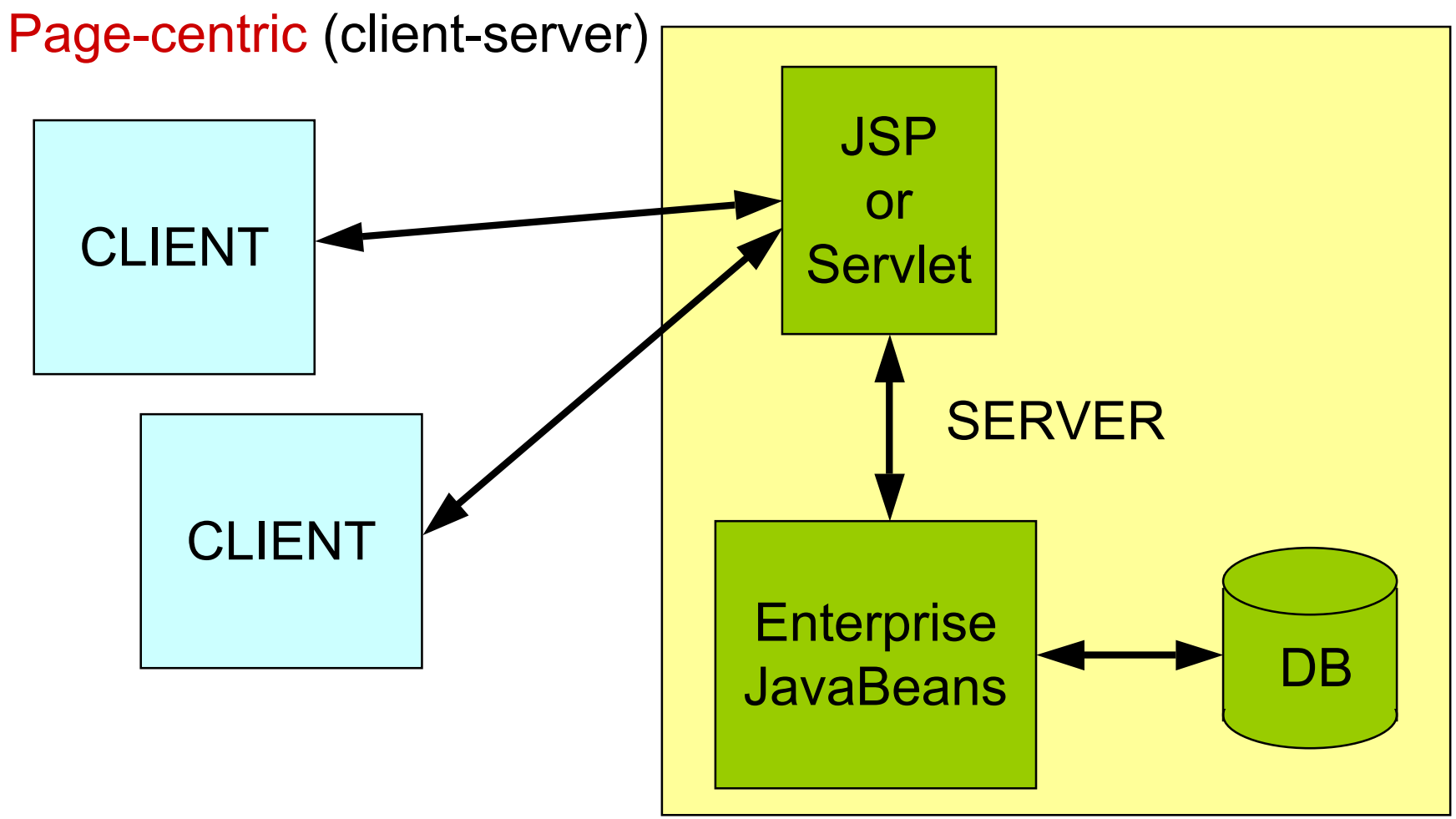
JSP

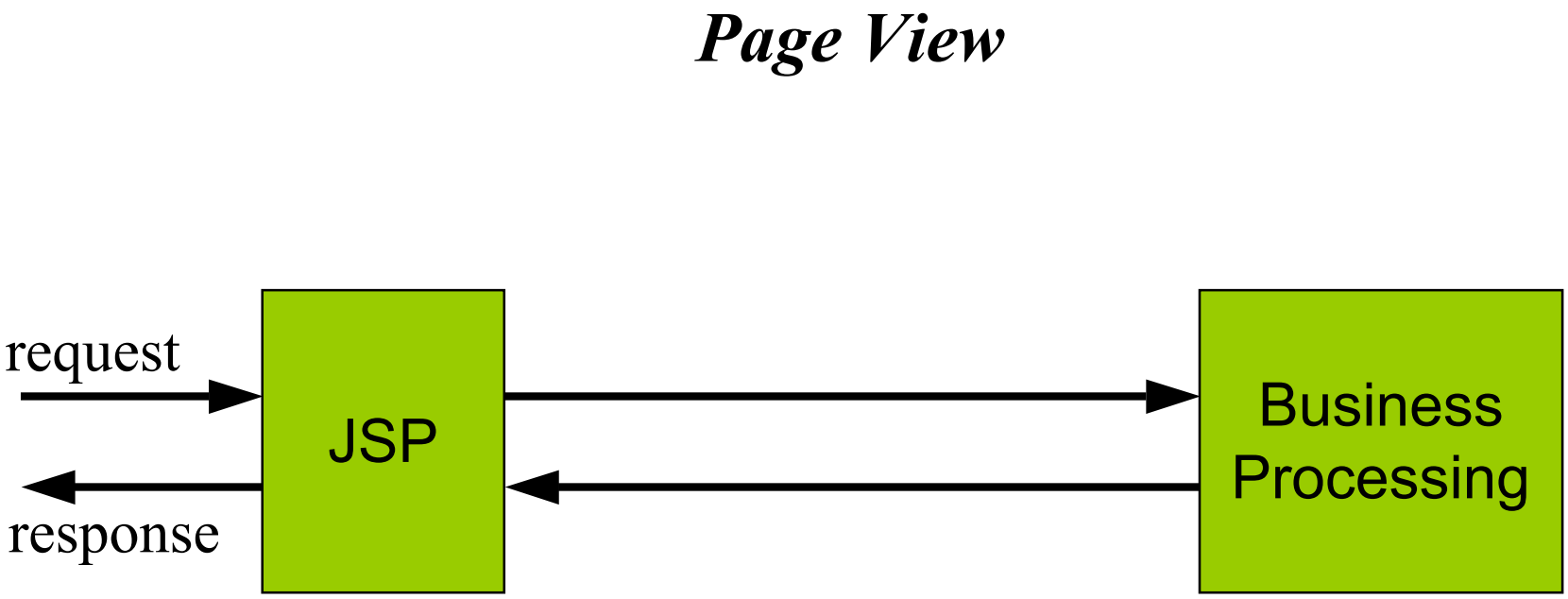
Common patterns

Common JSP patterns



Common JSP patterns

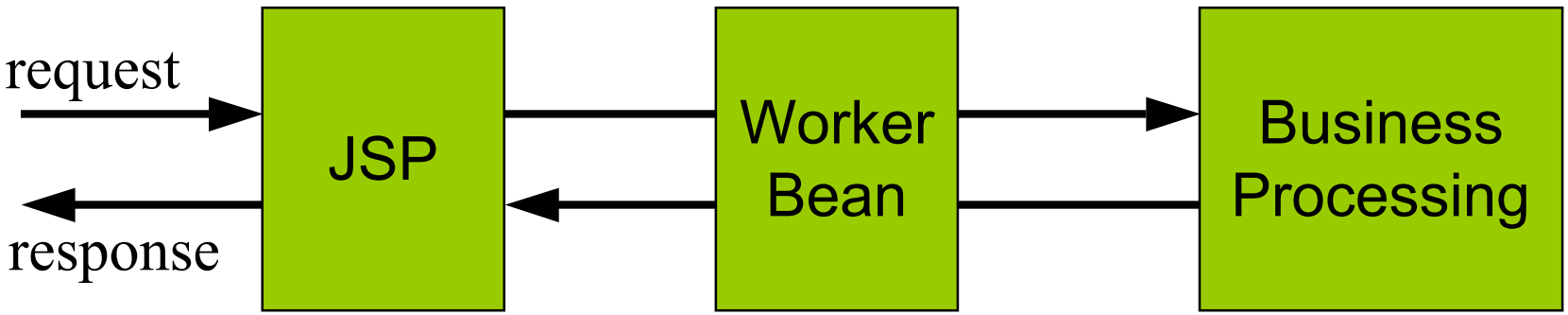
Page-centric 1 (client-server)



Common JSP patterns

Page-centric 2 (client-server)

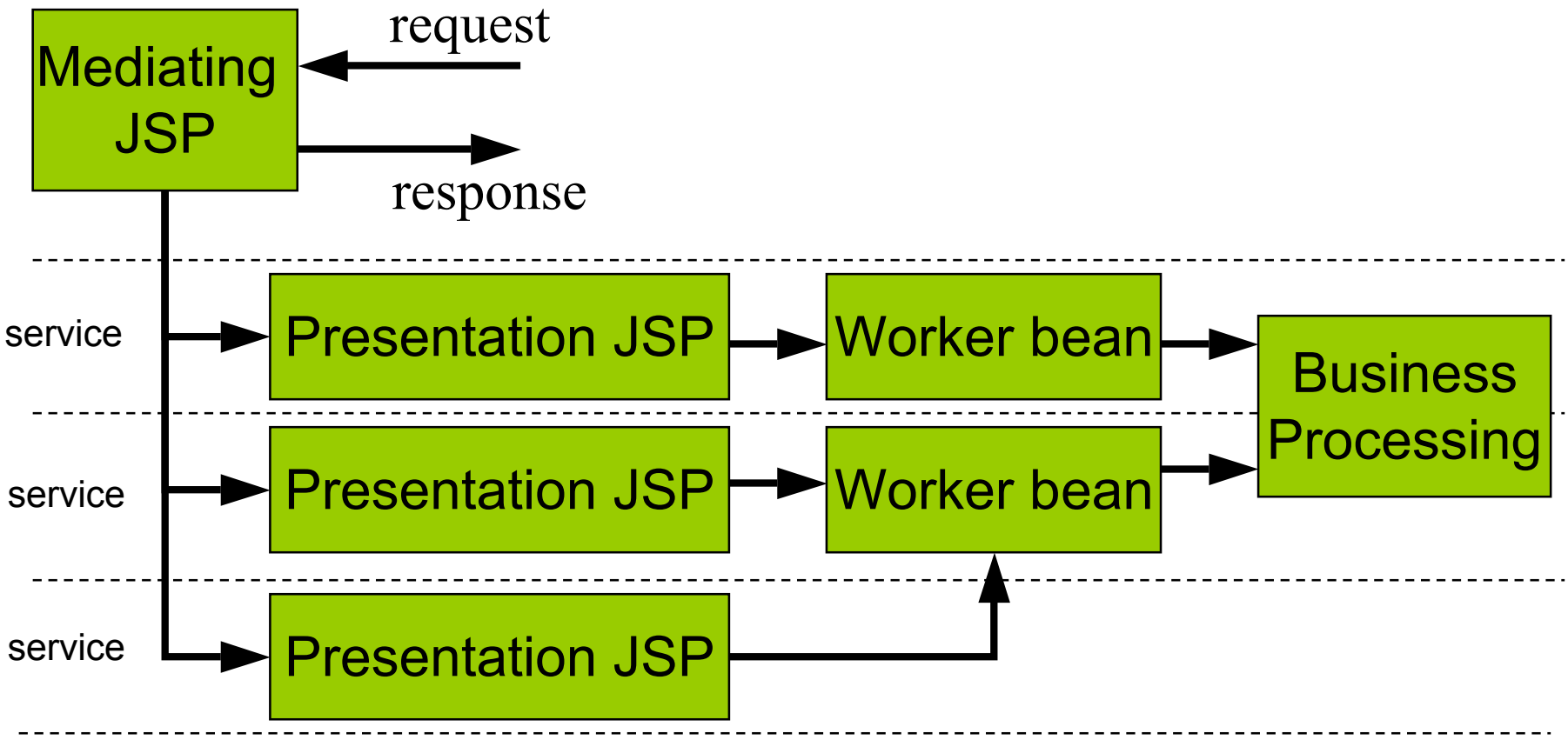
Page View with Bean



Common JSP patterns

Dispatcher (n-tier)

Mediator - View



SERVLETS:
Dispatching, monitoring, filtering

Dispatching

```
RequestDispatcher dispatch =
        cntx.getRequestDispatcher("/SecondServlet");
dispatch.forward(req,res);
```

```
RequestDispatcher dispatch =
        cntx.getRequestDispatcher("/SecondServlet");
dispatch.include(req,res);
```

Dispatching example

```
package servlets;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletContext;
import javax.servlet.RequestDispatcher;

public class SecondServlet extends HttpServlet  {
    public void doGet(HttpServletRequest req,HttpServletResponse res)
   throws IOException,ServletException {
      Printer out=res.getWriter();
      System.out.println("Second Servlet Called");
    }
 }
```

Dispatching example

```
package servlets;
    import javax.servlet.http.HttpServletRequest;
    import javax.servlet.http.HttpServletResponse;
    import javax.servlet.http.HttpServlet;
    import javax.servlet.ServletConfig;
    import javax.servlet.ServletContext;
    import java.io.IOException;
    import javax.servlet.ServletException;
    import javax.servlet.ServletContext;
    import javax.servlet.RequestDispatcher;

 public class FirstServlet extends HttpServlet  {
   public void doGet(HttpServletRequest req,HttpServletResponse res)
   throws IOException,ServletException {
      Printer out=res.getWriter();
      out.println("First Servlet Called");
      ServletConfig config = getServletConfig();
      ServletContext cntx = config.getServletContext();
      RequestDispatcher dispatch =
            cntx.getRequestDispatcher("/SecondServlet");
      dispatch.forward(req,res);
   }
}
```

Dispatching example

```
<servlet>
<servlet-name>FirstServlet</servlet-name>
<servlet-class>servlets.FirstServlet</servlet-class>
</servlet>

<servlet>
<servlet-name>SecondServlet</servlet-name>
<servlet-class>servlets.SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/firstservlet/*</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>SecondServlet</servlet-name>
<url-pattern>/SecondServlet/*</url-pattern>
</servlet-mapping>
```

Monitoring Servlets Lifecycle

Web context	Initialization and Destruction	ServletContextListener	ServletContextEvent
	Attribute added, removed, or replaced	ServletContextAttributeListener	ServletContextAttributeEvent
Session	Creation, invalidation, activation, passivation, and timeout	HttpSessionListenerHttpSessionActivationListener	HttpSessionEvent
	Attribute added, removed, or replaced	HttpSessionAttributeListener	HttpSessionBindingEvent
Request	A servlet request has started being processed by Web components	ServletRequestListener	ServletRequestEvent
	Attribute added, removed, or replaced	ServletRequestAttributeListener	ServletRequestAttributeEvent

Monitoring Servlets Lifecycle - Example

```
/* File : ApplicationWatch.java */
import javax.servlet.ServletContextListener;
import javax.servlet.ServletContextEvent;
public class ApplicationWatch implements ServletContextListener {
public static long applicationInitialized = 0L;
/* Application Startup Event */
public void contextInitialized(ServletContextEvent ce) {
    applicationInitialized = System.currentTimeMillis(); }
/* Application Shutdown Event */
public void contextDestroyed(ServletContextEvent ce) {}
}
```

Monitoring Servlets Lifecycle - Example

```
/* File : SessionCounter.java */
import javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;
public class SessionCounter implements HttpSessionListener {
private static int activeSessions = 0;
/* Session Creation Event */
public void sessionCreated(HttpSessionEvent se) {
    activeSessions++; }
/* Session Invalidation Event */
public void sessionDestroyed(HttpSessionEvent se) {
    if(activeSessions > 0) activeSessions--; }
public static int getActiveSessions() { return activeSessions; }
}
```

Monitoring Servlets Lifecycle - Example

```
<!-- Web.xml -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
    Application 2.3//EN" "http://java.sun.com/j2ee/dtds/web-
    app_2.3.dtd">
<web-app>
<!-- Listeners -->
<listener>
    <listener-class> com.stardeveloper.web.listener.SessionCounter
    </listener-class>
</listener>
<listener>
    <listener-class> com.stardeveloper.web.listener.ApplicationWatch
    </listener-class>
</listener>
</web-app>
```

Scope Objects

Web context	ServletContext	Web components within web context servlet.getServletConfig().getServletContext
Session	HttpSession	Web components handling requests that belong to a session
Request	ServletRequest	Web component handling the request
Page	PageContext	Web component in the JSP page

Main Methods:
Object getAttribute(String name)
void setAttribute(String name, Object o)
Enumeration getAttributeNames()

AOP

The programming paradigms of aspect-oriented programming (AOP), and aspect-oriented software development (AOSD) attempt to aid programmers in the separation of concerns, specifically cross-cutting concerns, as an advance in modularization.

Logging and authorization offer two examples of crosscutting concerns:
a logging strategy necessarily affects every single logged part of the system. Logging thereby crosscuts all logged classes and methods.

Same is true for authorization.

Filters (javax.servlet.filter)

Other classes that preprocess/postprocess request/response

A filter is an object than perform filtering tasks on either the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the doFilter method. Every Filter has access to a FilterConfig object from which it can obtain its initialization parameters, a reference to the ServletContext which it can use, for example, to load resources needed for filtering tasks.

Filters are configured in the deployment descriptor of a web application

Examples that have been identified for this design are

1) **Authentication Filters**
2) **Logging and Auditing Filters**
3) **Image conversion Filters**
4) **Data compression Filters**
5) **Encryption Filters**
6) **Tokenizing Filters**
7) **Filters that trigger resource access events**
8) **XSL/T filters**
9) **Mime-type chain Filter**

http://java.sun.com/products/servlet/Filters.html

Filters

Filters are important for a number of reasons. First, they provide the ability to encapsulate recurring tasks in reusable units.

Second, filters can be used to transform the response from a servlet or a JSP page. A common task for the web application is to format data sent back to the client. Increasingly the clients require formats (for example, WML) other than just HTML.

Filters

Filters can perform many different types of functions.

* Authentication-Blocking requests based on user identity.
* Logging and auditing-Tracking users of a web application.
* Image conversion-Scaling maps, and so on.
* Data compression-Making downloads smaller.
* Localization-Targeting the request and response to a particular locale.
* XSL/T transformations of XML content-Targeting web application responses to more that one type of client.

These are just a few of the applications of filters. There are many more, such as encryption, tokenizing, triggering resource access events, mime-type chaining, and caching.

Filters

The filtering API is defined by the Filter, FilterChain, and FilterConfig interfaces in the javax.servlet package. You define a filter by implementing the Filter interface.

The most important method in this interface is doFilter, which is passed request, response, and filter chain objects. This method can perform the following actions:

1. Examine the request headers.
2. Customize the request object and response objects if needed
3. Invoke the next entity in the filter chain (configured in the WAR). The filter invokes the next entity by calling the doFilter method on the chain object (passing in the request and response it was called with, or the wrapped versions it may have created).

Filter example

```
import javax.servlet.*; import javax.servlet.http.*; import java.io.*;
public class LoginFilter implements Filter {
protected FilterConfig filterConfig;
public void init(FilterConfig filterConfig) throws ServletException
        {this.filterConfig = filterConfig; }
public void destroy() { this.filterConfig = null; }
public void doFilter(ServletRequest req, ServletResponse res,
  FilterChain chain) throws java.io.IOException, ServletException {
        String username = req.getParameter("j_username");
        if (isUserOk(username)) chain.doFilter(request, response);
        res.sendError(
        javax.servlet.http.HttpServletResponse.SC_UNAUTHORIZED);
    }
    // implement here isUserOk()…
}
```

Example

```
<filter id="Filter_1">
<filter-name>LoginFilter</filter-name>
<filter-class>LoginFilter</filter-class>
<description>Performs pre-login and post-login operation</description>
<</filter-id>

<filter-mapping>
<filter-name>LoginFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Filters and sessions

```
public void doFilter(ServletRequest req, ServletResponse res,
  FilterChain chain) throws java.io.IOException, ServletException {
        HttpSession session = req.getSession(false);
        if (null == session || !(Boolean)session.getAttribute("auth")) {
           if (isUserOk(req.getParameter("user")))
              session=req(.getSession(true);
              session.setAttribute("auth",new Boolean(true));
           } else res.sendError(
          javax.servlet.http.HttpServletResponse.SC_UNAUTHORIZED);
        } chain.doFilter(request, response);
    }
```

Filters and parameters

```
java.util.ArrayList userList=null;
public void init(FilterConfig fc) throws ServletException {
    BufferedReader in;
    this.filterConfig = fc;
    userList = new java.util.ArrayList();
    if ( fc != null ) {
        try {
             String filename = fc.getInitParameter("Users");
             in = new BufferedReader( new FileReader(filename));
         } catch ( FileNotFoundException fnfe) {
              writeErrorMessage();return;
         }
         String userName;
         try {
            while ( (userName = in.readLine()) != null )
                userList.add(userName);
         } catch (IOException ioe) {writeErrorMessage();return;}
    }
}
public void destroy() { this.filterConfig = null; userList = null; }
```

Filters and parameters

```
<filter id="Filter_1">
<filter-name>LoginFilter</filter-name>
<filter-class>LoginFilter</filter-class>
<description>Performs pre-login and post-login operation</description>
<init-param>
<param-name>Users</param-name>
<param-value>c:\mydir\Users.lst</param-value>
</init-param>
</filter-id>
```

Filter sequencing

```
<filter>
    <filter-name>Uncompress</filter-name>
    <filter-class>compressFilters.createUncompress</filter-class>
  </filter>
  <filter>
    <filter-name>Authenticate</filter-name>
    <filter-class>authentication.createAuthenticate</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>Uncompress</filter-name>
    <url-pattern>/status/compressed/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>Authenticate</filter-name>
    <url-pattern>/status/compressed/*</url-pattern>
  </filter-mapping>
```

Both Uncompress and Authenticate appear on the filter chain for servlets located at /status/compressed/*. The Uncompress filter precedes the Authenticate filter in the chain because the Uncompress filter appears before the Authenticate filter in the web.xml file.

JSP

Tag Extension

http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html

JSTL

Core tags

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

```
<c:set var="foo" scope="session" value="..."/>

${foo}
```

Area	Function	Tags	Prefix
Core	Variable support	remove set	c
	Flow control	choose when otherwise forEach forTokens if	
	URL management	import param redirect param url param	
	Miscellaneous	catch out	

See http://download.oracle.com/javaee/5/tutorial/doc/bnakh.html

JSTL - xml

XML tags

<%@ taglib
uri="http://java.sun.com/jsp/jstl/xml"
prefix="x" %>

Area	Function	Tags	Prefix
XML	Core	out parse set	x
	Flow control	choose when otherwise forEach if	
	Transformation	transform param	

```
<c:if test="${applicationScope:booklist == null}" >
     <c:import url="${initParam.booksURL}" var="xml" />
    <x:parse doc="${xml}" var="booklist" scope="application" />
</c:if>
<x:set var="abook"
    select="$applicationScope.booklist/
        books/book[@id=$param:bookId]" />
    <h2><x:out select="$abook/title"/></h2>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnakq.html

JSTL - sql

Area	Function	Tags	Prefix
Database	Setting the data source	setDataSource	sql
	SQL	query dateParam param transaction update dateParam param	

XML tags

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

```
<sql:setDataSource dataSource="jdbc/BookDB" />
<c:set var="bid" value="${param.Add}"/>
<sql:query var="books" >
    select * from PUBLIC.books where id = ?
    <sql:param value="${bid}" />
</sql:query>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnald.html

JSTL-fn

Area	Function	Tags
Functions	Collection length	length
	String manipulation	toUpperCase, toLowerCase
		substring, substringAfter, substringBefore
		trim
		replace
		indexOf, startsWith, endsWith, contains, containsIgnoreCase
		split, join
		escapeXml

function tags

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

```
<c:if test="${fn:length(param.username) > 0}" >
  <%@include file="response.jsp" %>
</c:if>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnalg.html

JSTL-fmt

Area	Function	Tags	Prefix
I18N	Setting Locale	setLocale requestEncoding	fmt
	Messaging	bundle message param setBundle	
	Number and Date Formatting	formatNumber formatDate parseDate parseNumber setTimeZone timeZone	

i18n tags

<%@ taglib
uri="http://java.sun.com/jsp/jstl/fmt"
prefix="fmt" %>

```
<h3><fmt:message key="Choose"/></h3>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnakw.html

JSP custom tag

Ideally, JSP pages should contain no code written in the Java programming language (that is, no expressions or scriptlets). Anything a JSP page needs to do with Java code can be done from a

custom tag

- ***Separation of form and function.***
- ***Separation of developer skill sets and activities.***
- ***Code reusability.***
- ***Clarified system design.***

a JSP custom tag

```
<%@ taglib uri="/hello" prefix="example" %>
<HTML><HEAD><TITLE>First custom tag</TITLE></HEAD>
<BODY>
This is static output
<p />
<i><example:hello>HELLO THERE</example:hello></i>
This is static output
</BODY>
</HTML>
```

hello.doStartTag()

hello.doEndTag()

a JSP custom tag

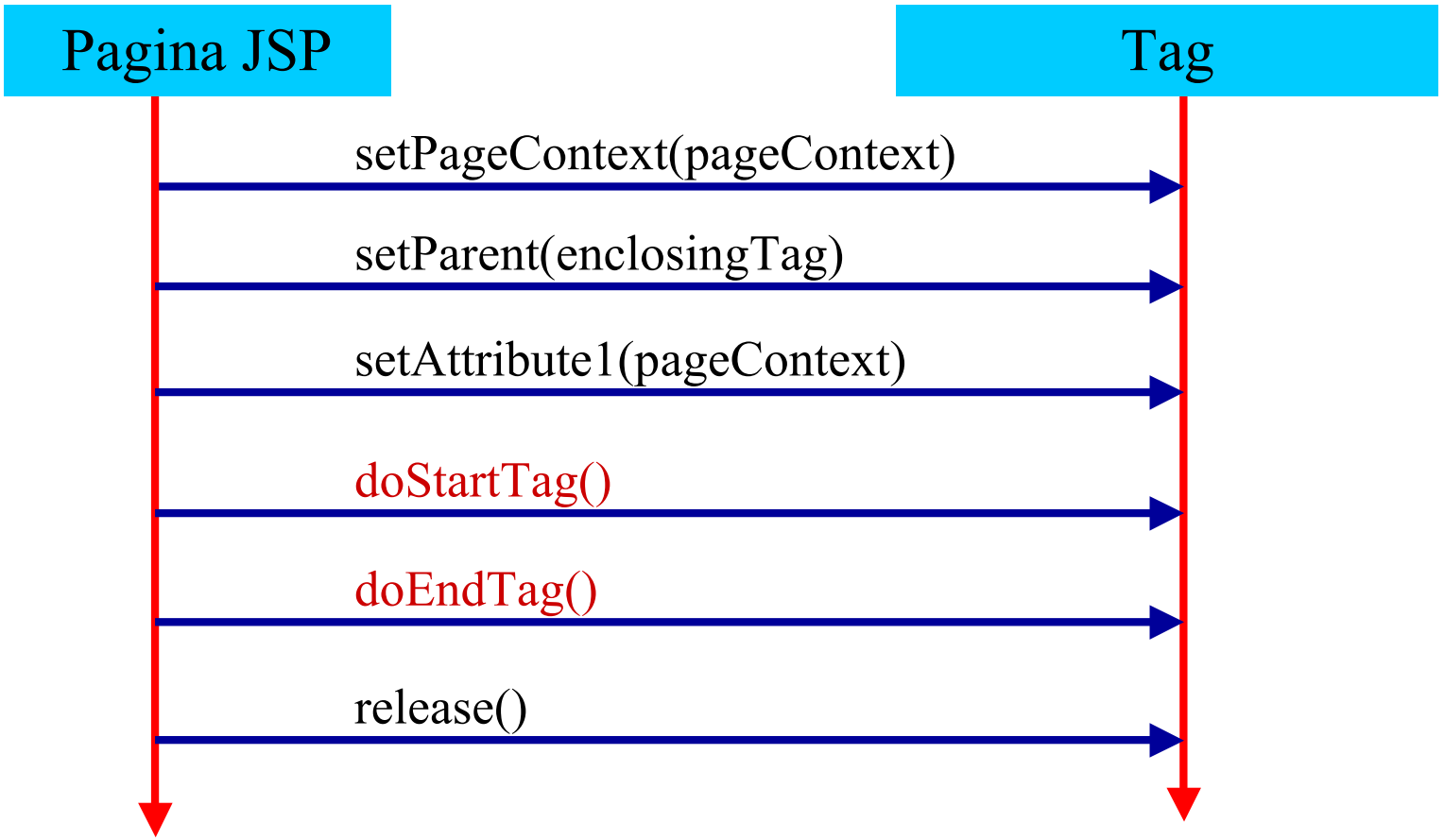
```
package jsptags;
import java.io.IOException;
import java.util.Date;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class HelloTag extends TagSupport {
  public int doStartTag() throws JspTagException {
    try {
      pageContext.getOut().write("Start tag found here<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
    return EVAL_BODY_INCLUDE;  // return SKIP_BODY;
  }
```

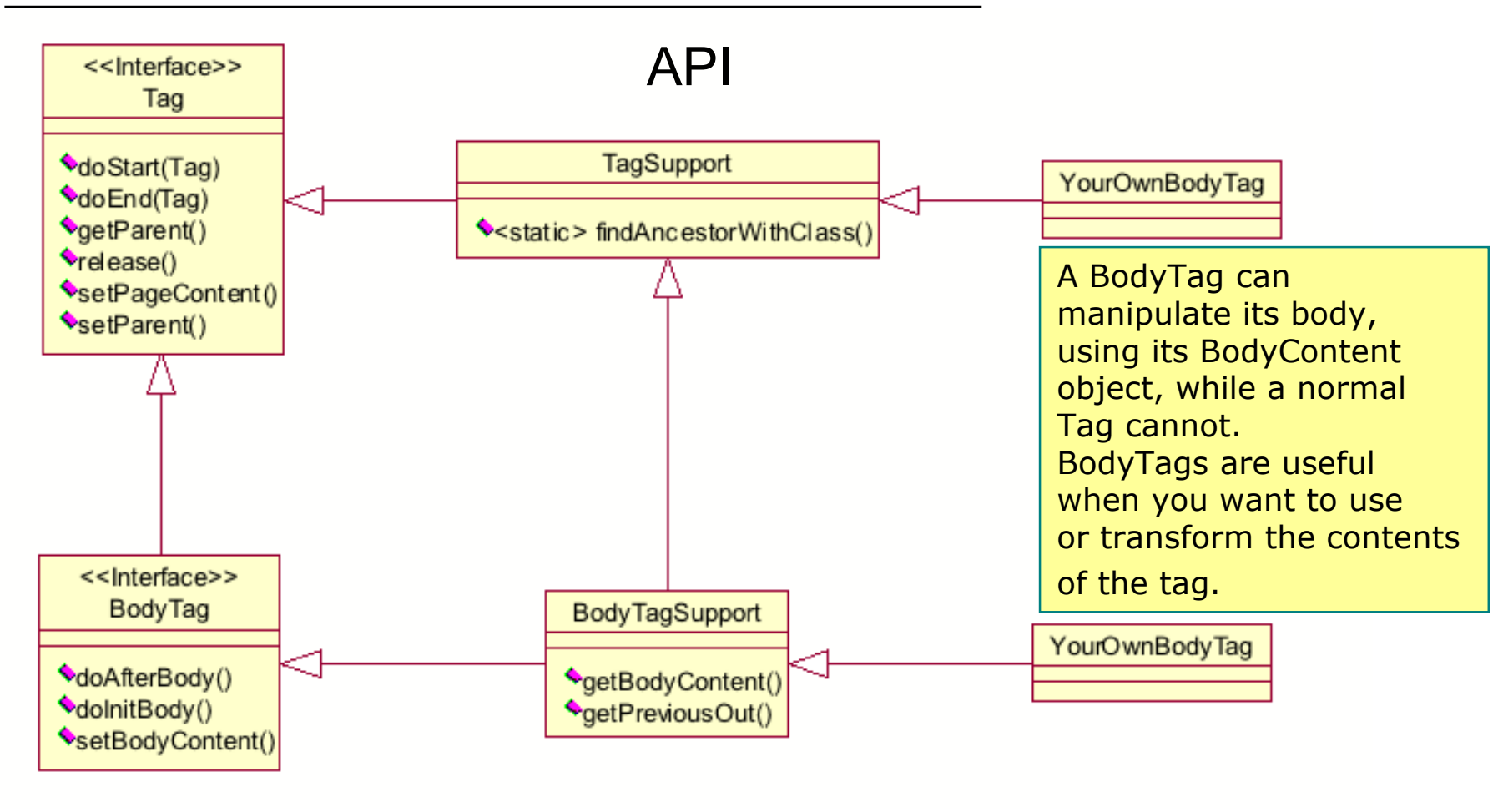
a JSP custom tag

```
…
public class HelloTag extends TagSupport {
…
 public int doEndTag() throws JspTagException {
    try {
      pageContext.getOut().write("End tag found<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
    return EVAL_PAGE;      // return SKIP_PAGE;
  }
}
```

Javax.servlet.jsp.tagext.Tag interface

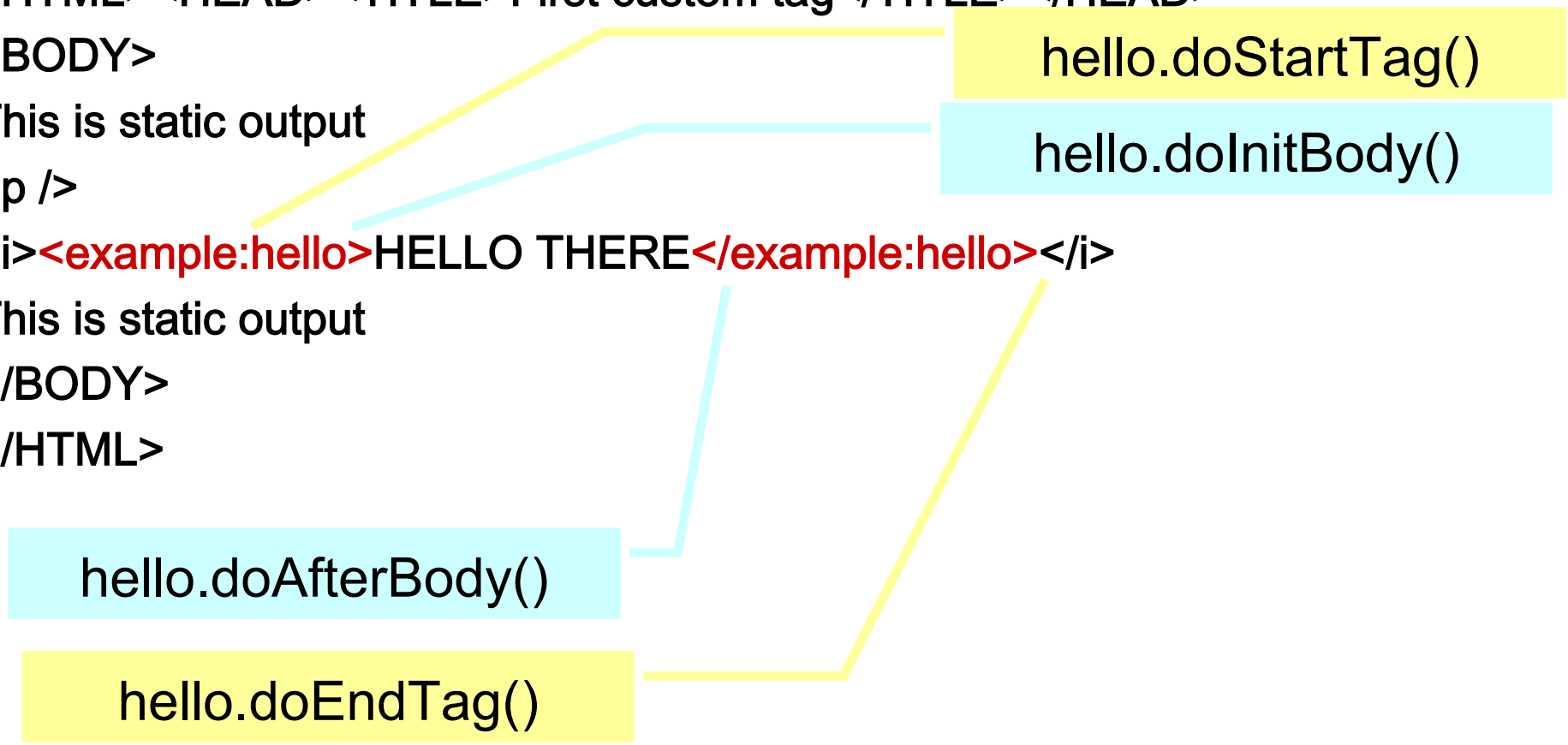


Class Diagram



a JSP custom tag

```
<%@ taglib uri="/hello" prefix="example" %>
<HTML><HEAD><TITLE>First custom tag</TITLE></HEAD>
<BODY>
This is static output
<p />
<i><example:hello>HELLO THERE</example:hello></i>
This is static output
</BODY>
</HTML>
```



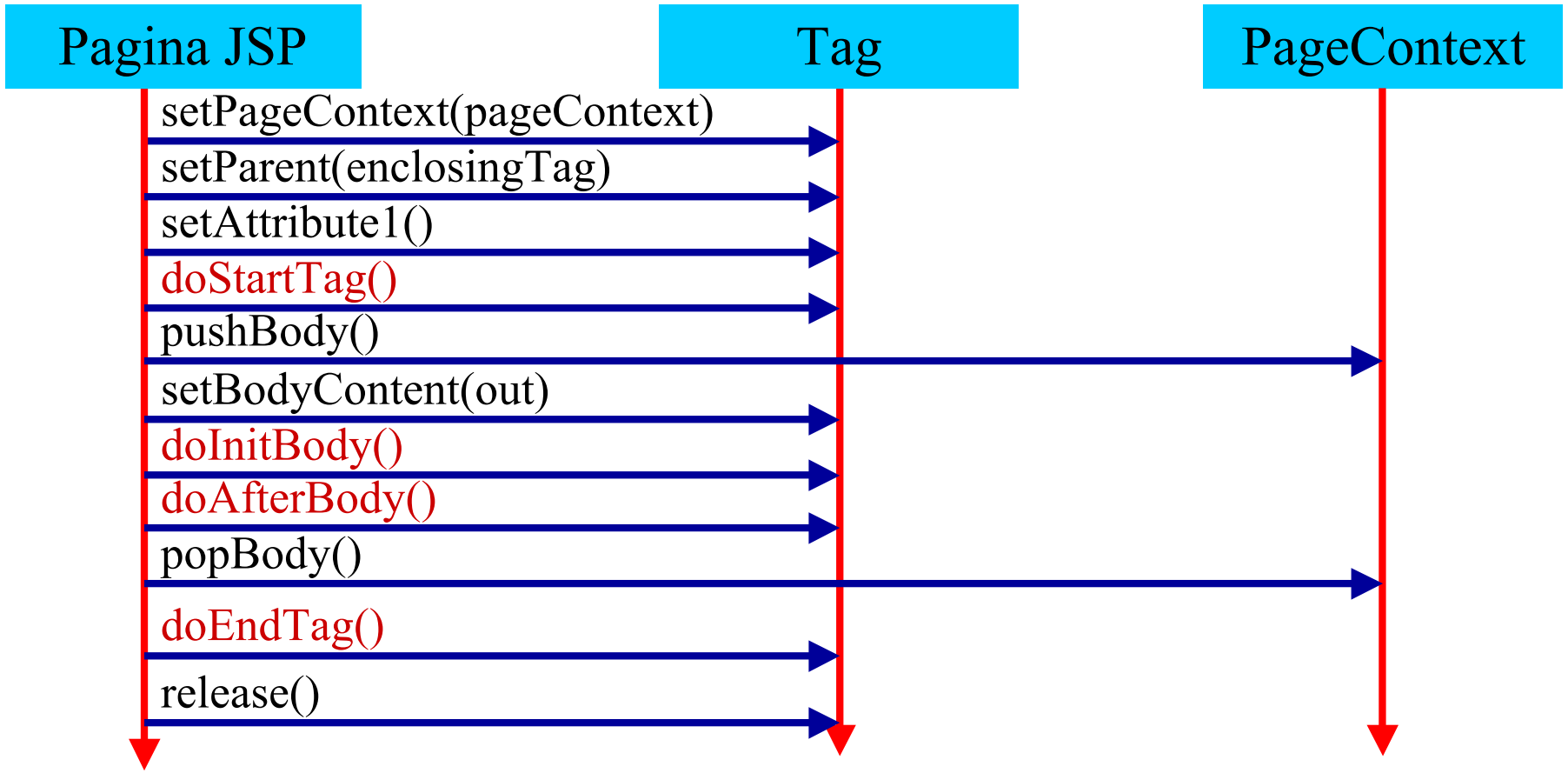
a JSP custom tag

```
package jsptags;
…
public class HelloTag extends BodyTagSupport {
  public int doStartTag() throws JspTagException {
    …
  }
public void doInitBody() throws JspTagException {
    try {
      pageContext.getOut().write("Init Body<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
  }
```

a JSP custom tag

```
 public int doAfterBody() throws JspTagException {
   try {
     pageContext.getOut().write("After Body<BR>");
   } catch (IOException e) {
     throw new JspTagException("Fatal error: could not write to JSP out");
   }
   return EVAL_BODY_TAG; // return SKIP_BODY;
 } */
 public int doEndTag() throws JspTagException {
    …
  }
}
```

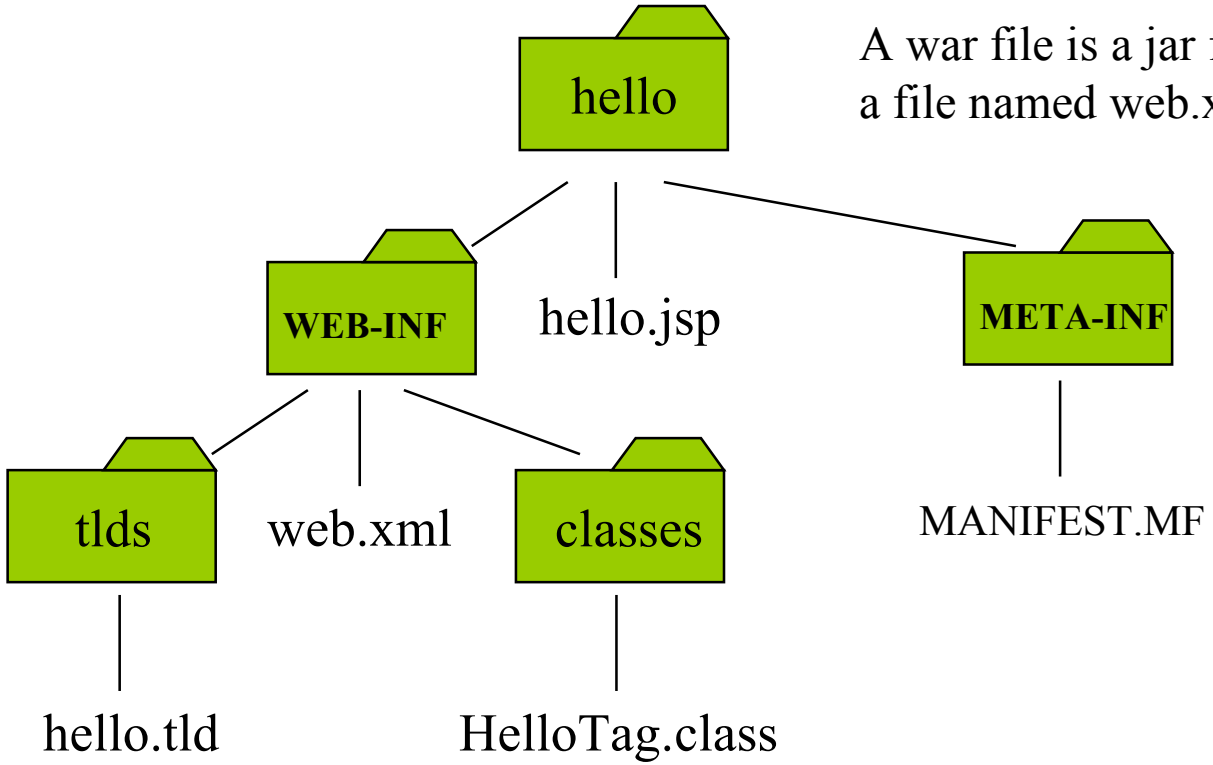
Javax.servlet.jsp.tagext.BodyTag interface



reversing body content

```
import java.io.IOException; import javax.servlet.jsp.*; import javax.servlet.jsp.tagext.*;
public class ReverseTag extends BodyTagSupport {
    public int doEndTag() throws JspTagException {
        BodyContent bodyContent = getBodyContent();
        if (bodyContent != null) {// Do nothing if there was no body content
            StringBuffer output = new StringBuffer(bodyContent.getString());
             output.reverse();
             try {
                 bodyContent.getEnclosingWriter().write(output.toString());
             } catch (IOException ex) {
                 throw new JspTagException("Fatal IO error");
             }
        }  return EVAL_PAGE;
    }
}
```

structure of the war file



A war file is a jar file with special directories and a file named web.xml in the WEB-INF directory

TLD

```
<?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>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>examples</shortname>
  <info>Simple example library.</info>
  <tag>
    <name>reverse</name>
    <tagclass>tagext.ReverseTag</tagclass>
    <bodycontent>JSP</bodycontent>
    <info>Simple example</info>
  </tag>
</taglib>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN'
    'http://java.sun.com/j2ee/dtds/web-app_2.2.dtd'>
<web-app>
 <display-name>tagext</display-name>
 <description>Tag extensions examples</description>
 <session-config>
   <session-timeout>0</session-timeout>
 </session-config>

 <taglib>
    <taglib-uri>/hello</taglib-uri>
    <taglib-location>/WEB-INF/tlds/hello.tld</taglib-location>
 </taglib>

</web-app>
```