINQ to SQL is designed to offer maximum flexibility in manipulating and persisting changes made to your objects

38

Modifying and Saving Entities – An Example

```
Northwind db = new
Northwind("c:\\northwind\\northwnd.mdf");

// Query for a specific customer
string id = "ABCD";
var cust = db.Customers.Single(c => c.CustomerID == id);

// Change the name of the contact
cust.ContactName = "New Contact";

// Create and add a new Order to Orders collection
Order ord = new Order { OrderDate = DateTime.Now };
cust.Orders.Add(ord);

// Ask the DataContext to save all the changes
db.SubmitChanges();
```

When **SubmitChanges()** is called, LINQ to SQL automatically generates and executes SQL commands in order to transmit the changes back to the database.

Important Notes

- A select query in LINQ will always return a sequence
- A sequence can be of IQueryable, IEnumerable, etc
- A sequence can be converted into a List by using the **ToList<[object]>()** function

40

Other Useful Functions

- InsertAllOnSubmit() – Bulk insert
- DeleteAllOnSubmit() – Bulk delete
- Distinct()
- First()
- OrderBy()
- Count()

41

- ASP.NET

42

What is ASP?

- Server-side programming technology
- Consists of static HTML interspersed with script
- ASP intrinsic objects (Request, Response, Server, Application, Session) provide services
- Commonly uses ADO to interact with databases
- Application and session variables
- Application and session begin/end events
- ASP manages threads, database connections, ...

43

HelloWorld.asp

```
<html>
<head><title>HelloWorld.asp</title></head>
<body>
<form method="post">
<input type="submit" id=button1 name=button1
value="Kiss Me" />
<%
if (Request.Form("button1") <> "") then
  Response.Write "<p>Hello, the current time is " &
Now()
end if
%>
</form>
</body>
</html>
```

44

ASP Challenges

- Coding overhead (too much code)
 - Everything requires writing code!
- Code readability (too complex; code and UI intermingled)
- Maintaining page state requires more code
- Reuse is difficult
- Deployment issues (e.g. DLL locking)
- Session state scalability and availability
- Limited support for caching, tracing, debugging
- Performance and safety limitations of script

45

What is ASP.NET

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services
- Like ASP, ASP.NET is a server-side technology
- Web Applications are built using Web Forms or MVC
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications

46

ASP.NET Architecture

- ASP.NET is built upon
 - .NET Framework
 - Internet Information Services (IIS), formally Internet Information Server
 - Is a set of Internet-based services for servers created by Microsoft for use with Microsoft Windows
 - Is the world's second most popular Web server in terms of overall websites behind the industry leader Apache HTTP Server

47

IIS

- IIS MMC Snap-In (IIS Manager)
 - Provides fine control over the configuration settings for the applications deployed on a Web server
 - Creates Virtual Directories
 - A mapping between URL and file path
 - E.g., the URL:
http://localhost/infs3204
maps to the file path:
C:_infs3204

48

- Web Service

49

Web Service Support

- ASP.NET provides support for XML Web services with the **.asmx** file. A **.asmx** file is a text file that is similar to a **.aspx** file. These files can be part of an ASP.NET application that includes **.aspx** files. These files are then URI-addressable.
- Similar to Web Forms, but
 - have a **.asmx** file extension
 - contains code, w/o UI
- Lives in a virtual directory
- Can have a code-behind
- Automatically generates WSDL
- Can use .NET Framework classes and custom assemblies and classes

50

Developing a Web Service

Codebehind

```
<%@ WebService Language="C#"
Codebehind="MyWebService.cs"
Class="FirstWebService.MathService" %>
```

Inline (in C#)

```
<%@ WebService Language="C#" Class="MathService" %>
using System.Web.Services;
public class MathService : WebServices {
    [WebMethod]
    public int Add(int num1, int num2) {
        return num1 + num2;
    }
}
```

51

Web Service: HelloWorld

```
<%@ WebService Language="C#" Class="HelloWorld" %>
using System;
using System.Web.Services;
public class HelloWorld
{
    [WebMethod]
    public string HelloO
    {
        return "Hi! The time is now " +
            DateTime.Now.ToString();
    }
}
```

52

Web Service: MathService

```
<%@ WebService Language="C#" Class="MathService" %>
using System; using System.Web.Services;
[WebService(Namespace="http://www.microsoft.com/MathService/")]
public class MathService {
    [WebMethod]
    public int Add(int a, int b)
    { return a + b;}
    [WebMethod]
    public int Subtract(int a, int b)
    { return a - b;}
    [WebMethod]
    public int Multiply(int a, int b)
    { return a * b;}
    [WebMethod]
    public int Divide(int a, int b)
    { if (b==0) return -1;
      return a / b;}
}
```

53

Consuming Web Services

- Locate the desired Web Service
 - UDDI
 - DISCO(Web Services Discovery Tool)
- Get detailed description of Web Service
 - WSDL
- Create a proxy that represents the Web Service
 - Proxy has the same methods/arguments/return values as the Web Service
- Application instantiates and uses the proxy as if it were a local object

54

Consuming Web Services

- Web Services are URL addressable
 - HTTP request/response
- Can request WSDL via URL
- Can invoke via:
 - HTTP-GET
 - HTTP-POST
 - HTTP-SOAP

55

Invoking

- HTTP-GET

```
http://localhost/MathService.asmx/Multiply?a=11&b=11
```

- HTTP-POST

```
POST /MathService.asmx/Multiply HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length
a=11&b=11
```

- Result is an XML document

```
<?xml version="1.0" encoding="utf-8" ?>
<int xmlns="http://www.microsoft.com/MathService/">121</int>
```

56

Invoking: HTTP-SOAP

- XML grammar for
 - WebMethod, Method parameter, results
- Supports all standard .NET data types and value classes
 - Additionally: classes, structs, datasets
- Class and struct marshalling
 - Serialization in XML format

57

Creating a Proxy

- Use wsdl.exe to generate a proxy

```
wsdl http://localhost/MathService.asmx?WSDL
```

- Creates MathService.cs
- Contains MathService class, derived from SoapHttpClientProtocol in the System.Web.Services.Protocols namespace
 - Or HttpClientProtocol or HttpPostClientProtocol
 - You can instantiate these classes dynamically
- Proxy embeds URL to the Web Service in the constructor

58

In Visual Studio

- Use Add Web Reference to search UDDI or to discover Web Services given a URL
- This builds a proxy, and you can start using the Web Service immediately
 - Visual Studio essentially calls disco.exe and wsdl.exe for you

59

TestServices.cs

```
using System;
using myNamespace;
public class TestMathService
{
    static void Main()
    {
        MathService ms = new MathService();
        string s; int x, y;
        Console.WriteLine("Enter first integer: ");
        s = Console.ReadLine();
        x = int.Parse(s);
        Console.WriteLine("Enter second integer: ");
        s = Console.ReadLine();
        y = int.Parse(s);
        int result = ms.Add(x, y);
        Console.WriteLine("ms.Add({0}, {1}) = {2}", x, y, result);
        Console.ReadLine();
    }
}
```

60

- Web Forms

61

What is Web Form?

- The ASP.NET Web Forms framework is a scalable CLR programming model that can be used on the server to dynamically generate Web pages
- In particular, it provides the ability to:
 - Create and use **reusable UI controls** that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write
 - Structure their page logic in an **orderly fashion**
 - Provide strong **WYSIWYG** design support

62

HelloWorld.aspx

```
<% Page language="c#" %>
<html>
<head></head>
<script runat="server">
public void B_Click(object sender, System.EventArgs e) {
    Label1.Text = "Hello, the current time is " +
        DateTime.Now;
}
</script>
<body>
    <form method="post" runat="server">
        <asp:Button onclick="B_Click" Text="Kiss Me"
            runat="server" /> <p>
        <asp:Label id=Label1 runat="server" />
    </form>
</body>
</html>
```

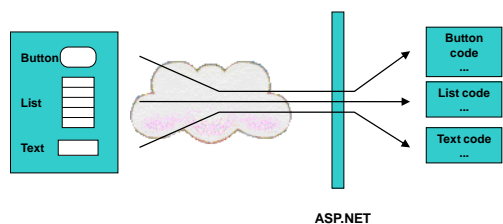
63

Programming Model

- Server-side programming model
- Based on controls and events
 - Just like Visual Basic
 - Not "data in, HTML out"
- Higher level of abstraction than ASP
- Requires less code
- More modular, readable, and maintainable

64

Control and Event Paradigm



65

Server Control Syntax

- Controls are declared as HTML tags with **runat="server"** attribute

```
<input type=text id=text1 runat="server" />
<asp:calendar id=myCal runat="server" />
```

- Tag identifies which type of control to create
 - Control is implemented as an ASP.NET class
- The **id** attribute provides programmatic identifier
 - It names the instance available during postback
 - Just like Dynamic HTML

66

Server Control Properties

- Tag attributes map to control properties

```
<asp:button id="c1" Text="HT" runat="server">
<asp:ListBox id="c2" Rows="2" runat="server">
```

- Tags and attributes are case-insensitive
- Control properties can be set programmatically

```
c1.Text = "HT";
c2.Rows = 2;
```

67

Server Code Blocks

- Server code lives in a script block marked `runat="server"`

```
<script language="C#" runat=server>
<script language="VB" runat=server>
```

- Script blocks can contain
 - Variables, methods, event handlers, properties
 - They become members of a custom Page object

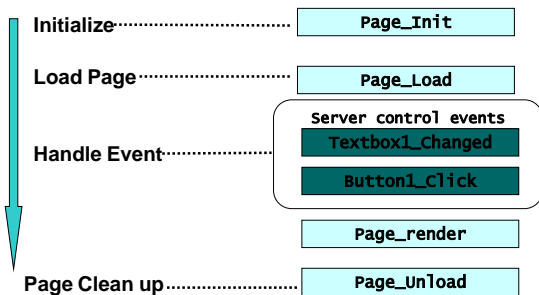
68

Page Events

- Pages are structured using events
 - Enables clean code organization
 - Avoids the "Monster IF" statement
 - Less complex than ASP pages
- Code can respond to page events
 - E.g. Page_Load, Page_Unload
- Code can respond to control events
 - Button_Click
 - Textbox_Changed

69

Page life cycle



70

Page Loading

- Page_Load fires at beginning of request after controls are initialized
 - Input control values already populated

```
protected void Page_Load(Object s, EventArgs e)
{
    message.Text = textbox.Text;
}
```

71

Page Loading

- Page_Load fires on every request
 - Use Page.IsPostBack to execute conditional logic
 - If a Page/Control is maintaining state then need only initialize it when IsPostBack is false

```
protected void Page_Load(Object s, EventArgs e) {
    if (! Page.IsPostBack) {
        // Executes only on initial page load
        Message.Text = "initial value";
    }
    // Rest of procedure executes on every request
}
```

72

Server Control Events

- Change Events
 - By default, these execute only on next action event, e.g. OnTextChanged, OnCheckedChanged
 - Change events fire in random order
- Action Events
 - Cause an immediate postback to server
 - E.g. OnClick
- Works with any browser
 - No client script required, no applets, no ActiveX® Controls!

73

Page Unloading

- Page_Unload fires after the page is rendered
- Useful for logging and clean up

```
protected void Page_Unload(Object s, EventArgs e) { MyApp.LogPageComplete(); }
```

74

Sever control

- ASP.NET ships with ~50 built-in controls
- Organized into logical families
 - HTML controls
 - Controls / properties map 1:1 with HTML
 - Web controls
 - Richer functionality
 - More consistent object model

75

HTML Controls

- Work well with existing HTML designers
- Properties map 1:1 with HTML
 - table.bgcolor = "red";
- Can specify client-side event handlers
- Good when quickly converting existing pages
- Derived from System.Web.UI.HtmlControls.HtmlControl
- Supported controls have custom class, others derive from HtmlGenericControl

76

HTML Controls

- <a>
-
- <form>
- <table>
- <tr>
- <td>
- <th>
- <select>
- <textarea>
- <button>
- <input type=text>
- <input type=file>
- <input type=submit>
- <input type=button>
- <input type=reset>
- <input type=hidden>

77

Web Controls

- Web Controls provide extensive properties to control display and format, e.g.
 - Font
 - BackColor, ForeColor
 - BorderColor, BorderStyle, Borderwidth
 - Style, CssClass
 - Height, width
 - Visible, Enabled

78

Web controls

- Web controls appear in HTML markup as namespaced tags
- Web controls have an **asp:** prefix

```
<asp:button onclick="button1_click" runat=server>  
<asp:textbox onchange="text1_changed" runat=server>
```

- Defined in the System.Web.UI.WebControls namespace
- This namespace is automatically mapped to the **asp:** prefix

79

What is Web Application?

- All resources (files, pages, handlers, modules, executable code, etc.) within a virtual directory and its subdirectories
 - Configuration files
 - Shared data (application variables)
 - `global.asax`
- Scopes user sessions
 - A session is a series of web page hits by a single user within a block of time
 - Shared data (session variables)
- A video from Microsoft to demonstrate how to build a web application using web forms
 - <http://www.asp.net/general/videos/build-your-first-asp-net-application-with-asp-net-web-forms>

83

- MVC

84

The Concept

- The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the **model**, the **view**, and the **controller**
- The MVC framework is defined in the `System.Web.Mvc` assembly.

85

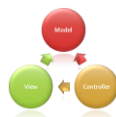
Why Separate Concerns

- Makes your app code more
 - Testable
 - Modifiable / refactorable
 - Reusable
 - Maintainable
 - Scalable

86

Model, View and Controller

- Models (database)
 - Set of classes that represent data and business rules for how data can be changed and manipulated
- Views (UI logic)
 - Components that display the user interface (UI). Typically, this UI is created from the model data
- Controllers (business logic)
 - Components that handle user interaction, work with the model, and ultimately select a view to render UI



87

Common Features with Web Form

- Both hosted in Visual Studio
- Both run on IIS
- Both use `.aspx` pages
- Both can use Master pages, though you can use a different view engine if you'd like in MVC (NHaml, Spark, Brail, NVelocity, etc.)
- Both can use any data access framework (ADO.NET, LINQ, Entity Framework, etc.)

88



Advantages of MVC

- Is easier to manage complexity by dividing an application into model, view, and controller
- Does not use view state or server-based forms. This makes MVC ideal to have full control over the behavior of an application
- Provides better support for test-driven development
- Works well for Web applications that are supported by large teams of developers and designers who need a high degree of control over the application behavior

89



Advantages of Web Form

- Allows separation of UI and business logic
- Is easy to use available components
- Has easy drag and drop experience
- Supports an event model that preserves state over HTTP and provides dozens of events
- Provides scalable session state management,
- Uses Page Controller pattern that adds functionality to individual pages easily
- Works well for small teams of Web developers and designers for RAD
- Is less complex and less code than MVC

90



Ruby On Rails

- Or ROR for short, is the framework which took the world by storm
- It made the MVC idea a lot more famous
- Ruby is the language on which this framework is built. The language is very powerful and expressive but runs pretty slow
- ROR is great for when your working on a non windows platform and want to write as few lines as possible and want to work in Ruby

91



Choose Web Forms if

- You like Web Forms
- Your brain hurts when reading about design patterns
- You don't like following patterns or best practices
- Your are not interested in embracing unit tests or TDD (test-driven development)
- You are being told to use it by your employer/client

92



Choose MVC if

- You want to embrace design patterns and are willing to put hard work into understanding them
- You are disciplined and want to follow design patterns and best practices
- You are interested in embracing unit tests
- You are interested in embracing TDD
- You are being told to use it by your employer/client
- You want to go deeper and deeper

93



Another Way to Look At It

- Building an intranet site with lots of data editing
 - Web Forms may be better suited
- Building an Internet site where HTML, performance, and scalability are paramount
 - ASP.NET MVC may be better suited

94

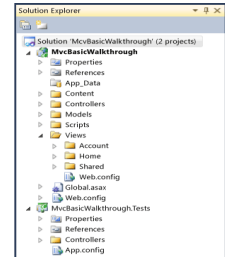
Building MVC Application

- Rough steps:
 1. New ASP.NET MVC Project
 2. Delete Home files
 3. Add model using LINQ 2 SQL
 4. Add repositories
 5. Add HomeController
 6. Create Views
 7. Add support for drop-downs
 8. Add validation support
 9. Add unit tests

95

MVC Application Structure

- The ASP.NET MVC framework includes a Visual Studio project template that helps you create Web applications that are structured to support the MVC pattern



96

Folder: App_Data & Content

- App_Data, which is the physical store for data. This folder has the same role as it does in ASP.NET Web sites that use Web Forms pages
- Content, which is the recommended location to add content files such as cascading style sheet files, images, and so on. In general, the Content folder is for static files

Folder: Controllers

- Controllers, which is the recommended location for controllers, Where you put Controller classes that handle URL requests
- The MVC framework requires the names of all controllers to end with "Controller", such as **HomeController**, **AccountController**, **LoginController**, or **ProductController**

Folder: Models

- Models, which is provided for classes that represent the application model for your MVC Web application, where you put classes that represent and manipulate data
- This folder usually includes code that defines objects and that defines the logic for interaction with the data store
- Typically, the actual model objects will be in separate class libraries. However, when you create a new application, you might put classes here and then move them into separate class libraries at a later point in the development cycle

Folder: Scripts

- Scripts, which is the recommended location for script files that support the application, where you put JavaScript library files and scripts (.js)
- By default, this folder contains ASP.NET **AJAX** foundation files and the **jQuery** library

Folder: Views

- Views, which is the recommended location for views. Views use ViewPage (.aspx), ViewUserControl (.ascx), and ViewMasterPage (.master) files, in addition to any other files that are related to rendering views, where you put UI template files that are responsible for rendering output
- The Views folder contains a folder for each controller; the folder is named with the controller-name prefix

102

Building Web Application Using MVC and LINQ to SQL

- Free ASP.NET MVC "NerdDinner" Tutorial
 - <http://nerddinnerbook.s3.amazonaws.com/Intro.htm>
- An excellent tutorial in helping you complete your pracs
- Codes are available for download

Summary

- This week
 - .NET Advances
 - ADO.NET
 - LINQ
 - ASP.NET
 - Web Service
 - Web Form
 - MVC
- Next week:
 - **Web Service: Basic Technologies**

103

References

- Web Services Tutorial
 - <http://www.w3schools.com/webservices/default.asp>
- Web Services and the .Net framework
 - <http://msdn.microsoft.com/en-us/library/dd560541.aspx>
- 10 Tips for Writing High-Performance Web Applications
 - [http://msdn.microsoft.com/zh-cn/magazine/cc163854\(en-us\).aspx](http://msdn.microsoft.com/zh-cn/magazine/cc163854(en-us).aspx)
- LINQ 2 SQL
 - <http://msdn.microsoft.com/en-us/library/bb425822.aspx>
- .NET MVC
 - <http://www.asp.net/mvc/tutorials/getting-started-with-mvc3-part1-cs>
 - <http://msdn.microsoft.com/en-us/library/dd381412%28VS.98%29.aspx>

104