Application Servers

Session 4 – Main Theme
JEE and .Net OMA Implementations
(Part 1)

Dr. Jean-Claude Franchitti

Icons / Metaphors

- Information
- Common Realization
- Knowledge/Competency Pattern
- Governance
- Alignment
- Solution Approach
**Web Application Servers Architectures (Evolution)**

- Traditional client-server technology
- CGI frameworks
- Page-based extended HTML environments
- Distributed object computing platforms
- Java-Based
- Object Management Architectures (OMAs)
- Component-based computing environments
- Web Services platforms
- Next generation application servers (reflective, multimedia- and agent enabled, MDA-compliant, etc.)

---

**Web Application Servers Architectures (Page-Based Extended HTML Environments Details)**

- Application Servers for Enhanced HTML (traditional)
  - a.k.a., Page-Based Application Servers
  - Tag-Oriented (e.g., Macromedia ColdFusion 5.0 Server)
  - Script Oriented (e.g., Microsoft IIS with ASP, PHP)
  - Mostly Used to Support Standalone Web Applications
  - Typically less expensive than standalone and IDE-based servers
  - HTML-based development
- New Generation Page-Based Script-Oriented App. Servers
  - First Generation Extensions (e.g., Microsoft IIS with COM+/ASP)
  - Servlet/JSP Environments
  - XSP Environment
  - Can now be used as front-end to enterprise applications
  - Hybrid development environments
**Distributed Object Computing Platforms**
- Provide an infrastructure for distributed communications enabling
- Still need to merge traditional web-oriented computing with object computing

**Object Management Architectures**
- DOC Platform + APIs to reusable services and facilities

**OMAs + Component Models -> JEE, CCM, DNA, ASP/.Net**

---

**CORBA OMA Services**

- **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
Traditional Web and Client Application Services

- Activation Services
  - Client: MIME Handler, and Applet Activation
  - Web Server: File-Stream, Process, Script Engine, Servlet, JSP, and EJB Activation
- Naming and Directory Service
  - DNS and Java-Based DNS Interfacing
  - NIS, NDS, LDAP
- Custom Transaction Services
  - Based on Encina, Tuxedo, etc.
- Custom Messaging Services
  - TIBCO
  - MQSeries

Agenda

1. Towards Microsoft’s .Net Component Model
2. Towards Sun’s JEE Component Model
3. Extensible Server Pages (XSP) Environment
4. Conclusion
Microsoft Active Platform
(Logical Architecture)

Active Platform

Microsoft Active Platform
(Features)

- Logically centralized architecture
- Physically de-centralized architecture
- Scripting
  - Client-Side: Improved user interface and data validation
  - Server-Side: Business rules and data access
  - VBScript and JavaScript built-in support
  - PerlScript support via ActiveX scripting engine (e.g., ActiveState’s)
  - Ability to mix scripting languages
- Active Server Components
  - Provide OS services to ASPs
  - Encapsulate business rules to ease programming
  - e.g., TextStream Component (FileSystem object)
**Microsoft Active Platform**

*(Physical Architecture)*

**Microsoft IIS - COM+/.Net - ASP**

- **IDE:**
  - Visual InterDev (ASP)
    - Management of site development process / Scripting
  - Alternative IDEs
    - Macromedia Drumbeat, Ultradev, Dreamweaver
    - NetObjects Fusion
    - Microsoft FrontPage 2000
    - Adobe GoLive
    - Visual Studio .Net

- **Server Platforms**
  - Windows 2000/NT
  - Use ChiliSoft (now Sun's/Oracle's) for other platforms
  - Platforms: Solaris, Linux, AIX, HP-UX, Windows
    - Immunity from current IIS attacks
      - e.g., Code red worms on Win 2000
    - Web Server: Apache, iPlanet
    - ChiliBeans support for Java
      - Similar to Microsoft with COM+ & ASP for C++
Microsoft IIS - COM+/.Net - ASP

- COM+ / ASP
  - Equivalent to JEE EJB / JSP
  - Included in Microsoft Windows 2000 Advanced Server
- COM+
  - Attributes (declare what runtime services are needed by the component)
  - Threads and database connection pools (access via Active Data Object API)
- ASP Object Model sub-systems
  - HTTP request
  - COM+ transaction
  - External COM+ components
- Other solution components:
  - Visual Studio 6.0-8.0 / .Net
  - Internet Security and Acceleration Server (ISA)
    - static content caching, fault tolerance, load balancing, request handling

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
Microsoft Component Object Model
(Logical Architecture)

MTS Components

ActiveX

Compound Documents (OLE)

MTS

COM Infrastructure

Distributed InterNet
Applications Architecture (DNA)

PRESENTATION SERVICES
Rich Client: EXE-based using scripting and WIN32 services directly
Thin Client: Web pages based either on HTML or Dynamic HTML

APPLICATION SERVICES
Internet Information Server
Microsoft Transaction Server
Active Server Pages
Microsoft Message Queue Server

TCP/IP, NetBIOS

DATA SERVICES
Active Data Objects (ADO)
OLE DB
COM Transaction Integrator (COMTI)
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

COM+ = DCOM/COM + MTS

http://members.tripod.com/gsraj/misc/ejbmts/ejbmtscmp.html
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

DCOM Class and Object Naming
(file moniker file: bind)

1. COM locates the file, and asks for object class id
2. If step 1 failed, COM searches the registry for object's class id
   'Excel Sheet 5'
3. COM creates the object and passes the file name
4. Object initializes directly to the file
Display name for class monikers:

- `display-name = "CLSID:" string-clsid-no-curly-braces
  "["; clsid-options] ":"
- `clsid-options = clsid-param ":" value`
- `clsid-param = none currently defined`

C++ example (tell moniker to use an object that can read a document instead of the document itself):

```cpp
ProgIDFromCLSID( &clsid, "xyz.activator.1")
CreateClassMoniker( clsid, &pmkClass )
MkParseDisplayName( pcb, "\northamerica\central\employee.doc", &dwEaten,
  pmkFile )
pmkFile->BindToObject( pcb, pmkClass, IID_IDispatch, &pDisp )
```
COM+ Catalog vs. Windows Registry

- Stores COM+ application attributes, class attributes, and computer-level attributes
- Guarantees consistency among attributes
- Provide common operations on top of attributes
- Two different stores
  - COM+ registration database
  - Microsoft Windows Registry (HKEY_CLASSES_ROOT)
    - COM components that do not require new COM+ services
    - Type library
    - Interface proxy/stub registration
  - Unified logical view via COM+ Admin Library
COM+ Queued Components

COM+ Events

- **Publishers**: Advertise events by creating and publishing an event class
- **Subscribers**: Subscribe to events by subscribing to a specific event class

COM+ events store event classes and subscriptions. COM+ later binds them together with an intermediate object - an instance of the event class. Publisher fires an event, event class receives it, and invokes the appropriate subscriber.
C++ Compiler Changes

- Sample Attributes:
  - in/out direction of parameter in a method
  - threading model
  - component housing (i.e., DLL or EXE)

Creating an MTS Component

- Example:

```java
try {
    // create the MTS component
    bank.IChecking server =
        (bank.IChecking) new bank.Checking();

    // invoke business methods on the component
    server.createAccount (1234, "Athul", 1000671.54d);
}
catch (Exception ex) {
    ex.printStackTrace();
}
```
**.NET Framework**

- **Common Language Runtime (CLR)**
  - Accessible to any compiler that produced Microsoft Intermediate Language (IL) code
- **Runtime Classes**
- **Visual Studio.Net (VS.Net)**
  - Includes VB.Net, VC.Net (C#), and Visual C++ .Net

---

**.Net Remoting**

(http://www.dotnetremoting.cc/)

- Native .Net Remote Object Invocation
- Allows Object Invocation and Communications Across Processes and Machines
- Wire Protocol and Format Independent
- Extensible via Custom Formatters (SOAP, Binary, etc.), Channels (HTTP, TCP, etc.), and Sink Chain (i.e., Channels + Formatters) Steps
- Flexible as Configuration can be Changed after Deployment with Little or no Code Changes
- Flexible (IIS not required on server) but more Complex than Web Services
.Net Remoting Vs. DCOM

- Not as Transparent as DCOM
- Has no COM Interop Involved
- Offers More Deployment Options
- Not Tied to a Specific Wire Protocol or Format
- Firewall Friendly
- Allows you to Specify which Objects are Remoted

.Net CLR and COM+

- CLR Introduces a Newer and Easier Programming Model for COM+
- The COM+ Services are still Available only at the OS/Unmanaged World
- Transitions Between Managed/Unmanaged Layers are Taken Care of by “System.EnterpriseServices” Namespace
  - System.EnterpriseServices.ContextUtil class provides access to contexts (as did CoGetObjectContext)
- Services are Made Available to Components Using Attributes
.NET, Win32, COM, and COM+

- .Net applications can call functions that reside in Win32 DLLs via the DllImport attribute of System.Runtime.InteropServices’ platform invocation (PInvoke) services
- .Net applications can call traditional COM components using Runtime Callable Wrappers (RCWs) which uses interop assemblies (i.e., .Net components) created via COM Interop using the TLBIMP utility
- COM+ services (transactions, object pooling, JIT activation) can be used from .Net (via System.EnterpriseServices namespace)
- .Net runtime uses COM+ to support some of its services (e.g., transactions for managed classes)

ContextUtil Class

- Useful Static Members
  - ActivityId
  - IsSecurityEnabled
  - TransactionId
  - DesactivateOnReturn
  - MyTransactionVote

- Rich Set of Member Functions
  - SetComplete
  - SetAbort
  - EnableCommit
  - DisableCommit
  - IsCallerInRole
  - GetNamedProperty
**.Net and ServicedComponent**

- COM+ Class Derives from “ServicedComponent”
  
  ```csharp
  [ComVisible(true)]
  [ObjectPooling(MinPoolSize=2, MaxPoolSize=5)]
  [Guid("57F01F20-9C0C-4e63-9588-720D5D537E66")]
  [Transaction(TransactionOption.Required)]
  public class SVCCompClass : ServicedComponent
  ```

- In COM+ 1.0, the List of Services Are not Changed
- In COM+ 1.5 (Microsoft Windows XP Professional and .Net server), Newer Services are Available to Leverage CLR Features

---

**Agenda**

1. Microsoft’s Pre .Net Component Model
2. Towards Sun’s JEE Component Model
3. eXtensible Server Pages (XSP) Environment
4. Conclusion
JEE Platform: The Whole is Greater than the Sum of its Parts

JEE: A Complete Computing Environment

- Platform Specification
  - Lists required elements of the platform
  - Lists policies to follow for a valid implementation
- Reference Implementation
  - Semantically correct prototype to test against
- Compatibility Test Suite
  - API-level compatibility, component-level tests, end-to-end compatibility
- Application Programming Model:
JEE Features

- Standard Documents and Software Libraries
  - Specification
  - Programming Model
  - Platform
  - Reference Implementation
  - Compatibility Test Suite
  - http://java.sun.com/javaee/

The Three Cs: Components, Containers, Connectors
JEE: Components

- Enterprise JavaBeans
  - Server-side solutions can be built without regards for the database, transaction server, or application they run on
- Servlets
  - Run on vast majority of web servers
- JavaServer Pages
  - Dynamic content leverages off the full power of Java

JEE: Containers

- Containers provide high-performance, scalable environments for JEE-enabled servers
- JEE-enabled servers support EJB-based components, servlets, and JSP-based pages
JEE: Connectors

- Connectors allow JEE-based solution to preserve, protect, and leverage off of existing enterprise investments

JEE: Unifying the Three Cs

- Single platform
- Standard platform-independent technology
- Applications built with components can be run on any JEE server, and are able to talk to enterprise-class systems that exist today
**JEE Server and Containers**

- **Browser**
- **Application Client**
  - Application Client Container
  - **Client Machine**

**J2EE Server**
- **Servlet**
- **JSP Page**
- **Web Container**
- **Enterprise Bean**
- **Enterprise Bean**
- **EJB Container**

**Database**

---

**Distributed Multi-tiered Applications**

**J2EE Application 1**
- **Application Client**
- **Dynamic HTML Pages**
- **Enterprise Beans**
- **Database**

**J2EE Application 2**
- **Dynamic HTML Pages**
- **JSP Pages**
- **Enterprise Beans**
- **Database**

**Client Tier**

**Web Tier**

**Business Tier**

**EIS Tier**

**Client Machine**

**J2EE Server Machine**

**Database Server Machine**
Servlets Technology

- Java’s standard mechanism for writing code that runs and extends the functionality of a servlet engine
- A servlet is to a server what an applet is to a browser
- HTTP servlets
  - Replacement for CGI
  - Standard mechanisms for handling cookies, sessions, session/application contexts
  - Advantages over CGI: performance, platform and web-server independance
- Servlet filters are new in the Java Servlet Specification 2.3 (2.5 is latest spec)

JSP Technology
(http://java.sun.com/products/jsp/)

- Answer to Microsoft’s Active Server Pages
- Provide mechanism for including tags/scriptlets into an HTML or XML page
- JSPs have .jsp extension and are processed using a special servlet
  - JSP page is compiled into a servlet upon first access or after each modification
- Ability to instantiate and access JavaBeans within JSP pages
Introduction to Servlets and JSPs

- See [http://www.java-shop.com/jsp.htm](http://www.java-shop.com/jsp.htm)
  - Java’s standard mechanism for writing code that runs and extends the functionality of a servlet engine
  - A servlet is to a server what an applet is to a browser
  - HTTP servlets
    - Replacement for CGI
    - Standard mechanisms for handling cookies, sessions, session/application contexts
    - Advantages over CGI: performance, platform and web-server independance
  - Servlet filters are new in the Java Servlet Specification 2.3
  - Answer to Microsoft’s Active Server Pages
  - Provide mechanism for including tags/scriptlets into an HTML or XML page
  - JSPs have .jsp extension and are processed using a special servlet
    - JSP page is compiled into a servlet upon first access or after each modification
    - Ability to instantiate and access JavaBeans within JSP pages

Servlets and JSPs Examples

- JSP displaying a banner image based on who is referring the user to the site:

```jsp
<%@ page import="com.ibm.jspredbook.*;"
errorPage="error.jsp" %>
<body bgcolor="#FFFFFF">
<!--the referer header is used to trap the url the user is coming from -->
<IMG SRC="/servlets/ImgServlet?from=<%=request.getHeader("Referer")%>">
</body>
</html>
```
Servlet referenced in the IMG tag of the previous slide (partial):
package com.ibm.projsp;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
public class ImageServlet extends HttpServlet {
    private String docHome = ".";
    public void service( HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        HttpSession session = request.getSession(true);
        ServletConfig config = getServletConfig();
        ServletContext application = config.getServletContext();
        File file = findFile(request, response);
        if (file == null) { return; } else {
            response.setContentType(application.getMimeType(file.getName()));
            response.setContentLength((int) file.length());
            sendFile(file, response);
        }
    }
}

Catalog of Patterns
(www.martinfowler.com/eaaCatalog)

- Domain Logic Patterns (e.g., Service Layer)
- Data Source Architectural Patterns
  - e.g., Data Mapper
- Object-Relational Behavioral Patterns
  - e.g., Unit of Work
- Object-Relational Structural Patterns
  - e.g., Inheritance Mappers
- Object-Relational Metadata Mapping Patterns
  - e.g., Repository
- Web Presentation Patterns
  - e.g., MVC, Application Controller
- Distribution Patterns (e.g., Remote Façade, DTO)
- Offline Concurrency Patterns (e.g., Implicit lock)
- Session State Patterns (e.g., Server Session State)
- Base Patterns (e.g., record set, value object)
MVC Pattern
Splits user interface interaction into three different roles

- MVC architecture decouples the code to handle user actions (controller), the data and business logic (Model), and the presentation (View)

MVC Review
**MVC or Model 2 Design Pattern**

- Used to implement Modern Web Applications as a combination of:
  - Servlets/Servlet filters
    - Controller receiving/filtering requests from the user
    - Updates the application’s model composed of JavaBeans
    - Passes the page request to a view JSP
  - Java Server Pages
    - Display information based on the current state of the application’s model
  - JavaBeans
    - Enable component reuse
  - Custom Tag Libraries
    - Make it possible to move source code out of the JSP where it is difficult to maintain and into reusable JavaBeans
  - Rich array of Java APIs
- See [http://www.mhsoftware.com/resources/iisjserv.html](http://www.mhsoftware.com/resources/iisjserv.html) for a comparison of IIS/ASP and Servlet/JSP technology

---

**Implementing the “V” of “MVC” Using JSPs**

- When the view is implemented as a JSP, the controller object (e.g., servlet) forwards processing of the request and the response to a JSP view
- Controller adds a reference to the model object to the user’s session or request object
- JSP gets a handle on the model object and constructs the HTML or other markup to be returned to the client
Implementing the “V” of “MVC” Using JSPs
(continued)

Architectural Considerations

- Page-Centric v.s. Dispatcher Type
  - Page-Centric architectures have a JSP handling the request directly
  - Dispatcher architectures include a Servlet that handles the request and delegates to a JSP
  - Sample architectural patterns:
    - Page-View (Page-Centric)
    - Page-View with Bean (Page-Centric)
    - Mediator-View (Dispatcher)
    - Mediator-Composite View (Dispatcher)
    - Service-to-Workers (Dispatcher)
Servlet / JSP Engines

- Apache + TomCat
  - [http://www.apache.org](http://www.apache.org)
- Apache + Jrun
- Apache + Jetty
- Mainstream JEE Applications Servers Come Bundled with an HTTPD service and Servlet/JSP engines

Application Controller Pattern
Data Transfer Object Pattern
object that carries data between processes to reduce the number of calls

Remote Facade Pattern
provide coarse-grained object façade to optimize network usage
Enterprise JavaBeans (EJBs)

- Enterprise Application Platforms:
  - Provide model for the development and deployment of reusable Java server components
  - Support component modeling
  - Handle the serving of EJBs
- Next Steps:
  - “Model-Centric” Application Development

Enterprise JavaBeans Container

The EJB Object interface intercepts all method calls and implements transactions, transaction management, persistence, and security services for the bean based on deployment descriptor services.

The EJB Home interface is accessible through RMI and implements all lifecycle services for the bean.
Enterprise JavaBeans and Services

Application Server
Container

Services
- Lifecycle
- Transaction
- Security
- Load Balancing
- Error Handling
- Threading
- Persistence*

* In the EJB 1.0 specification support for persistence services is optional. In the EJB 1.1 specification it is mandatory.

Sample DemoBean Application Architecture

DemoBean Architecture

The home and remote interface are generated at deployment time.
### Entity Beans in EJB Application Servers

- 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
Session and Entity Beans

- Transfer $100 from Account A to Account B
- Subtract $100
- Add $100

EJB Physical Partitioning

A Domain Name System (DNS) server routes incoming browser requests evenly across a pool of web servers. This technique is referred to as DNS round-robining. The application server provides fail-over if one of the web servers goes down.

The application server distributes load across all available EJB servers and provides fail-over if one of the EJB servers goes down.

EJBs communicate to the database through Java Database Connectivity (JDBC). The application server pools and manages database connections for maximum efficiency.
Creating an EJB Component

- Example:

```java
try {
    // get the JNDI naming context
    Context initialCtx = new InitialContext();

    // use the context to lookup the home interface
    CheckingHome home = (CheckingHome) initialCtx.lookup("checking");

    // use the home interface to create the enterprise Bean
    Checking server = home.create();

    // invoke business methods on the bean
    server.createAccount(1234, "Athul", 1000671.54d);
}
```

```java
catch (Exception ex) {
    ex.printStackTrace();
}
```
JEE OMA Services

- Activation Services
  - RMI Activation Framework
    - [http://docstore.mik.ua/orely/java-ent/jenut/ch14_01.htm](http://docstore.mik.ua/orely/java-ent/jenut/ch14_01.htm)
  - 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

---

Agenda

1. Microsoft’s Pre.Net Component Model
2. Towards Sun’s JEE Component Model
3. eXtensible Server Pages (XSP) Environment
4. Conclusion
Apache Cocoon

- XML based publishing Framework
- An Apache Software Foundation open source project
- Written in Java, runs mostly as a servlet
- Started as a simple servlet based XSL styling engine for http://www.apache.org site
- Current version is in the second generation of evolution
- Designed for scalability (uses SAX processing) -- can process huge documents using small amount of memory

Apache Cocoon (continued)

- Cocoon promotes the separation of Content, Logic, Presentation and Management in web-site design.
**Cocoon 2**

- Web Publishing framework implemented as a servlet
- Requires a servlet engine to operate
- Cocoon 2 has been re-architected to truly support the MVC pattern

**Cocoon processor:**
- Cocoon Java type that takes a DOM tree as an input and produces another

**Cocoon producer:**
- Cocoon Java type used to feed the initial XML content to the Cocoon processing pipeline
- e.g., Cocoon serves static XML documents using its built-in FileProducer

**Cocoon processing instructions** act upon a whole document, which generates a result document

```xml
<?cocoon-process type="xsp"/>
```

Result document is passed to the next Cocoon processor
- Similar to servlet chaining
- Alternatives: Rocket, and CPan’s

---

**Cocoon Servlet Directories & Files**

- Directory for auto mounting sub-sitemaps
- Cocoon Configuration file
- A sub-sitemap directory
- Main sitemap file
- Directory for log files
**Sitemap Goal**
- Used to de-couple the exposed URI space from the actual location of resources
- Allows easily changeable specification of processing steps

**Sitemap Contents**
- Component declarations
  - generators, transformers, serializers, ...
- Resource declarations
  - named collection of pipeline components
- Pipeline declarations
  - sequential arrangement of components for processing

---

**A sitemap is an XML file**

**Sitemaps are hierarchical --** A sitemap can point, explicitly or implicitly, to sub-sitemaps

**A sitemap is translated into a java program and is compiled into bytecode**

**Changes to sitemaps can be loaded dynamically and asynchronously**
Cocoon Request Processing

- Request is dispatched to matching pipeline
- Basic pipeline operation
  - The generator generates XML content
  - Zero or more transformers transform the content
  - The serializer writes it to the output stream
- Different Kinds of generators
  - File, Directory, XSP, JSP, Stream, …
- Different Kinds of transformers
  - XSLT, I18N, Log, …
- Different Kind of Serializers
  - HTML, XML, Text, PDF, SVG, …

Cocoon Processing Mechanisms
http://cocoon.apache.org/2.1/userdocs/concepts/index.html

- Dispatching based on Matchers.
- Generation of XML documents (from content, logic, relational DBMS, objects or any combination) through Generators
- Transformation (to another XML, objects or any combination) of XML documents through Transformers
- Aggregation of XML documents through Aggregators
- Rendering XML through Serializers
Sample Cocoon Pipeline

```xml
<map:pipeline>
  <map:match pattern="hello.html">
    <map:generate src="docs/samples/hello-page.xml"/>
    <map:transform src="stylesheets/page/simple-page2html.xsl"/>
    <map:serialize type="html"/>
  </map:match>
  <map:match pattern="images/**.png">
    <map:read src="resources/images/{1}.png" mime-type="image/png"/>
  </map:match>
  <map:handle-errors>
    <map:transform src="context://stylesheets/system/error2html.xsl"/>
    <map:serialize status-code="500"/>
  </map:handle-errors>
</map:pipeline>
```

Dynamic Content Generation from XSP

```xml
<!-- A simple XSP Page -->
<xsp:page language="java" xmlns:xsp="http://apache.org/xsp">
  <page>
    <title>A Simple XSP Page</title>
    <content>
      <para>dynamically generated list:</para>
      <ul>
        <xsp:logic>
          for (int i=0; i &lt; 3; i++) {
            <li>Item</li>
          </xsp:logic>
        </ul>
    </content>
    </page>
</xsp:page>
```
Introduction to XSPs

- See:
  - Apache Cocoon technology: http://xml.apache.org/cocoon/
  - XSP / JSP differences: http://www.onjava.com/lpt/a/620
  - Publishing Frameworks: http://www.oreilly.com/catalog/javxml/chapter/ch09.html#69379

- XSP:
  - Core technology available in Apache Cocoon 2
  - Approach separates content, style, and logic as XML files and uses XSL to merge them

- XSP engine
  - Implemented as a Cocoon processor that accepts an XSP as input
  - Translates XSP into equivalent source program, compiles, loads and executes it
  - XSP generates producers while JSP technology generates servlets
  - All XSP producers are derived from an abstract base class XSPPage

Minimal XSP Page

- XML document that has the following characteristics:
  - Processing instruction invoking the XSP processor:
    - <?cocoon-process type="xsp"/>
  - Document root element must be:
    - <xsp: page>
  - All language and Taglib declarations must appear as attributes in the root element tag:
    - e.g., <xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core"/>
  - Optional elements:
    - <xsp:logic> (procedural logic embedding) and <xsp:expr> (program expression inlining)
  - Optional processing of the resulting page via a style sheet for viewing purpose
    - <?cocoon-process type="xslt"?>
      - <?xml-stylesheet href="sample.xsl" type="text/xsl"?>
  - Note: Minimal JSP page is an HTML document
**XSP Example**

- **XSP logic tag:**

  ```
  <p>
  Good
  <xsp:logic>
    String timeOfDay = {
      new SimpleDateFormat("aa")
    ).format(new Date());
    if (timeOfDay.equals("AM")) {
      <xsp:content>Morning</xsp:content>
    } else {
      <xsp:content>Afternoon</xsp:content>
    }
  </xsp:logic>!
  </p>
  ```

- May be rephrased using a library tag as:

  ```
  <p>Good <util:time-of-day/>!</p>
  ```

---

**XSP v.s. JSP**

- **XSP**
  - Tailored to maximize code reuse
  - Allows separation of content from presentation
    - Developers handle content generation (content can static or generated via servlets or Java code)
    - XML/XSL authors handle style/presentation via style sheet modifications
  - As XSP processing occurs prior to styling, the content can be presented in various ways
  - Keep development teams well isolated
  - Can use IBM's Bean Scripting Framework (BSF) to support other scripting languages in addition to Java

- **JSP**
  - Popular and widely understood
  - Requires tight collaboration between application developers and presentation designers
  - At best presentation designers must understand how to use tag libraries
Using Cocoon as a Presentation Framework for Web-Services
www.pankaj-k.net/sdwest2002/Cocoon-FWS.ppt

Client Tier

Web Tier

Servlet Container

Cocoon

WS Registry

Web Services

Enterprise Apps

Cocoon as a Processing Framework for Web Services
(continued)
Sample Implementation

- Use Google Web Services API from http://www.google.com/apis to create a simple web application that accepts a search string from user and displays the list of result entries.
- Sample is based on the XSP contributed by Ugo Cei in Cocoon-dev mailing list
- Application components:
  - sitemap.xmap -- Sitemap for this application
  - index.html -- HTML file to accept search string
  - google.xsp -- XSP file that makes the SOAP call to Google Web Service using SOAP logicsheet.
  - search-results.xsl -- XSL stylesheet to transform SOAP response from Google Web Service to HTML page
- Downloadable from:
  - http://www.pankaj-k.net/sdwest2002/google.zip

Application Architecture

Browser

http://<host>/cocoon/mount/google/

index.html

…/search?q=...

executes google.xsp

Search results in HTML

Cocoon Servlet

Maps request to index.html

SOAP request

SOAP response

Google WS

Applies search-results.xsl
Agenda

1. Legacy Page-Based Application Server Technology
2. Distributed Object Computing Concepts
3. Object Management Architecture
4. Conclusion

Summary – Key Application Server Objectives

- Microsoft IIS - COM+/Net - ASP is a Page-Based Script-oriented application server
- COM+/Net and ASP are “equivalent” to JEE EJB and JSP
- Servlets are more efficient than traditional CGI approaches, and are not subject to the issues that arise from in-process approaches (ISAPI, NSAPI, fast-CGI, etc.)
- JSPs allow custom tags and Java scriptlets within HTML pages
- JSPs are a first step towards separation of content/presentation. True separation using that component model requires discipline
- Cocoon2 is a web publishing framework implemented as a servlet
- XSP is a core technology available in Cocoon 2 that allows true separation of content/style/logic
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).

Assignments & Readings

- Readings
  - Handouts posted on the course Web site
  - Explore the Microsoft IIS - COM+/.Net - ASP, TomCat, JRun, and Cocoon 2/XSP Environments
  - Read white papers/documentation at:
    - http://xml.apache.org/cocoon/
    - Also read Microsoft IIS, TomCat, JRun, and Cocoon 2 related whitepapers on the vendor sites
  - Modeling and Analysis of Software Architectures
    - XADL 2.0: http://www.isr.uci.edu/projects/xarchuci/
Assignment

- Assignment #2 (Ongoing):
  - #2a: Investigate distributed object computing platforms’ development environments for the technologies covered in this session. 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
  - #2b: See homework #2 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
Next Session: JEE and .Net OMA Implementations (Part 2)