Application Servers

Session 5 – Main Theme
JEE and .Net OMA Implementations
(Part 2)

Dr. Jean-Claude Franchitti

Icons / Metaphors

- Information
- Common Realization
- Knowledge/Competency Pattern
- Governance
- Alignment
- Solution Approach
Agenda

1. Component-Based Technology
2. .Net Overview
3. .Net Services
4. .Net Interoperability Support
5. J2EE Services
6. Conclusion

Components

- A component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces
- Graphically, a component is rendered as a rectangle with tabs
Components vs. Classes

- **Similarities:**
  - Both have names
  - Both may realize a set of interfaces
  - Both may participate in dependency, generalization, and association relationships
  - Both may be nested
  - Both may have instances
  - Both may be participants in interactions

Components vs. Classes (continued)

- **Differences:**
  - Classes represent *logical* abstractions;
  - Components represent *physical* things that live directly on a “node”
    - A node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and processing capability
Components vs. Classes (continued)

- Components represent the physical packaging of otherwise logical components and are at a different level of abstraction.
- The relationship between a component and the classes it implements can be shown explicitly by using a dependency relationship.
- Classes may have attributes and operations directly.
  - In general, components only have operations that are reachable only through their interfaces.

Components and Interfaces

- An interface is a collection of operations that are used to specify a service of a class or a component.
- All the most common component-based operating system facilities (e.g., COM+, CORBA, and Enterprise Java Beans) use interfaces as the glue that binds components together.
Components and Interfaces (continued)

- An interface that a component realizes is called an *export interface*
  - An interface that the component provides as a service to other components
- An interface that a component uses is called an *import interface*
  - An interface that the component conforms to and so builds on

- The fact that an interface lies between two components breaks the direct dependency between the components
- A component that uses a given interface will function properly no matter what component realizes that interface
Binary Replaceability

- The basic intent of every component-based operating system facility is to permit the assembly of systems from binary replaceable parts.
  - Can create a system out of components and then evolve that system by adding new components and replacing old ones.

Characteristics of A Component

- It is physical.
- It is replaceable.
- It is part of a system.
- It conforms to and provides the realization of a set of interfaces.
Three Kinds of Components

- Deployment components
  - Components that are necessary and sufficient to form an executable system.
  - e.g., dynamic libraries (DLLs) and executables (EXEs).

- Work product components
  - Essentially the residue of the development process, consisting of things such as source code files and data files from which deployment components are created

- Execution components
  - Created as a consequence of an executing system
Standard Elements

- UML defines five standard stereotypes that apply to components:
  - Executable
    - May be executed on a node
  - Library
    - A static or dynamic object library
  - Table
    - Represents a database table
  - File
    - Represents a document containing source code or data
  - Document
    - Represents a document

Component Diagrams

- A component diagram shows a set of components and their relationships
- Shows the organization and dependencies among a set of components
Sample Component Diagram

Sample Component Diagram (continued)
Sample Component Diagram
(continued)
OMG's MDA (Model-Driven Architecture) specification describes:

- a PIM - platform-independent models (i.e. business design)
- PSMs - the mapping of a PIM to one or more platform-specific model

MDA => Model Once, Generate Everywhere

Review MDA presentations:

- [http://www.io-software.com](http://www.io-software.com)
The first level of automation ~ 30 Years
(Platform independent to platform specific model projection)

Programming IDEs (e.g. JBuilder, Visual Age, NetBeans)

- Environment
  - PIM: Programming Language
    - Higher level of expression
    - Easier to understand
    - Portable
    - Standardized
  - Generator, Projection
    - Compiler Engine
      - Dependable
      - Flexible
      - Configurable
      - Optimizing
      - Complete: Linker, Debugger, Etc.
  - PSM: Diverse HW/OS Platforms
MDA= New automation levels ~ Last 8 Years
http://www.omg.org/mda

Architectural IDEs

Environment

PIM
Model (UML, BOM...) &
Modeling Style (J2EE, eEPC, COBOL, .NET...)

Generator, Projection

Generator Engine

Models to Code
Models to Models

PSM
P-Stack: A Level of Automation

• Dependable
• Flexible
• Configurable
• Debuggable
• Optimizing
• Complete

• Higher level of expression
• Easier to understand
• Portable
• Standardized

Architectural IDEs
http://www.io-software.com

The Unified Process

ArcStyler Core Modules

Open MDA/UML/XML Repository

Business Object Modeler
Pattern Refinement Assistant
UML Refinement Assistant
MDA-Engine Refinement Assistant
MDA-Engine with Meta IDE
Build, Deploy & Test Support

Optional Integrated Tools

IDS ARIS
Rational Rose
Programming IDE

Std. MDA Projections

J2EE/EJB, .NET
BEA WebLogic
IBM WAS NT, zOS
Borland, JBoss
Oracle, IONA

Architect Edition adds support for custom infrastructure
Towards XML Model Based Computing

- Step 1: Document Object Model
- Step 2: XML Data Binding
- Step 3: Standard XML Information Models
- Step 4: XML Application Services Frameworks
  - Processing, Rendering, Querying, Secure Messaging
- Step 5: XML-Based “Web Object Model”
  - Web Services Architecture
- Step 6: XML Model Driven Architectures (to come)

Agile Modeling & XP
http://www.agilemodeling.com/, http://www.agilemodeling.com/resources.htm

- Practices-based software process whose scope is to describe how to model and document in an effective and “agile” manner
- One goal is to address the issue of how to apply modeling techniques on software projects taking an agile approach such as:
  - eXtreme Programming (XP)
  - Dynamic Systems Development Method (DSDM)
  - SCRUM
  - etc.
- Using modeling throughout the XP lifecycle
- Agile Methodologies
  - Sample Project Development Methodology
Component Development with COM and CORBA

Client component

Component “Bus”

Server component

IDL Stub
**IDL Specification**

```idl
module Reservation
    {interface Flight_booking {
        Price ticket_price
            (in Date day, in Flight number)
            raises (invalid_date, invalid_number);
        exception invalid_date {Date when; };
        readonly attribute string name;
            ...
    }
};
```

---

**The Role of IDL**

- Provide language-neutral interface description of a module’s operations
- As a result, enable clients and suppliers written in different implementation languages
The Trouble with IDL

- Programmers must write IDL interface in addition to actual software
- If from an O-O language, IDL duplicates information present in the code, e.g. C++ header file
- Perils of duplication
- IDL compiler goes the wrong way!
- However: some tools ease the process.

Pre-.NET Approaches: Summary

- Object technology is best known basis
- Information hiding is key
- O-O sufficient by itself: need for autonomous components
- Current approaches too heavy: extra work to turn module (e.g. class) into component
- IDL is a killer
Application Platforms Today

Browser Apps  Web Services Apps  Local Apps  Other Apps

GUI Services  Transaction Services  Web Scripting  Data Access  Remote Access  More

Standard Library

Runtime Environment

Operating System

Current Web App. Environment: .NET vs. Java

.NET Windows Client

HTML JavaScript Browser

Wireless Device

Desktop  Web Server  App Server  DB Server

ASP.Net IIS  WebService SOAP  JSP/Servlets WebSphere  VS.NET COM+  ADO.NET  EJB  JDBC

Relational Database System
.NET Framework
Design Goals

- Dramatically simplifies development and deployment
- Unifies programming models
- Provides robust and secure execution environment
- Supports multiple programming languages

So What Really Is .NET?

- Software for connecting people, information, systems and devices?
  - Runtime for executing code
  - Tools to help develop applications
  - Server products to manage applications
  - Value-added services
- Managed environment for developing and executing components
.NET Framework Design Goals

- Incorporate Web standards and best practices
  - Designed with security in mind
- Unified programming models
- Simplified development
- Simple to deploy, run, & maintain

The .NET Evolution

Before COM, applications were completely separate entities with little or no integration.
The .NET Evolution

COM provides a way for components to integrate; however, each component must provide the “plumbing” and objects cannot directly interact.

With the .NET Framework common language runtime, components are built on a common substrate; no “plumbing” is needed and objects can directly interact.
Common Language Runtime (CLR)

- Base Class Library Support
- Thread Support
- COM Marshaler
- Type Checker
- Exception Manager
- Security Engine
- Debug Engine
- IL to Native Compilers
- Code Manager
- Garbage Collector
- Class Loader

Multilingual Development

- Fortran
- Smalltalk
- Ada
- Oberon
- Visual Basic
- Delphi
- Component Pascal
- Java
- Haskell
- Pascal
- C++
- C
- Mondrian
- RPG
- Visual Basic
- Objective Caml
- ML
- JScript
- J#
What Is A Web Service?

A programmable application component accessible via standard Web protocols

- Provide a Directory of Services on the Internet
- Web Services are defined in terms of the formats and ordering of messages
- Web Services consumers can send and receive messages using XML
- Built using open Internet protocols

What Is A Web Service?

ASP .NET Web Matrix Project

- Lightweight, simple, community-oriented tool for building ASP.NET apps
- Full WYSIWYG support
- Small (~ 1.4 Mb)
- Community features
  - IM integration, code sharing, chat features
- Available free-of-charge at www.asp.net
The Role of Open Standards

- .NET has been ratified through ECMA standards
  - ECMA-335 Common Language Infrastructure (CLI)
  - ECMA-334 C# Language
  - Participants include Intel, HP, IBM, Netscape
- .NET is Microsoft’s commercial implementation of these standards
- Some platform specifics have not been included
  - Windows Forms
  - ADO.NET
### Shared Source CLI (Rotor)

- Non-commercial implementation
- Available for research, academic and other non-profit use
  - Standards are available to everyone for profitable use too, of course
- Written entirely in C#
- Available for FreeBSD and WinXP
  - Ximian and Intel are developing Linux implementations at the moment
- Implemented for teaching, learning and research

### Compilation And Execution

<table>
<thead>
<tr>
<th>Compilation</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Code</td>
<td>Native Code</td>
</tr>
<tr>
<td>Language Compiler</td>
<td>JIT Compiler</td>
</tr>
<tr>
<td>Code (IL)</td>
<td>At installation or the first time each method is called</td>
</tr>
<tr>
<td>Metadata</td>
<td></td>
</tr>
</tbody>
</table>

At installation or the first time each method is called.
Simplified Development

- Completely eliminates COM plumbing
- No more…
  - Registration => self described apps
  - GUIDs => hierarchical namespaces
  - .IDL files => unified object model
  - HRESULTs => structured exceptions
  - IUnknown => common root object
  - AddRef/Release => garbage collector
  - CoCreateInstance => “new” operator

Simplified Development (continued)

- Common Type System
  - Common instance and type definition
- Enables clean OO programming
  - Classes and interfaces
  - Constructors, properties, methods, events
  - Cross language inheritance
- Built-in interoperability
  - With COM
  - With native (Win32 style) DLLs
**Robust Environment**

- Automatic lifetime management
  - All objects are garbage collected
- Exception handling
  - Error handling first class and mandatory
- Type-safety
  - No buffer overruns, No unsafe casts, Uninitialized variables

**Secure Environment**

- Security designed-in
- Code access security enforcement
  - Security based on the identity of code
  - Administratively configurable via policy
- ASP.NET integrated authentication of user
  - Windows identity, Passport®, forms-based, …
- Cryptography library with XML DSIG support
  - Digital signature for XML (www.w3.org/signature)
Simplify Deployment And Management

- Zero-impact install
  - Applications and components can be shared or private
- Side-by-side execution
  - Multiple versions of the same component can co-exist on a system
- Assemblies
  - Contain dependency information

Unified Programming Models

Consistent API availability regardless of language and programming model

-.NET Framework
  RAD, Composition, Delegation
  Subclassing, Power, Expressiveness
  Stateless, Code embedded in HTML pages

VB Forms
MFC/ATL
ASP
Windows API
How Much Simpler?

Windows API

```vbnet
HWND hwndMain = CreateWindowEx(
    0, "MainWClass", "Main Window",
    WS_OVERLAPPEDWINDOW | WS_HSCROLL | WS_VSCROLL,
    CW_USEDEFAULT, CW_USEDEFAULT,
    CW_USEDEFAULT, CW_USEDEFAULT,
    (HWND)NULL, (HMENU)NULL, (HWND)NULL, (HMENU)NULL, hInstance);
ShowWindow(hwndMain, SW_SHOWDEFAULT);
UpdateWindow(hwndMain);
```

.NET Framework

```vbnet
Dim form As New Form()
form.Text = "Main Window"
form.Show()
```

Framework, languages and tools

- VB
- C++
- C#
- JScript
- J#

- Common Language Specification
- ASP.NET: Web Forms, Web Services, Mobile Internet Toolkit
- Windows Forms
- Data (ADO.NET) and XML
- Base Class Library / System Classes
- Common Language Runtime
- Windows
- COM+ Services

Visual Studio.NET

Web Matrix
Factored And Extensible Framework

- The Framework is not a “black box”
- Any .NET class is available for you to extend through inheritance
  - Gives developers much more head room
- Plug and Play components and subsystems

The .NET Framework Library

- ASP.NET
  - Web Forms
  - Web Services
  - Mobile Internet Toolkit
- ADO.NET and XML
- Base Class Library
- Windows Forms
The .NET Framework Library

Base Framework

System
- Collections
- Configuration
- Diagnostics
- Globalization
- IO
- Net
- Reflection
- Resources
- Security
- ServiceProcess
- Text
- Threading
- Runtime
- InteropServices
- Remoting
- Serialization
Data And XML

System.Data
- OleDb
- Common
- SQLClient
- SQLTypes

System.Xml
- XSLT
- XPath
- Serialization

Web Components

System.Web
- Services
  - Description
  - Discovery
  - Protocols
- UI
  - HtmlControls
  - WebControls
- Caching
- Configuration
- Security
- SessionState
The .NET Platform is language neutral
  » All .NET languages are first class players
  » You can leverage your existing skills

Common language specification
  » Set of features guaranteed to be in all languages

.Net provides
  » Visual Basic®, C++, C#, J#, JScript®

Third-parties are building
  » APL, COBOL, Delphi, Pascal, Eiffel, Haskell, ML, Oberon, Perl, Python, Scheme, Smalltalk…
Standardization

- A subset of the .NET Framework and C# submitted to ECMA
  - Adopted as International standards in Dec. 2001
  - In the ISO fast-track process now
  - Co-sponsored with Intel, Hewlett-Packard
- Common language infrastructure
  - Based on common language runtime and base framework
  - Layered into increasing levels of functionality

The Basic .NET Component Unit

Result of compilation is an assembly
Assembly is a set of classes
Unit of:
- Delivery and deployment
- Naming (through namespaces)
- Security (level for granting permissions)
- Versioning & side-by-side execution

Self-documenting through “manifest”
The Single Product Principle

- There is one product: SOFTWARE

  e.g., Eiffel: use single, seamless notation throughout the development cycle
  Rely on tools to produce documentation
  Maintain just one product!

Single Product Principle in .NET

- It’s all in the metadata!

  Definition:
  - Metadata is information about a module’s external properties, not necessarily needed for executing the module, but retained after compilation along with the binary code
Examining an Assembly with ildasm

How is the Metadata Stored?

- Stuffed into PE (Portable Executable) format
- To Windows, result of compiling an assembly looks like ordinary binary (*.dll or *.exe), but contains all kinds of supplementary information
- Will execute on .NET only
The Portable Executable (PE) Format

<table>
<thead>
<tr>
<th>DOS 2.0 Compatible EXE Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
</tr>
<tr>
<td>OEM Identifier</td>
</tr>
<tr>
<td>OEM Info</td>
</tr>
<tr>
<td>Offset to PE Header</td>
</tr>
<tr>
<td>DOS 2.0 Stub Program &amp; Relocation Information</td>
</tr>
<tr>
<td>Unused</td>
</tr>
<tr>
<td>PE Header</td>
</tr>
<tr>
<td>(aligned on 8-byte boundary)</td>
</tr>
<tr>
<td>Object Table</td>
</tr>
<tr>
<td>Image Pages</td>
</tr>
<tr>
<td>import info</td>
</tr>
<tr>
<td>export info</td>
</tr>
<tr>
<td>fixup info</td>
</tr>
<tr>
<td>resource info</td>
</tr>
<tr>
<td>debug info</td>
</tr>
</tbody>
</table>

Execution Model: the Role of Metadata

- C#
- VB
- C++
- Eiffel

.NET languages

Language compilers

IL code (plus metadata)

Loader

JIT + verifier

Managed code

Execution

Unjitted routine call
### Metadata Contents

- **Manifest**: assembly description
  - Name, version, culture
  - Security properties: Needed permissions
  - Public key if present
  - Dependencies on other assemblies
- **List of classes**
  - For each class:
    - **Features**: methods, fields, properties, events
    - **Signatures** (argument and result types) of each
    - **Interfaces** it implements
- **Custom attributes** (see next)

### Working with Metadata

Letting a program access metadata:
- Use `System.Reflection`

Examining metadata interactively:
- Use `ILDasm`

Can convert metadata to:
- XML
- COM type libraries (regasm)

To *produce* metadata:
- Use `System.Reflection.Emit`
.NET Component Model

- Single product principle: full reflectivity; assemblies are self-documenting
- (Groups of) classes directly yield components
- No extra plumbing
- No IDL
- Full application of Object-Oriented principles
- Multi-language interoperability

Component-Based Software Development

- Creating re-usable language-neutral software components that conform to a binary component standard
- Removes many of the problems faced in trying to develop re-usable pieces of software
- Many standards exist – COM, CORBA, Java Beans, .Net
- 2 MS standards – COM (Component Object Model) and .NET
Problems with Traditional Development

Lack of binary compatibility
- XYZ Corp markets a C++ library containing a class MyClass
- MyClass is extremely useful so ABC Software Company creates a dozen of applications that use MyClass
- XYZ Corp decides to fix some bugs in MyClass. In doing so, they add another private class member.
- The new class is not binary-compatible with old one – ABC Software will have to recompile all its applications

Language compatibility
- Without COM, software developed in one language cannot be easily used from another
- C/C++ and Visual Basic example:
  - VB can use C-functions but only in a limited manner
  - VB cannot use C++ classes
  - C/C++ cannot use functions/procedures/classes developed in VB
- Result: headache for library vendors and users alike
Problems with Traditional Development

**Location dependence**
- Without COM, the code using a library is locked into having the library at a particular location, e.g. in a DLL sitting in a system directory.

COM to the Rescue

- Complete binary compatibility – new versions of a library can simply replace the old version. It’s not necessary to recompile the code that uses that library.
- Language independence:

| VB / C / C++ / FORTRAN / JavaScript / … | C++ | VB | C | Fortran |

COM components
**Conventional Solution**

- “Portable” base class library developed in C++ using only ANSI C++ features
- COM wrappers for the classes to make them accessible from other languages

---

**Conventional Solution (continued)**

![Diagram showing relationships between classes and environments]

- User desktop application in VB
- FORTRAN program
- Server-side script - JScript

**Base library (ANSI C++ only, portable)**

**COM wrappers (C++, tied to Windows)**

**IArray1D** **IArray2D** **IWave1D** **IWave2D**

**CArray1D** **CArray2D** **CWave1D** **CWave2D**

**CInternal1** **CInternal2**
Pros and Cons

Pros:
- Library can be used from a variety of languages
- Binary compatibility is maintained across versions
- Base class library is portable to all platforms supporting C++

Cons:
- Library has to be developed in a single language – C++/C
- Difficult to develop portable code
- Portability not complete due to differences in compilers
- Source code has to be distributed
- Base library cannot access rich functionality provided by the platform in runs on

The .Net Solution

What is .Net?
- Execution environment with managed memory
- Rich class library - .Net Framework
- Common Language Specification / Runtime that allows for seamless interoperability between different programming languages
How it Works

VB Source File → C# Source File → C++ Source File → Fortran Source File

VB.net compiler → C# compiler → C++ Net compiler → Fortran.Net compiler

Intermediate Language

OS / CPU independent

Just-In-Time Compiler

Executable code in memory

.Net Solution

Rich functionality provided by the .Net Framework:
- Essential classes: collections, string manipulation, math etc.
- Security
- User interface
- Networking
- Graphics
- Database access
- Etc.

Web Page

JScript

Application

VB

C#

Fortran

Others

VB

C++

Cobol

Base class library
More Pros and Cons

Pros:
- Seamless interoperability between different languages
- Binary compatibility across all CPUs and operating systems
- Effortless portability to all platforms supporting .Net
- Full interoperability with COM
- Rich platform functionality can be used in the base library
- Possibly reduced development time and increased robustness due to services provided by .Net e.g. memory management

Cons:
- Possible portability problems:
- A lot less platforms will be supported
- Cost of .Net implementations for non-Windows platforms is unknown
- Time of availability of non-Windows implementations is unknown
- Exactly which platforms will be supported is unknown
- Considerable performance penalty

.Net PetShop Logical Architecture
**.Net PetShop Physical Deployment Diagram**

Diagram showing the physical deployment diagram for .Net PetShop, with servers A and B running NOS, and the architecture involving presentation tier, business logic tier, and data access layer. Internet traffic is shared through NLB or similar load balancing technology.

**.Net PetShop 3.0 Application Architecture**

Diagram illustrating the 3.0 application architecture, with presentation tier using ASP.NET Web Forms and User Interface Processing, business logic tier using business components and enterprise services, and data access layer using DAL interface and factory, with specific technologies like Oracle DAAB, Oracle DAL, SQL Server DAL, and SQL DAAB.
Component Layers in .Net Applications/Services

Security Policies in .Net Applications/Services
- Security, Operational Management, Communication provided by Microsoft Windows OS Services
  - Active Directory Service
  - Message Queuing
  - Windows Management Instrumentation (WMI)
  - etc.
Microsoft .NET Platform
Smart Devices + Windows XP + Web Services

- [http://www.microsoft.com/net](http://www.microsoft.com/net)
- First set of Microsoft Web Services

.Net Summary

- .Net/Com+ makes it:
  - As easy to develop Server Components as it is to develop Client Components
  - As easy to deliver Enterprise applications and Web Services as it is to deliver workgroup apps!

- The .NET Framework:
  - Dramatically simplifies development and deployment
  - Unifies programming models
  - Provides robust and secure execution environment
  - Supports multiple programming languages
Summary of Pre-.Net Services in DNA

- Activation Services
  - DCOM Activation Framework
- Naming and Directory Service
  - DCOM Class and Object Naming (i.e., CLSIDs, ProgIDs, and Monikers)
- Trading Service
  - Microsoft Active Directory
- Transaction Service
  - COM+ MTS
- Messaging Service
  - COM+ MSMQ
- Local Transactions
- Distributed Transactions
- .NET Enterprise Services Transactions

Local Transactions

- A transaction is a set of separate but interdependent changes that are made to some persistent data and that must be performed together as an atomic operation

- ACID Properties:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

- Multiple select, insert, update, and delete operations are composed into a single atomic operation

- Local Transactions can be created by using:
  - Transact-SQL
  - ADO.NET
### Distributed Transactions

- **Microsoft Distributed Transaction Coordinator (DTC)**
  - Manages connections to multiple databases
  - Manages connections from multiple database clients
  - Coordinates work into a single transaction

### .NET Enterprise Services Transactions

- COM+ checks three bit flags to control transaction outcome and component life time
  - Per object context – Consistent flag and done flag
  - Per transaction – Abort flag
Overview of COM+ Services and the .NET Serviced Components

- COM+ Services
- COM+ Catalog
- COM+ Context
- COM+ Interception
- How the .NET Framework Integrates with COM+ Services
- How the .NET Serviced Components Works with COM+

COM+ Services

- COM+ is the application infrastructure in a multi-user environment that compose of the services that enable many users to access the application and underlying data at the same time.
- COM+ Services:
  - Transactions
  - Compensating Resource Managers
  - Resource Management
    - Just-in-Time (JIT) Activation
    - Object Pooling
  - Synchronization
  - Security
  - Loosely Coupled Events
  - Queued Components
**COM+ Catalog**

- All configuration data is kept in a database
- You use the catalog to determine run-time requirements when an object is activated
- The catalog can be accessed by the Component Services tool or administrative API

**COM+ Context**

- Context is a set of run-time requirements for a class
- The COM+ catalog stores run-time requirements as attributes, which are checked during activation
- Context properties flow with the method call, depending on configuration attributes
**COM+ Interception**

- Used to enforce context semantics
- Transparent to the caller and the called
- Activation interception and method-call interception

**How the .NET Framework Integrates with COM+ Services**

- The `System.EnterpriseServices` namespace provides necessary programming types
- Activation and interception are handled automatically through `ServicedComponent` class
How the .NET Serviced Component Works with COM+

Creating .NET Serviced Components

- Hosting Components in Component Services
- Transaction Attribute
- Transaction Voting
- Using the AutoComplete Attribute
- Setting Assembly Attributes
- Registering Assembly
Hosting Components in Component Services

- Add a reference to **System.EnterpriseServices** in your assembly
- The System.EnterpriseServices namespace provides:
  - **ContextUtil** class
  - **ServicedComponent** class
  - Assembly, class, and method attributes
- All component classes that need to be hosted within a Component Services application must inherit **ServicedComponent** class

Transaction Attribute

- Transaction attributes specifies how a class participates in transactions

```csharp
Imports System.EnterpriseServices
<Transaction(TransactionOption.Required)> Public Class bizStateU
    Inherits ServicedComponent
    ...
End Class
```

- Options:
  - Disabled, NotSupported, Required, RequiresNew, Supported
Transaction Voting - 1

- ContextUtil class provides transaction voting

```vbnet
Public Sub TransferClass(...)  
    Try  
        ...  
        ContextUtil.SetComplete()  
    Catch ex As Exception  
        ContextUtil.SetAbort()  
        Throw ex  
    End Try
    End Sub
```

Transaction Voting - 2

- Transaction Voting Options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Consistent Bit</th>
<th>Done Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetAbort</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>SetComplete</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>EnableCommit</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>DisableCommit</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

- Transaction is terminated when Done bit of all Context objects in transaction have True value
Using the AutoComplete Attribute

- AutoComplete attribute avoids using the SetAbort, SetComplete, and ContextUtil methods

```vbnet
<AutoComplete()> Public Sub SetStateU(ByVal pSQL As String)
    Dim comStateU As New SqlCommand(pSQL, conStateU)
    comStateU.ExecuteNonQuery()
    ' not SetComplete or SetAbort is required
End Sub
```
Setting Assembly Attributes

- The information is stored in the AssemblyInfo.vb file
  - ApplicationName
  - Description
  - ApplicationActivation
  - AssemblyKeyFile

- Create key file using Strong Name Utility sn.exe
  
  sn.exe -k StateU.snk

  `<Assembly: ApplicationName("StateU")>`
  `<Assembly: Description("State University Serviced Component")>`
  `<Assembly: ApplicationActivation(ActivationOption.Server)>`
  `<Assembly: AssemblyKeyFile("StateU.snk")>`

Registering Assembly

- Manual Registration:
  - Using .NET Framework Services Installation Utility – Regsvcs.exe to register and create Component Services application
  - Regsvcs.exe StateU.dll

- Automatic Registration:
  - Application registered on first use by client – Lazy Registration
DNA OMA Services

- Activation Services
  - DCOM Activation Framework
- Naming and Directory Service
  - DCOM Class and Object Naming (i.e., CLSIDs, ProgIDs, and Monikers)
- Trading Service
  - Microsoft Active Directory
- Transaction Service
  - COM+ MTS
- Messaging Service
  - COM+ MSMQ

Win32 Services

- Win32 executable that satisfy several properties
- Lifetime is controlled by the Service Control Mgr (SCM)
  - Service is registered with SCM, and understands and obeys SCM commands
- Service has its own login session or shares one with another service
  - Service runs as a user or local system and abides to applicable security
- Service implements a set of service-specific functions:
  - Starting up, message handler, communication back to SCM
COM+ Services

- COM+ Catalog (v.s. Windows Registry)
- COM+ Load Balancing
- COM+ In-Memory Database (IMDB)
- COM+ Object Pooling
- COM+ Queued Components
- COM+ Events
- C++ Compiler Changes

Agenda

1. Component-Based Technology
2. .Net Overview
3. .Net Services
4. .Net Interoperability Support
5. J2EE Services
6. Conclusion
.Net CLR Interoperability Support

- Object binding
  - Both early and late bound interfaces are supported
- Data marshaling and translation
  - Data type conversion is handled between managed and unmanaged data types
- Object lifetime management
  - Object references are managed to ensure that objects are either released or marked for garbage collection
- Object identity
  - COM object identity rules are enforced
- Exception and error handling
  - The runtime translates COM HRESULT values to .NET exceptions and vice versa

Calling a COM Component from .Net

- RCW: Runtime Callable Wrapper
**Runtime Callable Wrapper**

The RCW is a managed object and is allocated from the heap maintained by the CLR. As with any other managed object, references to the RCW are traced by the runtime, and the RCW is subject to garbage collection.

The RCW is responsible for the following interactions:
- Creating and binding to the underlying COM object
- Consuming COM interfaces and factoring the interfaces into a managed form
- Translating and marshaling data between environments
- Managing the lifetime of the wrapped COM object
- Translating COM HRESULT values into .NET exceptions

**Calling a .Net Component from COM**

- CCW: COM Callable Wrapper
COM Callable Wrapper

The runtime reads the type information for the COM component from its assembly metadata and generates a compatible CCW.

The CCW is responsible for the following interactions:
- Creating and binding to the underlying managed object
- Synthesizing several important COM interfaces (such as IUnknown and IDispatch) based on the object's type information
- Marshaling and translating data between environments
- Managing the lifetime of the .NET component
- Translating .NET exceptions into COM HRESULT values

COM+ Services

.Net components that participate in COM+ applications are called Serviced Components:
- Serviced Components must be registered in the COM+ catalog using the regsvcs tool (.Net Framework SDK)
- .Net components may be annotated using service-related attributes to specify the exact service requirements
- Serviced Components may share context, transactions, synchronization boundaries with COM+ components
Calling Native Win32 DLLs

.Net platform supports calling unmanaged coded in native Win32 DLLs:
- Interoperability mechanism is called Platform Invocation (P/Invoke)
- API has to be declared to the .Net runtime for each language
- .Net platform handles marshalling of data types, finds and invokes the correct function in the DLL, and manages the transition from managed to unmanaged code
- CLR also supports callbacks from API functions
- .Net does not currently support calling from a Win32 DLL into .Net managed code (must use COM interoperability then)

Web-Tier Horizontal Migration
Web-Tier Horizontal Migration (continued)

- Must translate ADO recordsets returned from the middle tier to ADO .NET datasets required by ASP .NET code, typically for data binding
- May want to enable the use of role-based security between an ASP .NET front end and a COM middle tier by properly configuring impersonation in the ASP .NET application
- Need to be aware of performance issues when communicating with STA-based COM components from managed code
  - .NET does not use COM apartments natively and joins a COM MTA by default when interacting with COM
  - This results in the intervention of a thread-switching proxy
- Must consider the interoperability and translation of managed and unmanaged data types
- Must deploy generated interoperability assemblies for middle tier COM components
- Must deploy the CLR on all Web servers
To transparently replace middle tier components with .NET components without affecting client code, you will need to maintain the original GUIDS and/or ProgIds of your COM components.

When attempting to transparently replace a COM component, you must properly handle replacement of the class interface generated by Visual Basic components.

You will need to translate the ADO .NET datasets returned from your migrated middle-tier components to ADO recordsets used in your original ASP code.

You must deploy the interoperability assemblies for the middle tier components.
Agenda

1. Component-Based Technology
2. .Net Overview
3. .Net Services
4. .Net Interoperability Support
5. J2EE Services
6. Conclusion

J2EE OMA Services

- Activation Services
  - RMI Activation Framework
  - JavaBeans Activation Framework
- Naming and Directory Service
  - JNDI and JNDI SPIs for CosNaming, RMI, NIS, NDS, LDAP
- Trading Service
  - Jini
- JTA and JTS
- Messaging Services
  - JMS
  - JavaMail
### CORBA OMA Services (for reference only)

- **Activation Services**
  - CORBA POA
  - CORBA Lifecycle

- **Naming Service**
  - CosNaming

- **Directory and Trading Services**
  - CosTrading

- **Object Transaction Service (OTS)**

- **Messaging Services**
  - CORBA Event Service
  - CORBA Notification Service

### Java Messaging Service

- **Handles both Synchronous and Asynchronous Messaging**
  - Topic / point-to-point messaging
  - Queue / publish-and-subscribe messaging

- **Common Way for Java Programs to Create / Send / Receive / Read Enterprise Messages**

- **Used in Conjunction with MOM Products**
  - e.g., TIBCO, MQSeries

- **Different Message Types**
  - TextMessages, MapMessages, ObjectMessages, etc.
// Lookup admin objects on default host
InitialContext ic = null;
ic = new InitialContext ();
ic.bind ();

// Lookup connection factory and Topic names
TopicConnectionFactory tcf =
    (TopicConnectionFactory) ic.lookup("primaryTCF");
Topic topic = (Topic)ic.lookup("primaryTopic");

// Dispose of InitialContext Resources
ic.dispose();

// Create and start a topic connection
TopicConnection topicConnection = tcf.createTopicConnection();
topicConnection.start ();
System.out.println("Creating topic session: not transacted, auto ack");
JMS Synchronous Subscriber Example (continued)

```java
// Create topic session on the connection just created
TopicSession topicSession = topicConnection.createTopicSession(false, 1);

// Create subscriber
TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);

// Listen for messages synchronously (blocking receive)
while (true) {
    TextMessage textmsg2 = (TextMessage) topicSubscriber.receive();
    System.out.println("Received : " + textmsg2.getText());
}
```

JMS vs. RMI

- **RMI**
  - Limited client connections
  - Poor performance (no resource pooling, no store-and-forward, no load balancing)
  - No guaranteed messaging and security
  - Static client/servers with location dependent code
- **JMS**
  - Specification / vendors address inherent RMI limitations
  - JMS may be implemented on top of RMI (java.rmi.server.RMISocketFactory) or sockets (java.net.Socket)
  - Socket-based implementations are better throughput
### JMS vs. JavaSpaces

- **JavaSpaces**
  - Distributed persistence and data exchange mechanism for Java
  - JavaSpace data is stored in entries (typed grouping of fields)
  - Client API allows writing, lookup, and removal of entries
  - Java programs/applets can use JavaSpaces to store their state
  - API includes support leasing, transactions, and events
  - Runs on top of RMI and provides an alternative to JMS
  - Only supports transient objects on a small scale
- **JMS**
  - Supports large scale transient objects
  - Designed to support access beyond firewalls
  - Supports high performance and response times

### JMS vs. InfoBus

- **InfoBus**
  - Solution for interconnecting Java components (e.g., Lotus’ Kona Applets) or arbitrary Java classes
  - Virtual traffic cop for components on a web page
  - Tags applets by content that can be exchanged among components
  - JavaBeans components registered can exchange data with applets
  - Provides data producer, consumer, and controller modules
  - Designed for components working together in the same JVM
- **JMS**
  - Supports communication across components that go beyond a single JVM
Java Shared Data Toolkit (JSDT)
- Library that allows the addition of collaborative features to Java applets and applications
- Supports the creation of network centric applications
  - e.g., shared whiteboards, chat environments, remote presentations, shared simulations, data distribution for enhanced group data workflow, etc.
- Supports many simultaneous users and uses raw IP delivery (highly collaborative, and high bandwidth services)
- Designed to work across the Internet (via HTTP) and firewalls

JMS
- Non real-time
- Can use raw IP for delivery or higher-level protocols (SMTP, RMI, Java sockets)

JMS vs. JSDT

When to Use JMS

- Use JMS in environments where CORBA has not been embraced and where Java applications needs to transact asynchronously
- Use RMI for synchronous messaging where performance is not an issue
- Although inefficient, JMS can be used across the firewall via HTTP tunneling (same as RMI/no data push from server to clients)
- Use to isolate programmers from low-level transport APIs while they focus on delivering enterprise-wide messaging
Component-Container Architecture

J2EE Container Environment
- J2SE Platform
- Java Enterprise APIs
- Java Enterprise Implementations
- Deployment Services
- Management Services

Component-Container Models
- EJB App Servers, Web App Servers (SRV/JSP), Applet Clients, Application Clients
Java Transaction Services (JTS)

- EJB model implements two-phase commit, transaction context propagation, and distributed transactions
- Transaction is an “ACID” unit of work
  - Atomic, Consistent, Isolated, Durable
- EJB transaction policies are defined at deployment time
  - Bean-demarcated transactions
  - Client-demarcated transactions
- Transactional scope
  - Scope includes any bean that participate in the bean method
- Transaction scope is traced by looking at the thread of execution, and is also determined by transaction attributes
Attributes may apply to a whole bean or individual methods
Six attributes are supported by container to specify the way it manages transaction:
- TX_NOT_SUPPORTED
- TX_SUPPORTS
- TX_REQUIRED
- TXQUIRES_NEW
- TX_MANDATORY
- TX_BEAN_MANAGED
**TX_NOT_SUPPORTED**

- Container invokes bean methods without a transaction context
- When bean method is invoked from within a transaction context, the container suspends that transaction context

**TX_SUPPORTS**

- Container includes the bean or method within the transaction scope in which it is invoked
- If the bean method is invoked without a transaction context, the container invokes the bean without a transaction context
- When bean method is invoked from within a transaction context, the bean and everything it accesses becomes part of the transaction
Containers invokes the bean within a transaction scope
When bean method is invoked from within a transaction context, the container invokes the bean from within that transaction context
When bean method is invoked without a transaction context, the container creates a new transaction context and passes it to any beans that are used by this bean method

Container always invokes the bean method with a new transaction context regardless of existing transaction contexts
New transaction context is passed to all beans or resources used by the bean method
**TX_MANDATORY**

- Container always invokes the bean method within the transaction context associated with the client
- `javax.jts.TransactionRequired` Exception is raised if the bean method is invoked without a transaction context
- Transaction context is passed to any beans used by the bean method

**TX_BEAN_MANAGED**

- Container does not manage the bean class’ transactional context
- The bean class uses JTA’s `javax.jtx.UserTransaction` to explicitly manage transaction boundaries
- If one method uses this attribute, all methods must manage transaction on their own
- More efficient for EJBs that provide stateless services
**SQL Transaction Isolation Levels**

- Determines how isolated transactions are for read purposes
- Isolation levels are defined in the ANSI/ISO SQL92 standards
- Prevents the following between concurrently executing transactions:
  - Dirty reads: transaction reading uncommitted data from another transaction
  - Non repeatable reads: transaction rereading data previously read and finding that another committed transaction has modified or deleted the data
  - Phantom reads: transaction executing a query and finding out that another committed transaction inserted addition rows in the set of records returned by the query

---

**EJB Isolation Levels**

- Levels mapped in JDBC to static variables defined in java.sql.Connection interface
- All methods invoked in the same transaction must have same isolation level
- TRANSACTION_READ_UNCOMMITTED
  - Transaction can read uncommitted data
- TRANSACTION_READ_COMMITTED
  - Transaction cannot read uncommitted data
- TRANSACTION_REPEATABLE_READ
  - Transaction cannot change data read by another transaction
  - More constrained than “read committed”
- TRANSACTION_SERIALIZABLE
  - Transaction has exclusive read/update privileges via data locking
TX SERIALIZABLE (continued)
- Does to data what the synchronized keyword does to methods
- Other transactions can neither write nor read the same data
- Most restrictive to guarantee the highest level of data integrity
- Slower as simple reads must wait in line
- EJBs need to be fine-tuned for performance by leveraging the level of reads that will occur on the database, and its handling of locking and choice of isolation level

EJB Isolation Levels

Client-Demarcated Transactions
- JTS specification is based on CORBA OTS 1.1
  - Implements a Java transaction manager that sits between an application and one or more transaction-capable resource managers (e.g., database servers, messaging systems, etc.)
- JTS includes the JTA API
  - JTA API can be used to group operations into logical transactions
- JTA services:
  - Transactional operations in client applications
  - Transactional operations in app. servers (on behalf of clients)
  - Global transaction management in a Java transaction manager (to coordinate multiple resource managers)
EJBs use JTA’s high-level transaction manager interface

- javax.transaction.UserTransaction interface is exposed to EJB programmers (required by EJB spec.) to communicate with the transaction manager and control transaction boundaries
- Important methods:
  - public void begin() - nested transaction support up to TM
  - public void commit() - may throw RollbackException
  - public int getStatus()
  - public void rollback() - remove Transaction/thread association
  - public void setRollbackOnly() - used to veto a transaction explicitly
  - public void setTransactionTimeout()

EJB server uses JTA’s high-level transaction manager interface and a standard Java mapping of the X/Open XA protocol (javax.transaction.xa.package)
JTA allows exposure of the UserTransaction object via JNDI
- Approach compromises portability as other EJB servers may not support this approach

```java
Context ctx = new InitialContext();
UserTransaction utx =
    (UserTransaction)ctx.lookup("ajndiname");
utx.begin();
// do work
utx.commit();
```

In the case of entity beans and stateless session beans:
- Transaction managed via UserTransaction must begin and end in the same method
- Reason is that this type of bean instances are shared across many clients via instance pooling and instance swapping on the server

Stateful beans:
- Allow UserTransaction object to span multiple method as there is only one instance associated with a client and it maintains conversational state
- Bean/transaction state is consistent even if the container may internally activate/passivate to leverage server resources

```java
public class MyStatefulBean implements SessionBean {
    public SessionContext ctx;
    public void setSessionContext(SessionContext ctx){
        this.ctx=ctx;
    }
    public void method1(){
        ctx.getUserTransaction().begin();
        // do some work
    }
    public void method2(){
        // do some more work }
    public void method3(){
        // do yet some more work and finally commit
        ctx.getUserTransaction().commit();
    }
```
Stateful beans (continued)
- Repeated calls to getUserTransaction() return a reference to the same UserTransaction object.
- UserTransaction.getStatus() may be called to check the UserTransaction state.
- Transaction state between method calls may be cached and database updates postponed to improve performance.
- javax.ejb.SessionSynchronization allows the server to inform a stateful session bean of the various stages in the transaction by invoking callback methods:
  - afterBegin(): notifies bean that a new transaction has started (invoked prior to the delegation of business methods to the instance).
  - afterCompletion(): notifies bean that transaction has completed (should be used to reset instance variables).
  - beforeCompletion(): notifies bean that transaction is about to be committed.
- Client-demarcated transactions across methods should be avoided as they are overly complex and may lock up resources.

Exception handling depends on the type of exception, isolation level, and transactional attribute of the bean method.
- Any exception thrown outside the transaction scope causes a rollback.
- Example involving two stateless session beans (MiddleBean/TranstestBean):
Stateless beans deployment scenarios for the example suggested in the previous slide:

<table>
<thead>
<tr>
<th>Parent (MIDDLEBEAN)</th>
<th>Child (TRANSTESTBEAN)</th>
<th>Transaction Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX.Requires_New</td>
<td>TX.Requires</td>
<td>Rollback because of second bean transaction context</td>
</tr>
<tr>
<td>TX.Requires_New</td>
<td>TX.Not_Supported</td>
<td>No rollback in second bean because it is not in context.</td>
</tr>
<tr>
<td>TX.Requires_New</td>
<td>TX.NotRequires_New</td>
<td>One exception-throwing method is rolled back, the other is not, because each is in a new context.</td>
</tr>
</tbody>
</table>

**Table 7:** The three scenarios that occur when this deliberately planted exception is thrown

Illustrated explanation of the three possible scenarios that result from throwing a deliberately planted exception as per the example suggested:
Exception Handling and Transactions (continued)

- Example Case 1:
  - Transaction context is propagated to TranstestBean
  - When exception is thrown in second bean, it falls within the transaction context and propagates up
  - Container traps the exception in the first bean and rolls back the transaction

- Example Case 2:
  - Second bean does not participate in the transaction, and the transaction context is not propagated to it
  - Exception thrown falls outside the transaction context and is detected by the container that rolls back the transaction
  - There is no way to undo any changes made by the second bean

- Example Case 3:
  - The second bean has a new transaction context for each method
  - The transaction of the exception-throwing method is rolled back
  - The exception moves up and the container rolls back the initial transaction
  - Second method executes successfully in its own transaction

Exception handling guidelines:

- If an exception is meant to signify that the method cannot complete successfully, it should not be caught
- If an exception is caught, the method should try to correct the problem and continue
- In order for the transaction to be rolled back, a method must throw an exception and propagate out of it
- Application exception will not cause rollback if they are thrown and caught within the transactional scope
- Runtime or unchecked exceptions always cause a transaction to rollback regardless of the transaction attribute or transactional scope
Unilateral Decisions:
- Transaction manager allows certain heuristic/speculative decisions based on the state of participating resources in a transaction and the underlying two-phase commit protocol
- Decisions occur when one of the resources in the transaction unilaterally decides to commit or rollback the transaction (without TM permission)
- Resulting exceptions caused by breaking a transaction atomicity:
  - `javax.transaction.HeuristicCommitException`
    - Rollback requested but all updates were committed instead
  - `javax.transaction.Heuristic.MixedException`
    - Some updates were committed and others were rolled back
  - `javax.transaction.HeuristicRollbackException`
    - All relevant updates were rolled back

Java and Database Technology

- **JavaSpaces**
  - Create and store objects with persistence
  - Allow process integrity

- **JDBC Data Access API**
  - Access tabular data sources from Java

- **J2EE database access and container managed persistence**

- **Pjama: Orthogonal Persistence for the Java Platform**
  - [http://www.sun.com/research/forest/opj.main.html](http://www.sun.com/research/forest/opj.main.html)
  - [http://www.dcs.gla.ac.uk/pjava/](http://www.dcs.gla.ac.uk/pjava/)
See Persistence Service Interface for Entity Beans:

See JBoss Persistence Manager Architecture:
- http://javatree.web.cern.ch/javatree/share/opt/net/jboss-2.0/docs/container.html
- JAWS (Just Another Web Store) is the default CMP (Container Manager Persistence) plugin that performs basic O/R functionality against a JDBC-store

See Patterns for Object Relational Access Layers
- http://www.objectarchitects.de/ObjectArchitects/orpatterns/EJBPersistence

### EJB Persistent Service Architecture

- Complexity of an optimal access layer
  - Object to Tuple Layer Logical Mapping
  - Physical Layer Mapping
    - Caching to reduce database traffic and disk I/O
    - Data Clustering
  - Application must still be maintainable at a reasonable cost

- See Session 6 Handouts and Presentations:
  - “Efficient CMP Development”
  - “Enterprise JavaBeans Patterns”
  - “Designing Databases for eBusiness Solutions”

### Persistence Service Performance Issues
EJB Component/Programming Model

- Represent sets of data (all or part of a database table or a view)
- Functionality limited to creation, update, and deletion of data
- Manage persistence of data
- Maintained in a cache
- Can be container or bean managed
  - Container-managed beans are under the control of an application server for persistence and transaction management
  - Container-managed beans are restricted in the type and complexity of data they can manage
  - Bean-managed beans rely on user provided code for persistence and transaction management
Session Beans in EJB Application Servers

- Handle the business logic of EJB applications
- May use multiple entity beans to gather application data

Anatomy of a CMP Bean

- EJB remote interface (extends `javax.ejb.EJBObject`)
  - Contains method signatures for any method that accesses or modifies data stored in the bean
- EJB remote implementation (implements `javax.ejb.EntityBean`)
  - Provides an implementation of all the methods defined in the remote interface, in addition to methods required by the application server
- EJB home interface (extends `javax.ejb.EJBHome` interface)
  - Declares method signatures for any method that creates new instances of the bean, and for all methods that are used to retrieve instances of the bean (finder methods)
- EJB key
  - Contains the unique primary key implementation for the bean
  - any class that is a legal value type in RMI-IIOP
- EJB finder helper interface (optional)
  - Contains one static java.lang.String field for each finder method declared in the EJB home interface
  - Strings are initialized with SQL queries executed dynamically when bean instances are retrieved in a finder method
  - Some server ignore this file and put queries in XML deployment descriptors using proprietary query language formats
- Deployment descriptor (XML -> serialized data object)
  - Names of EJB classes/interfaces, list of persistent fields, etc.
- Database scripts
  - Application server may generate them

**Container-Managed Persistence Architecture**

*Figure 1: Container managed persistence architecture "thin bean fat containers"*
Bean-Managed Persistence Architecture

Figure 2: Bean Managed Persistence Architecture “fat bean thin container”

EJB Development Approach #1

- Handcode Everything:
  - EJB remote interface
  - EJB remote implementation
  - EJB home interface
  - EJB key
  - EJB finder helper interface
  - Deployment descriptor
  - Database scripts
EJB Development Approach #3:
Handcode the Bean and Descriptor

EJB Development Approach #3:
Graphical Bean and Descriptor Generator
Four Patterns for Managing Persistence

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Entity Mgt.</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Container-managed Entity Beans with Session Beans</td>
<td>Container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2. Bean-managed Entity Beans with Session Beans</td>
<td>Bean</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3. Entity Beans Only</td>
<td>Container</td>
<td>Yes</td>
<td>No*</td>
</tr>
<tr>
<td>4. Session Beans Only</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Session beans are used in this example for complex database interactions, but they do not serve as a front-end for Entity beans. Instead, they access the database directly as an Entity bean would.

Container-Managed Entity Beans + Session Beans

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Beans-Managed Entity Beans + Session Beans

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bean</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Entity Beans Only

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
**Session Beans Only**

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Performance Characteristics**

**Pattern Comparison**

400 users, 50 iterations, 4 server instances/200MB

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Only</td>
<td>6.26</td>
<td>0.20</td>
<td>0.15</td>
<td>0.02</td>
<td>0.612</td>
</tr>
<tr>
<td>Container Managed</td>
<td>6.74</td>
<td>0.31</td>
<td>0.79</td>
<td>0.04</td>
<td>0.706</td>
</tr>
<tr>
<td>Bean Managed</td>
<td>7.02</td>
<td>0.31</td>
<td>0.95</td>
<td>0.04</td>
<td>0.687</td>
</tr>
</tbody>
</table>
Performance Characteristics (continued)

Pattern Comparison
100 users, 50 iterations, 4 server instances/200MB

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Only</td>
<td>4.49</td>
<td>0.30</td>
<td>0.35</td>
<td>0.08</td>
<td>0.426</td>
</tr>
<tr>
<td>Container Managed</td>
<td>1.52</td>
<td>0.06</td>
<td>0.07</td>
<td>0.04</td>
<td>0.112</td>
</tr>
</tbody>
</table>

SQL Statements vs. Finder Methods

SQL vs. Finder Method Comparison (for getting lists of data)

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>5.54</td>
<td>0.20</td>
<td>0.15</td>
<td>0.04</td>
<td>0.571</td>
</tr>
<tr>
<td>Finder Methods</td>
<td>9.99</td>
<td>1.06</td>
<td>2.92</td>
<td>0.05</td>
<td>1.457</td>
</tr>
</tbody>
</table>
Summary of Findings

- Pattern 1: (better choice)
  - Great for design and flexibility
    - Good separation of presentation, data, and business logic
  - Fast time to market
- Pattern 4:
  - Best for performance and throughput
- Pattern 3:
  - Faster time to market than pattern 1
    - No need to code session beans
    - Prohibitive performance cost

RedHat JBoss JMX Infrastructure

- JTS/JTA
- SECURITY
- DATA SOURCES
- REMOTE MANAGEMENT

JMX Implementation

- EJB CONTAINER
- DATABASES
- JAVA SERVER PAGES
- JMS

TomCat Servlet Container
Jetty Web Server/Servlet Container
RedHat JBoss
(standard ejb-jar.xml security elements)

Oracle WebLogic Application Server Platform
Oracle WebLogic Database Connectivity

Agenda

1. Component-Based Technology
2. .Net Overview
3. .Net Services
4. .Net Interoperability Support
5. J2EE Services
6. Conclusion
Summary

- OMG, Sun, and Microsoft DOC platforms all share the OMA architecture
- OMG’s OMA provides the most comprehensive and detailed set of specifications for CORBA facilities, and CORBA services
- JavaSoft has focused mostly on OMA services as part of the J2SE, J2EE, and J2ME architectures
- Microsoft provides platform specific support for services and an integrated set of horizontal and vertical facilities bundled with the OS

Summary (continued)

- JWS provides the convenience of desktop applications for downloaded applications with Zero Client Administration
- Microsoft’s DOC platforms implement the OMA architecture
  - Provide platform specific support for services
  - Support an integrated set of horizontal and vertical facilities
  - Capabilities are bundled with the Operating System
- COM+ introduces a new naming service technology
- DNA provide transactions and messaging services
- DNA support web-enabling which is being extended with trading capabilities via Web Services
- .Net/COM+ simplifies application development
  - Server Components v.s. Client Components
  - Enterprise apps & Web Services v.s. workgroup apps
Class Project

- Project Description
  - The project focus is two-fold:
    - Based on a framework-based enterprise application of your choice, you will implement common facilities and application-level services on top of various types of application server technologies to support the various aspects of your chosen application.
    - As you transition from one platform to another you will study and develop model-based migration and interoperability tools that leverage off of the cutting-edge concepts subsumed by modern Model Driven Architectures (MDAs).

Readings

- Readings
  - Handouts posted on the course web site
  - Explore .Net/COM+ Environment
  - Read related white papers/documentation on the .Net/COM+ environments
  - Explore J2EE environments
  - Read related white papers/documentation on the J2EE environments
Assignments

- Assignment:
  - #3a: Investigate J2EE/.Net development environments. Write a short report that documents your findings and recommendations with respect to selection criteria in support of development environments for application server technologies covered in this session
  - #3b: See homework #3 specification

Project Related Frameworks

- Project Frameworks Setup (ongoing)
  - Apache Web Server (www.apache.org)
  - Perl (www.perl.com)
  - Microsoft IIS with COM+.Net and ASP
  - Apache Tomcat
  - Adobe JRun4
  - Apache Cocoon 2/XSP
  - Visibroker, Orbacus
  - RMI-IIOP
  - Oracle WebLogic Suite 11g, IBM WebSphere V7, JBoss 3.x
Next Session: Service Oriented Architecture