

INFS 3204/7204

Service-Oriented Architecture



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

M3: .NET Basics

INFS3204/7204 - M3

1

M3 Topics

- Introductions to .NET
 - Fundamentals
 - Common Language Runtime (CLR)
 - .NET Framework
 - Components
 - ASP.NET and ADO.NET
 - Web Services
 - XML processing
 - Web forms
- C#

INFS3204/7204 - M3

2

About .NET

- Microsoft's strategy for developing large distributed software systems
 - "An open language platform for Web development"
 - "A component model for the Internet"
- Comparing to
 - COM (Component Object Model)
 - A component model for the desktop
 - CORBA (Common Object Request Broker Architecture)
 - An OO programming model for the Internet
 - Java
 - An OO programming model for the Internet, but for a single programming language

INFS3204/7204 - M3

3

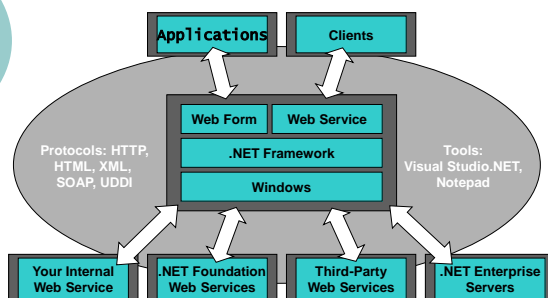
What is .NET

- A platform that supports the vision of how information technology will evolve
- Fundamentals:
 - Common Language Runtime (CLR)
 - .NET Framework

INFS3204/7204 - M3

4

.NET platform



INFS3204/7204 - M3

5

Web Services

- A programmable application component accessible via standard Web protocols
- The center of the .NET architecture
- Expose functionality over the Web
- Built on existing and emerging standards
 - HTTP, XML, SOAP, UDDI, WSDL, ...

INFS3204/7204 - M3

6

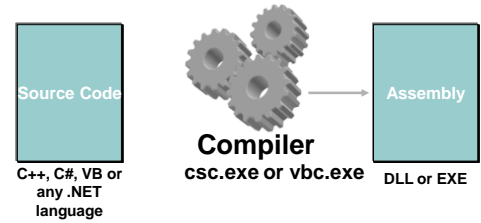
CLR

- The foundation of .NET Framework, providing core services for preparing and managing code execution
 - Verification, compilation, memory & thread management, code safety
- Support cross-language interoperability in tightly-integrated fashion
 - C#, VB, JScript, C++ (managed extension), and Others (COBOL, Perl, Eiffel, Python)
 - Define a class in one language, then use another language, and call a method of it

INFS3204/7204 - M3

7

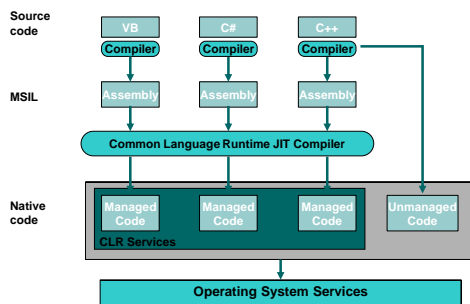
CLR Compilation



INFS3204/7204 - M3

8

CLR Execution Model



INFS3204/7204 - M3

9

MISL

- MSIL: Microsoft Intermediate Language
- Code execution under CLR
 - Write programs in languages supported by CLR
 - Translate source code into MSIL code
 - Covert MSIL code into native code
 - Execute code, with supporting infrastructures
 - Automatic memory management, security, interoperability, versioning support, cross-language debug

INFS3204/7204 - M3

10

Run-time Hosts

- CLR supports both Web server applications and traditional Windows user interface applications
- Each application requires a run-time host to start it
- The run-time host loads CLR into a process, creates the **application domains** within the process, and loads user code into the application domains
- Three run-time hosts
 - ASP.NET
 - Microsoft IE
 - Shell executables

INFS3204/7204 - M3

11

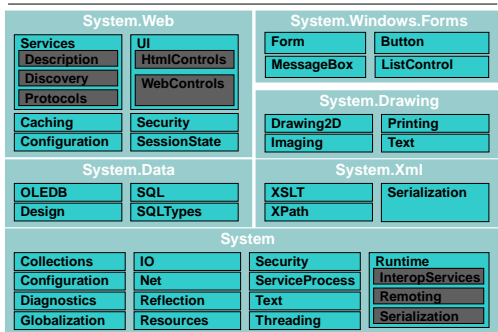
.NET Framework

- An OO programming environment, with a class library of comprehensive, reusable classes for software development, i.e.,
 - Common libraries (such as string management, data collection, database connectivity, IO)
 - GUI (Windows Forms)
 - ASP.NET and Web Forms
 - ADO.NET for access to various data sources
 - XML and Web Services
- Separate software components in different languages to be combined to form one functioning system

INFS3204/7204 - M3

12

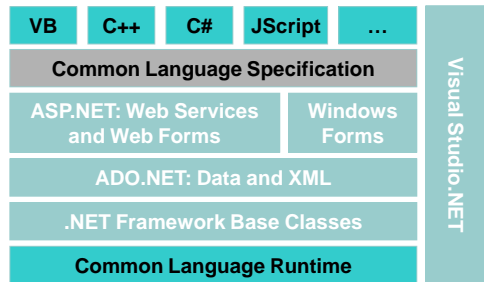
.NET Framework Classes



INFS3204/7204 - M3

13

The .NET Framework and Visual Studio.NET



INFS3204/7204 - M3

14

Supported Languages

- Visual Basic (VB.NET)
 - More OO; more language features; extensible types provided by a class library; better performance
- C++ with Managed Extension
 - Automatic memory management; self-describing objects (no need to use Interface Description Language anymore)
- C#
 - A new languages as part of .NET, taking full advantage of CLR, with benefits from both VB (productivity) and C++ (efficiency)
- JScript
 - Typed and typeless variables, classes, compiled code, better performance (vs. JavaScript & Java?)

INFS3204/7204 - M3

15

Cross-Language Interoperability

- Previous attempts
 - **Representation standards** to define the format of data exchanges between platforms (e.g., XDR, NDR)
 - **Architecture standards** for calling methods crossing boundaries between computer, processes and languages (e.g., RPC, COM, CORBA)
 - **Language standards** for distribution of course code across compilers and computers (e.g., ANSI C)
 - **Execution environment standards** to run same code in different machines (e.g., Java)
- None of them allows that classes and objects defined in one language can be seamlessly used in another language

INFS3204/7204 - M3

16

.NET's Solution

- A common **type system**
 - Defines the types found in the supported languages
- A **metadata system**
 - Stores metadata about the types at compile time and query them at runtime
 - EG, name, visibility, base classes, attributes and methods, type descriptions, security permissions...
- A **common language specification (CLS)**
 - Defines a set of rules that limit the type system to certain groups of facilities provided by CLR
 - EG, CLR supports both signed and unsigned integers, and CLS-compliant only supports unsigned integers
- A **debugger**
 - Allows the programmers to step through programs in different languages

INFS3204/7204 - M3

17

.NET vs. J2EE

- J2EE: one language, any platform
- .Net: any language, one platform
 - Increased performance
 - The performance of an application in .NET is generally better than J2EE
 - Increased productivity
 - According to Software Productivity Research (SPR), .Net increases productivity between 35% and 55%
 - Less source code
 - J2EE requires more source code than .NET to obtain the same functionality
 - More scalability
 - Five to ten times as much to handle the same workload using J2EE as would be required with the .NET platform

INFS3204/7204 - M3

18

What is ASP.NET?

- A Web development platform
 - Is compiled CLR code running at the server side
 - Supports 3 .NET-compatible languages
 - C#, VB and JScript
- ASP.NET is syntax-compatible with ASP
 - More than a new version of ASP
 - Not completely backward compatible with ASP
 - Using .aspx extension to differentiate
 - Migration from ASP to ASP.NET is trivial

INFS3204/7204 - M3

19

ASP Page: An Example

```
<%@ Language=VBScript %>
<html>
<head> <title> An example ASP Page </title> </head>
<body>
<form action="Hello.asp" method="post">
  <p> Your name: <input type="text" name="yourName"></p>
  <p> <input type="submit" value="Echo"> </p>
</form>
<%
  If Len(Request.Form("yourName")) > 0 Then
    Response.write "Hello, " & Request.Form("yourName")
  End If
%>
</body>
</html>
```

INFS3204/7204 - M3

20

New Features in ASP.NET

- Compiled code
 - ASP.NET can be written in any .NET languages, and compiled
 - ASP is limited to interpreted languages, e.g., VBScript
- Server controls (script with **runat="server"**)
 - These controls will be used to generate standard HTML, e.g., function implemented in other languages (such as using VB.NET) can be translated into JavaScript (for client-side validation)
- Code and Content Separation
 - All server code can be placed in a **.aspx.cs** file
 - The **.aspx** file is about HTML content only
- Infrastructure services
 - State/session management, security, caching...
- Easy extension

INFS3204/7204 - M3

21

What is ADO.NET

- Active Data Objects for the .NET framework
- Consistent and scalable solution for access to various data sources
 - Retrieve, manipulate, update relational, XML and application data

INFS3204/7204 - M3

22

Web Services: Provider

- Implement a class in a supported language
- Mark the method that will be accessible as part of the service:
 - **[WebMethod]** in C#,
 - **<WebMethod()>** in VB, or
 - **WebMethodAttribute** in JScript
- Save the code as a file with **.asmx** extension
- Make the file URI-addressable in Microsoft's IIS server

INFS3204/7204 - M3

23

Web Services: Client

- Get the WSDL description of the desired service
- Use **wsdl** tool in the .NET Framework to generate a proxy for the Web Service
 - A proxy is a local object that serves as a front end for a remote object
 - A proxy program can be in one of the supported language (eg, C#)
- Compile the proxy program into a library
- The client program can use the library to access the Web Service

INFS3204/7204 - M3

24



Programming XML

- .NET supports convenient XML programming
 - XML DOM, which works with an XML tree
 - XML Reader and Writer, which sequential access on an XML stream
 - XML validation on reading
 - XML Data Document, which works with relational data
 - XML Transformation, which allows an XML document to be transformed into another format using XSLT

INFS3204/7204 - M3

25



Web Forms (1)

- Built with ASP.NET
 - Logical evolution of ASP
 - Similar model: edit the page and go
- Requires less code
- New programming model
 - Event-driven/server-side controls
 - Rich controls (e.g. data grid, validation)
 - Data binding
 - Controls generate browser-specific code
 - Simplified handling of page state

INFS3204/7204 - M3

26



Web Forms (2)

- Allows separation of UI and business logic
- Uses .NET languages
- Easy to use components
- Simple configuration (XML-based)

INFS3204/7204 - M3

27



Web Forms (3)

- Caching (pages, fragments, custom)
- Scalable session state management
- Tracing support
- ASP.NET is extensible
- Automatic process rollover
- Forms-based authentication

INFS3204/7204 - M3

28

- 
-
- C#

INFS3204/7204 - M3

29



C#

- New language created for .NET
- Safe, productive evolution of C++
- Uses .NET Framework classes
- Key features:
 - Component-oriented
 - Everything is an object
 - Robust and durable code
 - Preserving your investment

INFS3204/7204 - M3

30

Hello World

```
using System;

class Hello {
    static void Main( ) {
        Console.WriteLine("Hello world");
        Console.ReadLine(); }
}
```

INFS3204/7204 - M3

31

Component-Oriented

- C# is the first "Component-Oriented" language in the C/C++ family
- What is a component?
 - An independent module of reuse and deployment
 - Includes multiple classes
 - Coarser-grained than objects
 - Often language-independent

INFS3204/7204 - M3

32

Everything is an Object

- Traditional views
 - C++, Java™: Primitive types are "magic" and do not interoperate with objects
 - Smalltalk, Lisp: Primitive types are objects, but at some performance cost
- C# unifies with no performance cost
- Improved extensibility and reusability
 - New primitive types: Decimal, SQL...

INFS3204/7204 - M3

33

Robust and Durable Software

- Garbage collection
 - No memory leaks and stray pointers
- Type-safety
 - No uninitialized variables, no unsafe casts
- Avoid common errors
 - E.g. if (x = y) ...
- One-stop programming
 - Fewer moving parts

INFS3204/7204 - M3

34

Preserving Your Investment

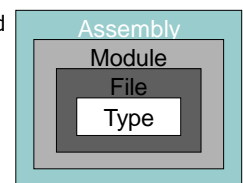
- C++ Heritage
 - Namespaces, pointers (in unsafe code), unsigned types, etc.
- Interoperability
 - What software is increasingly about
 - C# talks to XML, SOAP, COM, DLLs, and any .NET Framework language
- Increased productivity
 - Short learning curve
 - Millions of lines of C# code in .NET

INFS3204/7204 - M3

35

Program Structure

- Physical organization
 - Types are defined in files
 - Files are compiled into modules
 - Modules are grouped into assemblies



INFS3204/7204 - M3

36

Java & C# (1)

- Managed execution environment
 - Type safety enforced at run-time.
 - Garbage collection
 - Pointers not needed
 - C# permits pointers within code marked `unsafe`
 - Compile to machine (java)/language(C#) independent code
- Powerful reflection capabilities
 - Examine within a program, and manipulate internal properties
 - Dynamically discover elements in existing assemblies
- No header files, all code scoped to packages or assemblies
 - Can declare one class before another with circular dependencies

INFS3204/7204 - M3

37

Java & C# (2)

- Classes all descend from `Object` class
- Interfaces support multiple-inheritance
- Classes only support single inheritance
- Inner classes are supported
- Variables/constants/functions must belong to a class
- Bounds checking of arrays and strings
- All values are initialized before use
- `try/catch/finally` exception handling is used
- Objects can be locked to perform thread synchronization

INFS3204/7204 - M3

38

Types

- Reference Types (Same as Java)
 - Passed by reference
 - Stored in a garbage collected heap
- Value Types (Java: primitive types)
 - Passed by value
 - Stored on the run-time stack
 - Basic types:
 - `sbyte`, `byte`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `char`, `float`, `double`, `decimal`, `bool`, `enum`
- C#: Define new `struct` value types:
 - Best for small objects
 - Very efficient in arrays

```
struct Point {
    public int X, Y;

    Point(int x, int y)
    {
        this.X = x;
        this.Y = y;
    }
}
```

INFS3204/7204 - M3

39

Automatic Boxing/Unboxing

- **Boxing:** Value Type → Reference Type
- **Unboxing:** Reference Type → Value Type
- Java 1.4: Explicitly box value types into reference types (`boolean` → `Boolean`, `int` → `Integer`)
 - Needed for use in object collection classes
- C#: boxing/unboxing done automatically

INFS3204/7204 - M3

40

Constant & Readonly Fields

```
public class Foo { // JAVA
    public final static int MAX_TIMEOUT = 300;
    ...
}
```

```
public class Foo { // C#
    public const int MAX_TIMEOUT = 300;
    public readonly X = 1;
    public readonly Y;

    Foo(int init_y) { this.Y = init_y; }
    ...
}
Foo f = new Foo();
f.Y = 23; // compiler error
```

- **readonly**
 - Can only be initialised in declaration or constructor.

INFS3204/7204 - M3

41

Properties

```
class Zap // JAVA
{
    private int _volts;
    public int getVolts()
    { return _volts; }
    public void setVolts(int val)
    { _volts = val; }
}

class Zap // C#
{
    private int _volts;
    public int Volts
    { get { return _volts; }
      set { _volts = value; }
    }
}
```

```
Zap myZap = new Zap();

// Java:
myZap.setVolts(4);
myZap.setVolts(myZap.getVolts() + 1);

// C#:
myZap.Volts = 4;
myZap.Volts++; // get & set
```

INFS3204/7204 - M3

42

Main Method

- Execution begins at the static Main() method
- Can have only one method with one of the following signatures in an assembly
 - static void Main()
 - static int Main()
 - static void Main(string[] args)
 - static int Main(string[] args)

INFS3204/7204 - M3

43

Syntax

- Identifiers
 - Names for types, methods, fields, etc.
 - Must be whole word – no white space
 - Unicode characters
 - Begins with letter or underscore
 - Case sensitive
 - Must not clash with keyword
 - Unless prefixed with @

INFS3204/7204 - M3

44

Statements

- High C++ fidelity
- if, while, do require bool condition
- goto can't jump into blocks
- switch statement
- foreach statement
- checked and unchecked statements

INFS3204/7204 - M3

45

Statements

- Statement lists
- Block statements
- Labeled statements
- Declarations
 - Constants
 - Variables
- Expression statements
 - checked, unchecked
 - lock
 - using
- Conditionals
 - if
 - switch
- Loop Statements
 - while
 - do
 - for
 - foreach
- Jump Statements
 - break
 - continue
 - goto
 - return
 - throw
- Exception handling
 - try
 - throw

INFS3204/7204 - M3

46

foreach Statement

- Iteration of user-defined collections
- Created by implementing IEnumerable in class **customers**

```
foreach (Customer c in
customers.OrderBy("name")) {
    if (c.Orders.Count != 0) {
        ...
    }
}
```

INFS3204/7204 - M3

47

Synchronization

- Multi-threaded applications have to protect against concurrent access to data
 - Must prevent data corruption
- The lock statement uses an instance to provide mutual exclusion
 - Only one lock statement can have access to the same instance
 - Actually uses the .NET Framework System.Threading.Monitor class to provide mutual exclusion

INFS3204/7204 - M3

48

Synchronization: An Example

```
public class CheckingAccount {
    decimal balance;
    public void Deposit(decimal amount) {
        lock (this) {
            balance += amount;
        }
    }
    public void Withdraw(decimal amount) {
        lock (this) {
            balance -= amount;
        }
    }
}
```

INFS3204/7204 - M3

49

using Statement

- C# uses automatic memory management (garbage collection)
 - Eliminates most memory management problems
- However, it results in non-deterministic finalization
 - No guarantee as to when and if object destructors are called

INFS3204/7204 - M3

50

using Statement

- Objects that need to be cleaned up after use should implement the `System.IDisposable` interface
 - One method: `Dispose()`
- The `using` statement allows you to create an instance, use it, and then ensure that `Dispose` is called when done
 - `Dispose` is guaranteed to be called, as if it were in a `finally` block

INFS3204/7204 - M3

51

using Statement: An Example

```
public class MyResource : IDisposable {
    public void MyResource() {
        // Acquire valuable resource
    }
    public void Dispose() {
        // Release valuable resource
    }
    public void DoSomething() {
        ...
    }
}
```

```
using (MyResource r = new MyResource()) {
    r.DoSomething();
} // r.Dispose() is finally called
```

INFS3204/7204 - M3

52

Namespaces

- Namespaces provide a way to uniquely identify a type
- Provides logical organization of types
- Namespaces can span assemblies
- Can nest namespaces
- There is no relationship between namespaces and file structure
- The fully qualified name of a type includes all namespaces
- XML namespace?

INFS3204/7204 - M3

53

Namespaces: An Example

```
namespace N1 { // N1
    class C1 { // N1.C1
        class C2 { // N1.C1.C2
        }
    }
    namespace N2 { // N1.N2
        class C2 { // N1.N2.C2
        }
    }
}
```

- Best practice: Put all of your types in a unique namespace
- Have a namespace for your company, project, etc
- Look at how the .NET Framework classes are organized

INFS3204/7204 - M3

54

References

- In Visual Studio you specify references for a project
- Each reference identifies a specific assembly
- Passed as reference (/r or /reference) to the C# compiler:
 - `csc HelloWorld.cs /reference:System.WinForms.dll`

INFS3204/7204 - M3

55

Namespaces vs. References

- Namespaces provide language-level naming shortcuts
 - Don't have to type a long fully qualified name over and over
- References specify which assembly to use

INFS3204/7204 - M3

56

Using Visual Studio.NET

- Types of projects
 - Console Application
 - Windows Application
 - **Web Application**
 - **Web Service**
 - Windows Service
 - Class Library
 - ...

INFS3204/7204 - M3

57

Using Visual Studio.NET

- Building
- Debugging
 - Break points
- References
- Saving

INFS3204/7204 - M3

58

Summary

- This week
 - Introductions to .NET
 - Fundamentals
 - ASP.NET and ADO.NET
 - Web Services
 - XML processing
 - Web forms
 - C#
- Next week:
 - **.NET Advances**

INFS3204/7204 - M3

59

References

- **Microsoft .NET Home**
 - <http://www.microsoft.com/net/>
- **Microsoft C# Specification**
 - <http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx>
- **Microsoft C# Library**
 - [http://msdn.microsoft.com/en-us/library/aa287558\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/aa287558(VS.71).aspx)

INFS3204/7204 - M3

60