Application Servers

Session 2 – Main Theme
Page-Based Application Servers

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

1. Agenda
2. Legacy Page-Based Application Servers
3. 2nd Gen Page-Based Script Oriented App. Servers
4. Summary and Conclusion
Agenda

- Application Servers for Enhanced HTML
  - ColdFusion 5.0- MX 6.1/7.0/8.0/9.0 Environment
  - PHP 5.3.8 Environment
  - XML-Based Application Servers
  - Summary
  - Class Project Overview
- 2nd Generation Page-Based Script-Oriented Application Servers
  - Microsoft IIS with COM+/ .Net, and ASP Environment
  - Servlet and Servlet Engines
  - JSP and JSP Engines
  - Apache TomCat / Adobe (Macromedia) JRun
  - eXtensible Server Pages (XSP) Environment
  - Apache Cocoon2
  - Summary
- Readings
- Assignments #2 (posted) and #3 (to be posted)

Icons / Metaphors

- Information
- Common Realization
- Knowledge/Competency Pattern
- Governance
- Alignment
- Solution Approach
What is the class about?

- Course description and syllabus:

Agenda

1. Agenda
2. Legacy Page-Based Application Servers
3. 2nd Gen Page-Based Script Oriented App. Servers
4. Summary and Conclusion
Agenda – Application Servers for Enhanced HTML

- Introduction
  - ColdFusion 5.0- MX 6.1/7.0/8.0/9.0 Environment
  - PHP 5.3.8 Environment
  - XML-Based Application Servers

“Application Servers for Enhanced HTML”

- Examples
  - Macromedia ColdFusion 5.0 – MX 6.1/7.0/8.0/9.0 Server
  - Microsoft IIS with ASP
  - WithEnterprise Pty Ltd Tango 2000/WiTango
    - [http://www.witango.com](http://www.witango.com)
    - etc.
  - Typically less expensive than Servers for standalone use, and servers with IDEs
  - Technology stays within the familiar HTML confines
The Web Application Platform

Rapid Web Application Development
Historical System Differentiators

- Visual InterDev (ASP)
  - Management of site development process
  - Scripting
  - Macromedia’s Drumbeat or NetObjects’ Fusion can be used as alternative IDEs
- ColdFusion Studio 4.5 – MX 2004
  - HTML coding
  - Basic database integration
  - UltraDev 4 or Fusion can be used as alternative IDE (4.5)
  - Dreamweaver, Flash, Fireworks, FreeHand (MX 2004)

Technology

- IDE + Application Server
- IDE
  - Creates pages with mixture of HTML and proprietary tags or script code
  - Visual page creation (textual creation possible as well)
- Application Server
  - Evaluates the code upon user requests and provides HTML pages
Tagging vs. Scripting

- ColdFusion
  - Easy tag-oriented dynamic pages for simple tasks
  - Script use when more complex coding is required
    - arrays, case & switch statements, and error handling
- Example
  - Simple phone directory application: 2 custom tags + 1 SQL statement
  - Same would take 100 lines of ASP code ...

Agenda – Page-Based Tag-Oriented Application Servers

- Introduction
- ColdFusion 5.0- MX 6.1/7.0/8.0/9.0 Environment
- PHP 5.3.8 Environment
- XML-Based Application Servers
The ColdFusion Development Platform

ColdFusion Web Applications

<CFML>
<HTML>
JavaScript

Client

ColdFusion Server

ColdFusion Studio

ColdFusion Application

ColdFusion Server
How ColdFusion Works

The ColdFusion Development Process

- Write some code
- Save it as a page (use .cfm extension)
- View it in a browser
- Write some more code
- Save the page again
- View it in a browser
- etc.
Sample ColdFusion Application

```html
<HTML>
<HEAD>
<TITLE>My First Page</TITLE>
</HEAD>
<BODY>
<STRONG>ColdFusion</STRONG>
<CFSET ProductName = "ColdFusion">
</BODY>
</HTML>
```

Outputting a Variable Value

```html
<HTML>
<HEAD>
<TITLE>My First Page</TITLE>
</HEAD>
<BODY>
<STRONG>ColdFusion</STRONG>
<CFSET ProductName = "ColdFusion">
<CFOUTPUT>
#ProductName#
</CFOUTPUT>
</BODY>
</HTML>
```
Querying a Data Source

```html
<HTML>
<HEAD>
<TITLE>Course List</TITLE>
</HEAD>
<BODY>
<H1>Course List</H1>
<CFQUERY NAME="CourseList" DATASOURCE="cfsnippets">
SELECT CORNUMBER, CORNAME
FROM CourseList
</CFQUERY>
<CFOUTPUT QUERY="CourseList">
#CORNUMBER# #CORNAME#<BR>
</CFOUTPUT>
</BODY>
</HTML>
```

ColdFusion’s Search Engine (1/2)

- **CFCOLLECTION**
  - Sets up collections that Verity uses to search indexes
  - `<CFCOLLECTION ACTION="CREATE" or "REPAIR" or "DELETE" or "OPTIMIZE" or "MAP" COLLECTION="collection_name" PATH="path_of_verity_directory" LANGUAGE="English" or "German" or "Finnish" or "French" or "Dutch" or "Italian" or "Norwegian" or "Portuguese" or "Spanish" or "Swedish">

- **CFINDEX**
  - Indexes collections from documents or database records
  - `<CFINDEX COLLECTION="collection_name" ACTION="action" TYPE="type" TITLE="title" KEY="ID" BODY="body" CUSTOM1="custom_value" CUSTOM2="custom_value" URLPATH="URL" EXTENSIONS="file_extensions" QUERY="query_name" RECURSE="Yes" or "No" EXTERNAL="Yes" or "No" LANGUAGE="language">

- **CFSEARCH**
  - Searches indexes for a match to the search criteria
  - `<CFSEARCH NAME="search_name" COLLECTION="collection_name" TYPE="criteria" CRITERIA="search_expression" MAXROWS="number" STARTROW="row_number" EXTERNAL="Yes" or "No" LANGUAGE="language">

```
ColdFusion's Search Engine (2/2)

- CFSEARCH Variables:
  - What result variables are returned from CFSEARCH
    - #DocSearch.URL# (returns index' URL)
    - #DocSearch.KEY# (unique identifier)
    - #DocSearch.TITLE# (based on document title)
    - #DocSearch.SCORE# (relevance of search term)
    - #DocSearch.Summary# (contains index' 1st 500 chars)

- CFSEARCH Return Variables:
  - #DocSearch.recordCount#
  - #DocSearch.recordsSearched#
  - #DocSearch.columnList#

ColdFusion Features - Rapid Development

- Powerful and intuitive tag-based server scripting language
- Two-way visual programming and database tools
- Remote interactive debugging
- Web application wizards & tag-based component architecture
- Source control integration
- Secure file and database access via HTTP
ColdFusion Rapid Development

Original ColdFusion Server Architecture (5.0)
ColdFusion Features - Scalable Deployment

- Multi-threaded service architecture
- Database connection pooling
- JIT page compilation and caching
- Dynamic load balancing
- Automatic server recovery and fail-over

ColdFusion Features - Open Integration

- Database connectivity (ODBC, OLE-DB, native database drivers)
- Embedded support for full text indexing and searching
- Standards-based integration (directory, mail, etc.)
- CORBA and COM+ connectivity
- Open extensibility with C/C++
ColdFusion Features - Complete Security

- Integration with existing authentication systems (NT/Win 2000 domains, LDAP directory servers)
- Advanced access control to files and data sources
- Support for existing database security
- Server sandbox security
- Support for Web server authentication, security, and encryption
Original ColdFusion Studio’s WorkSpace

ColdFusion Studio

- Supports other languages than HTML
  - Handled Device Markup Language
  - Synchronized Multimedia Integration Language
- Visual Tool Markup Language
  - Support the inclusion of tag editing dialogs
  - Support the addition of XML capabilities
- CSS integration is clumsy (separate editor)
- Link management utility limited to page by page (no site diagramming)
Original ColdFusion HomeSite Editor

- HomeSite editor
  - Supports on-the-fly typing validation
  - DTD conformance
  - Basic syntax checking
  - Can categorize tag attributes by version and types
  - Can add custom tags and attributes

ColdFusion App Server

- Supports clustering
- Addresses performance and scalability issues at most levels
- Supports ODBC, OLE, and native drivers for Oracle and Sybase
- Also supports stored procedures
- Supports server load balancing (Bright Tiger Technologies’ ClusterATS) and failover
Server Platforms

- ColdFusion
  - Windows
  - Solaris
  - Linux
  - etc.
- ASP
  - Windows
  - Use “ChiliSoft” for other servers originally

ColdFusion Original Features Set (1/2)

- Advanced Editor
- Visual Database Tools
- Two-way Visual Programming
- Web Application Wizards
- Code Re-Use
- Interactive Debugging
- Dynamic Page Quality Assurance
- Tag Property Inspection
ColdFusion Original Features Set (2/2)

- Code Sweeper
- Extensible Tag Editors
- Custom Wizards
- Visual Tool Object Model
- Customizable Workspace
- Server-Side Source Control
- Shared Project Management
- One-Step Deployment
- Remote Team Development

ColdFusion 5.0 vs. MX 6.1/7.0/8.0/9.0

- See

- MX Features
  - Server Scripting (CFML, XML, JSPs/custom tags)
  - Integrated Application Services (Flash, Web services)
  - Flexible Application Deployment
  - High Performance Architecture
  - Advanced Development Capabilities (CFCs)
  - Enterprise Systems Integration
  - Advanced Server Management
Using CFCs As A Façade (keep code that invokes Java objects out of CFML)


Agenda – Page-Based Tag-Oriented Application Servers

- Introduction
- ColdFusion 5.0- MX 6.1/7.0/8.0/9.0 Environment
- PHP 5.3.8 Environment
- XML-Based Application Servers
PHP Technology

- Server-side, cross-platform HTML embedded scripting language
- PHP is an open source project of the Apache Software Foundation
- See http://www.php.net/index2.php
- Example (hello.php):
  ```php
  <html><head><title>PHP Test</title></head>
  <body>
  <?php echo "Hello World<p>"; ?>
  </body></html>
  ```

PHP Examples (1/3)

- Showing variables
  ```php
  <?php echo $HTTP_USER_AGENT; ?>
  ```
- Getting a list of web server variables
  ```php
  phpinfo();
  ```
- Checking for Internet Explorer
  ```php
  if(strstr($HTTP_USER_AGENT,"MSIE")) {
    echo "You are using Internet Explorer<br>";
  }
  ?>
  ```
Jumping in and out of PHP mode

```php
if(strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE')) {
    echo '<center><b>You are using Internet Explorer</b></center>
    <p>
} else {
    echo '<center><b>You are not using Internet Explorer</b></center>
    <p>
}
```

Flexible HTML Forms Handling

Typical HTML form:

```html
<form action="action.php" method="post">
    Your name: <input type="text" name="name">
    You age: <input type="text" name="age">
    <input type="submit">
</form>
```

Action.php is as follows:

```php
Hi <?php echo $name; ?>.
You are <?php echo $age; ?> years old.
```
Source and binaries downloadable from:

Includes
- CGI binary plus server API versions for Apache, AOLserver, ISAPI and NSAPI
- MySQL support built-in
- Many other extensions
XML Application Server Architecture (historical)

(HP Bluestone XML Server 1.0/Visual-XML)

XML Application Server at Work

(HP Bluestone XML Server 1.0/Visual-XML)

- See Session 2 handout on “XML MOM Application Server Frameworks”
XML Application Server at Work

(Binary Evolution Velocigen)

Agenda

1. Agenda
2. Legacy Page-Based Application Servers
3. 2nd Gen Page-Based Script Oriented App. Servers
4. Summary and Conclusion
Agenda – 2nd Generation Page-Based Script-Oriented App. Servers

- Introduction
  - Microsoft IIS with COM+/.Net, and ASP Environment
- Servlet and Servlet Engines
  - JSP and JSP Engines
  - Apache TomCat / Adobe (Macromedia) JRun
- Apache Cocoon2
- eXtensible Server Pages (XSP) Environment

Application Servers for Enhanced HTML Review

- Application Servers for Enhanced HTML
  - a.k.a., Page-Based Application Servers
  - Examples:
    - Macromedia ColdFusion 5.0-MX 6.1/7.0/8.0/9.0 Server
    - Microsoft IIS with ASP
    - WithEnterprise Pty Ltd Tango 2000/WiTango
    - etc.
- Typically less expensive than Servers for standalone use, and servers with IDEs
- Technology stays within the familiar HTML confines
  - Create pages with mixture of HTML and proprietary tags or script code using (third-party) IDE, HTML editor, or plain text editor
  - Application server evaluates the code upon user requests and provides HTML pages
Agenda – 2nd Generation Page-Based Script-Oriented App. Servers

- Introduction
- Microsoft IIS with COM+/.Net, and ASP Environment
- Servlet and Servlet Engines
- JSP and JSP Engines
- Apache TomCat / Adobe (Macromedia) JRun
- eXtensible Server Pages (XSP) Environment
- Apache Cocoon2

Microsoft Active Platform – Logical Architecture

Active Platform

Active Desktop
- HTML
- Sampling
- Components
- System Services
  - Activation
  - Audio
  - Fonts
  - Print

Active Server
- HTML
- Sampling
- Components
- System Services
  - Container Services
  - Active Directory
  - Security
  - Network Services

Standard Protocols including ActiveX
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
  - Used “ChiliSoft” for other platforms (http://www.chilisoft.net/ then Sun)
  - 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+ and ASP for C++)

Microsoft IIS - COM+/.Net - ASP

- COM+ / ASP
  - Equivalent to J2EE 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 / .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

![Diagram](image.png)

- MTS Components
- ActiveX
- Compound Documents (OLE)
- MTS
- COM Infrastructure
**COM+ = DCOM/COM + MTS**

http://members.tripod.com/gsraj/misc/ebmmts/ebmmtscomp.html

**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+ 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 (1/3)

(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 class id
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):

ProgIDFromCLSID( &clsid, "xyz.activator.1"
CreateClassMoniker( clsid, &pmkClass
MkParseDisplayname( pcb,
"\northamerica\central\employee.doc", &dwEaten,
&pmkFile
pmkFile->BindToObject( pcb, pmkClass, IID_IDispatch,
&pDisp )
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)

Distributed InterNet Applications Architecture (DNA)
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 ();
}
```

Agenda – 2nd Generation Page-Based Script-Oriented App. Servers

- Introduction
- Microsoft IIS with COM+/.Net, and ASP Environment
- Servlet and Servlet Engines
  - JSP and JSP Engines
  - Apache TomCat / Adobe (Macromedia) JRun
- Apache Cocoon2
- eXtensible Server Pages (XSP) Environment
Servlets (http://www.oracle.com/technetwork/java/javaee/servlet/index.html)
  - 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

---

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

```html
<%@ 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>
```
Servlets and JSPs Examples (2/2)

- Servlet referenced in the IMG tag of the previous slide (partial):

```java
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);
        }
    }
}
```

MVC Pattern - splits user interface interaction into three different roles

- View
- Controller
- Model
MVC Review

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


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 for a comparison of IIS/ASP and Servlet/JSP technology
Implementing the “V” of “MVC” Using JSPs (1/2)

- 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 (2/2)
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)

Servets and JSP Engines

- Apache + TomCat
  - [http://www.apache.org](http://www.apache.org)
- Apache + Jrun
  - [http://www.adobe.com](http://www.adobe.com)
- Apache + Jetty
- Mainstream J2EE Applications Servers Come Bundled with an HTTPD service and Servlet/JSP engines
Introduction

* Microsoft IIS with COM+/.Net, and ASP Environment
  * Servlet and Servlet Engines
    * JSP and JSP Engines
    * Apache TomCat / Adobe (Macromedia) JRun
  * Apache Cocoon2
  * eXtensible Server Pages (XSP) Environment

---

**Apache Cocoon**

[http://xml.apache.org/cocoon](http://xml.apache.org/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://java.apache.org](http://java.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

- 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
  - <?cocoon-process type="xsp"?>
  - Result document is passed to the next Cocoon processor
  - Similar to servlet chaining
- Alternatives: Rocket, and CPan’s (originally)
**Cocoon Sitemap**

- **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
Cocoon Sitemap

- 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

- 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

<!-- 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 i</li>
<xsp:expr>i</xsp:expr>
}</xsp:logic>
</ul>
</content>
</page>
</xsp:page>

Agenda – 2nd Generation Page-Based Script-Oriented App. Servers

- Introduction
- Microsoft IIS with COM+/.Net, and ASP Environment
- Servlet and Servlet Engines
  - JSP and JSP Engines
  - Apache TomCat / Adobe (Macromedia) JRun
- Apache Cocoon2
- eXtensible Server Pages (XSP) Environment
Introduction to XSPs

- See:

- 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/>!
  ```

---

**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

Cocoon as a Processing Framework for Web Services
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 \rightarrow Cocoon Servlet \rightarrow Google WS

http://<host>/cocoon/mount/google/index.html

Maps request to index.html

index.html

.../search?q=...

executes google.xsp

Search results in HTML

SOAP request

SOAP response

Applies search-results.xsl
Summary – Legacy Page-Based Application Servers

- Page-Based Application Servers are either based on HTML tagging or scripting
- Page-Based Application Servers are less expensive to use and simpler than Servers for standalone use, and servers with IDEs
- ColdFusion is based on HTML extensions, and supports the development of tag-oriented dynamic pages for simple tasks
- PHP is a server-side cross-platform HTML embedded scripting language
- XML Application Servers are either MOM- or POP-oriented and rely on server-side processing of XML documents
Summary – 2nd Generation Page-Based Script-Oriented App. Servers

- Microsoft IIS - COM+.Net - ASP is a Page-Based Script-oriented application server
- COM+.Net and ASP are “equivalent” to J2EE 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

Readings – Legacy Page-Based Application Servers

- Readings
  - Handouts posted on the course web site
  - Explore the ColdFusion 5.0-6.1/MX and PHP 4.3.3-5.1.6 Environments
  - Read white papers under technical resources at
    - ColdFusion, and PHP related whitepapers on vendors’ sites
  - Review Web/network programming, and HTML
Readings – 2nd Generation Page-Based Script-Oriented App. Servers

- 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/
    - http://commons.apache.org/bsf/
    - Also read Microsoft IIS, TomCat, JRun, and Cocoon 2 related whitepapers on the vendor sites
  - Java as CGI on Windows
  - Mapping HTML to XML (e.g., W4F)
  - Modeling and Analysis of Software Architectures
    - XADL 2.0: http://www.isr.uci.edu/projects/xarchuci/
  - Performance Comparisons

Project Frameworks

- Project Frameworks Setup (ongoing)
  - Apache Web Server (version 2.2.21/2.3, www.apache.org)
  - Perl (version 5.14.1, www.perl.org)
  - ColdFusion 5.0-MX 6.1/7.0/8.0/9.0 (www.adobe.com)
  - PHP 5.3.8
  - Microsoft IIS with COM+.Net and ASP
  - Apache Tomcat
  - Adobe (Macromedia) JRun
  - Apache Cocoon 2/XSP
Assignments (1/2)

- Homework #2:
  - Explore references to Application Server technology
  - #2a: Investigate page-based application server development environments. Write a short report that documents your findings and recommendations with respect to selection criteria in support of page-based development environments for application server technology
  - #2b: See homework #2 specification

Assignment (2/2)

- Homework #3 (To be posted):
  - Explore references to Application Server technology
  - #3a: Investigate page-based application server 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
  - #3b: See homework #3 specification
Any Questions?

Next Session: Virtual Machines & Distributed Object Computing Platforms

- VMs and Component Technologies
  - JVM and .Net CLR
  - Abstract Component Infrastructures
- CORBA
- RMI and RMI-IIOP
- .Net/COM+
- DOC Platforms Interoperability
- Web-Enabled DOC Applications