

# INTRODUCTION TO J2EE PLATFORM



# Enterprise application needs

- ❑ Access and integrating to existing enterprise information systems
- ❑ Evolve quickly from prototype to production
- ❑ Scalability to meet demand variations

# Access and integrating to existing enterprise information systems

- Enterprise Organizations needs to maintain existing information systems.
  - ◆ The goal is how to reuse these information assets.
  - ◆ Necessity of standard ways to access middle-tier and back-end services.
    - ◆ Database systems.
    - ◆ Transactions monitor.



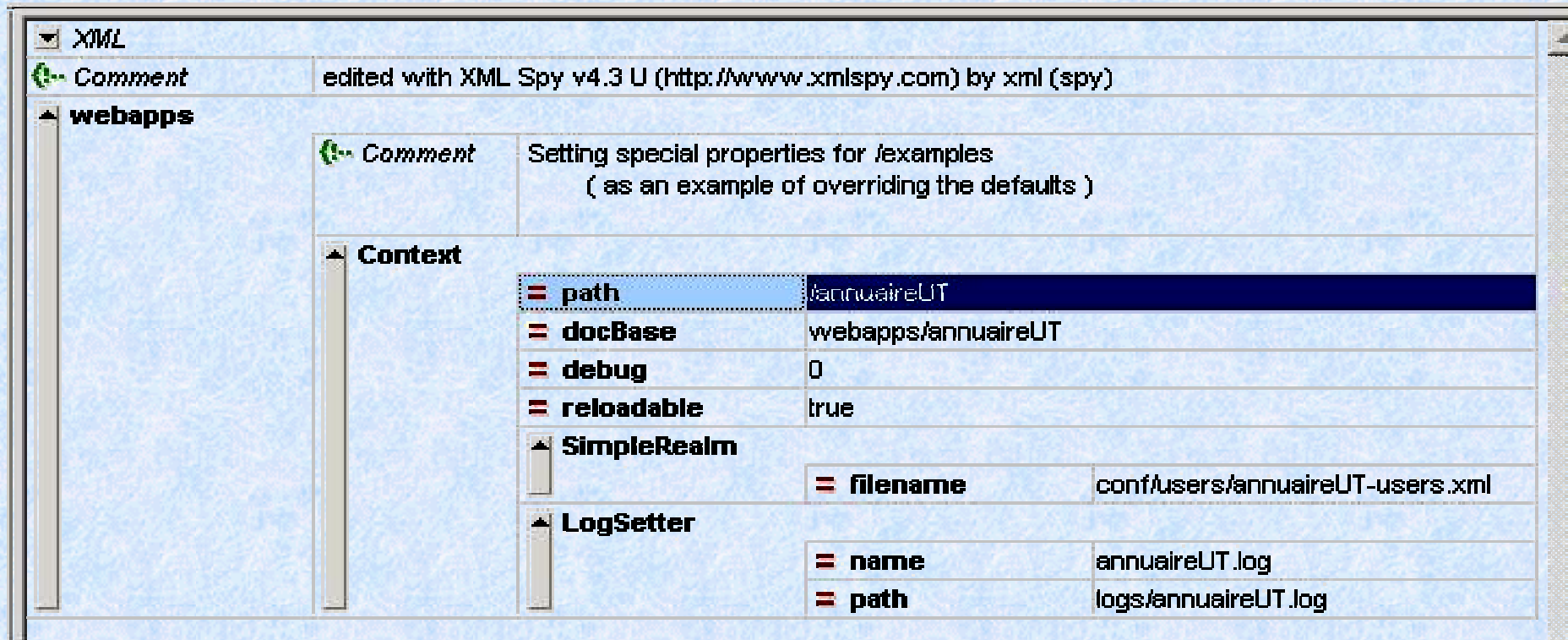
# Evolve quickly from prototype to production

- Based on java 2 platform standard edition.
  - ◆ Object Oriented design and component-based applications simplifies application maintenance. They can be updated and replaced independently.
  - ◆ Components help divide the labor of application development. Separation of user interface from business logic.

# Scalability to meet demand variations

- Through the use of XML files as application descriptor components can be configured for a specific environment without change the component source code

# Xml settings example



The screenshot shows the XML Spy v4.3 U interface. The main window displays an XML document structure. The root element is `XML`. Below it is a `Comment` element with the text "edited with XML Spy v4.3 U (http://www.xmlspy.com) by xml (spy)". The next level is `webapps`, which contains another `Comment` element with the text "Setting special properties for /examples ( as an example of overriding the defaults )". Underneath the comment is a `Context` element, which is expanded to show its configuration:

- `path`: /annuaireUT
- `docBase`: webapps/annuaireUT
- `debug`: 0
- `reloadable`: true

Inside the `Context` element, there are two sub-elements:

- `SimpleRealm`:
  - `filename`: conf/users/annuaireUT-users.xml
- `LogSetter`:
  - `name`: annuaireUT.log
  - `path`: logs/annuaireUT.log



# Advantages of an application model J2EE

- Through a set of specifications and several technologies J2EE platform offers several benefits in developing distributed applications.
  - ◆ Simplified development.
  - ◆ Integration with existing information systems.
  - ◆ Write-once-run anywhere™

# Architectural diversity

- J2EE provide a standard but doesn't specify or restrict containers's configuration.
  - ◆ J2EE platform can be made up of multiple containers on multiple platforms.



# What is an J2EE Application

- A J2EE application is composed by:
  - ◆ Enterprise beans.
  - ◆ Web client component modules.
  - ◆ All related files as
    - ◆ Graphics files (GIF, JPG).
    - ◆ HTML files.
  - ◆ A deployment descriptor: an app\_name.Xml that was read at runtime by the J2EE server.



# J2EE Application deployment

## Enterprise Archive File (EAR)

### Java Archive (JAR)

- enterprise beans
- All related files
- A deployment descriptor

### Web Archive (WAR)

- Web components
- A deployment descriptor

# Web Applications

## - Multi-tier applications -

- J2EE architecture is designed to provide a server-side and a client side support for multitier application.
- Multitiered applications are generally considered to be 3-tiered because they are distributed over three different location.
  - ◆ They can run on different devices.





# The various part of an application

- a *client tier*.

*Client Machine*

- a *middle tier*

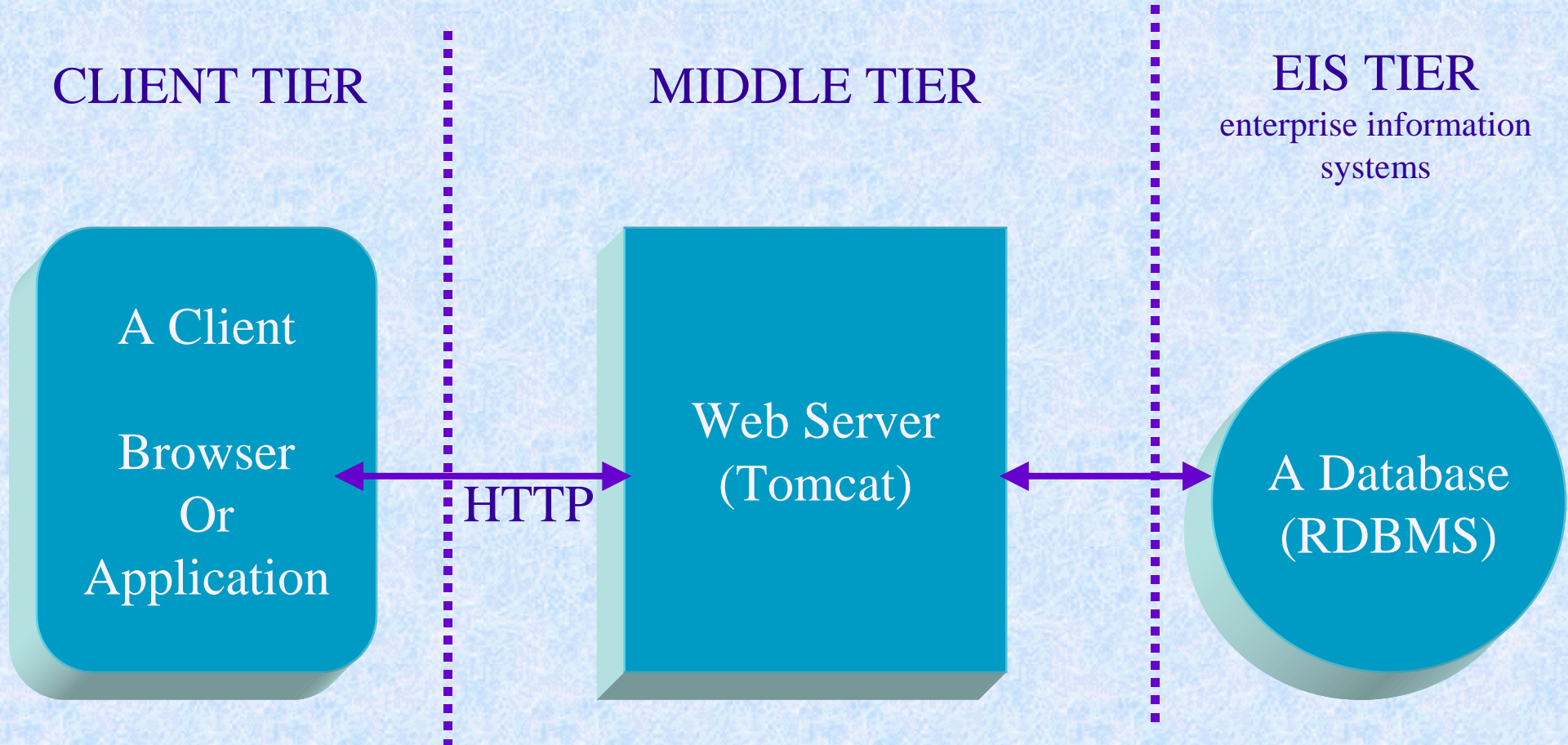
*J2EE Server  
Machine*

- a *back-end tier*.

*Database Server  
Machine*

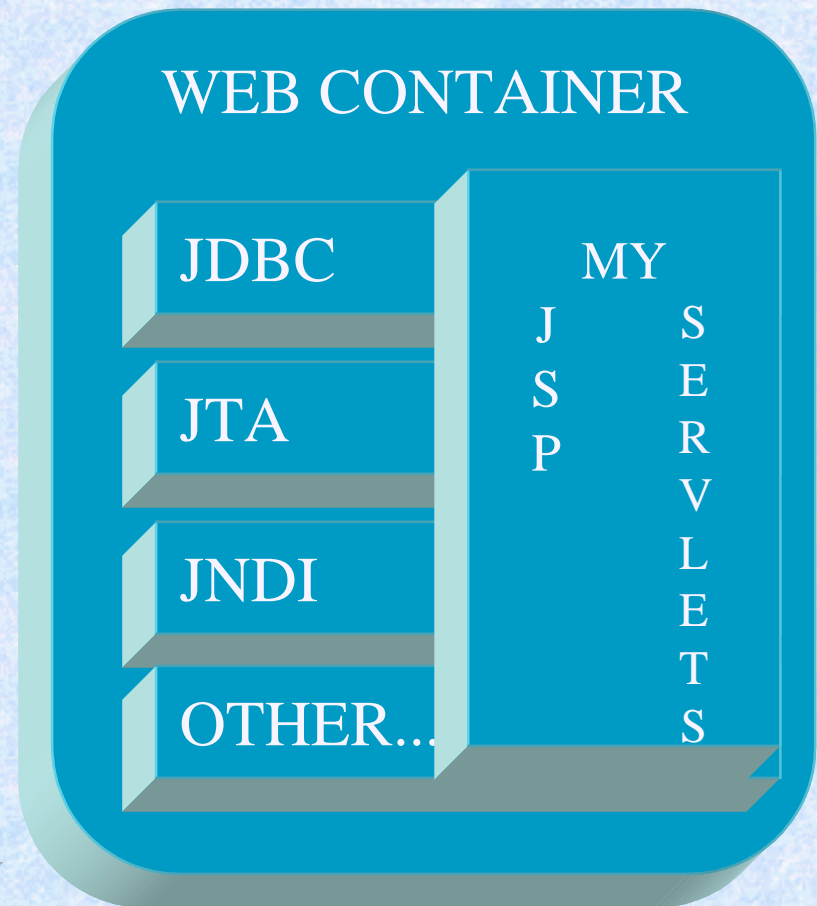


# A J2EE Enviroment



# Container-Based Component management

- The J2EE server provides underlying services in the form of a container for every component type.





# Container-Based Component management

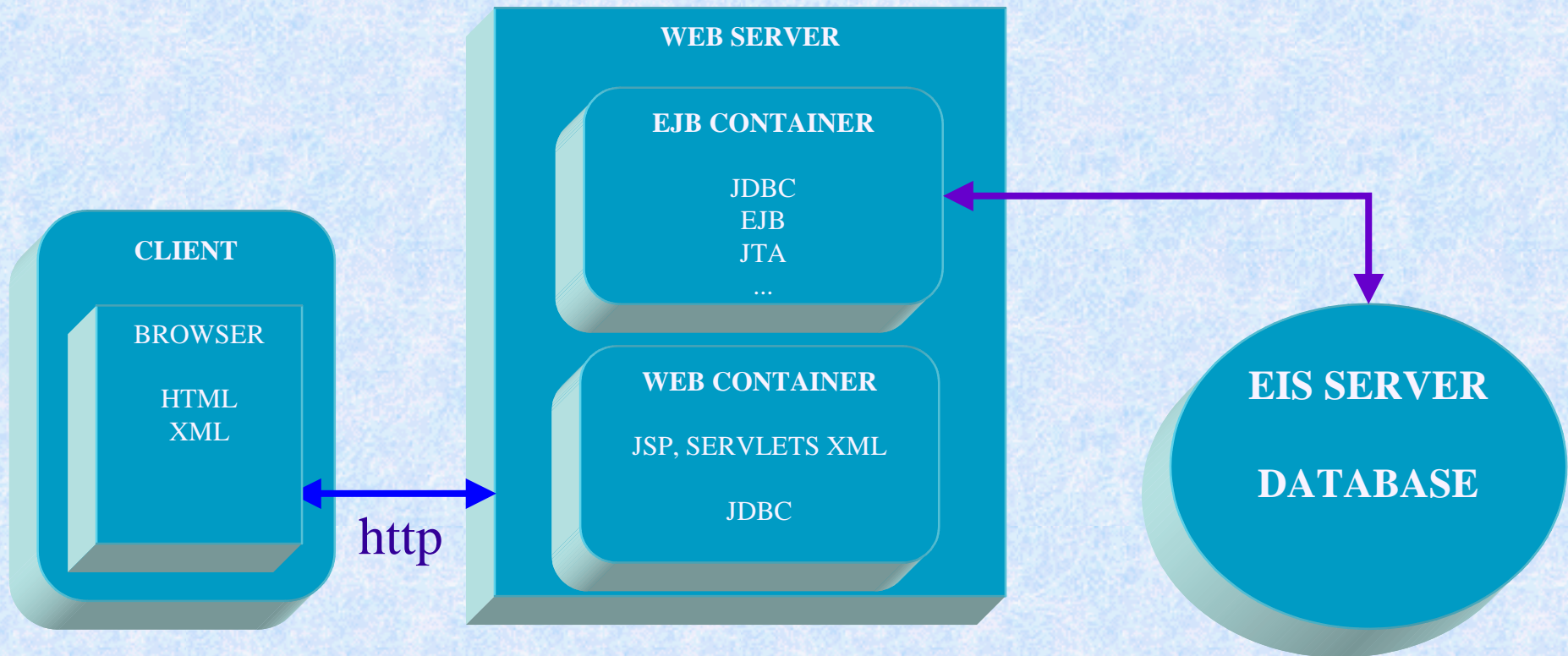
- Containers are the interface between a component and the low-level platform-specific functionality that supports that component.
  - ◆ Before a component can be executed it must be deployed into its container.
  - ◆ The assembly process involves specifying container settings for each component.
  - ◆ A component can expect these services to be available on any J2EE platform from any vendor.



# Containers types

- J2EE server provides.
  - ◆ A WEB container:
    - ◆ for managing execution of JSP pages and SERVLET components.
  - ◆ A EJB container:
    - ◆ for managing execution of enterprise beans.

# Components with Containers

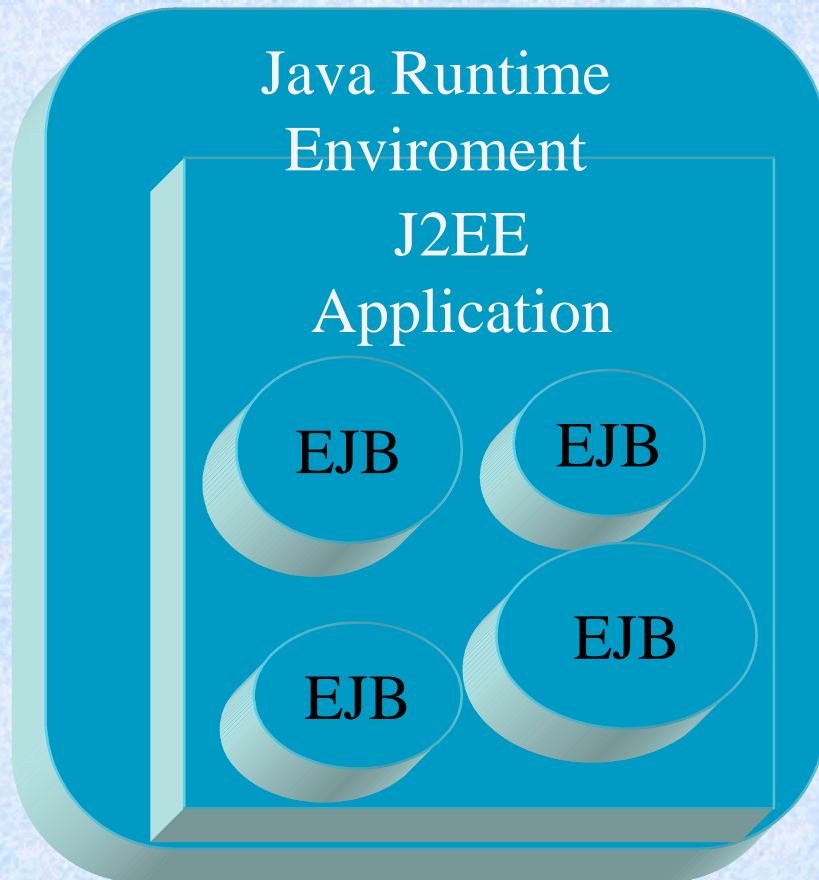




# J2EE Components – Java 2 Objects

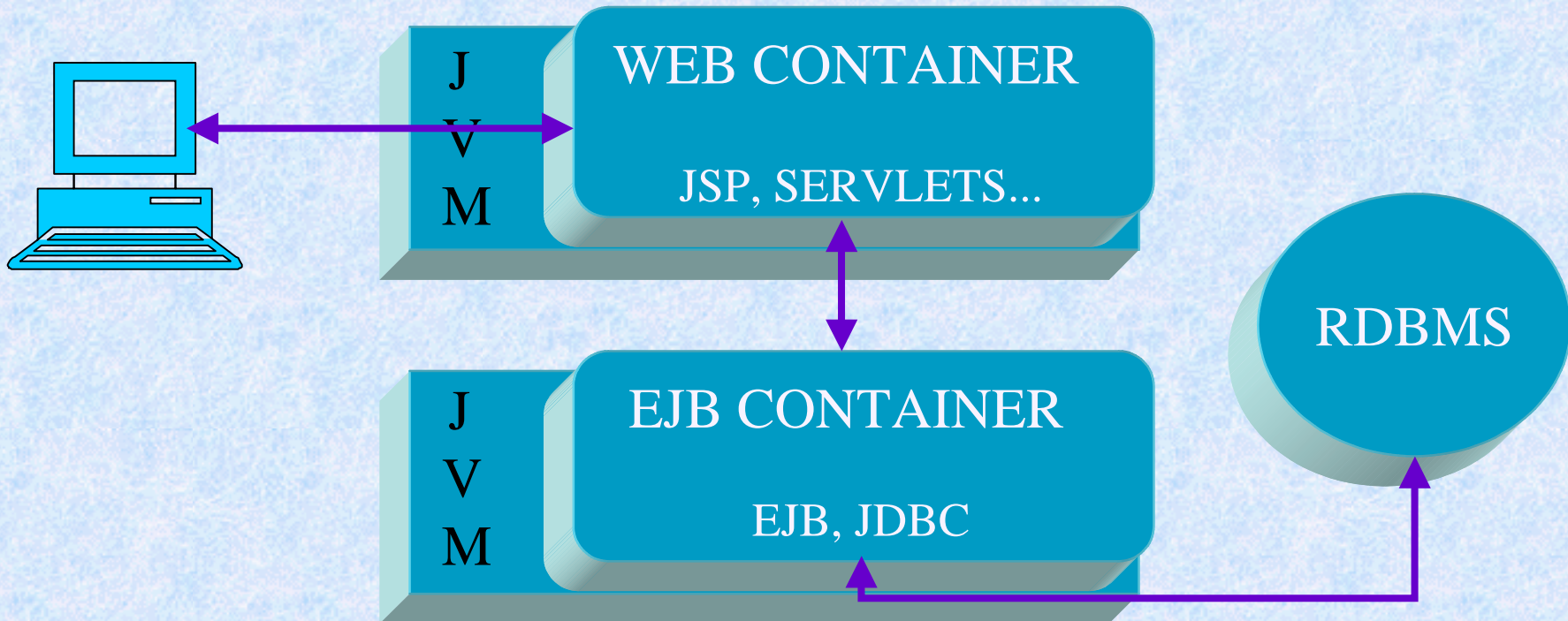
- J2EE components are
  - ◆ Assembled into a J2EE application.
  - ◆ Verified to be well formed, in compliance with J2EE specification.
  - ◆ Executed and managed by the J2EE Server.
- Java 2 Objects are
  - ◆ Component that simply run on a JVM.

# Components and java objects



# A divided Middle-Tier

Containers can be supported on different JVM



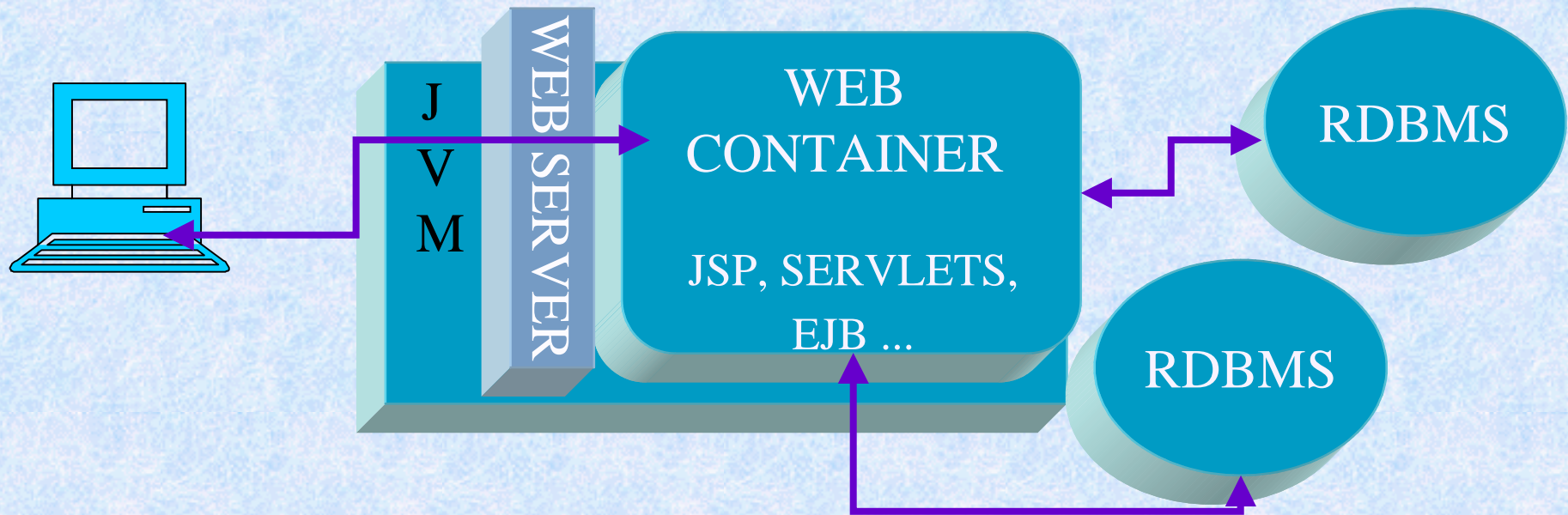


# A Common Middle-tier

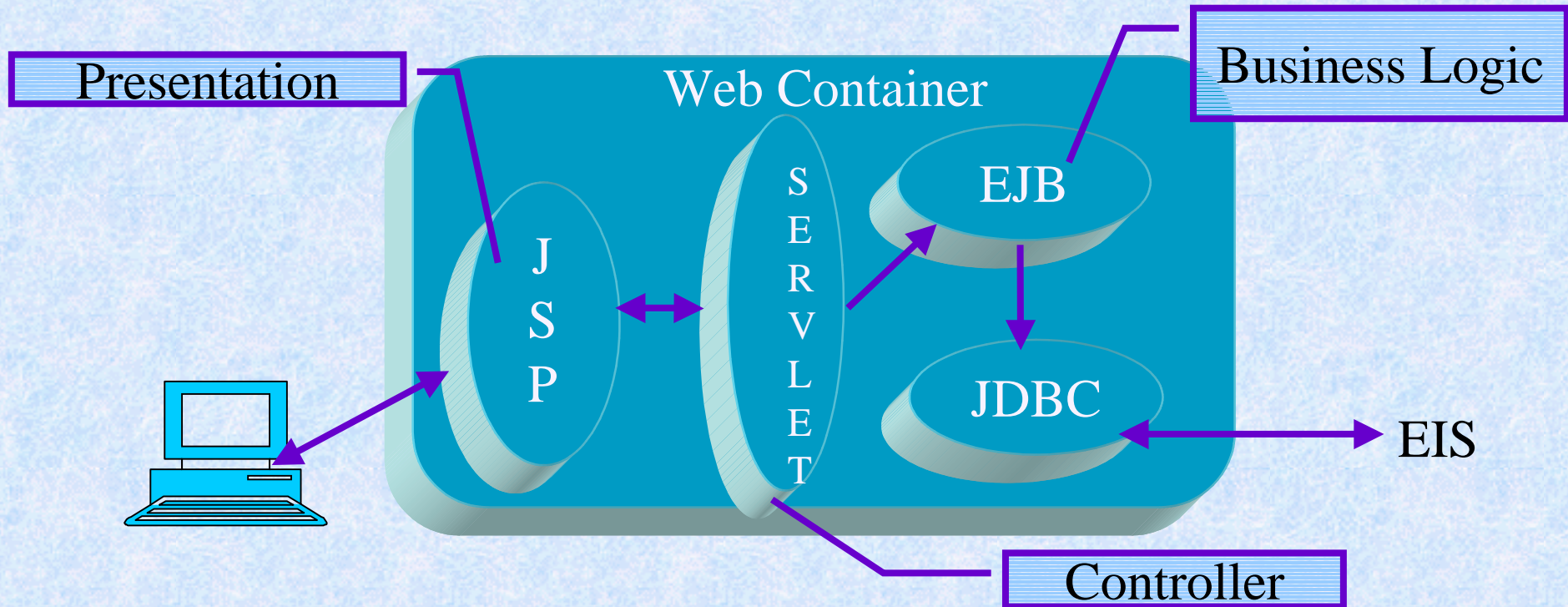
- In many cases we have the web container and the EJB container running within the same java virtual machine.
  - ◆ The term “web container” doesn’t necessarily mean a distinct process running on a distinct piece of hardware.

# Web Centric Application

The most used schema for many J2EE applications



# Presentation and Business Logic hosted in a Web Container





# JDBC API

- The J2EE platform requires the JDBC 2.0 api.
  - ◆ JDBC provides database-independent connectivity between J2EE platform and a wide range of RDBMS.
    - ◆ Perform connection and authentication to a database server.
    - ◆ Execute SQL statements.
    - ◆ Execute stored procedures.

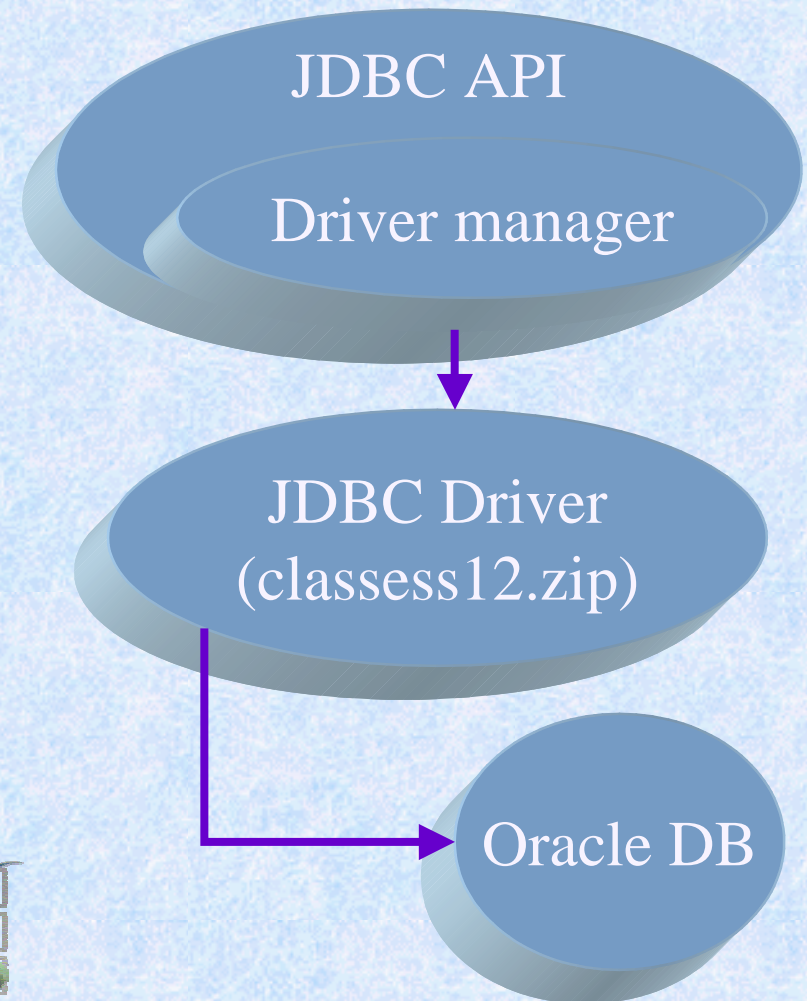
# JDBC DRIVER

- Those drivers come in four varieties.
  - ◆ Types intended for programmers writing applications.
    - ◆ Type 1: JDBC-ODBC bridge.
    - ◆ Type 2: Partial Java driver.
  - ◆ Types typically used by vendors of middleware or databases.
    - ◆ Type 3: Pure Java driver for database middleware.
    - ◆ Type 4: Direct-to-database pure Java.



# Type 2 JDBC Driver

- This type of driver converts JDBC calls into calls on the client API for Oracle, DB2 or other DBMS.





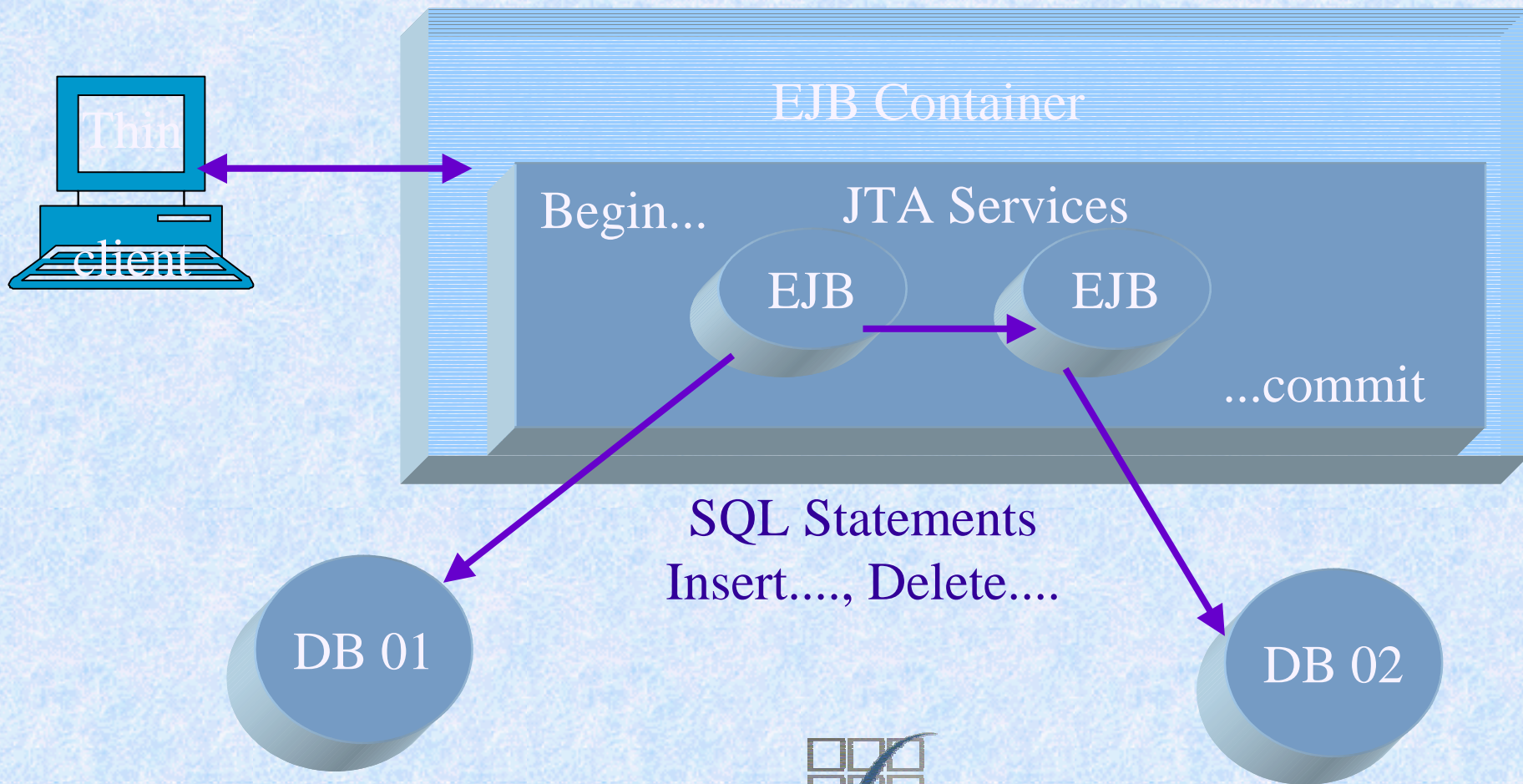
# Enterprise Java Beans

- A Java Object
  - ◆ Implements EJB Technology.
  - ◆ A Server-Side Component running in a J2EE Container.
  - ◆ Encapsulates the Business Logic of Application.

# Benefits using EJB

- Bean developer can concentrate on solving business problems.
  - ◆ EJB Container is responsible for services as Transaction Management (JTA).
- As result of implementing business rules the clients are thinner.
- A Bean can be assembled in more than one application.
  - ◆ Reduction of costs.

# Transactions





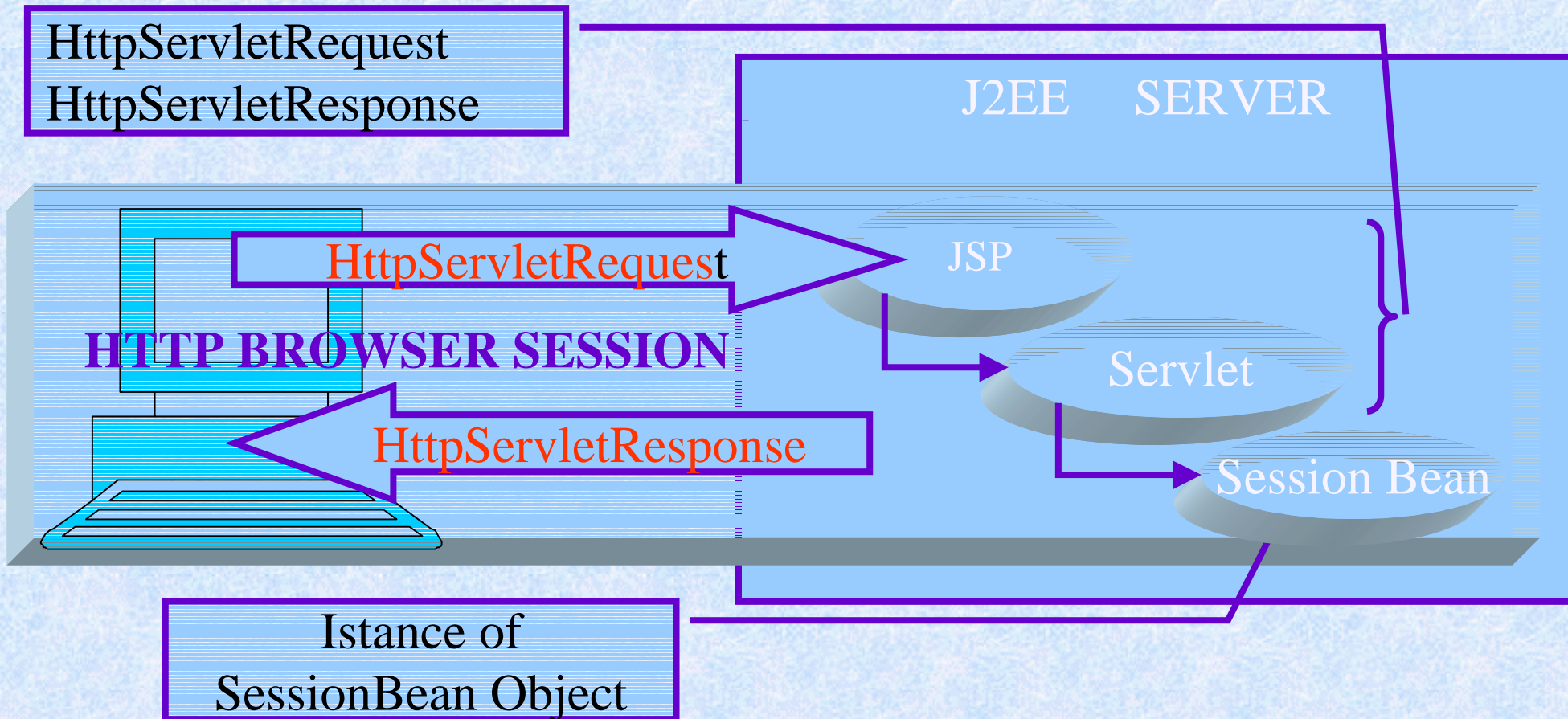
# Types of EJB

- **Session Bean.**
  - ◆ Performs task for a client inside the J2EE Server (A servlet can be a client).
- **Entity Bean.**
  - ◆ Represent a business entity object that exists in persistent storage.
- **Message-Driven Bean.**
  - ◆ Work as a listener for JMS API.

# Session Bean

- The session bean performs work for its client.
- Its Life cycle is like the interactive session.
- It is not persistent

# Session Bean Life Cycle





# Types of Session Bean

The state of an object consist of the values of its instance variables.

## ■ Stateful.

- ◆ Bean variables contain the state of a unique client-bean session (conversational state).

## ■ Stateless.

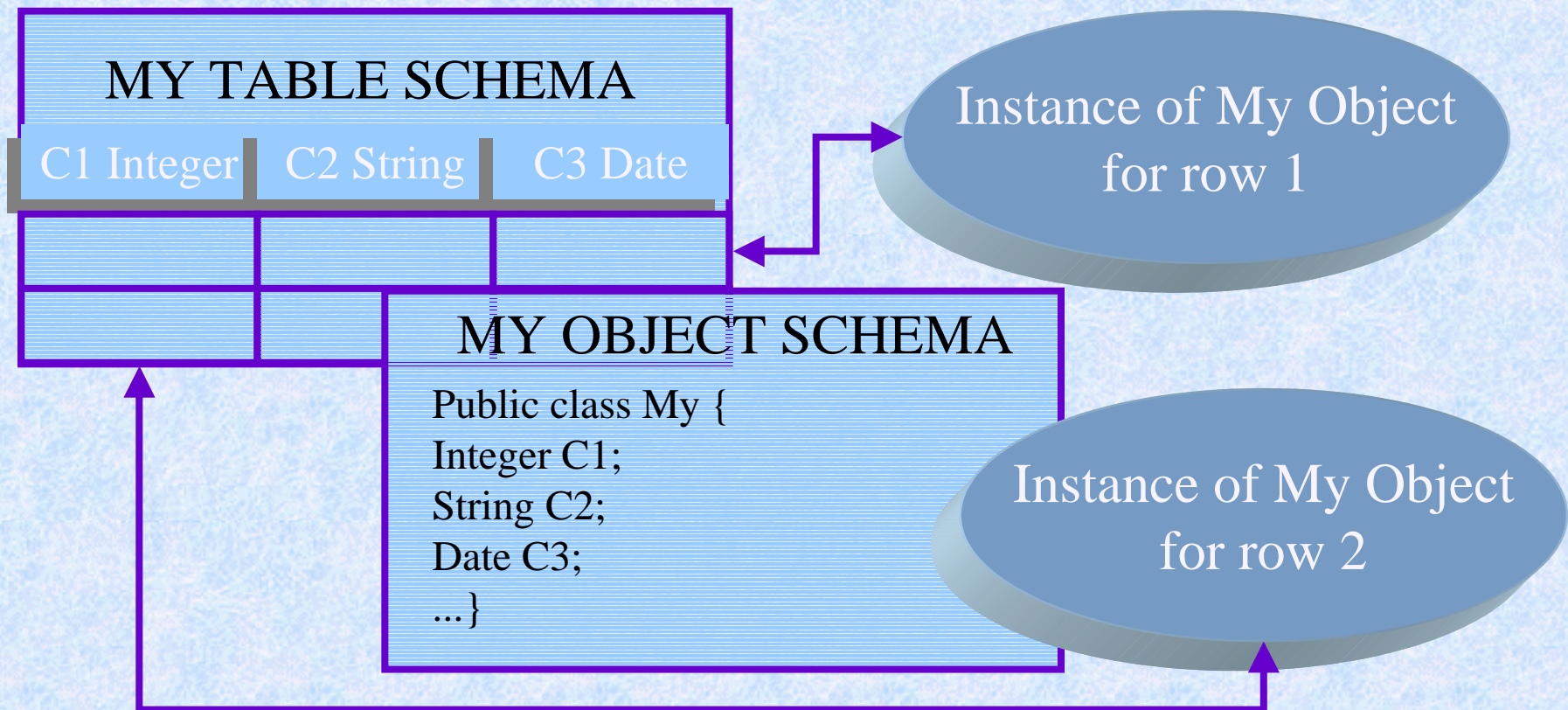
- ◆ Support multiple clients.
  - ◆ Bean variables contain state only for the duration of the invocation.



# Entity Bean

- Represents a business object in a persistent storage mechanism.
  - ◆ Allow shared access.
  - ◆ Have a primary key.
  - ◆ May cooperate with other entity beans.

# Just a row of a table

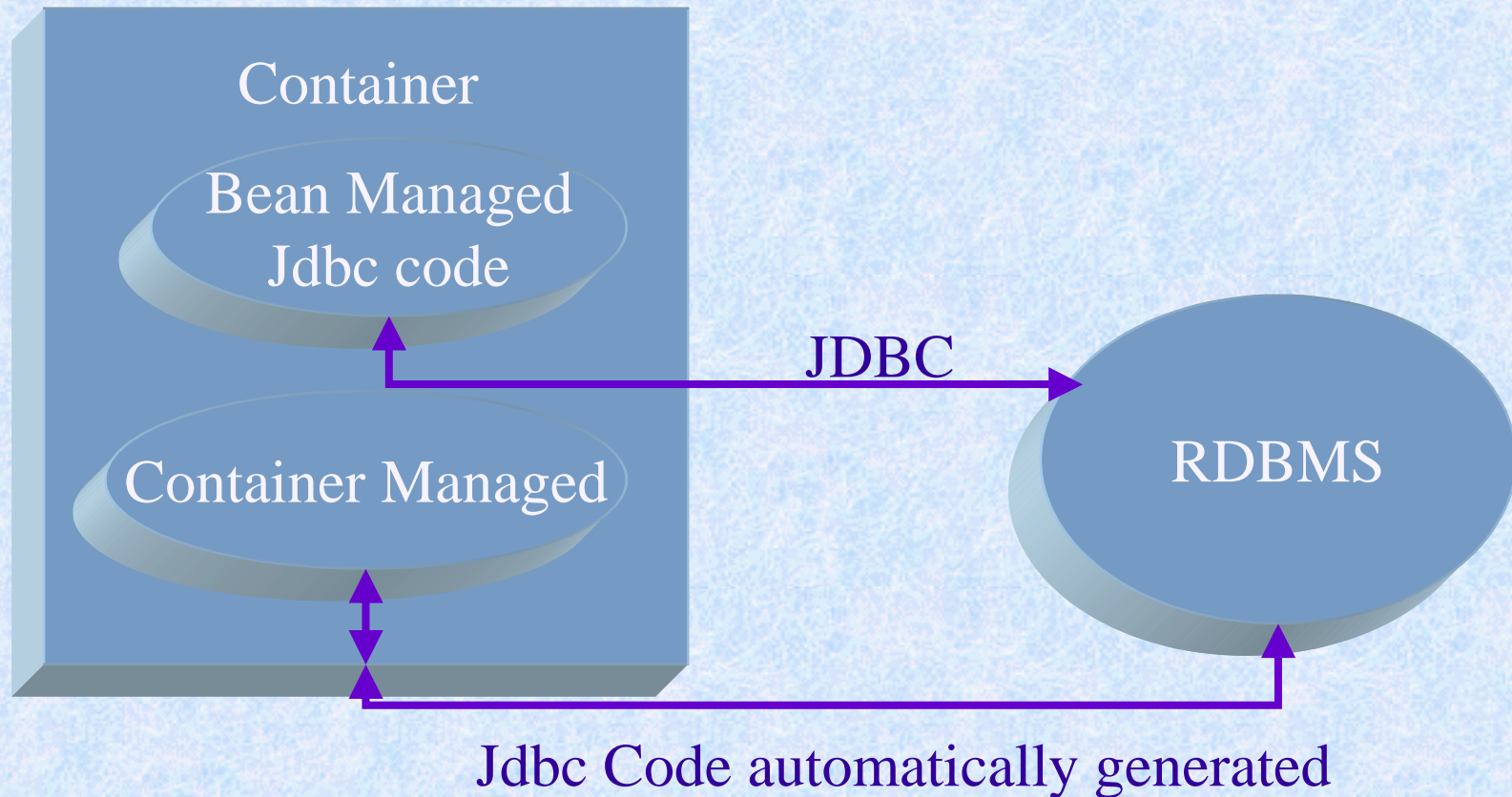




# Persistence

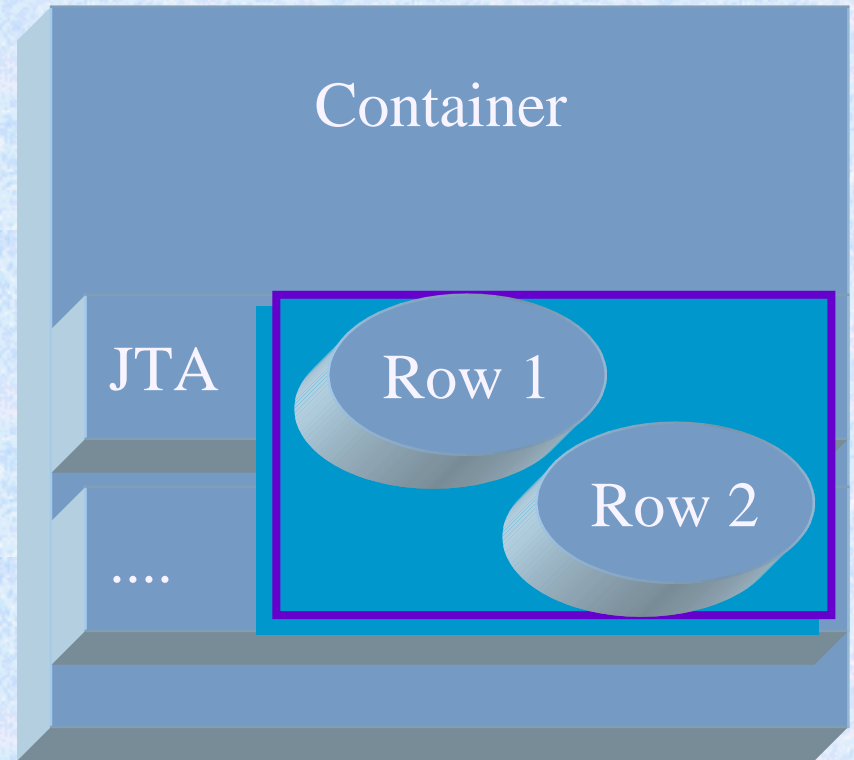
- This means that the entity bean's state exist beyond the lifetime of the J2EE Application.
- Two ways of managing bean persistence.
  - ◆ Bean-managed persistence.
  - ◆ Container-managed persistence.

# Managing Persistence



# Shared access

- Clients might want access to the same data represented with a bean for each row.
- Entity bean must work within transactions.





# Primary Key

- Like for a table of an RDBMS the primary key enables the client to locate a particular data represented by an instance of its specific Entity bean.
- Beans can be related each other like tables in a relational database.

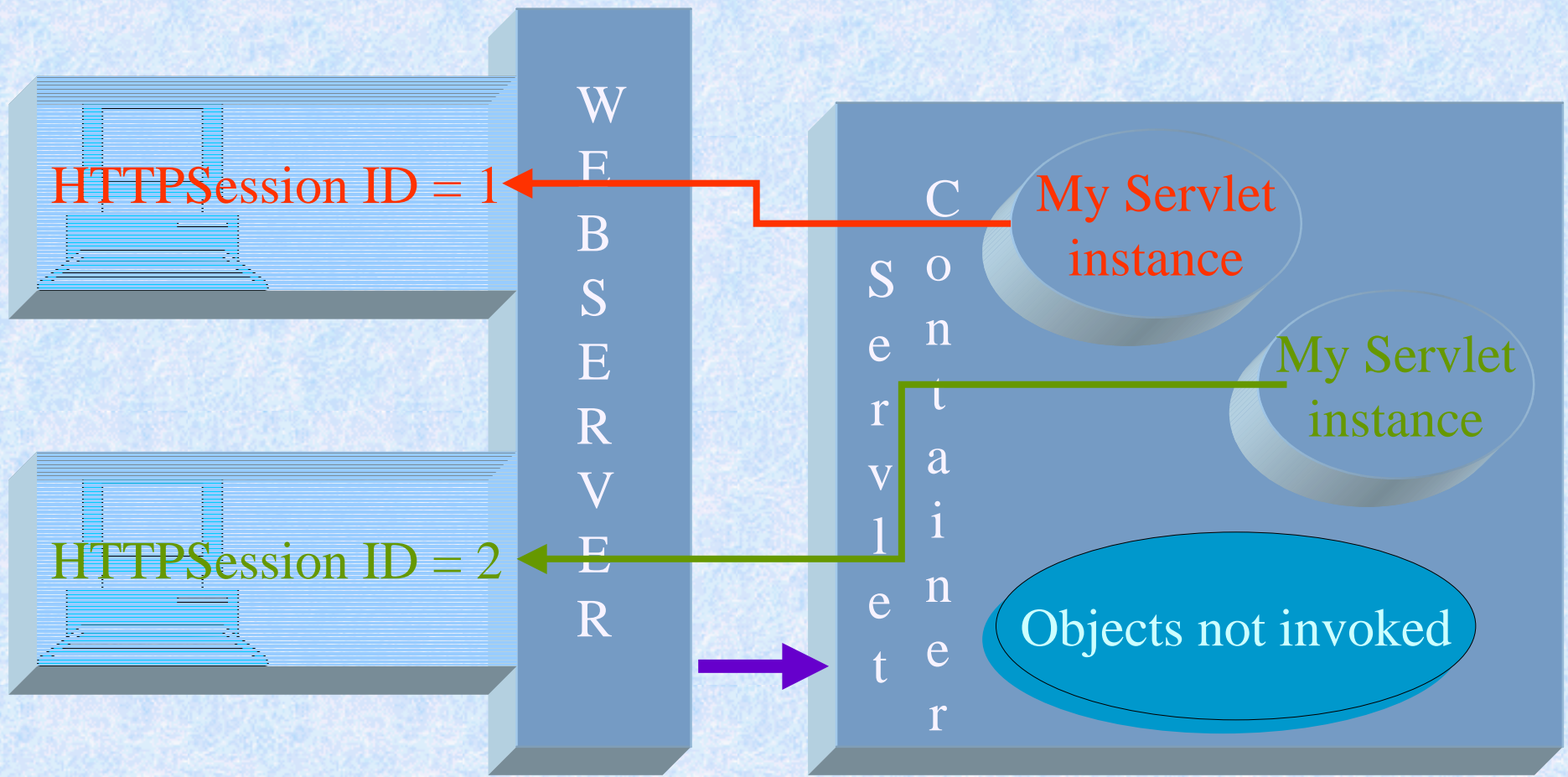
# Message-Driven Bean

- Java Message Service Technology allows J2EE Applications to process messages asynchronously.
  - ◆ A Message-Driven Bean is registered to a JMS listener and is managed by the container that can use JTA service for access to a database.

# Servlet

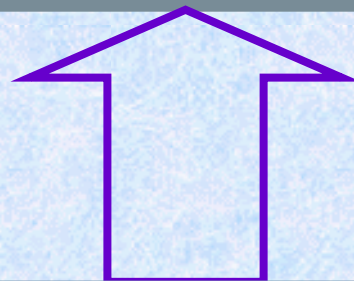
- A servlet is a Java-based web component, managed by a container, that generates dynamic content.





# Java Server Pages

JAVA SERVLET TECHNOLOGY



JAVA SERVER PAGES TECHNOLOGY

# A JSP Page

```
<html>
<!--
* Title:      ConsultOrga
* Description: This form is the start's form of Semide
* Date:      3 maggio 2002
* Author:    Cm - Isitel (Barbara Cavacchioli)
* Version:   1.0
-->
<head>
  <jsp:useBean id="AccessId" scope="session" class="consultation.ConsultOrgaThemeBean" />
  <jsp:setProperty name="AccessId" property="langue" value="<%=request.getParameter(\"codeLang\")%>"/>
  <%@ page import="consultation.*,management.*" %>
  <% String lang = ((ConsultOrgaThemeBean)request.getSession().getValue("AccessId")).getLangue(); %>
  <title><%=Manager.getHtmlValue("TITLECO",lang) %></title>

  <link rel="stylesheet" type="text/css" href="css_jsp/desil011.css">
</head>

<!-- Include the page FramesSearch.html (contains the frames tool_up: MenuSearch.jsp and tool_down: blank.html)
<jsp:include page="framesSearch.html" flush="true"/>
</html>
```



# JSP Elements

- Directive
  - ◆ Provide a global information.
- Action
  - ◆ May affect the current *out* stream using objects.
- Scripting - Based on Java programming language -
  - ◆ Declarations.
  - ◆ Scriptlets.
  - ◆ Expressions.

# Directive

A Directive is a message to the JSP Container that provide an information that is indipendent from any HTTPRequest received by the JSP page.

```
<% @ page import="consultation.*,management.*" %>
```

```
<% @ page isErrorPage="true" %>
```

```
<% @ include file= "URL"%>
```

# Action

An Action is related to the specific Request object received by the JSP. Some action types are standard but it is possible introduce new action types using the *taglib* directive.

```
<jsp:useBean id="AccessId" scope="session"  
class="consultation.ConsultOrgaThemeBean" />
```



# Accessing to a Bean Object

Using Bean Introspection the JSP Specification defines two types of actions to access to object declared with the *useBean* action.

```
<jsp:setProperty name="AccessId" property="langue"  
param ="codeLang"/>
```

```
<jsp:getProperty name="AccessId" property="status" />
```

# Accessing to other pages

JSP Specification defines two types of actions to interact with other JSP

```
<jsp:include page="framesSearch.html" flush="true"/>
```

```
<jsp:forward page="Error.jsp" />
```

# Tag Extension

- Define new Action tags.
  - ◆ New Actions are imported into a JSP using the *taglib* Directive.
  - ◆ A Tag Library is a collection of actions that encapsulate some functionality.
    - ◆ The Tag Library is associated with a Tag Library Descriptor.
  - ◆ The JSP use a TagHandler object to interact with the server.





# Tag Library Descriptor

- A TLD is an xml file used by the JSP container to interpret pages that uses the specific *taglib*.
  - ◆ Each action in the library is described by giving:
    - ◆ Its name.
    - ◆ The class for its tag handler class.
    - ◆ Informations on TagExtraInfoClass and all attributes of the action.



# A TLD File

XML

<b>version</b>	1.0
<b>encoding</b>	ISO-8859-1

DOCTYPE taglib

<b>PUBLIC</b>	"-//Sun Microsystems, Inc./DTD JSP Tag Library 1.1/EN"	"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd"
---------------	--	---

Comment: a tag library descriptor

taglib

Comment: after this the default space is "http://java.sun.com/j2ee/dtds/jsptaglibrary\_1\_2.dtd"

<b>tlbvers...</b>	1.0
<b>jspvers...</b>	1.1
<b>shortna...</b>	simple
<b>uri</b>	
<b>info</b>	A simple tab library for the examples

tag

Comment: A simple Tag

Comment: foo tag

tag

<b>name</b>	foo
<b>tagclass</b>	examples.FooTag
<b>teiclass</b>	examples.FooTagExtraInfo
<b>bodycontent</b>	JSP
<b>info</b>	Perform a server side action; uses 3 mandatory attributes

attribute (3)

	name	required
1	att1	true
2	att2	true
3	att3	true

# Scripting

JSP Specification 1.1 define just a language for the *language* attribute of the Page Directive.

```
<%! String lang = "english";%>
```

```
<% if(UserID.getStatus() != null) { ... }%>
```

```
<%= Manager.getHtmlValue("ORGANAME", lang)%>
```



# How work a JSP

- The JSP Container on a Web Server manage all request for JSP resources.
  - ◆ Its first role is to locate the jsp relative instance or to create one from the jsp resource.
  - ◆ Then process the request object and the response.



```

1: package jsp.consultation;
2:
3: import javax.servlet.*;
4: import javax.servlet.http.*;
5: import javax.servlet.jsp.*;
6: import consultation.*;
7: import management.*;
8:
9:
10: public class ConsultOrga_1 extends org.apache.jasper.runtime.HttpJspBase {
11:
12:     // begin [file="C:\\WebServer\\jakarta-tomcat-3.3\\webapps\\annuaireUT\\jsp\\consultation\\ConsultOrga.jsp";from=(9,2);
13:     // end
14:     public ConsultOrga_1( ) {
15:     }
16:     private boolean _jspx_inited = false;
17:     public final synchronized void _jspx_init() throws org.apache.jasper.JasperException {
18:         if (!_jspx_inited) {
19:             _jspx_inited = true;
20:         }
21:     }
22:     public void _jspService(HttpServletRequest request, HttpServletResponse response)
23:         throws java.io.IOException, ServletException {
24:         JspFactory _jspxFactory = null;
25:         PageContext pageContext = null;
26:         HttpSession session = null;
27:         ServletContext application = null;
28:         ServletConfig config = null;
29:         JspWriter out = null;
30:         Object page = this;
31:         String _value = null;
32:         try {
33:             try {
34:                 _jspx_init();
35:                 _jspxFactory = JspFactory.getDefaultFactory();
36:                 response.setContentType("text/html;charset=ISO-8859-1");
37:                 pageContext = _jspxFactory.getPageContext(this, request, response, "", true, 8192, true);
38:                 application = pageContext.getServletContext();
39:                 config = pageContext.getServletConfig();

```

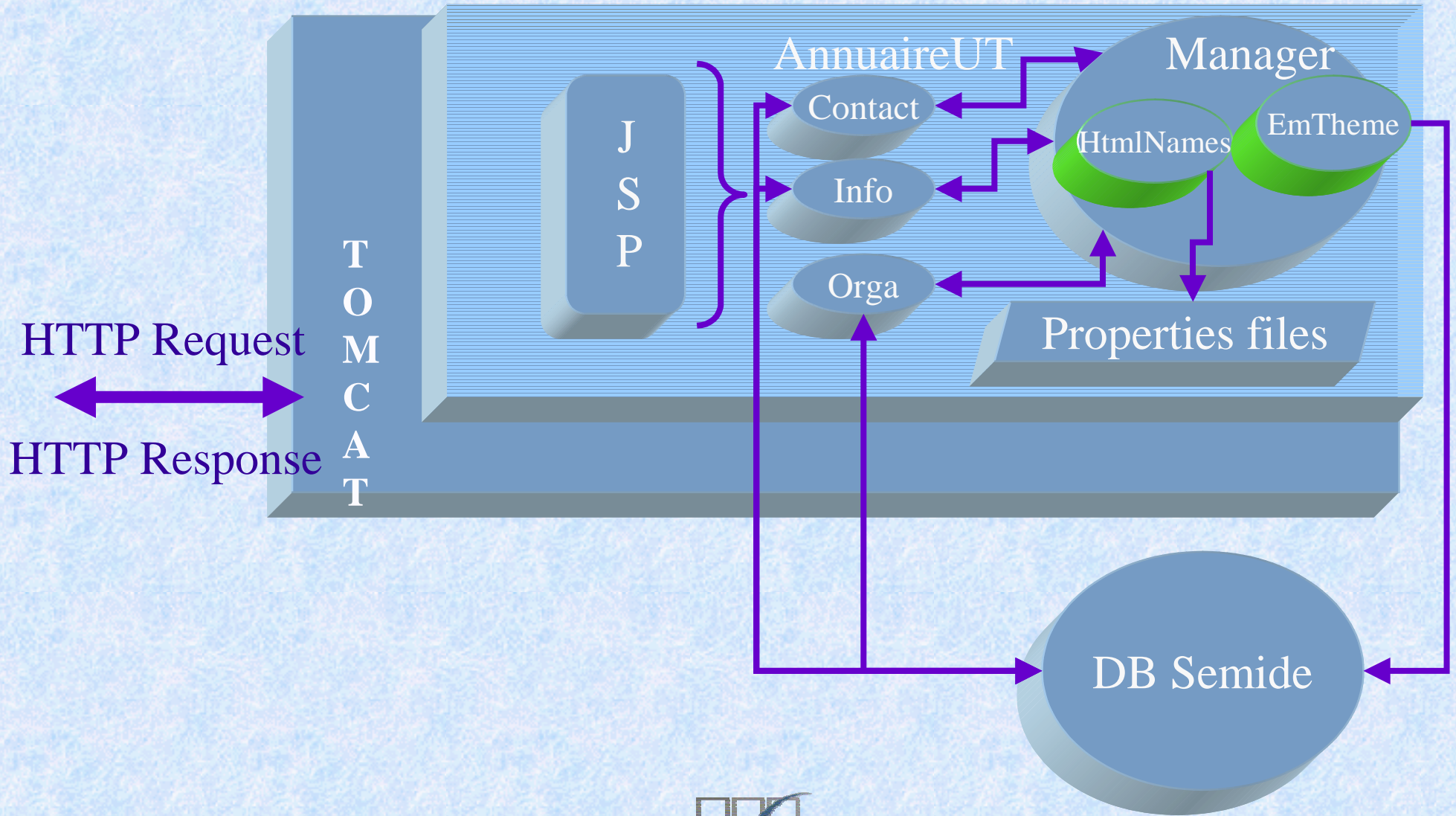
```

39:         config = pageContext.getServletConfig();
40:         session = pageContext.getSession();
41:         out = pageContext.getOut();
42:
43: // HTML // begin [file="C:\\WebServer\\jakarta-tomcat-3.3\\webapps\\annuaireUT\\jsp\\consultation\\ConsultOrga.jsp";from=(0,0);
44:         out.write("<html> \r\n<!--\r\n* Title:
45: ...
46:                 "<head>\r\n\t\t");
47: ...
48:         consultation.ConsultOrgaThemeBean AccessId = null;
49:         boolean _jspx_specialAccessId = false;
50:         synchronized (session) {
51:             AccessId= (consultation.ConsultOrgaThemeBean)
52:             pageContext.getAttribute("AccessId",PageContext.SESSION_SCOPE);
53:             if ( AccessId == null ) {
54:                 _jspx_specialAccessId = true;
55:                 try {
56:                     AccessId = (consultation.ConsultOrgaThemeBean) java.beans.Beans.instantiate(this.getClass().getClassLoade:
57:                     "consultation.ConsultOrgaThemeBean");
58:                 } catch (Exception exc) {
59:                     throw new ServletException (" Cannot create bean of class "+consultation.ConsultOrgaThemeBean", exc);
60:                 }
61:                 pageContext.setAttribute("AccessId", AccessId, PageContext.SESSION_SCOPE);
62:             }
63:         }
64:         if(_jspx_specialAccessId == true) {
65: .....
66:         org.apache.jasper.runtime.JspRuntimeLibrary.handle SetProperty(pageContext.findAttribute("AccessId"), "langue", request.
67:         >
68:         ...
69:         out.write("\r\n\t\t<title>");
70:         ...
71:         out.print(Manager.getHtmlValue("TITLECO", lang));
72:         ...
73:         out.write("</title>\r\n\r\n\t\t<link rel=\"stylesheet\" type=\"text/css\" href=\"css_jsp/desil011.css\">\r\n\t</head>

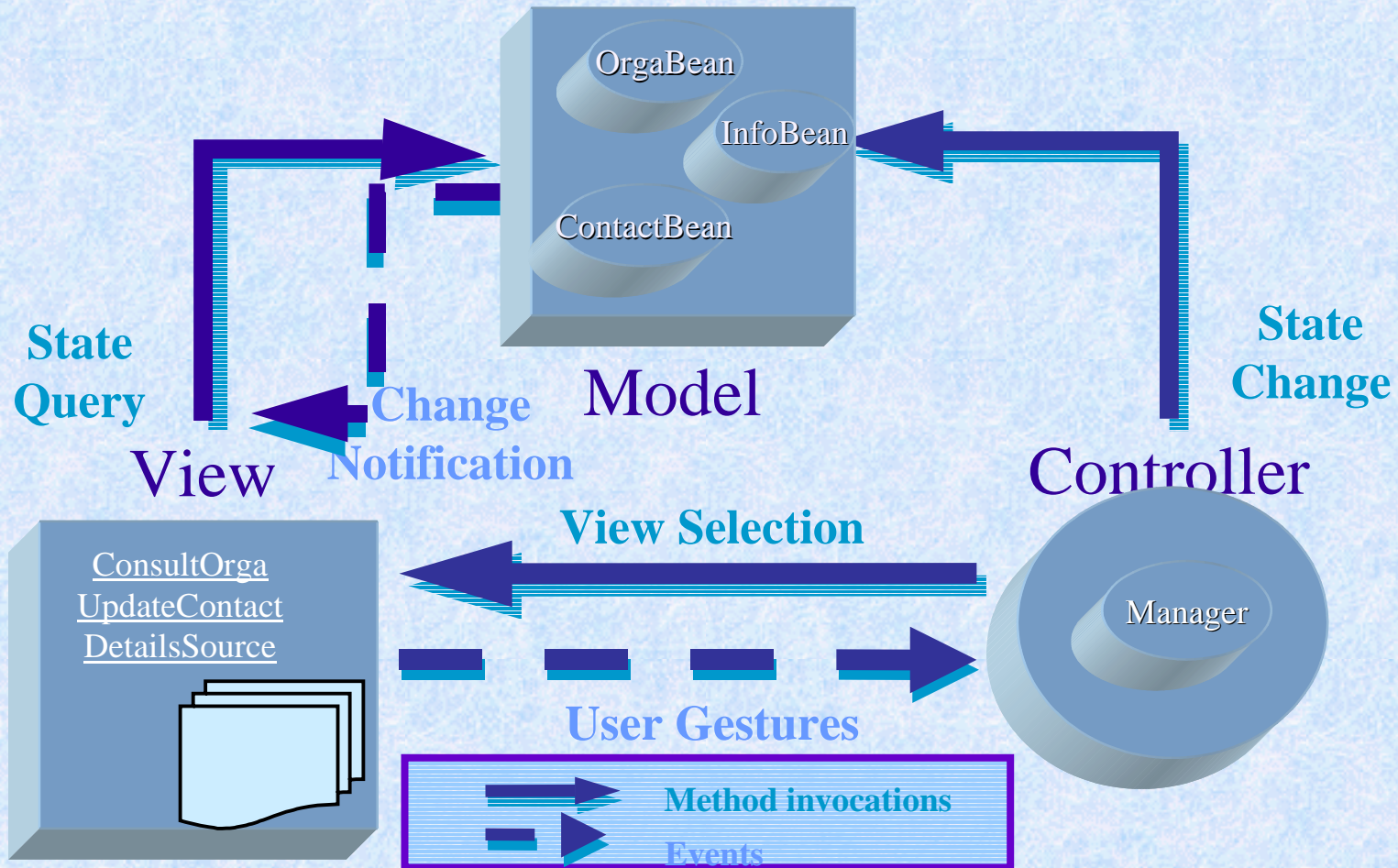
```



# Emwis Application



# Model View Controller



# Where to Get More Information

- Java2 Platform Enterprise Edition: <http://java.sun.com/j2ee>
- JavaBeans: <http://java.sun.com/beans>
- JSP: <http://java.sun.com/products/jsp>
- Servlet: <http://java.sun.com/products/servlet>