บทที่ 8

Maven Project

ธีระยุทธ ทองเครือ

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

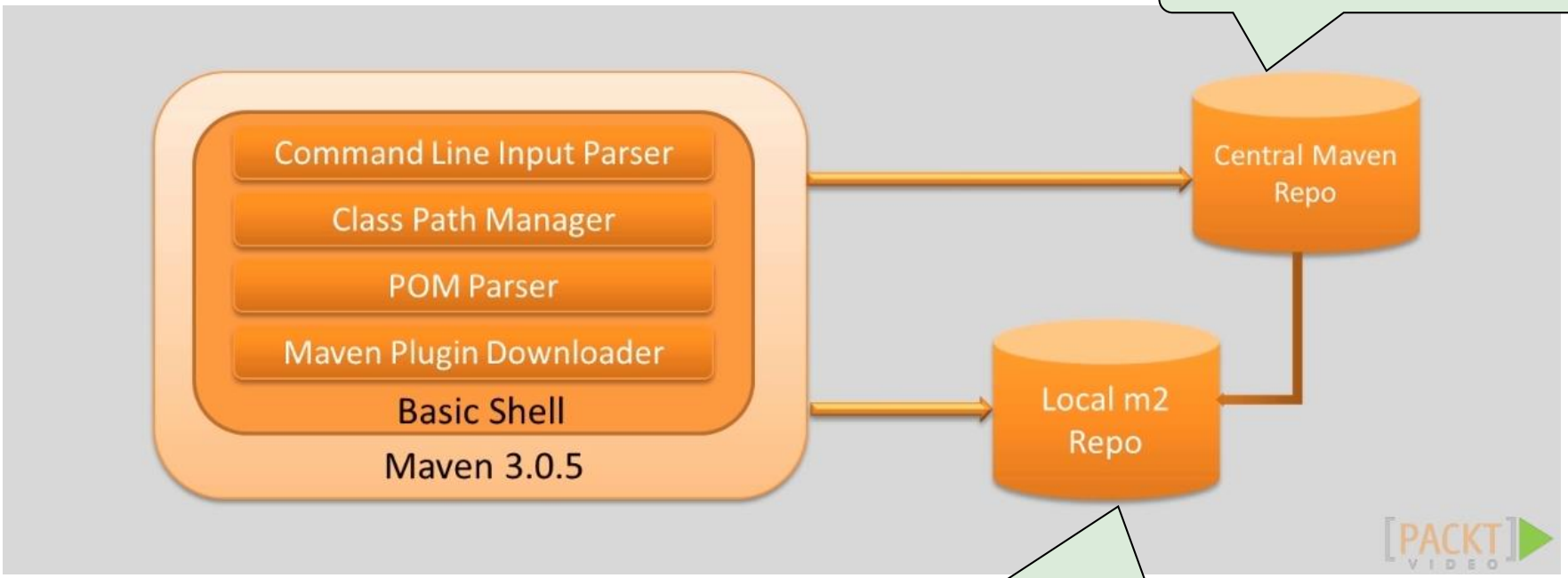
มหาวิทยาลัยขอนแก่น

Maven

- Maven คือ เครื่องมือสำหรับใช้ในการบริหารจัดการ Project โดยมีเป้าหมายคือ ช่วยในการ build project แบบอัตโนมัติ
- Maven ใช้ Project Object Model (POM) ซึ่งเป็นไฟล์ข้อมูลเกี่ยวกับการสร้าง Project ในรูปแบบ XML นักพัฒนาสามารถกำหนดข้อมูล เช่น
  - รายการไลบรารีที่อ้างอิง (Dependency list) ช่วยให้ไม่ต้องไปโหลดไฟล์ Library เอง
  - ส่วนการทำ Unit Test
  - การกำหนด environment ต่างๆ
- Maven มีแม่แบบ (archetype) ในการช่วยสร้างแอปพลิเคชันชนิดต่างๆ
- ใน Eclipse มี Maven ติดมาแล้ว สามารถสร้าง Project ในรูปแบบ Maven ได้เลย

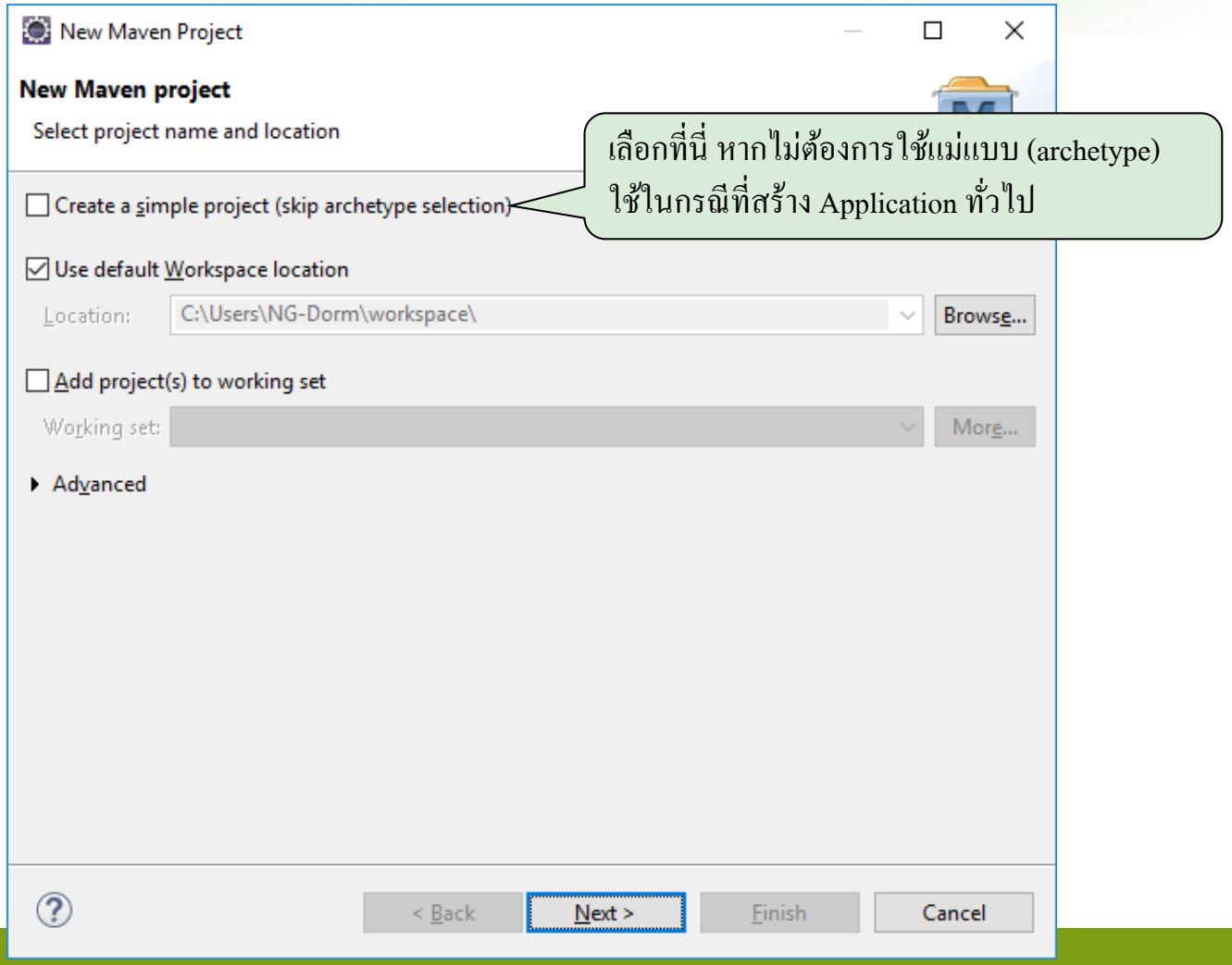
Maven Architecture

- Central Maven Repository คือ Server ที่ใช้เก็บ Library ทั้งหมด
- Library ที่ถูกอ้างอิงมาใช้แล้ว จะอยู่บน Local m2 Repository เพื่อใช้กับ Project อื่นๆไม่ต้องโหลดใหม่จาก Central Maven Repository ทุกครั้ง

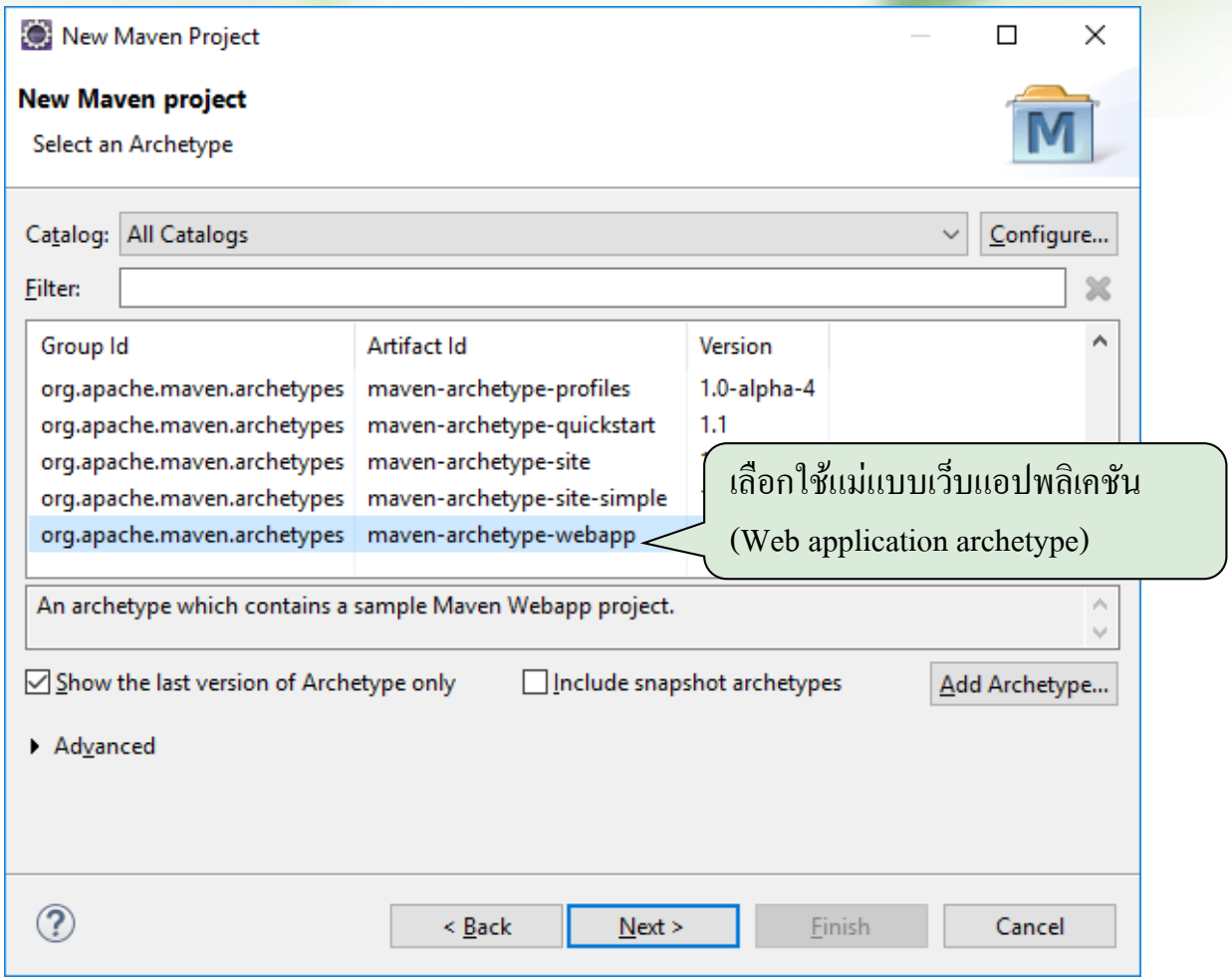


การสร้าง Maven Web Project

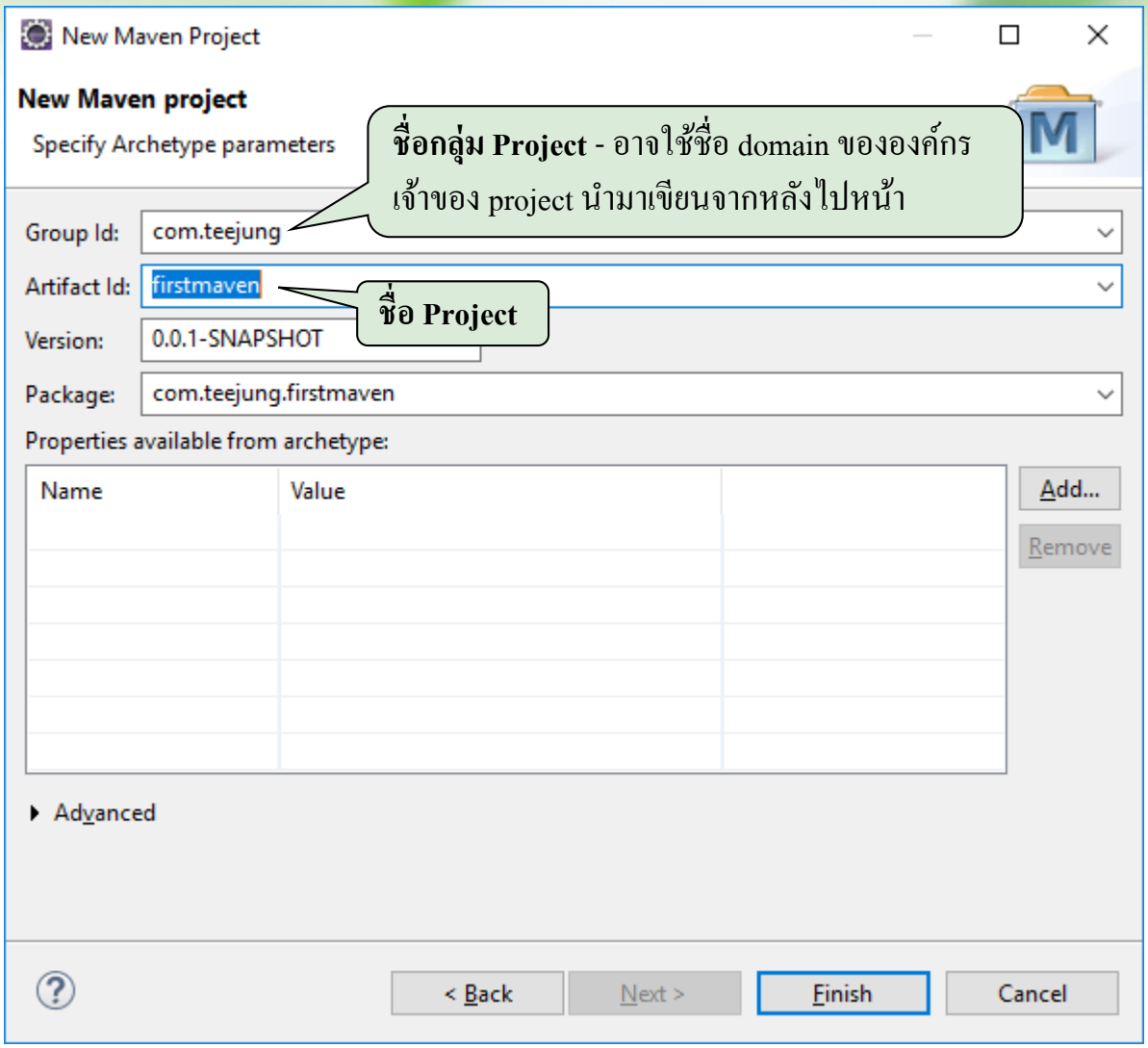
❖ เลือกเมนู File ▶ New ▶ Maven Project



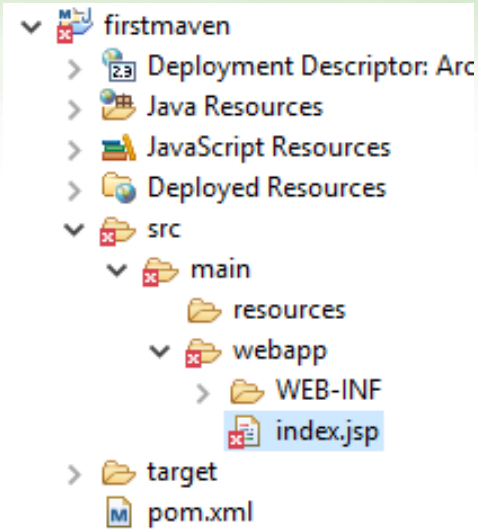
การเลือกแม่แบบ Project



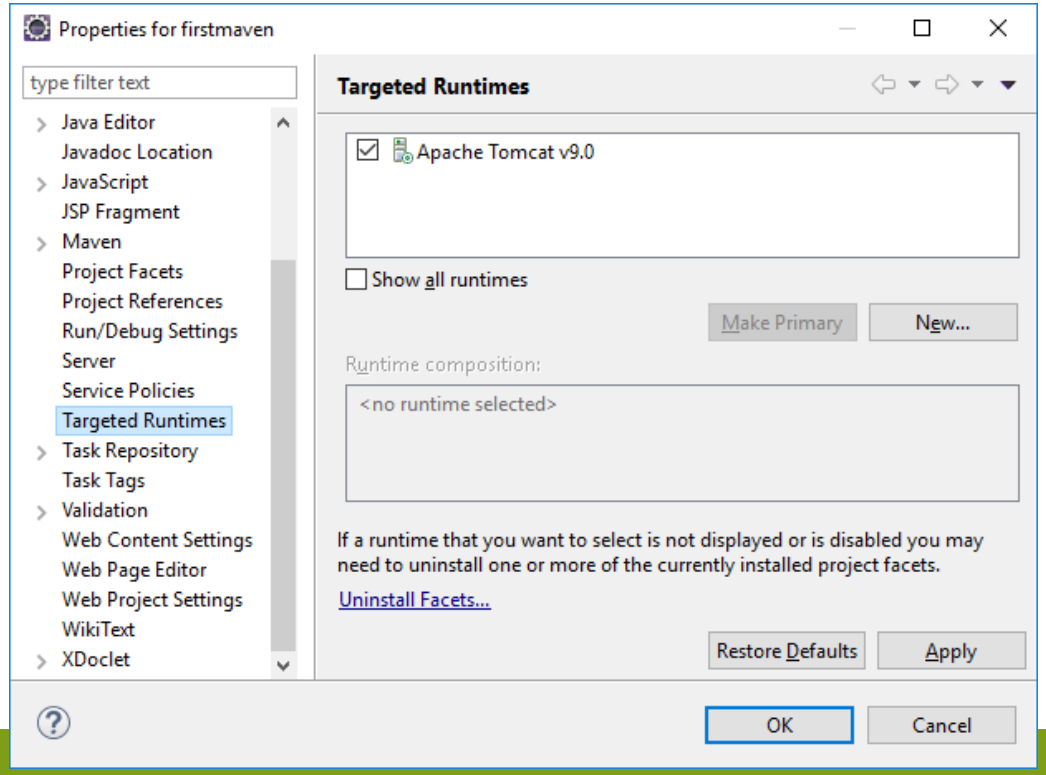
กำหนดชื่อกลุ่ม และชื่อ Project



กำหนด Server สำหรับ Project

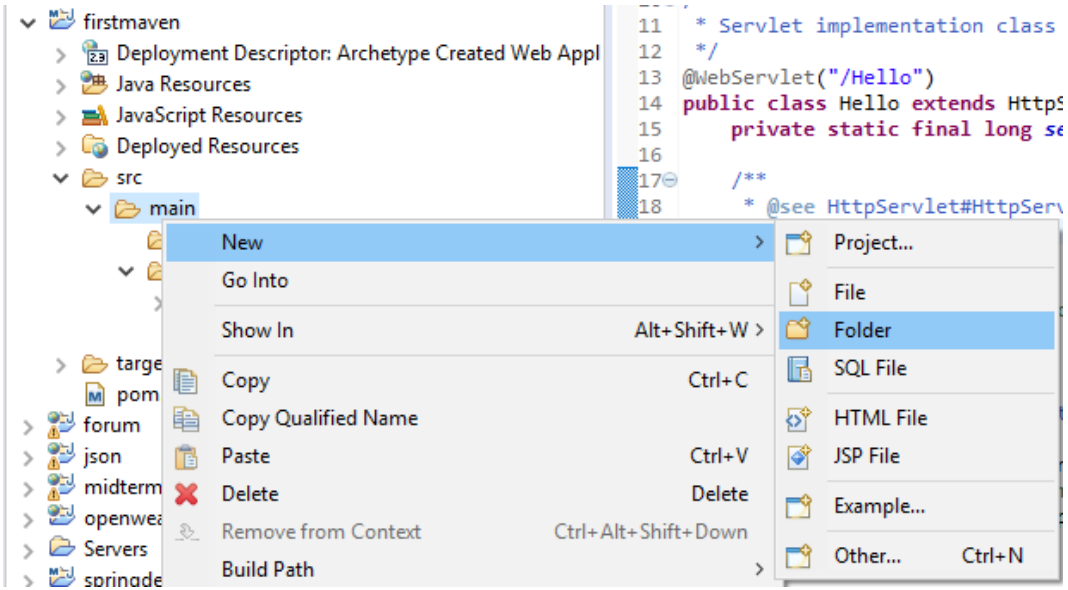


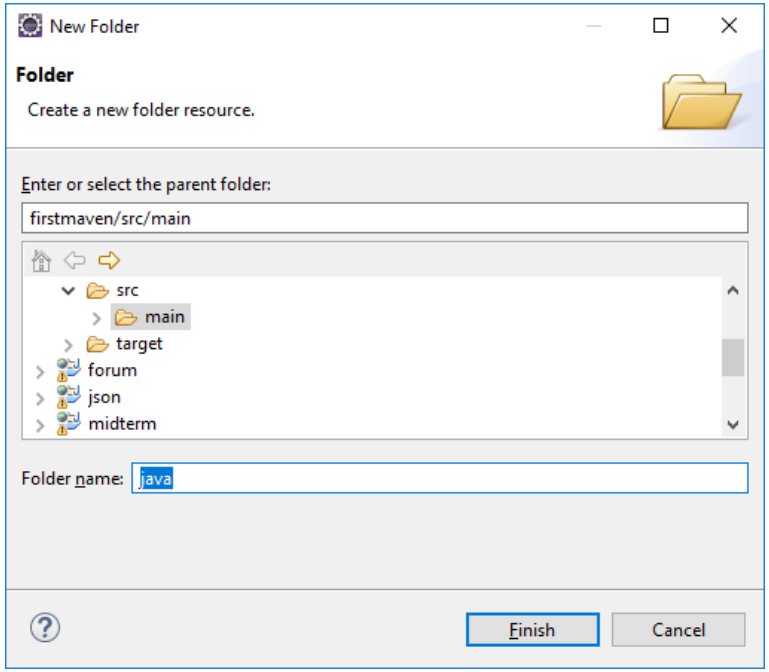
- Maven Project ที่สร้างขึ้นจะยังไม่รู้จักคลาส HttpServlet แก้ไขโดยคลิกขวาที่ชื่อ Project เลือก *Properties*
- เลือก *Targeted Runtimes*
- เลือก Web Server ที่เพิ่มไว้แล้ว เช่น Tomcat



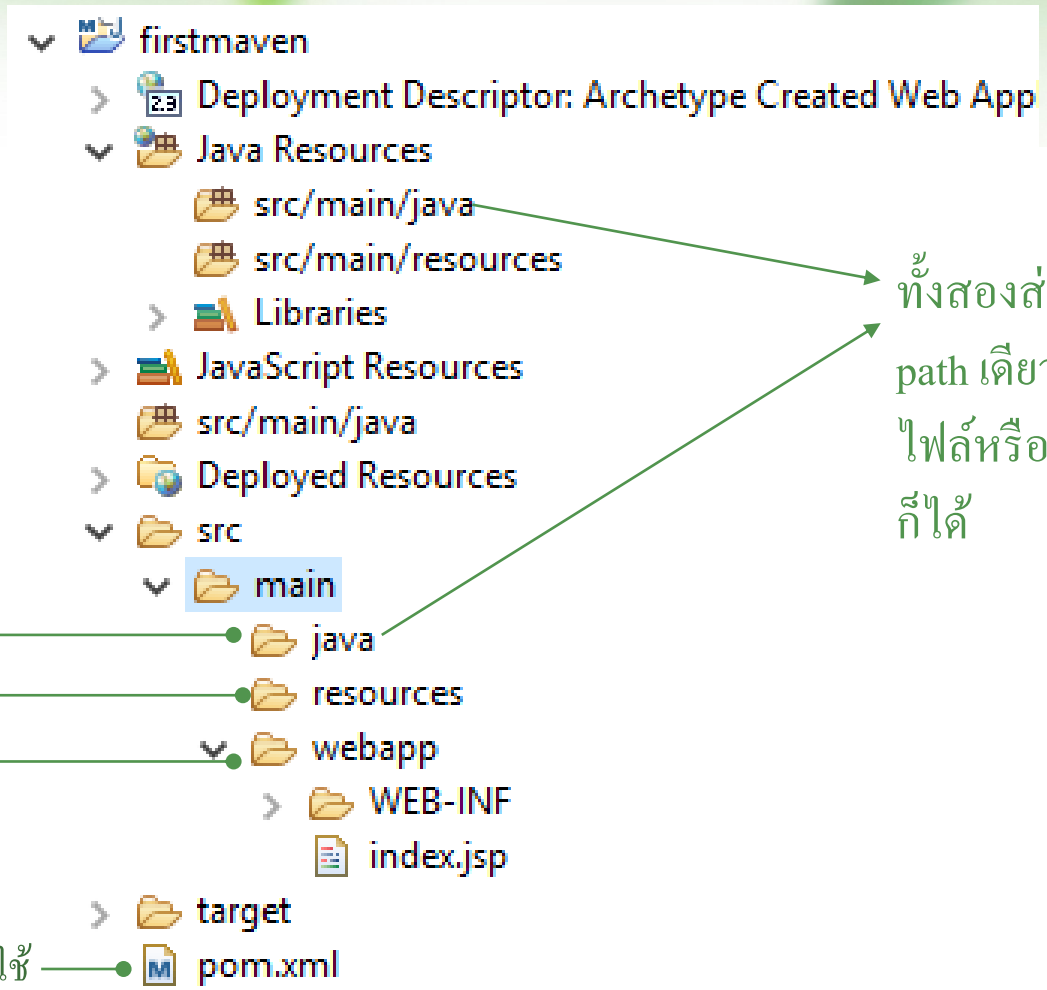
สร้างโฟลเดอร์เก็บโค้ด .java

- แม่แบบเว็บแอปพลิเคชันจะไม่สร้างโฟลเดอร์สำหรับเก็บไฟล์ .java มาให้ ให้สร้างโฟลเดอร์ใหม่เองชื่อ java ภายใต้โฟลเดอร์ main
- คลิกขวาที่โฟลเดอร์ main เลือก New > Folder
- ใส่ชื่อโฟลเดอร์ "java"





โครงสร้างโฟลเดอร์ Maven Project



ทดสอบการทำงาน JSP

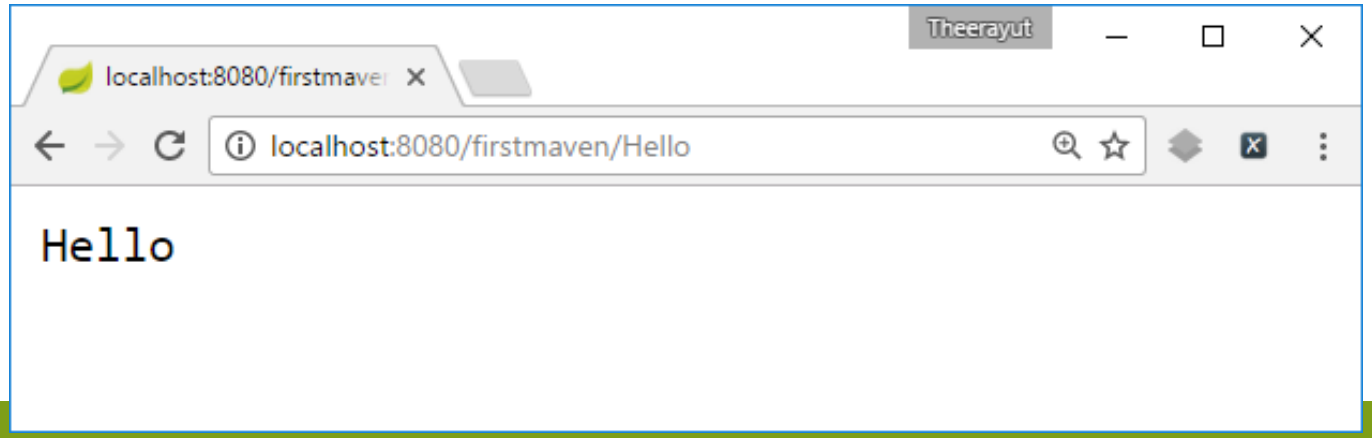
- เพิ่ม Maven Project ที่สร้างขึ้นเข้าไปใน Server
- Start Server และทดสอบรันไฟล์ index.jsp ที่แม่แบบสร้างไว้ให้แล้ว

ทดสอบการทำงาน Servlet

- สร้าง Servlet ในโฟลเดอร์ main/java
- restart server และทดสอบ

!!!! ใน Maven Project จะต้องใส่ path ของ Servlet เอง

```
@WebServlet("/Hello")
public class Hello extends HttpServlet {

  protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        response.getWriter().println("Hello");
  }
}
```



การเพิ่มไลบรารีสำหรับ Maven Project

- นักพัฒนาสามารถนำไลบรารีมาใช้ โดยหาได้จาก 2 แหล่ง
  - เว็บไซต์ผู้สร้าง Library โดยตรง
  - Central Maven Repository

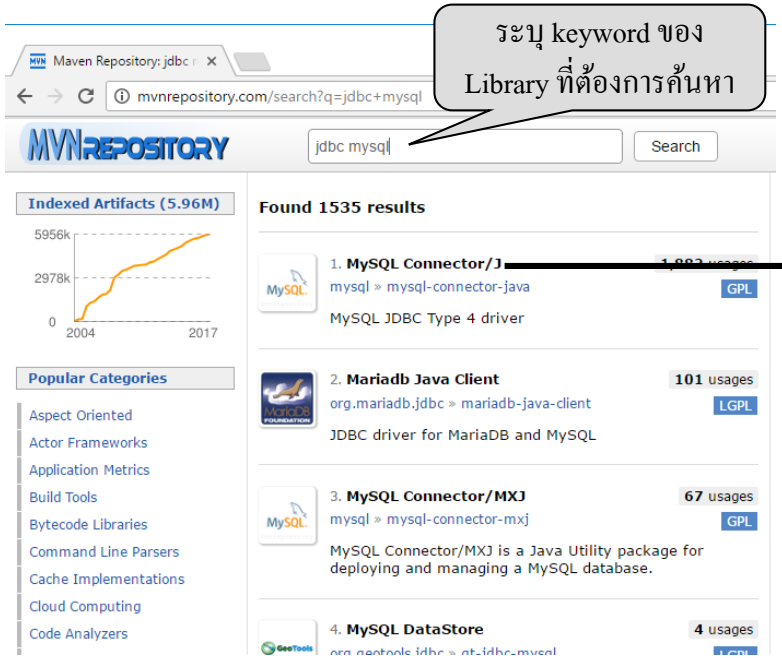
- การเพิ่ม Library คือ การนำชื่อ และรุ่นของ Library มาเพิ่มข้อมูลลงในไฟล์ pom.xml โดยวางภายใต้แท็ก <dependencies></dependencies> หลังจากบันทึกไฟล์ pom.xml แล้ว Maven จะติดต่อไปยัง Maven Repository เพื่อโหลดไฟล์ที่เกี่ยวข้องมาเก็บไว้ใน Project อัตโนมัติ ดังนั้นหากไม่เคยใช้ Library นั้นในเครื่องที่พัฒนาใช้ จะต้องต่อ internet ด้วย

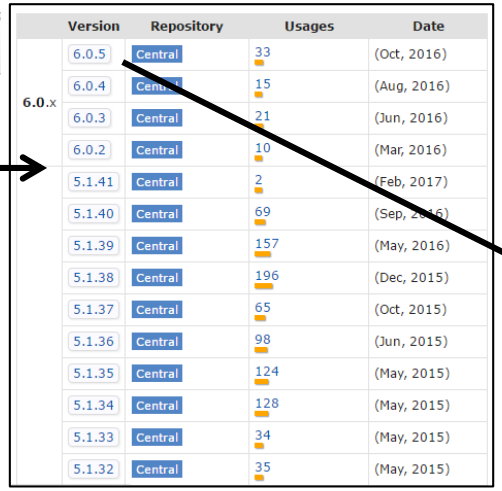
- การโหลด Library จาก Maven Repository จะทำเฉพาะครั้งแรกและครั้งเดียวเท่านั้น หากเป็น Library ที่เคยเรียกใช้แล้วจะนำไฟล์จาก cache มาเพิ่มให้โดยไม่มีการดาวน์โหลดใหม่

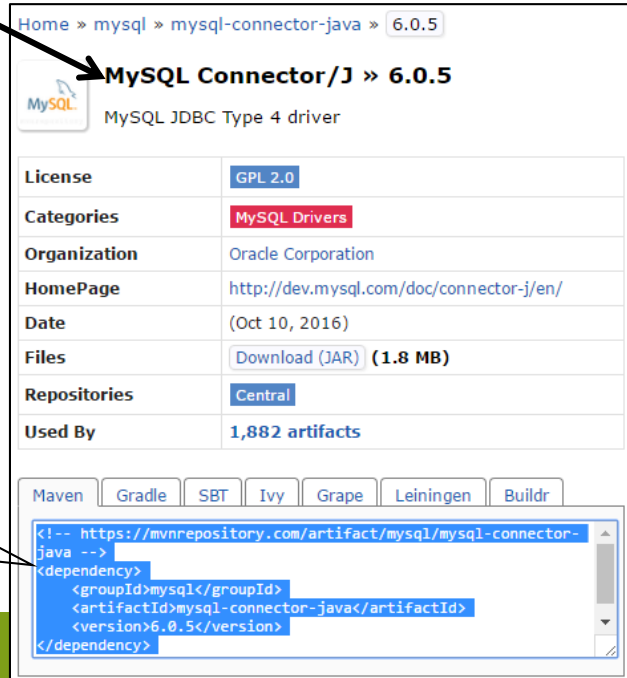
การค้นหาชื่อ Library จาก Maven Repository

❖ นักพัฒนาสามารถค้นหาชื่อ Library เพื่อนำมากำหนดใช้ใน Project ได้จาก

http://search.maven.org







ชื่อของ Library ที่จะ copy ไปวางไว้ในไฟล์ pom.xml

ตัวอย่างไฟล์ pom.xml

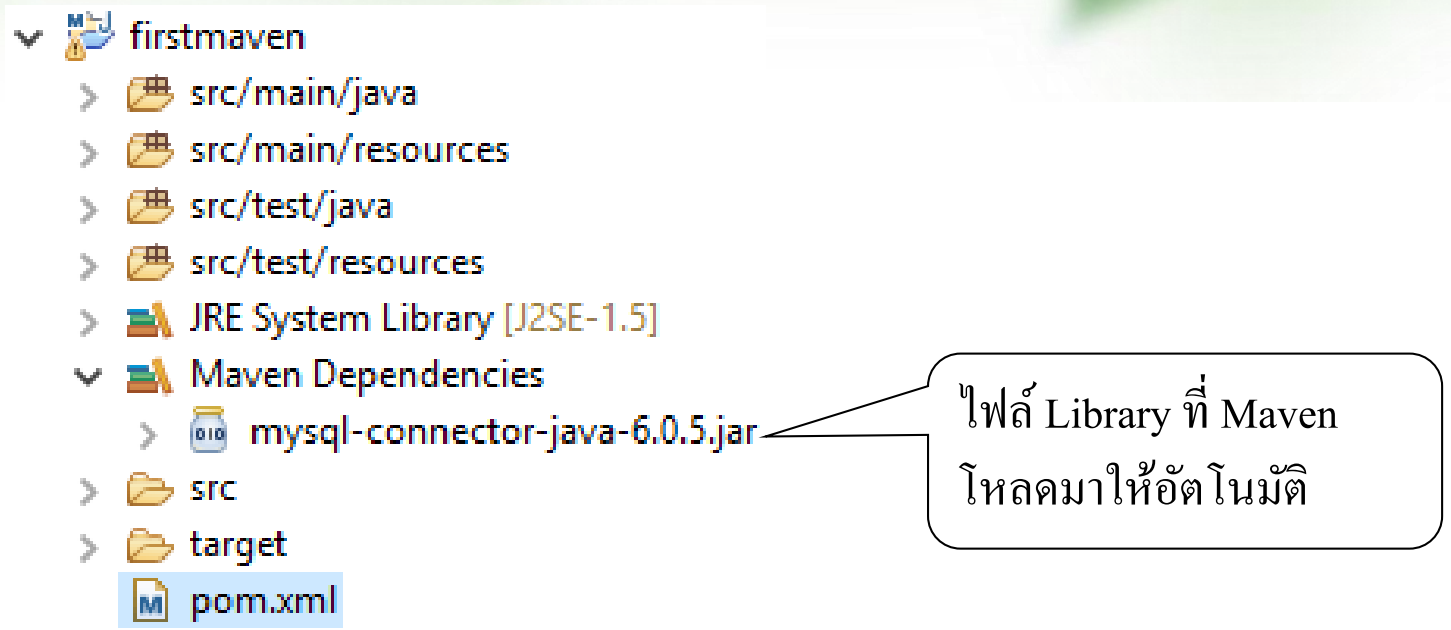
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.teejung</groupId>
  <artifactId>firstmaven</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>6.0.5</version>
    </dependency>

  </dependencies>

</project>
```

ข้อมูล Project

แท็กใช้ครอบชื่อ Library ทั้งหมดของ Project

แท็ก dependency ใช้ครอบชื่อ 1 Library

ตัวอย่างไลบรารีที่ถูกโหลดแบบอัตโนมัติ



หากมีการลบแท็ก **<dependency>** ของไลบรารีใดออก
ไฟล์ .jar ก็จะถูกดึงออกจาก Project แบบอัตโนมัติ
แต่ไฟล์ .jar นั้นยังเก็บพัก (cache) ไว้ในโปรแกรม Eclipse อยู่
จะถูกนำกลับมาใช้ได้เมื่อมีการเพิ่มไลบรารีอีกครั้งใน Project ใดๆก็ได้

การใช้ Library ที่โหลดเสร็จแล้ว

- เมื่อได้ Library ที่ต้องการแล้ว ไฟล์ต่างๆจะอยู่ใน Class Path อัตโนมัติ

- นักพัฒนาสามารถ import ในโค้ดโปรแกรม อ้างอิงคลาสจาก Library ที่ Maven โหลดมาใช้ได้ทันที

ทดสอบใช้ Library ติดต่อฐานข้อมูลกับ Servlet

```
@WebServlet("/Hello")
public class Hello extends HttpServlet {
  protected void doGet(HttpServletRequest request, …) {
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String dbURL = "jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
        Connection con = DriverManager.getConnection(dbURL, "root", "");
        Statement statement = con.createStatement();
        ResultSet resultSet = statement.executeQuery("select * from product");
        while (resultSet.next()) {
            int pid = resultSet.getInt("pid");
            String pname = resultSet.getString("pname");
            String pdetail = resultSet.getString("pdetail");
            int price = resultSet.getInt("price");
            out.println(pid + "," + pname + "," + pdetail + "," + price + "<br>");
        }
    } catch (ClassNotFoundException e) {
        System.err.println("Error loading driver: " + e);
    } catch (SQLException e) {
        System.err.println("Error database connection: " + e);
    }
  }
}
```