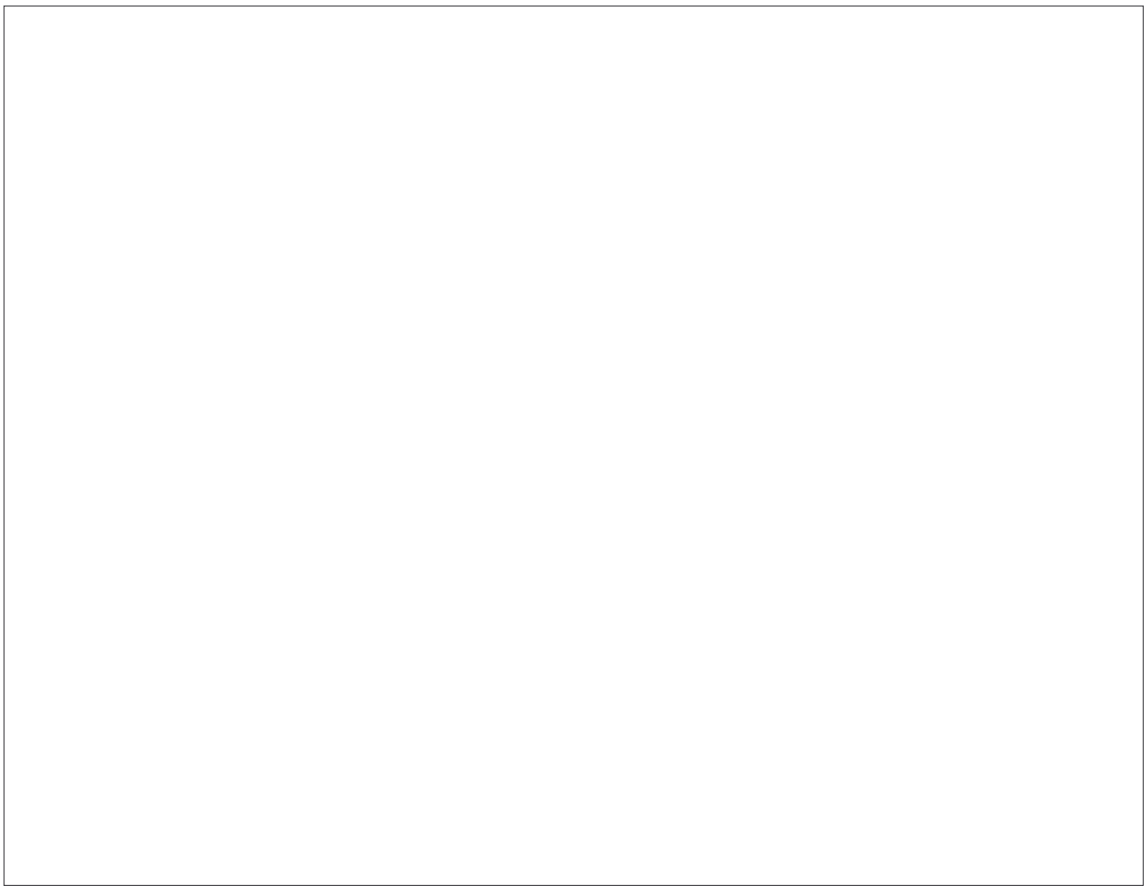
WebSphere Application Server v5.0

Developing and Deploying Web Applications



Agenda

dragonSlayer Team

Day 1

- WebSphere v5.0 Overview
- J2EE 1.3 Web Components
- WebSphere Studio Application Developer v5.0
- The WebSphere Test Environment in Application Developer
- Lab: A Login Servlet and an Access Filter
- Struts and JSTL
- Lab: Struts, JSTL with Application Developer
- WebSphere v5.0 Administration
- Lab: Deploying Web Applications to WebSphere

IBM Developer Relations dragonSlayer Team

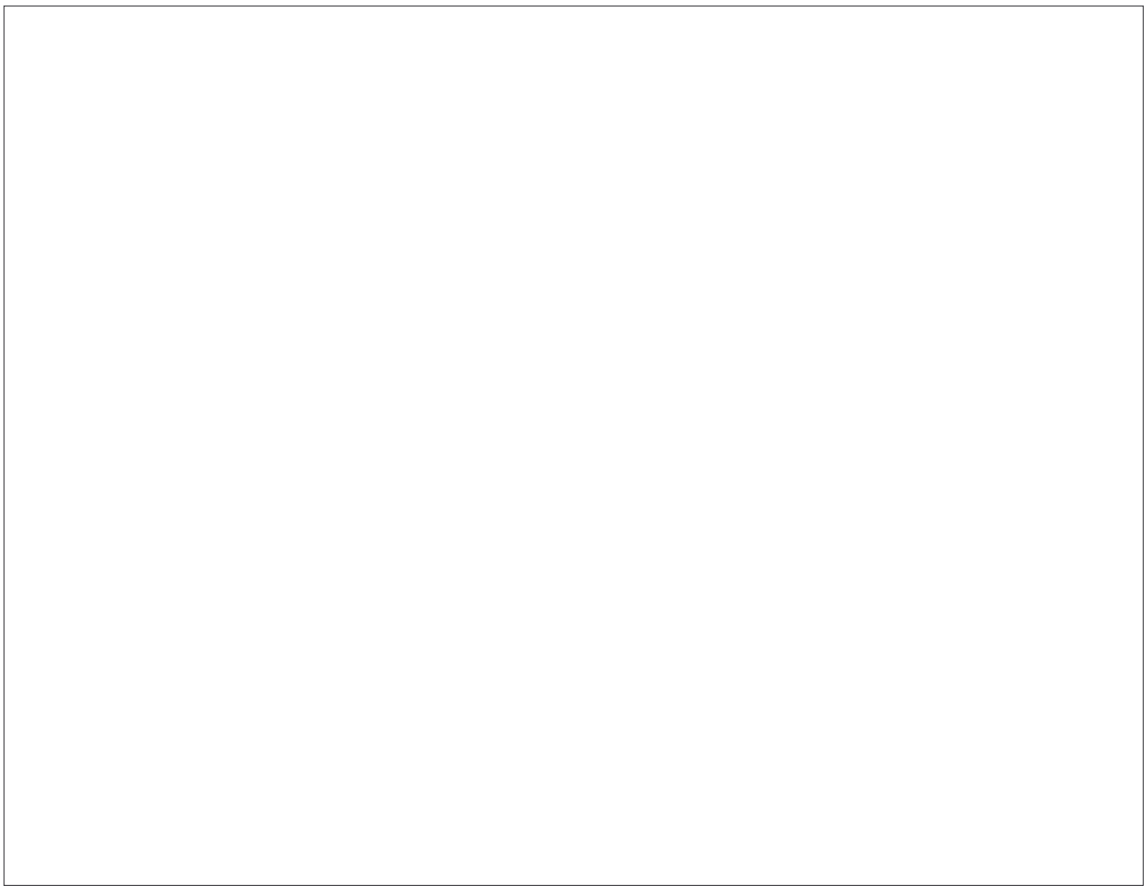
On Day 1 we focus on Web Component development using the WebSphere Test Environment included in Application Developer, and on deploying those components in WebSphere.

Day 2

- EJB Basics
- Entity Beans
- Developing EJBs with Application Developer
- Lab: Creating CMP Entity Beans
- Session Beans and Message Driven Beans
- Lab: Stateless Session Beans and Message Driven Beans
- WebSphere v5.0 Workload Management
- Web Services with WebSphere
- Lab: Deploying EJBs to WebSphere
- Lab: Adding a Web Services Interface to the Go-ForIt Application

IBM Developer Relations dragonSlayer Team

On Day 2 we will focus mainly on the EJB Development Environment in Application Developer and on deploying EJBs in WebSphere



WebSphere v5.0 Overview

dragonSlayer Team

WebSphere 5.0 Themes

- J2EE 1.3
- Scalable, Web/XML-based Admin Infrastructure
- Flexible Configurations
  - core Application Server, add-ons for scale and function
- Internal built-in JMS provider

IBM Developer Relations dragonSlayer Team

J2EE 1.3 Highlights

- EJB 2.0
- Servlet 2.3
- JSP 1.2
- Built-in JMS Provider
- INS (Interoperable Naming Service)

IBM Developer Relations dragonSlayer Team

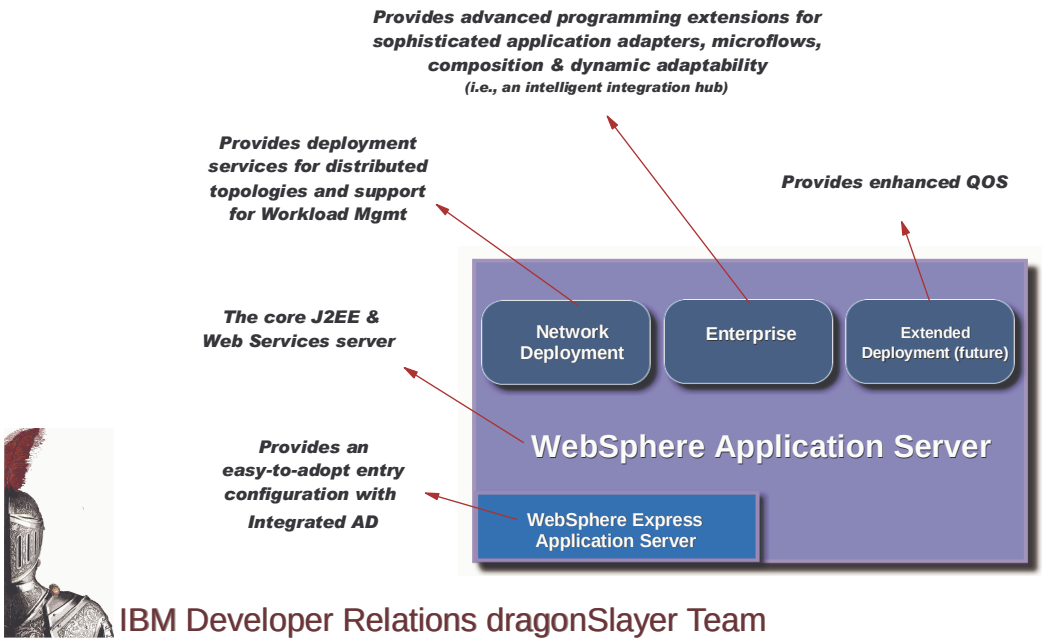
WebSphere Standards checklist

Standards	Level	WebSphere 5.0
J2EE	1.3	Fully certified and part of Sun's JCEE list
EJB	2.0	EJB 2.0 and EJB 1.1 support
JDK	1.3	JDK 1.3
Servlet	2.3	Servlet 2.3
JSP	1.2	JSP 1.2
JTS/JTA	1.0.1	w/distributed transactions
JMS	1.0.2	With Native Provider, and MQ plug-in
JDBC	2.0	2PC across heterogeneous databases
JNDI	1.2	JNDI 1.2 for EJB lookup and CosNaming
RMI/IIOP	1.3.1	Fully supported
JavaMail/JAF	1.2	Plus Domino support
SSL Security	2.0	JSSE 1.0.2 and JCE 1.2.1
XML JAXP	1.1	XML in EJBs
J-IDL/CORBA		IIOP 1.2
J2C	1.0	Bean and container managed
LDAP		Directory Server, SecureWay, iPlanet, ActiveDirectory
HTTP	1.1	**Yes**, plus across multiple Web servers
SOAP SOAP-SEC	2.3 1.0	Soap support for WebServices.
COM/ASP Support	1.0	w/Java wrapping & proxy
JMX	1.0	
XML4J	4.2	XML support:includes JAXP1.1, XMLSchema1.0, Xerces1.4.1
XSL	2.3.6	XSL parser

http://www-4.ibm.com/software/webservers/appserv/doc/latest/prereq.html

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 Packaging Positioning



IBM Developer Relations dragonSlayer Team

WebSphere Express 5.0

- Combined Development and Development Tools Offering
- Targeted at Developers
- Development Tools:
  - Entry WebSphere Studio Site Developer
- Programming model
  - Javascript/HTML/XML
  - JSP/Servlet
  - Web Services Consumption
  - No EJB support
- Entry Application Server:
  - Near zero based administration of AppServer
  - Full upward migration to higher value AppServer pkgs

IBM Developer Relations dragonSlayer Team

WebSphere Application Server 5.0

- Embedded JMS Provider
- Full J2EE and Web Services Programming Model
- Next Generation Web Services
  - WSIF
  - AXIS
- JDK 1.3.1
- Moved from WAS EE 4.x
  - CORBA Interoperability
  - ActiveX Interoperability
- Target: Development and small-scale deployments

IBM Developer Relations dragonSlayer Team

WebSphere Application Server 5.0 Network Deployment

- Adds Clustering
  - Workload Management (weighted WLM)
- Distributed Services
  - Security, Naming
- Includes a “DMZ CD”
  - incorporates Edge functionality into application server
    - load-balancing and content caching
- Distributed Systems Management
  - Single-system image, Cluster creation/management, Configuration and Application distribution, Monitoring
- Next Generation Web Services
  - Web Services Gateway
  - Private UDDI Registry

IBM Developer Relations dragonSlayer Team

The Web Services Gateway is a Middleware component that provides a framework between the Internet and intranet environment during Web Services invocations

Can be used to subset exposure of Enterprise Web Services to internet (proxy gateway)

Support for multiple transports and protocols
- SOAP/HTTP, SOAP/JMS, Direct Java via RMI-IIOP, Java over JMS

Benefits
- J2EE application
- Application server hosts the service proxy
- Provides centralized management of Web Services
- Handles protocol translation

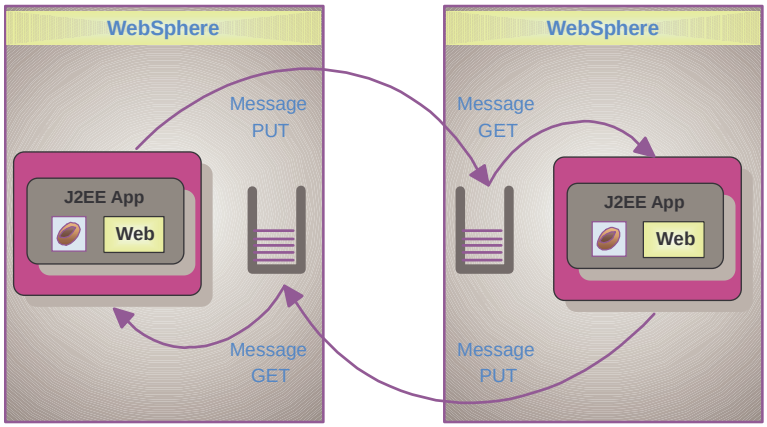
WebSphere Application Server 5.0 Enterprise

- J2EE Application Extensions:
  - Service Choreography
    - Microflows, Macroflows, Compensation via BPBeans
  - Application Profiling and Access Intent
  - Extended Unit of Work
    - Activity Session Service, Last Participant Support
  - Background and parallel processing
    - Asynch Beans, Scheduler, Deferred Execution
  - Dynamic Query
  - Container Managed Messaging
- Enterprise Integration:
  - CMP for Procedural RMs
  - Secure C++ Client
  - C++ Value type enhancements

Includes WebSphere Network Deployment

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 JMS



- WebSphere 5.0 includes a JMS implementation
  - installed as part of the application server installation
  - fully integrated with the Application server's administration and runtime

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 JMS support conforms to the J2EE 1.3 model and ncludes a JMS implementation.

It is fully integrated with the Application server's administration and runtime.

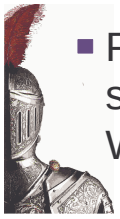
WebSphere 5.0 Administration Configuration Repository

- XML Document Repository
  - all configuration stored in collection of XML and XMI documents on the file system
  - no RDBMS
- WebSphere Common Configuration Model (WCCM)
  - Data-model representing the configuration of the system
  - Documented API for manipulating WebSphere configuration files
- Servers load directly off of documents
  - In a cluster, WebSphere manages synchronization of documents across machines
  - Application Binaries are managed as part of the repository

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 System Management - JMX

- Java Management Extensions (JMX) used both internally and externally to manage WebSphere runtime components and resources
- Provides
  - Runtime Attributes
  - Access to runtime operations
  - Access to configuration
  - Access to performance data



- Provides management tool vendors with a standard interface for monitoring and controlling WebSphere

IBM Developer Relations dragonSlayer Team

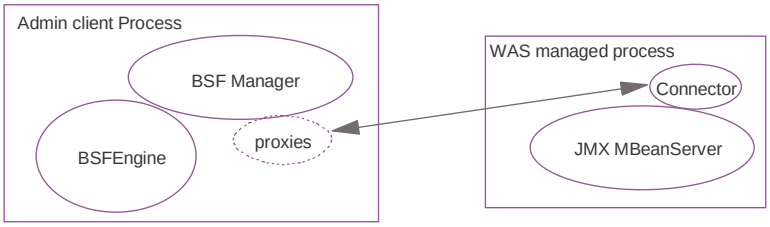
WebSphere 5.0 Web Administration

- Browser based Admin Console
  - similar to WebSphere 4.0 AEs Admin Console
- Access to both configuration data and runtime state/operations
- Course-grain admin security control and filtering
  - Administrator
  - Monitor
  - Configurator
  - Operator

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 Admin Scripting

- WebSphere 5.0 includes a scripting solution called **"wsadmin"**
  - based on Bean Scripting Framework (BSF)
  - Can use any scripting language supported by BSF to write scripts that configure and control your WebSphere installation
    - Only three languages have been tested and are supported - jacl, javascript, and jython (or jpython)
  - replaces wscp
- wsadmin makes various java objects available through language-specific interfaces
  - Scripts use these objects to communicate with MBeans running in WebSphere server processes



IBM Developer Relations dragonSlayer Team

Application Server Toolkit

- Includes a number of Eclipse-based tools
  - Debugger
  - Application Profiler
  - Eclipse-based Log Analyzer
  - Workbench

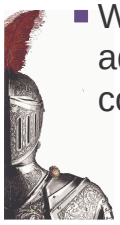
IBM Developer Relations dragonSlayer Team

This tool includes several Eclipse-based tools - including the Debugger, profiler, log analyzer, and the workbench itself.

This toolkit ships with the WebSphere Application Server, as a seperatly installable package. It has the same look and feel as WebSphere Studio, but has the reduced footprint for the non-Developer to use in a runtime environment.

Summary

- WebSphere 5.0 has full support for J2EE 1.3
- There are more flexible packaging options than in previous versions
- Except for WebSphere Express, all 5.0 versions come with an Embedded JMS Provider
- WebSphere 5.0 uses XML files as the repository for configuration information
- WebSphere 5.0 administrators can use a browser based admin console or a scripting interface to manage the configuration data



IBM Developer Relations dragonSlayer Team

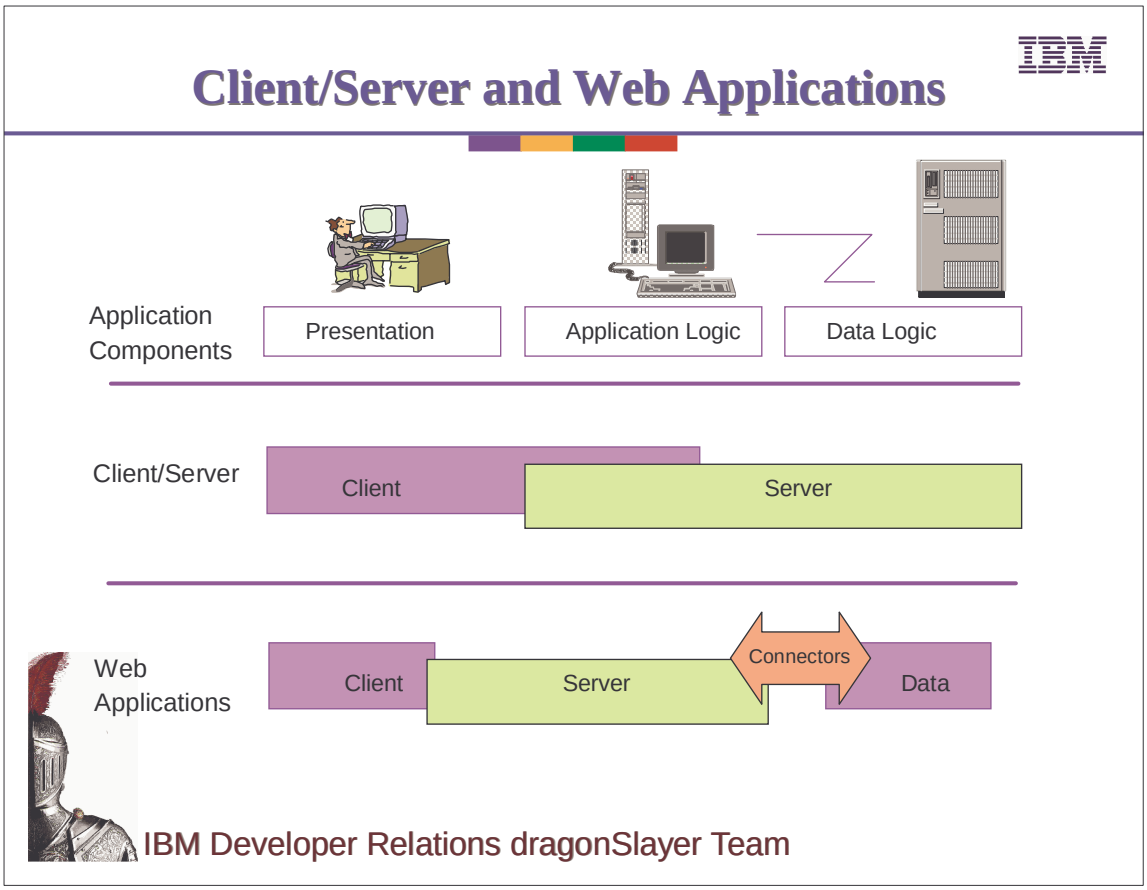
J2EE Web Component Development

dragonSlayer Team

Agenda

- Anatomy of a Web Application
- J2EE Server Model
- Servlets
- HTTPSession
- Servlet Filters
- JSP pages
- Tag Libraries
- JSP/Servlet Interaction

IBM Developer Relations dragonSlayer Team



Client/Server is a multi-tiered computing model that allows an application to split the presentation and application logic across different platforms. The client could have business logic built into its application.

Web Applications have a more distinct separation between presentation and application logic.

Web Application

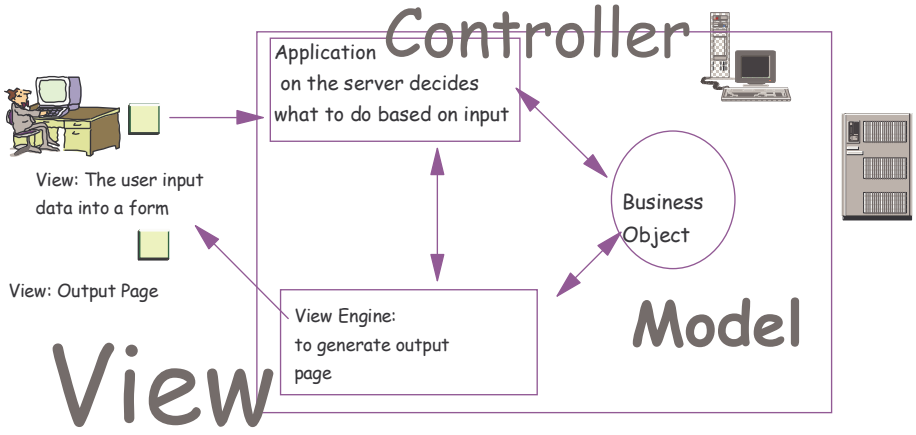
- The most common 'application structure' used in Web Applications is called Model View Controller (MVC)

- A good Analogy for MVC is a clock or a watch
  - The Model is the time [for example 10:10 PM] (Application Object)
  - The View is the clocks Presentation (Screen Presentation)
  - The controller is the clocks mechanism (defines the way the user interface reacts to user input)

IBM Developer Relations dragonSlayer Team

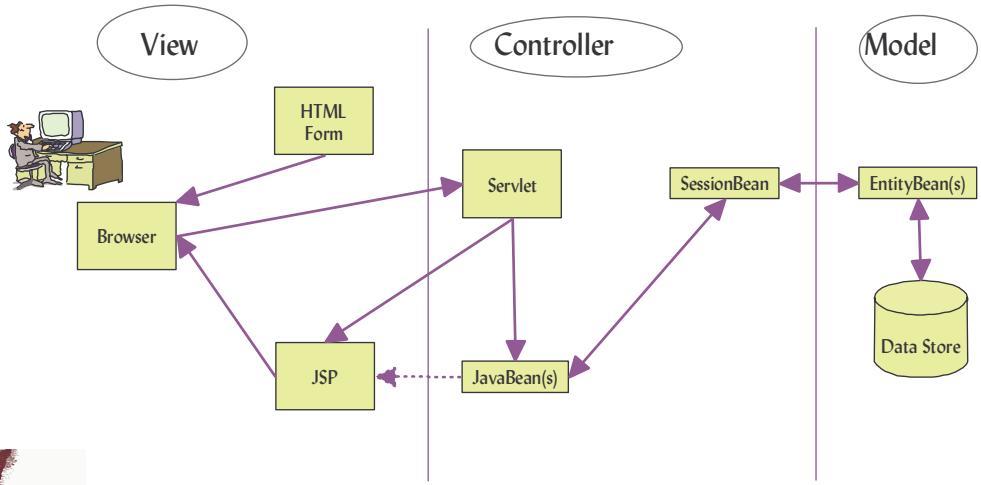
Model View Controller (MVC)

- User inputs data into a form and submits it
- This is a simple representation of the MVC architecture



IBM Developer Relations dragonSlayer Team

Application Architecture



IBM Developer Relations dragonSlayer Team

More on MVC

- View
  - Implemented using static forms and JSPs
  - Usually developed by Graphic Artists with little or no programming skills
- Controller
  - Implemented using Servlets which may use Session Beans and/or JavaBeans
  - Usually developed by programmers
- Model
  - Implemented using Entity Beans or just JavaBeans in some applications
  - Developed by more experienced programmers

IBM Developer Relations dragonSlayer Team

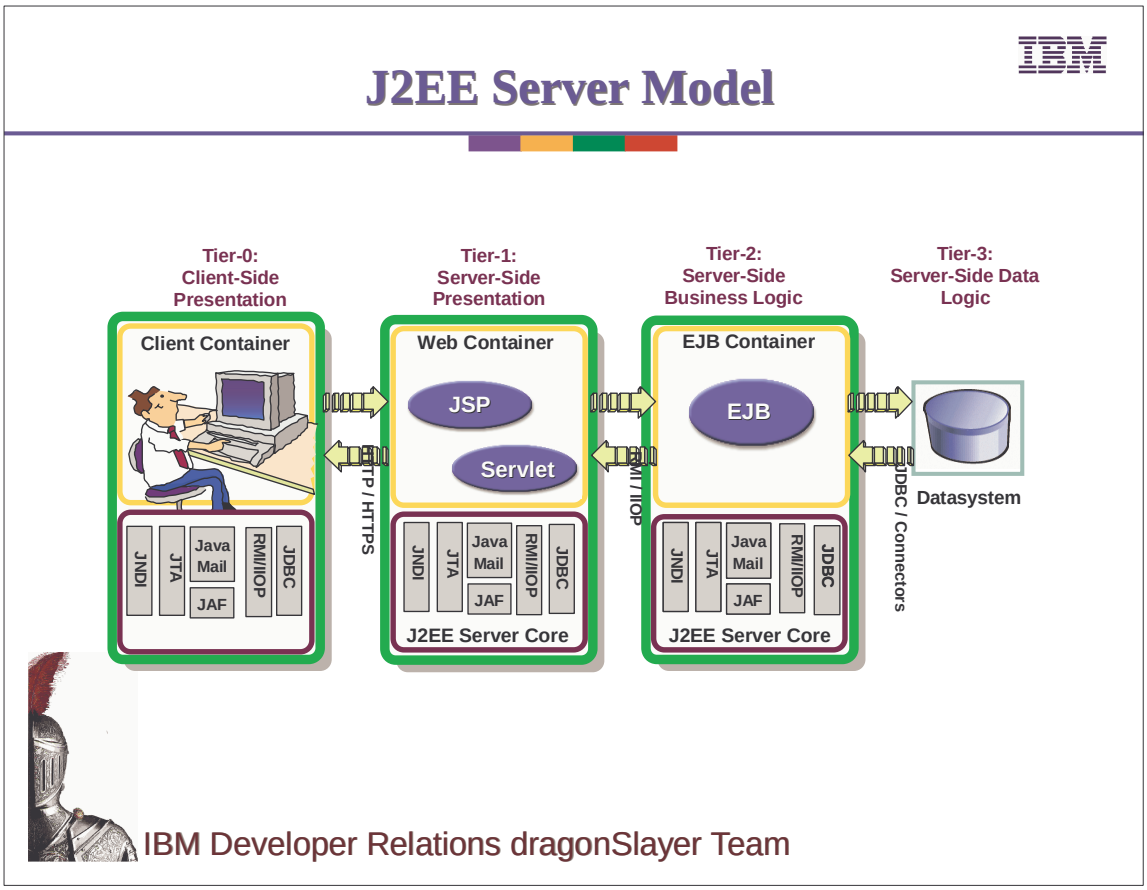
Introducing the Go-ForIt Application

- Simplified version used in labs in this course
- A User-to-Business application for a fictitious company developed by consultants in Developer Relations
- Company provides a means for clients to request services and 'personal assistants' (PAs) to respond to requests
- Developed using eXtreme programming (XP- a set of practices for software development)
- A series of ongoing technical articles describing how the GoForIt application was developed are available on the Web

IBM Developer Relations dragonSlayer Team

More information about XP and the GoForIt project can be found at the follwoing URL:

http://www.ibm.com/developerworks/ibm/library/i-slayers.html?dwzone=ibm



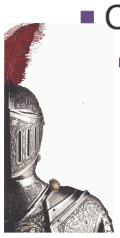
The discussion on application architectures leads naturally to introducing Java 2 Enterprise Edition. The J2EE specification defines a packaging structure which maps perfectly to the MVC architecture discussed in the previous charts.

All segments of a J2EE applications are packaged in a separate container. At this point, it is sufficient to mention that, according to J2EE, not only the Enterprise Java Beans, but Servlets, JSPs, and even the Java "thick" clients, have their own separate containers. This allows for a very clean separation of roles and for a very natural mapping into tiers: the client container implements the GUI - or physical rendition of the information on the screen, the Web Container, where JSPs and Servlets reside, implements the presentation logic, while the EJB container is a natural candidate to hold the business logic.

The containers and the code that is contained in the containers take advantage of wide array of standard J2EE functions and APIs - provided by the J2EE platform - in our case, WebSphere Application Server 5.0.

J2EE Technologies

- Components
  - Software units provided by application developers
    - Servlets, JSPs, EJBs, client components
  - Run inside the container provided by the platform
    - Container offers services and comms support
- Services
  - Functions available to the J2EE components
    - JDBC, JTA/JTS, JNDI, ...
    - Implemented by the platform provider (WebSphere)
- Communications
  - Enable communication between components
    - RMI/IIOP, JavaMail, JMS, ...
    - Provided by the container



IBM Developer Relations dragonSlayer Team

Components are to be provided by the application developers and include servlets, Java Server Pages, and Enterprise Java Beans. These components live inside J2EE containers - which are provided by the middleware vendors, such as IBM with its WebSphere Application Server. A J2EE container needs to provide support for a number of services and communications.

Services are functions that are accessible to the components via a standard set of APIs. For example, a component has to be able to access a relational database by using the JDBC APIs, while it can use the JNDI APIs to access the naming services. These APIs need to be supported by the container.

Components need to be able to communicate with each other therefore the containers need to provide the appropriate communication mechanisms to make this happen. Example of communications included in the J2EE standards are RMI/IIOP for remote method calls, JavaMail for programmatical access to e-mail, and JMS for accessing messaging technologies.

Web Components - Servlets

- Represents a service
- Usually requested via URL
- Servlets are loaded by an Application Server
  - Upon initializing the Application Server (servlet preload)
  - At first client request
  - Upon servlet reload

IBM Developer Relations dragonSlayer Team

The single URL might represent either a GET or POST request method.

For example as part of the Login form in the GoForIt application the following line

<FORM action="/goforit/LoginServlet" method = POST>

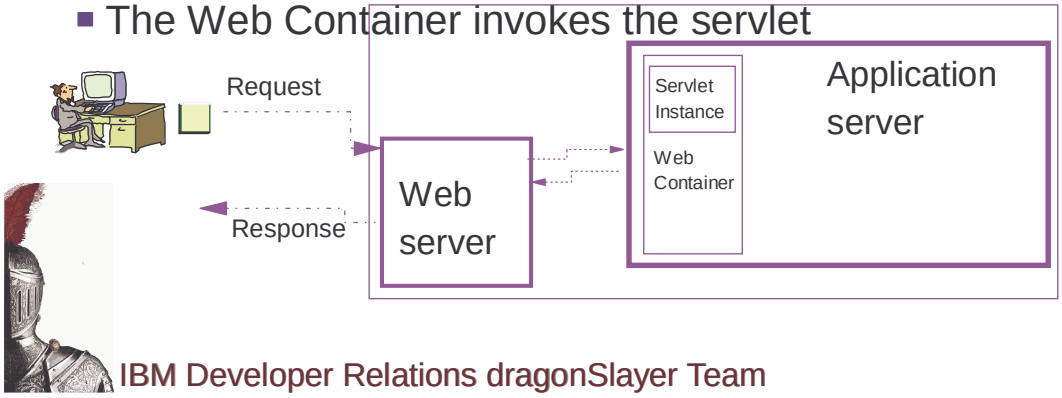
uses the POST method to initiate the LoginServlet

Servlet Invocation

- The client makes a request to the Web server naming a servlet as part of the URL

```
<FORM action="/goforit/LoginServlet" method="POST">
```

- The web server forwards the request to the Web Container, which locates an instance of the servlet class
- The Web Container invokes the servlet

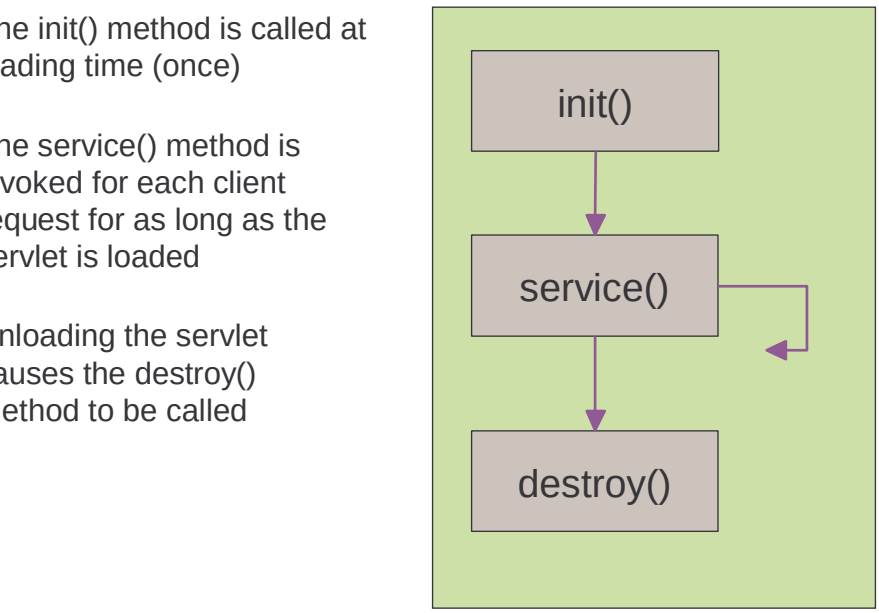


IBM Developer Relations dragonSlayer Team

The flow at a high level is as follows:
✓The client makes a request to the web server naming the servlet
✓The web server passes the request onto the application server which contains the web container
✓The web container locates an instance of the servlet class
✓The servlet is invoked

Servlet Lifecycle

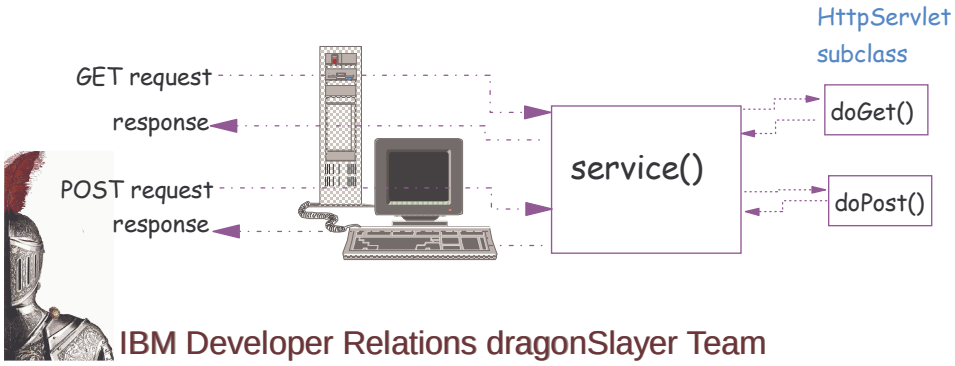
- The init() method is called at loading time (once)
- The service() method is invoked for each client request for as long as the servlet is loaded
- Unloading the servlet causes the destroy() method to be called



IBM Developer Relations dragonSlayer Team

HTTP Servlet subclass

- javax.servlet.http.HttpServlet
- Parent class of all HTTP based servlets
- Has two HTTP specific methods
  - doGet(): handles a GET request (URL)
  - doPost(): handles a POST request (HTML form)



IBM Developer Relations dragonSlayer Team

Every servlet must implement the javax.servlet.Servlet Interface

Most servlets implement it by extending javax.servlet.http.HttpServletor javax.servlet.GenericServlet

An HTTP servlet subclasses HttpServlet, which is a subclass of GenericServlet with added HTTP-specific functionality

GET method originally used for 'getting' information. For example a user requesting information may be required to fill out a field that describes the information he is requesting and then initiate a GET. The information that the user inputs is appended to the request URL in what's called a query string. GET requests can be bookmarked so should not be used to place an order etc.

POST method originally used to 'post' information to a server. POST uses a different technique to send information to the server. All of the data, of unlimited length, is sent directly over the socket connection as part of its HTTP body. This exchange is invisible to the client. POST requests cannot be bookmarked.

HTTP Servlet subclass

- To write a new servlet you should extend the HttpServlet class and override one or both of these methods as well as init() and destroy() if necessary
- doGet() and doPost() are called by the default implementation of service()

```
public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        // do something
    }
    public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        // do something
    }
}
```

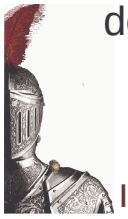
IBM Developer Relations dragonSlayer Team

There are additional doXXX methods as well, e.g., doOptions, doTrace, doDelete, doPut, that correspond to other HTTP methods. However, goGet and doPost are the most commonly used.

In most servlets you will override only one of these methods. It's a good design practice to have each servlet do one thing and do it well.

Requests and Responses

- The service(), doGet() and doPost() methods each have two parameters:
  - HttpServletRequest: provides access to request data (e.g. parameters)
  - HttpServletResponse: provides services to allow the servlet to supply a reply to the requesting client or redirect the client to a different URL
- Servlet programming amounts to reading a request and either writing a response or delegating the response to another resource



IBM Developer Relations dragonSlayer Team

The abstract method

void service (ServletRequest req, ServletResponse res)

of the servlet interface is implemented by HttpServlet. This method creates the corresponding HttpServletRequest and HttpServletResponse objects and dispatches an HttpServlet-specific service method with signature

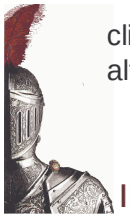
void service (HttpServletRequest req, HttpServletResponse res)

The recurring tasks performed by most servlets are "reading" parameters that are part of the client HTTP request and "writing" results back to the client as part of the HTTP response.

HttpServletResponse

- Serves as the communication channel to the client
- Allows the servlet to return content and/or errors*
- Sets the content header (type, length, etc.)*
- Redirects the server to return a particular URL

* Although servlets can write HTML directly back to a client browser this is not recommended. We will present an alternative approach later



IBM Developer Relations dragonSlayer Team

javax.servlet.HttpServletResponse is an Interface that is implemented by Web Container

These are the most commonly used methods in HttpServletResponse

getWriter()*
Returns a PrintWriter for output

setContentType(String type)*
Sets the content type for this response
Type is a MIME type

sendRedirect(String aURL)
Redirects the browser to a new URL

HttpServletRequest

- Represents a client request
- "Getters" for the different aspects of the request:
  - Request header, language, method, etc.
  - Request URL as a String
  - Servlet "path"
  - Client security type
  - Access request parameters (by name)
  - Scope for data sharing among participant objects in the request (i.e. servlet can forward request to another servlet or Java Server Page - more on this later)

IBM Developer Relations dragonSlayer Team

javax.servlet.HttpServletRequest is also an Interface.

There are many important "getters" associated with this Interface

getParameterNames()
Returns an Enumeration of parameters on the HTML page

getParameterValues(String name)
Returns the value of a multi-valued parameter

getParameter(String name)
Returns the value of a specific named parameter

In addition you will find:

getHeader(String), getHeaderNames()
getCookies()
getMethod()
getRemoteUser()
getRequestedSessionId()
getRequestURI()
getServletPath()

The accessors for the request scope "storage" is getAttribute(String) and setAttribute(String, Object).

HttpSession - Managing Application State

- Web Applications must manage state information such as current customer data
- Typically an application will involve several servlets
  - Servlets need to be stateless
- The HttpSession Interface is the application state management API
  - Represents a client/server connection
  - Its lifetime spans multiple servlets
  - Identified within requests via a session identifier

IBM Developer Relations dragonSlayer Team

We've seen that servlets should be stateless. This means they should not store any state information within themselves. The session is the proper repository for application state.

Note that

javax.servlet.http.HttpSession

is a Java Interface. In WebSphere the actual runtime object (implementation of HttpSession) is

com.ibm.servlet.websphere.servlet.session.IBMSession

HttpSession

- Ask for a Session using an HttpRequest object
  - request.getSession(boolean create)
- The HttpRequest object returns the current HttpSession
  - if create is true and no current Session exists a newly created Session is returned
- HttpSessions store application-specific data via a "key"
  - void setAttribute(String, Object)
  - Object getAttribute(String)

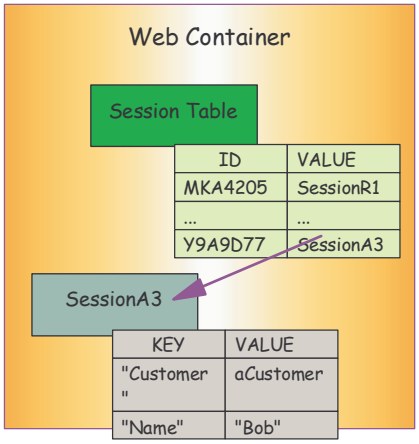
IBM Developer Relations dragonSlayer Team

If you set create to "false" and a Session doesn't exist it returns null. Always check for this.

Sessions act like Hashtables or Properties. You can store any object on a Hashtable, but it should probably be Serializable for various reasons. This is automatic if you're storing a true JavaBeans.

Sessions at Runtime - Server

- Sessions are managed by the Web Container
- Registered by ID
- The Session ID must be delivered to the client initially and presented back to server on subsequent requests

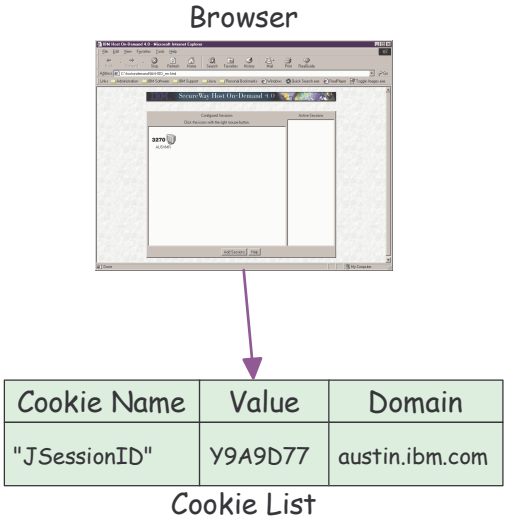


IBM Developer Relations dragonSlayer Team

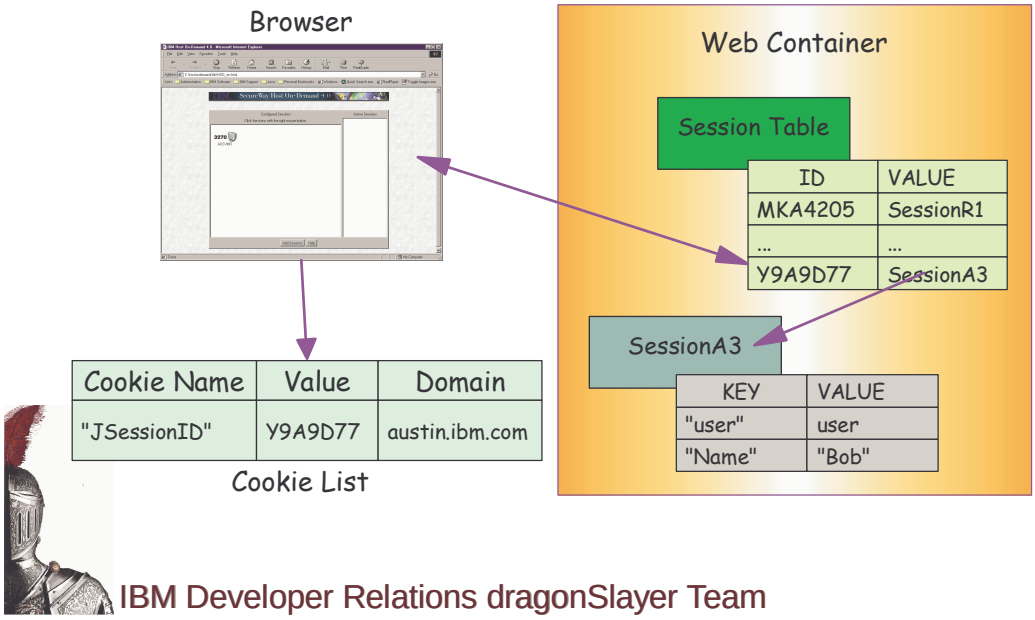
Sessions at Runtime - Client

- The preferred (default) delivery vehicle for session IDs is "transient cookie"
- Alternative "URLencoding" supported by HttpServletResponse
  - Requires ad-hoc support for client side script generated URLs

IBM Developer Relations dragonSlayer Team



Sessions at Runtime



IBM Developer Relations dragonSlayer Team

Session Invalidation

- Sessions can be invalidated either programmatically or through a time-out
  - session.invalidate() removes all values from session
- The session time-out (inactive interval) can be set for the application server as a whole
- Also session.setMaxInactiveInternal(int) can provide a session specific time-out
- Sessions are lost to client when browser is closed

IBM Developer Relations dragonSlayer Team

When a session is invalidated, it causes all of the values to be removed.

Web Components - Servlet Filters

- Allows developer to:
  - Intercept a request before it reaches a servlet
  - Modify the response after the servlet has processed the request and before the client receive the response
  - Send the response directly without sending to the next filter or the servlet
- When to use filters:
  - Authentication, logging and auditing, encryption
  - Image conversion, data compression, tokenizing filters
  - Filters that trigger resource access events
  - XSL/T filters that transform XML content
  - MIME-type chain filters
  - Caching filters
- Filters can be developed independently of the rest of the application
  - Just plug them into your Web module

IBM Developer Relations dragonSlayer Team

All filters implement the javax.filter.filter interface, which contains the following methods:

init(FilterConfig): Called by the Web container when the filter initializes. Only one instance per filter declaration in the deployment descriptor

destroy(): Called by the Web container when the filter closes. Allows filter to free resources obtained in init()

getConfig()

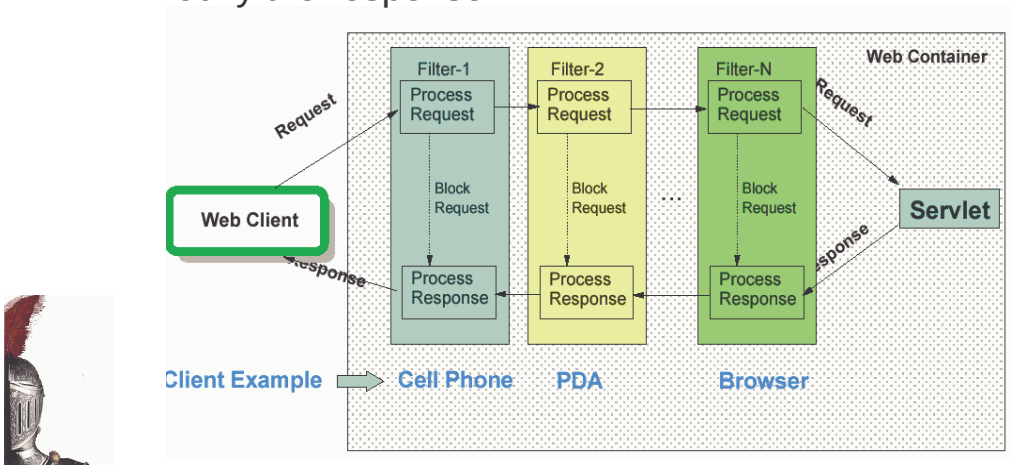
setFilterConfig()

doFilter(): Represents the main function that can modify the request and the response. This filter can implement the following pattern, or a subset of this pattern:

- Create customized implementation of the Request object (ServletRequest or HttpServletRequest) to modify the request and header
- Invoke the next entity in the filter chain, which can include the next filter, or the target Web resource (if this is the last filter, as defined in the deployment descriptor). Calling the doFilter method on the FilterChain affects the invocation of the next entity object, passing in the request and response it was called with, or wrapped versions it created. The filter can block the request, by not calling the doFilter method and sending the response back instead.
- Examine the response and wrap the Response object passed in to its doFilter method, with a customized implementation of ServletResponse or HttpServletResponse, to modify response headers or data.

Servlet Filters at runtime

- Filters can
  - Modify the request
  - Block the request and send the response direclty
  - Modify the response



IBM Developer Relations dragonSlayer Team

This chart shows how filters can be chained to provide for a cascaded processing of the HTTP request and response.

The chart also shows that a software provider may customize the application by adding filters depending on the clients that are going to be used.

For example, when installing the application at a customer's site where the web browser is going to be used, only the rightmost filter is going to be needed.

When installing the application for a customer that needs access through a cell phone, we may configure additional filters - but the logic of the application doesn't change and the filters can be developed and configured independently.

Servlet Filters and web.xml

- Filters are defined in Web Deployment Descriptors (web.xml) in the WAR file
  - Specify filter name, filter class and optional initialization parameters
  - For each filter, define the filter mapping
  - This specifies on which resource(s) to associate the filter
  - Can be associated with a single Web resource (Servlet, JSP, Static resource), or a group of Web resources (via URI)
- Filters are invoked in the same sequence as defined in the DD

Example: Filter Definition

```
<filter>
  <filter-name>Image Filter</filter-name>
  <filter-class>com.acme.ImageFilter</filter-class>
</filter>
<filter>
  <filter-name>Logging Filter</filter-name>
  <filter-class>com.sample.LoggingFilter</filter-class>
</filter>
```

Example: Corresponding Filter Mappings

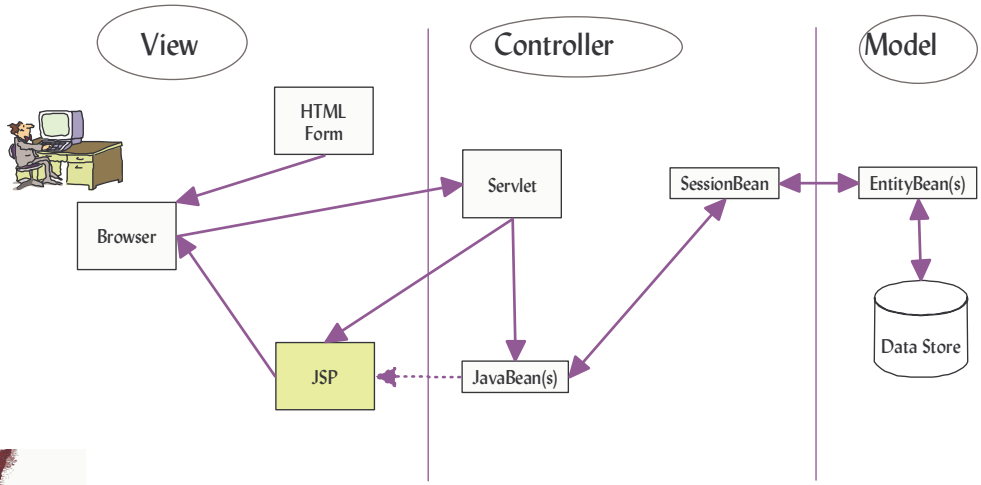
```
<filter-mapping>
  <filter-name>Image Filter</filter-name>
  <servlet-name>ImageServlet</servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>Logging Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

IBM Developer Relations dragonSlayer Team

Filter configurations defined in a Web application in the deployment descriptor, using the filter element, specifying the filter name, class and initialization parameters.

Associate filters with a single Web resource (servlet, JSP, static resource), or a group of Web resources. Make this association using the servlet name, or using the URL-pattern. This approach uses a filter-mapping element in the deployment descriptor.

Web Components - JSP pages



IBM Developer Relations dragonSlayer Team

Java Server-side Scripting

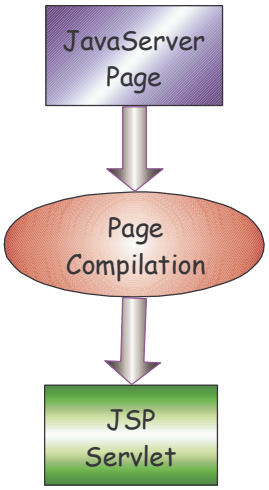
- JSP pages give the developer the ability to do Server-side Scripting in Java
  - A composite page (html and "code") is processed on the server prior to delivery to client
  - The server processes embedded code to produce dynamic content
- JSP (source) files look like HTML with some additional "Java code"

IBM Developer Relations dragonSlayer Team

The JavaServer Pages(JSP) technology is meant to provide a simplified, fast way to create web pages containing dynamically-generated content.
The technology, by virtue of it being based on Java, is designed to make it easier to build the visual components of web-based applications that work with a wide variety of web servers, application servers, browsers and development tools.

JSP Execution Model

- The JavaServer Page is converted into a servlet (JSP Servlet) by the Application Server before it is executed
- If the JSP Servlet is currently loaded, and there is a request for the page, the JSP Servlet's service method is invoked



IBM Developer Relations dragonSlayer Team

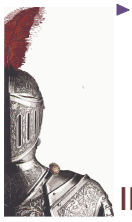
A JSP page is executed by a JSP engine, which could either be iinstalled in a web server or be part of a JSP-enabled application server such as IBM WebSphere Application Server

When the JSP engine receives requests from a client to a JSP page, it generates responses from the JSP page to the client. JSP pages are compiled into Java Servlets.

When a JSP page is first called, if the corresponding Servlet does not yet exist, it is compiled into a Java Servlet class and stored in the server memory.

JSP 1.2 Syntax: Directives

- Used to set "global" values, e.g., class declaration, required imports, output content type, etc.
- Syntax: `<%@ page variable="value"%>`
- Some variables are:
  - language
  - import
  - content_type
  - extends
  - isErrorPage



IBM Developer Relations dragonSlayer Team

Directives are instructions that are processed by the JSP engine when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries(discussed later). Directives are defined between <%@ and %>.

language - currently only "java" and "javascript" is supported
import - a list of imports to include in the generated JSP servlet
content_type - MIME type of JSP output (default is text/html)
extends - Superclass for the JSP servlet
isErrorPage - if true then an implicit object called exception of type Throwable is available in the JSP
errorPage - URL of page to handle any uncaught exceptions

JSP 1.2 Syntax - Scriptlet*

- Direct Java code makes up body of generated "method"
- Syntax: <% // any legal Java code %>
- Some of the special variables "known" by scriptlet
  - request: HttpServletRequest object
  - response: HttpServletResponse object
  - out: the servlet output writer
  - application: ServletContext
  - session: HttpSession object (if session page directive set to true)
  - exception: Throwable object (if isErrorPage set to true)

*Although you can embed Java code directly in JSPs, this is not recommended. We will present an alternative approach in the next lecture

IBM Developer Relations dragonSlayer Team

JSP pages can include small scripts, called scriptlets, in a page. Scriptlets are blocks of Java code embedded within a JSP page. Scriptlet code is inserted verbatim into the servlet generated from the page, and is defined between <% and %>.

JSP Tags

- Most JSP processing will be implemented through JSP-specific XML-based tags.
- JSP 1.2 includes a number of standard tags, referred to as the core tags. These include:
  - jsp:useBean This tag declares the usage of an instance of a JavaBeans component. If the Bean does not already exist, then the JavaBean component instantiates and registers the tag.
  - jsp:setProperty This sets the value of a property in a Bean.
  - jsp:getProperty This tag gets the value of a Bean instance property, converts it to a string, and puts it in the implicit object "out".
  - jsp:include
  - jsp:forward
- JSP 1.2 allows for custom tags (discussed later)

IBM Developer Relations dragonSlayer Team

JSP 1.2 Expressions - Syntax

- Used to output computed values
- Syntax:
  - <%= *an_expression* %>
- Semantics:
  - The expression is evaluated
  - Result is converted to a String and displayed
- Example
  - <%=new java.util.Date() %>

IBM Developer Relations dragonSlayer Team

JSP 1.2: XML Views of JSP Pages

- JSP 1.2 now allows a JSP page to be an XML document
  - In addition to the standard JSP syntax that JSP 1.1 allows
  - "JSP document" is the term used to define the JSP page in XML form
  - Uses the same file extension (.jsp) as a JSP page
  - JSP document must have jsp:root as the top element
    - jsp:root cannot appear in a regular JSP page
    - It is not valid to mix standard syntax and XML syntax in the same jsp file
  - JSP documents can pass directly to the Web container
- Advantages to coding using a JSP document:
  - XML view of a JSP page can be used to validate the JSP page against some DTD, XSD
  - XML aware tools can manipulate JSP documents
  - A textual representation can generate a JSP document by applying an XML transformation,such as XSLT
  - Will become important as more and more content is authored as XML

IBM Developer Relations dragonSlayer Team

JSP 1.2: XML Syntax example

```
<html>
<title>positiveTagLib</title>
<body>
<%@ taglib uri="http://java.apache.org/tomcat/examples-taglib" prefix="eg" %>
<%@ taglib uri="/tomcat/taglib" prefix="test" %>
<%@ taglib uri="WEB-INF/tlds/my.tld" prefix="temp" %>
<eg:test toBrowser="true" att1="Working">
Positive Test taglib directive </eg:test>
</body>
</html>
```

Example: Standard JSP page

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
   xmlns:eg="http://java.apache.org/tomcat/examples-taglib"
   xmlns:test="urn:jsptld:/tomcat/taglib"
   xmlns:temp="urn:jsptld:/WEB-INF/tlds/my.tld"
   version="1.2">
<jsp:text><![CDATA[<html>
<title>positiveTagLig</title>
<body>

]]</jsp:text>
<eg:test toBrowser="true" att1="Working>
<jsp:text>Positive test taglib directive</jsp:text>
</eg:test>
<jsp:text><![CDATA[
</body>
</html>
]]</jsp:text>
</jsp:root>
```

Example: Equivalent JSP document

IBM Developer Relations dragonSlayer Team

Web Components: Tag Libraries

- Introduced in JSP 1.1 specification - Allows the definition and implementation of custom JSP tags
- More flexible than <jsp:getProperty ../>

1. Can handle conditional logic and iteration
2. Can render data sets where number of items are not known in advance (eg for HTML tables)
3. Can customize output based on user characteristics
4. Can write your own or take advantage of many prewritten tag libraries e.g. JSTL (more on JSTL later)

IBM Developer Relations dragonSlayer Team

Using tag libraries

Custom tags contain 3 components:

1. **Tag Handler class** - java code that provides the implementation for a given tag
2. **Tag library descriptor file** - An XML document that maps tags to tag implementations
3. **JSP file** that uses the tag library

IBM Developer Relations dragonSlayer Team

The JSP loads the xml based descriptor file and then can reference specific tags in that library.

Each entry in the Tag libarary descriptor file contains a path to a java class - the tag handler class which does the actual processing

Tag Handler Class

- A tag handler class should extend either the TagSupport class or the BodyTagSupport class in the javax.servlet.jsp.tagext package
  - When tag body is ignored or returned unchanged can use TagSupport
    - e.g. ... <mytags:helloworld>This is unchanged </mytags:helloworld> or
    - <mytags:helloworld />
  - else extend BodyTagSupport
    - e.g. ...<mytags:helloworld>This text can be modified/accessed by the tag handler class</mytags:helloworld>

IBM Developer Relations dragonSlayer Team

There is also an interface called IterationTag that is implemented by both TagSupport and BodyTagSupport that allows you to have iteration support in your tags. e.g. Your tag may have to iterate through all the objects in a collection.

Tag Handler class

- To write a Tag handler you provide implementations of the doStartTag() and doEndTag() methods as needed
- If BodyTagSupport is extended you can also provide implementations of doInitBody() and doAfterBody()
- These routines will have access to :
  1. request, response, session objects
  2. tag attributes specified in JSP using the tag
  3. Tag body (if BodyTagSupport is extended)

IBM Developer Relations dragonSlayer Team

The doInitBody() and doAfterBody() methods provide the ability to implement logic before and after processing the body of the tag.

doAfterBody() is required for subclasses of TagSupport or BodyTagSupport if they need iteration support. In this case your implementation should return IteratorTag.EVAL_BODY_AGAIN from doAfterBody() to continue iterating and the container will keep calling doAfterBody(). You stop the iteration by returning Tag.SKIP_BODY and the container will call doEndTag() to end the processing of the tag

Example of a Tag Handler class

```
public class ExampleTag extends TagSupport {
   private String _foo= null;
   public int doStartTag() {

        try {
          pageContext.getOut().print("The attribute foo  is set to: " + getFoo());
         } catch (IOException e) {
          System.out.println("ExampleTag  exception " + e.getMessage());
        }

      return (SKIP_BODY);
   }

   public String getFoo() {
      return _foo;
   }

   public void setScope(String newFoo) {
      _foo= newFoo;
   }
}
```

Tag library descriptor files

- XML file that identifies tag class to the server and associates it with a particular tag name
- Should contain <tag> ...</tag> for each defined tag
- Child elements for <tag> ...</tag>
  - **name** - defines the base tag name
  - **tagclass** - fully qualified name of the tag handler class
  - **info** - a short description of the tag
  - **bodycontent** - Should be set
    - **EMPTY** - for tags without bodies
    - **JSP** - for bodies that can be interpreted as JSP source
    - **TAGDEPENDENT** - to process body in tag class
  - **<attribute>**...**</attribute>** - for each tag attribute

IBM Developer Relations dragonSlayer Team

name is how the jsp references the tag
tagclass is the path to the class file
info is not used programmatically
bodycontent specifies how to process the data between the openning and closing "tag library" tags
a tag can have as many attribute sets as needed.

An example of a TLD file. Note that several tags can be defined in a single file

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
  "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
  <!-- These attributes will be fixed  -->
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>

  <shortname>goforit</shortname> <!-- shortname for this taglib -->
  <uri>http://www.go-forit.com/taglibs/ErrandList</uri> <!--unique for this taglib-->

  <tag>
    <name>errandList</name>
    <tagclass>com.goforit.taglib.ErrandListTag</tagclass>
    <bodycontent>EMPTY</bodycontent>
    <info>Go-ForIt's tag to display a list of errands</info>
    <attribute>
      <name>scope</name>
      <required>true</required>
      <rtexprvalue>false</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

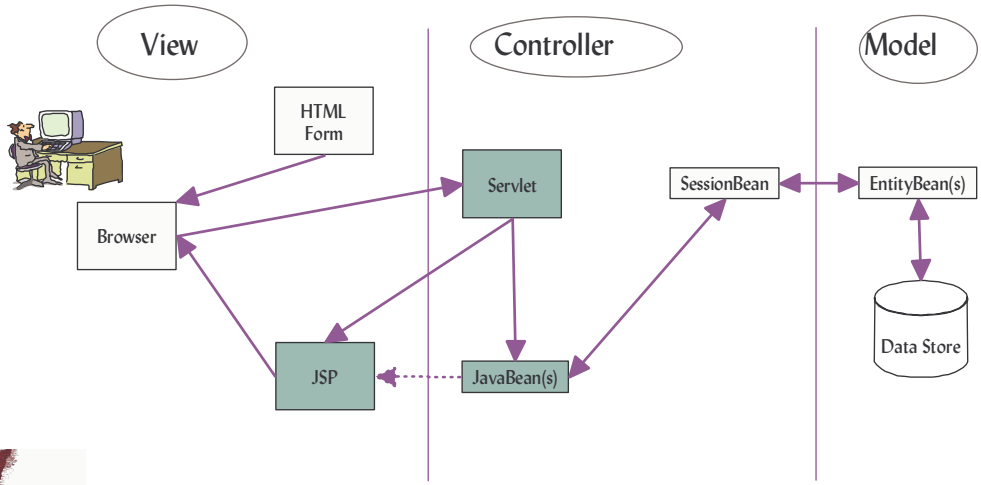
JSP files using taglibs

JSPs have 2 ways to get access to the tags in a tag library

1. Mention the tld file directly (relative to the directory of containing JSP)
   - e.g. <%@ taglib uri="goforit.tld" prefix="goforit" />
2. Mention a user defined URI and map URI to tld in Application Server
   - e.g. <%@ taglib uri="http://taglib.goforit.com" prefix="goforit"%> http://taglib.goforit.com mapped to goforit.tld in Web Application configuration

IBM Developer Relations dragonSlayer Team

JSP/Servlet Interaction



IBM Developer Relations dragonSlayer Team

Servlet/JSP interactions

- From our previous discussions we've seen
  - Servlets can contain code to output HTML
  - JSPs can contain Java code
- Neither one of these is recommended
  - HTML/JSP tools don't support HTML in servlets
  - Java code in JSPs is compiled at run time making it difficult to debug and modify safely
- Need an approach where JSPs contain no Java code (scriptlets) and servlets contain no HTML

IBM Developer Relations dragonSlayer Team

Recommended approach

- Separating presentation and content
  - Servlet interacts with back-end to obtain dynamic data to be displayed back to user
  - Dynamic data is encapsulated in Java classes/beans
  - Servlet transfers control to a JSP to render the dynamic + static data
  - JSP returns results to client
  - Programmers own data beans and servlets, page designers own JSPs

IBM Developer Relations dragonSlayer Team

Scopes for sharing data between Servlets and JSP pages

- request
  - via getAttribute() and setAttribute() in HTTPServletRequest
- session
  - via getAttribute() and setAttribute() in HTTPSession
- application
  - via getAttribute() and setAttribute() in ServletContext
    - There is one context per web application
    - Servlets call getServletContext() to get the ServletContext for the application it is deployed in

IBM Developer Relations dragonSlayer Team

In all cases, setAttribute(String, Object) requires a String identifier for each object stored in a particular scope.

The recipient of the object should call getAttribute(String) using the same identifier and cast it back to it's runtime type

Recommended approach - servlet side

- 1 - Wrap dynamic data in a Java class/bean

```
UserDataBean user = new UserDataBean();
user.setName(...) ...
```

  - 2 - Put the bean where a JSP can find it - this is done with the HttpRequest object, an HttpSession object or a ServletContext object

```
// Example using the HttpRequest object. The 1st parm is a string
//  to identify this object. Receiving JSP uses this identifier
to
// the bean from the same HttpRequest object
request.setAttribute("user", user);
```

IBM Developer Relations dragonSlayer Team

Recommended approach - servlet side

- 3 - Transfer control to a JSP - done with a RequestDispatcher which can be obtained from the ServletContext

```
// Transfers control to the specified JSP with your servlet's
// request and response objects

   getServletContext()
      .getRequestDispatcher("/aFile.jsp")
      .forward(request, response);
```

IBM Developer Relations dragonSlayer Team

Recommended approach - JSP side

- 1 - Use the <jsp:useBean .. /> tag to accept data from a servlet controller

```
<jsp:useBean id="user" type="com.goforit.UserDataBean"
   class="com.goforit.UserDataBean" scope="request" />
```

  - **id** is used to fetch and refer to bean later
  - **type** is Java type of object/bean - can use interfaces here (eg java.util.Enumeration)
  - **class** is the actual class (eg a class that implements the Enumeration interface)
  - **scope** (request/session/page/application) specifies where to look for the object

IBM Developer Relations dragonSlayer Team

Recommended approach - JSP side

- 2 - Use <jsp:getProperty .. > tag to get bean data
  - e.g. <jsp:getProperty name="user" property="name">
  - name is same as id attribute in jsp:useBean tag

  property refers to method on bean (ie getName()). Return value is rendered as a string and inserted into the data stream at the point where the jsp:getProperty tag is in the JSP file

```
<html>
    <body>
    <B>Welcome <jsp:getProperty name="user" property="name"></B>
    ...
    </body>
</html>
```

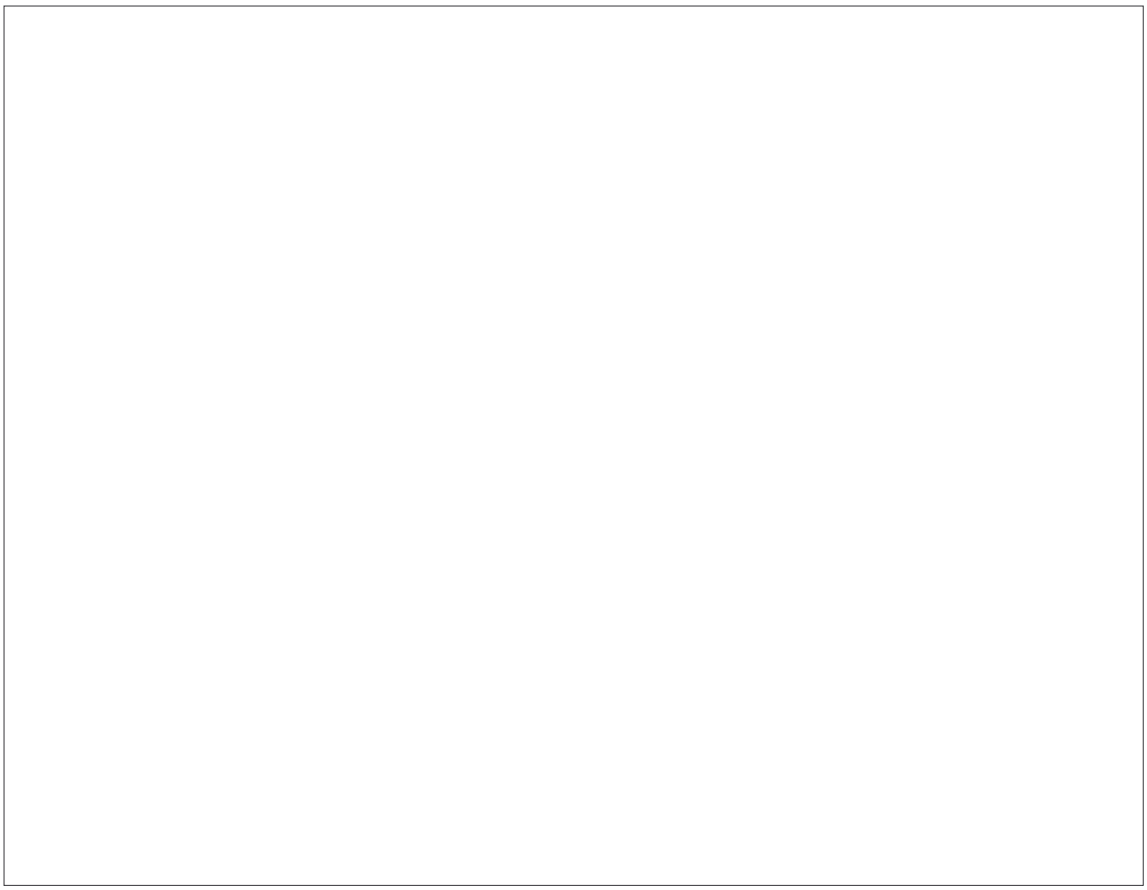
Note: Custom tags can be used instead of jsp: getProperty for more complex data rendering

IBM Developer Relations dragonSlayer Team

Summary

- MVC is the recommended application structure for Web Applications
- J2EE 1.3 Web Components include
  - Servlets
  - Servlet Filters
  - JSP Pages
  - Tag Libraries
- The HttpSession API allows developers to maintain state in their web applications between multiple HTTP requests

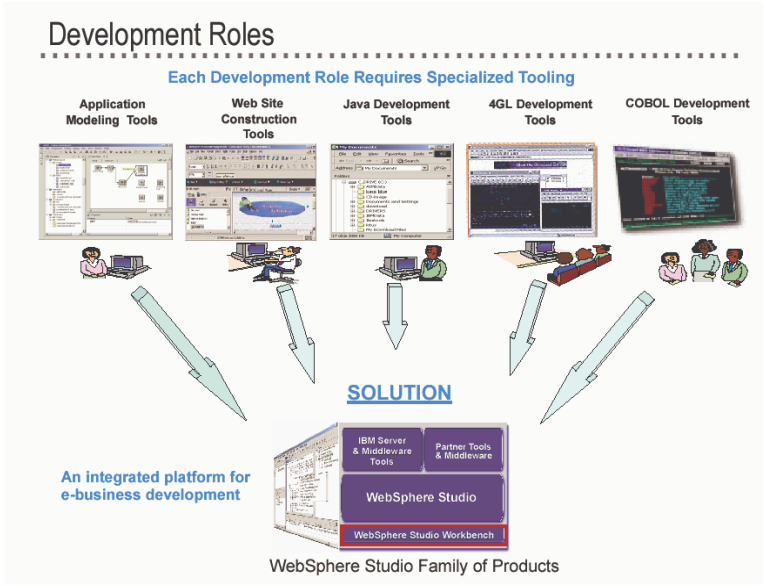
IBM Developer Relations dragonSlayer Team



WebSphere Studio Application Developer 5.0 Overview

dragonSlayer Team

Development Tools



IBM Developer Relations dragonSlayer Team

Each of these development roles demands unique functionality in their Application Development tooling to make them more productive. Things like wizards, connectors, and all of the other forms of Application Development tooling automation that free the programmer from the lower-level, mundane tasks and allow them to focus on the higher-level creativity that tasks that really leverage their intellectual capital.

Traditional tools architecture is closed and monolithic

Hard to integrate or even use together

Each tool usually implements proprietary tool services

Difference look and feel, specification semantics, UI, resource organization and management, editor, . . .

Different programming tool for each role

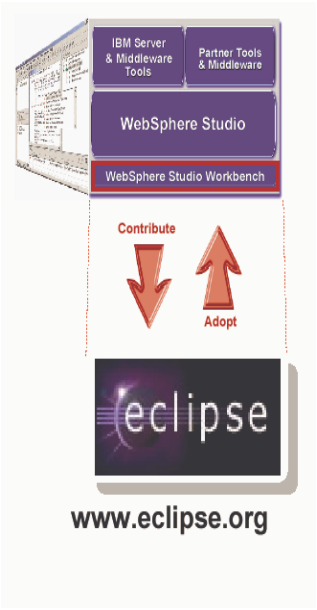
Multiple tools from different vendors for the same role

No integration between roles, tools or vendors

WebSphere Studio Workbench

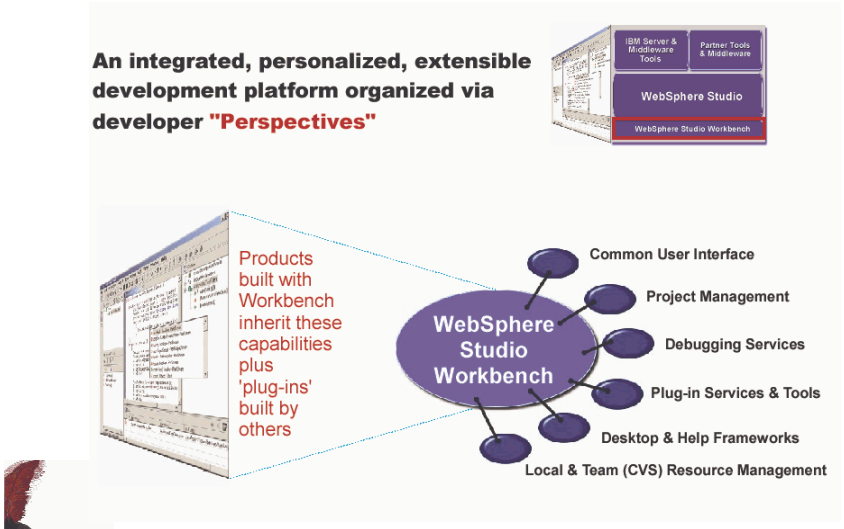
- Workbench is IBM's commercially supported implementation of Eclipse
  - Not a product but a Foundation for the WebSphere Studio family of tools
- $40M software/R&D contributed as initial Eclipse technology
  - IBM will continue participation in Eclipse development, and adopt enhancements
  - IBM will not sell the Workbench but will make it available for IBM and partner tool developers to provide a consistent integration platform for WebSphere development

IBM Developer Relations dragonSlayer Team



All WebSphere 5.0 Studio products are built on top of the Eclipse 2.0 based version of Studio Workbench

WebSphere Studio Workbench



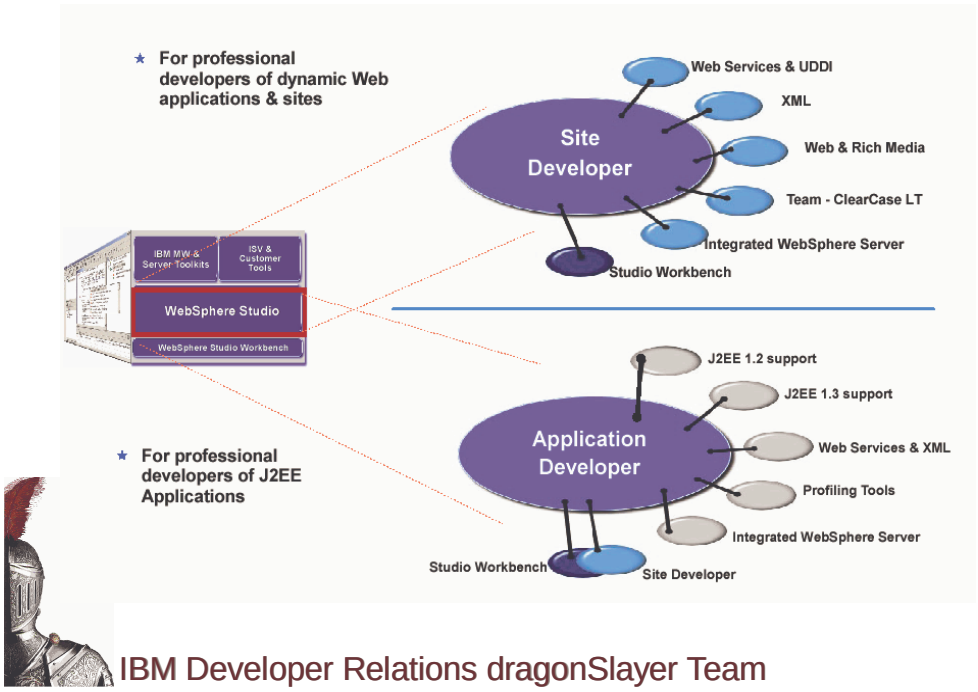
IBM Developer Relations dragonSlayer Team

The Workbench provides a set of services that all tools share including those for a

- common user interface, desktop and help system
- common interface to local resources and to shared resources
- through repository interface plug-ins
- common project management and debug facilities
- local and team resource management framework

The common UI provides an integrated platform experience for developers - enabling the personalized organization of tools, data and information in Perspectives that address specific developer roles and tasks

WebSphere Studio 5.0 Family of Products



IBM Developer Relations dragonSlayer Team

The Site Developer package is a set of tools and perspectives for professional developers of Web sites and Web applications

It delivers tools supporting open Web standards including XML, Java Server Pages, and Web services as well as Java and JavaScript through the tools inherited from the Workbench..
and rich media tools required for developing a high quality user experience

The Web services tools include those for creating services from Java components and publishing their descriptions to a UDDI repository; and for browsing a UDDI repository for available services and linking to them from the Web application via a Java Server Page

In addition to the CVS repository interface inherited from the Workbench, Site Developer add an interface to Rational's ClearCase LT and includes a ClearCase LT repository

Site developers can test their work as they develop it through an integrated WebSphere Application Server

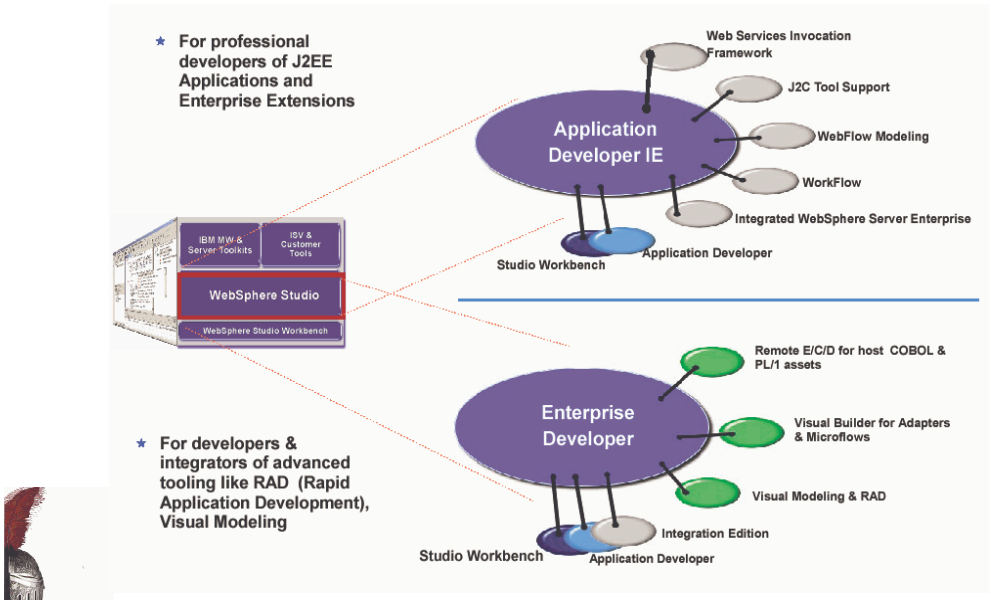
WebSphere Studio Application Developer extends the Site Developer solution - and adds:
A more comprehensive set of Java and J2EE development tools optimized for professional and team development

Data mapping tools for linking the application to the data in the databases - supporting leading databases including DB2

Performance profiling tools to help the application developer optimize the application's performance as it is being developed

The integration of the development and performance optimization tools with the WebSphere Application Server test environment makes Application Developer a very powerful environment

WebSphere Studio 5.0 Family of Products



IBM Developer Relations dragonSlayer Team

WebSphere Studio Application Developer Integration Edition extends the Application Developer. Included in Integration Edition is support for building applications around JCA adapters and Workflows.

WebSphere Studio Enterprise Developer takes the WebSphere Studio products to another level. Including all of the support of tegration Edition, Enterprise developer also supports developing applications for zOS besides WebSphere.

WebSphere 5.0 tooling matrix

WebSphere v5 Server	WebSphere Studio v5 Tool
WebSphere Express	• WebSphere Express • Site Developer • Application Developer
WebSphere Application Server for Developers	• Site Developer • Application Developer
WebSphere Application Server	• Site Developer • Application Developer
WebSphere Application Server - Network Deployment	• Site Developer • Application Developer
WebSphere Application Server Enterprise *	• Integration Edition
WebSphere Application Server (all versions) zOS	• Enterprise Developer

IBM Developer Relations dragonSlayer Team

WebSphere Express features WebSphere Studio Site Developer and a version of WebSphere which does not contain EJB support.

WebSphere Studio Application Developer 5.0 Features

- WAS 5.0, Express and 4.0 support
- SOAP, WSDL, UDDI
- JSP 1.1/1.2 & Servlet 2.2/2.3 support
- JDK 1.3 support
- Web Services tools (creation & consumption)
- XML tools
- Database tools
- Server configuration
- Deployment to WebSphere 5.0, Express and AEs/AE 4.0
- Deployment to Apache & Jakarta/Tomcat

IBM Developer Relations dragonSlayer Team

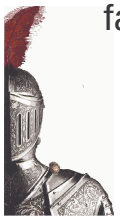
WebAphere Studio Application Developer Features, cont'd

- EJB 1.1 and 2.0 creation, mapping, testing, assembly and deployment tools
- Creation of all required J2EE packaging artifacts
- Version control and team support (CVS)
- Java IDE for editing and debugging of server side artifacts
- Automatic JUNIT test case generation
- Database support for DB2, DB2/400, DB2/390, Oracle, Sybase, SQL Server, CloudScape

IBM Developer Relations dragonSlayer Team

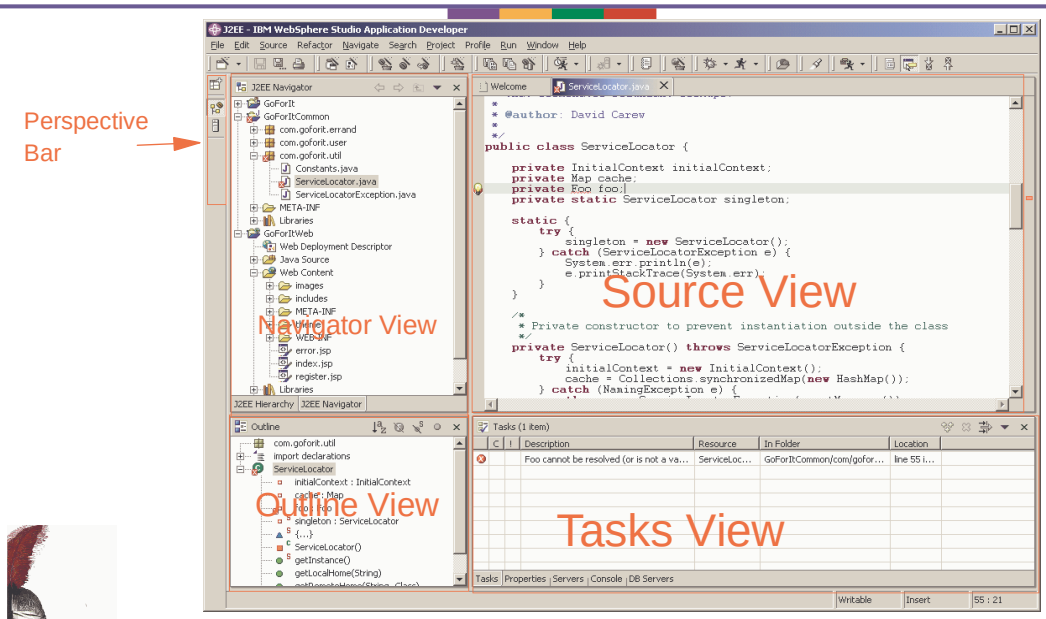
Perspectives

- A group of related views and editors is called a perspective
- Application Developer contains several perspectives
  - J2EE perspective
    - default perspective when you open the Application Developer workbench
  - Java perspective
  - Web perspective
  - Data perspective
  - Server perspective
  - ...plus more...
- Pick, define, and use your favorite perspectives and your favorite tools



IBM Developer Relations dragonSlayer Team

Workbench - Organization
Default J2EE Prespective



IBM Developer Relations dragonSlayer Team

Elements of a Perspective

- Perspectives allow team members to focus on the task at hand
- Perspectives are comprised of:
  - Editors
    - Java editor, Page Designer, XML editor
    - ...and more...
  - Views
    - Console (standard in/out), Tasks, Outline
    - ...and more...

IBM Developer Relations dragonSlayer Team

J2EE Perspective

- Full EJB 2.0 and 1.1 support
- Embedded JMS Provider for Message Driven Bean testing
- Visual tools for editing EJB DD
- WAR/EAR deployment support
- Java projects can be automatically linked to jar files in an Enterprise project
- Object - Relational mapping for EJBs
- Access Beans development
- All metadata exposed as XMI
- EJB test client web application

IBM Developer Relations dragonSlayer Team

Enhanced unit test environment for J2EE
- WAS or Tomcat
- Create multiple projects with different Unit Test configurations/instances
  - allows for versioning of unit test env
  - share unit test env across developers

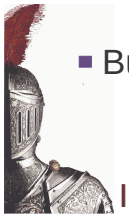
O/R mapping for EJBs
- top-down, bottom-up, or meet-in-the-middle

All metadata exposed as XMI
- no hidden metadataid

Updated EJB test client
- HTML-based
- J2EE programming model
- built-in JNDI registry browser

Server Perspective

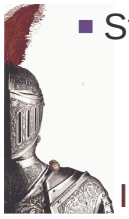
- Built-in WebSphere Unit Test Environment (WAS 5.0, Express and 4.0.2 AEs)
- Built-in Wizards allow developer to create additional Unit Test Configurations / Instances
- Built-in configurations for
  - WebSphere 5.0, Express, 4.0
  - Apache Tomcat
  - TCP/IP Monitoring Server
- Support for Local and Remote servers
- Built-in EJB Test client



IBM Developer Relations dragonSlayer Team

Web Perspective

- HTML/JSP editing
  - WYSIWYG page design, source edition, page preview
- HTTP/FTP import
- WAR import/export
- Links view (Studio Classic Relations view)
- Parsing/link management
- Built-in Servlet, Database, and JavaBean wizards
- Built-in Web Services wizards
- JSP debugging support
- Site style and template support
- Struts development environment (more on this later)



IBM Developer Relations dragonSlayer Team

HTML/JSP editing
WebSphere Page Designer

Parsing/link management
automatically fix links when resources are moved/renamed
Built in servlet, database, javabean wizards
quick generation of HTML/Servlet/JSP

Java Perspective

- Ships with JDK 1.3
- Pluggable JDK support
  - defined at project level
- Java snippet support
- Task sheet
- Code assist
- Import wizard
- Code formatter
- Class outliner
- One debugger for local/remote debugging
- Integrated unit test environment
- Refactoring support

IBM Developer Relations dragonSlayer Team

Refactoring support includes:
rename/move support for method/class/packages
fixes all dependences for renamed element
extract method, self encapsulate field
pull up method

Other Perspectives

- XML Perspective
  - XML Schema and DTD editors, XML to RDMS mapping
- Data Perspective
  - Manage databases (create tables, visual query building
- Profiling Perspective
  - Attach to local/remote agents for capturing performance data, JVMPI monitoring, Resource monitors
- Web Services Perspective
  - Tools to discover, create, test, and publish
  - Builtin UDDI Test Registry
- Team Perspective
  - For integration with Source Code Control system. Ships with ClearCase and CVS support, plugins available from other vendors (e.g. Merant PVCS)
- Component Test Perspective
  - Automatic generation of JUNIT testcases

IBM Developer Relations dragonSlayer Team

Summary

- WebSphere Workbench/Eclipse
- Products in WebSphere Studio 5.0 family
- Perspectives/tools available in WebSphere Studio Application Developer

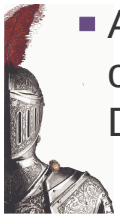
IBM Developer Relations dragonSlayer Team

The WebSphere Test Environment in Application Developer 5.0

dragonSlayer Team

The WebSphere Test Environment

- Included as a feature of WebSphere Studio Application Developer 5.0
- Contains:
  - IBM WebSphere Application Server 5.0
  - IBM WebSphere Express 5.0
  - IBM WebSphere Application Server Advanced Edition Single Server 4.02



- Allows easy development and testing of J2EE 1.3 or 1.2 applications within the Application Developer environment

IBM Developer Relations dragonSlayer Team

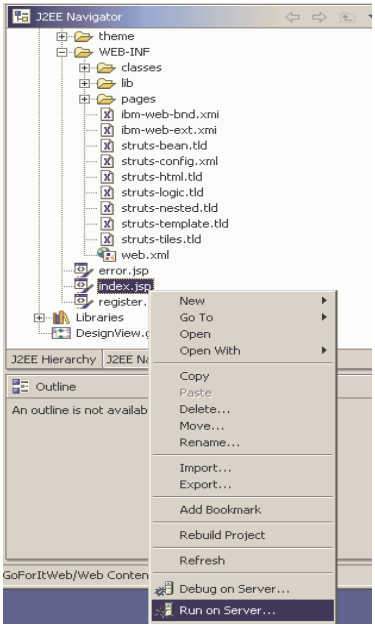
Launching the WebSphere Test Environment

- Go to:
  - Server Perspective

IBM Developer Relations dragonSlayer Team

Launching the WebSphere Test Environment (cont'd)

- Alternatively, can run the resource (servlet, jsp, html page, EJB) from the Navigator directly
- Web resources will run in a browser, EJBs will run in a special web application called the EJB Test Client



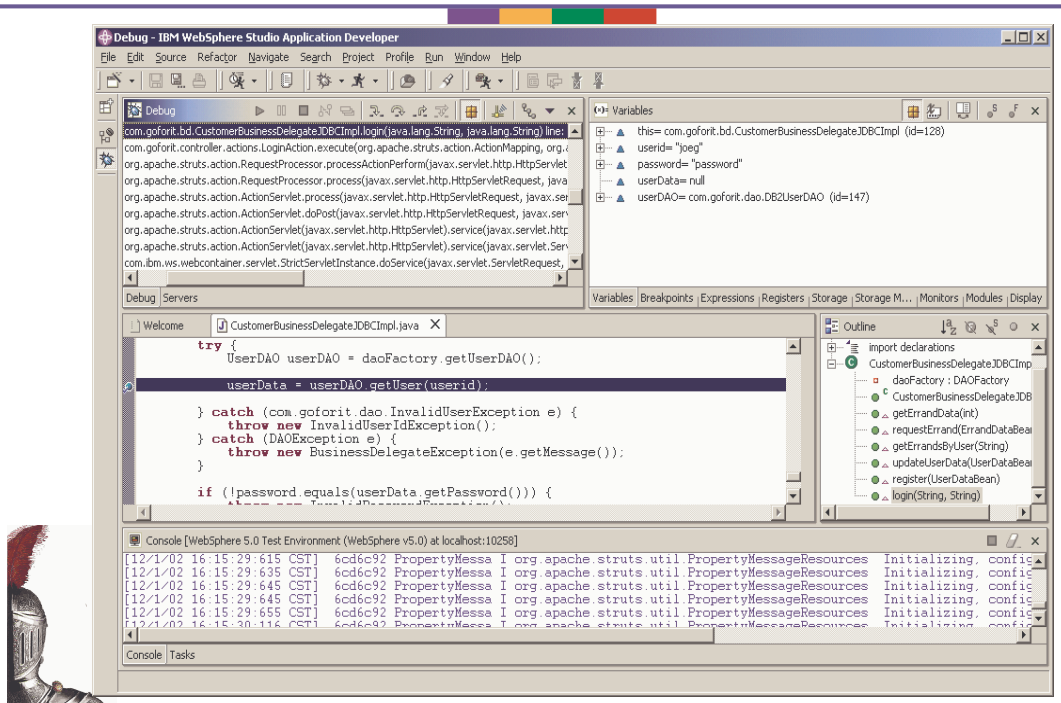
IBM Developer Relations dragonSlayer Team

Full Use of Internal Debugger

- Use the integrated debugger to stop on breakpoints and step in and through the web application code running in various threads
- Use built-in inspectors to display or evaluate objects, members, methods

IBM Developer Relations dragonSlayer Team

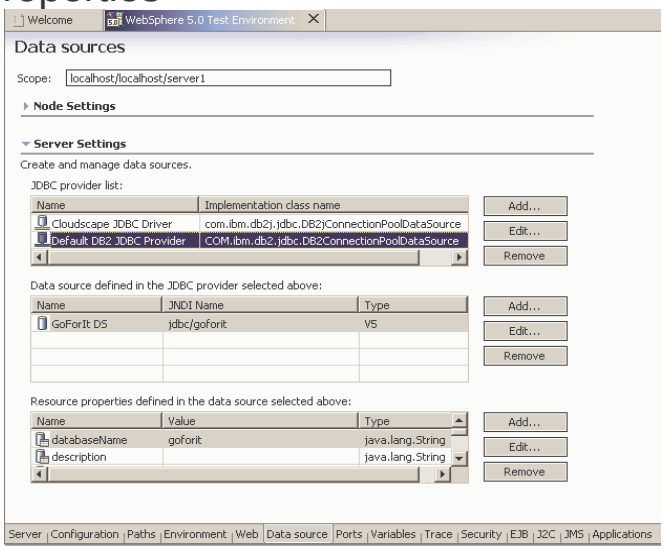
Full use of Integrated Debugger (cont'd)



IBM Developer Relations dragonSlayer Team

Visual Editor for Server configuration

- Can configure all the WebSphere server properties



IBM Developer Relations dragonSlayer Team

Server configuration

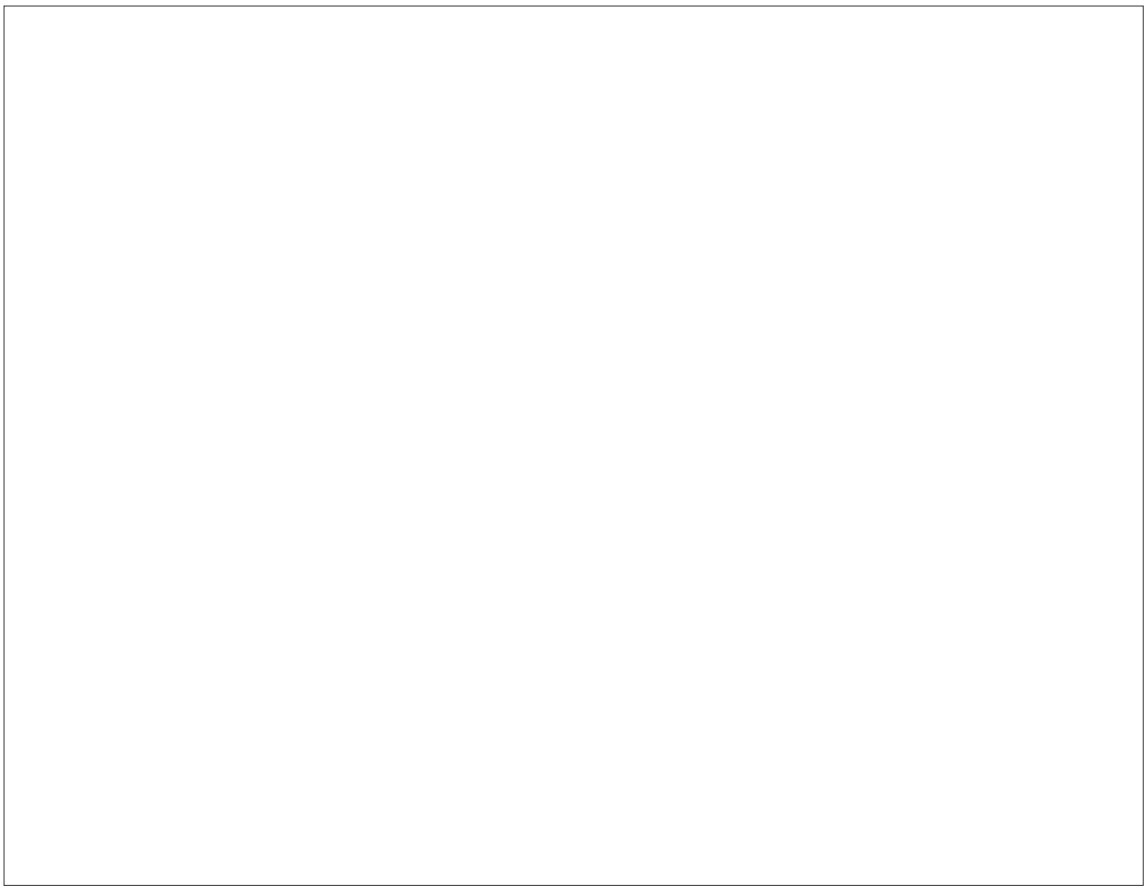
- Can configure
  - JDBC data sources
  - JCA resource adapters
  - Server class path and environment variables
  - TCP/IP ports used by server
  - J2EE Security
- Can also enable the standard WebSphere Admin Console

IBM Developer Relations dragonSlayer Team

Summary

- You've seen how to:
  - Invoke the WebSphere Test Environment feature
  - Use the Server Configuration to manage server configuration

IBM Developer Relations dragonSlayer Team



Struts and JSTL

dragonSlayer Team

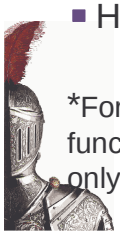
Agenda

- Overview of Struts
- Struts support in Application Developer 5.0
- Overview of JSTL
- JSTL Core Tags
- JSTL support in Application Developer 5.0

IBM Developer Relations dragonSlayer Team

What is Struts ?

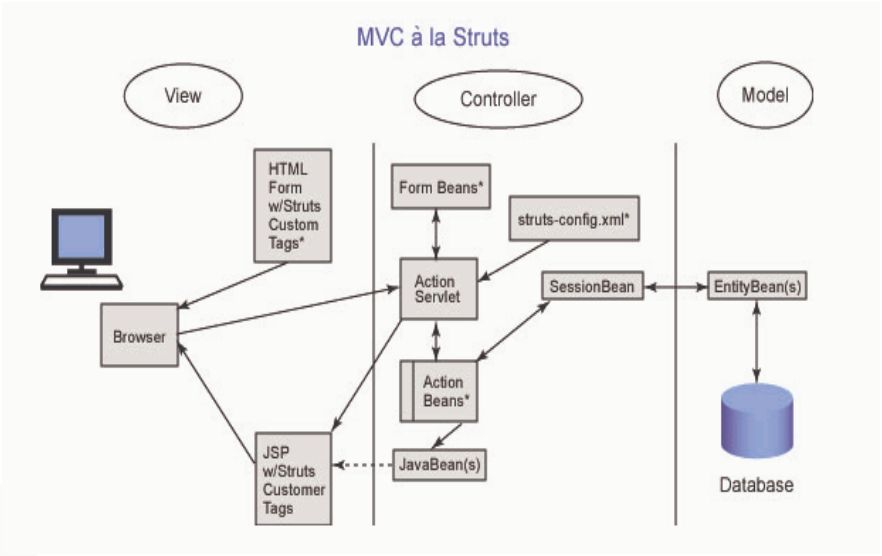
- In MVC terms, a framework for the controller tier of Web based applications
- Open source project managed by the Apache Software Foundation
  - Written originally by Craig McClanahan, the lead developer of Tomcat 4.0
- Release version is 1.02. Version 1.1 is available but still in beta*
- Has garnered a lot of mindshare among J2EE developers



*For Apache open source projects, the term **beta** means that no new functionality will be added before the release version is made available, only bug fixes and optimizations will be added

IBM Developer Relations dragonSlayer Team

Struts and MVC



IBM Developer Relations dragonSlayer Team

The Struts framework provides a series of extension points that let developers offload many of the repetitive tasks required to build a Web application. These tasks include:

Application flow
The ability to have logical names for application components, such as servlets and JSP pages, and to define the flow between components in terms of the logical names. This allows the binding of the logical names to different physical names at runtime. For a given client request, several different outcomes can be defined for the application and then the appropriate action can be taken depending on the outcome of an operation.

Internationalization
The ability to easily translate the view components into different languages without affecting the application code.

Validation
Perform server-side and client-side (JavaScript) validation.

The moving parts of Struts

- Actions
  - Struts uses the FrontController J2EE design pattern and provides a single servlet to handle all browser requests to the web application.
  - Developers subclass a Struts class called Action for each individual request
  - The mapping of specific requests to Actions is done in an XML file that is loaded by the Struts servlet at startup
- Form beans
  - When a request includes parameters (e.g. when an HTML form is submitted), Struts encapsulates the parameters in a Form bean before handing the request off to a specific Action
  - Form bean also have validation so that user input can be validated before the appropriate Action is called
- Custom Tags
  - Struts provides custom JSP tags to support
    - the mapping of HTML form elements to Form bean properties
    - the pre-population of an HTML form with data from an Form bean
    - internationalization, such as providing text that is determined by the user's locale
    - logic, such as showing a different title for a page based on how it's being used

IBM Developer Relations dragonSlayer Team

The Struts servlet is usually referred to as the action servlet

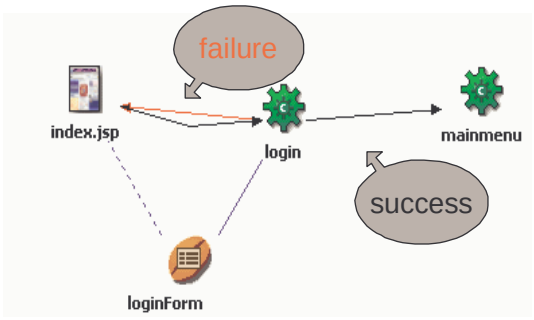
Other Struts components include

Plugins
Because Struts uses the FrontController pattern and provides the single servlet needed to handle all requests to our applications, it would appear there's no way to provide servlet specific initialization (for anything other than the action servlet) without modifying or extending the Struts action servlet. For example, if we were porting a non-Struts Web application to Struts, and the original application had several servlets, some or all of which had their own init() and destroy() methods, we would still need to have this code run.
To solve this problem, the Struts framework provides a class called Plugin that can be used to extend the function of the init() and destroy() methods of the action servlet. Application-specific subclasses of Plugin can be written and then defined in the Struts configuration file. Parameters can de defined for each plug-in and the configuration file as well, to emulate the way non-Struts applications can define initialization parameters for individual servlets in the application's web.xml file.

Validators
Once an extension to Struts, the validation framework is now part of the Struts 1.1 distribution. The validation framework provides a set of validators to validate a user's input into an HTML form. These include validators to check for valid e-mail addresses, to check for valid credit card numbers (using the Luhn formula), and to check whether a required field has been filled in. The framework can be easily extended to provide custom validators. The validation framework also has support for multiple locales so different formats can be used by the validators based on locale (such as DD/MM/YY versus MM/DD/YY for a date field depending on the user's locale). The mapping of forms and form fields to specific validators or sets of validators is done in an XML file that the validator plugin loads when it is initialized.

Struts - an example

- Application flow of a Struts based version of the Go-ForIt login function



IBM Developer Relations dragonSlayer Team

There are 2 Actions - login and mainmenu. login handles a login request and mainmenu dispalys the mainmenu

There is one Form Bean the represents the data filled in by the user when they login.

The logn Actionis configured in the Struts configuration file to forwrad to two ifferent resources. The index.jsp page and the mainmenu action. These are labelled "success" and "failure"

The login Form Bean

```
public class LoginForm extends ActionForm {
        private String userName = null;
        private String password = null;

       /**
        *  Getters and setters would go here
        */
        .....

        public ActionErrors validate(
                ActionMapping mapping,
                HttpServletRequest request) {

                ActionErrors errors = new ActionErrors();
                // Validate the fields in your form, adding
                // adding each error to this.errors as found, e.g.
                if ((userName == null) || (userName.length() < 1)) {
                        errors.add("username", new ActionError("error.username.required"));
                }

                if ((password == null) || (password.length() < 1)) {
                        errors.add("password", new ActionError("error.password.required"));
                }
                return errors;
        }
}
```

IBM Developer Relations dragonSlayer Team

The Login Action

```
public ActionForward execute(
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        LoginForm loginForm = (LoginForm) form;
            ....
        /*
         * Go to  database  to validate user
         */
        try {
                    ...

        } catch (InvalidUserIdException e) {
                /*
                * Registration exceptions go back to registration page w/error message
                */
                errors.add(
                        ActionErrors.GLOBAL_ERROR,
                        new ActionError("error.invalid.username"));
                saveErrors(request, errors);
                return (new ActionForward(mapping.getInput()));

        }
            ...
        return (mapping.findForward("mainmenu"));

}
```

IBM Developer Relations dragonSlayer Team

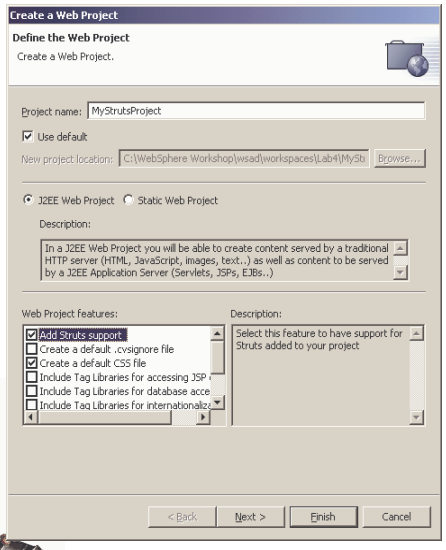
The Struts Config file

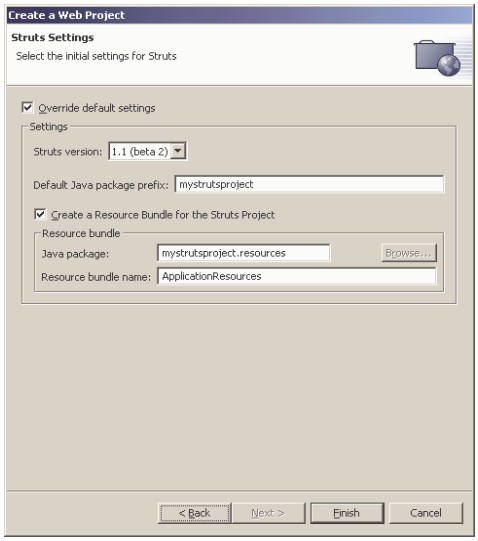
```
<struts-config>

   <!--  Form Beans  -->
   <form-beans>

      <form-bean name="loginForm" type="com.goforit.controller.forms.LoginForm" />

   </form-beans>

   <!--  Action Mappings  -->
   <action-mappings>

      <action name="loginForm" path="/login" scope="request"
              type="com.goforit.controller.actions.LoginAction" input="/index.jsp">
         <forward name="success" path="/protected/mainmenu.do" />
      </action>

   </action-mappings>

   <!--  Message Resources -->
   <message-resources parameter="com.goforit.controller.resources.ApplicationResources" />

</struts-config>
```

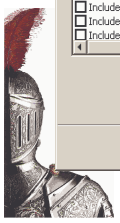
IBM Developer Relations dragonSlayer Team

Struts support in Application Developer 5.0

- Can add Struts support during Web Project creation
  - Both Struts 1.02 and 1.1 Beta 2 are supported







IBM Developer Relations dragonSlayer Team

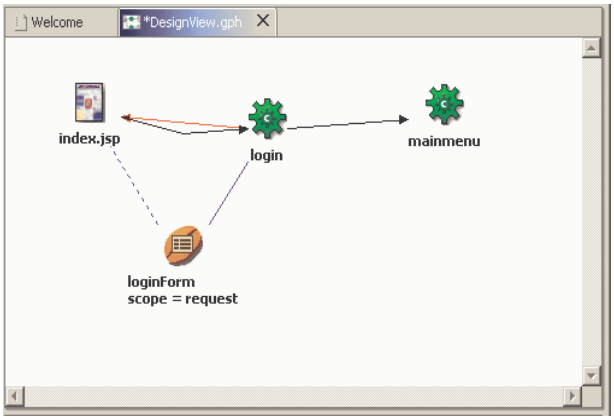
Wizards for Struts Components

- Action wizard
  - Create an Action subclass
    - All required methods are generated
    - Can optionally add Action mapping to Struts config file
- Form Bean wizard
  - Create a Form bean
    - Can optionally introspect existing HTML/JSP files with forms and automatically generate bean properties
- Action mapping wizard
  - Can optionally create an Action subclass

IBM Developer Relations dragonSlayer Team

Struts Visual Editor

- Application Developer includes a Visual editor for Struts development
  - Can add components and their relationships and then bring up wizards for each
  - Can drag existing resources (HTML/JSP, Actions, Form Beans) from Navigator view into diagram



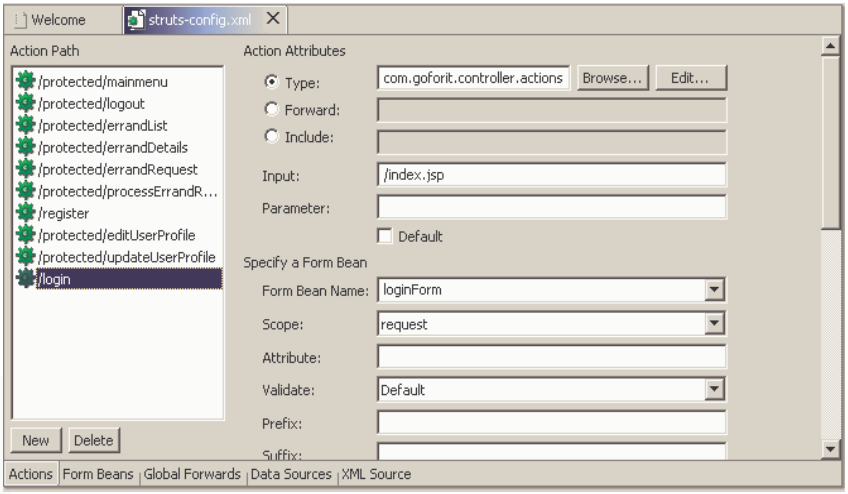
IBM Developer Relations dragonSlayer Team

Double clicking on an added component in the Visual editor brings up the appropriate wizard

Diagrams are files with a .gph extension

Visual editor for struts configuration file

- Tabs for various components
  - Source view allows editing source directly



IBM Developer Relations dragonSlayer Team

JSTL

- Java Standard Tag Libraries - JSR 52
  - JSP 1.2 based set of tag libraries developed using JCP
  - Standardizes a set of common functions that developers previously had to code themselves or obtain from different external sources
  - Consists of 4 separate libraries
    - Core - Tags for looping, including/excluding content based on runtime conditions, generating URLs for session tracking, importing form other resources and redirecting responses to different URLs
    - XML - Tags for parsing XML documents, transforming them using XSTL, looping over a set of nodes in an XML document and performing conditional processing based on node values
    - I18N - Tags for internationalization and locale specific formatting
    - Database access - Tags for accessing databases via JDBC directly

IBM Developer Relations dragonSlayer Team

URIs and recommended prefixes

- Web Containers that support JSTL will recognize the following URIs
- The prefixes listed are not mandatory but are recommended in the JSTL spec

Library	URI	Prefix
Core	http://java.sun.com/jstl/core	c
I18N	http://java.sun.com/jstl/fmt	fmt
XML	http://java.sun.com/jstl/xml	x
Database	http://java.sun.com/jstl/sql	sql

Example declaration in a JSP using the Core library

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
```

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 ships with an implementation of JSTL 1.0

JSTL Core Tags

- The most commonly used core tags
  - General purpose tags
    - c:out - write to JSPWriter for the page
    - c:set - sets the value of a runtime variable
  - Conditional tags
    - c:if - conditionally includes some JSP text
    - c:choose - JSP equivalent to Java switch statement
      - c:when - nested in c:choose, specifies a boolean condition, equivalent to case in switch statement
      - c:otherwise - nested in c:choose, equivalent to default in switch statement
  - Looping tags
    - c:forEach - repeats JSP content in body until end conditions are met
      - Exposes access to loop index and has flags that indicate first and last iterations of the loop

IBM Developer Relations dragonSlayer Team

c:choose id equivalent to the switch statement in C and Java where **break** is used after each case statement

<c:choose>

<c:when> ..</c:when>
<c:when> ..</c:when>
<c:when> ..</c:when>
<c:otherwise> ... </c:otherwise>

</c:choose>

The first <c:when> that evaluates to true will have it's body included. If none of the <c:when> boolean conditions evaluate to true then the <c:otherwise> body will be included

JSTL Expression Language (EL)

- A language for accessing runtime data
  - All JSTL tags recognize EL in their attribute values
  - JavaScript like access to bean properties, java.util.Map entries, array elements and elements of java.util.List
    - examples
      - ${myObject.property} - a bean with a getProperty() method
      - ${myObject["myKey"]} - a Map with an entry with the key "myKey"
      - ${myObject[aVar]} - acessing the Map using a runtime variable
      - ${myObject[1]} - an array or list entry
      - ${myObject[aVar + 1]} - accessing the array or list using a runtime variable
      - Properties can be nested
        - ${myObject.property.arraySubProperty[3]}

IBM Developer Relations dragonSlayer Team

It is expected that EL will be part of JSP 2.0

JSTL - EL

- EL expressions can evaluate to true or false
  - examples
    - ${myObject.intProperty % 2 == 1} - evaluates to true if the value of intProperty is an odd number
    - ${empty myObject.stringProperty} - evaluates to true if stringProperty is null or ""
  - This feature is most commonly used with the conditional tags

```
<c:if test = "${!empty myObject.firstName}" >
  <c:out Hello <c:out value="${myObject.firstName}" />
</c:if>
```

IBM Developer Relations dragonSlayer Team

A JSTL example

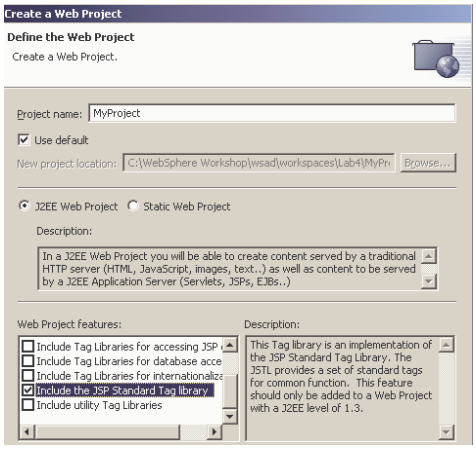
```
<jsp:useBean id="errands" scope="request" type="java.util.Collection"></jsp:useBean>
<c:choose>
  <c:when test="${empty errands}">
    <P><B> You haven't requested any errands !</B>
  </c:when>
  <c:otherwise>
    <c:forEach items="${errands}" var="eachErrand" varStatus="stat">
      <c:if test = "${stat.first}">
          <!-- Setup HTML table here -->
      </c:if>

        <!-- Format each row of data -->
        <TR> ....
          <TD align="center"><c:out value="${eachErrand.id}"/></TD>
            .
            .
            .
       </TR>
       <c:if test = "${stat.last}">
         <!-- Close HTML table -->
       </c:if>
    </c:forEach>
  </c:otherwise>
<c:choose>
```

IBM Developer Relations dragonSlayer Team

In this example we loop through a collection and format an HTML table.

JSTL support in Application Developer 5.0

- New Web Project wizard will add jstl.jar to your WEB-INF/lib if JSTL support is checked

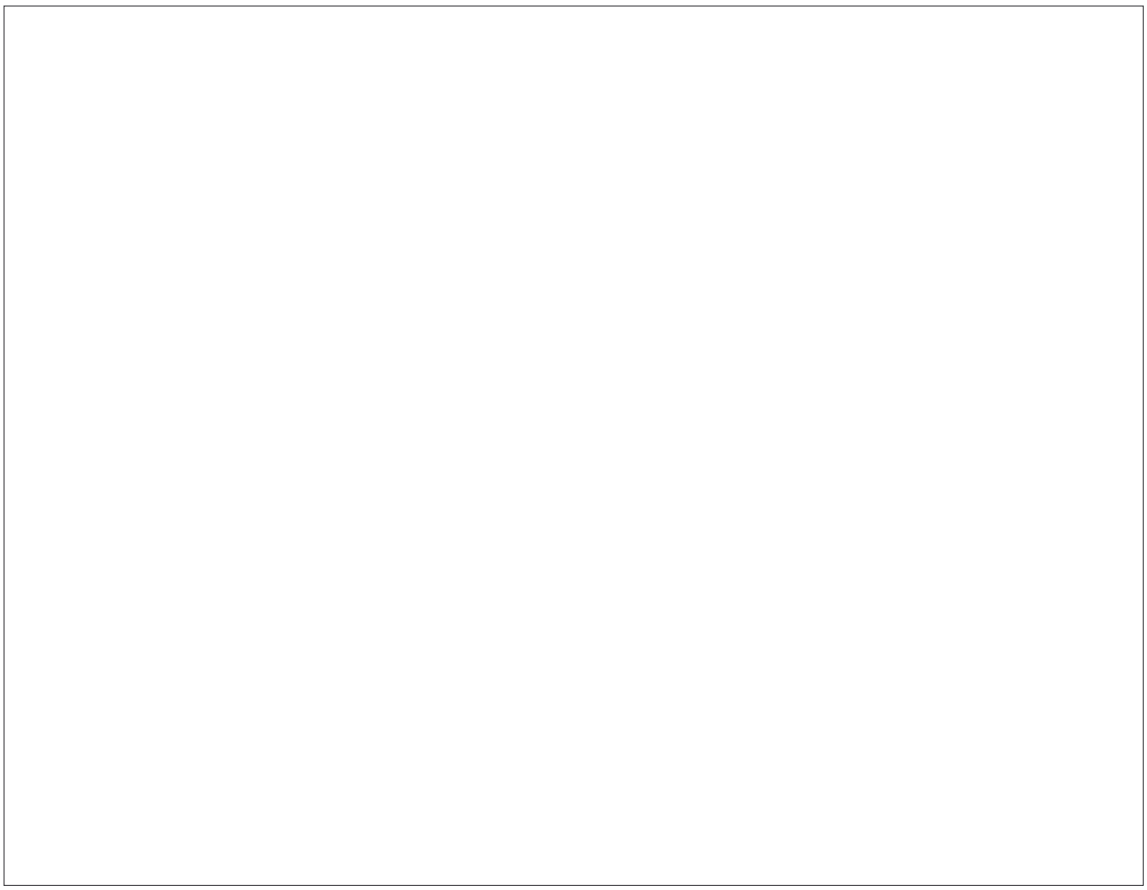


IBM Developer Relations dragonSlayer Team

Summary

- We covered the basics of Struts and the JSTL core tags
- We also looked at the support in Application Developer for Struts and JSTL

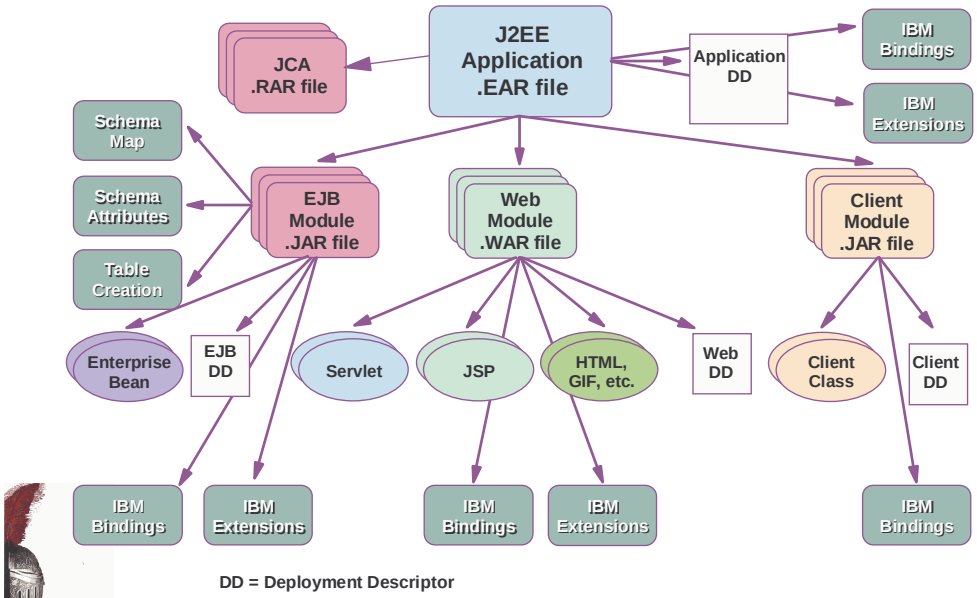
IBM Developer Relations dragonSlayer Team



WebSphere 5.0 Administration

dragonSlayer Team

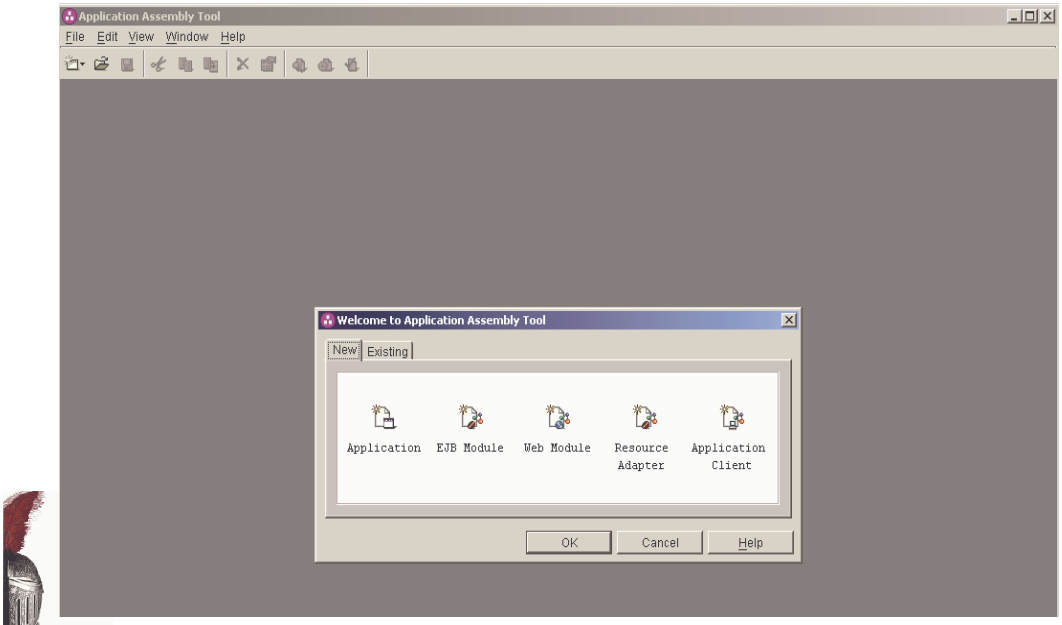
WebSphere 5.0 Application Packaging



IBM Developer Relations dragonSlayer Team

RAR modules can be installed globally or packaged with each Enterprise Application

Application Assembly Tool (AAT)

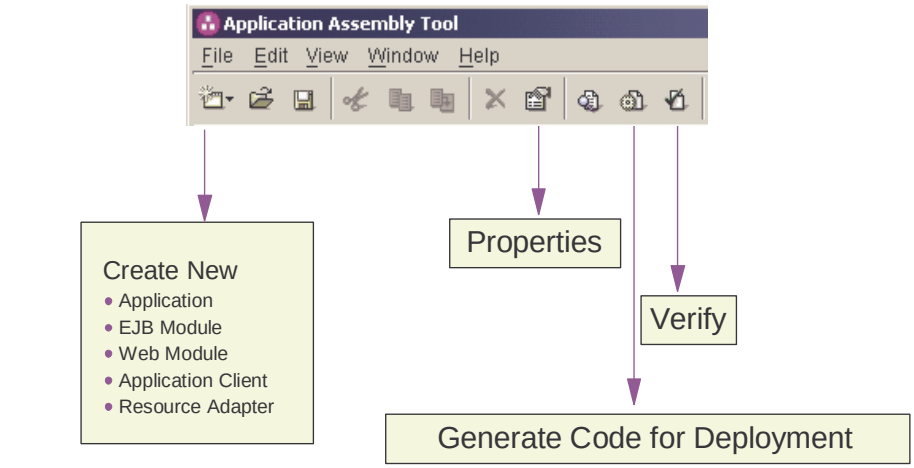


IBM Developer Relations dragonSlayer Team

AAT can be started from the First Steps window, the Windows Start menu or the command line. AAT assembles enterprise applications for deployment into WebSphere. It also allows editing of existing modules and applications. The WebSphere Admin Server does not need to be running to launch AAT.

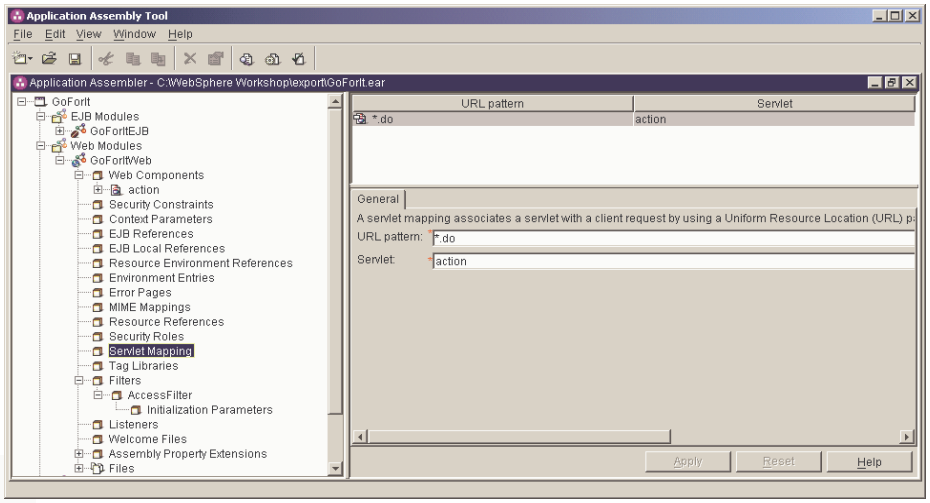
Location:
%WAS_ROOT%\bin\assembly.bat
$WAS_ROOT/bin/assembly.sh

AAT Toolbar



IBM Developer Relations dragonSlayer Team

Editing Module and Application Properties



IBM Developer Relations dragonSlayer Team

After completing the wizard or after opening an existing module or enterprise application, traverse through tree in the left pane, select the object whose properties you wish to edit, modify the values and save the file.

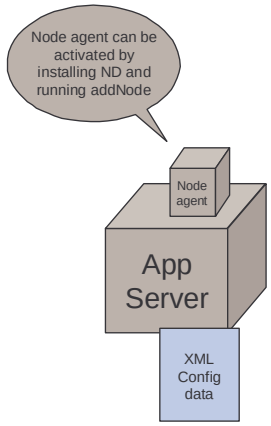
Assembly and Deployment in Application Developer

- Tools to Export resources as
  - EAR file
  - EJB Jar file
  - WAR file
  - Jar file
  - RAR file
  - Application Client JAR file
- Add WARs, EJB JARs, create context root, set security, etc using the various deployment descriptor editors.

IBM Developer Relations dragonSlayer Team

WebSphere 5.0 Base - Administration

- Configuration data is in XML files
- Admin client programs are used to modify configuration settings
  - Web based Admin Console
  - WebSphere Scripting (wsadmin)



IBM Developer Relations dragonSlayer Team

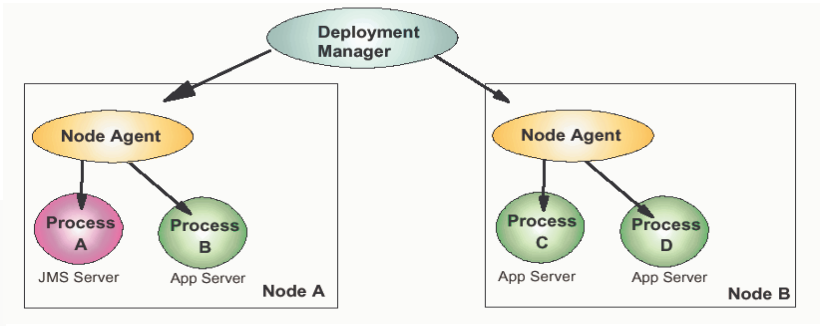
WebSphere Application Server 5.0 Base resides on a single node. That is, each machine holds a separate installation of the product that is unaware of installations on other machines.
The configuration files are in XML.

In the Base Application Server, the node agent code is there but is dormant. The server.xml for the node agent is created when addNode is done

WebSphere Application Server Network Deployment provides centralized administration of multiple nodes, allowing you to administer nodes on multiple machines.

WebSphere 5.0 Distributed Administration

- Distributed Administration requires WebSphere Network Deployment to be installed
- Terminology
  - Managed Process - individual server or process like Application Server or JMS Server
  - Node - Consists of a set of Managed processes, managed via a Node Agent
  - Cell - Aggregation of Nodes. A Deployment Manager on the Cell controls and communicates with all the Node Agents.

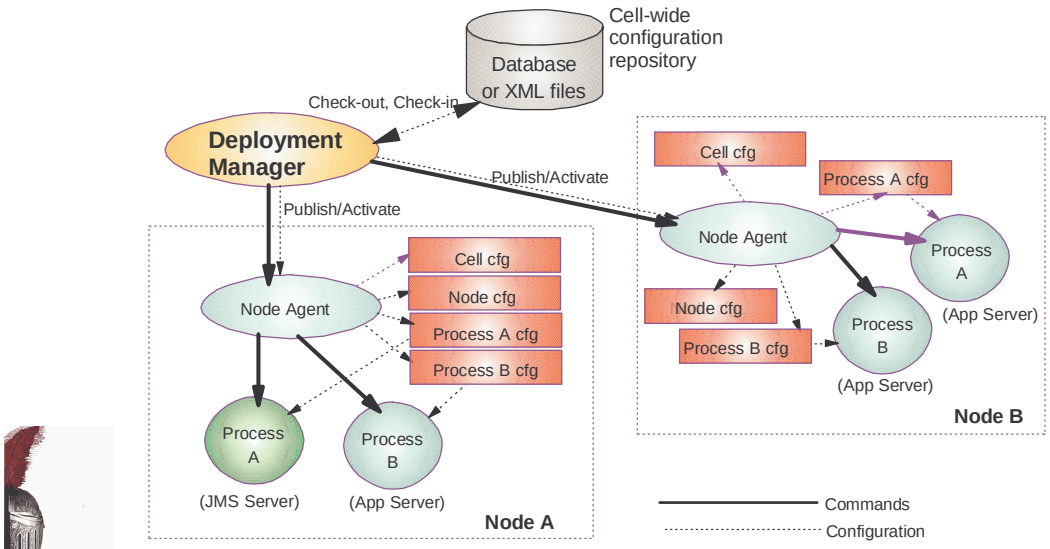


IBM Developer Relations dragonSlayer Team

Network Deployment allows you to manage multiple WebSphere Application Server, Version 5 nodes from a single, central location. You can install Network Deployment on any machine in the network to create a network deployment manager cell. The machine on which you install Network Deployment does not require WebSphere Application Server, Version 5 to be installed. Once you have installed and started the network deployment instance, you can use the addNode tool provided with WebSphere Application Server, Version 5 to add a WebSphere Application Server, Version 5 instance (node) to the network deployment cell. Once a node has been added to a network deployment cell, the network deployment manager for the cell assumes responsibility for the configuration of any application server nodes that belong to the cell. The network deployment manager creates configuration files for each WebSphere Application Server node which has been added to its cell.

A node is a set of managed servers on a physical machine in a topology composed of one or more machines. A node contains a WebSphere Application Server installation. A managed server is a single WebSphere Application Server JVM instance, running in its own process. A node cannot span multiple machines, but a machine can have multiple nodes, each with multiple managed servers.

Distributed WebSphere Administration Topology



IBM Developer Relations dragonSlayer Team

XML Document Repository- all configuration stored in collection of XML and XMI documents on the file system
no RDBMS

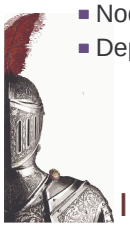
WebSphere Common Configuration Model (WCCM)
- Data-model representing the configuration of the system
- Documented API for manipulating WebSphere configuration files

Servers load directly off of documents

In a cluster, WebSphere manages synchronization of documents across machines
Application Binaries are managed as part of the repository

Configuration/Application Data

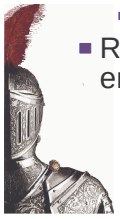
- Administration points within a Cell
  - Deployment Manager - manage everything under the cell - recommended
  - Node Agent - manage everything under the node, not the cell
  - Managed Process - manage the server process configuration, not the node or the cell
  - Can have the Admin client attach to any of the above process
- Deployment Manager contains the MASTER copy of the configuration/application files
  - Admin Client programs are used to modify configuration settings
  - Individual nodes and servers synchronize the files with the master configuration files (repository)
  - Only changes made at the cell level are permanent
  - Config changes made at the Node Agent or Server level are temporary
    - At next data update time, the master data is pushed down to the Nodes
- Deployment Manager checks in/out the configuration files from the master repository
- Node Agent receives updates of configuration/application data from
- Deployment Manager at each file synchronization



IBM Developer Relations dragonSlayer Team

When does synchronization occur ?

- File Synchronization settings can be set in the Admin Console
- If Automatic Synchronization flag is ON synchronization occurs:
  - Periodically. The time interval between the synchronizations can be set by the user.
  - When the Node Agent starts up and the Deployment Manager is already running
  - When the Deployment Manager starts up and the Node agent is running
- By default, Automatic Synchronization flag is on and the Synchronization interval is 60 seconds
- End user can force explicit synchronization
  - using Wsadmin
  - using Admin Console
- If Startup Synchronization is on
  - Refers to startup of an Application Server
- Recommendation: Increase the Sync interval in a production environment to reduce the overhead



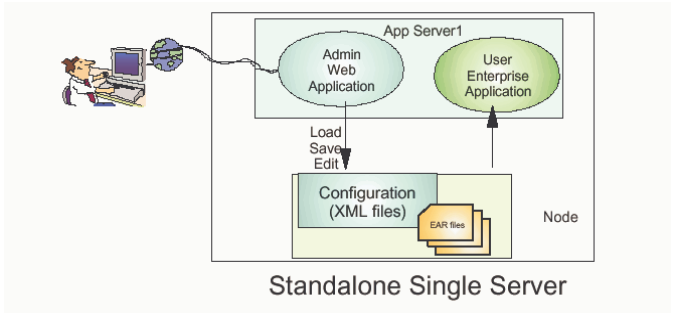
IBM Developer Relations dragonSlayer Team

Synchronization includes all:

- Configuration Files
- Application Binaries

WebSphere 5.0 Admin Console

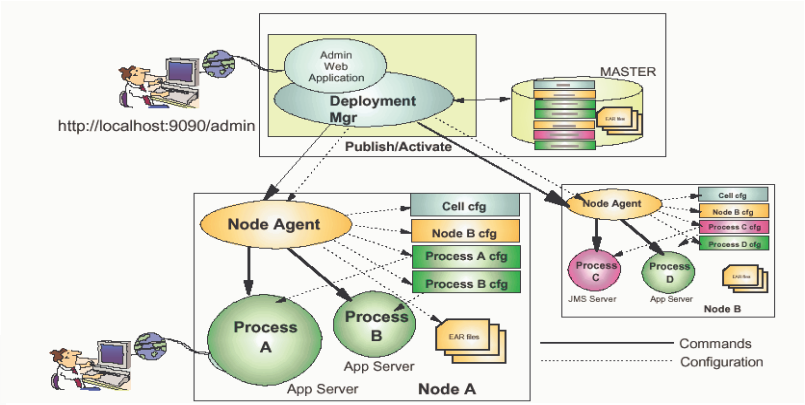
- The Admin Web Application runs within an Application Server instance on a node.
- Browser accesses the Admin Web Application
  - Can perform configuration/operational changes



IBM Developer Relations dragonSlayer Team

Admin Console in a distributed environment

- In a distributed environment, changes made to individual managed processes are local and will be overwritten at the next sync



IBM Developer Relations dragonSlayer Team

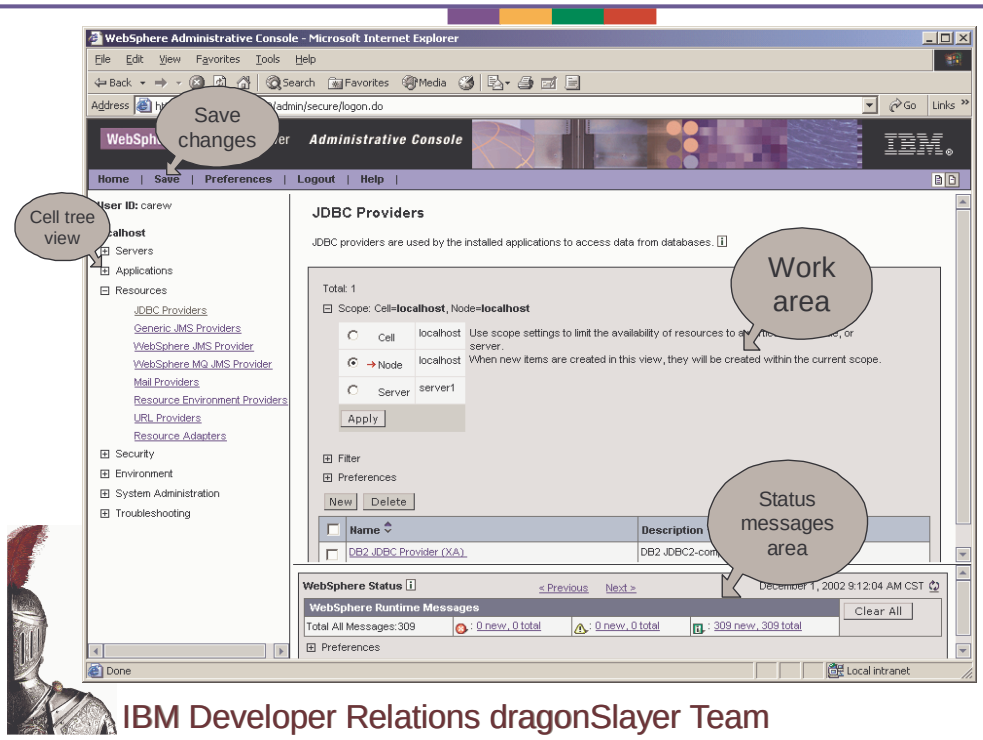
Starting the Admin Console

- Admin Console Application gets installed during WAS 5.0 installation.
- Start the server process on which the Admin Console application is installed
  - In the case of Base Application server, issue: startserver server1
- Access via web Browser "http://<hostname>:9090/admin/"
- No Password is required if the Global security is disabled
- User name is requested and is used to track and save user-specific configuration data.
  - Will be able to recover from unsaved previous session changes.
  - user Workspace : <was50-root>/wstemp/USER/workspace

IBM Developer Relations dragonSlayer Team

The user provided for the admin console when security is disabked does not have to be a valid user in any user repository

Admin Console



IBM Developer Relations dragonSlayer Team

The WebSphere administrative console is a graphical, Web-based tool that you use to manage the IBM WebSphere Application Server administrative server. The administrative console supports all product administrative activities.

The console has these main areas:
The taskbar , The cell tree view , The work area , The status messages area
You can resize these areas as desired.

The taskbar
The taskbar provides a way for you to return to the console Home page, to save changes that you have made to administrative configurations, to access product information and to log out of the console.

The cell tree view
You use the tree view on the left side of the console to survey, select, and manage components in a WebSphere administrative cell.

The workarea
The console workarea provides links to pages that provide step-by-step instructions on installing application, updating applications, and creating application servers. The console also provides a search function for locating and viewing resource objects and some information about adjusting the console to meet your needs.

The status messages area
The messages area at the bottom of the console lists messages returned by the WebSphere Administrative Server as well as messages about events such as successful completion and fatal errors.

Scripting with wsadmin

- New scripting interface for WebSphere Application Server v5.0, called "wsadmin"
- wsadmin offers a number of Advantages
- Same scripting tool for all versions of WebSphere v5.0
- Based on the Bean Scripting Framework (BSF)
- Current supported languages for wsadmin - more to come
  - jacl - Java Command Language based on Tcl scripting
- Robust scripting features and programming model very similar to Java
- All admin console functionality is available using wsadmin

IBM Developer Relations dragonSlayer Team

wsadmin is the new scripting interface for WebSphere replacing WSCP and XMLConfig. It is based on the Bean Scripting Framework which offers a number of advantages. The first one being that the same scripting interface will be available for all versions of WebSphere. This is different from WebSphere v4.0 where WSCP and XMLConfig were only available with WebSphere Advanced Edition (AE) and SEAppInstall was only available in WebSphere Advanced Single Server Edition (AEs).

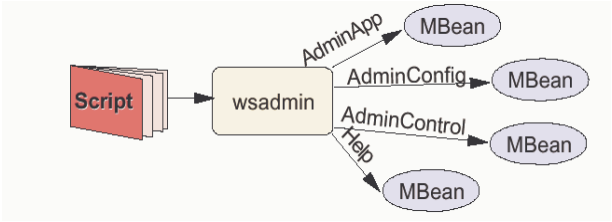
For WebSphere v5.0, Jacl will be supported, with other languages supported in future versions. Jacl is based on the scripting language Tcl. More information about Tcl is available at http://www.tcl.tk/scripting/.

Some of the major advantages of using Jacl and Wsadmin is that Jacl offers may robust features and prebuilt functions. The programming model for Jacl is easy for Java developers to use and allows customers to utilize their existing investment in scripting skills.

In WebSphere v4.0 there was limited support for some of the Tcl commands. With wsadmin, there is full support of all the Tcl commands.

How wsadmin works

- wsadmin acts as an interface to Java objects for access by scripts
  - Objects communicate with MBeans (JMX management objects)
- Objects perform different operations
  - AdminConfig
    - Create or change the WebSphere configuration
  - AdminApp
    - Install, modify, or administer applications
  - AdminControl
    - Work with live running objects and perform traces and data type conversion
  - Help
    - Display general help information and details about which MBeans are running



IBM Developer Relations dragonSlayer Team

The JMX specification defines an interface called MBeanServer for communicating with MBeans -- the scripting interface called AdminControl is based on this interface.

wsadmin uses the same interface as a user would using the Web Console to make configuration changes and control the WebSphere server.

With the different commands available for wsadmin there is a clear distinction between working with configuration settings and objects running on the server. This distinction should ease the understanding of what is being changed when working with wsadmin.

ANT support in WebSphere 5.0

- WebSphere 5.0 provides the following ANT tasks
  - wsInstallApp task enables you to install a new application into a WebSphere Server or Cell.
  - wsUninstallApp task enables you to uninstall an existing application from a WebSphere Server or Cell.
  - wsListApps task lists all the applications installed on a WebSphere Server or Cell.
  - wsStartServer task enables you to start a standalone server instance.
  - wsadmin task executes the WebSphere command-line administration tool with the specified arguments.
  - wsejbdeploy task executes the WebSphere EJB Deploy tool on the specified Jar file with the specified options.

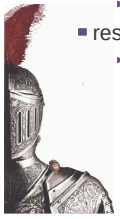
IBM Developer Relations dragonSlayer Team

Ant is a scripting tool that lets you construct your build scripts in much the same fashion as the "make" tool in C or C++. Ant is a subproject of the Apache Jakarta project, part of the Apache Software Foundation. You can use a large number of built-in tasks in Ant without any customization. Because Ant is Java-based, it is platform-independent. It is well suited for building Java applications, but can be used for other build tasks as well. One of its important features is that you can use Java to write new Ant tasks to extend production build capabilities.

ANT tasks can be used to compile code, create EJB JAR files, WAR files and EAR files. ANT can also be used to perform tasks such as generating deployed code for your EJBs , installing an application , launching a client application and running JUnit tests.

Other Command line tools

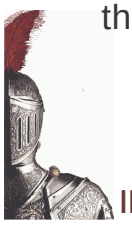
- dumpNameSpace
  - Displays JNDI namespace and entries
- ivt (Installation Verification Tool)
  - Syntax: ivt <hostname> <port#> (ivt myhost 9080)
- JspBatchCompiler
  - Pre-compile JSPs in a web module
- mb2mdb
  - Convert a WAS 4.0 EE message bean into a message driven bean
- tperfviewer
  - Tivoli Performance Viewer (previously known as Resource Analyzer)
- syncNode
  - Forces synchronization between the node and the Deployment Manager
- versioninfo
  - Provides IBM WebSphere Application Server Version Report. Syntax:- versioninfo.bat
- serverstatus
  - Retrieves server status. Syntax:- serverStatus <server name>
- backupConfig
  - utility to back-up configuration of your node to a file
- restoreConfig
  - utility to restore the configuration of your node after backing it up using the backupconfig command



IBM Developer Relations dragonSlayer Team

Summary

- Application assembly can be done with AAT or Application Developer
- WebSphere 5.0 has a distributed architecture for application binaries and configuration data
- WebSphere 5.0 has 2 ways to modify configuration data
  - Via Web based admin console
  - Via a scripting interface
- WebSphere 5.0 supports ANT by providing a series of tasks that interface with the scripting environment



IBM Developer Relations dragonSlayer Team