COMP 3800SEF/3820SEF/S380F/S380W Web Applications: Design and Development
Lab 3: Servlet: Parameters and Attributes

This lab will cover the following topics. Answers are available in the branch `lab03ans` in the GitHub repository https://github.com/FloraZHANGJingyu/wadd2026.git.

- Differences between the Context init parameter, Servlet init parameter, and Context attribute
- Using the "synchronized" code block for thread safety of shared memory

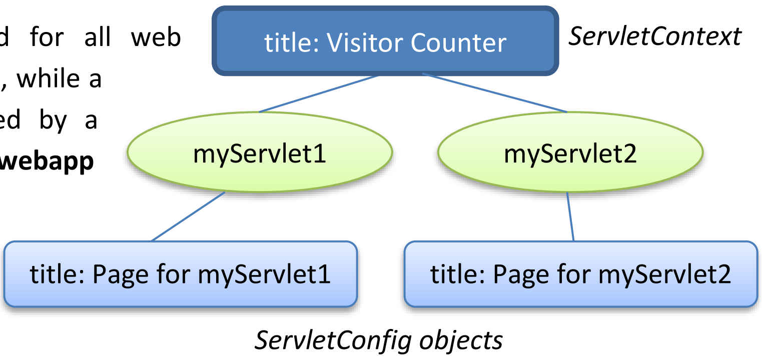
Task 1: Context and Servlet init parameters, Context attributes, and the use of "synchronized"

We will create a web application that has a visitor counter, which counts the visits on all servlets.

1. In IntelliJ, create a **Gradle Web Application** project with the following properties:
   - Generator: **Jakarta EE**
   - Name: **Lab03**
   - Template: **Web application**
   - Application Server: **Tomcat 10.1**
   - Build system: **Gradle**
   - Group: **hkmu.wadd**
   - Jakarta EE Version: **Jakarta EE 10**

2. Right-click on **src > main > java** to create a Servlet class **hkmu.wadd.VisitCounterServlet**.

3. A **context init parameter** can be used for all web components (e.g., servlets) in the web app, while a **servlet init parameter** can only be used by a particular servlet. Expand **src > main > webapp** and open the **deployment descriptor /WEB-INF/web.xml**. Add the following servlet name, URL pattern, context init parameter and servlet init parameter:



```
<context-param>
    <param-name>title</param-name>
    <param-value>Visitor Counter Site</param-value>
</context-param>

<servlet>
    <servlet-name>myServlet1</servlet-name>
    <servlet-class>hkmu.wadd.VisitCounterServlet</servlet-class>
    <init-param>
        <param-name>title</param-name>
        <param-value>Page for myServlet1</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>myServlet1</servlet-name>
    <url-pattern>/visit1</url-pattern>
</servlet-mapping>
```

4. Put the following doGet method (for serving an HTTP GET request) to VisitCounterServlet:

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
    ServletContext context = this.getServletContext();
    Integer count;
    synchronized (context) {
        count = (Integer) context.getAttribute("counter");
        if (count == null) {
            count = 1;
        } else {
            count++;
        }
        context.setAttribute("counter", count);
    }

    ServletConfig config = this.getServletConfig();
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    PrintWriter out = response.getWriter();
    out.append("<!DOCTYPE html>\r\n")
            .append("<html>\r\n")
            .append(" <head>\r\n")
            .append(" <title>" + config.getInitParameter("title") + "</title>\r\n")
            .append(" </head>\r\n")
            .append(" <body>\r\n")
            .append("<h1>" + config.getInitParameter("title") + "</h1>\r\n")
            .append(" The site " + context.getInitParameter("title"))
            .append(" was visited for " + count + " times.\r\n")
            .append(" </body>\r\n")
            .append("</html>\r\n");
}
```

5. In the deployment descriptor /WEB-INF/web.xml, create another servlet **myServlet2** of the same servlet class with similar servlet init parameter and URL pattern, where "1" is updated to "2".

   Useful hotkey: In web.xml, select the <servlet> and <servlet-mapping> tags of myServlet1, and use the hotkey **Ctrl + D** to duplicate the selected content. Update the <servlet-name>, <servlet-class>, <param-value>, and <url-pattern> for myServlet2.

6. Study the above doGet() method, which involves the Integer variable count and the **context attribute** counter. Different from parameters, attributes are like variables in the Java program. The web container will create/allocate a thread for each servlet request. At the same time, all the threads can access the ServletContext object and hence its context attribute counter.

   Using the **synchronized** code block guarantees that at most one thread can run the code block at any time. Using **synchronized(context)** makes the context object the lock for this code block. In other scenarios, we can also use the servlet object as a lock, i.e., **synchronized(this)**.

   Note also the typical flow for initializing context/session/request attributes. An attribute is an object and its default value is null. If we find that an attribute has a null value, we need to create a new object with the correct type and the suitable initial value and set the attribute with the setAttribute method.

7. Whenever you update some Java code in a web app, you need to **build** the web app again and **redeploy** it (Shift + F10). A web browser will be opened and display the default index page of the project at http://localhost:8080/Lab03/ (you need to update the **context root** to `Lab03` (see Lab 2 notes) and note also that the URL is case-sensitive for everything ***after*** the Tomcat's server URL).

8. Run the project and increase the visitor counter by accessing the two mapped URLs, respectively.

9. To **understand the life cycle of a Servlet**, add the following `init()` and `destroy()` methods to `VisitCounterServlet`. Then, deploy the web app, access the URLs again, and finally undeploy it.

```java
@Override
public void init() {
    System.out.println("Servlet " + this.getServletName() + " has started.");
}

@Override
public void destroy() {
    System.out.println("Servlet " + this.getServletName() + " has stopped.");
}
```