Servlet and JSP

Banu Prakash

Objectives

- Understand different mechanisms used to store conversational state of a client.
- Understand how to use JSP implicit object “session”.
- Understand how different users of a web application can share information.

HTTP Protocol

- HTTP Protocol is a stateless protocol which means that each page request is considered independent of any other request.
  - A web server does not understand if a page request comes from someone who has already requested a page or if the person is visiting the page for the first time.
  - It treats each request in the same way.
- In some scenarios, e.g. ecommerce and banking web applications this is inconvenient and we need some way to tie every page request together as a session of a single visitor.

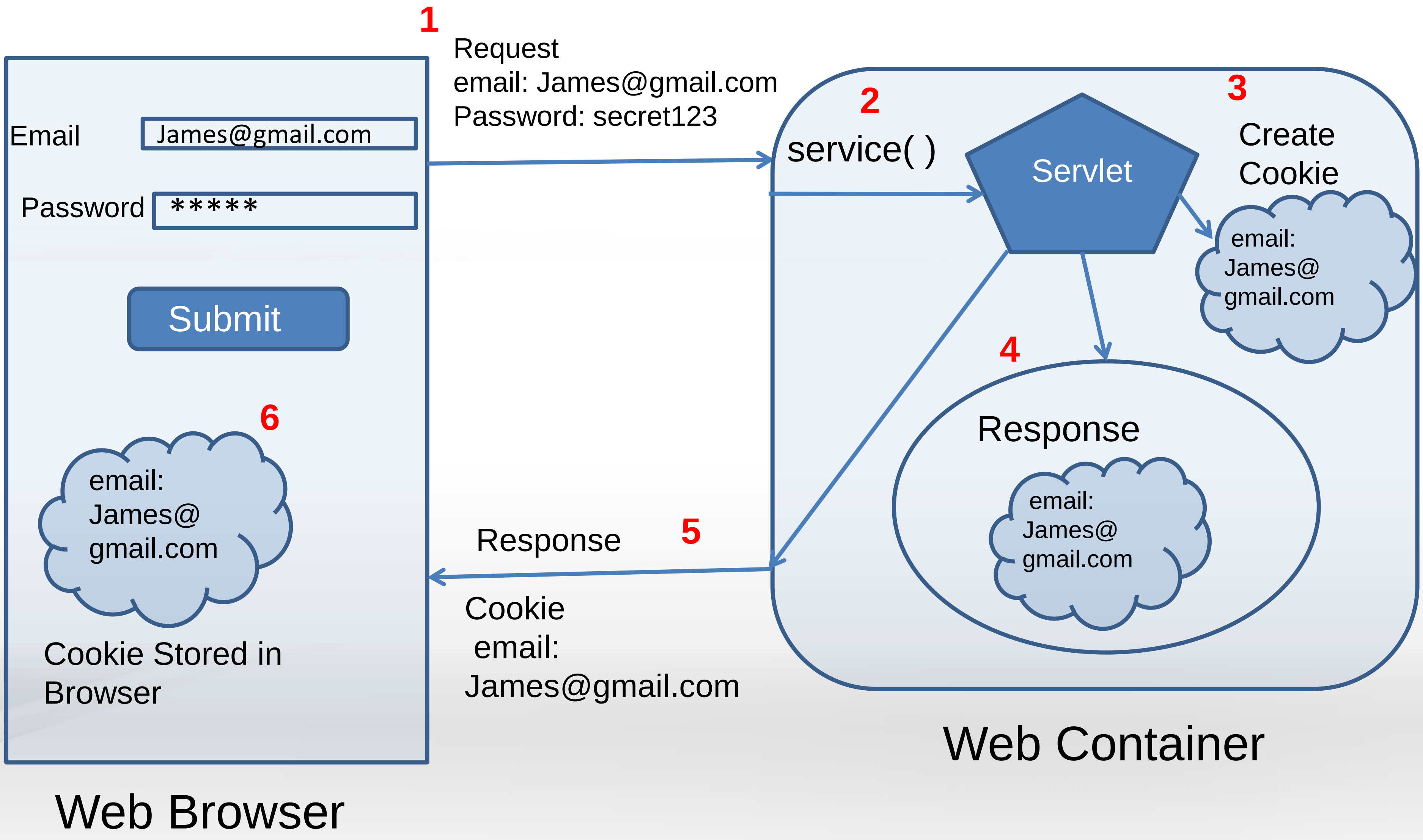
Session Tracking

- Session tracking is a mechanism which helps the servers to maintain the conversational state of a client.
- Different mechanisms that can be used for session tracking:
  - Hidden Fields
  - URL Rewriting
  - Cookies
  - HttpSession API.

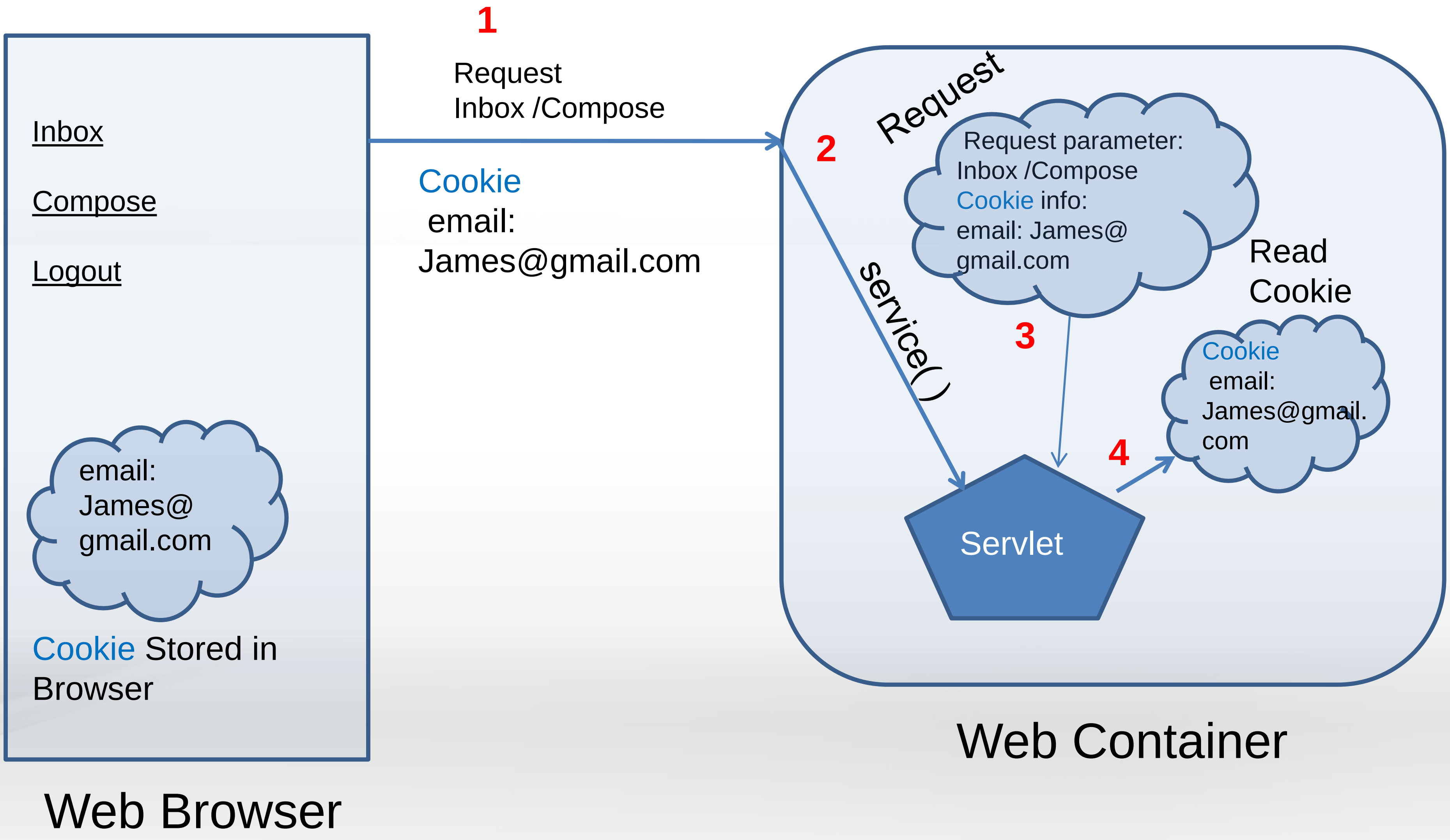
Cookie

- Cookies can be used for session tracking.
- A *cookie* is a bit of information sent by a web server to a browser that can later be read back from that browser.
- Browser receives a cookie and it saves the cookie.
- Browser sends the cookie back to the server each time it accesses a page on that server

How cookies work?



How cookies work?



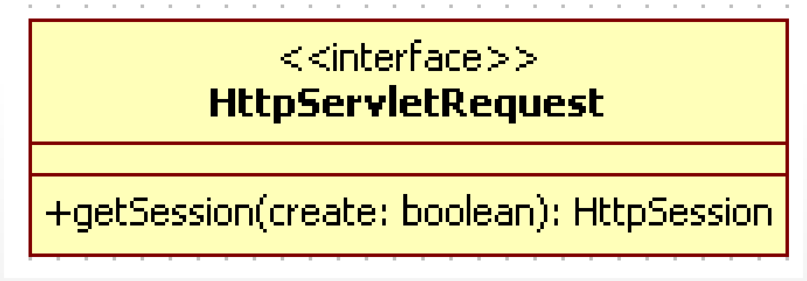
Cookie and Servlet API

- Creating a Cookie using constructor:
  - public Cookie(String name, String value)
  - This creates a new cookie with an initial name and value
- Sending a cookie to the client:
  - public void addCookie(Cookie cookie) method of HttpServletResponse
- Retrieving all cookies sent by the web browser:
  - public Cookie[ ] getCookies() method of HttpServletRequest.

HttpSession API

- HttpSession interface is used by the Web Container to create a session between the HTTP server and an HTTP client.

- A Servlet uses its request object's getSession() method to retrieve the current HttpSession object:

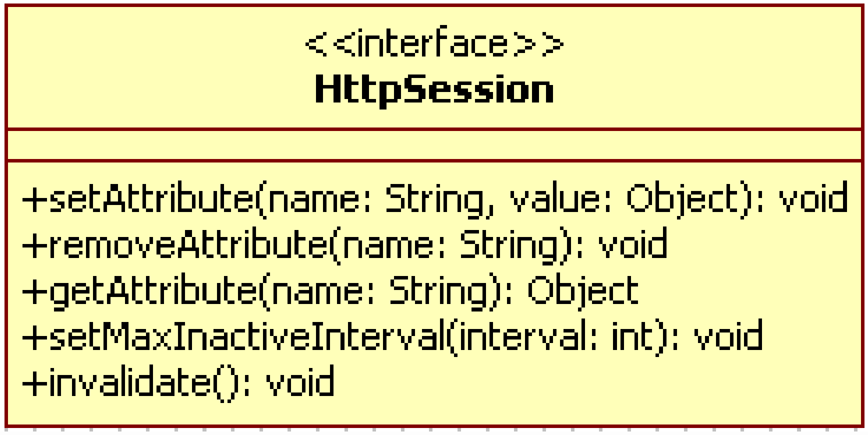


This method returns the current session associated with the user making the request.

If the user has no current valid session, this method creates one if **create** is true or returns null if **create** is false.

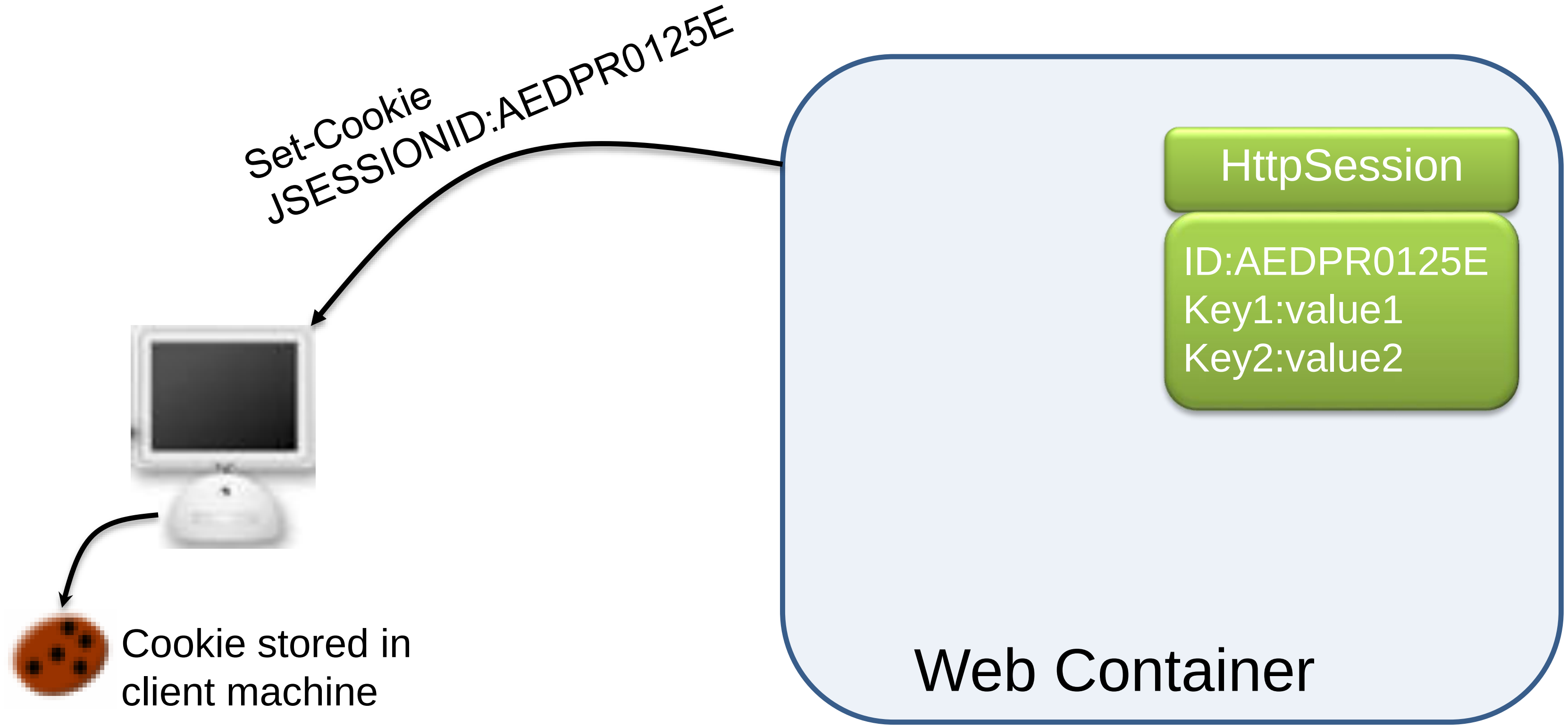
HttpSession API

- void **setAttribute**(String name, Object value)
  - Binds an object to this session, using the name specified.
- void **removeAttribute**(String name)
  Removes the object bound with the specified name from this session.
- Object **getAttribute**(String name)
  Returns the object bound with the specified name in this session, or null if no object is bound under the name.
- void **setMaxInactiveInterval**(int interval)
  Specifies the time, in seconds, between client requests before the servlet container will invalidate this session


<<interface>>
HttpSession
+setAttribute(name: String, value: Object): void
+removeAttribute(name: String): void
+getAttribute(name: String): Object
+setMaxInactiveInterval(interval: int): void
+invalidate(): void

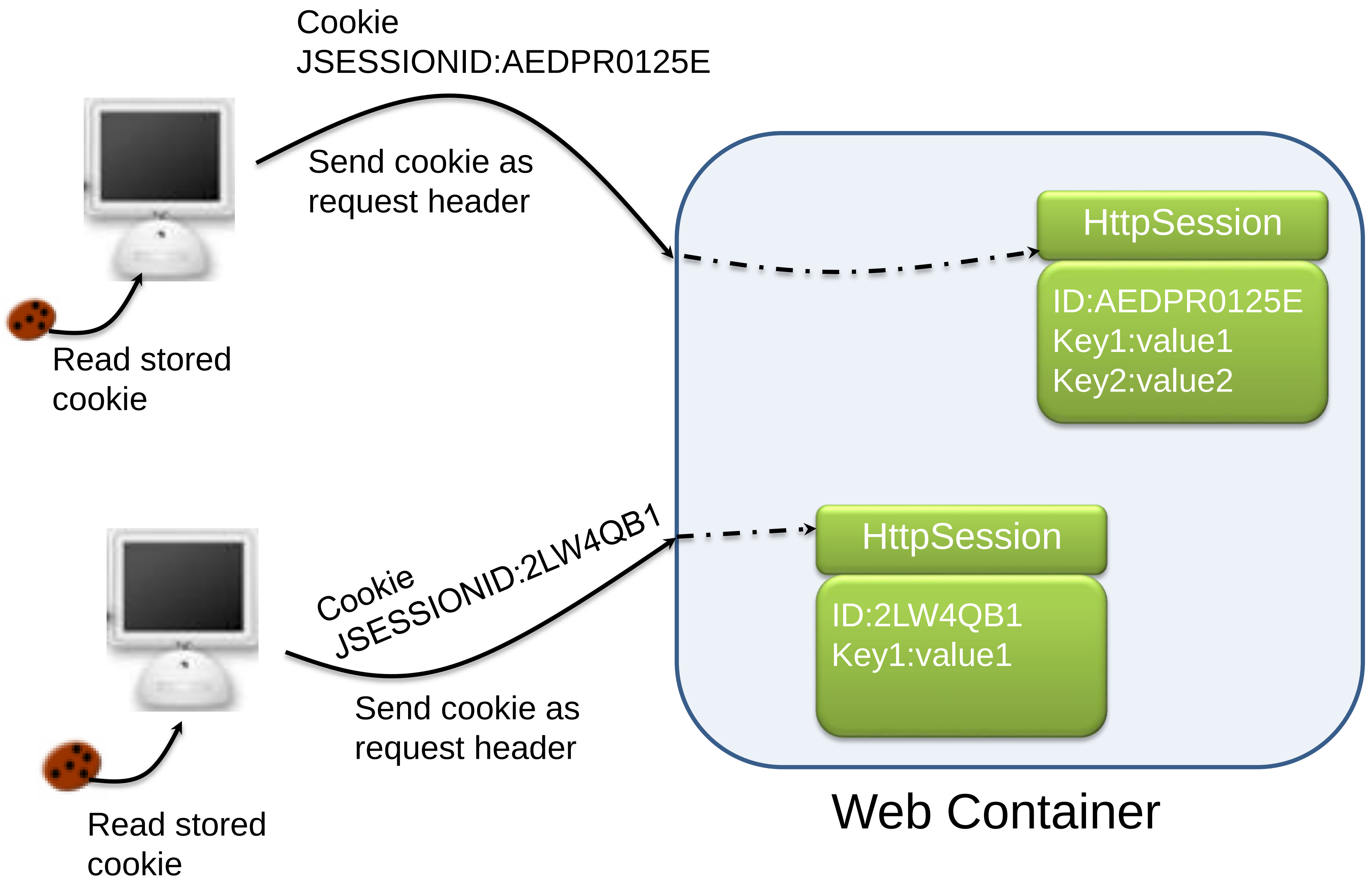
- void **invalidate**()
  Invalidates this session then unbinds any objects bound to it.

HttpSession API



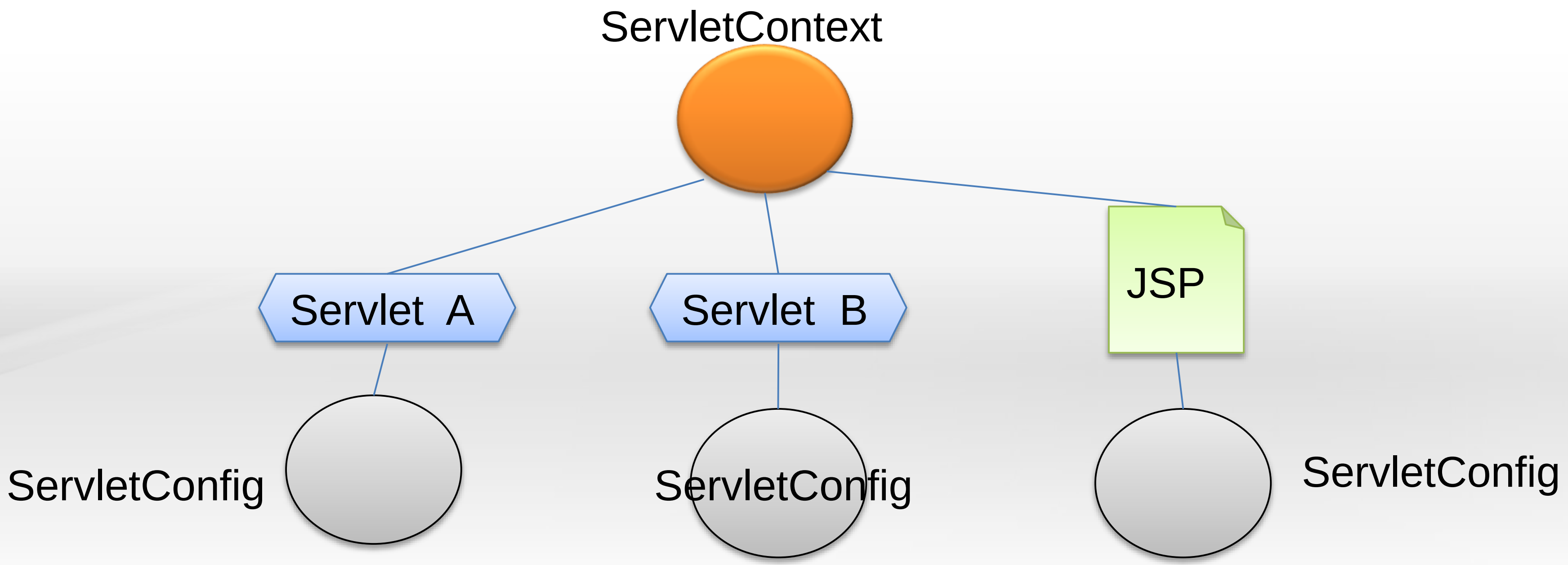
Every Session created for Client’s will have an unique Session ID.
The Session ID will be sent to the client in the form of Cookie or URL-Rewriting.
Client browser stores cookies and for every new request cookie information will be sent to the server through request headers.

HttpSession API



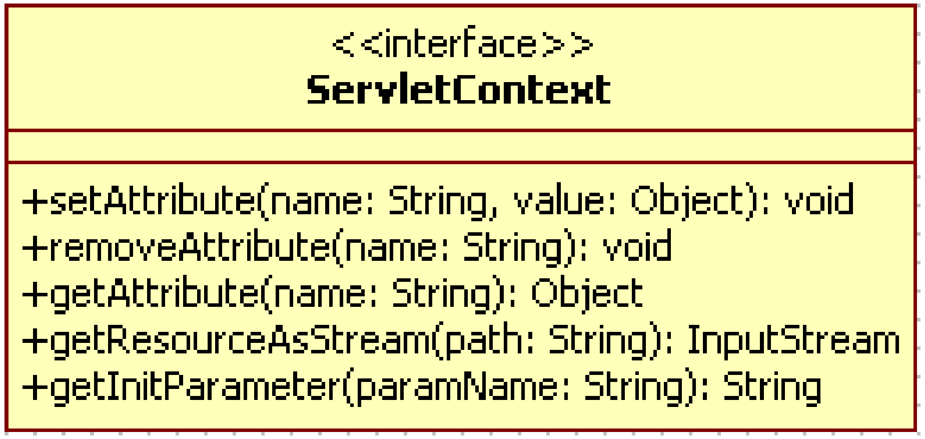
ServletContext

- ServletContext encapsulates a web application.
- ServletContext is created one per Web Application.
- Two main uses of ServletContext are:
  - Sharing information between Servlet's and users .
  - Accessing passive server resources like configuration files present in web application.



ServletContext

- void setAttribute(String name, Object object)
  Binds an object to a given attribute name in this ServletContext.
- void removeAttribute(String name)
  Removes the attribute with the given name from this ServletContext.
- Object getAttribute(String name)
  Returns the servlet container attribute with the given name, or null if there is no attribute by that name.
- java.io.InputStream getResourceAsStream(String path)
  Returns the resource located at the named path as an InputStream object.



<<interface>>
ServletContext
+setAttribute(name: String, value: Object): void
+removeAttribute(name: String): void
+getAttribute(name: String): Object
+getResourceAsStream(path: String): InputStream
+getInitParameter(paramName: String): String

- String getInitParameter(String name)
  Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.

Code Snippet: Reading context parameters

The context – parameters configured in web.xml file

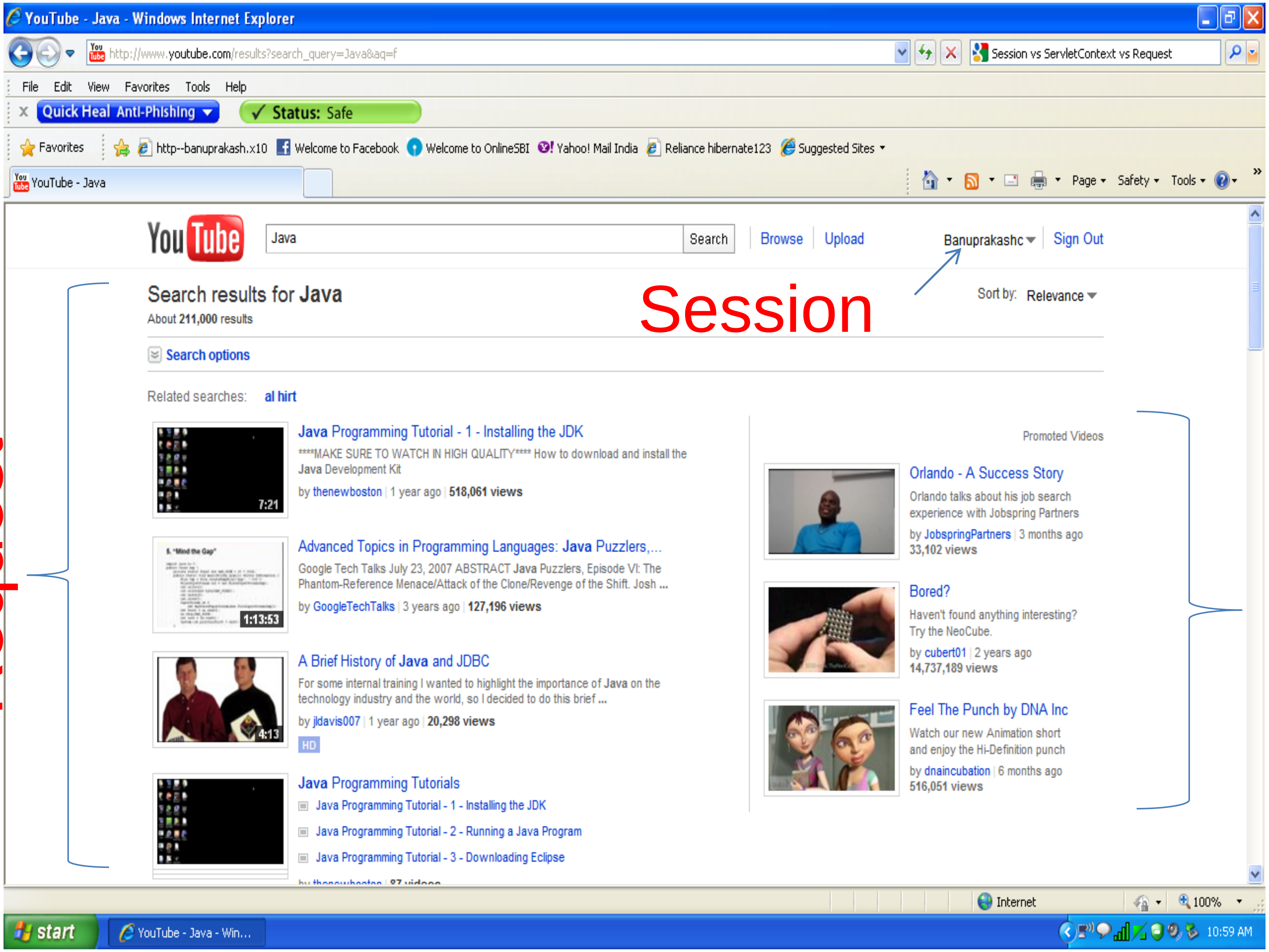
```
<context-param>
  <description>Background color for all servlets</description>
  <param-name>backgroundColor</param-name>
  <param-value>lavender</param-value>
</context-param>
```

Servlet code to read Context parameters

```
private String bgColor;

public void init() {
    /*
     * Get Servlet Context object instantied by the container.
     * Web Container creates one ServletContext object per application.
     */
    ServletContext context = getServletContext();
    /*
     * Read Context initialization parameter
     */
    bgColor = context.getInitParameter("backgroundColor");
}
```

Request, Session and ServletContext all in one



YouTube - Java - Windows Internet Explorer

http://www.youtube.com/results?search_query=Java&aq=f

Session vs ServletContext vs Request

File Edit View Favorites Tools Help

Quick Heal Anti-Phishing Status: Safe

Favorites http--banuprakash.x10 Welcome to Facebook Welcome to OnlineSBI Yahoo! Mail India Reliance hibernate123 Suggested Sites

YouTube - Java

Page Safety Tools

Java Search Browse Upload Banuprakashc Sign Out

Search results for **Java**

About **211,000** results

Sort by: Relevance

Search options

Related searches: al hirt

Java Programming Tutorial - 1 - Installing the JDK
****MAKE SURE TO WATCH IN HIGH QUALITY**** How to download and install the Java Development Kit
by thenewboston | 1 year ago | **518,061 views**

Advanced Topics in Programming Languages: Java Puzzlers,...
Google Tech Talks July 23, 2007 ABSTRACT Java Puzzlers, Episode VI: The Phantom-Reference Menace/Attack of the Clone/Revenge of the Shift. Josh ...
by GoogleTechTalks | 3 years ago | **127,196 views**

A Brief History of Java and JDBC
For some internal training I wanted to highlight the importance of Java on the technology industry and the world, so I decided to do this brief ...
by jldavis007 | 1 year ago | **20,298 views**
HD

Java Programming Tutorials
Java Programming Tutorial - 1 - Installing the JDK
Java Programming Tutorial - 2 - Running a Java Program
Java Programming Tutorial - 3 - Downloading Eclipse

Promoted Videos

Orlando - A Success Story
Orlando talks about his job search experience with Jobspring Partners
by JobspringPartners | 3 months ago
33,102 views

Bored?
Haven't found anything interesting? Try the NeoCube.
by cubert01 | 2 years ago
14,737,189 views

Feel The Punch by DNA Inc
Watch our new Animation short and enjoy the Hi-Definition punch
by dnaincubation | 6 months ago
516,051 views

Internet 100%

start YouTube - Java - Win... 10:59 AM

Session

Request

ServletContext

JSP Directives

- Directives are messages to the JSP container in order to affect overall structure of the Servlet.

Syntax:

```
<%@ directive {attr=value}* %>
```

- Types of directives:
  - Page
    - Communicate page dependent attributes to the JSP.
  - Include
    - Used to include text or code into JSP at the time of translation.
  - Taglib
    - Includes a tag library that the JSP should interpret.

Page Directive

- Attributes of Page directive
  - The import attribute
    - <%@ page import="java.util.List, java.util.ArrayList" %>
    - <%@ page import="com.banu.entity.Book" %>
      - The packages (and their classes) are available to scriptlets, expressions, and declarations within the JSP file.
  - The session attribute
    - Whether the client must join an HTTP session in order to use the JSP page.
    - <%@ page session="**true" %>**. the session object refers to the current or new session.
    - <%@ page session="**false**" %> You cannot use the session object in the JSP page.
    - The default value is true.

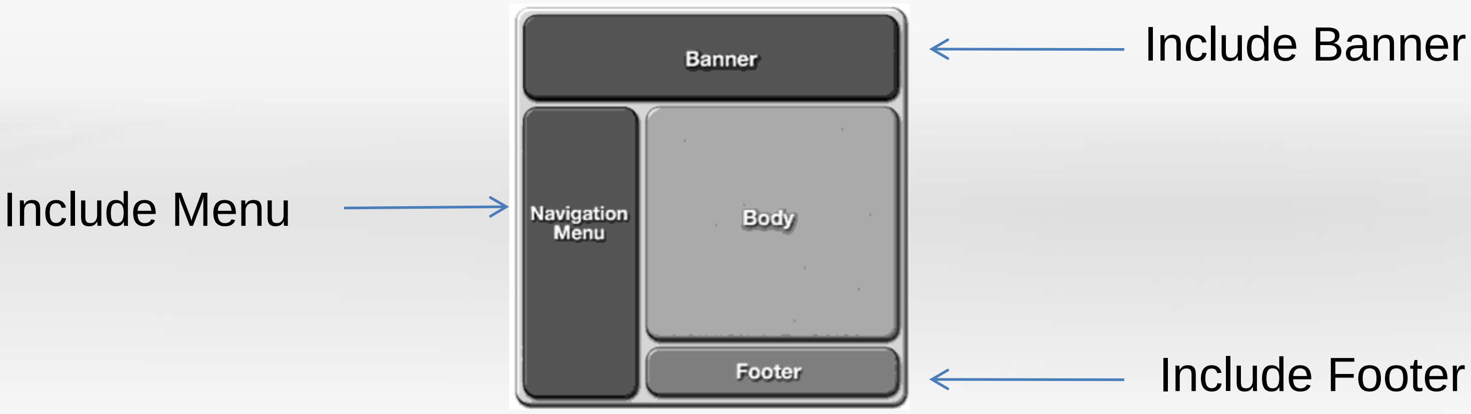
Page Directive

- Attributes of Page directive
  - The isThreadSafe attribute
    - <%@ page isThreadSafe=“**true|false” %>**
    - The default value is true, which means that the JSP container can send multiple, concurrent client requests to the JSP page.
    - If you use false, the JSP container sends client requests one at a time to the JSP page
  - The errorPage attribute
    - <%@ page errorPage=“relativeURL” %>
    - A pathname to a JSP file that this JSP file sends exceptions to.
  - The isErrorPage attribute
    - <%@ page isErrorPage=“true|false” %>
    - If set to true, you can use the exception object in the JSP file

Include directive

- <%@ include file="filename" %>

- The file attribute is interpreted as relative URL.
- The include directive includes the file at the time of translation. i.e., include at the time when JSP is converted into Servlet.
- If it starts with the forward slash, it is interpreted as relative to the context of the application [context specific path], otherwise it is interpreted as relative to the current JSP file.



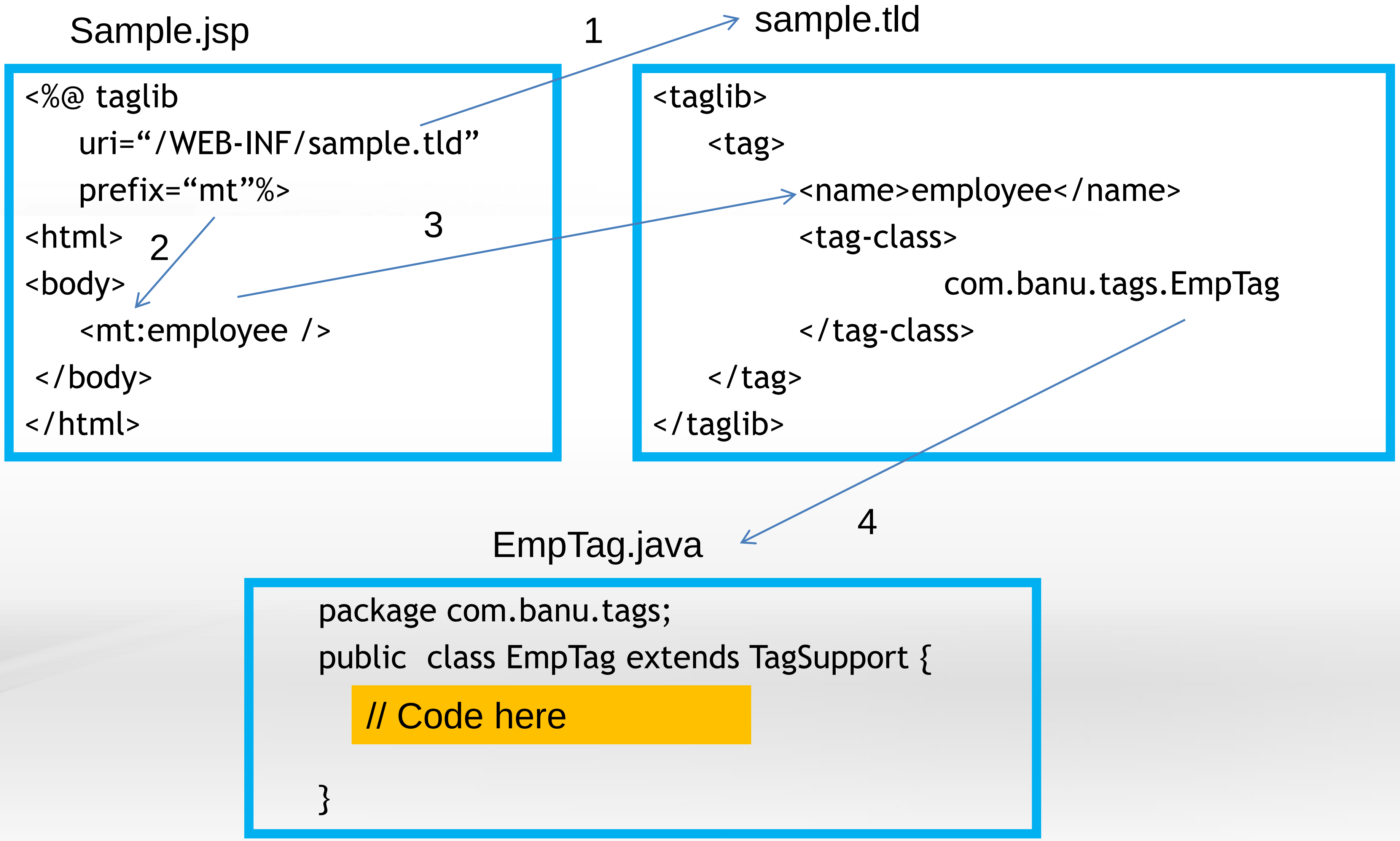
Custom Tag Library

- *Custom tags* are user-defined JSP language elements that encapsulate recurring tasks.
- Custom tags eliminate the need to write scriptlets, in which Java code is embedded in the JSP page.

- Why use Custom Tags?
  - Using custom tags to replace embedded scriptlets makes the JSP page more readable and easier to maintain.

  - Because Java functionality resides in separate custom tag libraries, the code can be easily reused in multiple JSP page

How do they work?



Sample.jsp

```
<%@ taglib
     uri="/WEB-INF/sample.tld"
     prefix="mt"%>
<html>
<body>
     <mt:employee />
 </body>
</html>
```

sample.tld

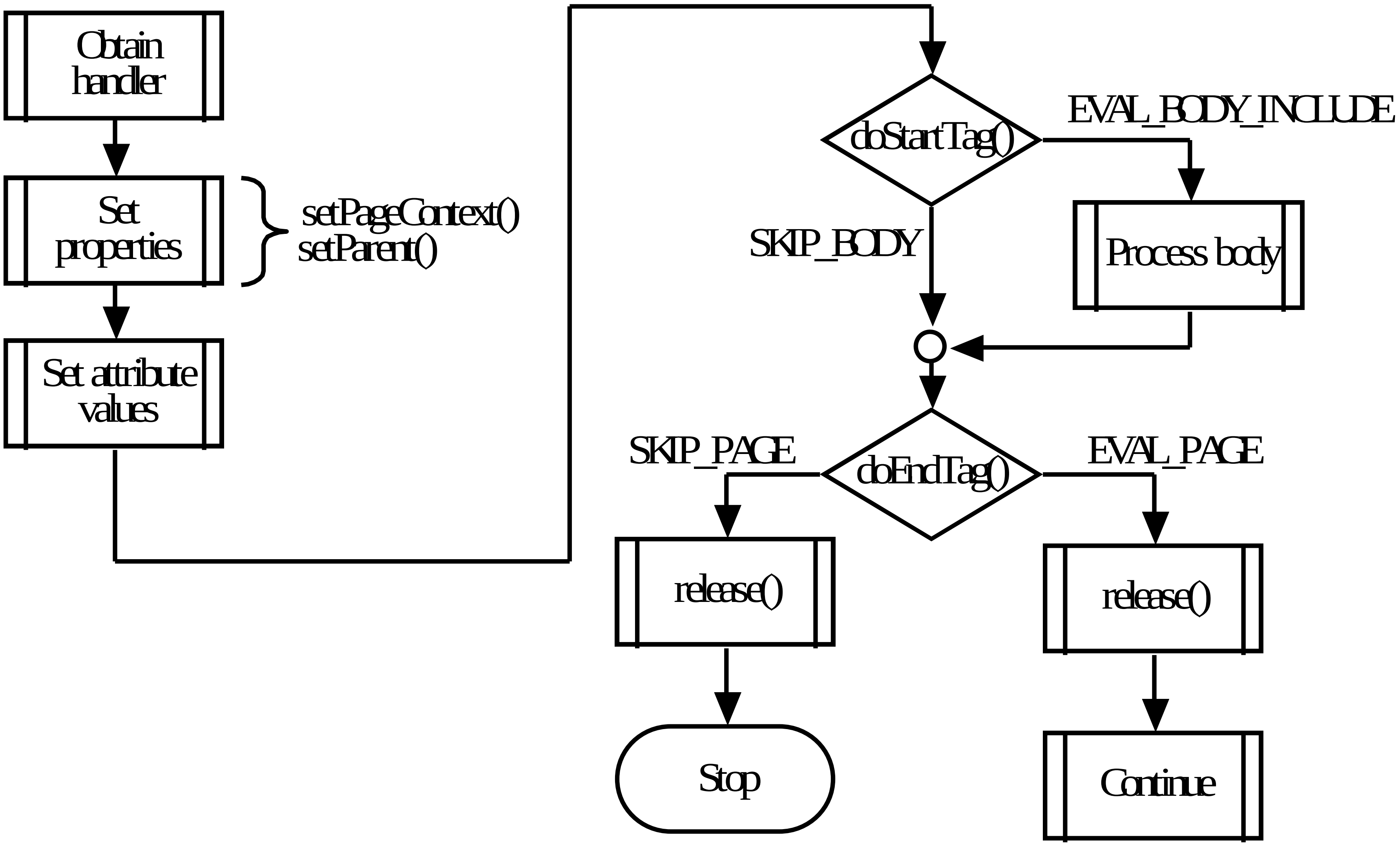
```
<taglib>
     <tag>
          <name>employee</name>
          <tag-class>
                    com.banu.tags.EmpTag
          </tag-class>
     </tag>
</taglib>
```

EmpTag.java

```
package com.banu.tags;
public  class EmpTag extends TagSupport {
   // Code here

}
```

Tag handler class life cycle



Java Server Pages Standard Tag Library (JSTL)

- The Java Server Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications.
- JSTL has support for common, tasks such as
  - iteration and conditionals tags.
  - manipulating XML documents tags.
  - Internationalization tags.
  - SQL tags.

Java Server Pages Standard Tag Library (JSTL)

- Why use JSTL?
  - use a single standard set of tag libraries that are already provided by compliant Java EE platforms.
  - Portability of your applications.
  - You don't have to write your own tags.
- Different JSTL tags:
  - Core (prefix: c)
    - Variable support, Flow control, URL management
  - XML (prefix: x)
    - Core, Flow control, Transformation
  - Internationalization (i18n) (prefix: fmt)
    - Locale, Message formatting, Number and date formatting
  - Database (prefix: sql)
    - SQL query and update
  - Functions (prefix: fn)
    - Collection length, String manipulation

Java Server Pages Standard Tag Library (JSTL)

- Taglibrary for Core tags:
  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
- Core tag Variable support
  - <c:set>
  - <c:remove>
- Core tag Conditional support
  - <c:if>
  - <c:choose>
  - <c:when>
  - <c:otherwise>
- Core tag Iteration support
  - <c:forEach>

JSTL Core tags

- <c:out> tag renders data to a page.
- Example:

Book ISBN : <c:out value="${param.isbn}“ default=“ISBN not set” />

Attribute	Description	Required	Default
value	Data to output	YES	NONE
default	Fallback data if value is empty	NO	BODY
escapeXML	To escape special characters	NO	true

JSTL Core tags

- Conditional Actions
- **<c:if> - processes the body if *test is true***
- ***Example:***

```
<c:if test="${param.price  != null}">
      Book Price : <c:out value="${param.price}" />
</c:if>
```

Attribute	Description	Required	Default
test	Condition to evaluate	YES	NONE
var	Name of the variable to store test condition result	NO	NONE
escapeXML	To escape special characters	NO	true

JSTL Core tags

- Iterator Actions
- **<c:forEach> -** repeats the nested body content over a collection or for a fixed number of times.

Example:

```
<c:forEach var ="I”  begin="1"  end="10">
    ${i}  <br />
</c:forEach>


<c:forEach items="${applicationScope.books}" var="book">


</c:forEach>
```

Attribute	Description	Required	Default
items	Collection to loop over	NO	NONE
var	Name of variable to hold the current item.	NO	NONE