Application Servers
G22.3033-003

Session 12 - Main Theme
Web Services and Portal Technology

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- Web Services Definitions and Standards
- From Portals to Remote Portlets Web Services (RPWS)
- Web Services Security
- Web Services and Business Processes/Workflow
- Summary
- Readings
- Final Project
  - Due by 12/23 at midnight
  - No extensions
Summary of Previous Session

- Continued from Session 11:
  - Microsoft .Net & Web Services
  - Model Driven Architectures
  - Component Development Summary
  - Reflective Application Servers
  - Intelligent Agents
  - Multimedia/Broadband Application Servers
  - Other Component Models Directions
- Summary
- Readings

Part I

Web Services Definitions and Standards

http://www.w3c.org
http://www.ws-i.org
http://www.oasis.org
Review

- Web
  - Application to user communication across the internet.

- Web Services
  - Application to application communication across the internet.
  - Self-contained and modular applications.
  - Can be described, published, located, and invoked over a network.
  - Applications can be built by discovering and orchestrating network-available services.

- A Web Service is described using a formal XML notation called service description.

W3C Definition of Web Services

- W3C Web Services Architecture WG definition:

  "A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages via internet-based protocols."

"Our definition of the term ‘Web Services’ does not presuppose the use of SOAP as a packaging format or a processing model. Nor does it presuppose the use of WSDL as a service description language...."
WS-I Definition of Web Services

• Web Services-Interoperability (www.ws-i.org) defines “Basic Web Services” as applications built with:
  – SOAP
  – WSDL
  – UDDI
  – XML Schema

What this All Means

• Before Web Services, Web-based e-Business was:
  – Browsing of linked documents
  – Manually initiated transactions & purchases
  – Downloading files
  – All initiated manually via a Web browser
• Web Services is a model for using the Web to:
  – Automatically initiate processes via the Web using self contained software applications
  – Method for describing, publishing, promoting, registering, & initiating processes dynamically in a distributed environment
  – New ways of using the web, including intelligent agents, marketplaces & auctions
  – And… Not necessarily using a Web browser!
• Actually, the Web is not required
  – How about ”Net Services?”
• All done using XML standards (i.e., SOAP, WSDL, UDDI)
Web Services Enablers

- Reliable & Transparent Interconnectivity
  - Web Protocols
- Structured Information
  - XML Schemas & validation
- Application Interface Standards
  - UDDI, WSDL, SOAP
- Consistent Definitions
  - Profiles, Test Suites & Scenarios
- Business Process Interface Standards
  - ebXML, BTP, WSBPEL, etc.
- Security / Infrastructure Standards
  - SAML, XACML, WS-Security, etc.

Terminology

- Service Description
  - Message Format
  - Transport Protocol
  - Location
  - Encapsulates implementation details

- Key Standards Used
  - Hyper Text Transport Protocol (HTTP)
  - Extensible Markup Language (XML)
  - Simple Object Access Protocol (SOAP)
  - Web Services Description Language (WSDL)
  - Universal Description, Discovery and Integration (UDDI)
Web Services – SOAP

WWW c1993
global, hyperlinked document store

<table>
<thead>
<tr>
<th>Operation Protocol</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST (a new document)</td>
<td>GET (a document)</td>
</tr>
<tr>
<td>PUT (an updated document)</td>
<td>DELETE (a document)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Naming</th>
<th>URIs</th>
</tr>
</thead>
</table>

| Data | HTML document | PNG image |
### WWW c2001

*global, hyperlinked, *machine-readable document store*

<table>
<thead>
<tr>
<th>Operation Protocol</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming</td>
<td>URIs</td>
</tr>
<tr>
<td>Data</td>
<td>HTML</td>
</tr>
<tr>
<td></td>
<td>XML</td>
</tr>
<tr>
<td></td>
<td>XHTML</td>
</tr>
<tr>
<td>namespaces</td>
<td></td>
</tr>
</tbody>
</table>

### WWW c2005

*global, hyperlinked, machine-readable data store with a computational layer*

<table>
<thead>
<tr>
<th>Web Services</th>
<th>Resource-Oriented (REST)</th>
<th>Distributed Objects (SOAP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation Protocol</td>
<td>HTTP</td>
<td></td>
</tr>
<tr>
<td>Naming</td>
<td>URIs, for concepts, not just documents</td>
<td></td>
</tr>
<tr>
<td>&quot;Don’t say &quot;colour&quot; say <a href="http://example.com/2002/std6%col">http://example.com/2002/std6%col</a> &quot;&quot; - TBL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data with Semantics</td>
<td>XML</td>
<td>RDF</td>
</tr>
<tr>
<td></td>
<td>XML Schema</td>
<td>RDF Schema</td>
</tr>
</tbody>
</table>
Web Service Model

- The Web Services architecture describes three roles
  - Service provider
  - Service requester
  - Service registry
- It describes three basic operations
  - Publish
  - Find
  - Bind.
- It also has two artifacts
  - Service
  - Service description
Web Services Model

- The web services development lifecycle
  - Build
  - Deploy
  - Run
  - Manage

- To perform three operations (publish, find, and bind) in an interoperable manner, there must be a Web services stack that embraces standards at each level.

- Model Based on Web Services Stack

Architecture Overview

(Conceptual Web Services Stack)
## Architecture Overview

*(Transport and Messaging Layer)*

- Requirements for each node:
  - Ability to build SOAP message.
  - Ability to parse SOAP message.
  - Ability to communicate over a Network.

![Architecture Overview](image)

## Architecture Overview

*(Web Services Description Stack)*

- How does the Service Requester know what format the requester message should use?
- The Web Services architecture uses WSDL for base level service description.
Architecture Overview
(UDDI Data Structures)

- UDDI provides a mechanism for holding descriptions of Web Services. There are four fundamental data structures in a UDDI entry.

Architecture Overview
(Service Publication and Discovery)

- Service Publication
  - Producing Service Description
  - Publishing Service Description
    - Direct Publish
    - Internal Enterprise UDDI node
    - Portal UDDI node
    - Partner Catalog UDDI node
    - E-marketplace UDDI node

- Service Discovery
  - Acquiring Service Descriptions
    - Depends on Service publication
  - Consuming Service Descriptions
Web Service for Real e-business

- **Security**
  - Web Services require an end-to-end security.
    - end to end authentication
    - end to end authorization
    - end to end integrity
    - end to end confidentiality

- **Quality Of Services**
  - Reliable messaging is a key feature.
    - best-effort, at-least-once, at-most-once, exactly-once

- **Manageability**
  - Management of the Web Service infrastructures.
  - Management of Web Services between Enterprises.

- **Context Awareness**
  - “Intelligent” Web services refer to being aware of the context of the user/system/provider.
    - Device Type, User Profiles, Geographic Information

---

Web Service for Real e-business

- **Conversation and Activities**
  - An *Activity service* which allows one to define the operational context of one or a series of requests, controlling duration and participants.
  - A *Conversation Service* which provides interaction styles for Web Services.

- **Intermediaries**
  - Web Services has intrinsic support for intermediaries.
  - Intermediaries open the door for value added services.
Web Services Standardization

OASIS Web Services TCs

- Asynchronous Service Access Protocol
- Framework for Web Services Implementation
- Web Services Business Process Execution Language
- Web Services Composite Application Framework
- Web Services Distributed Management
- Web Services for Remote Portlets
- Web Services Interactive Applications (completed)
- Web Services Reliable Messaging
- Web Services Security
OASIS Security TCs

- Application Vulnerability Description Language
- Digital Signature Services
- eXtensible Access Control Markup Language
- Provisioning Services
- Public Key Infrastructure
- Rights Language
- Security Services
- Web Application Security
- Web Services Security
- XML Common Biometric Format

Conclusion

- Web Services stack illustrates:
  - A well defined layer architecture.
  - Rapid Integration of applications.
  - Enterprise dynamic software Integration.
- Web Service Stack must be supported by the following standards efforts:
  - Reliable messaging support.
  - End-to-end security model.
  - Web Service Management.
Discussion

- The vertical tower of Web Services Stack is still evolving, which tower do you think is most important to be addressed?
- If we solve all the issues related to Web Service standardization, will Web Services become THE distributed computing platform for dynamic software integration?
- Is it an evolution/or revolution in the context of Web?
- What could be the most disadvantage of using Web Services?

Part II

*From Portals to Portlets*

[http://www.devx.com/ibmwebservices/Article/11781](http://www.devx.com/ibmwebservices/Article/11781)
Portals

A “Portal" describes a Web Application that is designed to present a ton of information in a concise and centered way, thereby making the Internet easier to use. Popular examples would be Yahoo.com or Netscape.com, etc.

Portal History

• A number of portal groups have developed computational web portals
  – Basic capabilities like job submission, file management, monitoring,…
  – Some code specific input forms
  – See www.computingportals.org
• All of these basically follow a three-tiered architecture.
Portal Problems

- Different portal groups have no well established way of sharing services
  - Different technologies (Java, Perl, CORBA, EJB) in middle tier.
  - Different protocols (GRAM, IIOP, …)
- Consequently, lots of redundancy, reinvention
  - Example: batch script generation

How To Fix

- There are essentially two “interfaces” in the three tiered model.
  - Middle tier to backend service proxy
  - User interface to middle tier
- Web services address the service sharing problem.
- Jetspeed (or similar) addresses the user interface problem.
Web Services Proposition

• We have to agree to common service definitions, interfaces, and wire protocols
  – What is job submission? What are the methods? How do I find your job submitter? How do I talk to it?

• Web services already have solutions
  – WSDL, UDDI or WSIL, SOAP

• The WSDL file tells me everything I need to write a client for your SOAP service.

More Web Services

• Atomic web services include:
  – Site specific services (submitting and monitoring jobs on a site for example)
  – General purpose services (batch script generation, input forms for a particular code).
  – Connection services (getting the batch script to the job submitter).

• We construct higher level services out of atomic pieces of these types.
Half the Battle

- So now I have reusable services. If someone writes a great batch script generator, the developer can just use it in his/her portal.
- But it leads to a proliferation of user interfaces from every developer.
- How do I manage the growing number of interfaces? How can I reuse a great interface that someone else has written?
Portal Component Model

- The solution is to use “portlets”
  - Single purpose UI components.
  - Complicated portals can be built out of a collection of portlets.
  - Portal becomes a container of customizable portlets.
- Portal developers publish a list of available portlets.
- Users can create and customize their own portals by including only the portlets that interest them.
- JetSpeed is a free, open source implementation of these ideas.

Portal and Portlets

- Portal are commonly used to provide people with access to information and applications in a condensed form.
- Two integration points for Web Services and Portlets
  - Portlets that use Web Services as a backend.
  - Portlets that are described, wrapped and published as Web Services.
Web Services as a Portlet

- Each Web Service naturally has a user interface specified as “just another port”
  - Customizable for universal access
- This gives each Web Service a Portlet view specified (in XML as always) by WSRP (Web services for Remote Portals)
- So component model for resources “automatically” gives a component model for user interfaces
  - When you build your application, you define portlet at same time

Web Services have other ports (Grid Service) to be OGSI compliant

Jetspeed Architecture

- HTML
  - Turbine Servlet
    - JSP template
  - Screen Manager
  - PortletController
    - PortletControl
      - Portlets
        - XML
          - RSS, OCS, or other Local or remote
        - HTML
          - Local files
      - ECS Root to HTML
      - ECS
        - ECS
          - Portlet
            - JSP or VM
              - Local templates
            - WebPage
              - Remote HTML
      - Portlets
        - User implemented using Portal API
Portlets and Portal Stacks

- User interfaces to Portal services (Code Submission, Job Monitoring, File Management for Host X) are all managed as portlets.
- Users, administrators can customize their portal interfaces to just precisely the services they want.
Choose Portlet Layout

Choose 1-column Layout

Original 2-column Layout

Architecture of Collaborative Interactive Application as a WebService (SVG)
JetSpeed

- Defines several types of portlets, including
  - WebPagePortlet: portlet window contains a remote web page from another server.
  - JSP Portlet: loads a local portlet into a window.
- Portlets are arranged in the container page with HTML tables.
- Portal administrators:
  - Publish a list of available portlets in an XML registry
  - Create custom skins
- Users:
  - Select desired portlets from registry
  - Arrange layouts, customize appearance
  - PSML files preserve customization
Sample Portlet Registry

```xml
<?xml version="1.0" encoding="UTF-8"?>
<registry>
  <portlet-entry name="GridsFileBrowser" hidden="false" type="ref"
    parent="WebPagePortlet" application="false">
    <meta-info>
      <title>GridsFileBrowser</title>
      <description>File Browser for Grids</description>
    </meta-info>
    <parameter name="dont_remove_applet" value="yes"
      hidden="false"/>
    <parameter name="dont_remove_script" value="yes"
      hidden="false"/>
    <url>http://some.server/FileBrowser.jsp</url>
  </portlet-entry>
</registry>
```

Portlet Layout

Turbine Screen

Portlet Controller

Portlet Control

Portlet

(from localhost)

Portlet Control

Portlet

(from http://some.where.else/)
WSRP Motivation

Enable the sharing of portlets (markup fragments) over the internet with a common interface

⇒ Cross vendor publishing and consuming of content

Technical Motivation for WSRP

• API users need to compose a presentation and workflow from the underlying APIs, resulting in high setup costs for the API user.
• API users need to track API changes resulting in high maintenance costs for the API user.
• The application provider’s quality standards are not reflected in the resulting application.
• Application provider loses opportunity to “brand” the user experience.
Presentation vs. data oriented WS

Presentation Service

Data service

Presentation WS visual & user facing & interactive

Actors in a WSRP World

• **Portlet:** A particular configuration of a component that renders its state in markup.

• **Producer:** The service provider hosting the remote interactive Web service (for example, weather.com as a weather service provider)

• **Consumer:** Integrates the remote service into its Web application, oftentimes using a portal toolkit (for example, Yahoo Weather or a corporate portal)
General Interaction Flow

The interaction only modifies *navigational state*
General Interaction Flow (2)

The interaction modifies *any type of state*

WSRP and the WS-Stack

<table>
<thead>
<tr>
<th>WSRP</th>
<th>UDDI</th>
<th>WS-Security</th>
<th>WS-Policy</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WSDL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SOAP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XML</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCP/IP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WSRP RoadMap

Steps in finalizing v1 and assuring real interoperability

Define Conformance Tests
Finalize v1 as TC Spec
Start on v2: Cross-Portlet Coordination + ...
Final Review Period
OASIS Standard

JetSpeed Problems

• Problems with HTML forms
  – The response page of an html form in a WebPagePortlet will be displayed in the entire browser window, not the portlet.
• Also does not preserve new page in portlet after following links.
• Netscape 4 has limited capability to handle tables.
• Page refreshes not preserved in portlets.
• DoD PET’s Online Knowledge Center (OKC) is addressing these.
Portal To Do List

• Incorporate OKC bug fixes and extensions
• Solve HTML forms problem
• Solve Netscape 4 problem
• Customize security (authentication and authorization) to support GSI, Kerberos
• Need a shared security model between the UI services and the remote services.

Part III

Web Services Security
What is different?

• Model Designed to be intercepted
  – How to Secure every “hop”
  – Only as strong as the weakest link
• Loosely Coupled
  – How to Establish Trust
• Security Policies
  – Comprehensive control
• Interoperability
  – All Products / All Vendors ???

Same Basic Requirements

• Identity
  – Who sent the message?
• Integrity
  – Was the message tampered with while in route to the recipient?
• Non-Repudiation
  – Can the sender claim they never sent the message?
• Confidentiality
  – Can only the intended recipient read the message?
• Interoperability
• Audit and Traceability
Securing Web Services

Security Context 1
(Machine to Machine Communications)

Security Context 2
(Machine to Machine Communications)

Security Context 3
(Business Transaction)

• Identity
• Integrity
• Non-Repudiation
• Confidentiality
• Interoperability
• Audit and Traceability

Web Services

Security Requirements for Web Services

• Authentication: Participants in a message exchange recognize each other and the creators of the content
• Authorization: Actions on resources are checked against permissions
• Auditing: Participants have a record of what happened
• Integrity: Message content wasn’t altered inappropriately during transit
Security Requirements for Web Services (cont.)

- Confidentiality: Content is not visible to non-authorized parties
- Non-repudiation: A message sender can’t refute the action
- Trust: Participants have to agree to work together

Web Services Security Today

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Kerbos &amp; X.509 Certificates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WS-E Security Context Token</td>
</tr>
<tr>
<td></td>
<td>SAML for federated identity and integration with Single Sign-On</td>
</tr>
<tr>
<td></td>
<td>XKMS for SOAP based key management</td>
</tr>
<tr>
<td>Non-Repudiation/Integrity</td>
<td>XML Digital Signatures</td>
</tr>
<tr>
<td>Authorization</td>
<td>WS-Trust, WS-Security Policy, WS-PolicyAttachment</td>
</tr>
<tr>
<td></td>
<td>XACML for XML Security Policies</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>IPSEC, HTTP(s) &amp; SOAP(s) using SSL</td>
</tr>
<tr>
<td></td>
<td>XML Encryption</td>
</tr>
<tr>
<td>Interoperability</td>
<td>WS-I</td>
</tr>
<tr>
<td>Audit / Tracibility</td>
<td>WS-Context</td>
</tr>
</tbody>
</table>
Part IV

*Web Services and Business Processes/Workflow*

**Business Processes and Workflows**

- Business Processes are graph of activates that carry out some meaningful business operations.
- Web Services are attractive components of Workflows.
- Workflow is a primary mechanism to compose Web Services.
- Workflows are business processes that are run in an IT environment using tools such as IBM MQSeries Workflow.
### Case Study: A Simple Web Services Workflow

- **A Buyer Service** is ordering goods from seller service.
- **A Seller Service** is a Web Service whose interface is defined using WSDL.
- The **Buyer Service** is invoking the order method on the seller service using SOAP and the WSDL definition for the seller service.
- The **Buyer Service** knows what to expect in the SOAP reply message because this is defined in the WSDL definition for the Seller Service.

![Diagram of a simple web services workflow](image)

### Enabling Web Services

- **Credit Validation Service**

![Diagram of enabling web services](image)
Business Processes and Workflows

- Private and Public Workflow

![Diagram showing private and public workflow with services and steps]

Business Processes and Workflows

- Workflow Granularity

![Diagram showing workflow granularity with services and steps]
Orchestration vs. Choreography

• Orchestration – describes how web services can interact with each other at the message level, including the business logic and execution order of the interactions
• Choreography – tracks the sequence of messages that may involve multiple parties and multiple sources, including customers, suppliers and partners

BPEL4WS (Business Process Execution Language for Web Services)

• A layer on top of WSDL
• Models the behaviour of web services in a business process interaction
• Control logic required to coordinate web services
• In current implementations interpreted and executed by an orchestration engine (centralized!)
• Support for long transactions (compensations)
BPEL4WS

• Executable process – models the behaviour of participants in a specific business interaction, essentially modelling a private workflow (orchestration)
• Abstract process – specifies the public message exchange between parties (choreography)

BPEL4WS

• Metamodel
BPEL4WS

- Activity

BPEL4WS vs. WSCI + BPML

Collaborative Protocols

<table>
<thead>
<tr>
<th>BPEL4WS Abstract Processes</th>
<th>WSCI</th>
</tr>
</thead>
</table>

Executable Business Processes

<table>
<thead>
<tr>
<th>BPEL4WS Executable Processes</th>
<th>BPML</th>
</tr>
</thead>
</table>

BPEL4WS (IBM, Microsoft, BEA)

WSCI/BPML (Sun, Intalio, SAP)
Differences between workflows and service composition

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Service composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Tasks</td>
<td>· Service types</td>
</tr>
<tr>
<td>· Resources (closed world)</td>
<td>· Service providers (open, dynamic)</td>
</tr>
<tr>
<td>· Resource rules</td>
<td>· Service selection rules</td>
</tr>
<tr>
<td>· Pull (worklists)</td>
<td>· Push (messages)</td>
</tr>
<tr>
<td>· Basic security (for pull)</td>
<td>· Certificates</td>
</tr>
<tr>
<td>· C++, Java, Corba, RMI</td>
<td>· HTTP, XML</td>
</tr>
<tr>
<td>· Static process</td>
<td>· Adaptive process</td>
</tr>
<tr>
<td>· &quot;Standard&quot; Black box</td>
<td>· Interface, Conversations</td>
</tr>
<tr>
<td>· No task transaction model</td>
<td>· Transactional models for e-services</td>
</tr>
</tbody>
</table>

Composite Service Description Language (CSDL)

- Concepts developed by the workflow community extended by new features:
  - 2 levels: service flow and methods flow
  - Data mapping from input / output parameters into XML
  - Certificates – which certificate should be used
  - Service templates for compositions
  - Dynamic conversation – the ability of dynamically selecting the best available service from the repository of conversations (concept similar to service communities)
CSDL

- Composite service as a process schema modelled by a graph
- Service nodes – invocations of basic or composite services
- Decision nodes – alternatives and execution flow control
- Event nodes – send and receive notifications
- Composite service may include the definition of input and output data (Java basic type or vectors, generic objects, XML documents)

FoodOnWheels Example
WfMS-driven CS execution

Peer 2 Peer Approach

Based on:
Boualem Benatallah, Quan Z. Sheng, Marlon Dumas “The Self-Serv Environment for Web Services Composition”, IEEE Internet Computing, January / February 2003
Self-Serv Web Service Composition

- Middleware infrastructure for the composition of WS
- Declarative composition of new services from existing ones
- Multiattribute dynamic selection of services within a composition
- P2P orchestration of composite service executions
- Business logic of a composite service expressed as a state chart

Major concepts

- Elementary service – an individual Web accessible application that does not explicitly rely on another Web service
- Composite service – aggregates multiple Web services which are referred to as its components
- Service community – a container of alternative services, provides descriptions of desired services without referring to any actual provider. Selection postponed until the moment of invocation
Dynamic WS selection

- Membership in service community (container):
  - Explicit mode – collection of services the container can access set during its definition time and is valid the container's entire life
  - Query mode – collection of member services specified as a query to a service registry (e.g. UDDI)
  - Registration mode – services register in the container to become its members. They can leave and reenter any time.
- Mapping between operations defined by the container and operations offered by member services *(also parameter mapping ???)*

Dynamic WS selection

- Scoring of WS – at container-invocation time a multi-attribute utility selection function is calculated to rank every service
  - Attribute values are specified in service definition (e.g. price) or derived from execution logs (e.g. expected execution time)
  - Requesters can provide weights to every attribute in a container
Peer-to-Peer Orchestration

- Self-orchestration – execution of composite services without relying on a central scheduler
- State coordinator – one for each state in the state chart. At runtime:
  - Receiving notification from other coordinators and determining when to enter the state
  - Notifying other coordinators that execution is complete
- Initial coordinator – processes invocations to the composite service, initiates the service, collects the outputs and returns them to the requester, detects and handles failures (time-outs and limited number of retrials)

Peer-to-Peer Orchestration

- Messages:
  - Between state coordinators (control flow)
  - Between coordinators and component services
- Routing tables – two per coordinator:
  - Preconditions – e.g.
    Preconditions(TA)={ready(ITA)[true],ready(DBF)[true]}
  - Postprocessing – e.g.
    Postprocessing(TA)={[true]/notify(AB)}
- P2P provides greater scalability (distributes message-processing workload across several servers)
Typical scenario

Complete Travel Planning Service (e.g. Expedia)

Outlook

- Very intensive research topic
- Many proposals (both industry and research)
- Still no universal and complete solution (planning, definition and implementation)
- Centralised vs. P2P approach
- Standard based vs. component based approach
- Problems with transactions – dissimilar transaction support in different WS makes composition difficult
Part V

Conclusion

Readings

- Readings
  - Handouts posted on the course web site
  - Explore Web Services environments
  - Explore Portlet environments
  - Read related white papers/documentation on Web Services and Portlet environments
**Project Frameworks**

- **Project Frameworks Setup (ongoing)**
  - Apache Web Server (version 1.3.20+, www.apache.org)
  - Perl (version 5.x+, www.perl.com), PHP
  - Microsoft IIS with COM+ and ASP
  - Apache Tomcat
  - (Macromedia JRun)
  - Apache Cocoon 2/XSP
  - Visibroker, Orbacus
  - RMI-IIOP
  - WebLogic, WebSphere, JBoss
  - ExoLab.org OpenCCM
  - Microsoft DNA
  - Apache’s XercesJ, XalanJ, XMLSpy, Antenna House XML Formatter, Apache’s FOP, X-smiles
  - JWS, XMI Toolkit, IBM’s Web Service Toolkit
  - POSE, KVM for J2ME, NanoXML
  - IBM Alphaworks WSTK/.Net, XMI Toolkit, EJBMaker, WebMethods, etc.

**Assignment**

- Explore the references to Web Services-Enabled Application Server technology
- All Assignments (#1-4): due by 12/07/09 by midnight
- Final Project Report: due on 12/23/09 by midnight
- Final Project Demo: online or in person by 12/23/09