

## Lesson 01 Demo 04

### Life Cycle of Servlet

**Objective:** To understand the life cycle of a Servlet and how the init(), service(), and destroy() methods work

**Tools Required:** Eclipse IDE

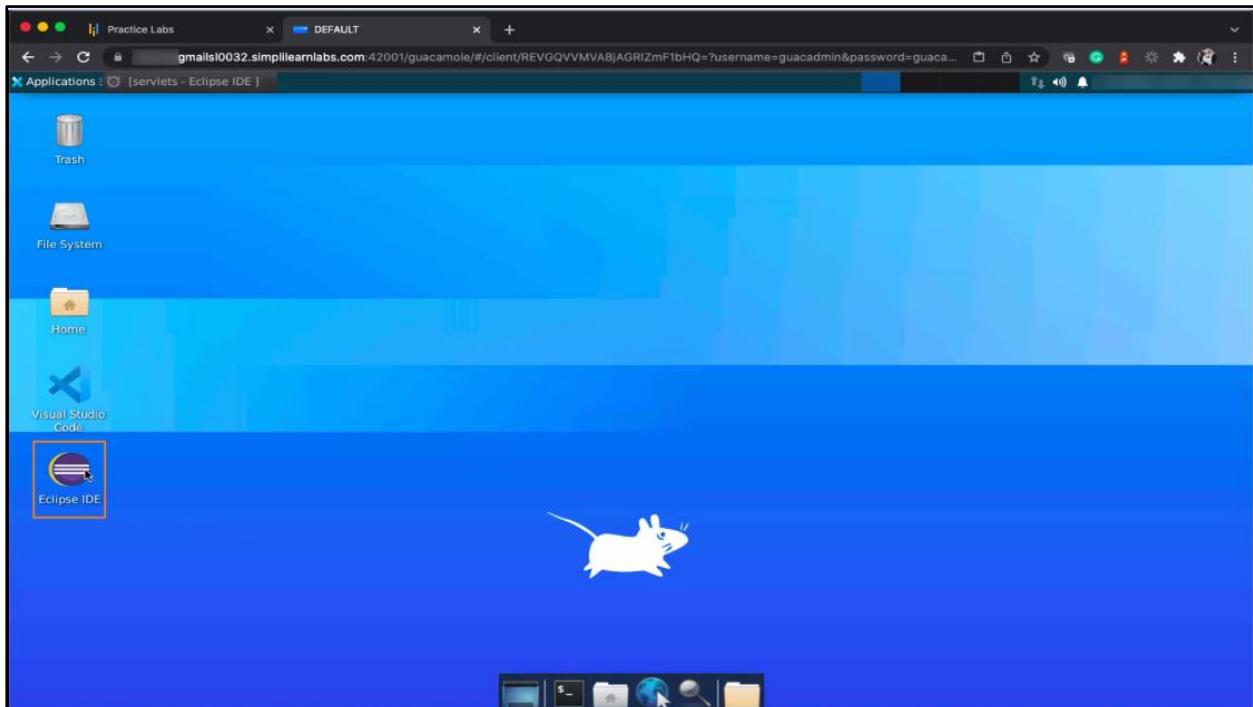
**Prerequisites:** None

#### Steps to be followed:

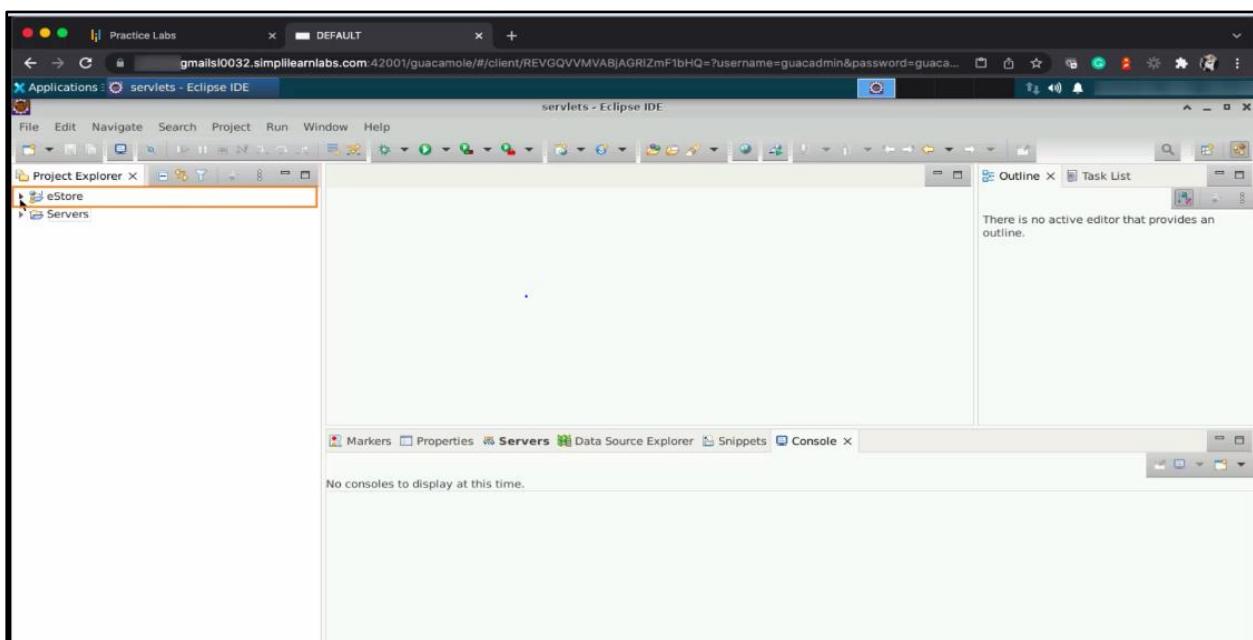
1. Create the init and destroy method
2. Make changes in internal service
3. Create a hyperlink for Hello Servlet
4. Run code to check how the life cycle works

#### Step 1: Create the init and destroy method

## 1.1 Open Eclipse IDE

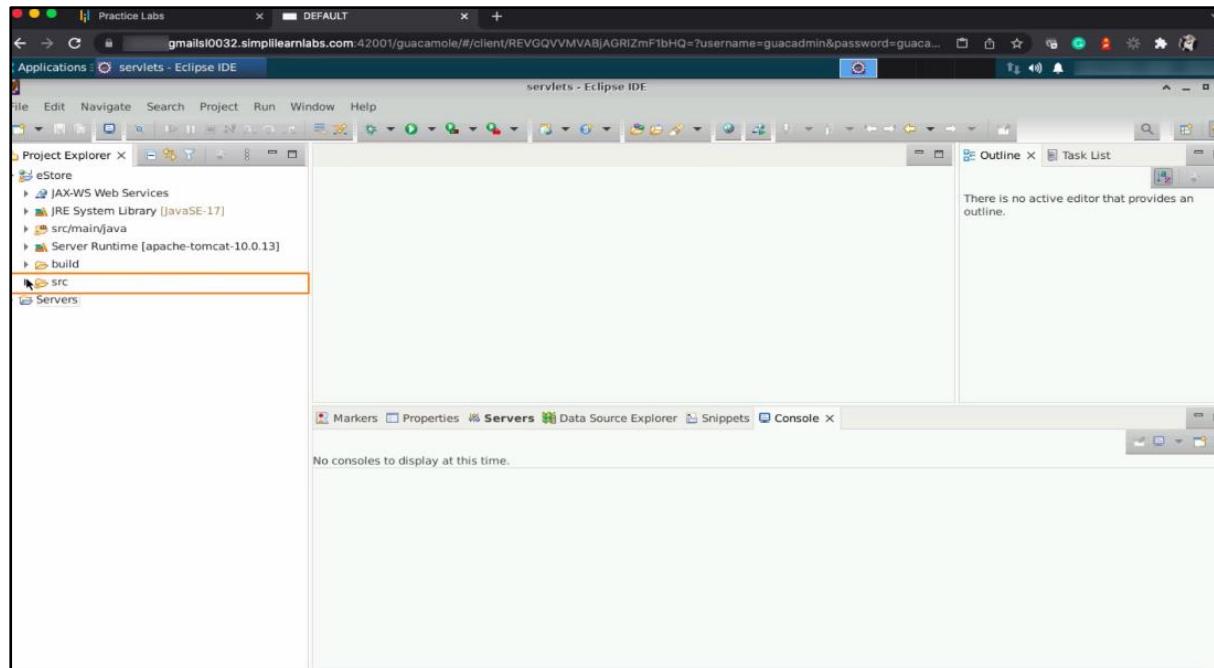


## 1.2 Open the eStore project

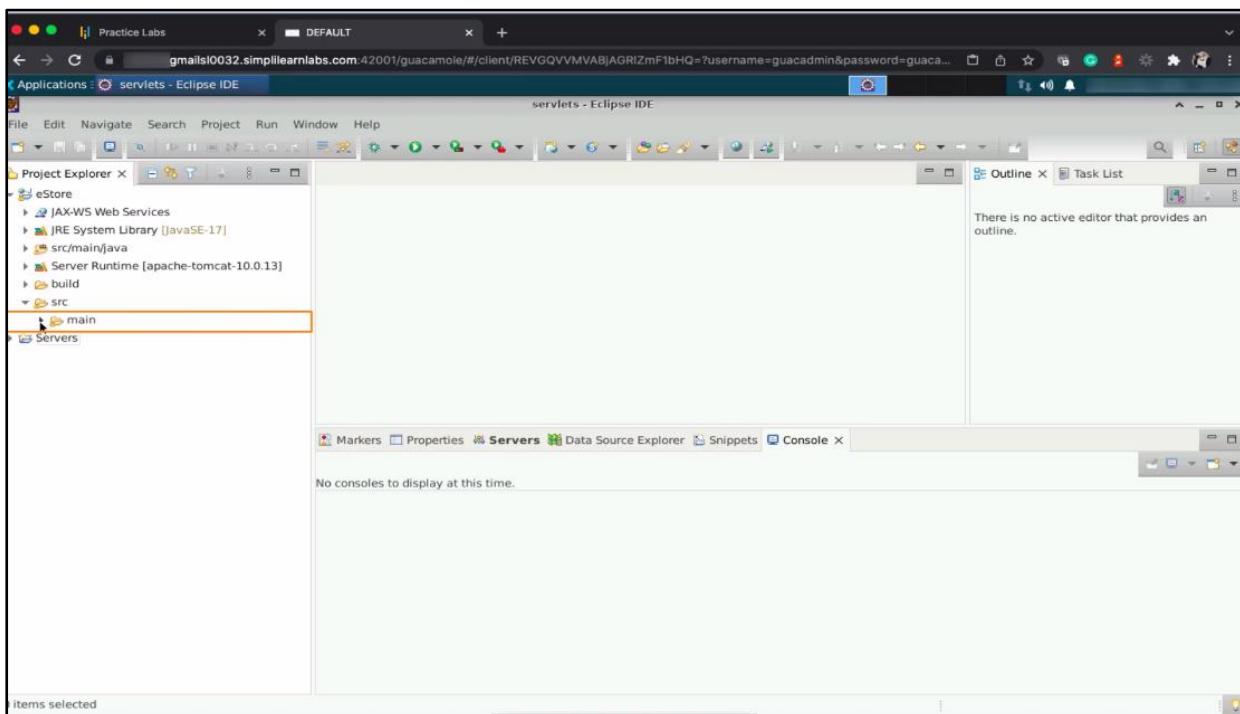


**Note:** Please refer to the previous demos on how to create the eStore project

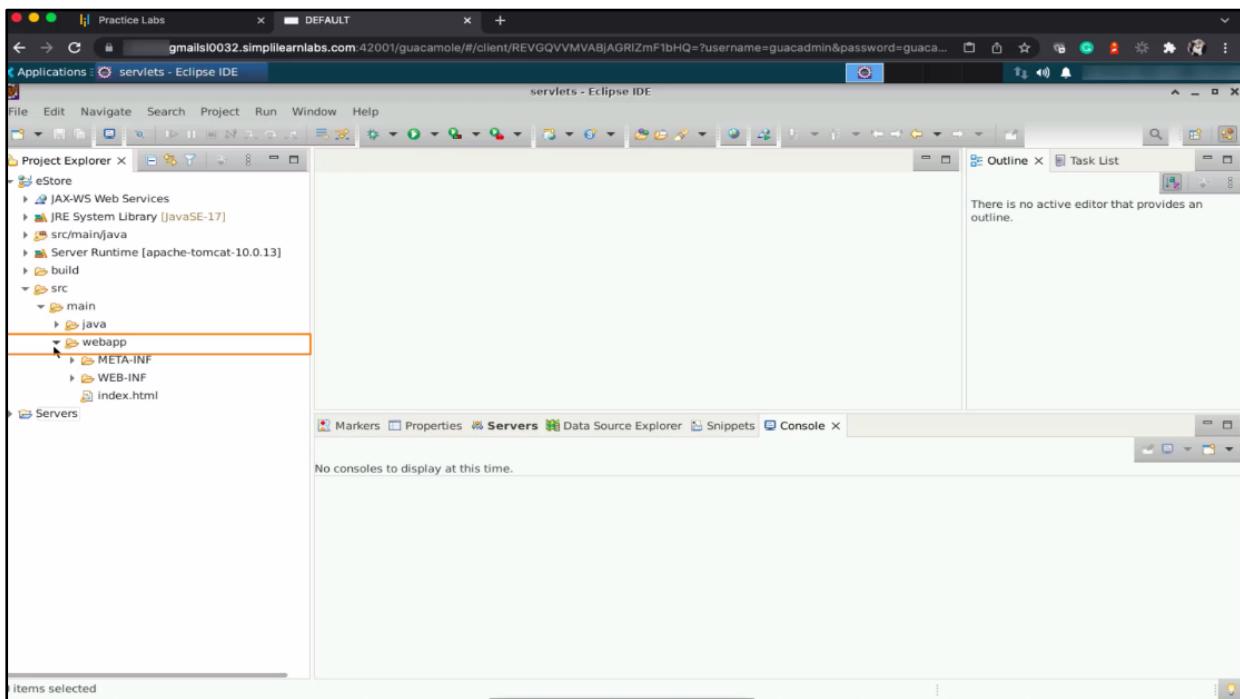
### 1.3 Go to the **src** directory



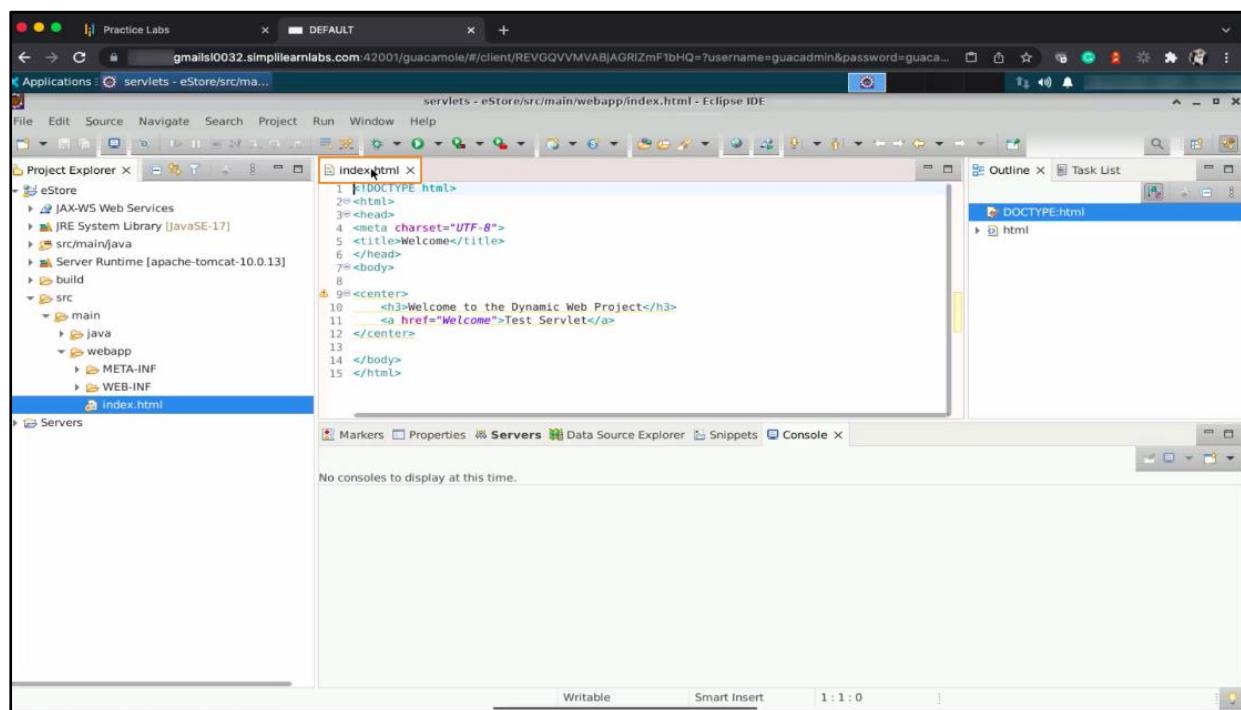
## 1.4 Select the main folder



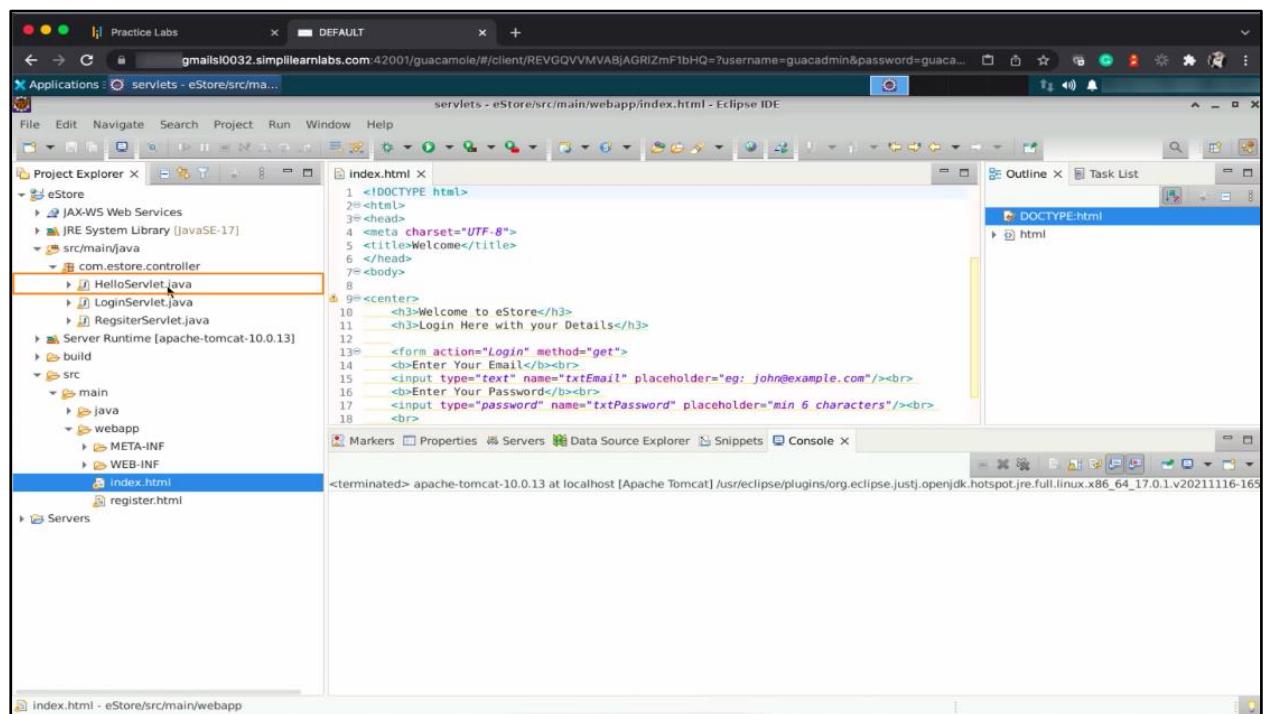
## 1.5 Open the webapp folder



1.6 Open the **index.html** file to see the HTML form that will be used later



1.7 Now, open the **HelloServlet.java** file to define the methods



## 1.8 Define the method `init()` and override it in the parent class `HttpServlet`

```

dex.html  *HelloServlet.java X
package com.estore.controller;

import jakarta.servlet.ServletConfig;

/**
 * Servlet implementation class HelloServlet
 */
public class HelloServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

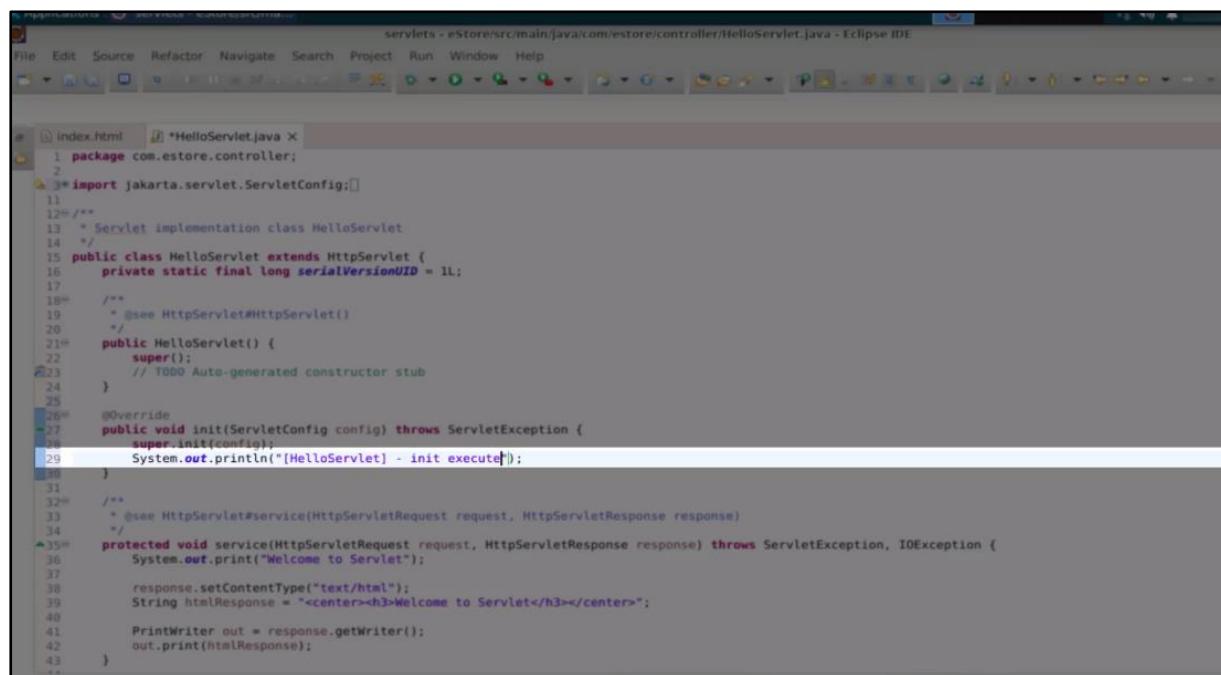
    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    /**
     * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
     */
    protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.print("Welcome to Servlet");

        response.setContentType("text/html");
        String htmlResponse = "<center><h3>Welcome to Servlet</h3></center>";
    }
}

```

## 1.9 Print the message `Hello Servlet` using the `System.out.println()` method



The screenshot shows the Eclipse IDE interface with the title bar "servlets - eStore/src/main/java/com/estore/controller/HelloServlet.java - Eclipse IDE". The code editor displays the HelloServlet.java file. The code is identical to the one shown in the previous code block, with the addition of a println statement in the init method:

```

1 package com.estore.controller;
2
3 import jakarta.servlet.ServletConfig;
4
5 public class HelloServlet extends HttpServlet {
6     private static final long serialVersionUID = 1L;
7
8     /**
9      * @see HttpServlet#HttpServlet()
10     */
11    public HelloServlet() {
12        super();
13        // TODO Auto-generated constructor stub
14    }
15
16    @Override
17    public void init(ServletConfig config) throws ServletException {
18        super.init(config);
19        System.out.println("[HelloServlet] - init execute");
20    }
21
22    /**
23     * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
24     */
25    protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
26        System.out.print("Welcome to Servlet");
27
28        response.setContentType("text/html");
29        String htmlResponse = "<center><h3>Welcome to Servlet</h3></center>";
30        PrintWriter out = response.getWriter();
31        out.print(htmlResponse);
32    }
33
34
35}

```

### 1.10 Write another `System.out.println()` method to print the message **service executed**

```

11
12 /**
13  * Servlet implementation class HelloServlet
14 */
15 public class HelloServlet extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     /**
19      * @see HttpServlet#HttpServlet()
20     */
21     public HelloServlet() {
22         super();
23         // TODO Auto-generated constructor stub
24     }
25
26     @Override
27     public void init(ServletConfig config) throws ServletException {
28         super.init(config);
29         System.out.println("[HelloServlet] - init executed");
30     }
31
32     /**
33      * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
34     */
35     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
36         System.out.println("[HelloServlet] - service executed");
37
38         response.setContentType("text/html");
39         String htmlResponse = "<center><h3>Welcome to Servlet</h3></center>";
40
41         PrintWriter out = response.getWriter();
42         out.print(htmlResponse);
43     }
44 }
45

```

### 1.11 Create the `destroy()` method, which will be executed when the Servlet instance is going to be destructed

```

@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    System.out.println("[HelloServlet] - init executed");
}

/**
 * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
 */
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("[HelloServlet] - service executed");

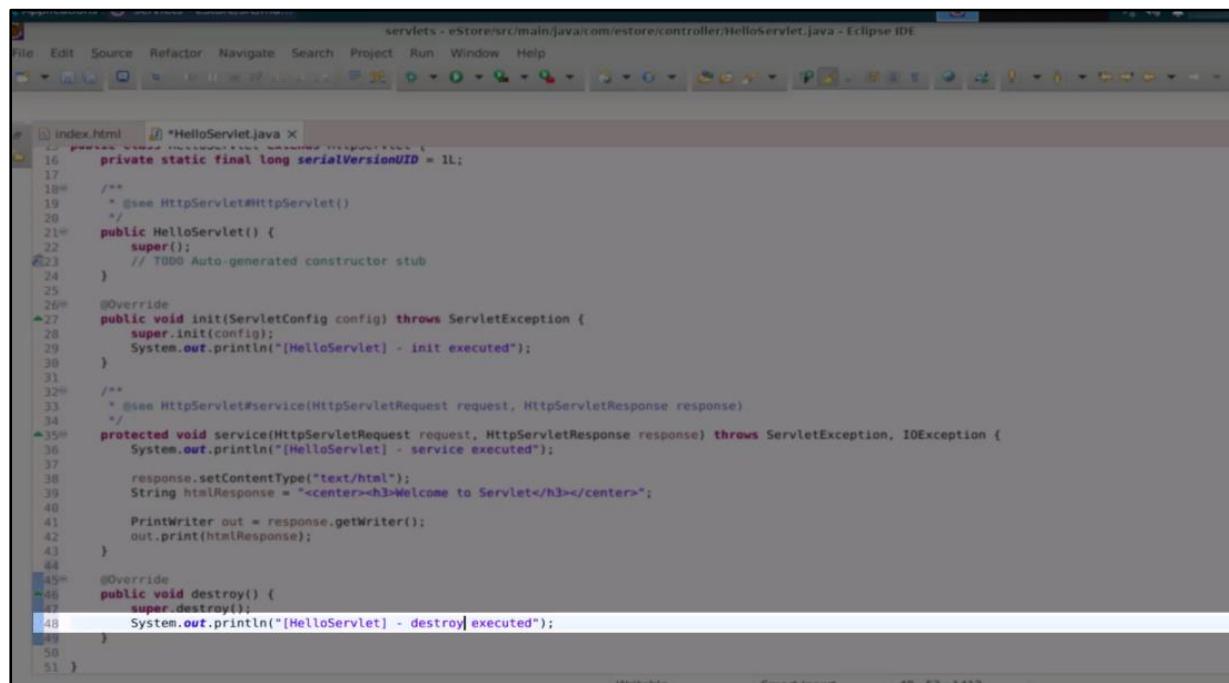
    response.setContentType("text/html");
    String htmlResponse = "<center><h3>Welcome to Servlet</h3></center>";

    PrintWriter out = response.getWriter();
    out.print(htmlResponse);
}

@Override
public void destroy() {
    super.destroy();
}

```

### 1.12 Write the message as **destroy** executed

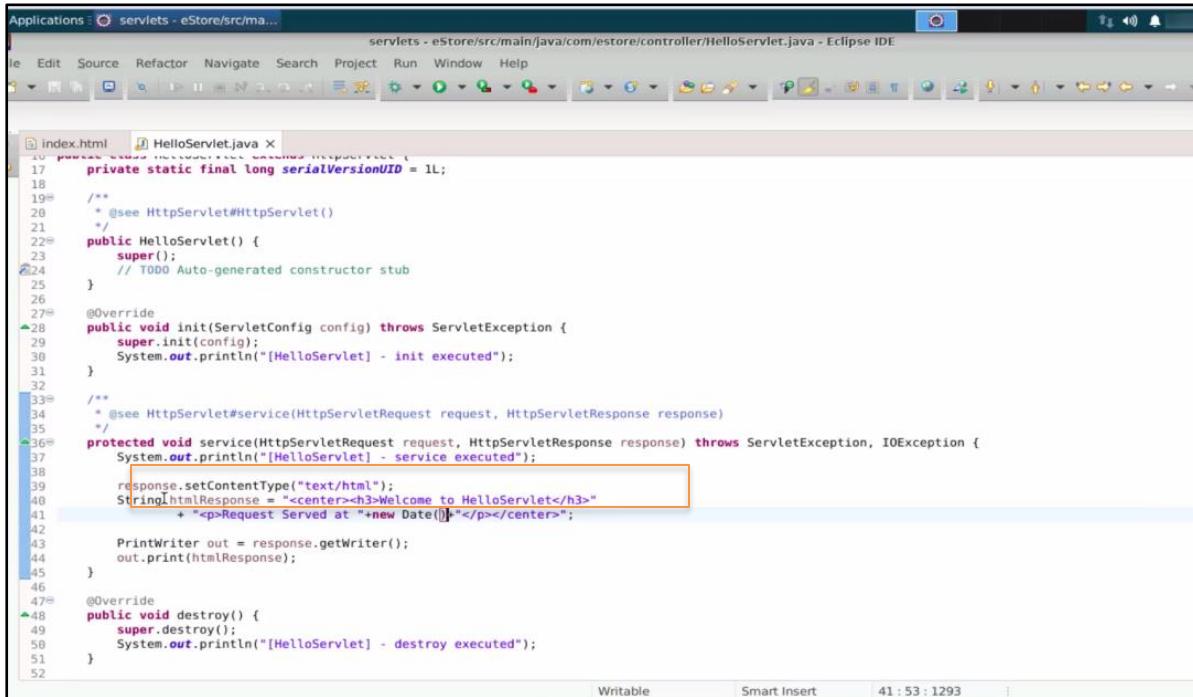


The screenshot shows the Eclipse IDE interface with the title bar "servlets - eStore/src/main/java/com/estore/controller/HelloServlet.java - Eclipse IDE". The code editor displays Java code for a servlet named HelloServlet. The code includes annotations like @Override and @see, and methods like init and service. A cursor is positioned at the end of the word "executed" in the destroy method's println statement.

```
index.html *HelloServlet.java
16  private static final long serialVersionUID = 1L;
17
18  /**
19   * @see HttpServlet#HttpServlet()
20   */
21  public HelloServlet() {
22      super();
23      // TODO Auto-generated constructor stub
24  }
25
26  @Override
27  public void init(ServletConfig config) throws ServletException {
28      super.init(config);
29      System.out.println("[HelloServlet] - init executed");
30  }
31
32  /**
33   * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
34   */
35  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
36      System.out.println("[HelloServlet] - service executed");
37
38      response.setContentType("text/html");
39      String htmlResponse = "<center><h3>Welcome to Servlet</h3></center>";
40
41      PrintWriter out = response.getWriter();
42      out.print(htmlResponse);
43  }
44
45  @Override
46  public void destroy() {
47      super.destroy();
48      System.out.println("[HelloServlet] - destroy| executed");
49  }
50
51 }
```

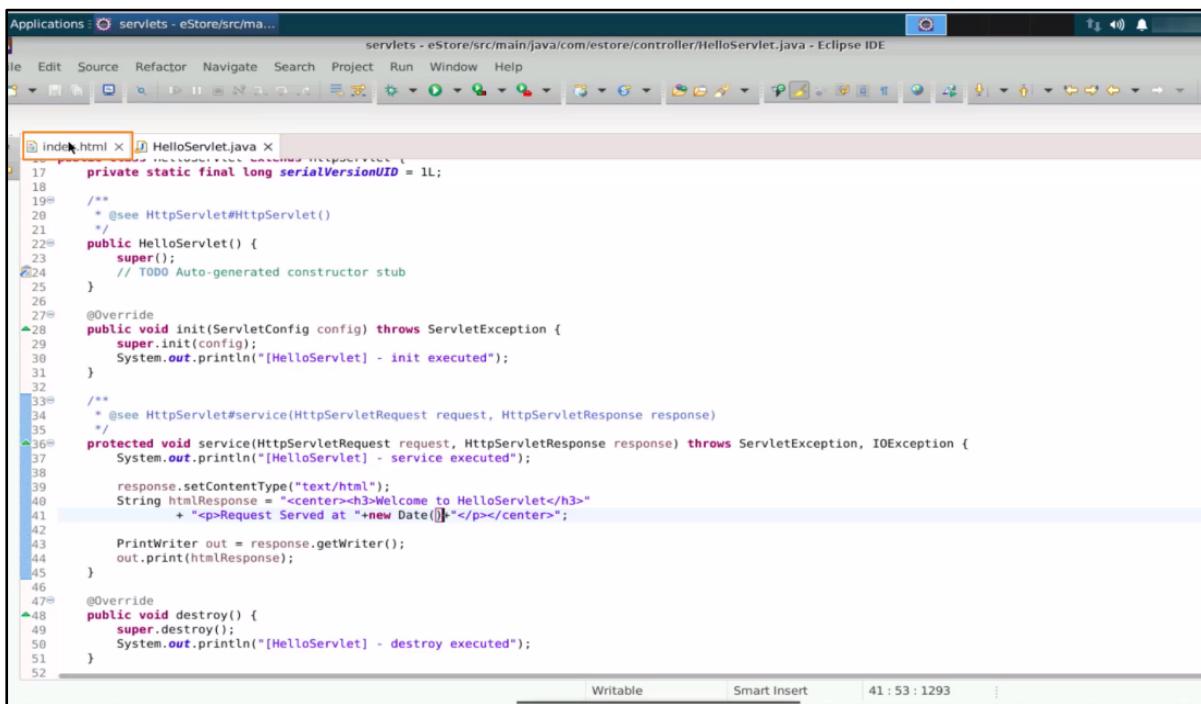
## Step 2: Make changes in internal service

2.1 Update the HTML response as **Welcome to HelloServlet**. Add a **paragraph** tag and print the timestamp at which the request was served (lines 39 and 40)



```
index.html  HelloServlet.java X
10  package com.estore.controller;
11
12  import javax.servlet.http.HttpServlet;
13
14  import javax.servlet.http.HttpServletRequest;
15  import javax.servlet.http.HttpServletResponse;
16
17  private static final long serialVersionUID = 1L;
18
19  /**
20   * @see HttpServlet#HttpServlet()
21   */
22  public HelloServlet() {
23      super();
24      // TODO Auto-generated constructor stub
25  }
26
27  /**
28  * @see HttpServlet#init(javax.servlet.ServletConfig)
29  */
30  public void init(javax.servlet.ServletConfig config) throws ServletException {
31      super.init(config);
32      System.out.println("[HelloServlet] - init executed");
33  }
34
35  /**
36  * @see HttpServlet#service(javax.servlet.http.HttpServletRequest,
37  *      javax.servlet.http.HttpServletResponse)
38  */
39  protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
40      System.out.println("[HelloServlet] - service executed");
41      response.setContentType("text/html");
42      String htmlResponse = "<center><h3>Welcome to HelloServlet</h3></center>" +
43          "+ "<p>Request Served at "<new Date()>"</p></center>";
44      PrintWriter out = response.getWriter();
45      out.print(htmlResponse);
46  }
47
48  /**
49  * @see HttpServlet#destroy()
50  */
51  public void destroy() {
52      super.destroy();
53      System.out.println("[HelloServlet] - destroy executed");
54  }
55
56
```

## 2.2 Open the index.html page



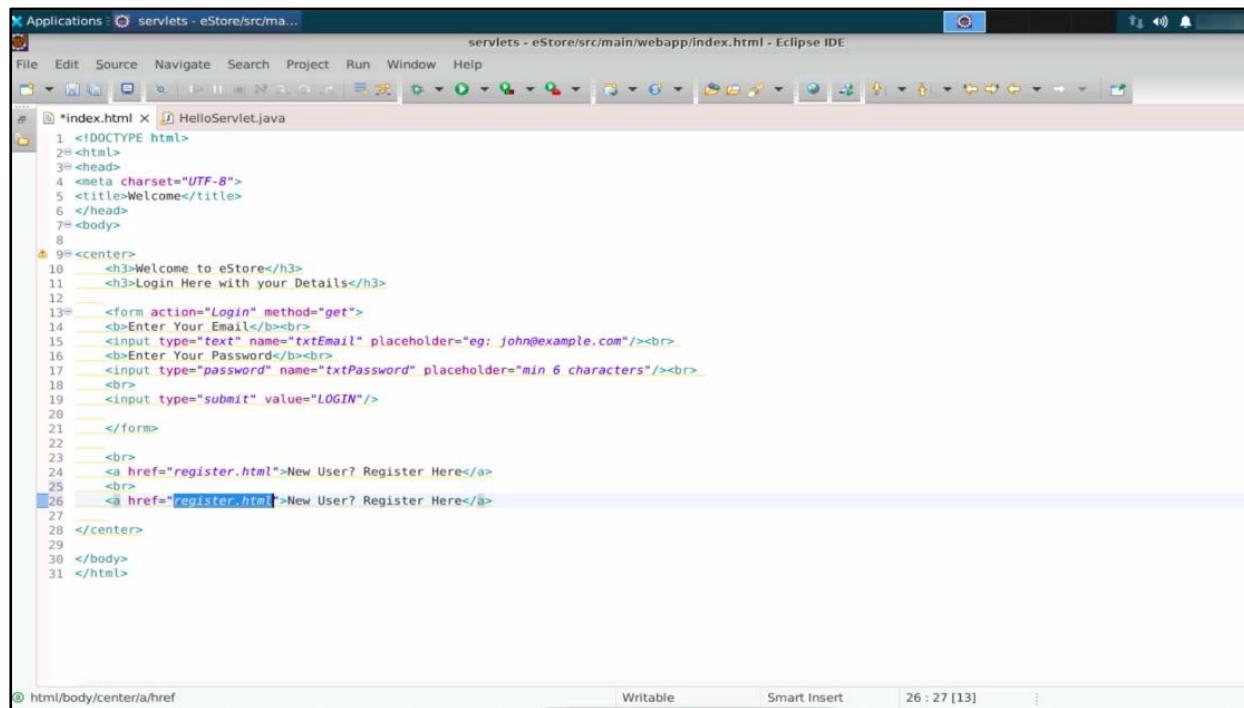
The screenshot shows the Eclipse IDE interface with two files open in the editor:

- index.html**: A file containing an HTML document with a single line of code: <p>Request Served at <new Date()></p>
- HelloServlet.java**: A Java servlet class with the following code:

```
private static final long serialVersionUID = 1L;
/*
 * @see HttpServlet#HttpServlet()
 */
public HelloServlet() {
    super();
    // TODO Auto-generated constructor stub
}
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    System.out.println("[HelloServlet] - init executed");
}
/** 
 * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)
 */
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("[HelloServlet] - service executed");
    response.setContentType("text/html");
    String htmlResponse = "<center><h3>Welcome to HelloServlet</h3>" +
        "<p>Request Served at <new Date()></p></center>";
    PrintWriter out = response.getWriter();
    out.print(htmlResponse);
}
@Override
public void destroy() {
    super.destroy();
    System.out.println("[HelloServlet] - destroy executed");
}
```

## Step 3: Create a hyperlink for Hello Servlet

### 3.1 Create a href link that will send a request to the HelloServlet

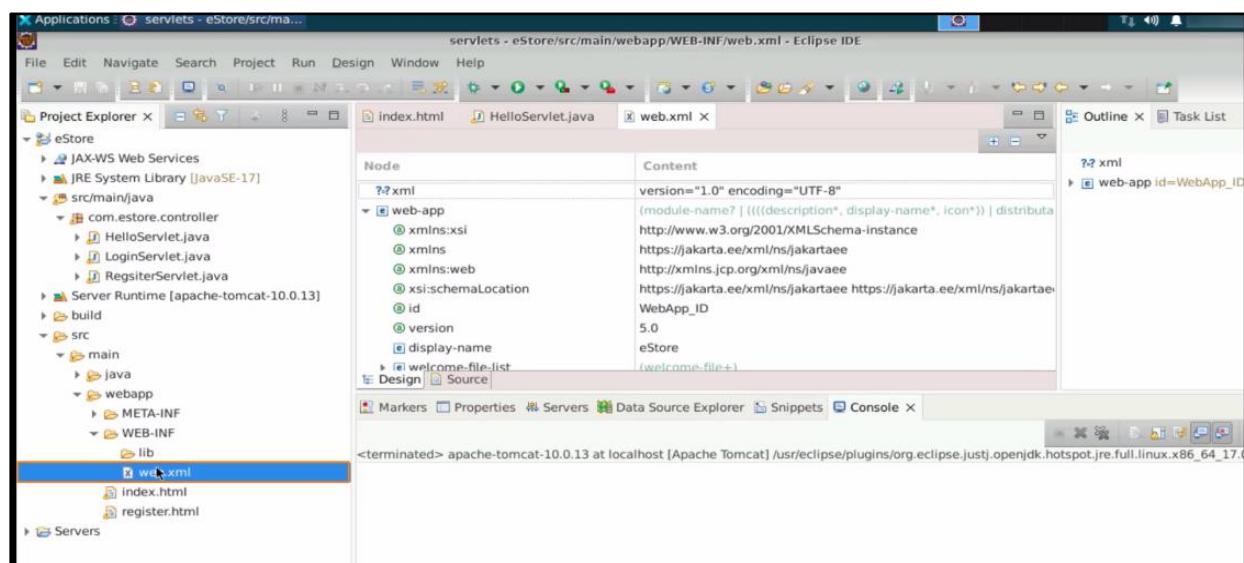


```

    *index.html x HelloServlet.java
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Welcome</title>
6 </head>
7 <body>
8
9 <center>
10 <h3>Welcome to eStore</h3>
11 <h3>Login Here with your Details</h3>
12
13 <form action="Login" method="get">
14 <b>Enter Your Email</b><br>
15 <input type="text" name="txtEmail" placeholder="eg: john@example.com"/><br>
16 <b>Enter Your Password</b><br>
17 <input type="password" name="txtPassword" placeholder="min 6 characters"/><br>
18 <br>
19 <input type="submit" value="LOGIN"/>
20
21 </form>
22
23 <br>
24 <a href="register.html">New User? Register Here</a>
25 <br>
26 <a href="register.html">New User? Register Here</a>
27
28 </center>
29
30 </body>
31 </html>

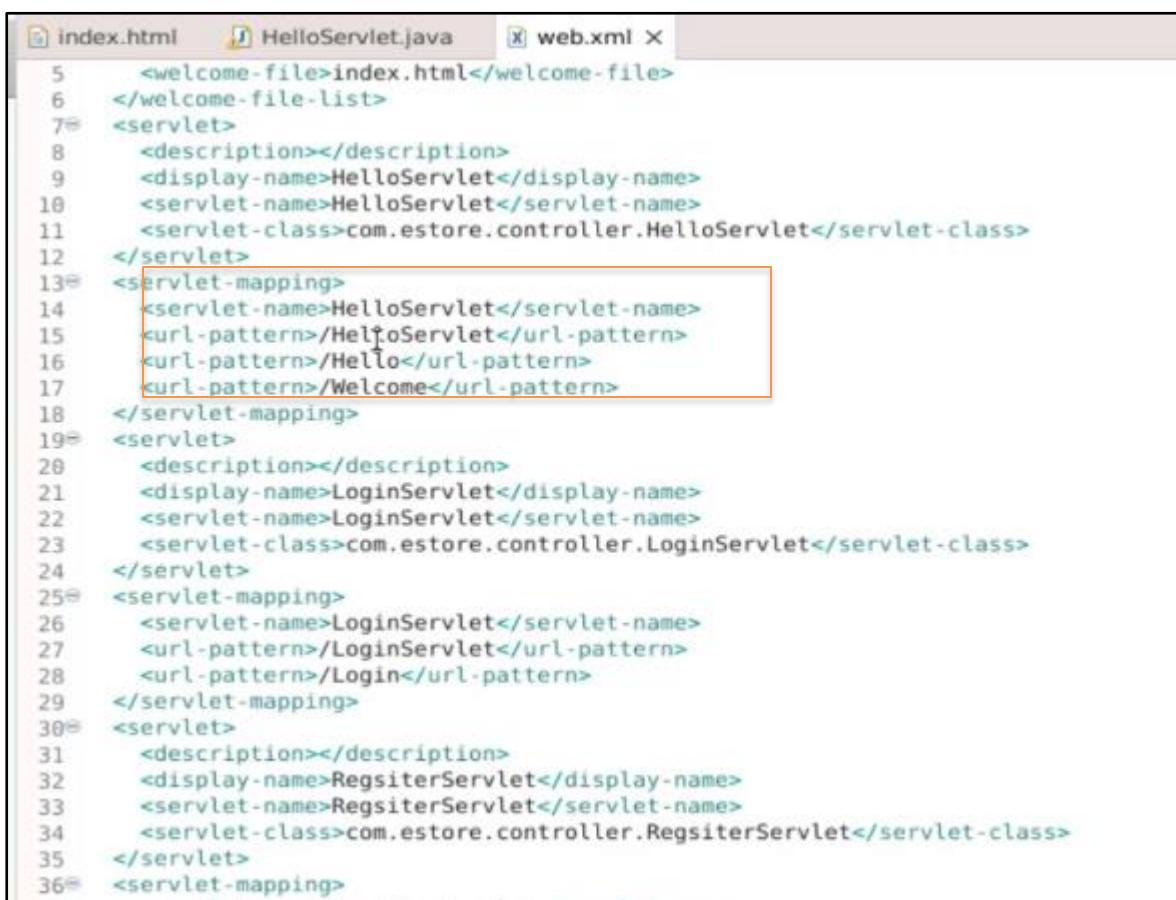
```

### 3.2 Open the web.xml file and copy the <url-pattern> for HelloServlet



Node	Content
??xml	version="1.0" encoding="UTF-8"
web-app	<web-app> <display-name>eStore</display-name> <welcome-file-list> <welcome-file>index.html</welcome-file> </welcome-file-list> </web-app>

<terminated> apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86\_64\_17.0.1

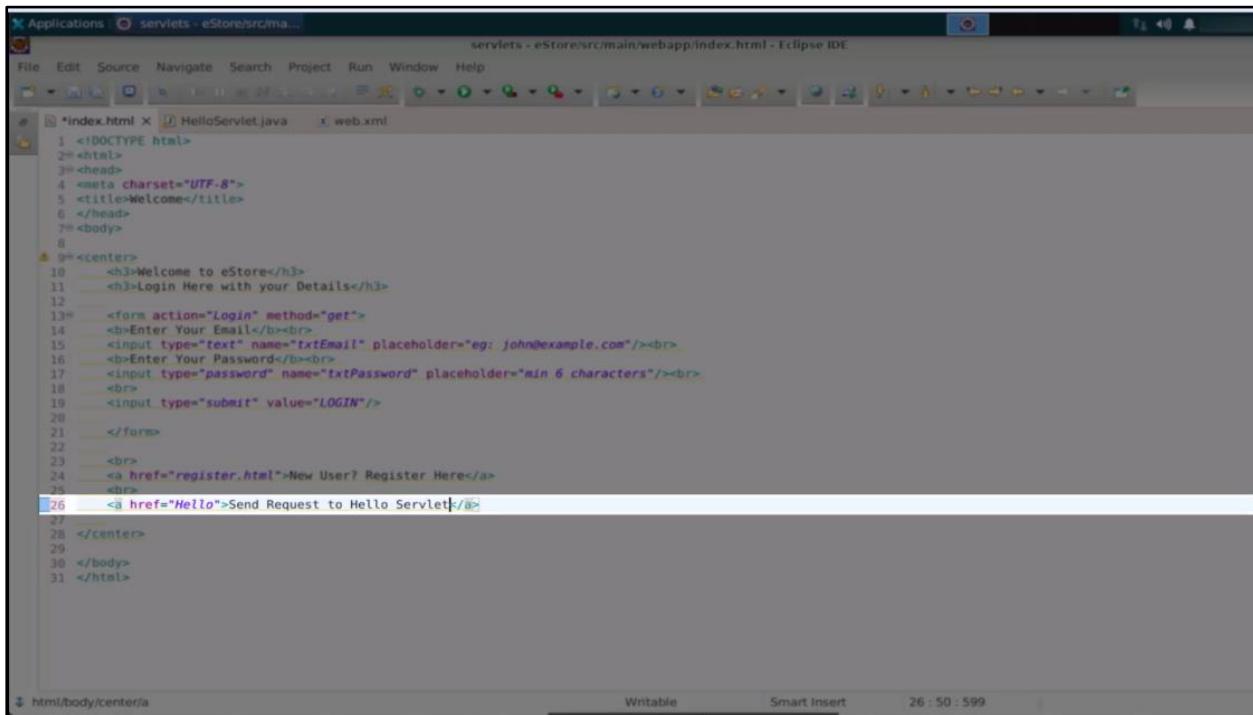


The screenshot shows a code editor with three tabs: index.html, HelloServlet.java, and web.xml. The web.xml tab is active and displays the following XML configuration:

```
index.html  HelloServlet.java  web.xml X
5      <welcome-file>index.html</welcome-file>
6  </welcome-file-list>
7< servlet>
8      <description></description>
9      <display-name>HelloServlet</display-name>
10     <servlet-name>HelloServlet</servlet-name>
11     <servlet-class>com.estore.controller.HelloServlet</servlet-class>
12 </servlet>
13< servlet-mapping>
14     <servlet-name>HelloServlet</servlet-name>
15     <url-pattern>/HelloServlet</url-pattern>
16     <url-pattern>/Hello</url-pattern>
17     <url-pattern>/Welcome</url-pattern>
18 </servlet-mapping>
19< servlet>
20     <description></description>
21     <display-name>LoginServlet</display-name>
22     <servlet-name>LoginServlet</servlet-name>
23     <servlet-class>com.estore.controller.LoginServlet</servlet-class>
24 </servlet>
25< servlet-mapping>
26     <servlet-name>LoginServlet</servlet-name>
27     <url-pattern>/LoginServlet</url-pattern>
28     <url-pattern>/Login</url-pattern>
29 </servlet-mapping>
30< servlet>
31     <description></description>
32     <display-name>RegsiterServlet</display-name>
33     <servlet-name>RegsiterServlet</servlet-name>
34     <servlet-class>com.estore.controller.RegisiterServlet</servlet-class>
35 </servlet>
36< servlet-mapping>
```

The section from line 13 to line 18 is highlighted with a red box.

### 3.3 Go back to the **index.html** page and add the URL mapping inside the **href** tag

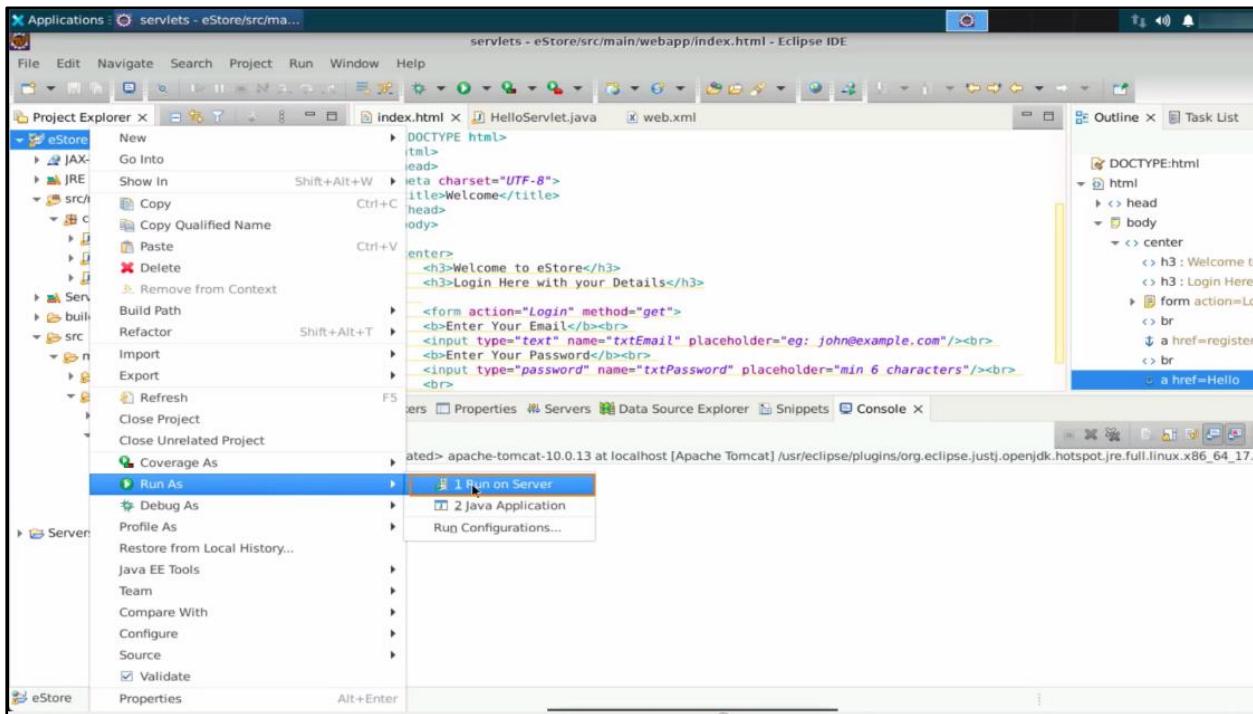


The screenshot shows the Eclipse IDE interface with the title bar "Applications - servlets - eStore/src/main/webapp/index.html - Eclipse IDE". The editor window displays the content of the index.html file. The code includes HTML tags for a welcome message, a login form, and two links: one to register.html and one to HelloServlet. The line number 26 is highlighted, showing the href attribute for the second link.

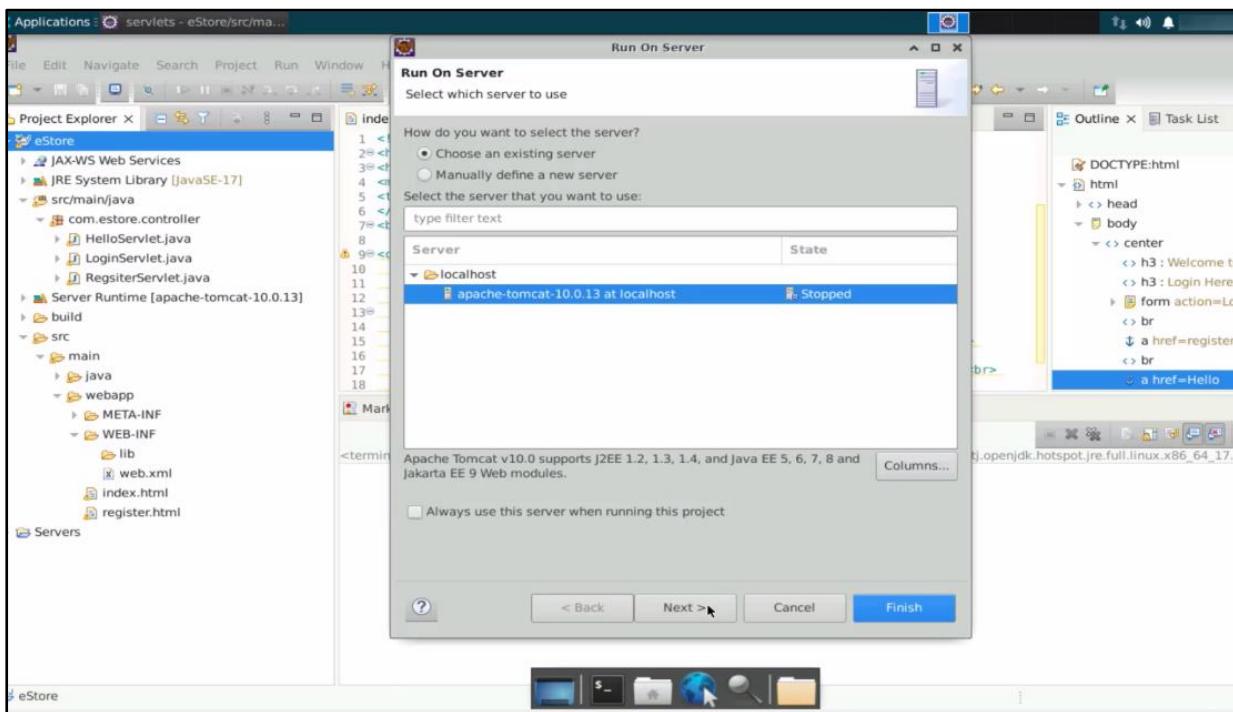
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Welcome</title>
</head>
<body>
<center>
<h3>Welcome to eStore</h3>
<h3>Login Here with your Details</h3>
<form action="Login" method="get">
<b>Enter Your Email:</b><br>
<input type="text" name="txtEmail" placeholder="eg: john@example.com"/><br>
<b>Enter Your Password:</b><br>
<input type="password" name="txtPassword" placeholder="min 6 characters"/><br>
<br>
<input type="submit" value="LOGIN"/>
</form>
<br>
<a href="register.html">New User? Register Here</a>
<br>
<a href="Hello">Send Request to Hello Servlet</a>
</center>
</body>
</html>
```

## Step 4: Run code to check how the lifecycle method works

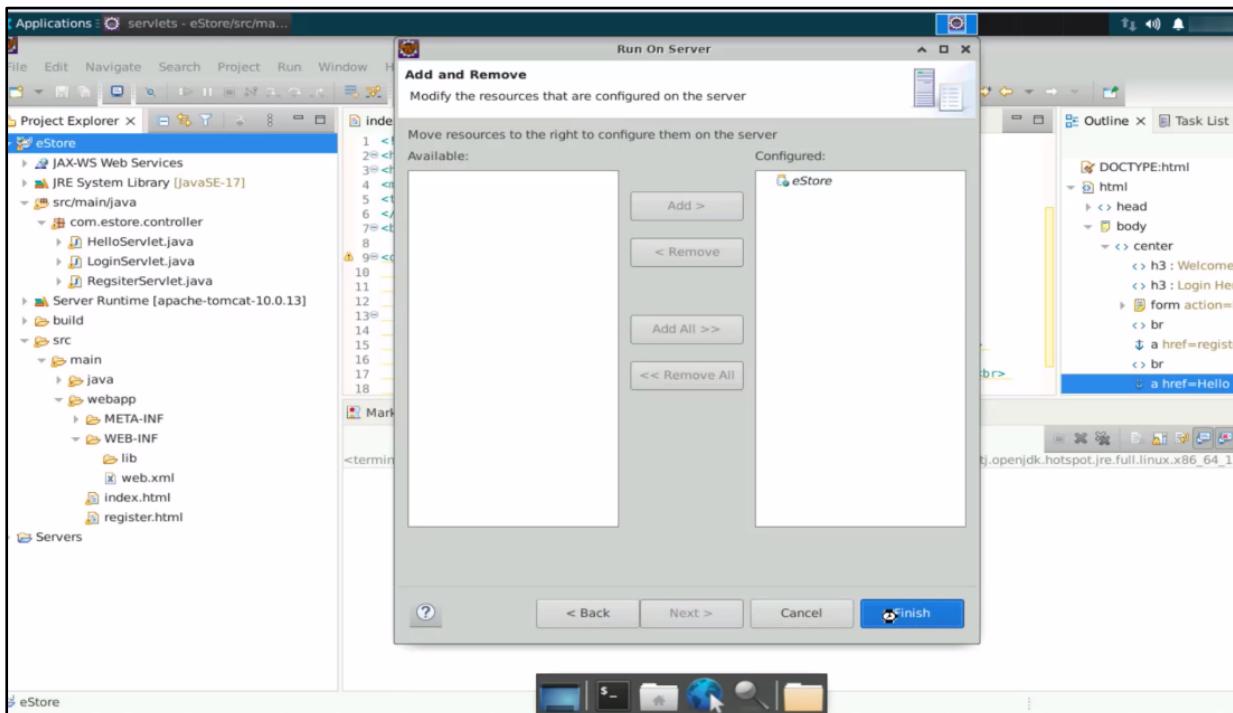
4.1 Save and run the code on the server. Right-click on the project, click on **Run As**, and select **Run on Server**



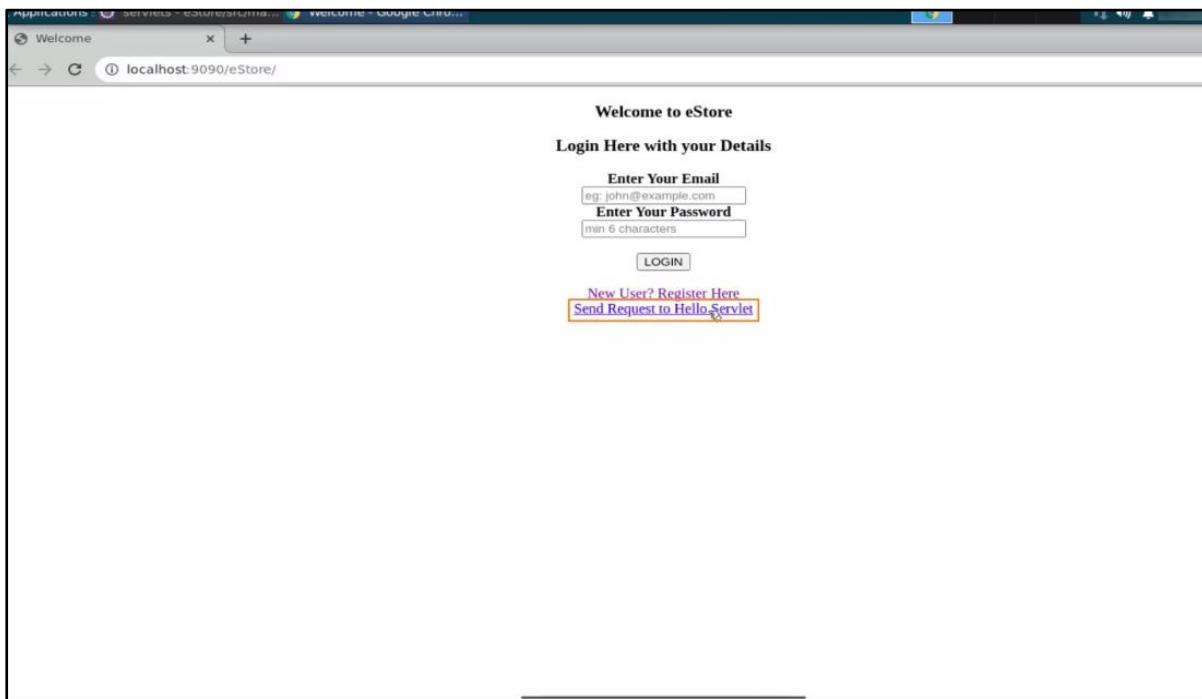
#### 4.2 Choose the apache-tomcat-10.0.13 at localhost server and click Next



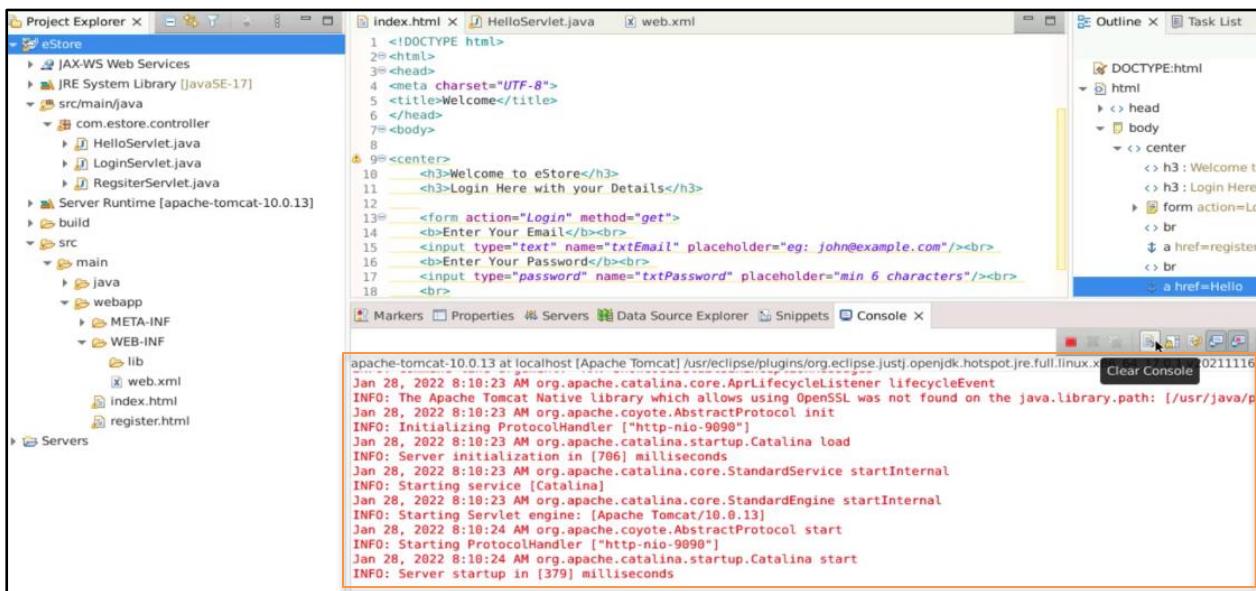
#### 4.3 Click Finish



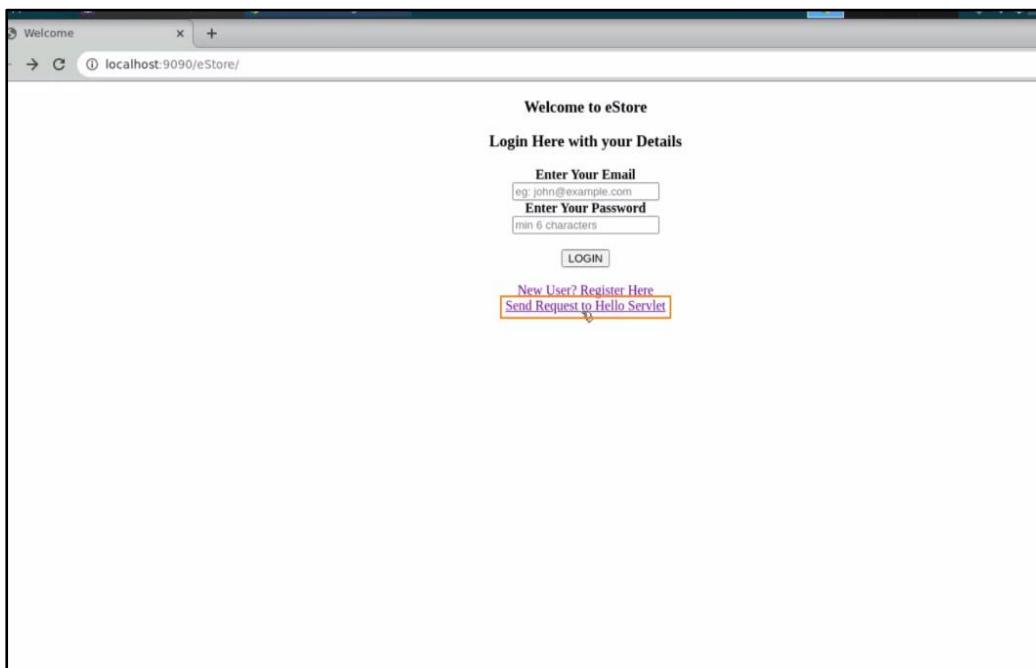
You can see the hyperlink as **Send Request to Hello Servlet**.



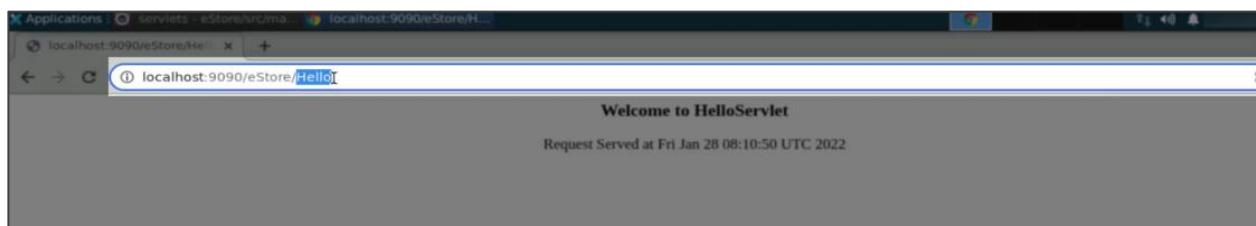
4.4 Go back to **Eclipse IDE** and check the **Console** tab. You will not be able to see any log or print statement as shown below:



4.5 Go back and click on the **Send Request to Hello Servlet** link



You can see that the **HelloRequest** has been forwarded to the **Hello Servlet**, and it prints the timestamp of when the request was served.



4.6 Go back to the Console tab. You can see that the two methods, init and service, are executed.

The screenshot shows the Eclipse IDE interface. On the left, there are three tabs: index.html, HelloServlet.java, and web.xml. The index.html tab contains the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Welcome</title>
6 </head>
7 <body>
8
9 <center>
10 <h3>Welcome to eStore</h3>
11 <h3>Login Here with your Details</h3>
12
13 <form action="Login" method="get">
14 <b>Enter Your Email</b><br>
15 <input type="text" name="txtEmail" placeholder="eg: john@example.com"/><br>
16 <b>Enter Your Password</b><br>
17 <input type="password" name="txtPassword" placeholder="min 6 characters"/><br>
18 <br>
```

The HelloServlet.java tab shows a Java class definition. The web.xml tab is partially visible. On the right, the Outline view shows the structure of the HTML document. At the bottom, the Console tab displays the following output:

```
apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116
[HelloServlet] - init executed
[HelloServlet] - service executed
```

4.7 Reclick on the hyperlink in the browser. You will see one more service executed in the **Console** tab, as the init method gets executed only once.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a Java web application project structure with files like DOCTYPE.html, index.jsp, and HelloServlet.java.
- Java Editor:** Displays the code for index.jsp, which includes HTML for a login form and a link to HelloServlet.
- Console Tab:** Shows log entries from the Apache Tomcat server:

```
apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v2021111
[HelloServlet] - init executed
[HelloServlet] - service executed
[HelloServlet] - service executed
```

The last log entry is highlighted with a blue box.

4.8 In **HelloServlet.java**, use the **config** method and get the **init** parameter from the server, which can be a URL

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "eStore".
- Editor:** Displays the Java code for **HelloServlet.java**. The code includes an **init** method that prints the value of the **url** init parameter.
- Terminal:** A separate window shows the output of the application running on Apache Tomcat 10.0.13. It displays the message "[HelloServlet] - init executed" followed by "[HelloServlet] - service executed".

```
private static final long serialVersionUID = 1L;
/*
 * @see HttpServlet#HttpServlet()
 */
public HelloServlet() {
    super();
    // TODO Auto-generated constructor stub
}
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    System.out.println("[HelloServlet] - init executed");
    //String url = config.getInitParameter("url");
}

```

apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full/linux.x86\_64\_17.0.1.v20211116-[HelloServlet] - init executed  
[HelloServlet] - service executed  
[HelloServlet] - service executed

4.9 Go to the **Console** tab in the **HelloServlet.java** file. You will see that the Servlet is automatically destroyed, and it prints **destroy executed**.

The screenshot shows the Eclipse IDE interface. In the top left, there are three tabs: index.html, HelloServlet.java (which is currently selected), and web.xml. Below these tabs is the Java code for HelloServlet.java:

```
16  public class HelloServlet extends HttpServlet {  
17      private static final long serialVersionUID = 1L;  
18  
19      /**  
20       * @see HttpServlet#HttpServlet()  
21       */  
22      public HelloServlet() {  
23          super();  
24          // TODO Auto-generated constructor stub  
25      }  
26  
27      @Override  
28      public void init(ServletConfig config) throws ServletException {  
29          super.init(config);  
30          System.out.println("[HelloServlet] - init executed");  
31          //String url = config.getInitParameter("url");  
32      }  
33  }
```

In the bottom right corner of the code editor, there is a red rectangular box highlighting the "Console" tab. The "Console" tab is active, showing the following log output:

```
apache-tomcat-10.0.13 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hots  
[HelloServlet] - init executed  
[HelloServlet] - service executed  
[HelloServlet] - service executed  
Jan 28, 2022 8:13:34 AM org.apache.catalina.core.StandardContext reload  
INFO: Reloading Context with name [/eStore] has started  
[HelloServlet] - destroy executed I  
Jan 28, 2022 8:13:34 AM org.apache.catalina.core.StandardContext reload  
INFO: Reloading Context with name [/eStore] is completed
```

**Observation:** There are two important callbacks associated with the Servlet lifecycle: the **init** and **destroy** methods. Between these two steps, an HTTP method should be implemented. The **init** and **destroy** methods will be executed only once.