Web Container Components Servlet – JSP – Tag Libraries

Servlet

– Standard Java class to handle an HTTP request and response

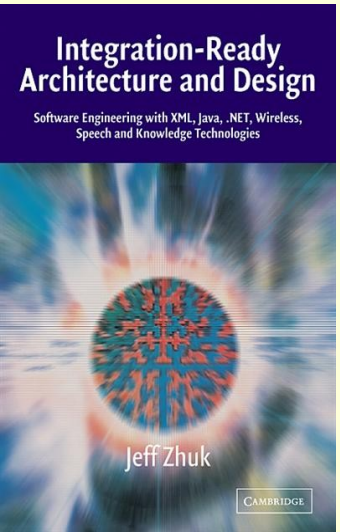
– Mixes Java and HTML code

JSP

– Allows some separation of Java and HTML code

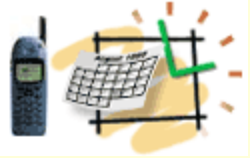
• Using JavaBeans as “backend” helpers

– JSP scriplets mix a lot of Java and HTML code



Tag libraries

– Define custom tags to remove Java code from JSP pages



Java Servlet and JSPs

Supported by all current application servers,

JSPs are automatically translated into Java servlet classes.

Java servlets and JSPs are collaborative mechanisms; they can be used in together or in isolation.

In JSP, HTML code looks exactly as it would appear in a ".html" file. The JSP interpreter automatically translates into servlet compatible form.

JSPs introduced the notion of "tags";
The custom or user-defined tags is also supported by the JSP spec.

Java Standard Tag Library, Struts, BEA WebLogic , and more tag library frameworks are available.

Java Servlets serve HTTP

The Hypertext Transfer Protocol (HTTP), like HTML was developed by Tim Berners-Lee and CERN in1989 with the main purpose to transfer arbitrary data, and therefore is not restricted to just text, as its name might suggest.

HTTP is a stateless, connectionless protocol
HTTP 1.1 requests
– GET, POST, HEAD, PUT, TRACE, DELETE, OPTIONS, CONNECT

Although the HTTP 1.1 spec. contains 7 types of *requests*, many simple HTTP servers only know the commands “GET”, “POST”, and “HEAD”

HTML forms have (hidden or visible) input fields and the ACTION field
Default request method is GET

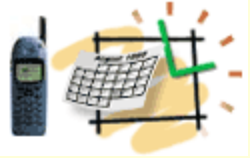
```
<FORM ACTION="/directoryName/ServletClassName“ METHOD=“POST”>
Name: <INPUT TYPE="TEXT" NAME=”name"/>
Password: <INPUT TYPE=“PASSWORD" />
<CENTER><INPUT TYPE=”SUBMIT"/>GO!</CENTER>
</FORM>
```

What is Servlet?

A servlet extends functionality of a web server similar to an applet in a browser.

- **Servlet is a piece of Java code**
- **Servlet is loaded by a Web server to handle client requests**
- **Servlet's code stays in memory when the request terminates**
- **Servlet can retain its connection across requests**
- **Servlet can chain a client request to another Servlet (**servlet chaining**)**
- **Servlets usually:**
  1) **Accept input data**
  2) **Run or *delegate* business logics**
  3) **Dynamically generate or just transfer HTML response page**

For example, when a browser sends a request to the web server, the server may forward the request to a servlet that can process the request and construct a response, usually an HTML page, that is returned to the browser.



Servlet Code Planning

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AbstractServlet extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

     //Get HTTP headers, cookies, session and form data from the request
     //Apply business logic, e.g., user authorization
     //Fill outgoing "response" to specify the HTTP response
     //including status code, headers, cookies, session data,
     // send output HTML page back to a client

   }
}
```

Servlet Code Example

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloServlet extends javax.servlet.http.HttpServlet {

    public void doGet(
      HttpServletRequest request, HttpServletResponse response)
      throws javax.servlet.ServletException, java.io.IOException {

        String html = ”<HTML>\n" +
"<HEAD><TITLE>HTML Title</TITLE></HEAD>\n" +
"<BODY>\n" +
"<H1> This is just an example</H1>\n" +
"</BODY></HTML>";

        response.getWriter().println(html);
    }
}
```

Servlet Best Practices

New services can be added run time as new JSPs/ASPs or Java™/.NET classes

```
//serviceName and serviceDetails are to be populated
// by servlet doPost() , doGet() or service() methods

String serviceName = request.getParameter("service");
Hashtable serviceDetails = getServiceDetails();

Service service =          // known or new service
(Service) Class.forName(serviceName).newInstance();

String content = service.run(serviceDetails);
response.setContentType("text/html");  // "application/xsl" and etc.
response.getWriter().println(content);
```

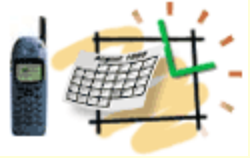
XML based Service API allows us to describe any existing and future service

```
<ServiceRequest service="Mail" action="get">
  <Param><paramName1=…></Param>
</ServiceRequest>
```

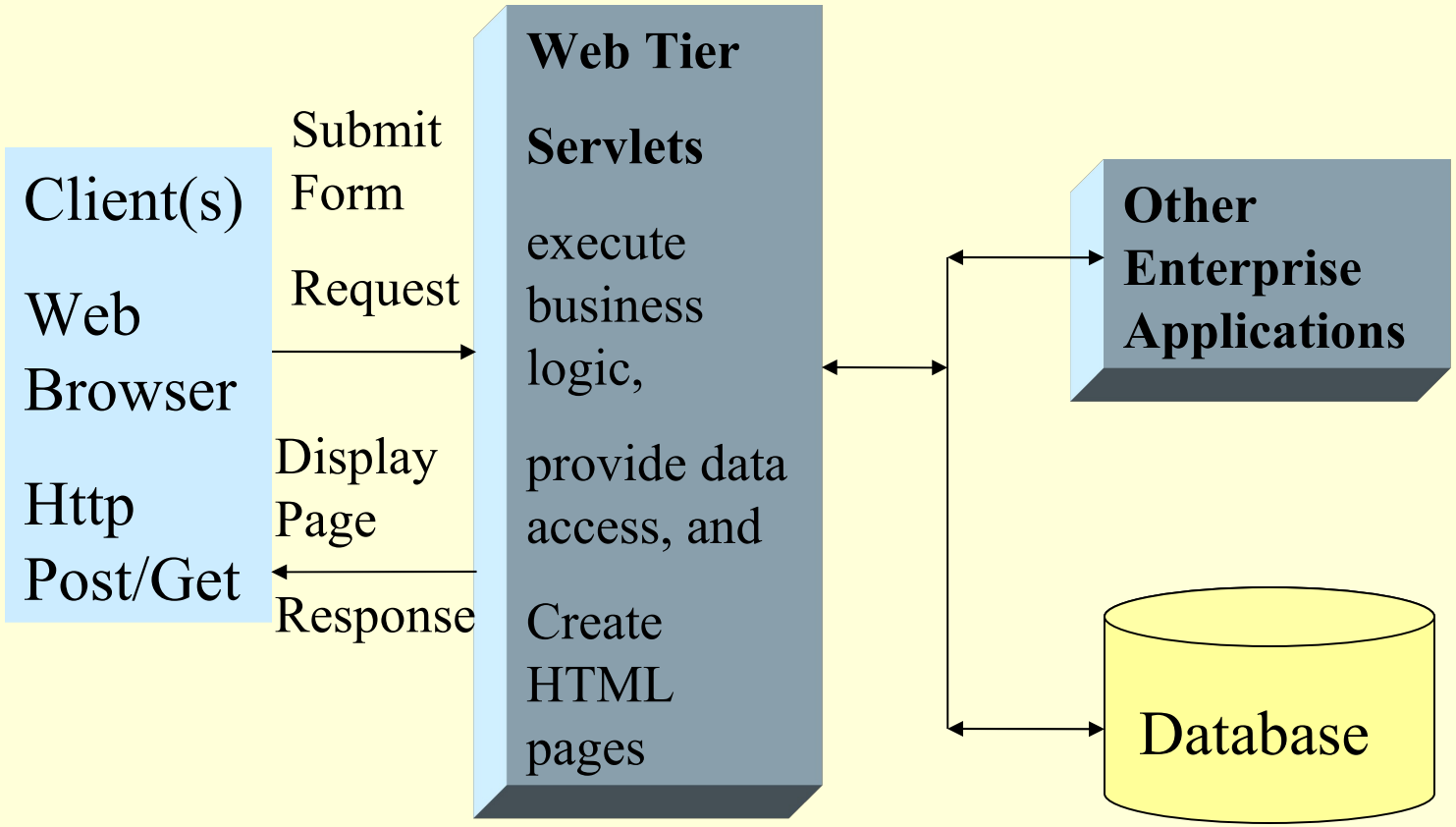
We can find both Dispatcher and Factory patterns in this example. This approach makes it possible to create a unified API for client – server communications. Any service (including new, unknown design time services) can be requested by a client without code change.

Servlet's Initialization and "web.xml"

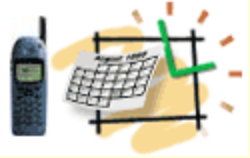
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app id="WebApp">
  <display-name>JavaSchoolApp</display-name>
  <context-param>
    <param-name>sqlLocation</param-name>
    <param-value>sql</param-value>
    <description>Location of SQL statements relative to WEB-INF</description>
 </context-param>
          <!-- Initialization code is in the ITSServletContextListener.java -- >
<listener>
    <listener-class>com.its.util.ITSServletContextListener</listener-class>
</listener>
<servlet>
   <servlet-name>action</servlet-name>
   <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
   <init-param>
      <param-name>config</param-name>
      <param-value>WEB-INF/struts-config.xml</param-value>
   </init-param>
</servlet>
```



Servlets Summary
Servlet-Centric Architecture



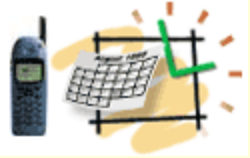
Java code in servlets mix together business and presentation layers



Servlet Summary

1. A *servlet extends functionality of a web server* similar to an applet in a browser.

2. Servlets usually:
   1.
   2.
   3.



Servlet Summary

1. A *servlet extends functionality of a web server* similar to an applet in a browser.

2. Servlets usually:
   1) Accept input data
   2) Run or *delegate* business logics
   3) Dynamically generate HTML response page

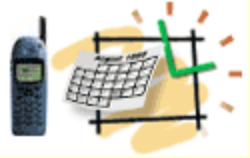
3. In Servlet-based architecture there is a mix …

JSP Concepts

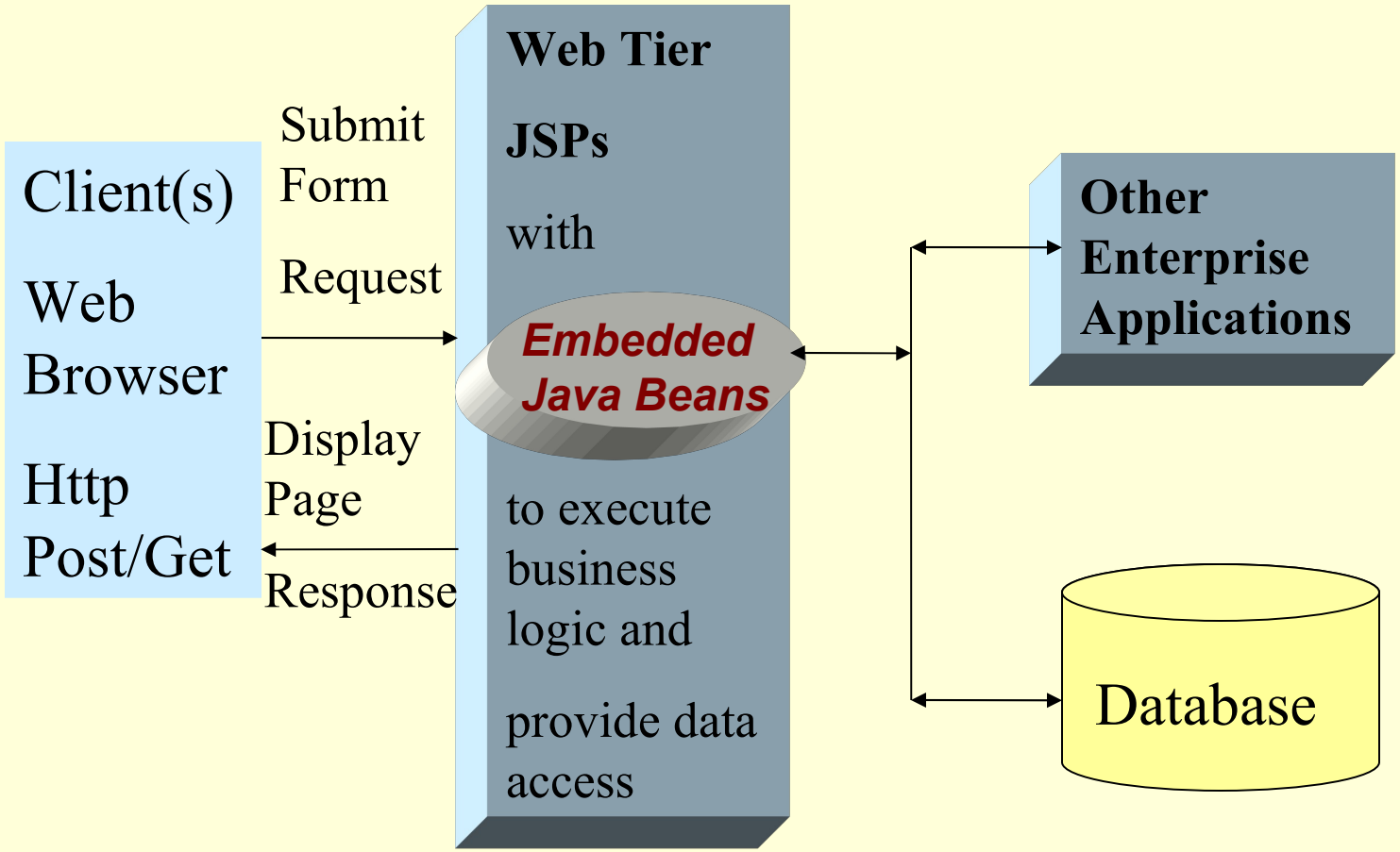
JSP, similar to Microsoft ASP Technology, simplifies web design. Web Designers who understand HTML can create most of the page using HTML tags and leave the rest for programmers

- **As compared to ASP, JSP is better language for dynamic part**
  - **Portable to multiple servers and operating systems**
- **As compared to pure servlets, JSP is more convenient to create HTML**
  - **Can use standard tools (e. g., HomeSite, DreamWeaver MX)**
  - **Model / view / controller paradigm**
  - **JSP programmers still need to know servlet programming**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD><TITLE>Simple date with JSP</TITLE></HEAD>
<BODY>
Today's date is <%= new java.util.Date( ) %>.
<P>
Regular HTML for rest of on-line store's Web page
</BODY>
</HTML>
```



MVC Model 1: JSP-to-JSP Page-Centric Architecture



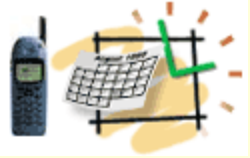
Interrelated JSP pages provide presentation, control, and business processing with scriplets and embedded Java beans encouraging “spaghetti” code in JSP.

JSP Custom Tag Library

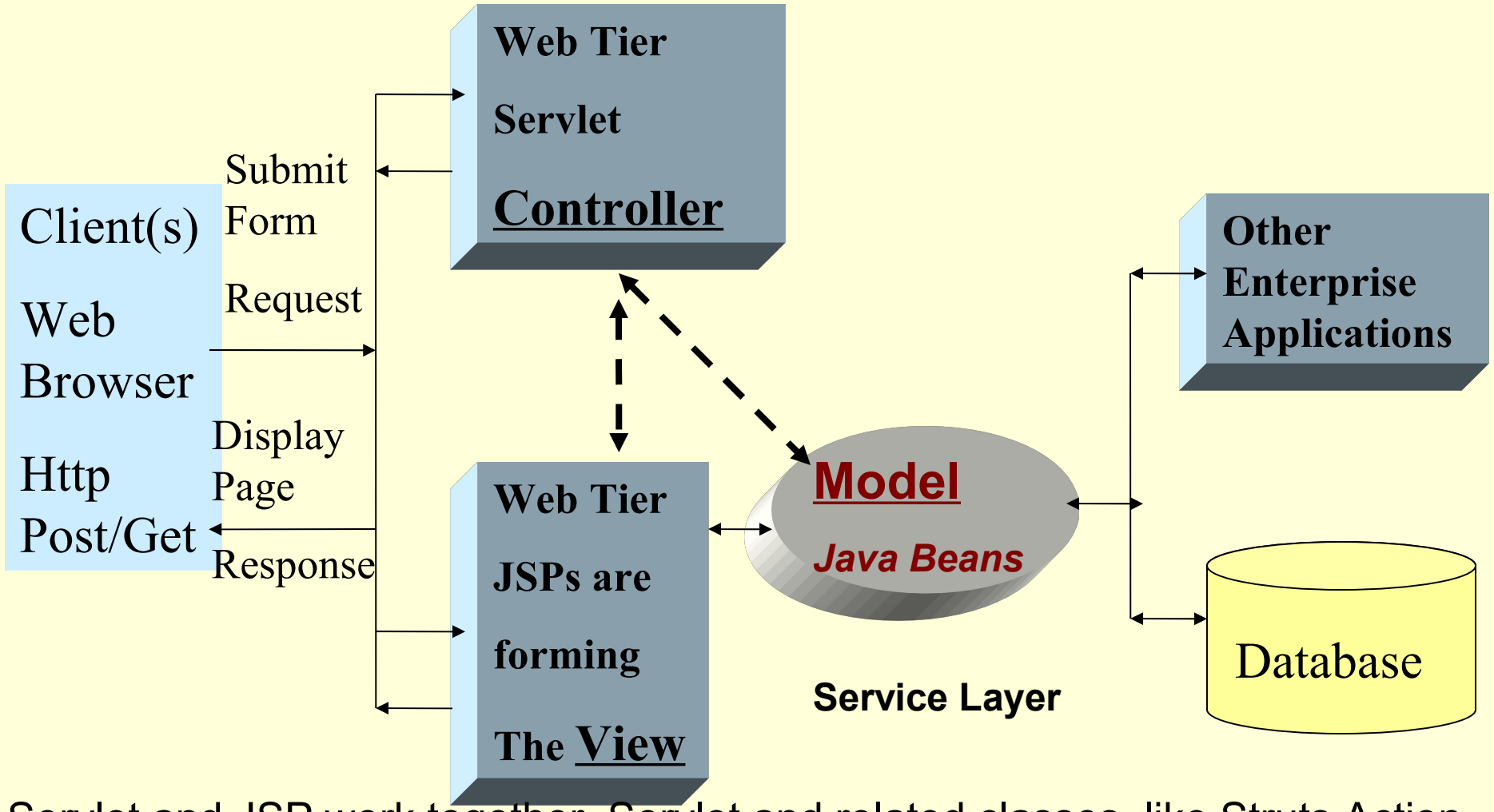
- **What is Tag Library?**

- **• Standard actions**
- **– In JSP, *actions* are elements that can create and access**
- **programming language objects and affect output**
- **– JSP spec 1.1 defines standard actions that can be used to interact with any Bean**
- **• useBean, setProperty, getProperty, include, forward,**
- **param, plugin**

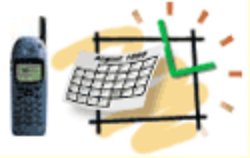
- **– JSP 1.1 and above supports development of reusable modules called *custom actions***
- **– A custom action is invoked by using a *custom tag* in a JSP page**
- **– A *tag library* is a collection of custom tags**



MVC Model 2 - Better Separation of Business and Presentation Layers



Servlet and JSP work together. Servlet and related classes, like Struts Action, control application logic and interact with services while JSP forms the presentation



Custom Tag Library Examples

- JSTL
- Struts
- WebLogic
- Displaytag
- Your own custom tag library

JSP/Tags Summary/Repetition

- JSP provides better ….
- Custom Tags ….

Web Container Components Servlet – JSP – Practice

Open Dynamic Web Application Project in Eclipse

Create JSP - index.jsp

Remember what is JSP?

– Allows some separation of Java and HTML code

• Using JavaBeans as "backend" helpers

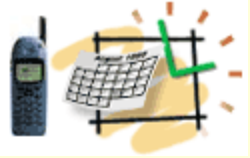
– JSP scriplets provide access to Java in HTML

Create Servlet – train.ServletExample

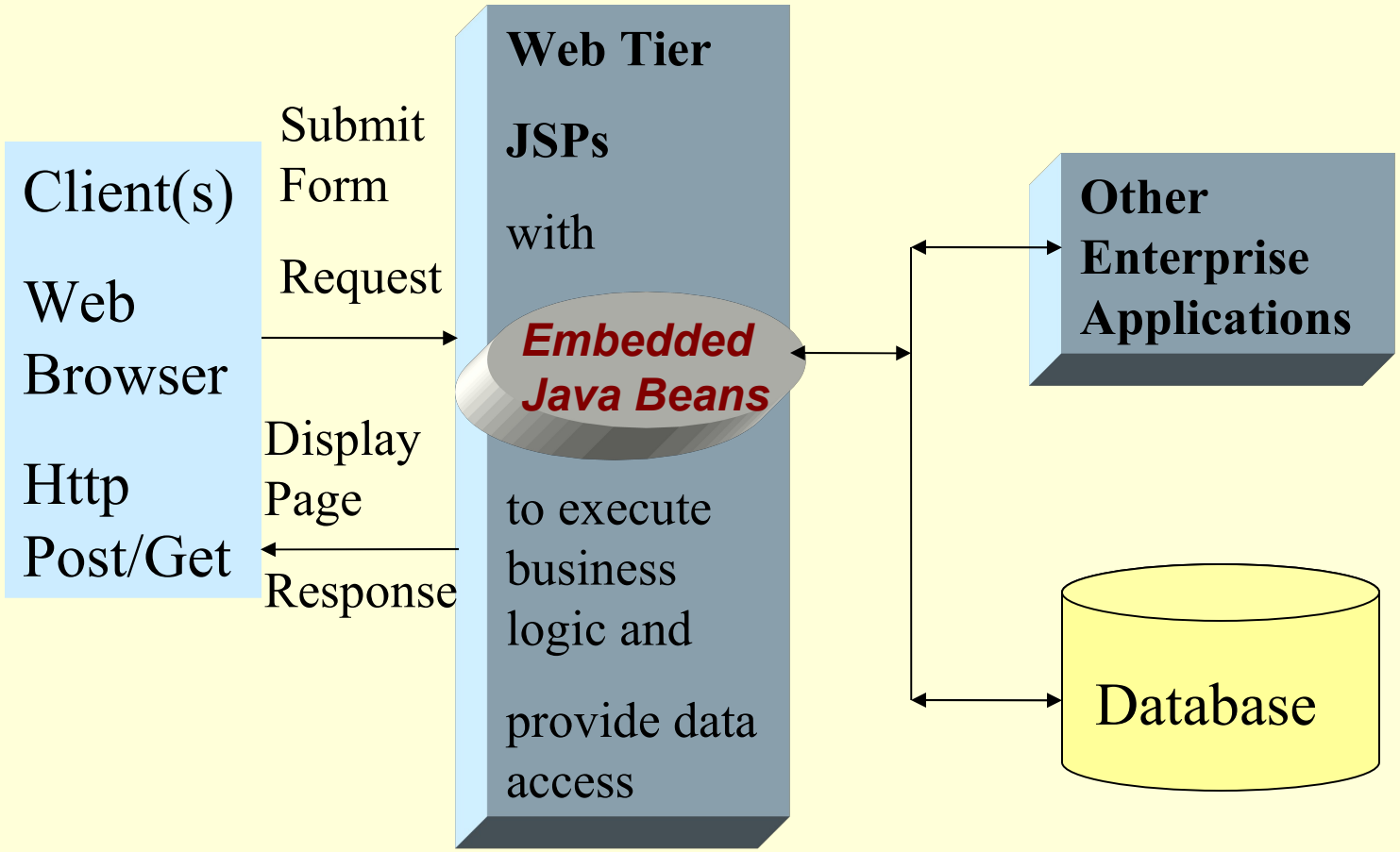
Remember what is servlet?

– Standard Java class to handle an HTTP request and response

– Often mixes Java and HTML code



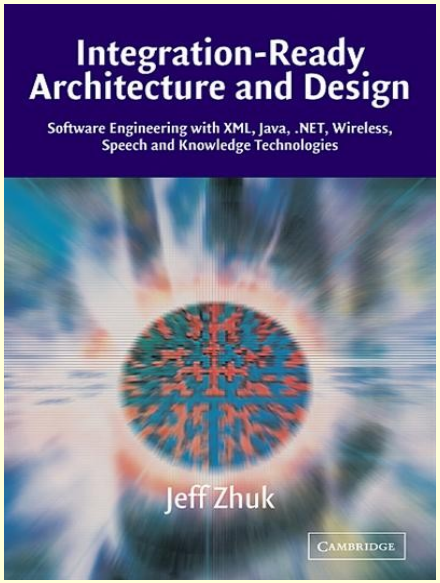
MVC Model 1: JSP-to-JSP Page-Centric Architecture



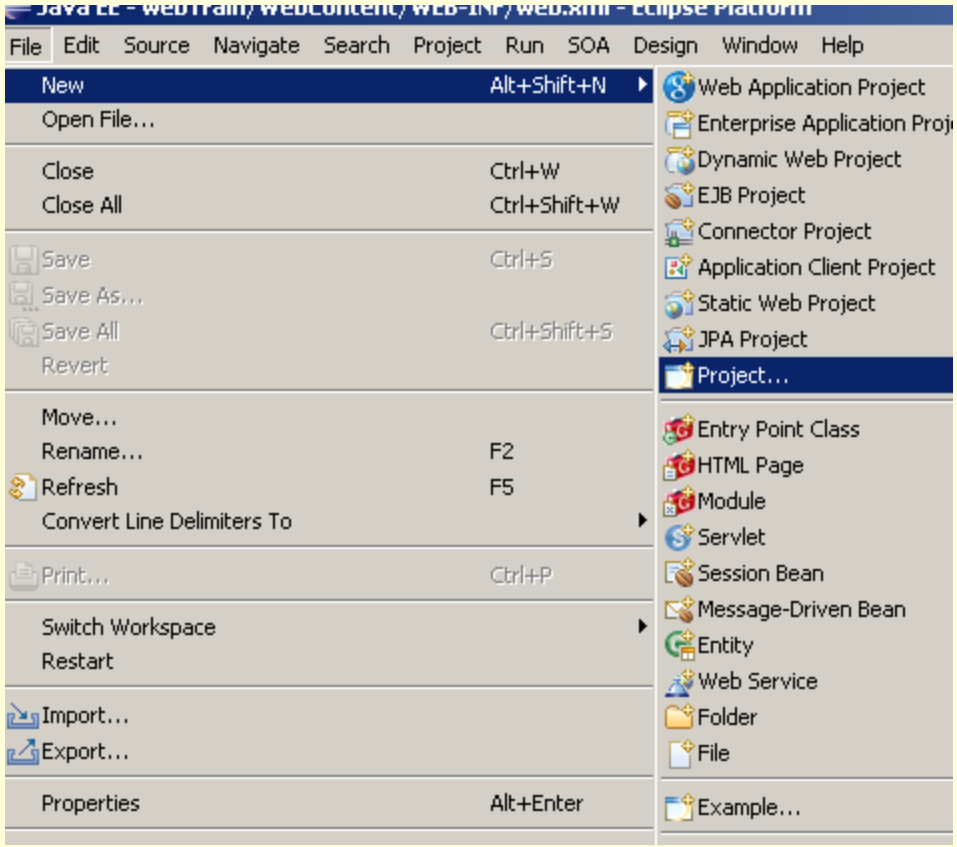
Interrelated JSP pages provide presentation, control, and business processing with scriplets and embedded Java beans encouraging “spaghetti” code in JSP.

JSP/Tags Summary/Repetition

- JSP provides better ….
- Custom Tags ….
- Practice ...

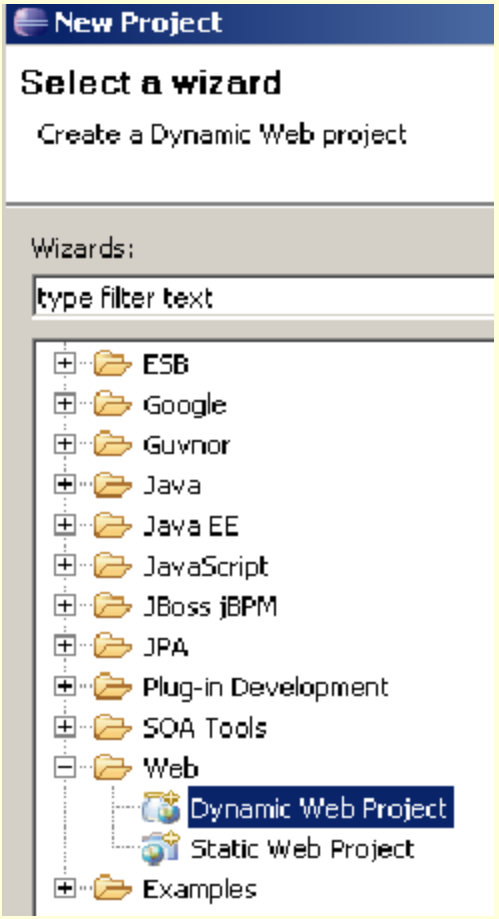


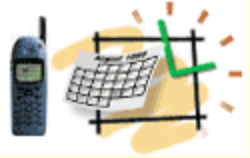
Open NEW Dynamic Web Application Project in Eclipse



Project name: webTrain

NEW – Project -
Web – Dynamic Web Project

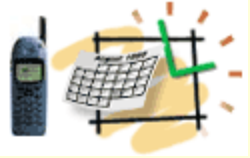




Create index.jsp

WebContent right mouse click – NEW JSP file – index.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD><TITLE>Simple date with JSP</TITLE></HEAD>
<BODY>
Today's date is <%= new java.util.Date( ) %>.
<P>
Regular HTML for rest of on-line store's Web page
</BODY>
</HTML>
```



Deploy Your Application

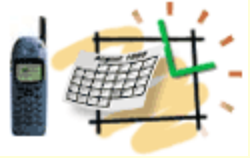
- Find the webTrain directory in the Eclipse workspace
- Copy the WebContent directory to c:/uop/apps

Zip the content of the folder (without the folder itself) and rename the compressed file into "yourName.war"

Email this file to me and I will Copy the file "yourName.war" to the deployment directory, for example, Tomcat/Webapps

Point the Browser to the following URLs to test your application:

http://yourServer:8080/yourName/index.jsp



Servlet Code Planning

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AbstractServlet extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

     //Get HTTP headers, cookies, session and form data from the request
     //Apply business logic, e.g., user authorization
     //Fill outgoing "response" to specify the HTTP response
     //including status code, headers, cookies, session data,
     // send output HTML page back to a client

   }
}
```

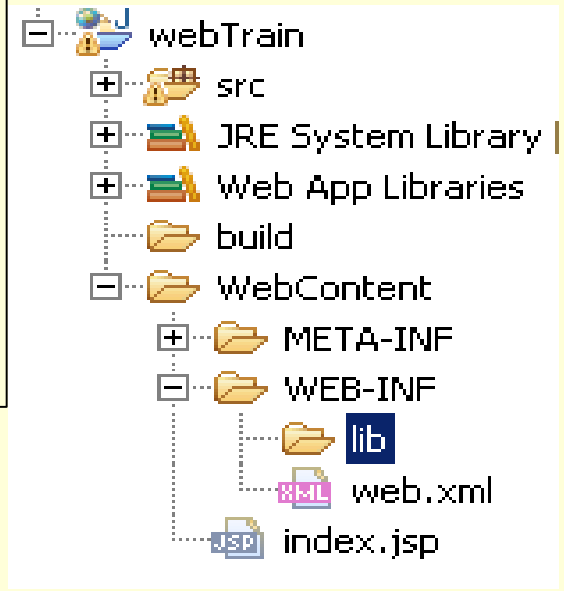
Prepare Eclipse to Servlets

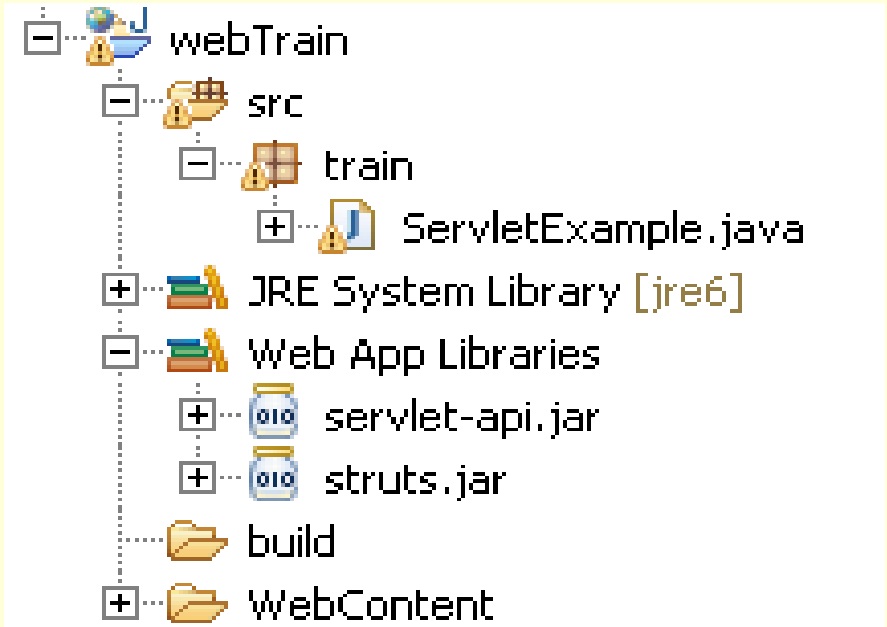
1.**Provide the Servlet Library**

Place **servlet-api.jar** in the Eclipse - WEB-INF/lib

Right mouse click on the jar – Add to Build Path

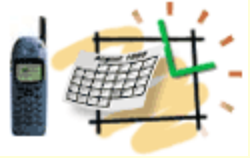
2. Create the **train** package under the **src** and create the **ServletExample** class in this package



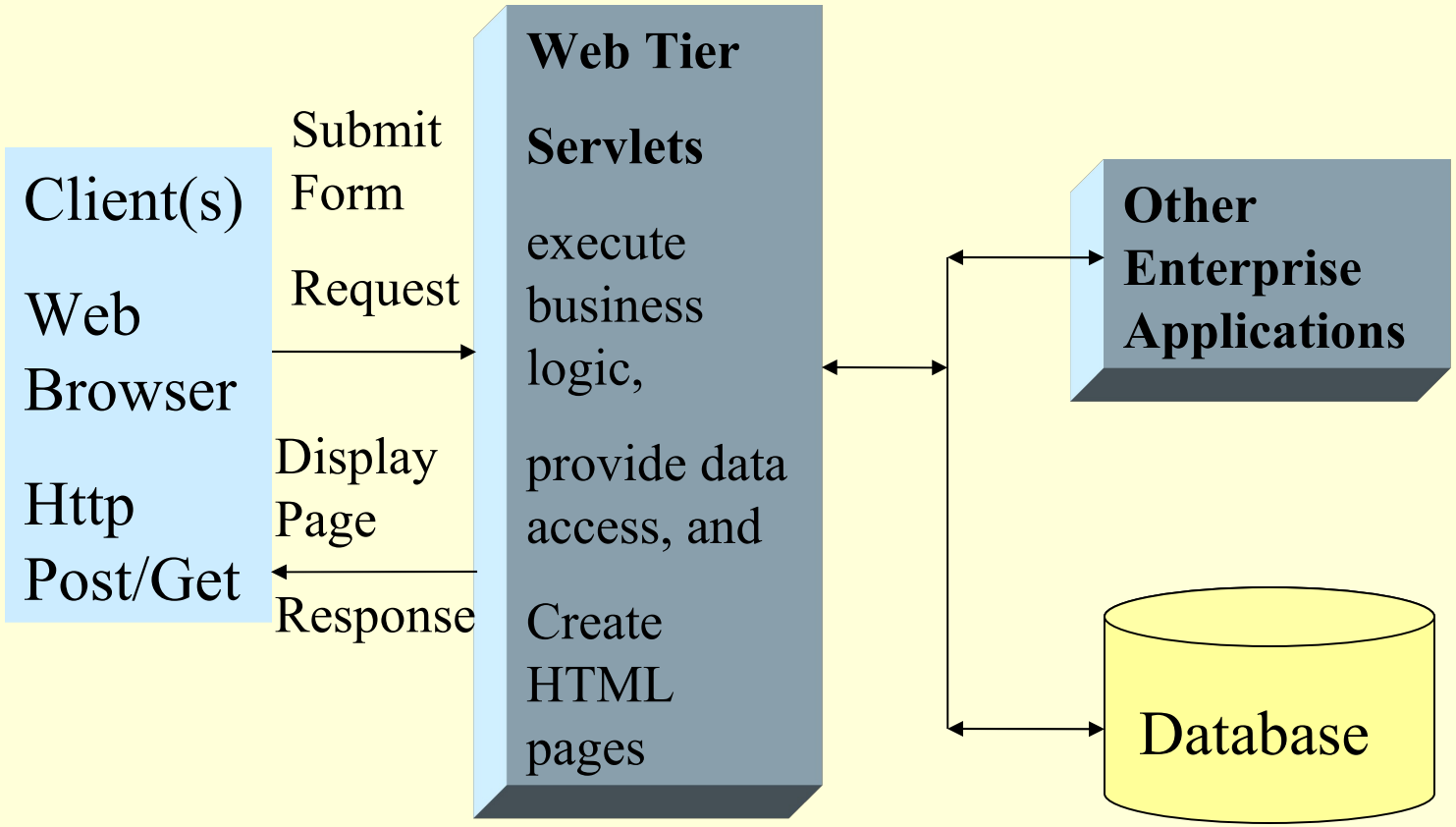


Servlet's Initialization and "web.xml"

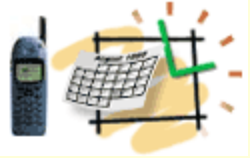
```
<?xml version="1.0" encoding="UTF-8"?>
... Keep existing code generated by Ecllipse... and add after the <web-app .....
<web-app id="WebApp">
    <display-name>yourName</display-name>
   <context-param>
     <param-name>dsName</param-name>
     <param-value>jeffcoTrain</param-value>
   </context-param>

   <servlet>
     <servlet-name>action</servlet-name>
     <servlet-class>train.ServletExample</servlet-class>
   </servlet>

   <servlet-mapping>
     <servlet-name>action</servlet-name>
     <url-pattern>*.do</url-pattern>
   </servlet-mapping>
   <session-config>
     <session-timeout>30</session-timeout>
   </session-config>
</web-app>
```



Servlets Summary
Servlet-Centric Architecture



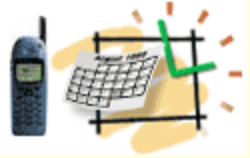
Java code in servlets mix together business and presentation layers



Servlet Summary

1. A *servlet extends functionality of a web server* similar to an applet in a browser.

2. Servlets usually:
   1.
   2.
   3.

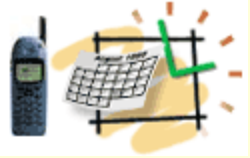


Servlet Summary

1. A *servlet extends functionality of a web server* similar to an applet in a browser.

2. Servlets usually:
   1) Accept input data
   2) Run or *delegate* business logics
   3) Dynamically generate HTML response page

3. In Servlet-based architecture there is a mix …



Deploy Your Application

- Make sure that your code was compiled with no read spots
- Find the webTrain directory in the Eclipse workspace
- Copy build/classes into WebContent/WEB-INFO
- Copy the WebContent directory to c:/uop/apps
- Zip the content of the folder (without the folder itself) and rename the compressed file into “yourName.war”
- Deploy the file “yourName.war” to the deployment directory, for example, Tomcat/Webapps

Point the Browser to the following URLs to test your application:

http://yourServer:8080/yourName/index.jsp

http://yourServer:8080/yourName/ServletExample.do