JSP

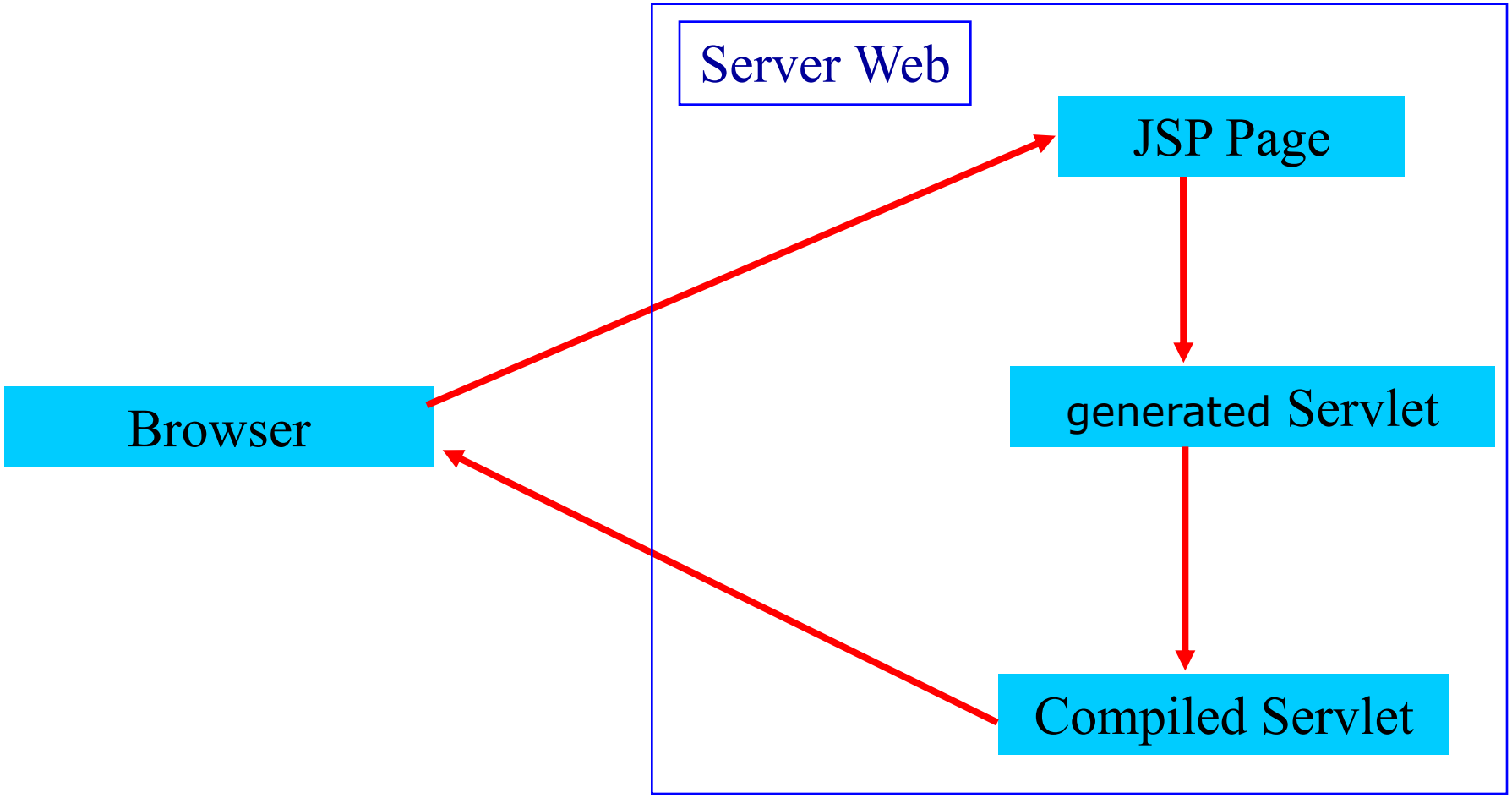
Basic Elements

For a Tutorial, see:
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSPIntro.html

Simple.jsp

```
<html>
 <body>
    <% out.println(“Hello World”); %>
  </body>
</html>
```

JSP Lifecycle



JSP nuts and bolts

Syntactic elements:

<%@ directives %>

<%! declarations %>

<% scriptlets %>

<%= expressions %>

<jsp:actions/>

<%-- Comment --%>

Implicit Objects:

- request
- response
- pageContext
- session
- application
- out
- config
- page

JSP nuts and bolts

Syntactic elements:

<%@ directives %> → Interaction with the CONTAINER

<%! declarations %> → In the initialization of the JSP

<% scriptlets %> → In the service method

<%= expressions %> → In the service method

<jsp:actions/>

Scriptlets

A scriptlet is a block of Java code executed during the request-processing time.

In Tomcat all the scriptlets gets put into the service() method of the servlet. They are therefore processed for every request that the servlet receives.

Scriptlet

Examples :

```
<% z=z+1; %>
```

```
<%
        // Get the Employee's Name from the request
        out.println("<b>Employee: </b>" +
        request.getParameter("employee"));
        // Get the Employee's Title from the request
        out.println("<br><b>Title: </b>" +
        request.getParameter("title"));
  %>
```

Expressions

An expression is a shorthand notation that sends the evaluated Java expression back to the client (in the form of a String).

Examples :

```
<%= getName() %>
```

```
<%@ page import=java.util.* %>
Sono le <%= new Date().toString(); %>
```

Expressions

```
<html><body>
 <%! String nome=“pippo” %>
<%! public String getName() {return nome;} %>
<H1>
Buongiorno
<%= getName() %>
</H1>
 </body></html>
```

Declarations

A declaration is a block of Java code used to:

define class-wide variables and methods in the generated servlet.

They are initialized when the JSP page is initialized.

<%! DECLARATION %>

Examples:

<%! String nome="pippo"; %>

<%! public String getName() {return nome;} %>

Directives

A directive is used as a message mechanism to:

pass information from the JSP code to the container

Main directives:

page

include (for including other STATIC resources at compilation time)

taglib (for including custom tag libraries)

Directives

<%@ DIRECTIVE {attributo=valore} %>

main attributes:

<%@ page language=java session=true %>

<%@ page import=java.awt.*,java.util.* %>

<%@ page isThreadSafe=false %>

<%@ page errorPage=URL %>

<%@ page isErrorPage=true %>

Standard actions

Standard action are tags that affect the runtime behavior of the JSP and the response sent back to the client.

Look like single-tag HTML. The purpose is to make the JSP source look more like a standard HTML compare to a JSP source which is full with scriplets (raw java code)

<jsp:include page="URL" />

For including STATIC or DYNAMIC resources at request time

<jsp:forward page="URL" />

What is a Java bean?

A bean is a Java class that:

- Provides a public no-argument constructor
- Implements java.io.Serializable (for sending it the client browser), read through session
- Has Set/get methods for properties

Standard actions involving beans

<jsp:useBean id="name" class="fully_qualified_pathname" scope="{page|request|session|application}" />

<jsp:setProperty name="nome" property="value" />

<jsp:getProperty name="nome" property="value" />

usebean1.jsp, viewpagebean.jsp

<%@include@%> or <jsp:include> ?

When should I use a JSP <%@include@%> directive, and when should I use a <jsp:include> action?

A JSP <%@include@%> directive (for example, <%@ include file="myfile.jsp" @%>) includes literal text "as is" in the JSP page and is not intended for use with content that changes at runtime. The include occurs only when the servlet implementing the JSP page is being built and compiled.

The <jsp:include> action allows you to include either static or dynamic content in the JSP page. Static pages are included just as if the <%@include@%> directive had been used. Dynamic included files, though, act on the given request and return results that are included in the JSP page. The include occurs each time the JSP page is served.

See also

http://java.sun.com/blueprints/qanda/web_tier/index.html#directive

<%-- Comment --%> or <!-- Comment --> ?

When should I use JSP-style comments instead of HTML-style comments?

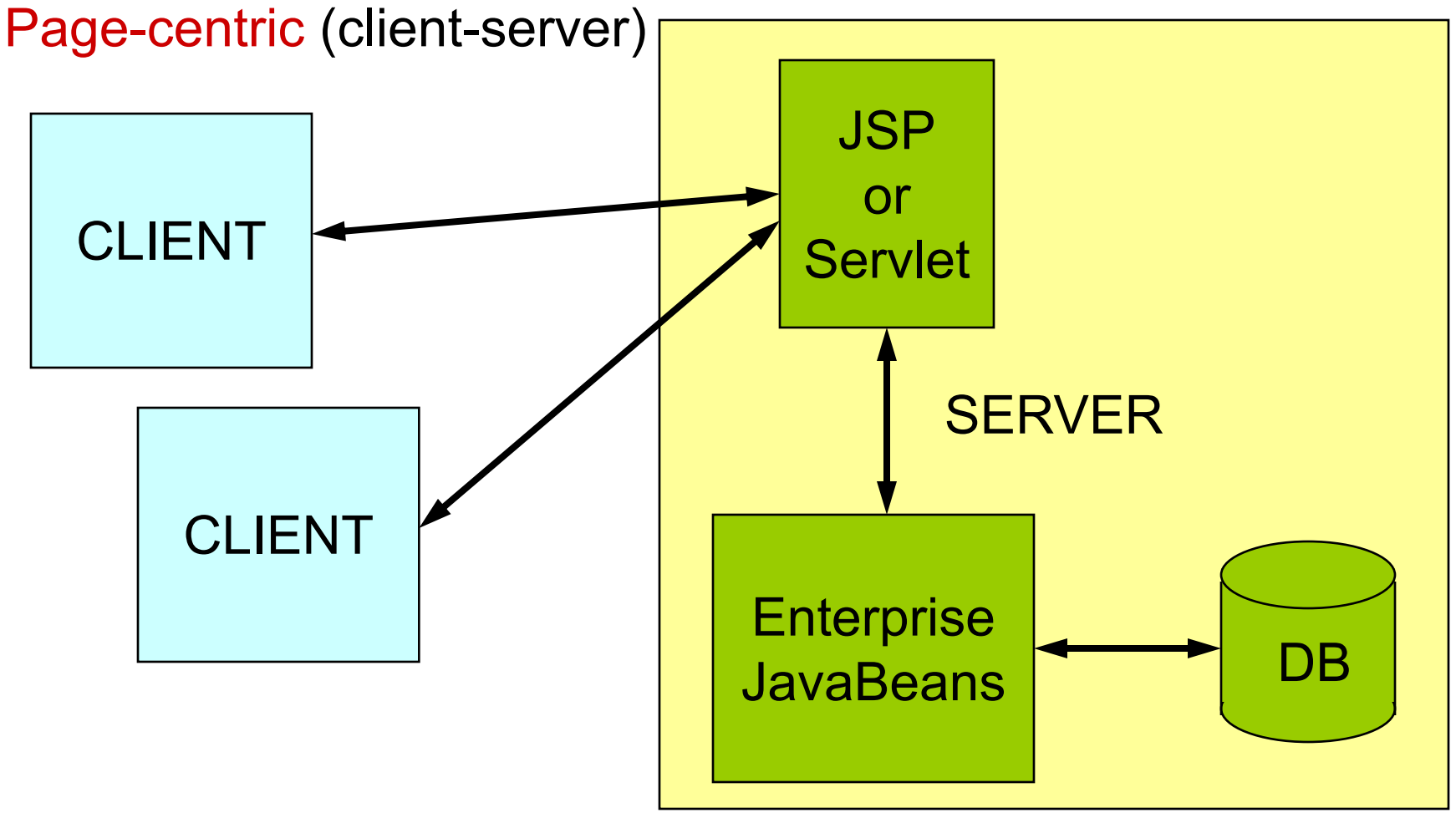
- **Always use JSP-style comments unless you specifically want the comments to appear in the HTML that results from serving a JSP page.**
- **JSP-style comments are converted by the JSP page engine into Java comments in the source code of the servlet that implements the JSP page. Therefore, JSP-style comments don't appear in the output produced by the JSP page when it runs. HTML-style comments pass through the JSP page engine unmodified. They appear in the HTML source passed to the requesting client.**
- **JSP-style comments do not increase the size of the document that results from the JSP page, but are useful to enhance the readability of the JSP page source, and to simplify debugging the servlet generated from the JSP page.**

(taken from:
http://java.sun.com/blueprints/qanda/web_tier/index.html#comments

JSP

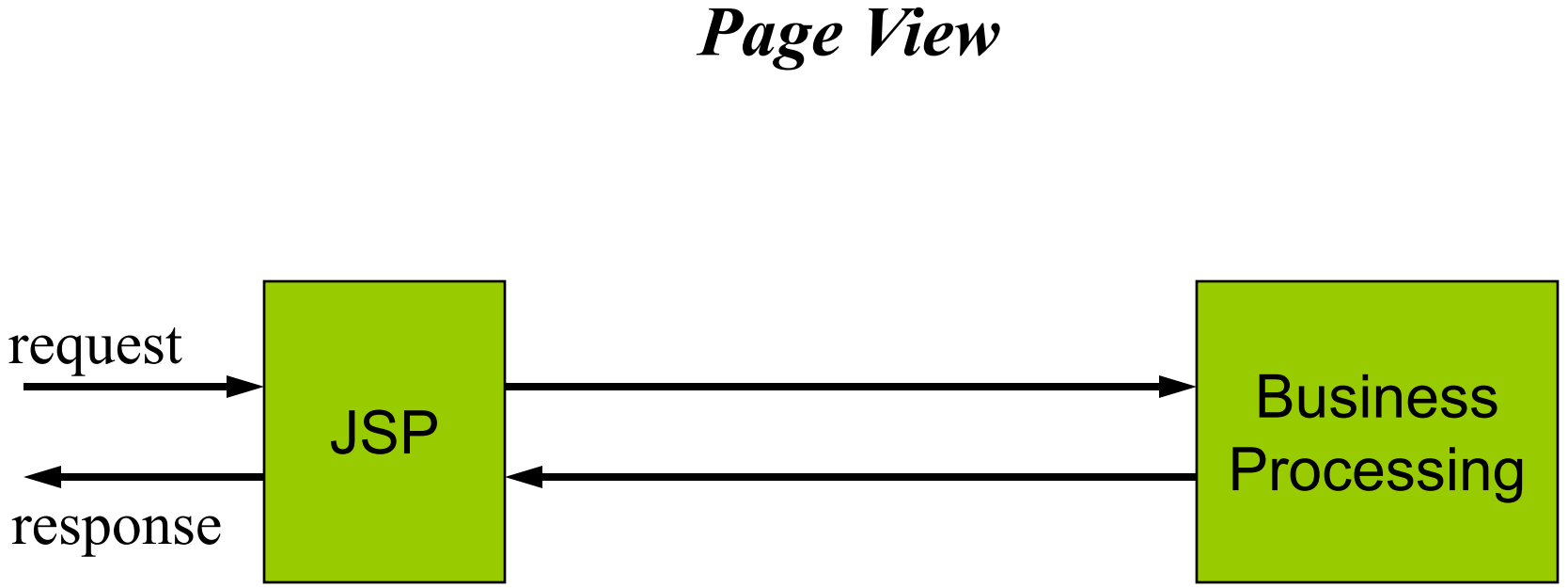
Common patterns

Common JSP patterns



Common JSP patterns

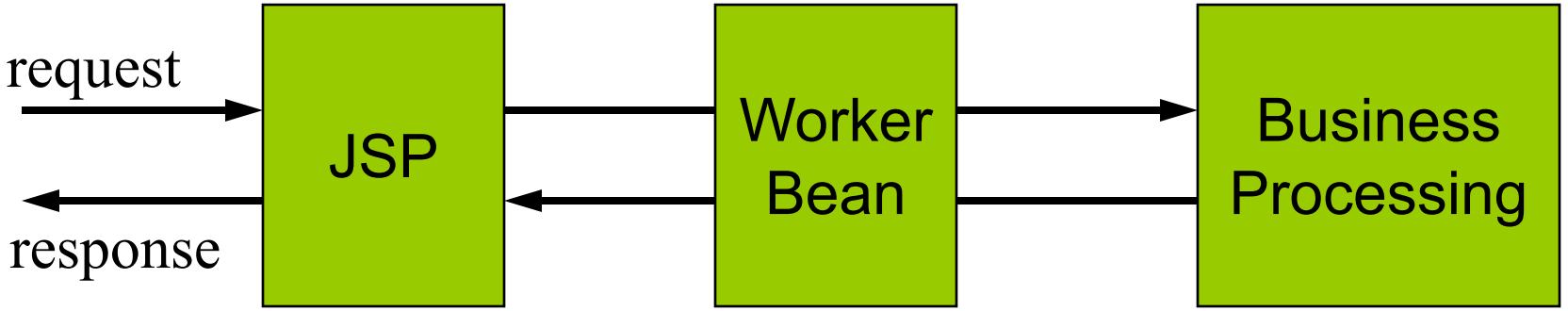
Page-centric 1 (client-server)



Common JSP patterns

Page-centric 2 (client-server)

Page View with Bean



Common JSP patterns

Dispatcher (n-tier)

Mediator - View