Introduction to JSP Development

using NetBeans, CVS, and SQLTags

Objectives

- Students should have a good understanding of the development tools and how they fit into the overall development process.

- Upon completion, students should be familiar with JSP development concepts and be prepared to begin developing JSP web pages.

Experience

- Your experience?
- Java, JSP?
- ASP, CF, PHP, Other?
- Object Oriented?
- HTML, JavaScript?

Outline

- Lesson 1- Overview
- Lesson 2- Introduction to JSP (optional)
- Lesson 3- NetBeans IDE
- Lesson 4- Concurrent Versioning System
- Lesson 5- SQLTags

Lesson 1- Overview

The Big Picture

Lesson 1- Overview

- Objective: to gain an understanding of the overall development process and become familiar with specific development tools.

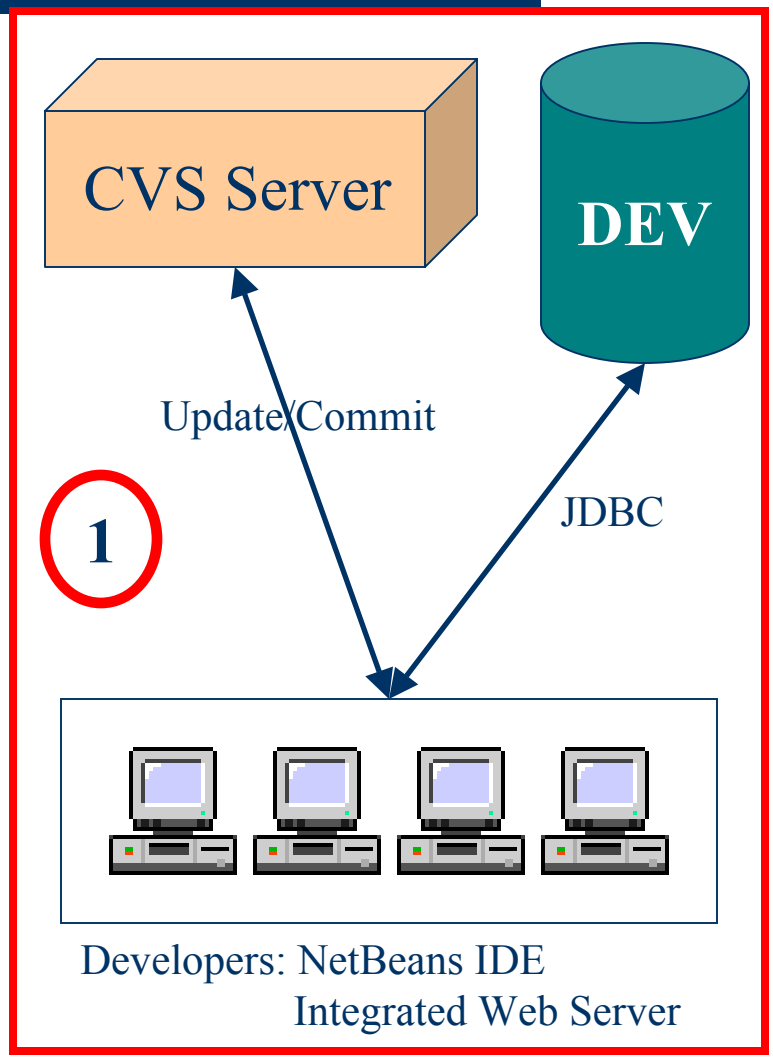
- Understand how the individual components fit into the overall process

Lesson 1- Overview

- Process
  - Development
    - CVS Checkout/Update
    - Modify/Test Pages
    - CVS Commit
  - Test
    - CVS Tag
    - CVS Export
  - Production
    - CVS Export
    - Possible Branching
- Tools
  - Development
    - NetBeans IDE on each workstation
    - 1 CVS & DB Server
  - Test
    - 1 Web & DB Server
  - Production
    - Web Server(s)
    - DB Server(s)

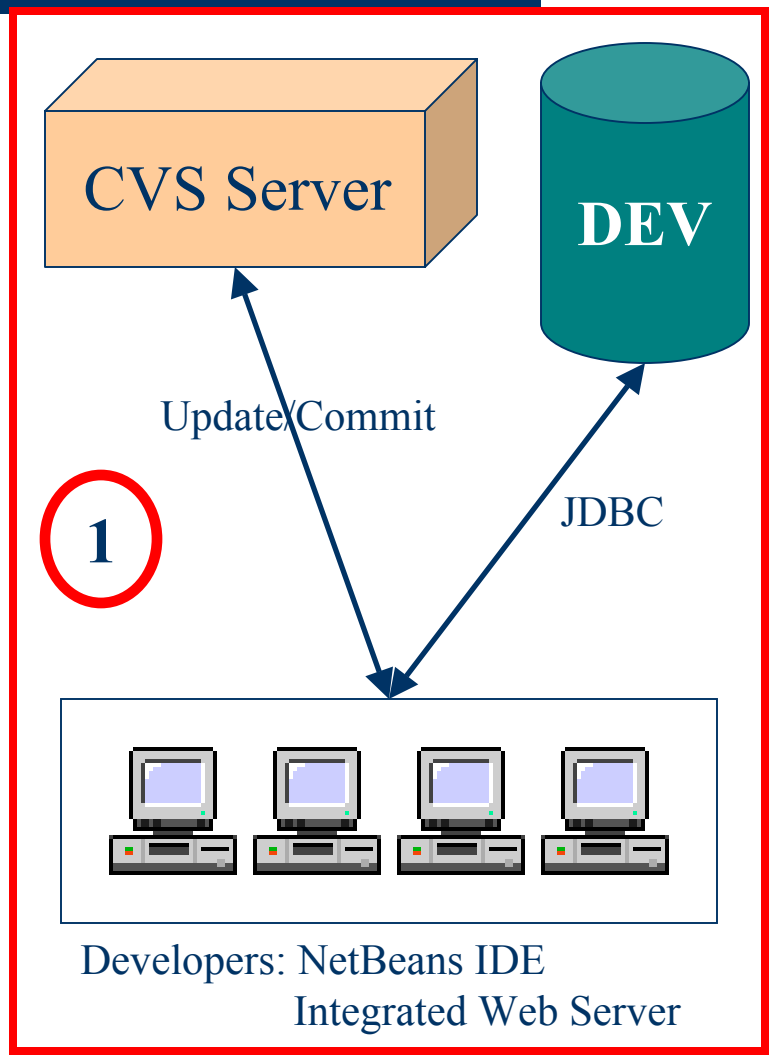
Development Tools

- NetBeans IDE on each desktop
  - Integrated web server, editor, and debugger included in NetBeans
- CVS* Repository on one server
  - * or External Version Control System like VSS
- Development database server

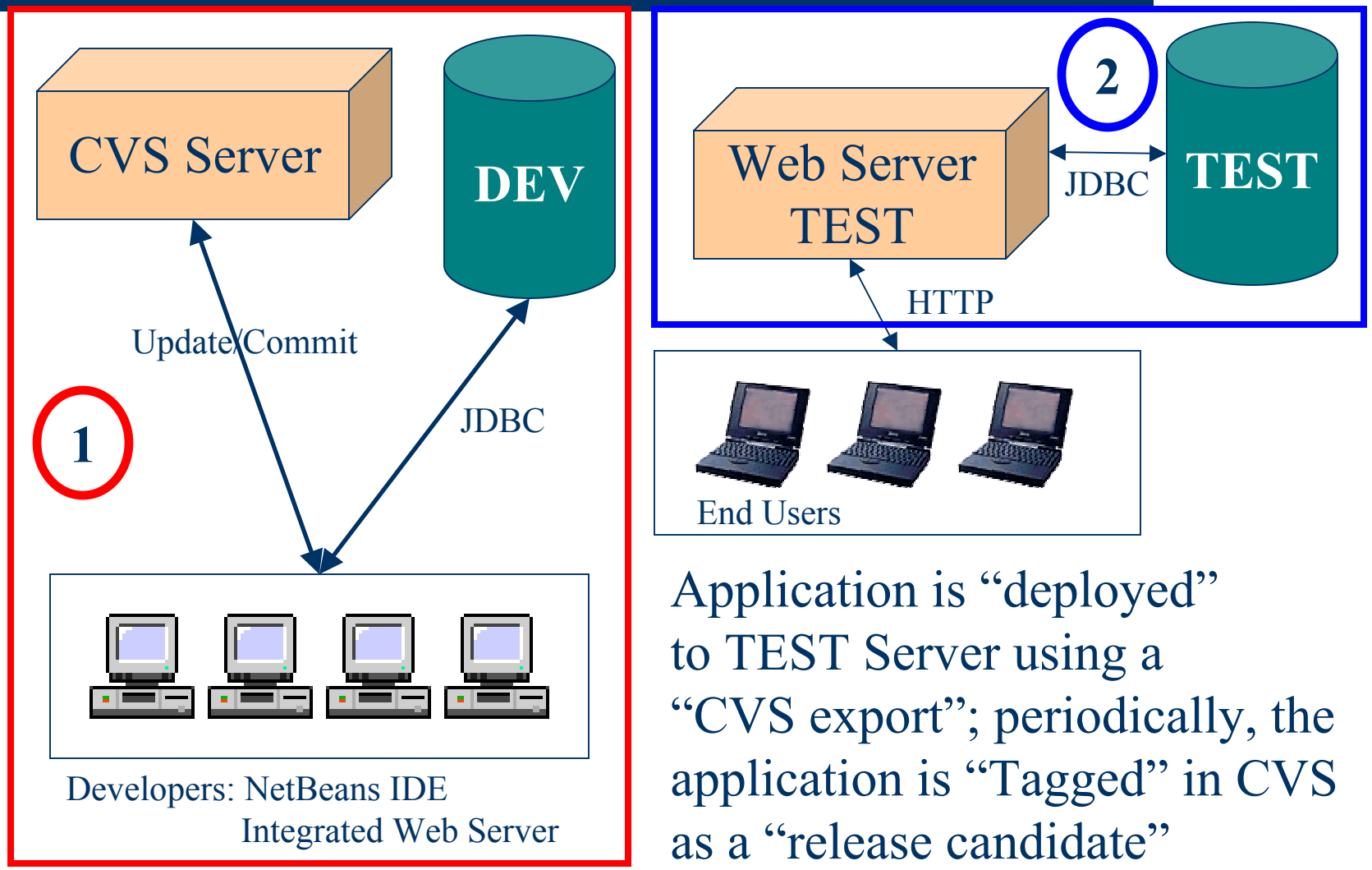


Development Process

- NetBeans- Project setup
- CVS Checkout/Update
  - Copy of entire application
  - Checks for conflicts
  - Updates local copies
  - Gets “clean” copies (oops)
- Modify & Test Pages
  - (Do what developers do)
- CVS Commit
  - posts changes to CVS

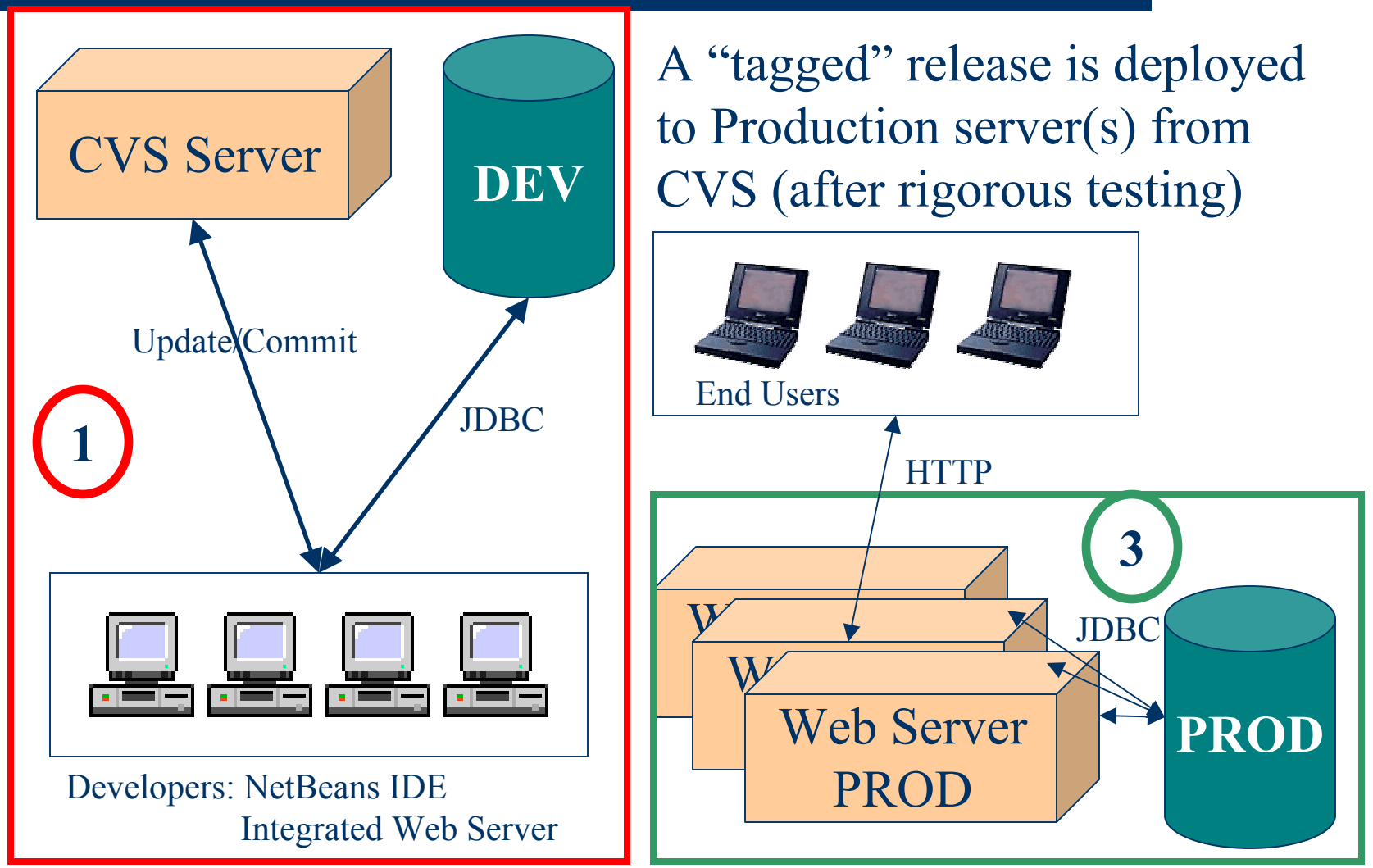


Overall Configuration- Test



Application is "deployed" to TEST Server using a "CVS export"; periodically, the application is "Tagged" in CVS as a "release candidate"

Overall Configuration- Production

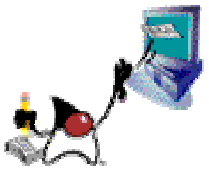


Lesson 1- Review

- Process
  - Development
    - CVS Checkout/Update
    - Modify/Test Pages
    - CVS Commit
  - Test
    - CVS Tag
    - Integration testing
  - Production
    - Support live application
- Tools
  - Development
    - NetBeans IDE on each workstation
    - 1 CVS & DB Server
  - Test
    - 1 Web & DB Server
  - Production
    - Web Server(s)
    - DB Server(s)

Questions?

Lesson 2- JSP Introduction



JAVASERVER PAGES™
DYNAMICALLY GENERATED WEB CONTENT

Lesson 2- JSP Introduction

- Objective: Upon completion, student should have basic understanding of JSP and be prepared to develop simple JSP pages

- Reality Check: **Show of Hands**
  - What is your JSP Experience Level?
    - Expert
    - Experienced
    - Beginner
    - What is JSP, anyway?

JSP Overview

- JSP looks similar to ColdFusion or Active Server Pages (ASP)
- Basically, JSP pages are HTML pages with:
  - limited “Snippets” of Java code,
  - calls to JavaBeans, and
  - Custom Tags.
- JSPs are maintained within a “Web Module” in NetBeans

JSP is Servlet

- JSP is Servlet
  - When a JSP page is hit for the first time, it is automatically compiled into a servlet.
  - The Servlet is then available to web server
  - Servlets have built-in application, session, request, and page variables

JSP Built-in “Scopes”

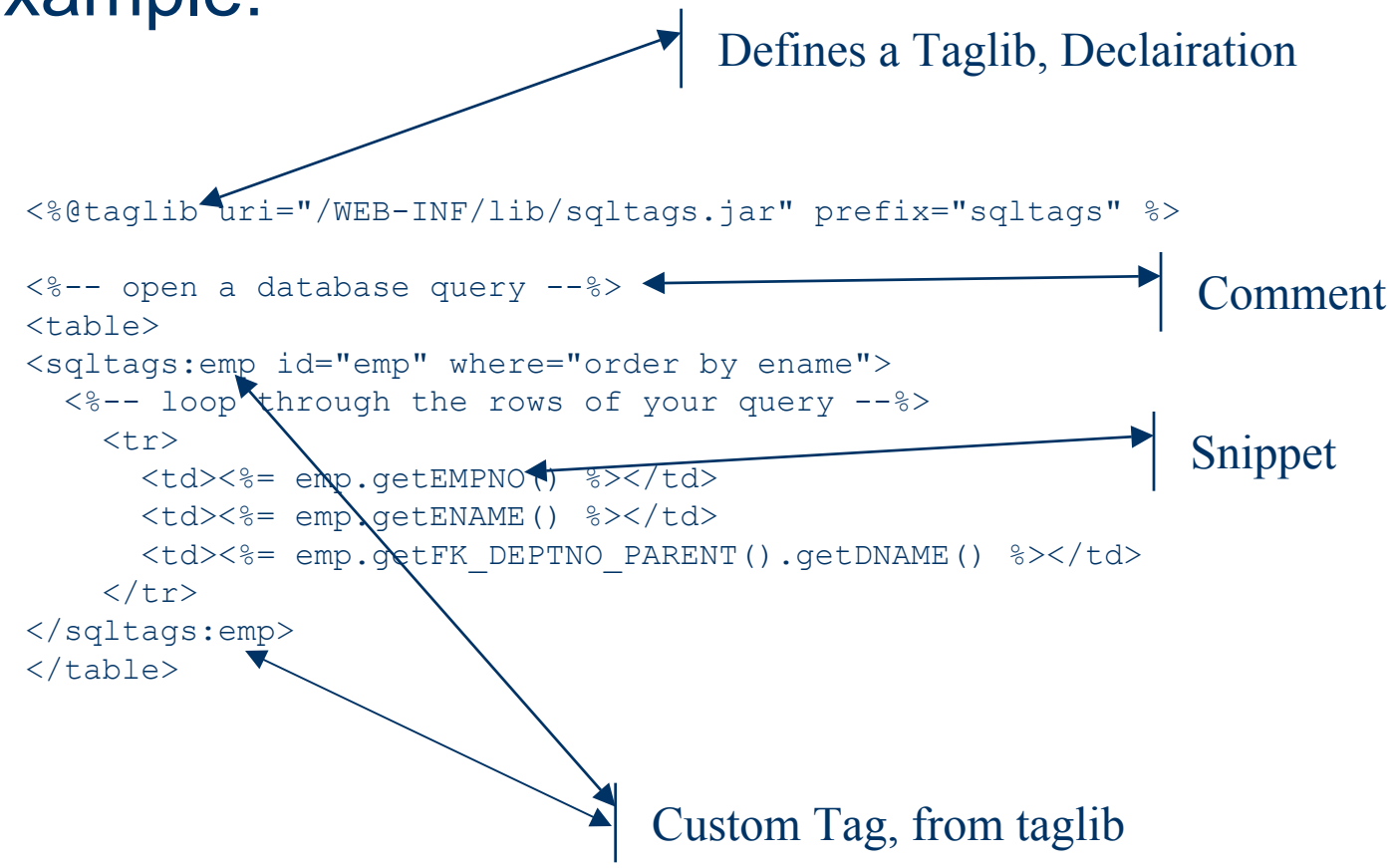
- Page
  - Available within one page, not available to forwarded or included pages
- Request
  - Available to single user for duration of one “page hit”
  - Contains “CGI” Vars
    - `getParameter()`
    - `getRemoteUser()`
  - Is Available to forwarded and included pages
- Session
  - Available to single user from page-to-page until session time-out
- Application
  - Available to and shared by all users
  - initialized once, each time web server is started.
- User Defined Attributes
  - available to all scopes
    - `setAttribute()`
    - `getAttribute()`

Components of a JSP Page

- Declarations
  - `<%@ ... %>`
  - Standard headers
- Snippets
  - `<% ... %>`
  - `<%= ... %>`
- Comments
  - `<%-- ... --%>`
- JavaBeans
  - useBean
  - scope
- Tags, Tags, Tags
  - Built-in
  - Custom
- And, of course, HTML

JSP & Tag Example

- Example:



JSP Declarations

- Declarations
  - <%@ … %>
  - Like #includes within a “C” program
    - imports
    - content-type of the output page
  - Typically, declarations are the same for most every page. A Template can be used.
  - Example:

```
<%@import “org.sqltags.*” %>
<%@ taglib uri="tags.jar" prefix=”my" %>
```

JSP Snippets

- Snippets
  - Most coding is done within snippets; however, avoid excessive use.
  - A block of Java code is introduced using the "<%" syntax. Must be complete Java statement.
  - **<% … %>**
  - In-line expressions are introduced with the "<%=" syntax. Is simply a valid "string" expression.
  - **<%= … %>**

- Examples:

```
<% String s = new String( "Steve" );
   out.println("name is "+s);
%>

id is <%= request.getRemoteUser() %>
```

- In-line expressions are simply stuffed into a "print" statement.

JSP Comments

- JSP Comments that are hidden from the browser

```
<%-- this is a comment, the comment
   continues until the ending
   syntax, this text will not get
   to the browser
 --%>
```

- HTML Comments
  - passed on to browser

```
<!-- html comment -->
```

Web Module

- A Web Module contains “jsp” files plus the WEB-INF directory structure.
  - WEB-INF/
    - classes/
    - lib/
- CLASSPATH typically contains:
  - WEB-INF/classes/
  - WEB-INF/lib/*.jar
- JavaBeans, Tags installed under WEB-INF

JSP JavaBeans

- JavaBeans
  - Basically, a JavaBean is a Java Class that follows some simple conventions.
  - Provides a default (zero argument) “constructor”
  - Provides access to Class Variables with “setter” and “getter” that follow a well-defined pattern:
    - Class Property: “myName” would be accessible as:
      - `void setMyName(String myName)`
      - `String getMyName()`

JSP JavaBeans

- Hide Java code from JSP page, and that's good!
- Are reusable
- Bean Scope
  - Request
  - Session
  - Application
- Really Cool!
- Nice way to build "utilities" for many tedious tasks
- AVOID embedding HTML within your JavaBeans
- AVOID embedding excessing Java within JSP pages

JSP JavaBeans

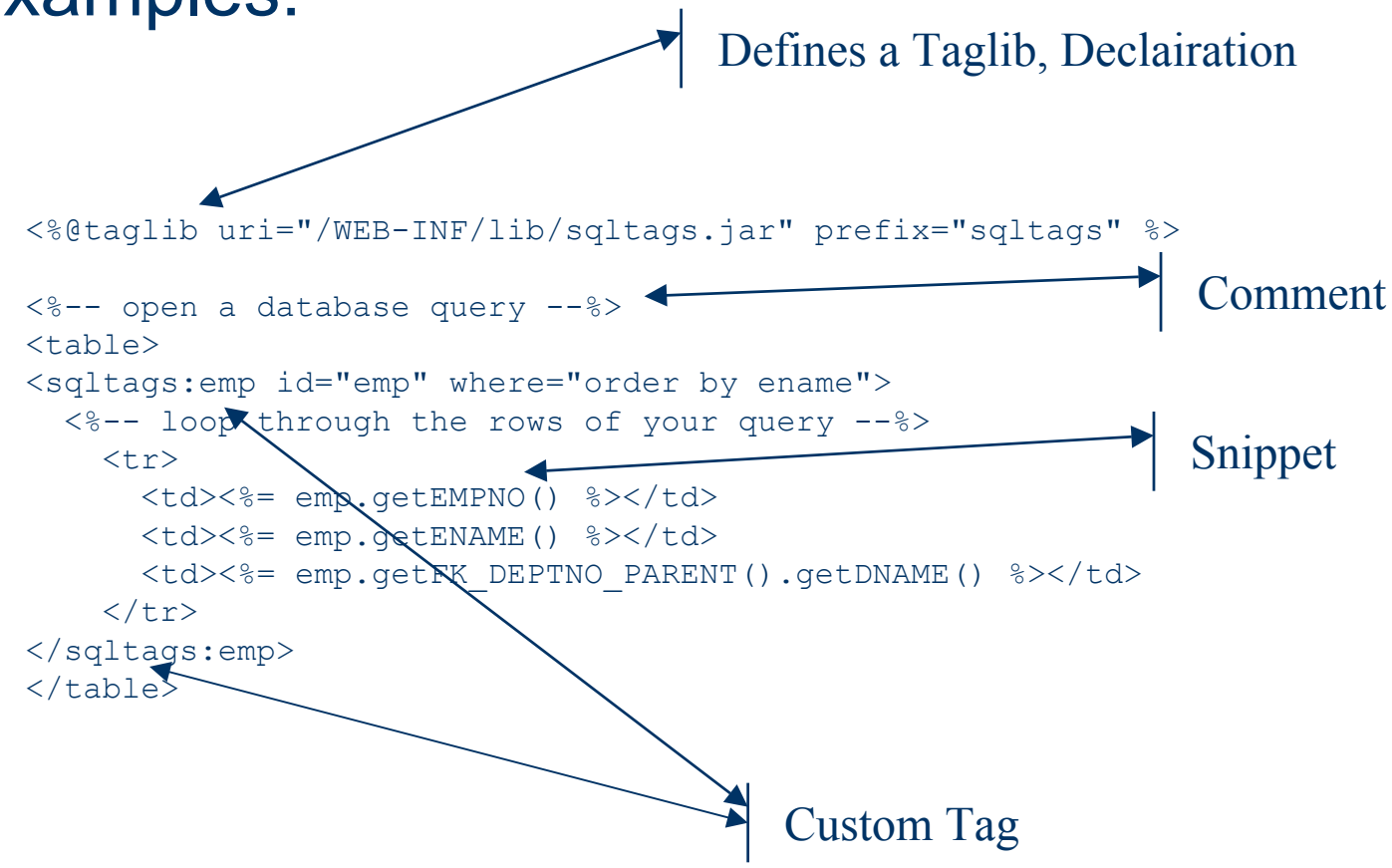
- Define a project under your official domain:
  - `com.yourdom.projx.beans`
  - `WEB-INF/classes/com/yourdom/projx/beans`
- Invoked by <jsp:useBean/>
- Scope controls how long the Bean “lives”
  - Page, **request** (default), session, or application
- Calls the default “constructor” and uses “reflection” to access class variables
  - getName() - returns variable “name”
  - setName() - sets the “name” variable
- Also, all “methods” are available to JSP page

JSP Tags

- JSP natively supports custom tags
  - Another way to "hide" Java code from JSP.
    - `<jsp:useBean\>`
    - `<jsp:forward\>`
- Many free taglibs are available
    - SQLTags
    - Mailer Tags
    - Java Standard Tag Library (JSTL)
- Similar to JavaBeans
- Custom Taglibs
  - A "TLD" maps custom tags to their supporting Java classes
  - Supporting Classes define behavior of:
    - doStart
    - doAfterBody
    - doEnd
  - The "tag body" is evaluated repeatedly until SKIP is returned

JSP & Tag Example (Review)

- Examples:



JSP Hints

- Helpful hints
  - You'll be mostly using a bit of snippets, perhaps a few JavaBeans, and Tags (for JDBC access).
  - The "request object" and "request parameters" also critical for obtaining "form inputs."
  - Again, AVOID HTML in JavaBeans
  - AVOID "excessive" Java within the JSP
  - SQLTags will provide easy database access

JSP Review

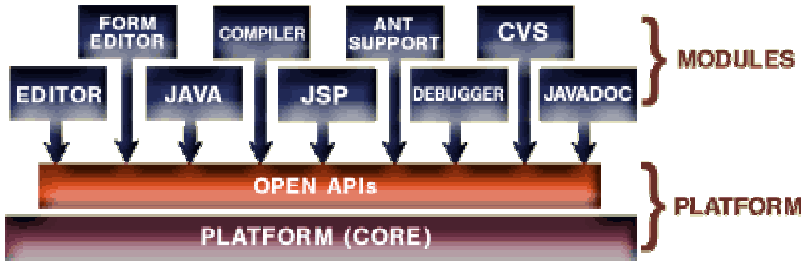
- JSP looks similar to ColdFusion or Active Server Pages (ASP).
- Basically, JSP pages are HTML pages with limited “Snippets” of Java code, calls to JavaBeans, and Custom Tags.
- JSPs are maintained within a “Web Module” in NetBeans

- Questions?

Lesson 3- NetBeans IDE

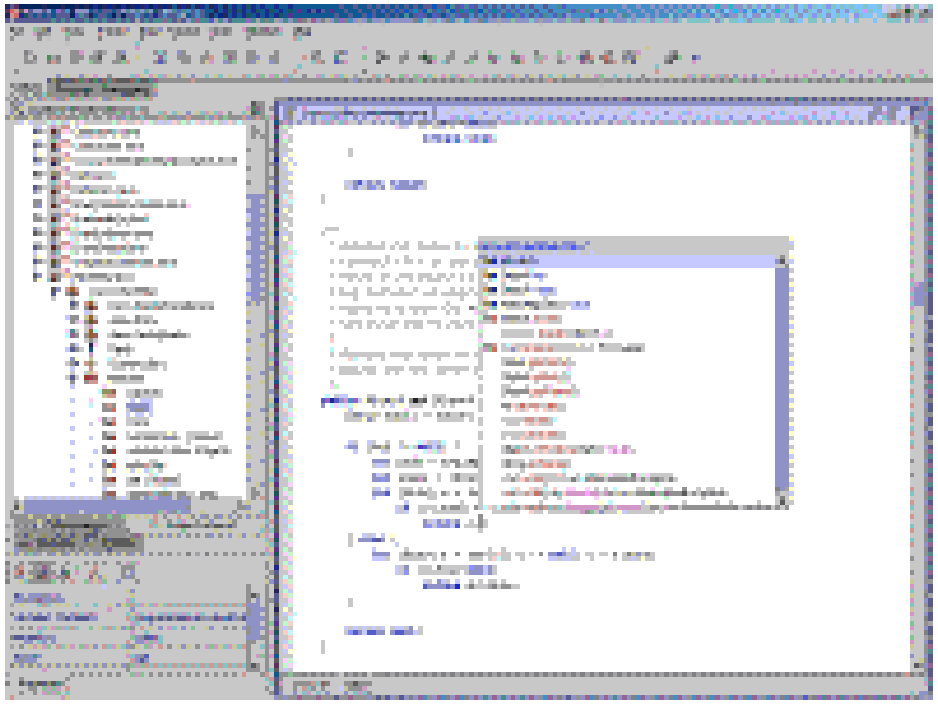
Lesson 3- NetBeans IDE

- Objective: to be introduced to the NetBeans Integrated Development Environment.

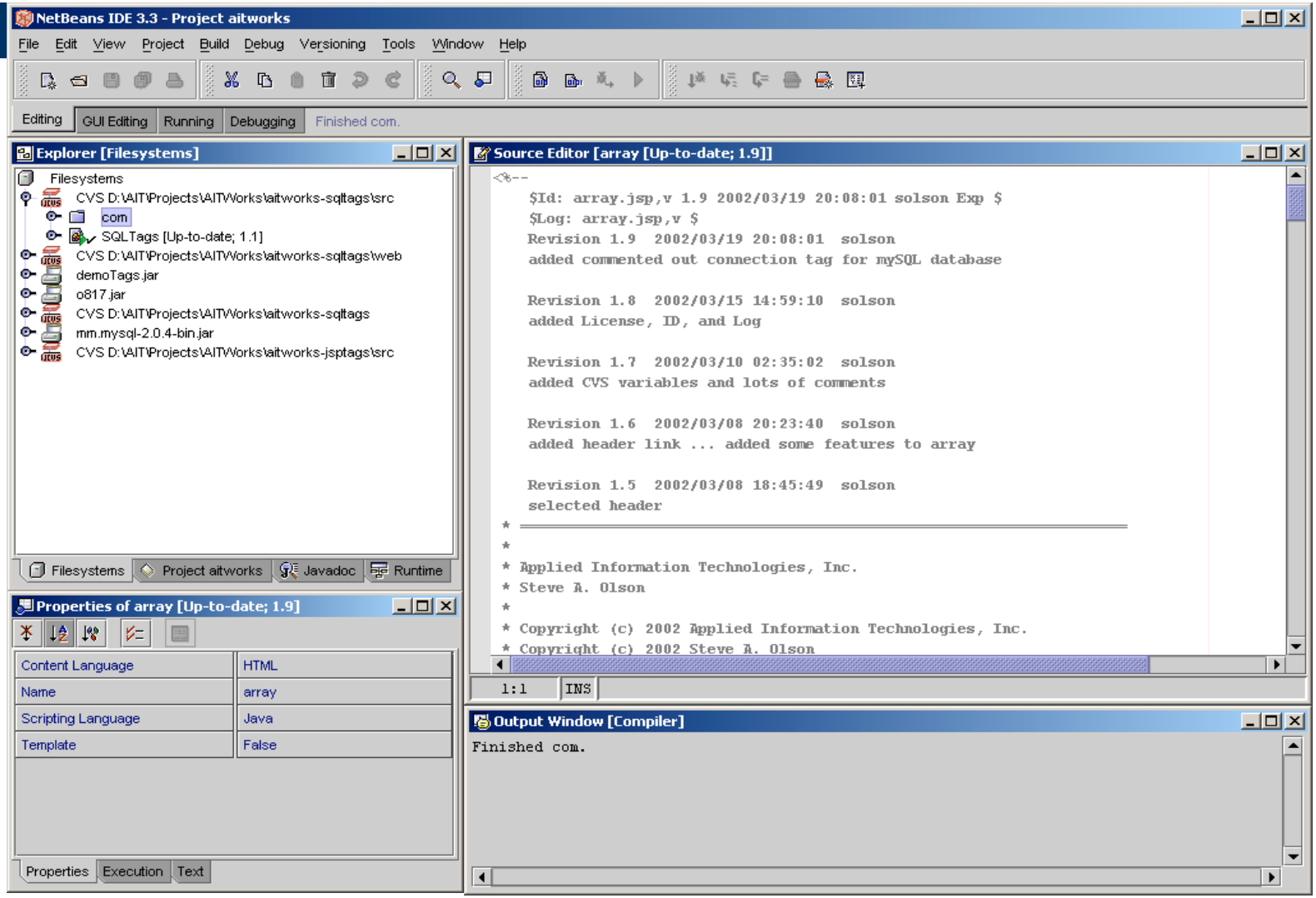


NetBeans IDE- Overview

- Project Manager
- Mounts
- Web Modules
- CVS Integration
- Work Spaces
- Debugger



NetBeans- Screen shot Editing



NetBeans- Projects

- Project Manager
  - Allows developers to switch between different projects
  - Allows developers to switch between different versions of the same project
  - Defines location for local files
  - “Remembers” mounts
  - Controls Java CLASSPATH

NetBeans- Mounts

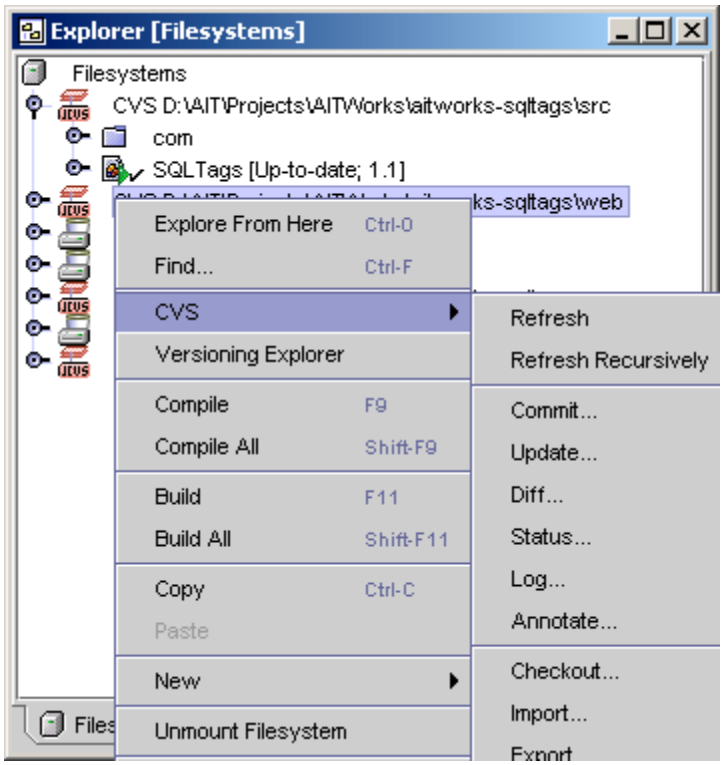
- Manages files within projects
- Mount Types
  - Local Directory
  - CVS Repository **
  - JAR file
  - FTP Server
- CVS Mount supports built-in version control
- Each Mount is added to CLASSPATH
- for Web Modules
  - WEB-INF/
  - WEB-INF/classes
  - WEB-INF/lib/*.jar
    - Also added to CLASSPATH
- Add required JAR files to WEB-INF/lib

NetBeans- Web Modules

- JSP development is accomplished within a Web Module
- Basically, Web Modules are defined by the WEB-INF directory structure
- Implications to CLASSPATH
- WAR tool for deployment
- Supported by Built-in Web Server

NetBeans- CVS Integration

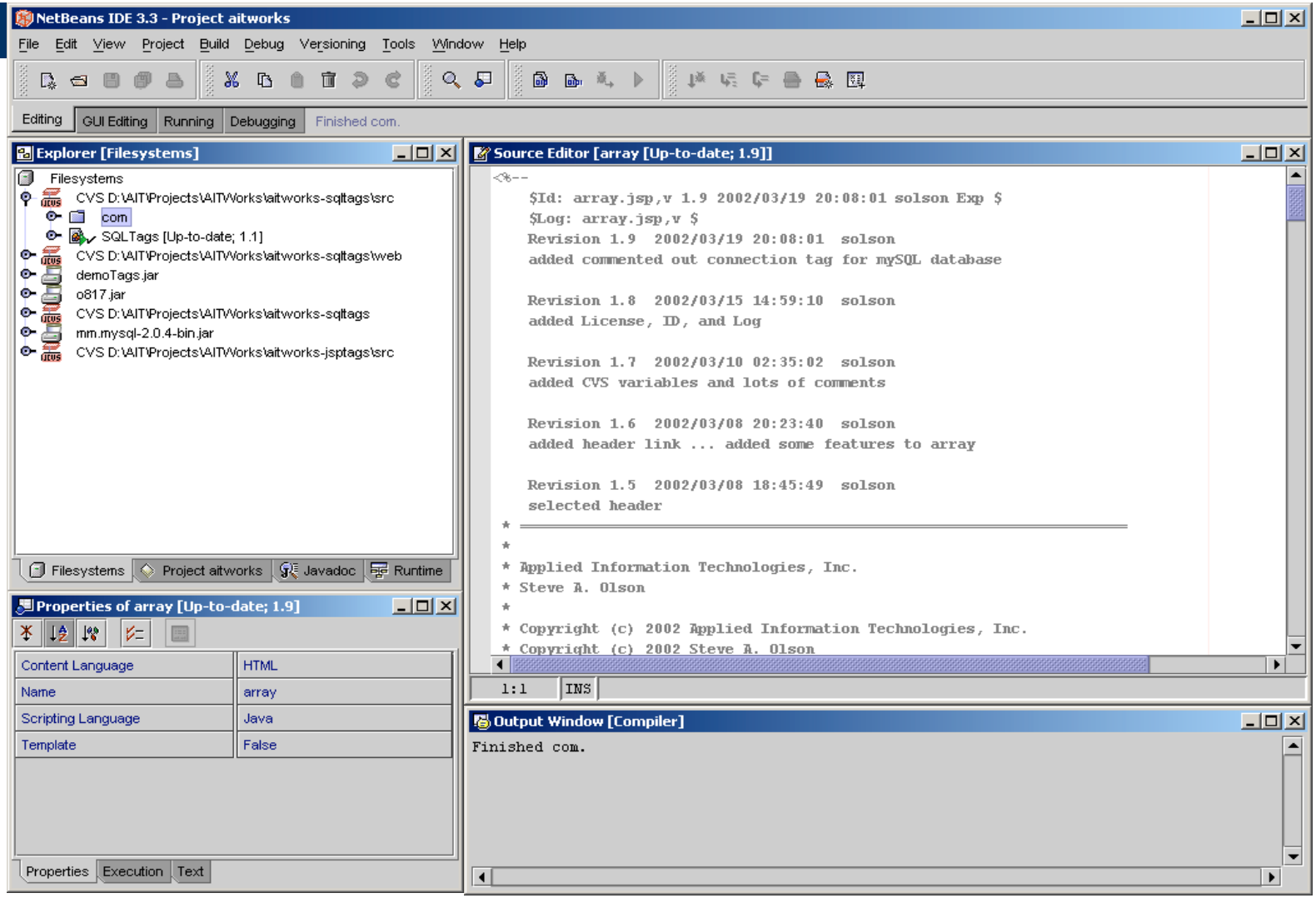
- Within any CVS Mount
- Right-click an item
- Options:
  - Refresh
  - Commit
  - Update
  - Diff
  - …



NetBeans- Work Spaces

- Three main work spaces; however, they are configurable
  - *Editing* - Explorer, Editor, Output, Properties
  - *GUI Editing* - not needed for Web Mods
  - *Debugging* - Watches, Editor/Debugger, Output
- Each work space controls a set of windows specific for the task
- Hint: Switching brings back windows

NetBeans- Screen shot Editing



NetBeans- Editing/Running

- Double-click to Edit
- Right-click -> Execute to run a JSP page
- Right-click-> Execute- (force reload) to run JSP page and restart Tomcat Server (changes to server configuration)
- Right-click-> Compile to compile a JSP page

- Execute:
  - NetBeans will save and compile the JSP page
  - Then Launch the Tomcat Servlet engine
  - Then Launch the Browser with appropriate URL to access the page
  - Server messages displayed in “output” window.
- Compile
  - Errors are displayed in “output” window
  - Double-click on errors to edit code

NetBeans- Debugger

- Select a JSP page from the Explorer window
- Then select
  - Debug->Start
  - from the main menu
  - The page will execute within the debugger
- Break points and Watches can be set within work space

Options Include:

- Single Step
- Step Into
- Run to Cursor
- Run to Break Point
- Finish

Mouse Over shows variable's values

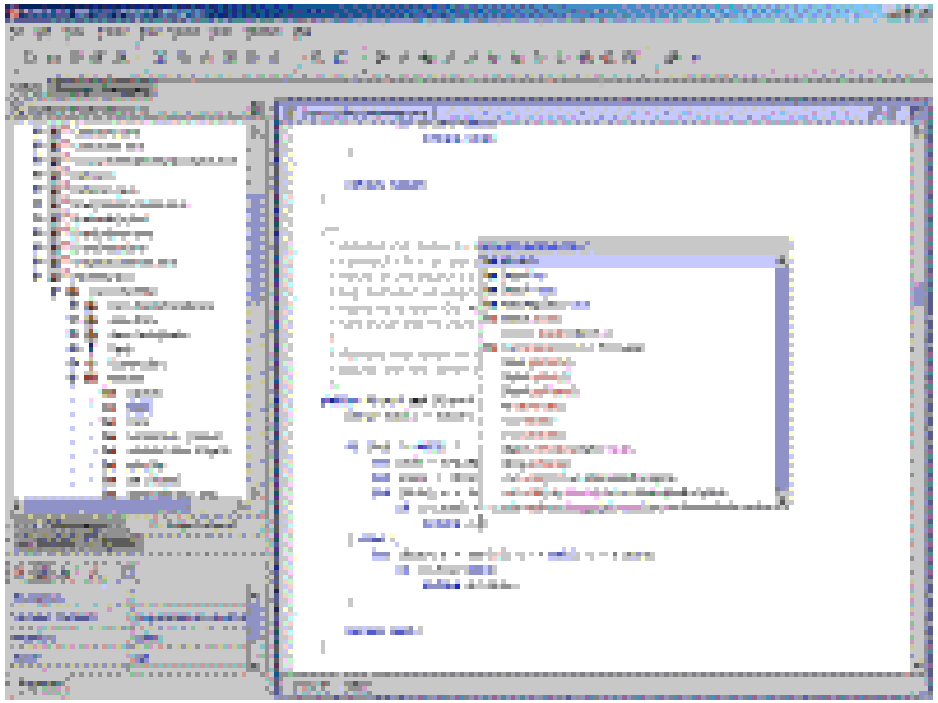
Stack viewer

More …

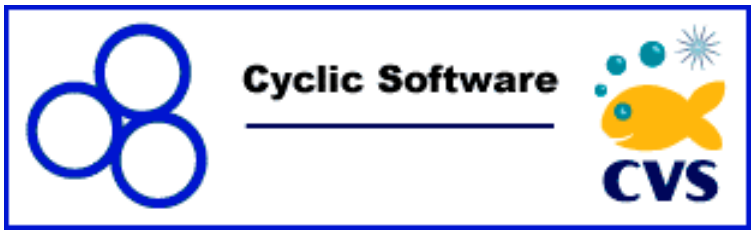
NetBeans IDE- Review

- Project Manager
- Mounts
- Web Modules
- CVS Integration
- Work Spaces
- Debugger

- Questions?

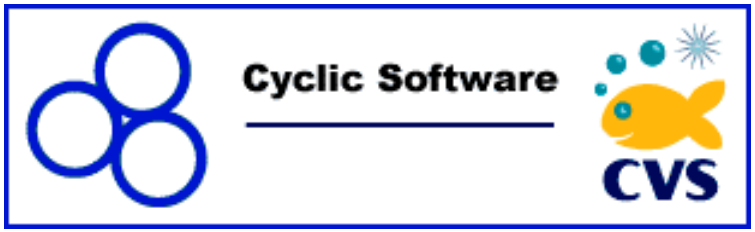


Lesson 4- CVS



Lesson 4- CVS

- Objective: to be introduced to the Concurrent Versioning System.
- Highlighting main features, only.



CVS Overview

- Repository Setup
- Checkout
- Update
- Add
- Remove
- Commit
- Tag

CVS Background

- CVS does not provide for the traditional “checkout” where a file is locked by one developer, instead all files are writable
- Developers periodically refresh their local copy with “update” command
- Modified Files are sent to CVS using the “commit” command
- Conflicts are possible and must be managed and merged

CVS Repository Setup

- Repository Setup
  - Unix Server is Recommended (not required)
  - Create “owner” account
  - run “cvs init”
  - Modify
    - CVSROOT/passwd - defines users and passwords
    - CVSROOT/cvswrappers - defines binary files
  - Import initial files, if necessary
  - Configure `/etc/inetd.conf`

CVS- Checkout

- CVS checkout
  - Only used to initialize local copy
  - Gets latest version of every file
  - Can be used to checkout a “branch” or “tag” for maintenance of previous versions
  - All update, diff, commit commands are “relative” to the checkout

CVS Update

- CVS Update
  - Update often
  - Update “Get Clean Copy”
    - Throw away local changes and overwrite with latest copy from repository
  - Always update just prior to “commit”
  - If in doubt, update!

CVS Add

- CVS Add
  - Adds a new file or directory to the Repository
  - Be careful with “Binary” files, use -kb (binary) option
  - Be sure to “commit” after Add or file will not be available

CVS Remove

- CVS Remove
  - Removes a file from Repository
  - Must delete file prior to remove, or select option to delete file from NetBeans
  - Be sure to “commit” the remove or file will stay in the Repository

CVS Commit

- CVS Commit
  - Commits a change to the Repository based on the local copy
  - Changes, Adds, and Removes must all be “committed” before they will take effect
  - Be sure to “update” prior to “commit” to deal with “conflicts” if they occur

CVS Tag

- CVS Tag
  - Marks all files within the Repository with the given name
  - Really, it just adds an entry within each file mapping the current version to the tag name
  - Used to “define” a version of the application as a whole so that it can be reconstructed later
  - Allows for easy maintenance of legacy versions without stalling progress on main development trunk-- very powerful!

CVS Review

- Repository Setup
- Checkout
- Update
- Add
- Remove
- Commit
- Tag

Questions?

Lesson 5- SQLTags

Lesson 6- SQLTags

- Objective: to be introduced to the SQLTags taglib.
- Highlighting main features, only.

What is SQLTags?

- SQLTags is a free, open source, **object-relational mapping toolkit** and "development framework" that provides a new and innovative approach to data-driven web applications development.

- At the SQLTags core is a **generator that builds a JavaBean and a JSP tag for each table** within a specified database schema. Additionally, a set of built-in JSP tags and run-time support classes facilitate database development within JSP.

- The generator packages everything into a **single "Java Archive" or "jar" file** for easy deployment into your Web Application.

- **SQLTags is Independent** of any specific J2EE Application Server or Database avoiding vendor "lock-in" at either level.

- **SQLTags is quick and easy to learn**.

How Does SQLTags Work?

- The Generator gets input from the user and generates a “Java Archive” or JAR file based on JDBC database metadata.
  - JDBC Driver, username, password, schema info
  - Jar output filename, options …
- The “jar” file, taglib, and “dataSource” mappings are added to the deployment descriptor.
  - Web.xml: taglib directive, dataSource, options ...
- Finally, JSP pages are authored using the taglib.

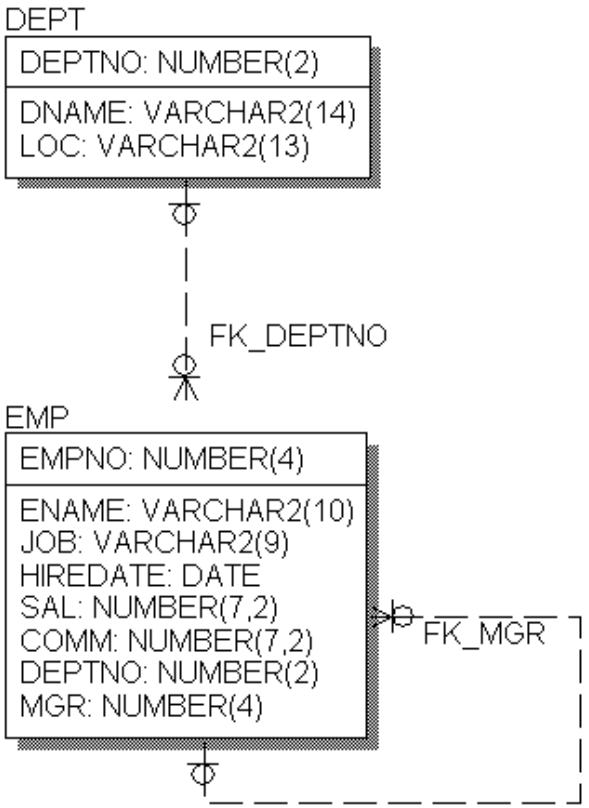
Why Use SQLTags?

- Eases building, maintaining data-driven JSP pages.
- Eases impact of database change
  - Select, Insert, Update, and Delete statements are generated (based on database metadata).
  - Simply re-run generator whenever database is changed.
- Supports Java and JSP industry standards like:
  - JavaBeans, JSP Standard Tag Lib (JSTL), Struts, JSP 2.0.
- Works well with EA Server (Jaguar)
  - Easily deploys to EA Server within Web Application (WAR).
  - Can use Container's Pooled Data Sources and Roles.
- Not specific to EA Server, any JSP 1.2 container

Why Use SQLTags? (con't)

- Faster JSP Development
  - Automatically associates "request parameters" to columns
  - Built-in Data Manipulation Language (DML) methods:
    - Insert, update, delete
    - Built-in array processing for multi-row updates
  - Automatic iteration based on "where clause" or "array params"
  - Built-in paging with next, previous tags
  - Built-in access to container's security using allow, deny tags
  - Supports parent-child nesting of tags based on foreign keys
- Quick and easy to learn.
- One line of JSP eliminates lots of tedious work
- Shorter development cycle, more productive.

SQLTags Mapping Example

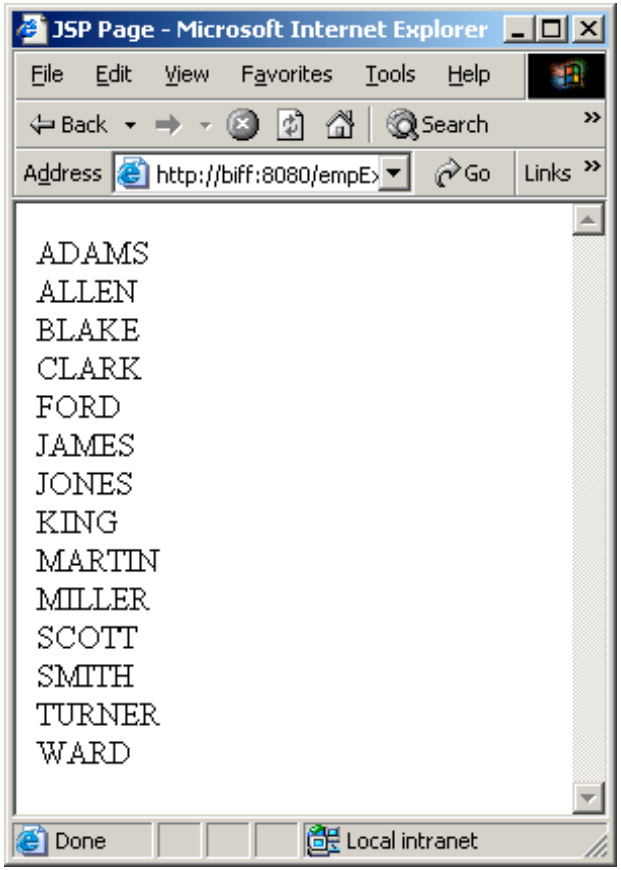


- DEPT table maps to <x:dept/>
- EMP table maps to <x:emp/>
- Columns are “JavaBean” accessible
- Parent foreign keys are also “JavaBean” accessible
  - [Example follows]
- Tags can be nested based on foreign keys
  - [Example follows]

SQLTags Complete Example

Simple Page

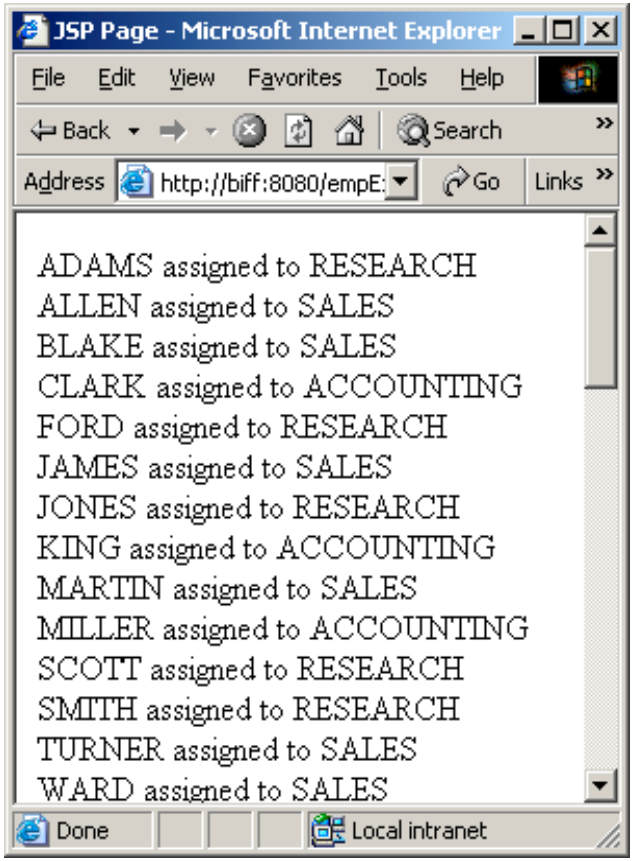
```
<%@ taglib uri="/WEB-INF/lib/demoTags.jar"
           prefix="sqltags" %>
<%@ taglib uri="/tags/jstl-core" prefix="c" %>

<html>
<head><title>JSP Page</title></head>
<body>
<sqltags:connection id="connect">

     <sqltags:emp id="e" where="order by ENAME">
        <c:out value="${e.ENAME}"/><br>
     </sqltags:emp>

</sqltags:connection>
</body>
</html>
```



Parent FK Lookups

Parent Lookup Example

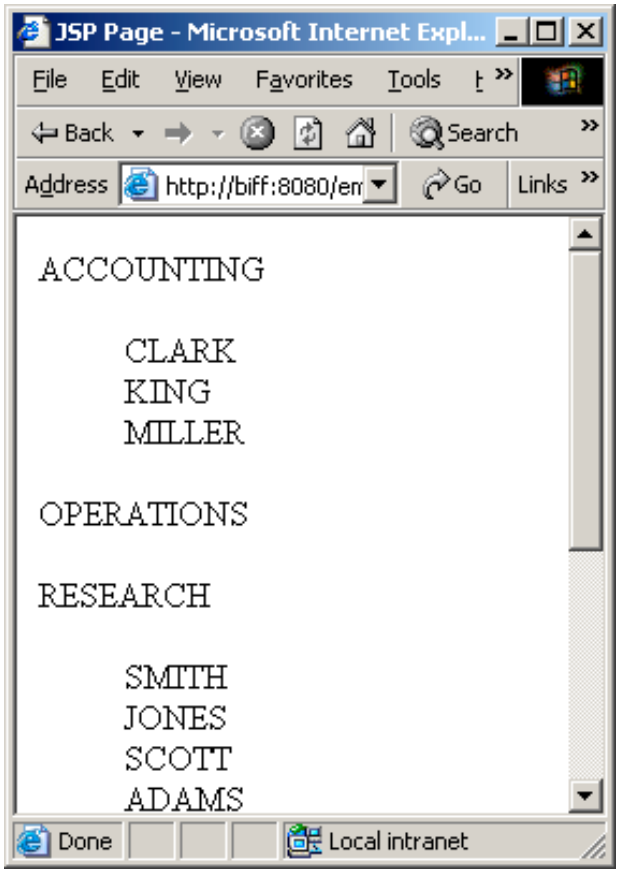
```
<%@ taglib uri="/WEB-INF/lib/demoTags.jar"
           prefix="sqltags" %>
<%@ taglib uri="/tags/jstl-core" prefix="c" %>
<html>
<head><title>JSP Page</title></head>
<body>
<sqltags:connection id="connect" >

    <sqltags:emp id="e" where="order by ename">
         <c:out value="${e.ENAME}"/> assigned to
         <c:out value="${e.FK_DEPTNO_PARENT.DNAME}"/>
         <br>
    </sqltags:emp>

</sqltags:connection>
</body>
</html>
```



Parent-Child Nesting

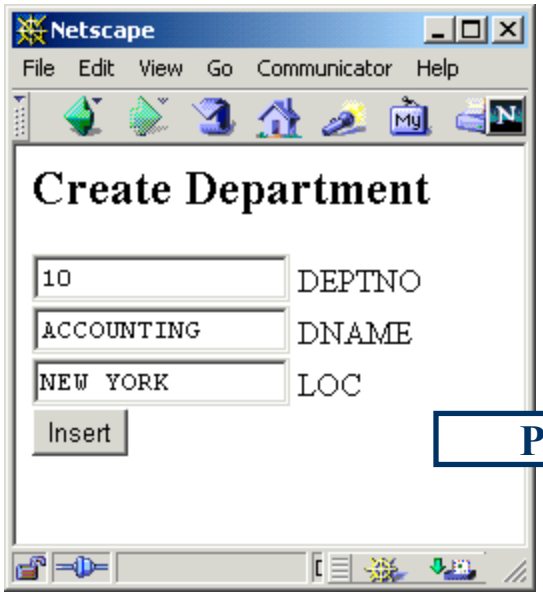
Parent-Child Example

```
<%-- edited for space --%>
<sqltags:connection id="connect" >

 <sqltags:dept id="d" where="order by DNAME">
      <c:out value="${d.DNAME}" />
      <BLOCKQUOTE>
      <sqltags:emp id="e" foreignKey="FK_DEPTNO"
                   parentName="d">
        <c:out value="${e.ENAME}"/>
        <br>
      </sqltags:emp>
      </BLOCKQUOTE>
 </sqltags:dept>

<%-- edited for space --%>
```



Request Properties

```
<h2>Create Department</h2>
<FORM>
        <INPUT NAME="DEPTNO" SIZE="15" VALUE="10"> DEPTNO
    <BR><INPUT NAME="DNAME"  SIZE="15" VALUE="ACCOUNTING"> DNAME
    <BR><INPUT NAME="LOC"    SIZE="15" VALUE="NEW YORK"> LOC
    <BR><INPUT NAME="fop" TYPE="submit" VALUE="Insert">
</FORM>
```



Since the "properties" attribute is "true" and the **field names in the Form match the column names** from the DEPT table the values will automatically be assigned.

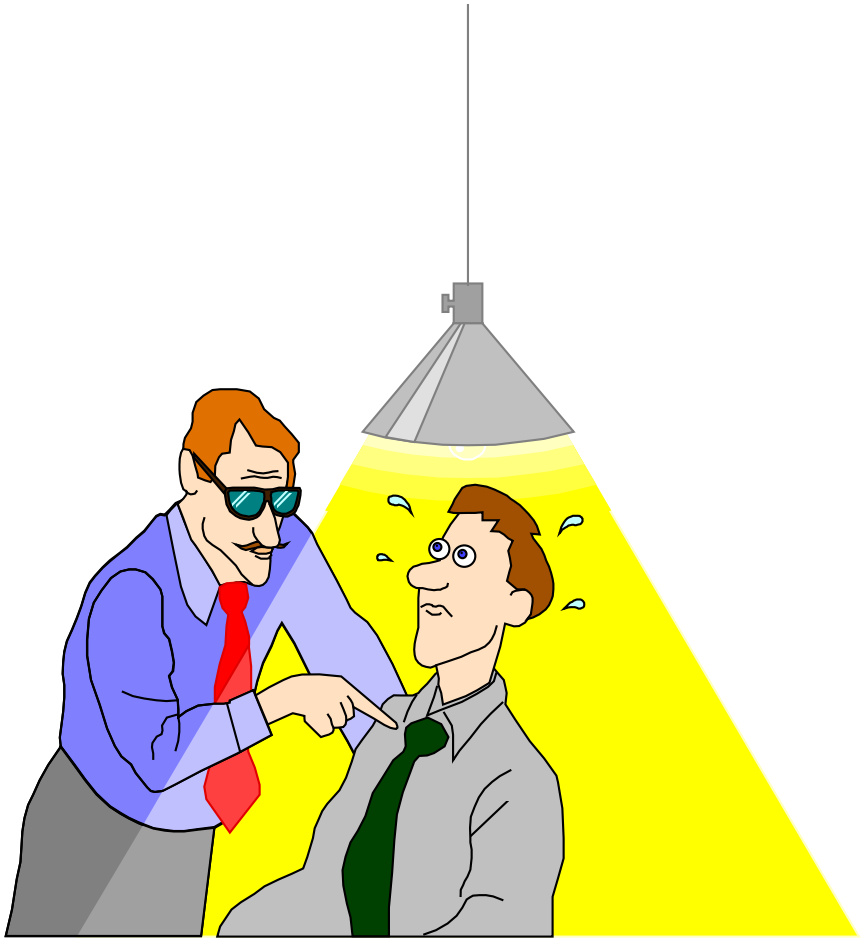
Processed By

```
<sqltags:dept
    id="d"
    buttonName="fop"
    properties="true"
    />
```

SQLTags Notes

- SQLTags is good for data-centric web applications with simple to moderate levels of complexity.
- Web applications that closely follow a solid data model are ideal for SQLTags.
- Complex, process intensive applications are, generally, **not** good candidates for SQLTags.

Questions? Comments?



- Database/JDBC …
- Java/JSP …
- User Interface …

- Generator
- JSP Development

Outline

- Lesson 1- Overview
- Lesson 2- Introduction to JSP (optional)
- Lesson 3- NetBeans IDE
- Lesson 4- Concurrent Versioning System
- Lesson 5- SQLTags