Java Servlet Pages

No more out.println("<html>")...

EHSDI

eBuzima

Why Java Servlet Pages

- Trying to design a webpage using `println` calls inside a JAVA file is not good because:
  - It mixes business logic and presentation (the opposite of MVC)
  - Its difficult to design a page without seeing the layout of the HTML

What is a Java Servlet Page

- It's a JSP file that contains HTML, with additional tags and bits of Java code, e.g.

```
<html>
  <head>
    <title>My First JSP</title>
  </head>
  <body>
<%
  out.println("Hello World");
%>
  </body>
</html>
```

Java + HTML

Scriptlets

- A *scriptlet* is just a block of Java code between inside a `<% ... %>` tag
- For example:

```
<%
  String message = "Hello World";
  out.println(message);
%>
```

```
<% out.println(Math.random()); %>
```

Expressions

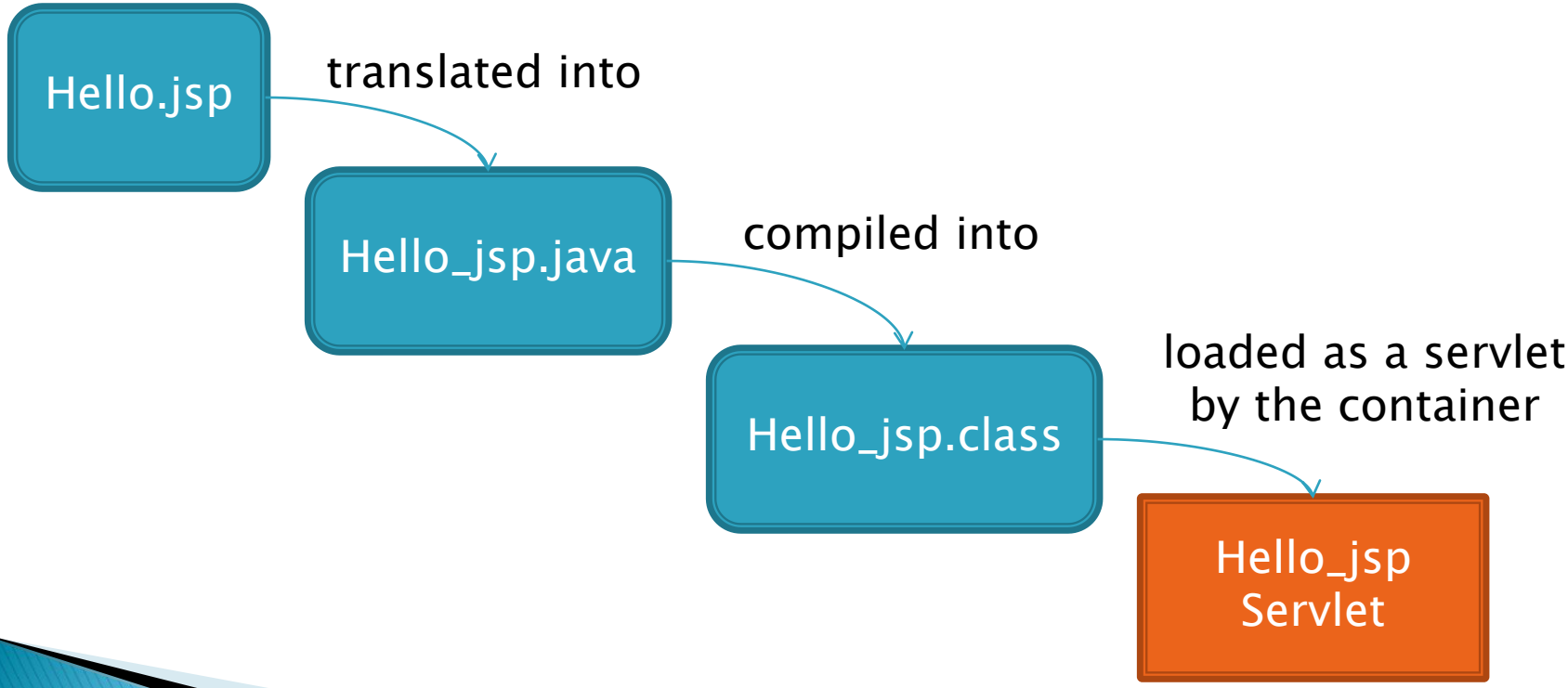
- An expression is a piece of Java code which evaluates to a printable value
- This is placed inside a `<%= ... %>` tag
- No need to call `println`
- No semicolon at the end

```
<% out.println("Hello World"); %>
```

```
<%= "Hello World" %>
```

How it Works

- The JSP file is automatically transformed into a servlet



How it Works

- The HTML in the JSP becomes println calls
- The Scriptlets are copied directly

Hello.jsp

```
<html>
<head>
<title>My JSP</title>
</head>
<body>
<%
  String msg = "Hello World";
  out.println(msg);
%>
</body>
</html>
```

Hello_jsp.java

```
PrintWriter out = response.getWriter();
response.setContentType("text/html");
out.write("<html>");
out.write("<head>");
out.write("<title>My JSP</title>");
out.write("</head>");
out.write("<body>");
out.write("<body>");
String msg = "Hello World";
out.write(msg);
out.write("</body>");
out.write("</html>");
```

Directives

- A *directive* is an instruction to be given to the container
- They are put in a `<%@ ... %>` tag
- For example:

```
<%@ page import="java.util.*" %>
```

```
<%@ include file="header.jsp" %>
```

Example: Include Directive

page.jsp

```
<html>
<head>
<title>Website</title>
</head>
<body>
<h1>My Website</h1>
<h2>Home</h2>
<p>Welcome...</p>
<p>Copyright 2009</p>
</body>
</html>
```

header.jsp

```
<html>
<head>
<title>Website</title>
</head>
<body>
<h1>My Website</h1>
```

page.jsp

```
<%@ include file="header.jsp" %>
<h2>Home</h2>
<p>Welcome...</p>
<%@ include file="footer.jsp" %>
```

footer.jsp

```
<p>Copyright 2009</p>
</body>
</html>
```

Declarations

- A scriptlet's code is copied into the service method, and so is called every time a request is handled by the service method
- If we have code we want to be called once in the lifetime of the servlet, it can be put in a *declaration* `<%! . . . %>` tag
- Even methods can be declared, e.g.

```
<%! int count = 0; %>
```

```
<%! int getCount() {
  return ++count;
} %>
```

Comments

- There are two kinds of comment you can put in a JSP:
  - `<!-- HTML comments -->`
    This will be sent to the client like any other HTML tag
  - `<%-- JSP comments --%>`
    This will be stripped from the final HTML

Summary of JSP Elements

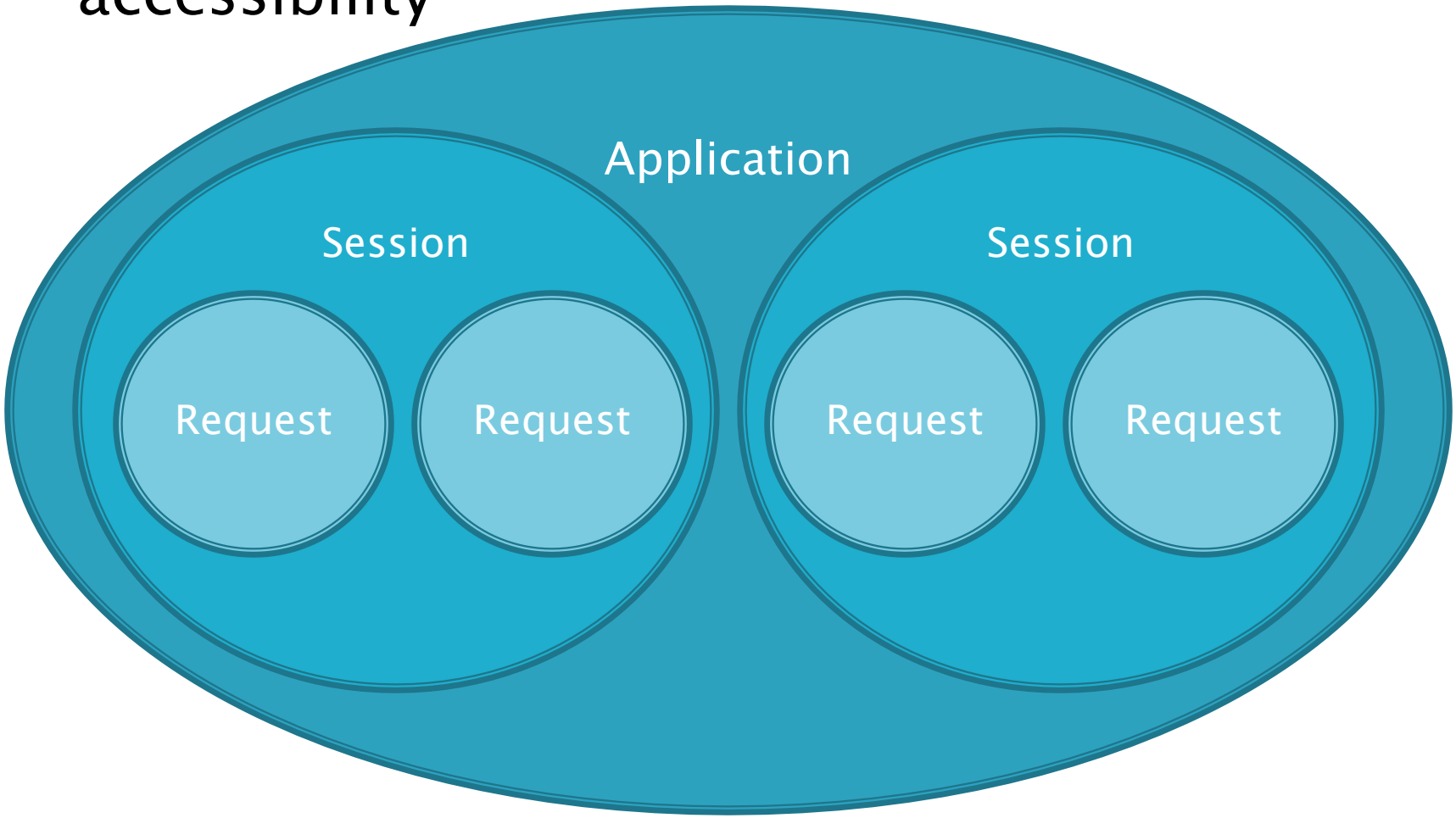
Tag	Name	Description
`<% ... %>`	**Scriptlet**	Puts Java code directly into the servlet service method
`<%= ... %>`	**Expression**	Puts an expression into `out.println()` inside the servlet service method
`<%@ ... %>`	**Directive**	Gives an instruction to the compiler
`<%! ... %>`	**Declaration**	Puts variable and method declarations into the servlet class, outside of the service method
`<%-- ... --%>`	**Comment**	A comment which won't be included in the HTML response

Attributes

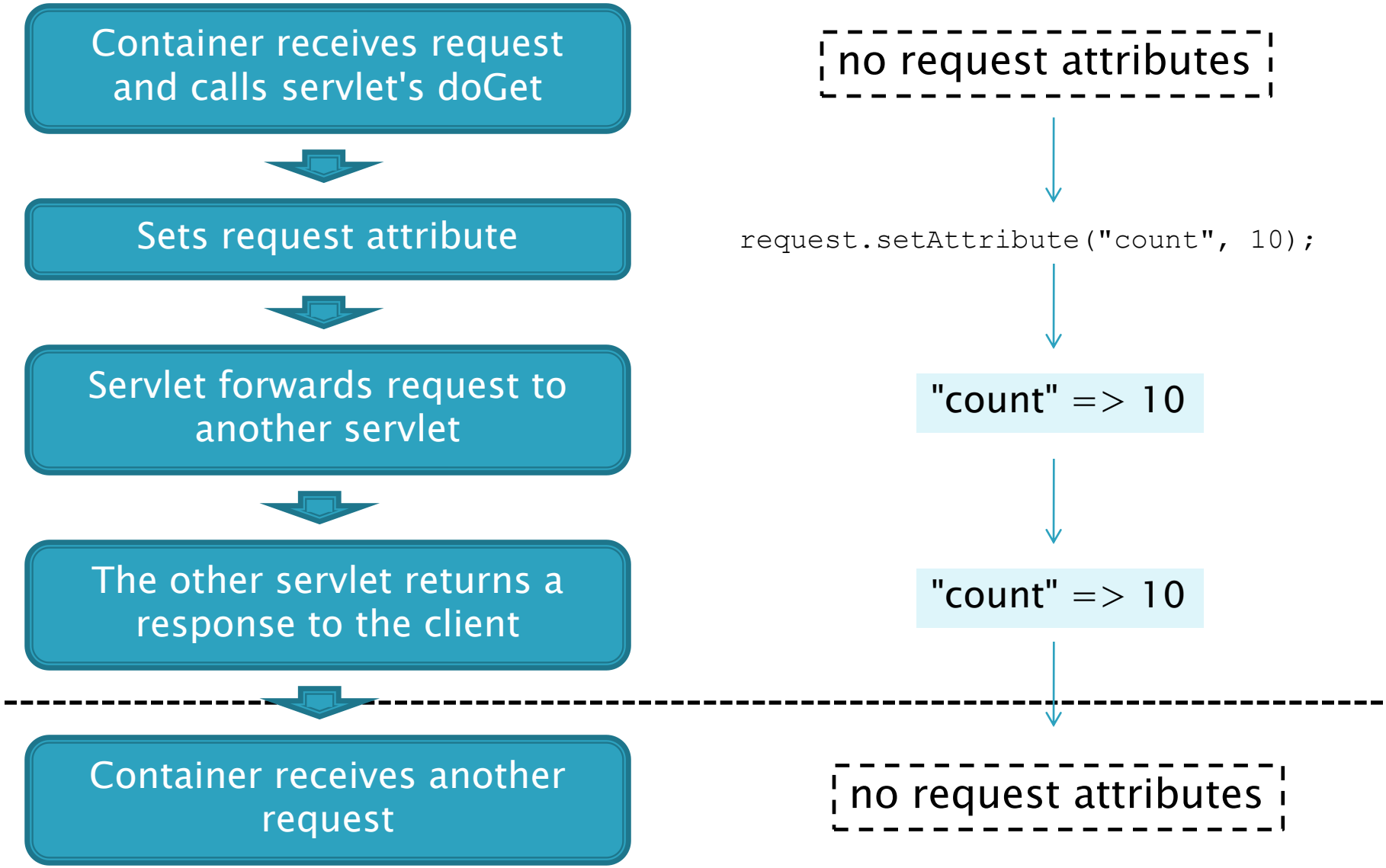
- We've already seen how attributes (data values) can be set on the session object, e.g.
  - `session.setAttribute("count", 10);`
- But we can also set attributes on the request object, e.g.
  - `request.setAttribute("count", 12);`
- .. and also on the application object
  - `application.setAttribute("count", 14);`
- Each of these will have a different scope...

Scopes

- Attributes have different scopes, i.e. level of accessibility



Example: Request Scope



Summary of Scopes

Name	Description
page	objects and attributes associated with a page. A page is represented by a single compiled Java servlet class (along with any include directives).
request	objects and attributes associated with a request initiated by the client. A request may span multiple pages (due to forward directives).
session	objects and attributes associated with a single user experience for a client. A session can and often will span multiple client requests.
application	objects and attributes associated with an entire Web application. This is essentially a global scope spanning an entire Web application across multiple pages, requests, and sessions.

Initialization Parameters

- We've seen how initialization parameters can be given for a servlet, in the DD
- This avoids hard-coding values (which might change) into servlet Java code, e.g.

```
<web-app>
  <servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>test.TestServlet</servlet-class>
    <init-param>
      <param-name>email</param-name>
      <param-value>bob@pih.org</param-value>
    </init-param>
  </servlet>
  ...
</web-app>
```

Initialization Parameters

- We can do the same for a JSP by adding it to the DD as a servlet
- Instead of specifying the Java class, we specify the JSP file

```
<web-app>
  <servlet>
    <servlet-name>TestServlet</servlet-name>
    <jsp-file>/test.jsp</jsp-file>
    <init-param>
      <param-name>email</param-name>
      <param-value>bob@pih.org</param-value>
    </init-param>
  </servlet>
  ...
</web-app>
```

Initialization Parameters

- We can access the initialization parameters from the JSP, using the `config` object, e.g.

```
<%= config.getInitParameter("email") %>
```

Implicit Objects

- This is a variable that is by default, available in every JSP page, e.g.
  - `out` – an implicit object, of type `JspWriter`, used to write data to the HTTP response
  - `config` – an implicit object used to access servlet configuration values such as initialization parameters

Summary of Implicit Objects

Name	Type	Typical Uses
out	JspWriter	outputting HTML
request	HttpServletRequest	getting request parameters, attributes and cookies
response	HttpServletResponse	adding cookies
session	HttpSession	getting session attributes
application	ServletContext	getting container-level parameters such as context path
config	ServletConfig	getting servlet parameters from the DD
pageContext	PageContext	getting attributes from any scope

References

- Books
  - Head First Servlets and JSP (O'Reilly)
- Websites
  - http://java.sun.com/javaee/reference/tutorials/