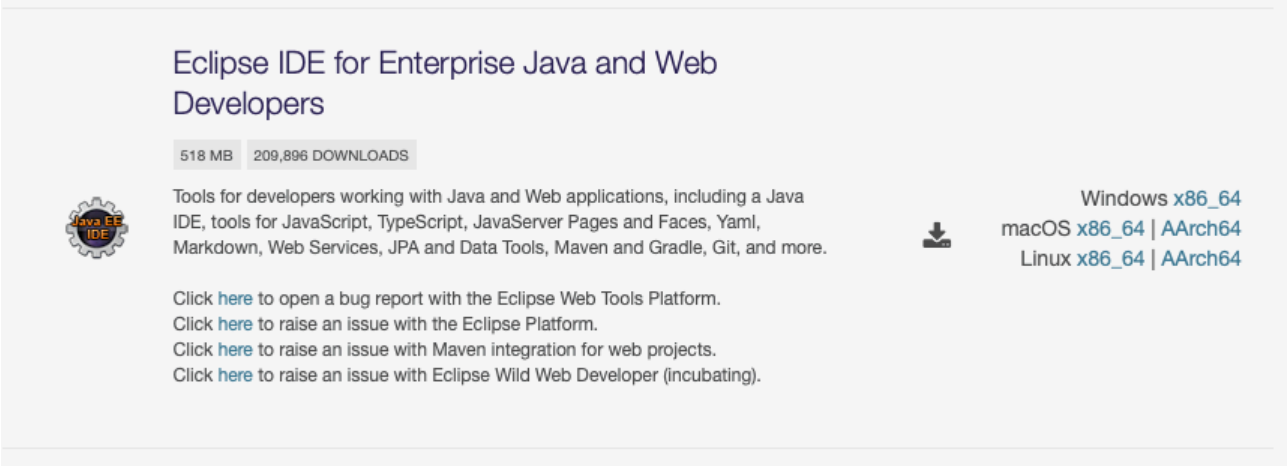
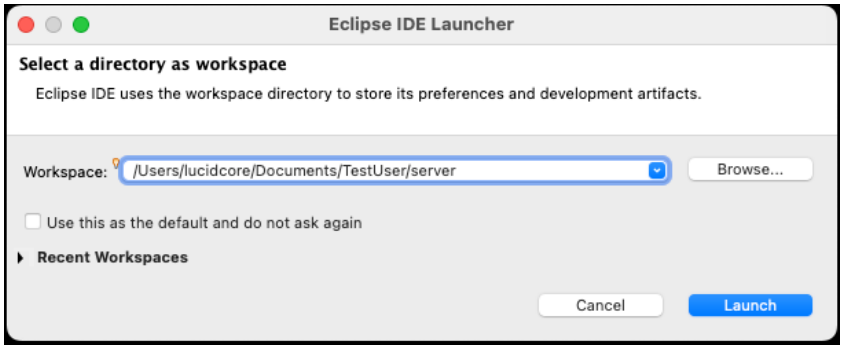
Week 1

목표 : 로컬에 사용가능한 서버를 설정한다.

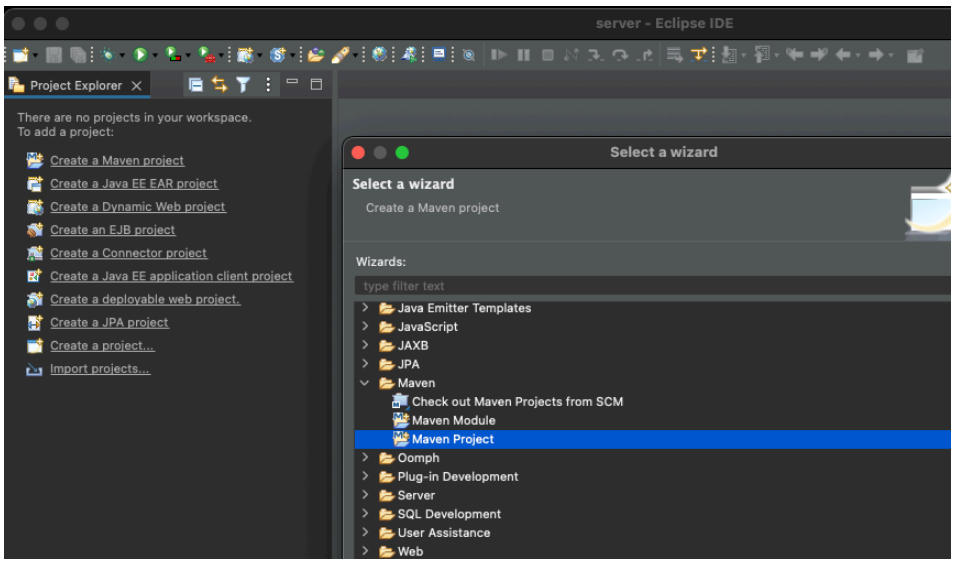
1. eclipse를 다운로드 받아서 설치한다.
    1. https://www.eclipse.org/downloads/packages/
    2. 자신이 사용하고 있는 컴퓨터의 OS와 CPU종류에 맞는 Eclipse IDE for Enterprise Java and Web Developers를 다운받고 설치한다.



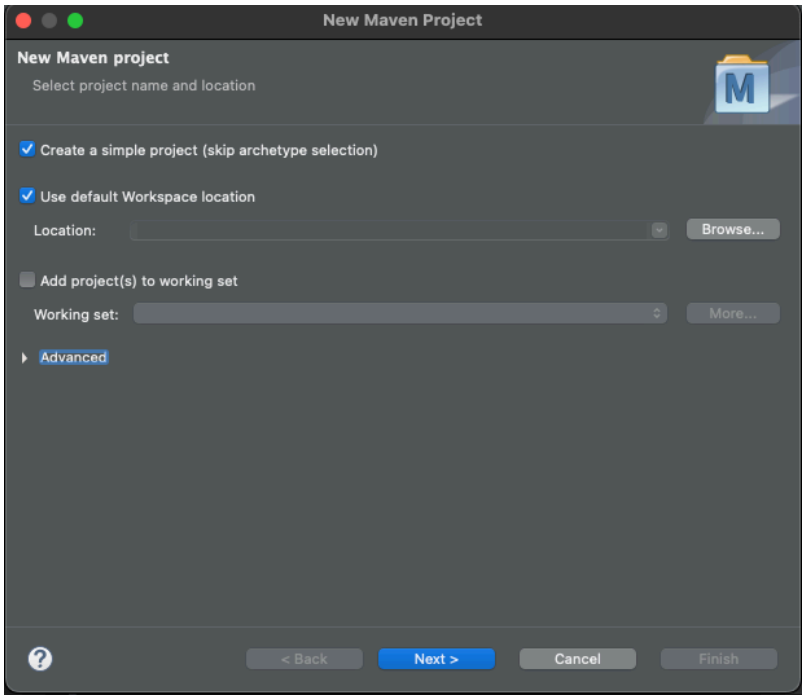
2. eclipse를 열고 적절한 위치에 WorkSpace를 설정한다.

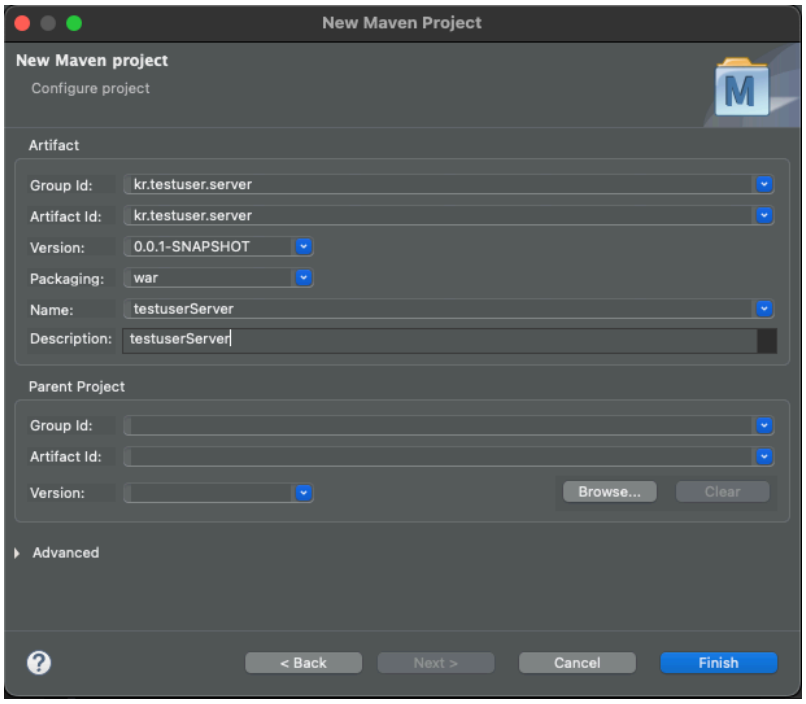


3. Maven 프로젝트를 생성한다.
    1. File -> New -> Other -> Maven -> Maven Project를 선택한다. File -> New 에서 Maven Project가 보이면 바로 선택해도 됨.



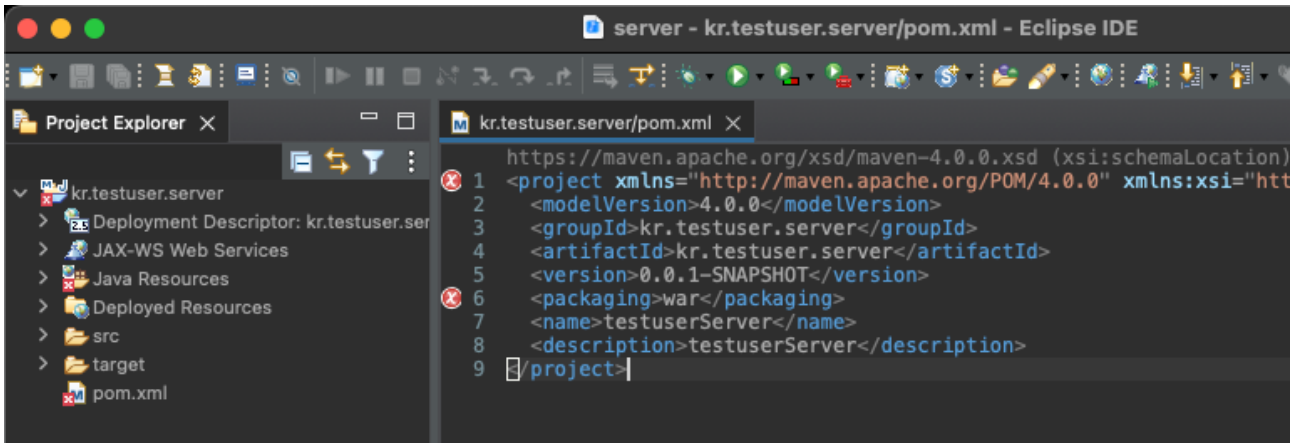
4. Maven프로젝트 기본을 설정한다.





적절하게 내용을 채우고 finish를 눌러서 프로젝트를 생성한다.(Packaging을 war로 변경해야 한다.)

5. 적절하게 finish를 눌러서 프로젝트를 생성한다.



6. pom.xml에 dependency라이브러리를 설정한다.
<description>아래에 아래의 코드를 추가한다. (모두 필요하다고 할수는 없으나 빼는 것도 일이기 때문에 대충 그냥 다 때려 넣습니다.)

```
<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 https://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.testuser.server</groupId>
  <artifactId>kr.testuser.server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>testuserServer</name>
  <description>testuserServer</description>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-web-api</artifactId>
      <version>8.0.1</version>
      <scope>provided</scope>
     </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.containers</groupId>
      <artifactId>jersey-container-servlet-core</artifactId>
      <version>2.32</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.media</groupId>
      <artifactId>jersey-media-json-jackson</artifactId>
      <version>2.32</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.media</groupId>
      <artifactId>jersey-media-multipart</artifactId>
      <version>2.32</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.jersey.inject</groupId>
      <artifactId>jersey-hk2</artifactId>
      <version>2.32</version>
    </dependency>
```

```
    <dependency>
    <groupId>org.apache.logging.log4j</groupId>
      <artifactId>log4j-core</artifactId>
      <version>2.17.1</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>1.7.32</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.7.32</version>
    </dependency>
  </dependencies>

  <build>
    <finalName>testuserServer</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.3.1</version>
      </plugin>
      <plugin>
        <groupId>org.eclipse.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>9.4.35.v20201120</version>
        <configuration>
          <scanIntervalSeconds>30</scanIntervalSeconds>
          <webApp>
            <contextPath>/st</contextPath>
          </webApp>
          <httpConnector>
            <port>80</port>
          </httpConnector>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

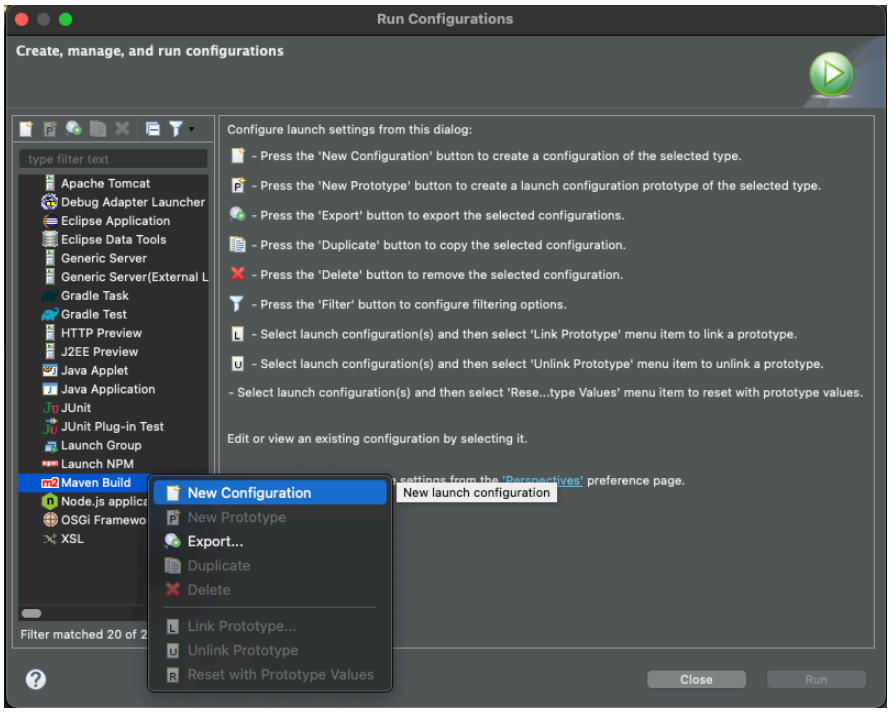
7. 테스트용 Servlet을 작성한다.
    1. 왼쪽의 Project Explorer에서 “Java Resources”에서 src/main/java를 선택한다.
    2. 마우스 오른쪽 버튼을 눌러서 팝업메뉴에서 New -> Class를 선택한다.
    3. TestServlet클래스를 생성하고 아래의 코드를 입력한다.

```
package kr.testuser.server;

import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.core.Response.ResponseBuilder;
import javax.ws.rs.core.Response.Status;

@Path("/test")
public class TestServlet {
	@GET
	@Consumes({MediaType.APPLICATION_JSON})
	@Produces({MediaType.APPLICATION_JSON})
	@Path("/getTest/{pathParam}")
	public Response testFunction(@Context HttpServletRequest request,
				@PathParam("pathParam") String pathParam,
				@QueryParam("queryParam") String queryParam) {
		try {
			ResponseBuilder builder = Response
				.status(Status.INTERNAL_SERVER_ERROR)
				.entity("Tet")
				.type(MediaType.APPLICATION_JSON)
				.header("Access-Control-Allow-Origin", "*")
				.header("Access-Control-Allow-Credentials", "true");
			return builder.build();
		} catch (Exception e) {
			e.printStackTrace();
			ResponseBuilder builder = Response
				.status(Status.INTERNAL_SERVER_ERROR)
				.entity(e.getMessage())
				.type(MediaType.APPLICATION_JSON)
				.header("Access-Control-Allow-Origin", "*")
				.header("Access-Control-Allow-Credentials", "true");
			return builder.build();
		}
	}
}
```

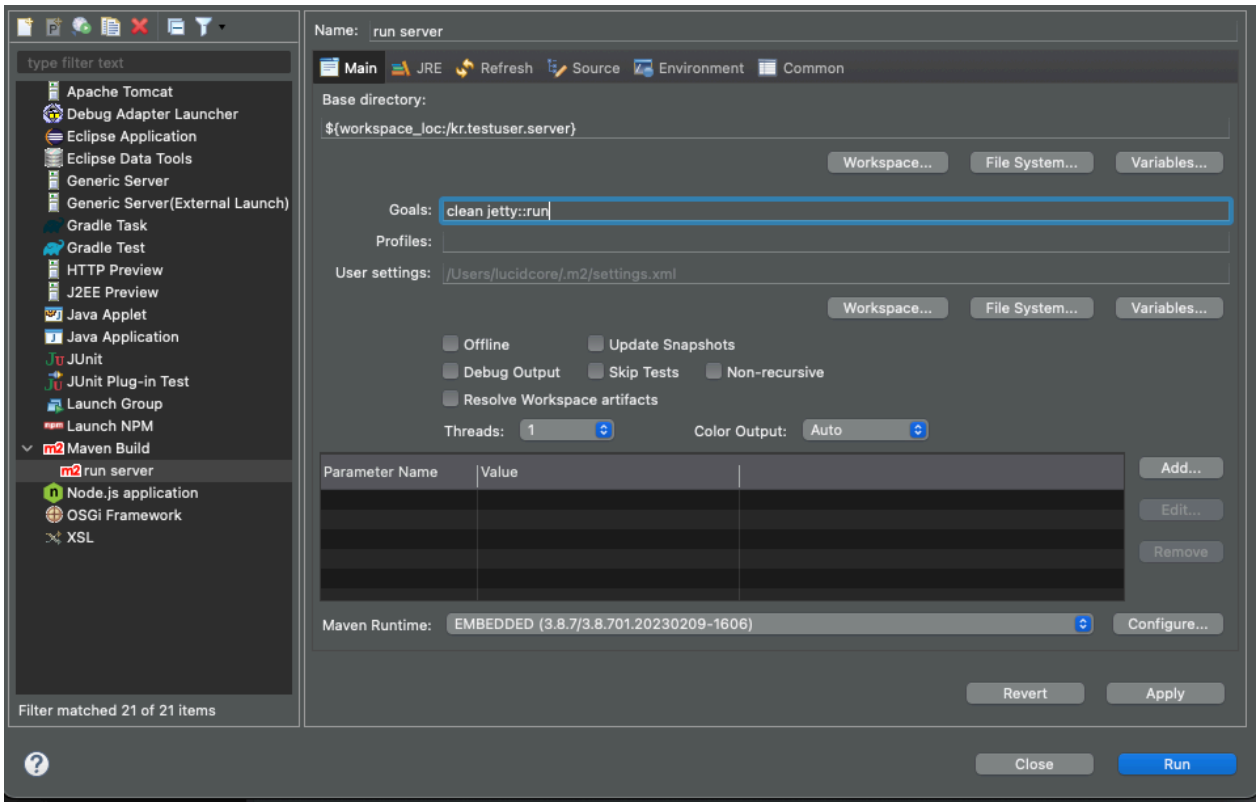
8. web.xml을 추가한다.
    1. 왼쪽의 Project Explorer에서 src -> main -> webapp을 선택한다.
    2. WEB-INF 디렉토리를 생성한다.
    3. web.xml을 생성하도 아래의 내용을 입력한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns="http://xmlns.jcp.org/xml/ns/javaee"
           xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/
xml/ns/javaee/web-app_3_1.xsd"
           version="3.1">
  <display-name>TestServer Application</display-name>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
```

9. Maven Build설정을 추가한다.
    1. Run -> Run Configuration -> Maven Build -> New Configuration을 선택한다.



    2. Name / Base directory / Goals을 적절히 입력한다.



10. Apply를 눌러서 저장하 Run을 눌러서 실행한다.
    1. 하단부의 Console에 “Started Jetty Server”가 뜨면 서버의 설정은 완료된 것임
11. 테스트를 위해서 부라우저를 띄워서 작성한 URL이 잘 뜨는지 확인한다.

1. http://127.0.0.1/st/svc/test/getTest/test1?queryParam=test2
    1. /st는 pom.xml에 설정된 경로임
    2. /svc는 web.xml에 설정된 경로임
    3. /test는 TestServlet에 제일 상단에 정의된 Path임
    4. /getTest는 testFunction에 정의된 Path임.
    5. {pathParam}은 뒤에 url경로로 입력받는 변수임. TestServlet에서 pathParam변수 내용에는 test1이 들어 있음.
    6. “Queryparam”은 parameter로 입력받는 변수로 TestServlet에서 queryParam변수 내용에는 test2가 들어 있음.
2. 부라우저로 위의 URL을 열어보면 화면에 “Tet”문자열이 표시될 것임.