

## กิจกรรมที่ 4 สร้าง และใช้งาน Web Service

### 1) วัตถุประสงค์

เรียนรู้หลักการสร้างและใช้งาน Web Service แบบ RESTful พื้นฐาน และการทดสอบการทำงานของ Web Service ที่สร้าง และการทำงานกับโครงสร้างไฟล์ข้อมูลแบบ JSON

### 2) เครื่องมือหรือไฟล์ที่ใช้สำหรับกิจกรรม (เพิ่มเติมจากเครื่องมือที่ใช้ในกิจกรรมที่ 3)

- โปรแกรม Eclipse แบบ JavaEE
- Tomcat Server 9.0
- การเชื่อมต่อเครือข่ายอินเทอร์เน็ตของเครื่องคอมพิวเตอร์ขณะฝึกปฏิบัติตามกิจกรรม เนื่องจากอาจมีการดาวน์โหลดข้อมูลบางส่วนจาก resource ที่โปรแกรมเรียกใช้งาน เช่น Jersey library

### ข้อแนะนำเบื้องต้น

1. ถ้ามีขีดสีแดงใต้ข้อความหรือคำสั่งใด ๆ ให้ตรวจสอบการพิมพ์ตัวสะกดอักษร ว่าตรงกับตัวอย่างภาพหรือไม่
2. บางครั้งเมื่อแทรกคำสั่งโค้ดในโปรแกรมแล้ว ต้องทำการ save ไฟล์ก่อนเสมอ เพื่อให้โปรแกรม eclipse โหลดข้อมูลบางส่วนทางเน็ตก่อนการรันหรือใช้งานโปรแกรมได้ (อาจใช้เวลาพอสมควร ให้สังเกต status ของโปรแกรมทางมุมล่างขวา)

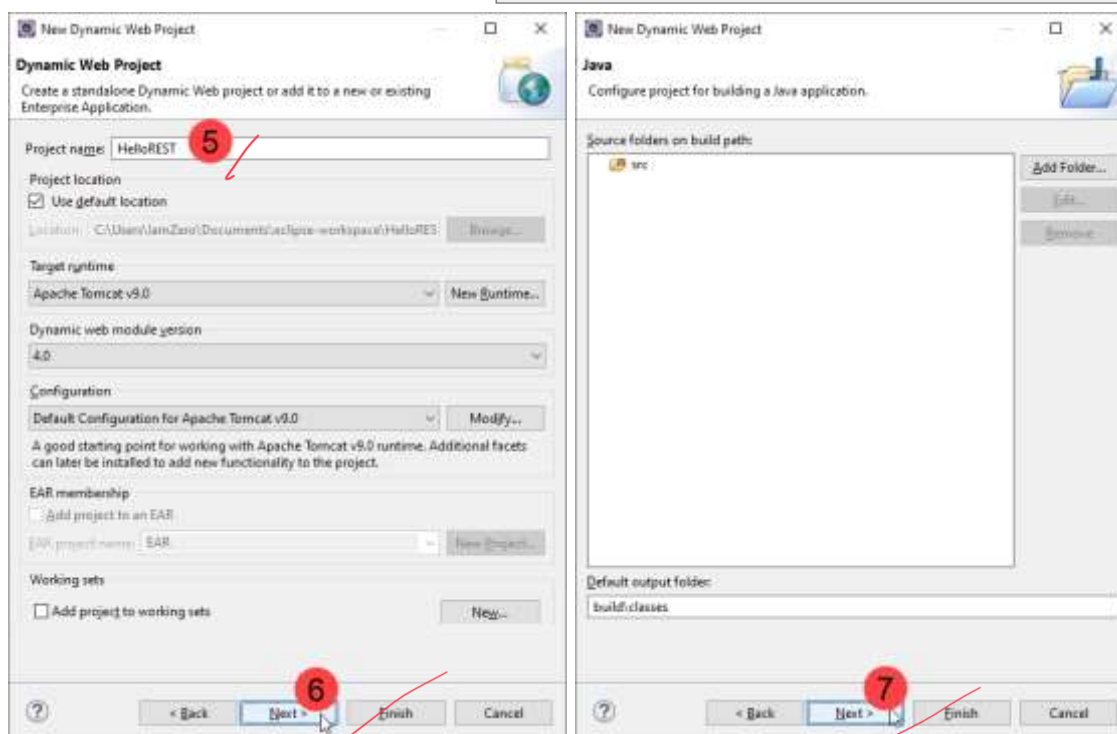
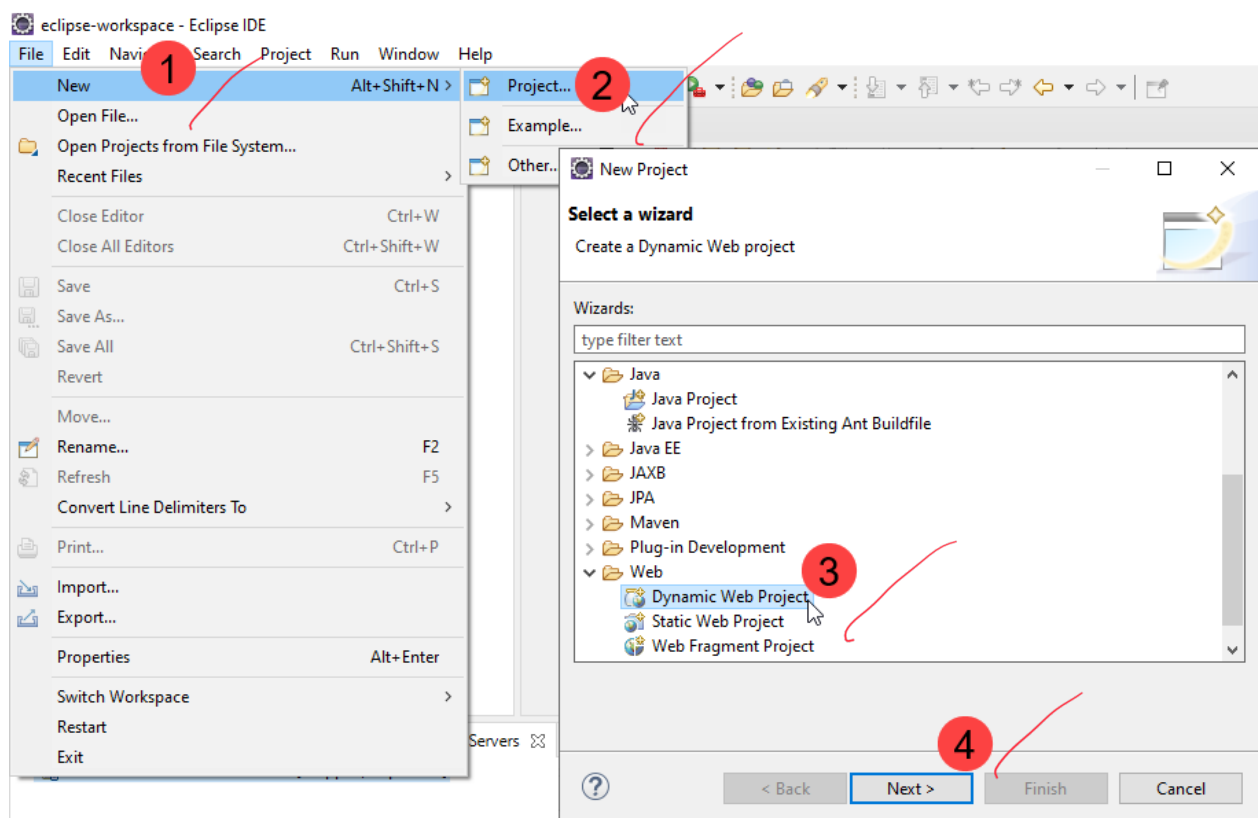
### 3) รายละเอียดกิจกรรม

กิจกรรมที่ 4 ประกอบด้วยกิจกรรมย่อยที่ต้องฝึกปฏิบัติตามลำดับกิจกรรม ดังนี้

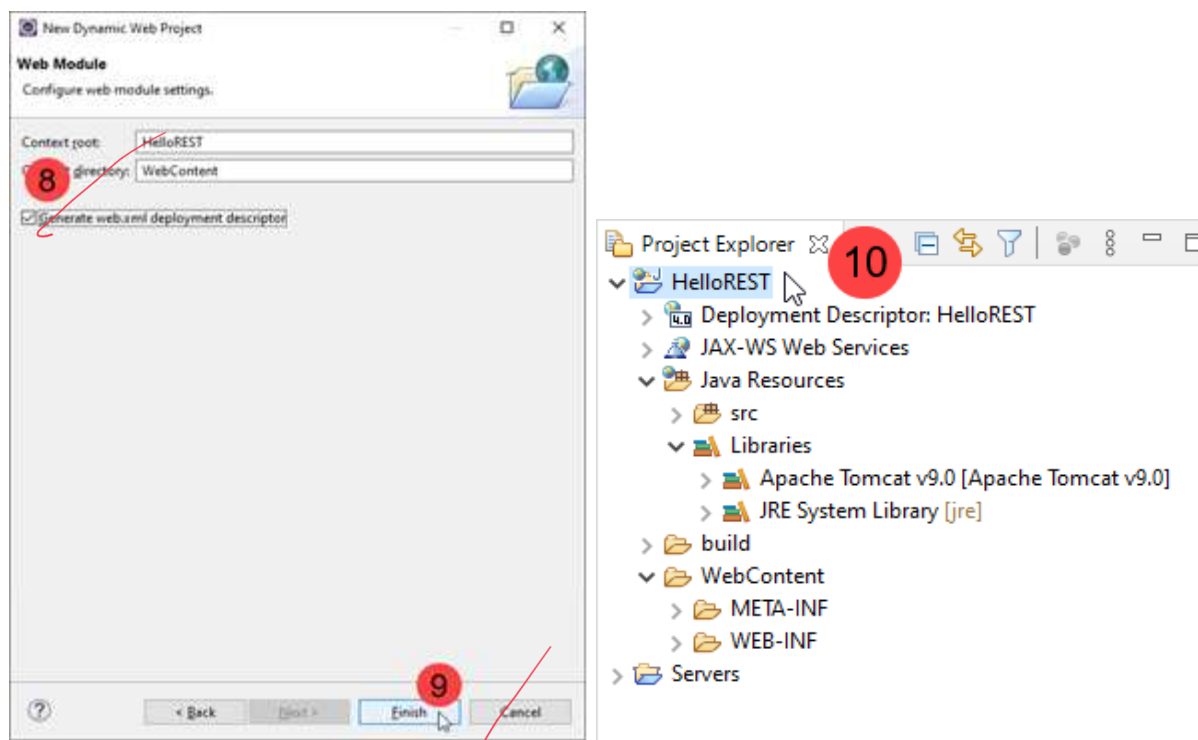
#### กิจกรรม 4.1 การสร้างไฟล์โปรเจกต์ทำงานทางฝั่งเซิร์ฟเวอร์

เป็นการสร้างไฟล์โปรเจกต์เพื่อให้ทำงานทางฝั่งเซิร์ฟเวอร์ในรูปแบบการใช้ Jersey library ร่วมกับเครื่องมือ build tool แบบ Maven ในโปรแกรม eclipse เพื่อสร้าง RESTful Web Service ดังนี้

- 4.1.1. สร้างไฟล์โปรเจกต์แบบ ~~Dynamic Web Project~~ และตั้งชื่อไฟล์โปรเจกต์เป็น ~~HettoREST~~ พร้อมทั้งกำหนดสร้างไฟล์ web.xml ดังภาพที่ 4.1.1



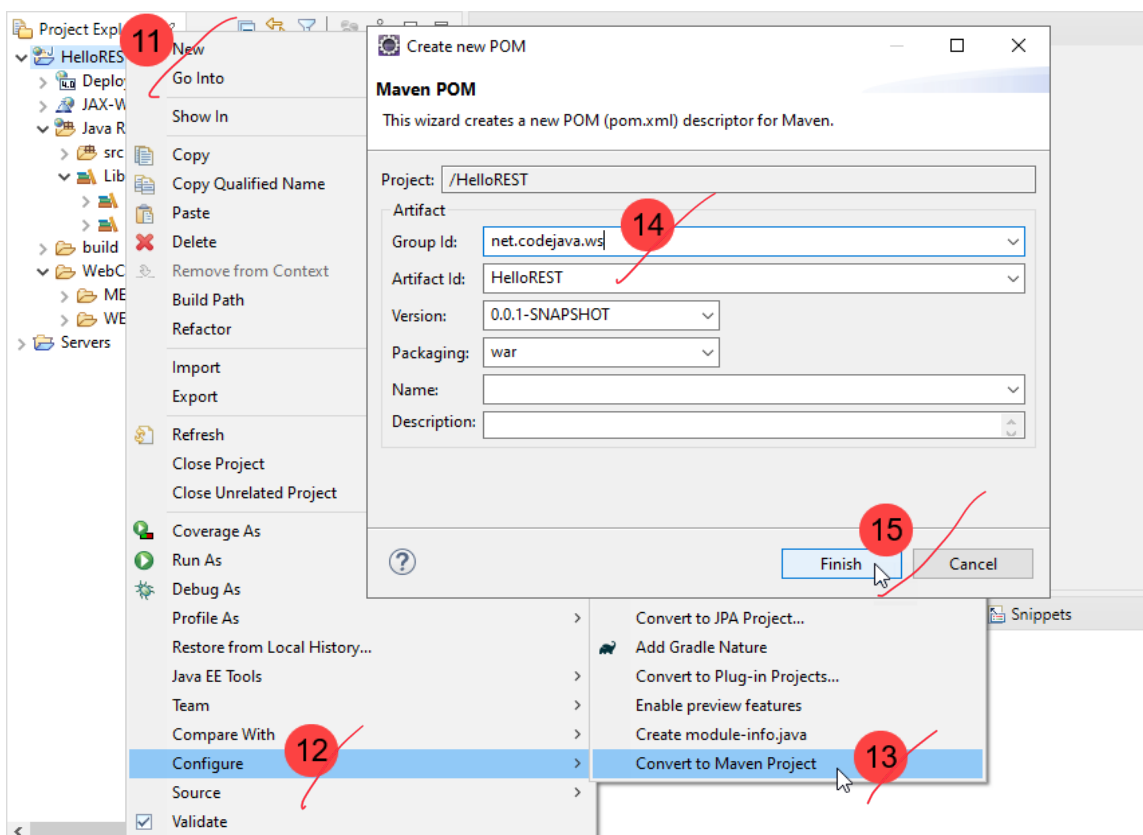
ก. ขั้นตอนการสร้างไฟล์โปรเจกต์ชื่อ HelloREST



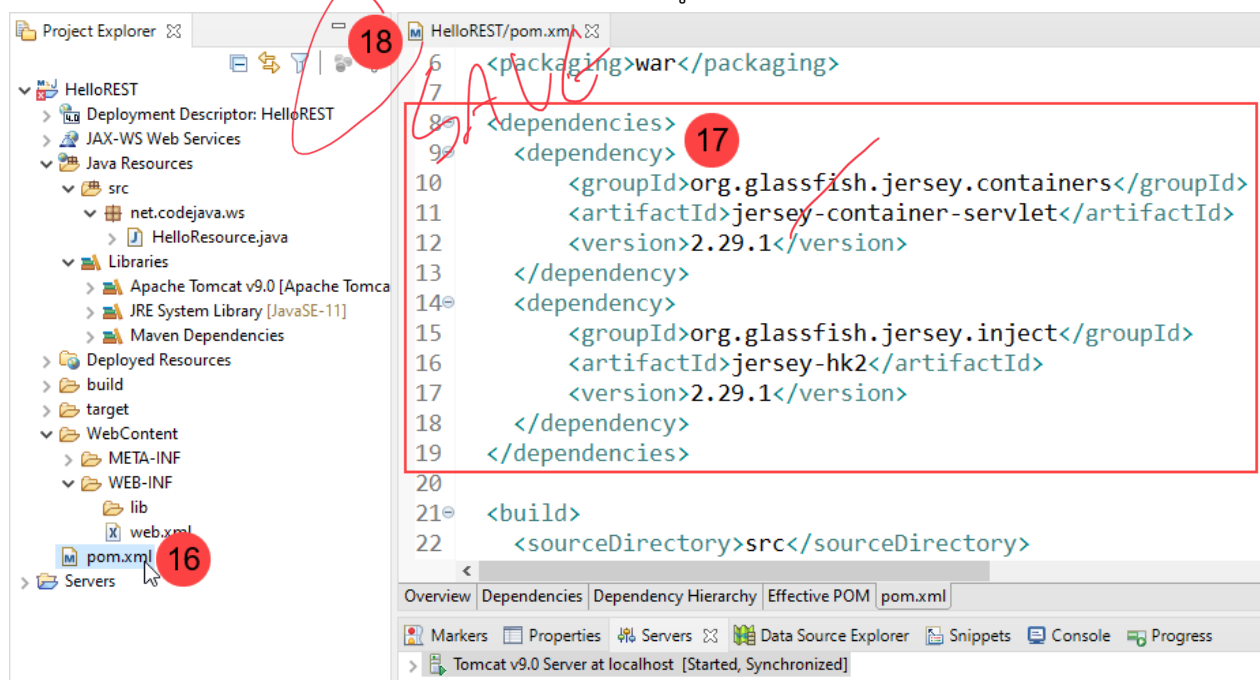
ข. การกำหนดไฟล์ web.xml และหน้าต่างผลลัพธ์ของการสร้างไฟล์โปรเจค HelloREST

ภาพที่ 4.1.1 ลำดับการสร้างและหน้าต่างผลลัพธ์ของไฟล์ HelloREST

4.1.2. ทำการปรับแต่ง (configure) ไฟล์ HelloREST ด้วยการแปลงไฟล์เป็นรูปแบบ Maven เพื่อสะดวกต่อการ  
ใช้ Jersey library ดังภาพที่ 4.1.2



ก. การปรับแต่งไฟล์เป็นรูปแบบ Maven

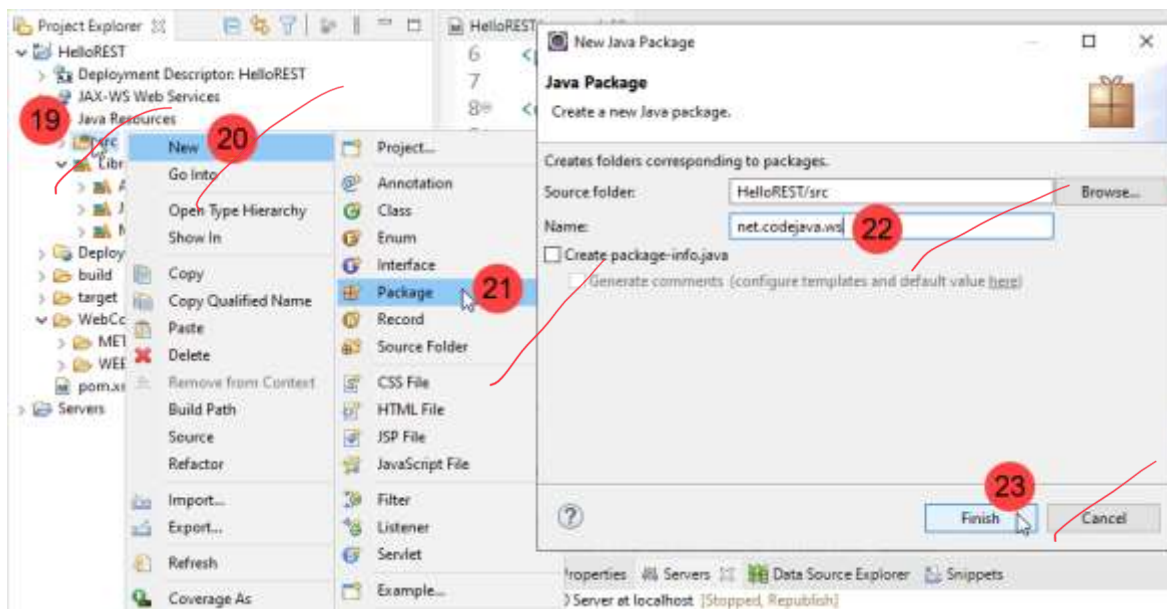


ข. หน้าต่างรูปแบบไฟล์ pom.xml และการเพิ่มแท็กคำสั่ง

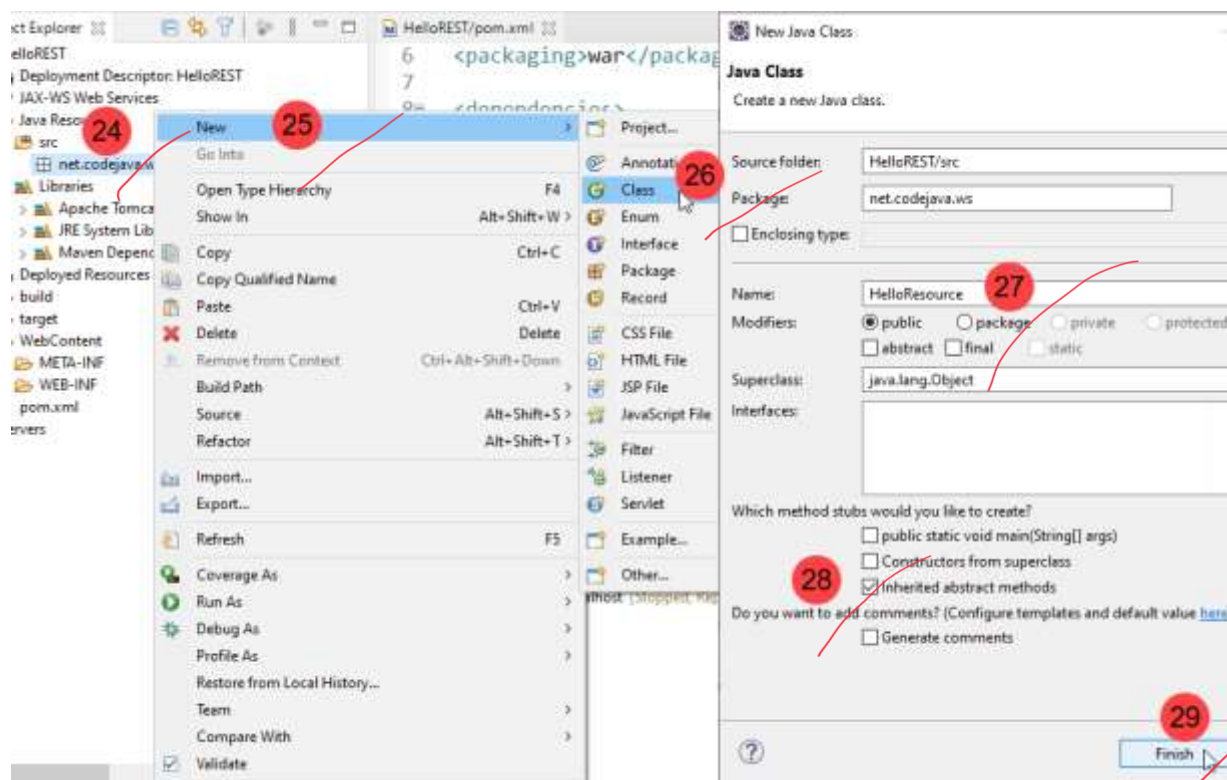
ภาพที่ 4.1.2 ลำดับการแปลงไฟล์เป็นแบบ Maven และเพิ่ม jersey library

เปิดไฟล์ pom.xml ตามหมายเลข 16 และแทรก เพิ่มแท็กคำสั่ง dependency ตามในกรอบหมายเลข 17 ลงในไฟล์ ดังภาพที่ 4.1.2 ข. แล้วทำการ save ไฟล์ (หมายเลข 18) และทำให้โปรแกรม eclipse ทำการ download ไฟล์ที่จำเป็นทางอินเทอร์เน็ต

4.1.3. ทำการสร้างไฟล์แบบ package ชื่อ net.codejava.ws และไฟล์แบบ class ชื่อ HelloResource.java ดังภาพที่ 4.1.3




ก. การสร้างไฟล์แบบ package



ข. การสร้างไฟล์แบบ class ชื่อ HelloResource ภายใต้ไฟล์ package

ภาพที่ 4.1.3 ลำดับการสร้างไฟล์ package และไฟล์ class

4.1.4. เพิ่มคำสั่งตามกรอบหมายเลข 30 ในไฟล์ HelloResource ดังภาพที่ 4.1.4 แล้ว save ไฟล์



```

1 package net.codejava.ws;
2
3 import javax.ws.rs.GET; 30
4 import javax.ws.rs.Path;
5 import javax.ws.rs.Produces;
6 import javax.ws.rs.core.MediaType;
7
8 @Path("/stou")
9 public class HelloResource {
10     @GET
11     @Produces(MediaType.TEXT_PLAIN)
12     public String getStou() {
13         return "STOU Welcome";
14     }
15 }
16
  
```

ภาพที่ 4.1.4 คำสั่งในไฟล์ HelloResource.java

4.1.5. เปิดไฟล์ web.xml เพื่อปรับแต่ง (configure) ด้วยการแทรก เพิ่มข้อมูลคำสั่ง Jersey Servlet ตามกรอบหมายเลข 32 ดังภาพที่ 4.1.5 แล้ว save ไฟล์



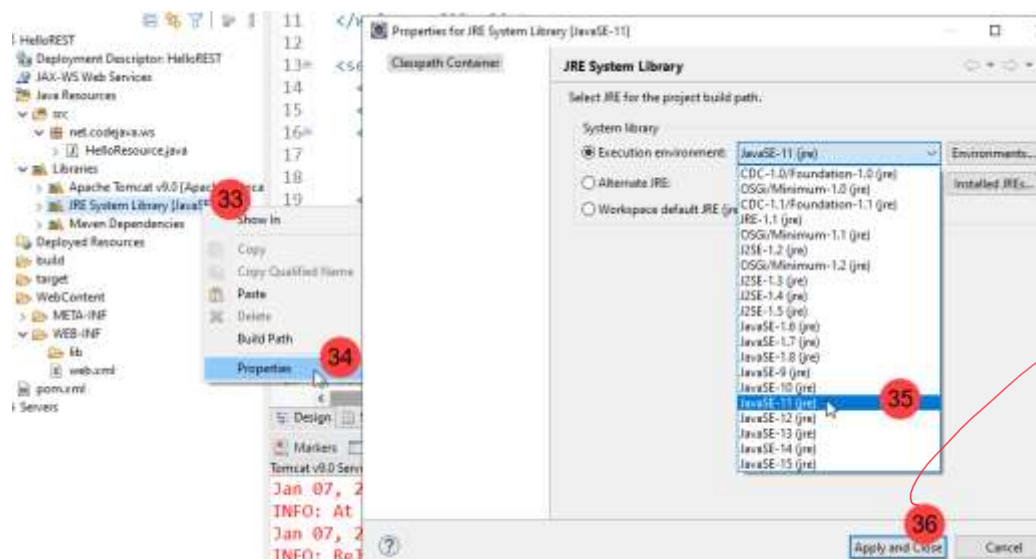
```

11 <welcome-file-list>
12
13 <servlet> 32
14     <servlet-name>Jersey REST Service</servlet-name>
15     <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
16     <init-param>
17         <param-name>jersey.config.server.provider.packages</param-name>
18         <param-value>net.codejava.ws</param-value>
19     </init-param>
20     <load-on-startup>1</load-on-startup>
21 </servlet>
22
23 <servlet-mapping>
24     <servlet-name>Jersey REST Service</servlet-name>
25     <url-pattern>/rest/*</url-pattern>
26 </servlet-mapping>
27 </web-app>
  
```

ภาพที่ 4.1.5 ชุดคำสั่ง Jersey Servlet ในไฟล์ web.xml

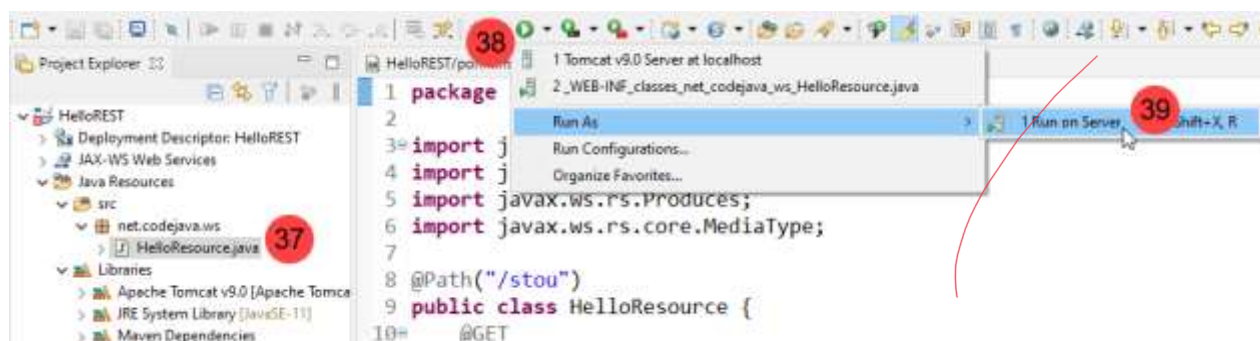
4.1.6. ทำการเปลี่ยนรุ่นของ JRE System Library เป็น JavaSE-11 ตามลำดับดังภาพที่ 4.1.6



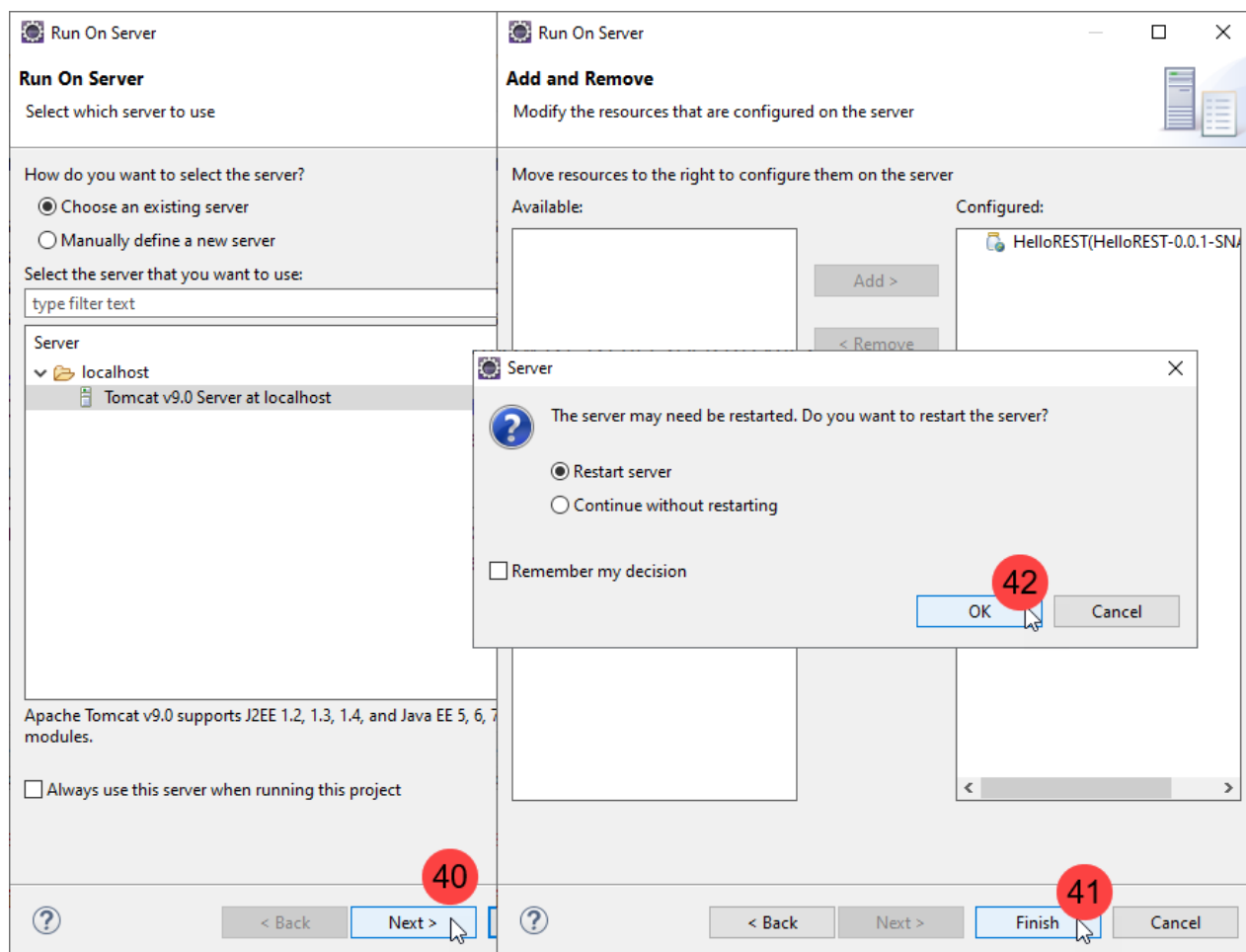


ภาพที่ 4.1.6 การเปลี่ยนรุ่นของ JRE System Library

4.1.7. เลือกไฟล์ HelloResource.java เพื่อทำการ Run as แบบทำงานบน Tomcat Server ดังภาพที่ 4.1.7



ก. การสั่งรันโปรแกรมของไฟล์ HelloResource.java

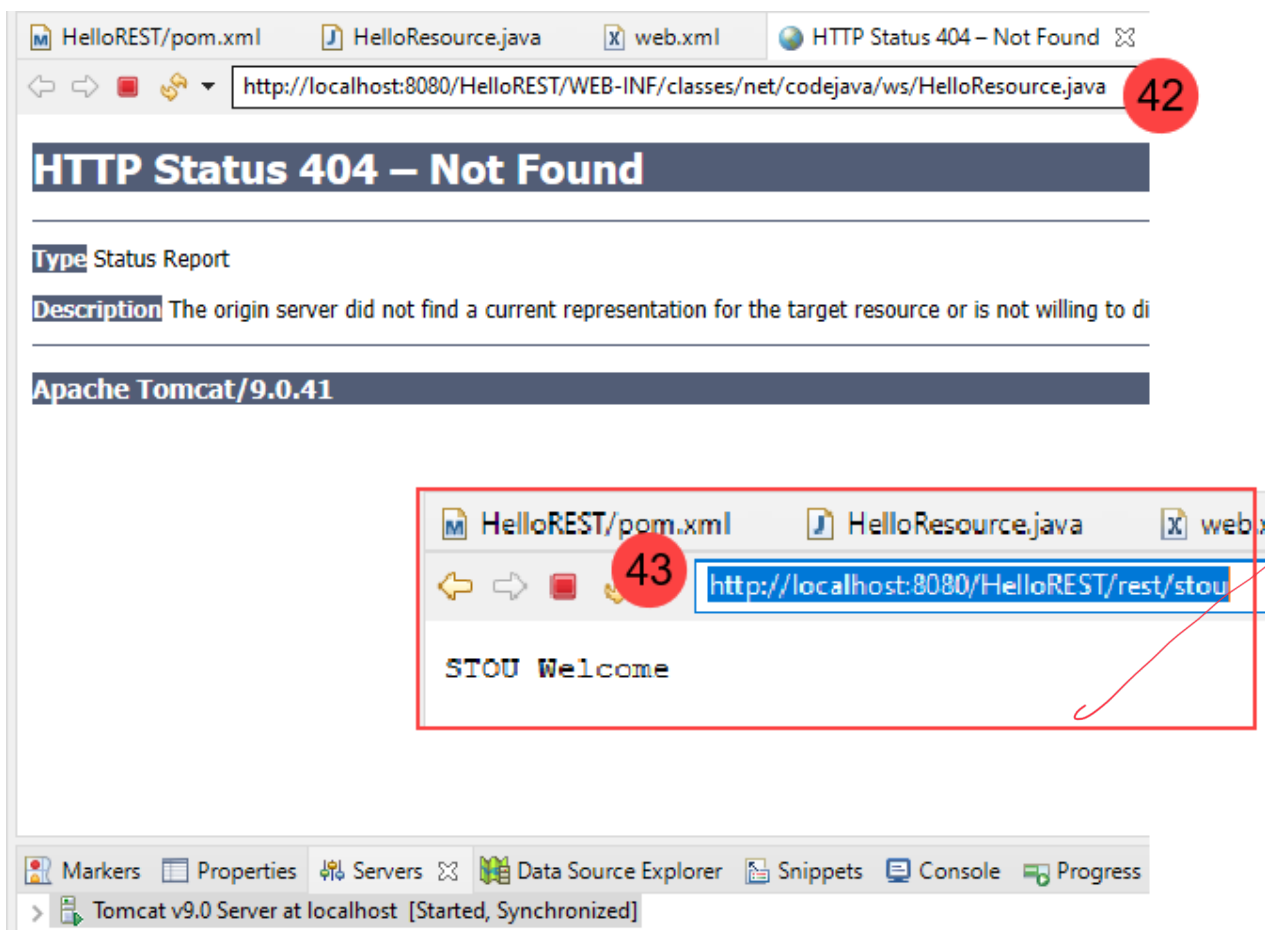


ข. การเลือก option ของการรันโปรแกรม Tomcat Server

ภาพที่ 4.1.7 การเริ่มการทำงานของไฟล์บน Tomcat Server

4.1.8. ผลลัพธ์จากการรัน และการปรับแก้ URL ใหม่ คือ “http://localhost:8080/HelloREST/rest/stou” ตามหมายเลข 43 แล้วกด Enter ได้ผลดังภาพที่ 4.1.8 ซึ่งเป็นวิธีการทดสอบการทำงานของ RESTful Web Service รูปแบบหนึ่ง

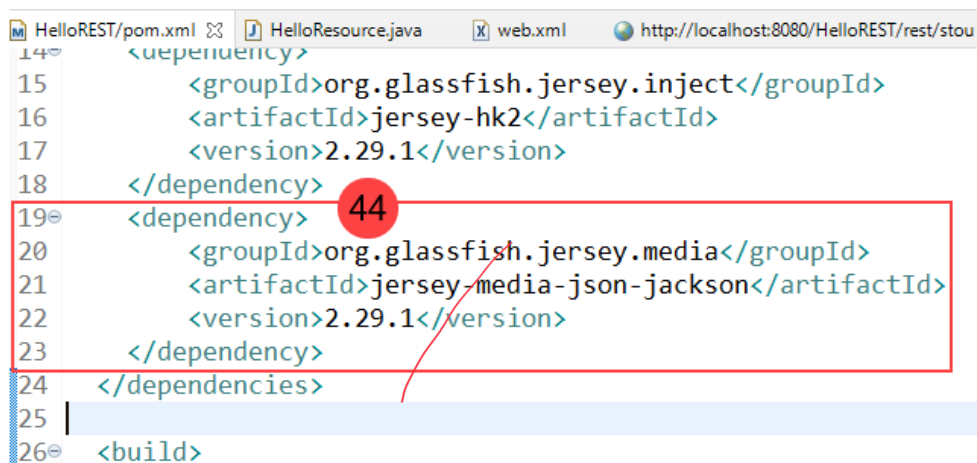




ภาพที่ 4.1.8 ผลจากการรันไฟล์ HelloResource.java

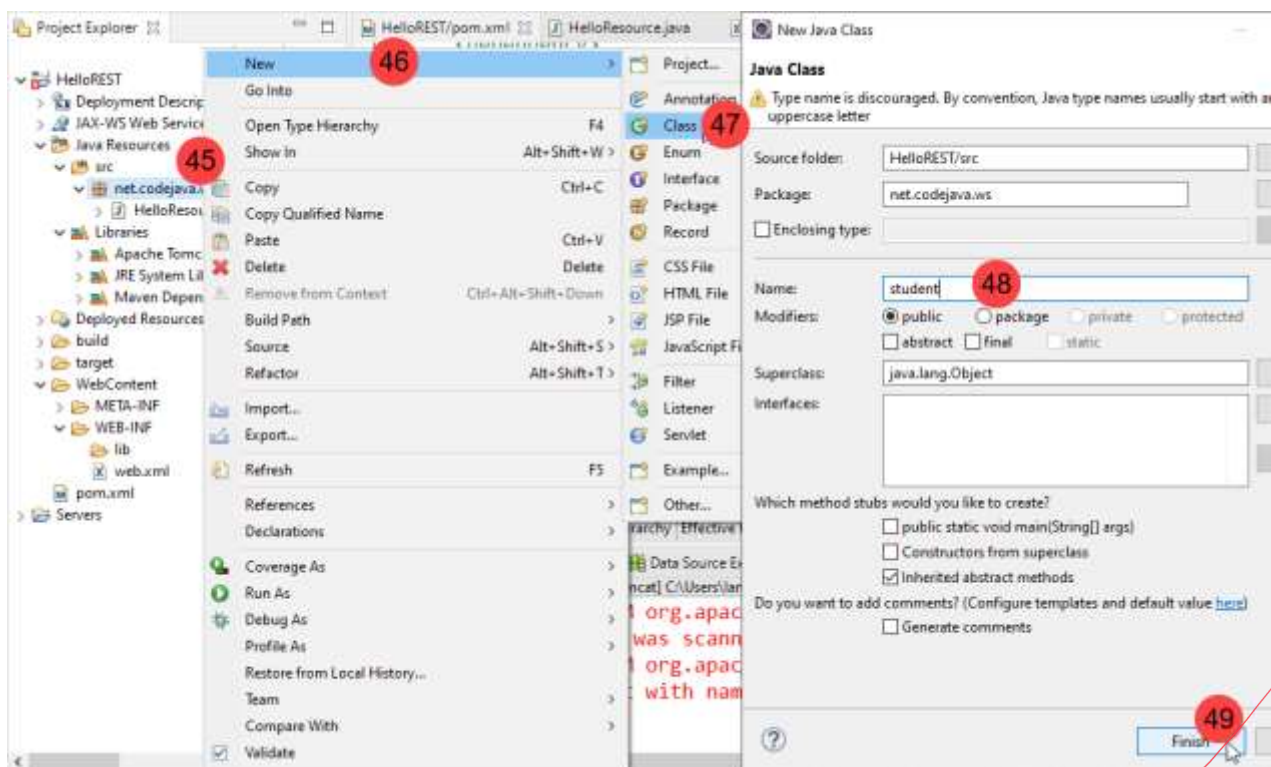
## กิจกรรม 4.2 การทำงานของ RESTful Web Service กับข้อมูลแบบ JSON

4.2.1. เปิดไฟล์ pom.xml และเพิ่มคำสั่งตามหมายเลข 44 ดังภาพที่ 4.2.1 แล้วทำการ save ไฟล์



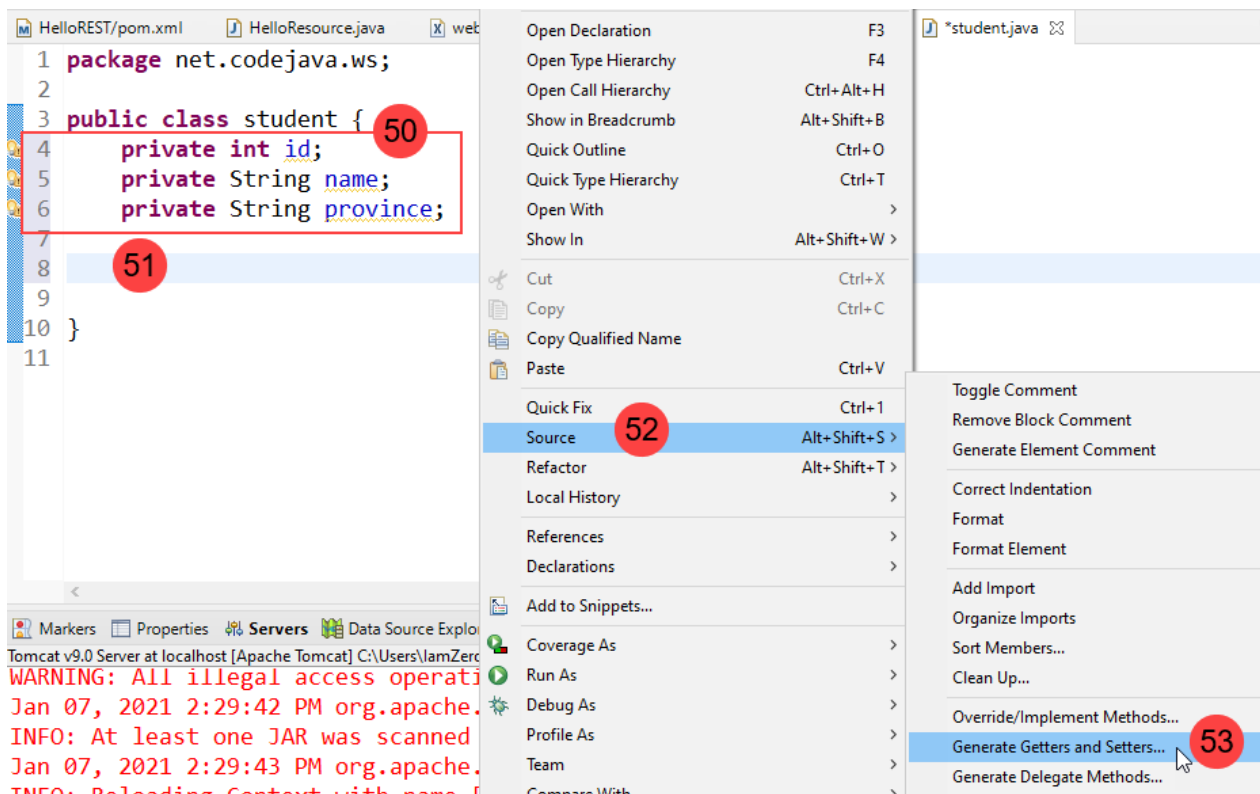
ภาพที่ 4.2.1 การเพิ่มคำสั่งสำหรับการทำงานกับข้อมูลแบบ JSON

4.2.2. สร้าง class ไฟล์ข้อมูลแบบ JSON ชื่อ student.java ดังภาพที่ 4.2.2

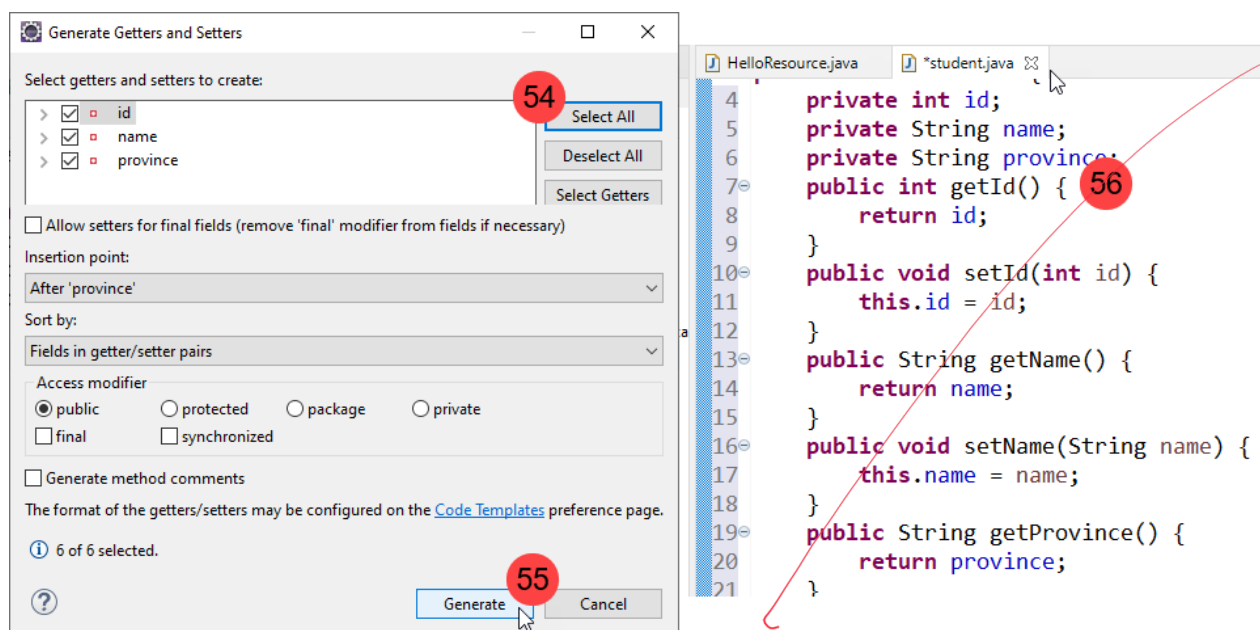


ภาพที่ 4.2.2 การสร้างไฟล์ชื่อ student.java

4.2.3. เปิดไฟล์ student.java แล้วเพิ่มตัวแปรตามในกรอบหมายเลข 50 และในบรรทัดถัดมา คลิกขวานบนเมาส์ (หมายเลข 51) และทำตามลำดับ ดังภาพที่ 4.2.3



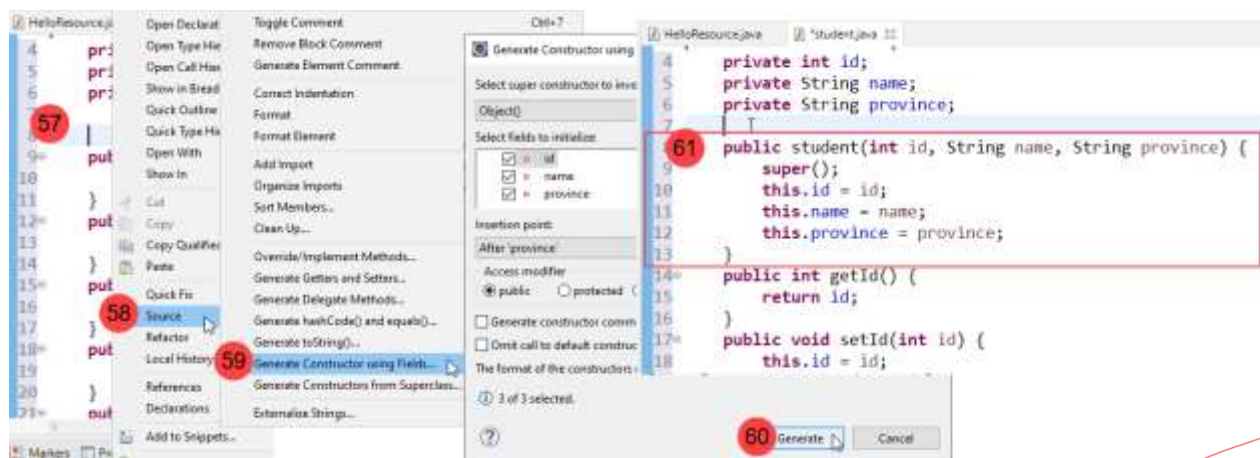
### ก. การเพิ่มตัวแปรและคำสั่งในไฟล์ student.java



ข. ผลของการ Generate Getters and Setters ของตัวแปรในหมายเลข 50

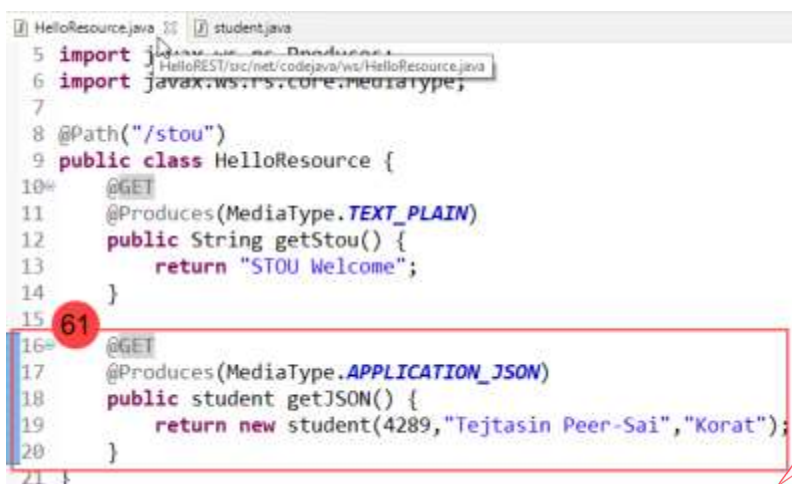
ภาพที่ 4.2.3 การเพิ่มคำสั่งในไฟล์ student.java

4.2.4. แทรกคำสั่งเพิ่มเติมในไฟล์ student.java ด้วยการคลิกขวานเมาส์ (หมายเลข 57) และทำตามลำดับ ดังภาพที่ 4.2.4



ภาพที่ 4.2.4 การแทรกคำสั่งเพิ่มเติมในไฟล์ student

4.2.5 เปิดไฟล์ HelloResource.java เพื่อแทรกคำสั่งสำหรับการจัดการข้อมูลแบบ JSON ตามกรอบหมายเลข 61 ดังภาพที่ 4.2.5



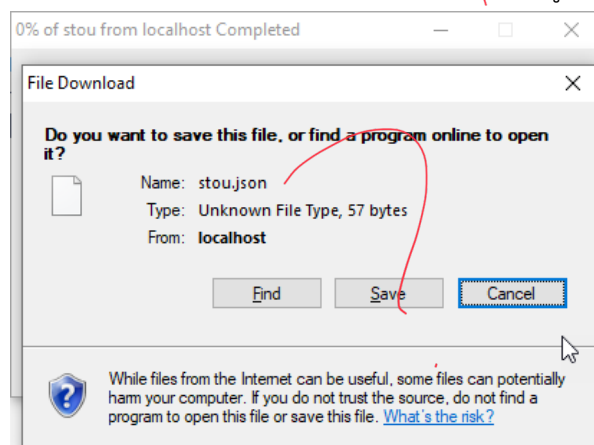
```

1  HelloResource.java
2  student.java
3
4  import java.net.DatagramSocket;
5  import java.net.Socket;
6  import javax.ws.rs.core.MediaType;
7
8  @Path("/stou")
9  public class HelloResource {
10     @GET
11     @Produces(MediaType.TEXT_PLAIN)
12     public String getStou() {
13         return "STOU Welcome";
14     }
15
16     @GET
17     @Produces(MediaType.APPLICATION_JSON)
18     public student getJSON() {
19         return new student(4289, "Tejtasin Peer-Sai", "Korat");
20     }
21 }

```

ภาพที่ 4.2.5 การแทรกคำสั่งในไฟล์ HelloResource.java

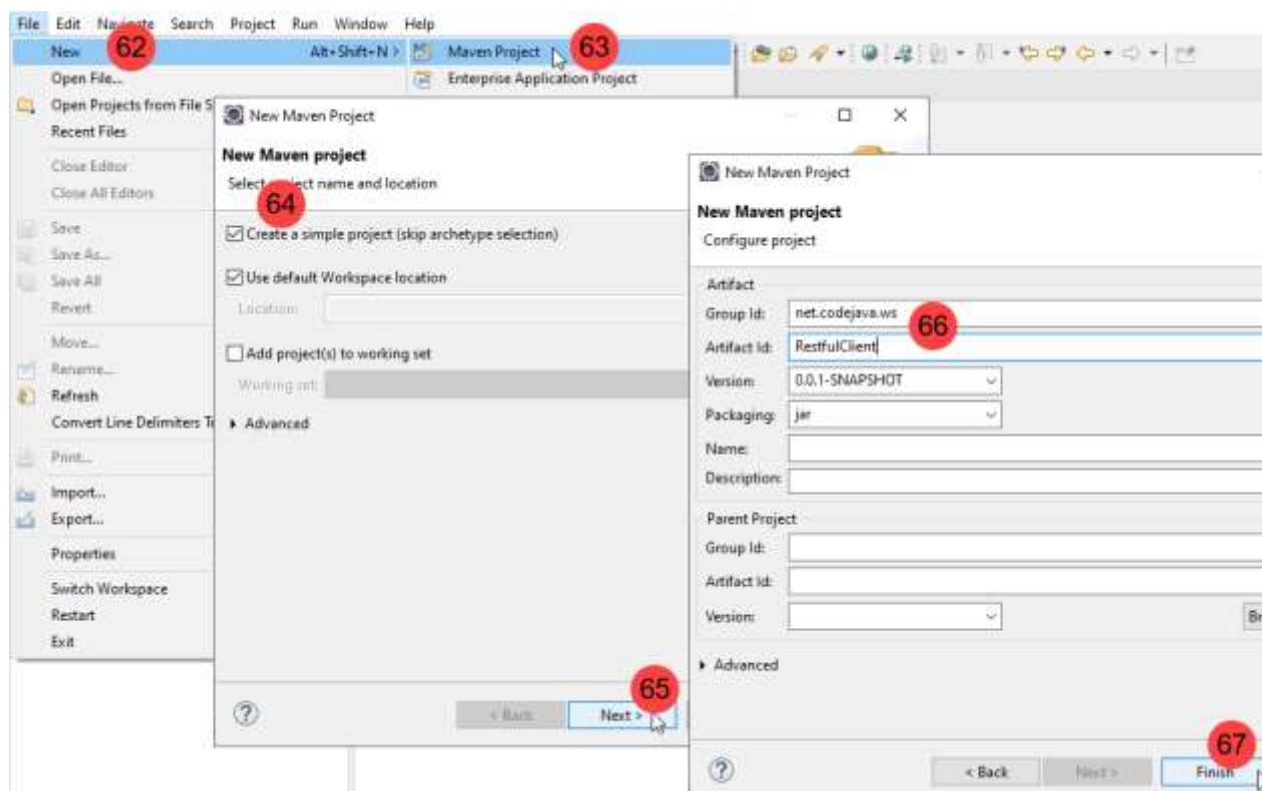
4.2.6. ทดสอบโปรแกรมด้วยการรันไฟล์ HelloResource.java และทำการป้อน address ของ URL ใหม่ด้วย “http://localhost:8080/HelloREST/rest/stou” และได้ผลลัพธ์ของไฟล์ข้อมูลแบบ JSON ดังภาพที่ 4.2.6



ภาพที่ 4.2.6 ผลของการรันกับรูปแบบข้อมูลแบบ JSON

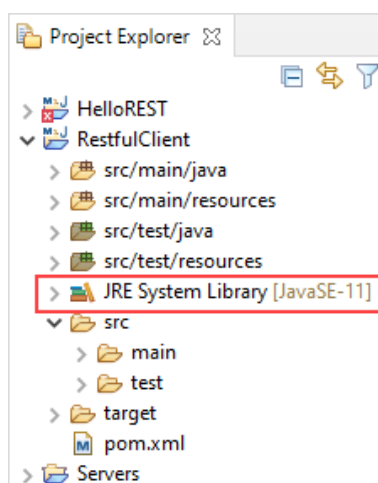
### กิจกรรม 4.3 การเรียกใช้ RESTful Web Service ทางฝั่งไคลเอนต์

4.3.1. สร้างไฟล์โปรเจกต์ใหม่ชื่อ RestfulClient สำหรับการทำงานทางฝั่ง Client ด้วยเครื่องมือ Maven project ตามลำดับ ดังภาพที่ 4.3.1



ภาพที่ 4.3.1 การสร้างไฟล์โปรเจก RestfulClient

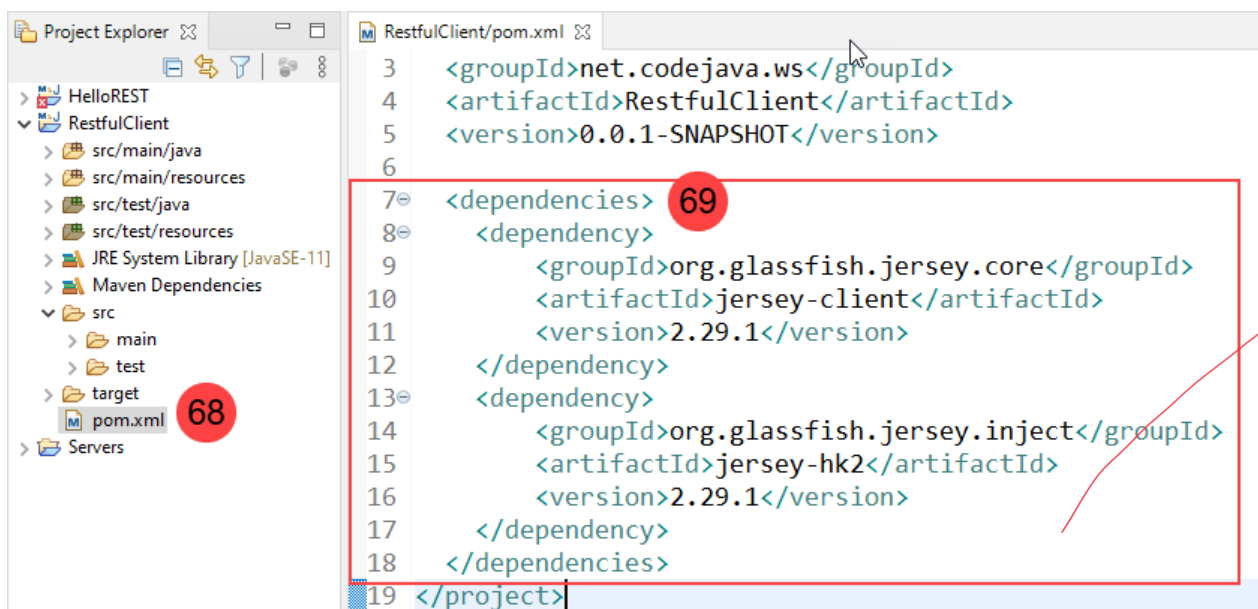
4.3.2. เปลี่ยนรุ่นของ JRE System Library จาก J2SE-1.5 เป็น JavaSE-11 (ขั้นตอนเหมือน หัวข้อ 4.1.6) ได้ผลดังภาพที่ 4.3.2



ภาพที่ 4.3.2 การเปลี่ยนรุ่นของ JRE System Library

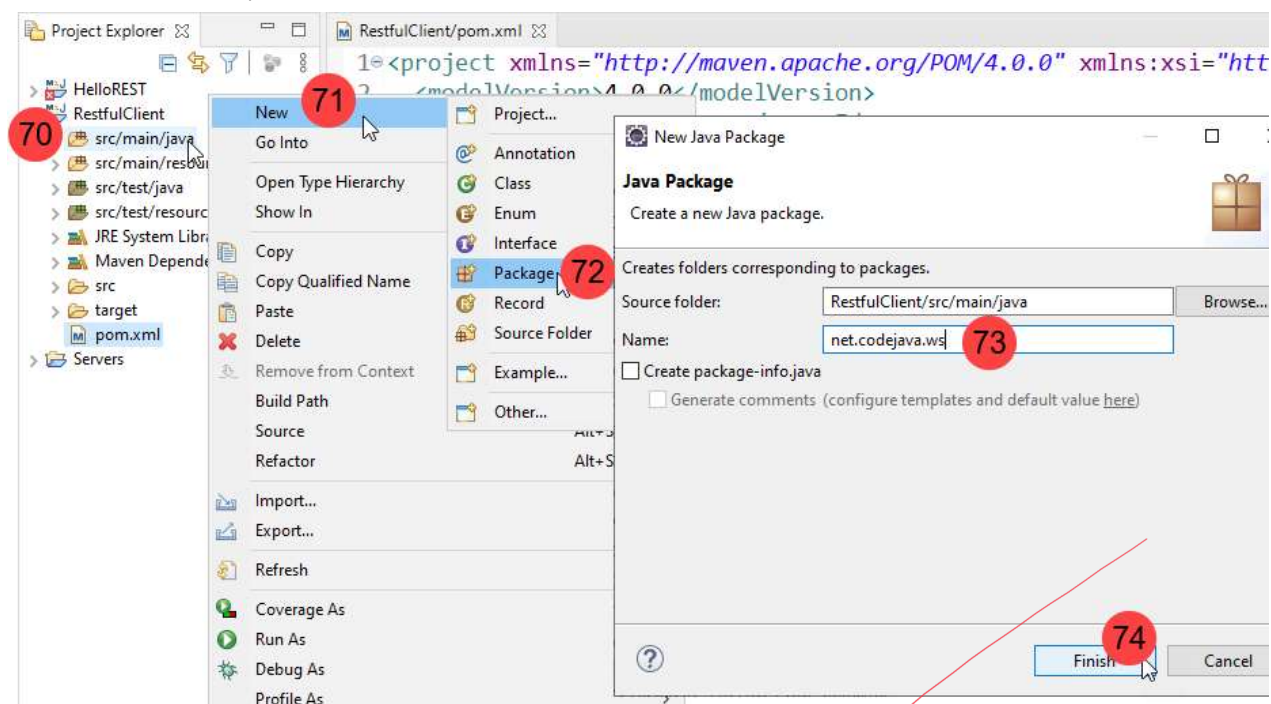
4.3.3. เปิดไฟล์ pom.xml ของโปรเจก RestfulClient เพื่อกำหนดรูปแบบการทำงานแบบ Jersey RESTful ตามลำดับ ดังภาพที่ 4.3.3



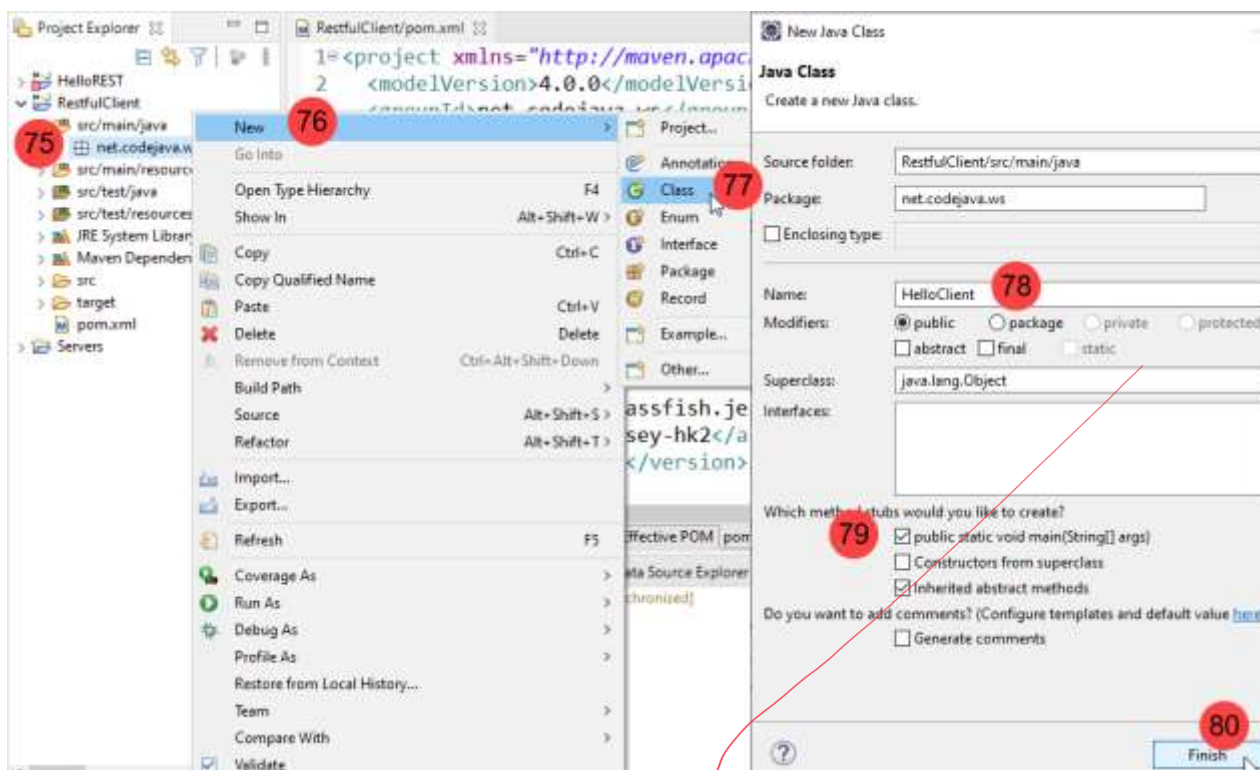


ภาพที่ 4.3.3 การเพิ่มคำสั่งเพื่อกำหนดการทำงานแบบ Jersey ในไฟล์ pom.xml

4.3.4. สร้างไฟล์แบบ package ชื่อ net.codejava.ws และไฟล์แบบคลาส ชื่อ HelloClient.java



ก. การสร้างไฟล์ package ชื่อ net.codejava.ws

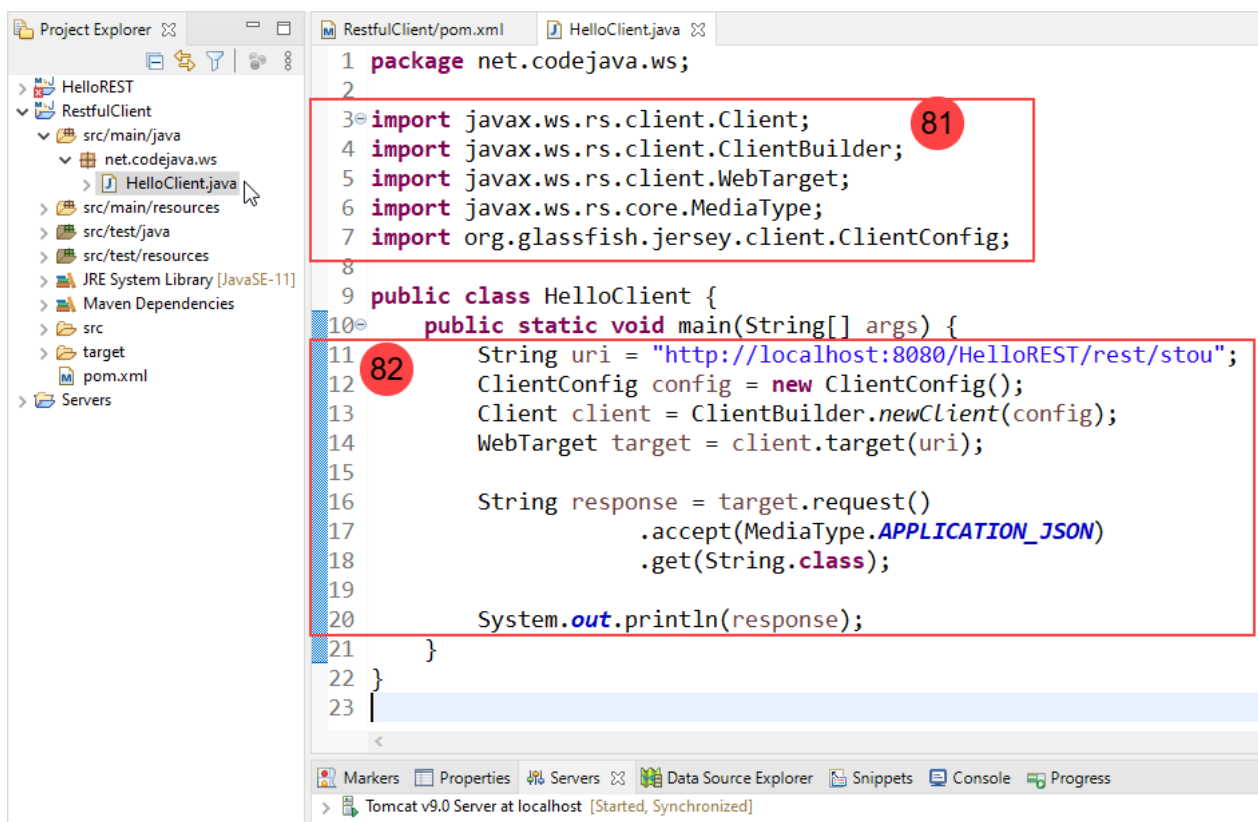


ข. การสร้างไฟล์แบบคลาส ชื่อ HelloClient.java

ภาพที่ 4.3.4 ลำดับการสร้างไฟล์แบบ package และแบบคลาส

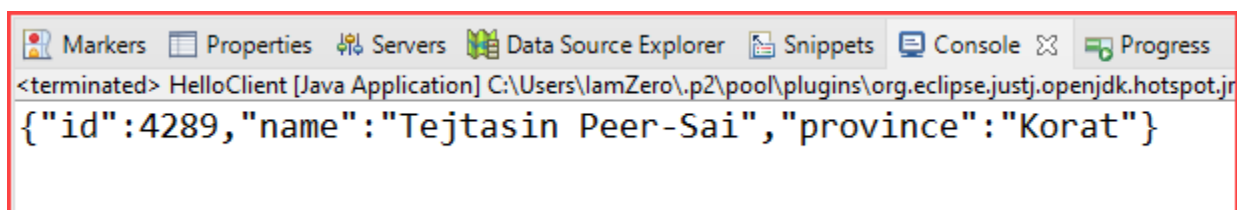
4.3.5. เปิดไฟล์ HelloClient.java เพื่อแทรกคำสั่ง ดังภาพที่ 4.3.5





ภาพที่ 4.3.5 การแทรกคำสั่งในไฟล์ HelloClient.java

4.3.6. ทำการรันไฟล์ HelloClient.java ทางฝั่ง Client และได้ผลลัพธ์ดังภาพที่ 4.3.6



ภาพที่ 4.3.6 ผลการรันไฟล์โปรเจกต์ทางฝั่ง Client