EBU6501 - Middleware

Week 2, Day 1: Further Programming- Servlet and JavaServer Pages

Dr. Gokop Goteng

Queen Mary
University of London

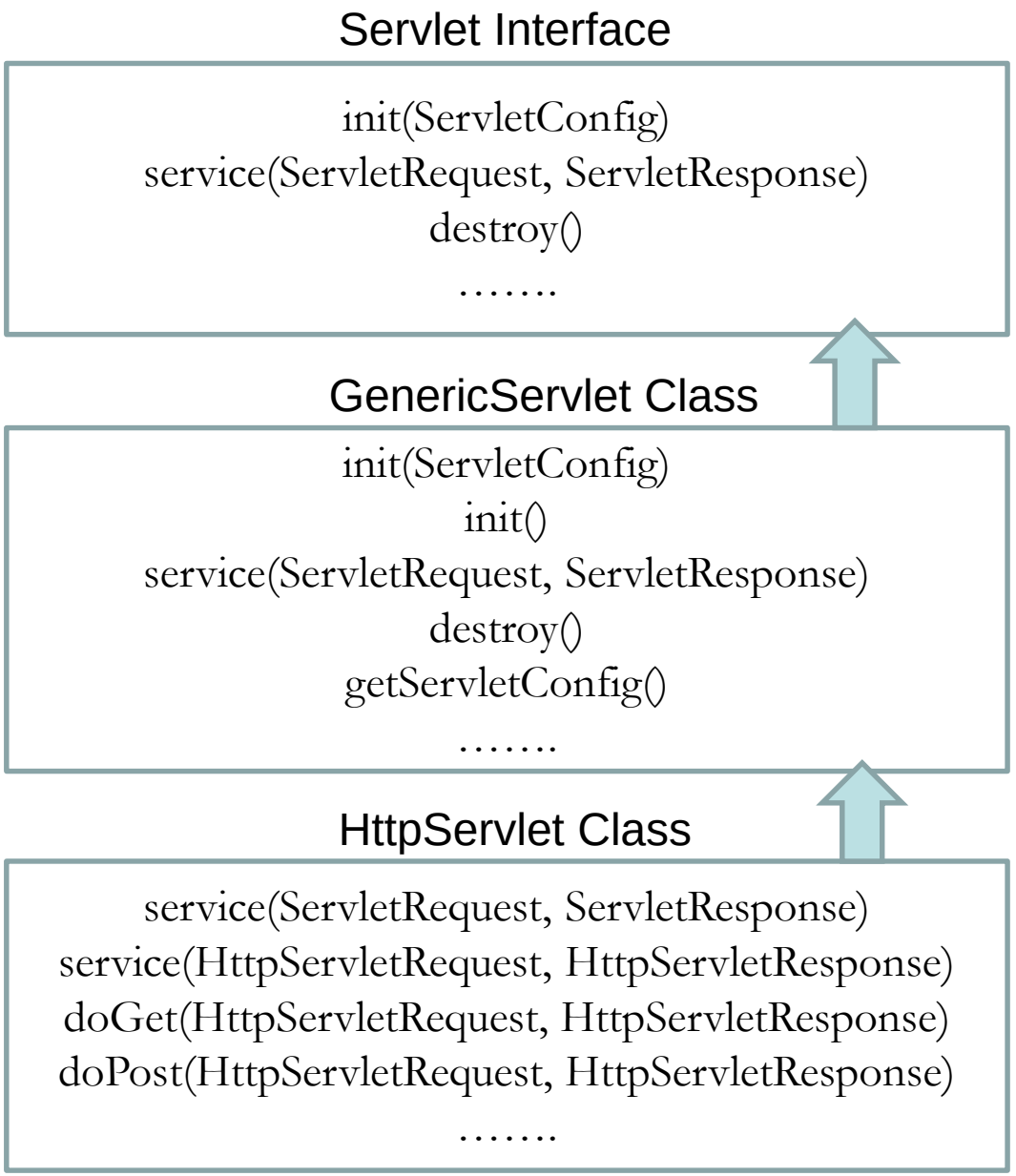
Lecture Aim and Outcome

- Aim
  - The aim of this lecture is to teach students how to write java programs using servlet and JSP
- Outcome
  - At the end of this lecture students should be able to:
    - Understand and use Servlet Interfaces and Classes in JSP programming
    - Understand the lifecycle of simple and classic tags
    - Configure JSP error pages

Lecture Outline

- Servlet Interface
- Servlet Lifecycle Methods
- ServletConfig and ServletContext Objects
- ServletRequest and ServletResponse Interfaces
- Servlet Format
- Output Objects
- Learning Servlet Programming by Example
- JSP – Looping
- JSP – Conditional Statements
- JSP- Choose/When/Otherwise
- JSP-Configure Error Pages
- Simple and Classic Tags
- Classic Tag Lifecycle

Servlet Interface



Servlet Interface

Relationship between Servlet Interface and Generic Class

javax.servlet.Generic Servlet implements javax.servlet.Servlet

Relationship between Generic Class and HttpServlet Class

javax.servlet.http.HttpServlet extends javax.servlet.GenericServlet

Relationship between user defined Class and HttpServlet Class

MyServlet extends javax.servlet.http.HttpServlet

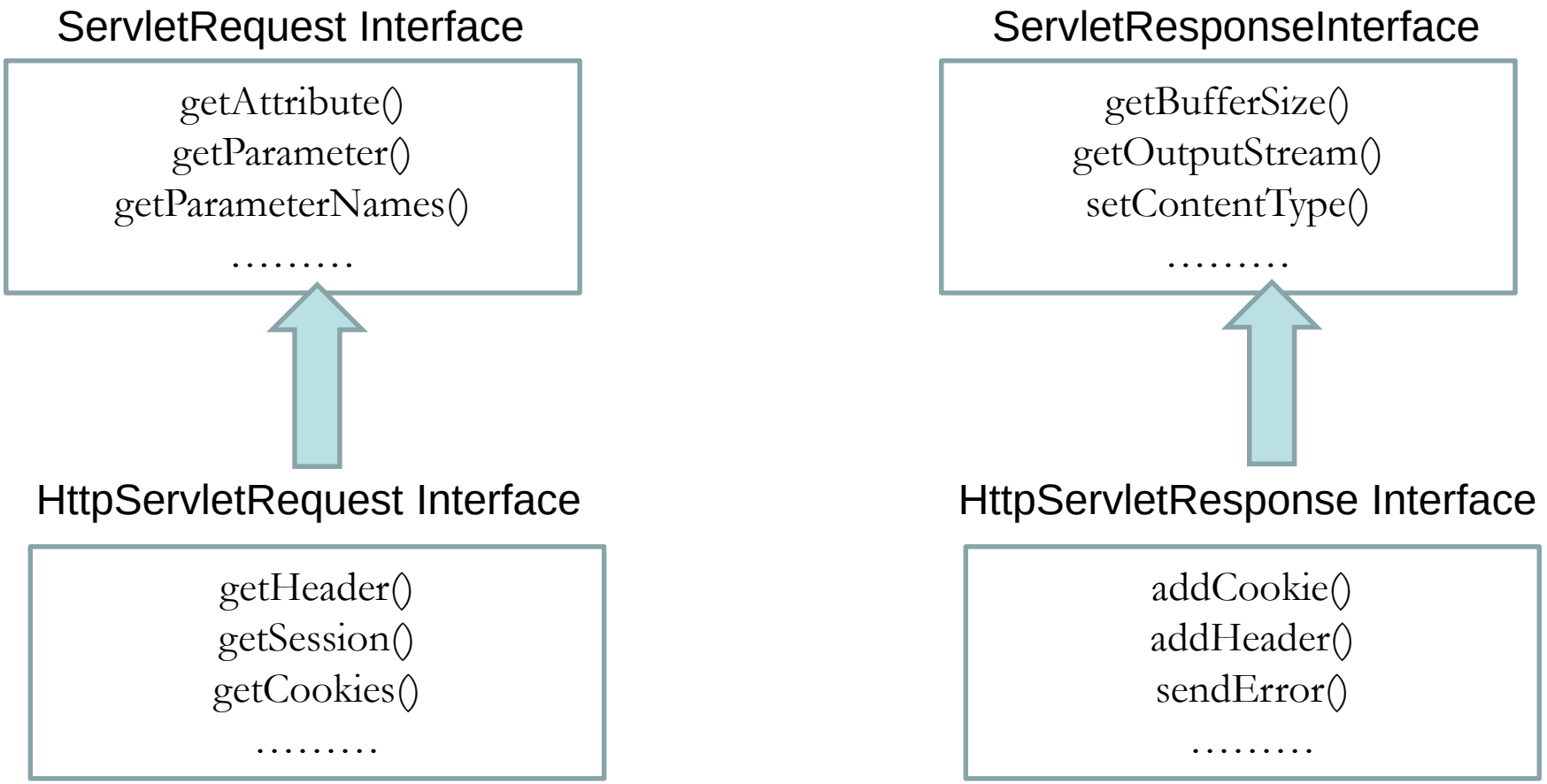
Servlet Lifecycle Methods

- init()
  - Container calls the init() method AFTER the servlet instance must have been created
  - The init() method must be called before the container can call the service() method
  - The user can override the init() method
  - It is called only once
  - It is used for initialising the servlet with needed parameters, configuration settings, etc
- service()
  - The container calls the service() method when it gets the request from a servlet and creates a separate thread for the request
  - The method invokes methods such as doPost(), doGet(), etc that perform the actions that the servlet intends to accomplish
  - The service() method is not normally overridden by users
  - The methods doPost(), doGet(), etc
    - The service() method runs the action methods such as the doPost(), doGet(), etc based on which of the HTTP methods (GET, POST, etc) is used
    - These methods run the action codes
- destoy()
  - The container calls the destroy() method when the service() method has finished processing all the methods (doGet(), doPost(), etc) that has been assigned to it and return all results.
  - The destroy() method is called only ONCE
  - It sends the objects to garbage collector to free the Java memory of unnecessary data

ServletConfig and ServletContext Objects

- ServletConfig
  - Each servlet has one ServletConfig
  - Each servlet uses it to define deployment time information and parameters without hard coding them
  - It is used to access ServletContext
  - Initialisation parameters are defined in the DD
  - The parameters defined affect only the servlet that is meant for
- ServletContext
  - This is defined once for a whole single application
  - This is used to define web application parameters
  - It is configured in the DD and contains initialisation parameters for the web application
  - It is also used to get application information, versions, server information

ServletRequest and ServletResponse Interfaces



Servlet Format

```
public class MyServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
                response.setContentType("text/html");

                ServletContext ctx = getServletContext();
                InputStream is = ctx.getResourceAsStream("/hello_butp.html");

                ……..
        }

……..
}
```

Output Objects

- PrintWriter

  PrintWriter pw = response.getWriter();

  pw.println("My Test Servlet Output Text");

  – This is used to print text to a character stream

- OutputStream

  ServletOutputStream sos = response.getOutput();

  sos.write("M Test Byte Array Outputs");

  – This can print any output, text, html, bytes, etc

Learning Servlet Programming by Example

```
// Source Code for HelloWorld Example

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

public class HelloWorld extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException
   {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Hello World!</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Hello World!</h1>");
      out.println("</body>");
      out.println("</html>");
   }
}
```

JSP - Looping

- <c:forEach>
  - The tag allows programmers to iterate over arrays and collections

```
<c:forEach var="bookTitle" items="${bookShelf}" >

  ${book}
</c:forEach>
```

  - bookTitle is the element and bookshelf is the entire array where all books are kept

JSP – Conditional Statements

- <c:if>
  - This tag allows programmers to perform certain actions when a condition is satisfied

```
<c:if test="${studentNo le 2000}  " >
      <jsp:include page ="student_bupt.jsp" />
</c:if>
```

  - <c:if> does not have else statement such as <c:else>
  - You can however nest the <c:if> tag

JSP- Choose/When/Otherwise

- The <c:choose>/ <c:when> / <c:otherwise> tags play the role of the if/else and switch/case statements

```
<c:choose>
    <c:when test="${studentName == "John"}>
            John is male
    </c:when>
    <c:when test="${studentName == "Mary"}>
            Mary is female
    </c:when>
    <c:otherwise>
            I don't know
    </c:otherwise>
</c:choose>
```

JSP-Configure Error Pages

- Web programmers use error pages to display errors when users don't choose the right site or when the URL they click does not exist
- These error pages are declared in the deployment descriptor (DD)

```
<error-page>
    <exception-type>java.lang.Throwable</exception-type>
     <location>/myErrorPage.jsp</location>
</error-page>

<error-page>
    <error-code>404</error-code>
     <location>/myErrorPage.jsp</location>
</error-page>
```

- Home work
  - Find out other specific error page declarations

Simple and Classic Tags

- Tag files are used to implement tag functionalities using JSP
  - Call the code implementation, process data, open files, etc
- Special Java classes are used to implement the handlers
  - SimpleTagSupport, TagSupport and BodyTagSupport classes
- The tag handlers are Simple Tag and Classic Tag

Class Work

- Class Discussions