

INFS 3204/7204

Service-Oriented Architecture



Dr Heng Tao SHEN
ITEE, UQ
Semester 2, 2009.

M4: .NET Advances

M4 Topics

- .NET advances
 - ADO.NET
 - ASP.NET
 - Web Forms
 - Web Services

- ADO.NET

Evolution of Databases

- File-based
- Hierarchical
- Network
- Relational (RDBMS)
- Object-oriented
- XML

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

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++

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

INFS3204/7204 – M4

7

What Is ADO.NET?

- ADO .NET is a collection of classes, interfaces, structures, and enumerated types that manage data access from relational data stores within the .NET Framework
 - These 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!

INFS3204/7204 – M4

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

INFS3204/7204 – M4

9

Data Access Styles

- Connected: Forward-only, read-only
 - Application issues query then reads back results 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

INFS3204/7204 – M4

10

ADO.NET Classes

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

INFS3204/7204 – M4

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

INFS3204/7204 – M4

12

IDbCommand Interface

- Represents a statement to be sent to a data 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 (compiled) version of command
- `ExecuteReader` returns rows, `ExecuteNonQuery` doesn't, `ExecuteScalar` returns single value
- Additional properties, methods and collections depend on the provider

INFS3204/7204 - M4

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

INFS3204/7204 - M4

14

System.Data.OleDb Namespace

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

INFS3204/7204 - M4

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();
```

INFS3204/7204 - M4

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`

INFS3204/7204 - M4

17

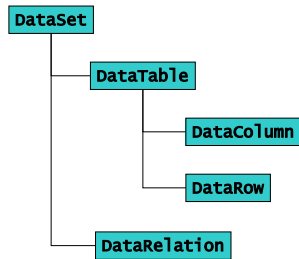
DataSet

- 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`

INFS3204/7204 - M4

18

DataSet



INFS3204/7204 - M4

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

INFS3204/7204 - M4

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();
```

INFS3204/7204 - M4

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

INFS3204/7204 - M4

23

- ASP.NET

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, ...

INFS3204/7204 - M4

24

INFS3204/7204 - M4

25

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>
```

INFS3204/7204 - M4

26

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

INFS3204/7204 - M4

27

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
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications

INFS3204/7204 - M4

28

ASP.NET Key Features

- Web Forms
- Web Services
- Built on .NET Framework
- Simple programming model
- Maintains page state
- Multibrowser support
- XCOPY deployment
- XML configuration
- Complete object model
- Session management
- Caching
- Debugging
- Extensibility
- Separation of code and UI
- Security
- ASPX, ASP side by side
- Simplified form validation
- Cookieless sessions

INFS3204/7204 - M4

29

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

INFS3204/7204 - M4

30

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

INFS3204/7204 - M4

31

- Web Forms

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

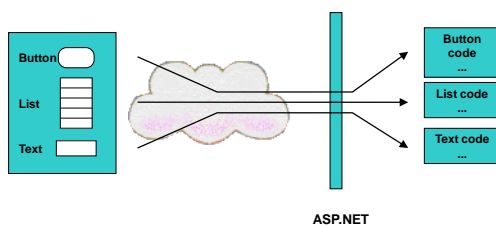
HelloWorld.aspx

```
<%@ 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>
```

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

Control and Event Paradigm



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

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;
```

INFS3204/7204 - M4

38

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

INFS3204/7204 - M4

39

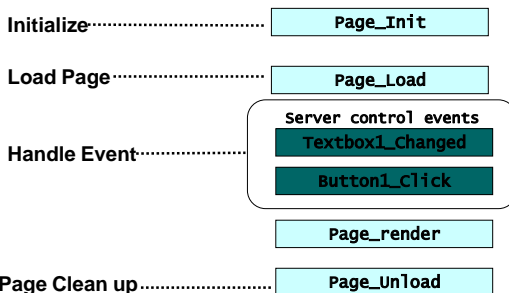
Page Events

- 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

INFS3204/7204 - M4

40

Page life cycle



INFS3204/7204 - M4

41

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;
}
```

INFS3204/7204 - M4

42

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
}
```

INFS3204/7204 - M4

43

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!

INFS3204/7204 - M4

44

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();  
}
```

INFS3204/7204 - M4

45

Server 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

INFS3204/7204 - M4

46

HTML Controls

- Work well with existing HTML designers
- Properties map 1:1 with HTML
 - `table.backgroundColor = "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`

INFS3204/7204 - M4

47

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>`

INFS3204/7204 - M4

48

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`

INFS3204/7204 - M4

49

Web controls

- Web controls appear in HTML markup as namespaces 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

INFS3204/7204 - M4

50

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)

INFS3204/7204 - M4

54

- Web Service

INFS3204/7204 - M4

55

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 an .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

INFS3204/7204 - M4

56

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;  
    }  
}
```

INFS3204/7204 - M4

57

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();  
    }  
}
```

INFS3204/7204 - M4

58

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;
    }
}
```

INFS3204/7204 - M4

59

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

INFS3204/7204 - M4

60

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

INFS3204/7204 - M4

61

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>
```

INFS3204/7204 - M4

62

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

INFS3204/7204 - M4

63

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 `HttpGetClientProtocol` or `HttpPostClientProtocol`
 - You can instantiate these classes dynamically
- Proxy embeds URL to the Web Service in the constructor

INFS3204/7204 - M4

64

In Visual Studio.NET

- 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.NET essentially calls `disco.exe` and `wsdl.exe` for you

INFS3204/7204 - M4

65

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();
    }
}
```

INFS3204/7204 - M4

66

Summary

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

INFS3204/7204 - M4

67

References

- **ADO.NET**
 - <http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/dask/adosp1v.htm>
- **ASP.NET**
 - <http://www.asp.net/>
 - <http://msdn.microsoft.com/net/aspnet/default.asp>
- **Building Web Services with SOAP and ASP.NET**
 - <http://msdn.microsoft.com/msdnmag/issues/01/02/WebComp/webcomp.asp>

INFS3204/7204 - M4

68