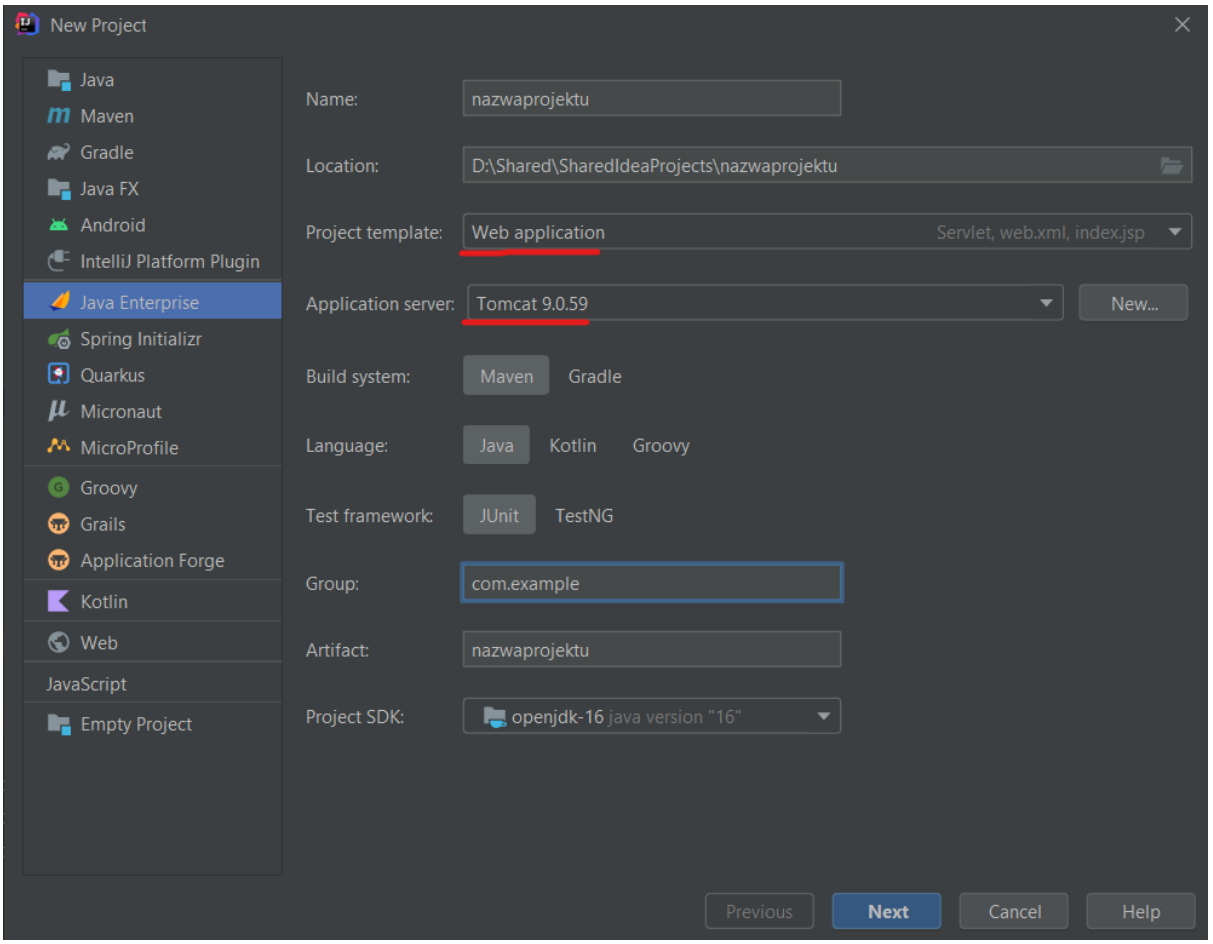
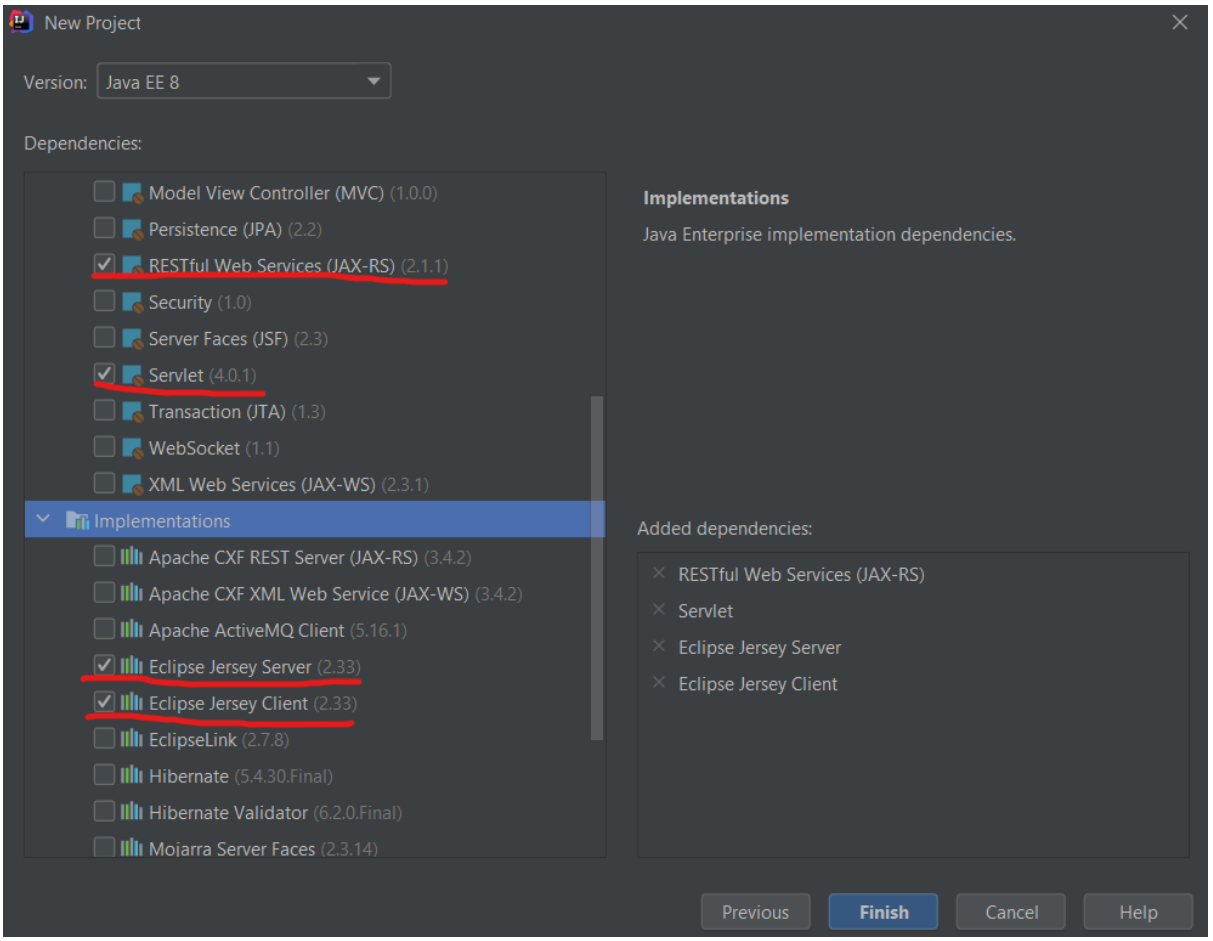
Konfiguracja:

Pobieramy i rozpakowujemy Tomcat 9, (Tomcat 10 nie działał, Tomcat 8 może zadziała)

New Project ->Java Enterprise -> wybieramy Web application
Jako application server wybieramy new i wskazujemy folder z rozpakowanym tomcatem **(nie w wersji 10)**
Co do SDK ja używałem 16-ki i nie napotkałem problemów. Słyszałem głosy, że musi być JDK < 9, ale mi działało na nowszych.



Ja na start wybrałem takie dependencies

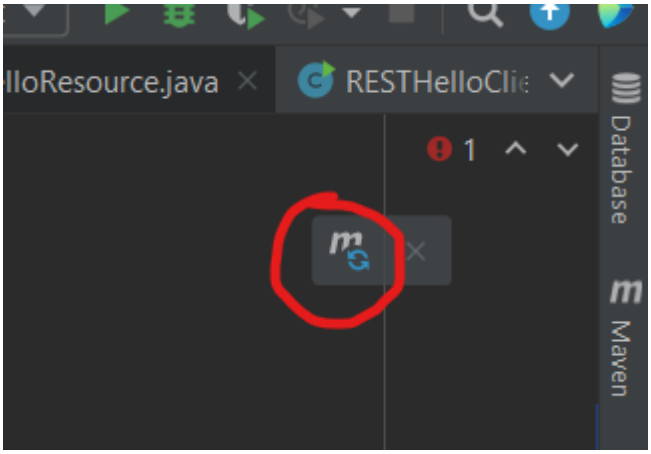


Polecam w tym miejscu kliknąć play i kontrolnie zobaczyć czy projekt startowy InteliJ działa.

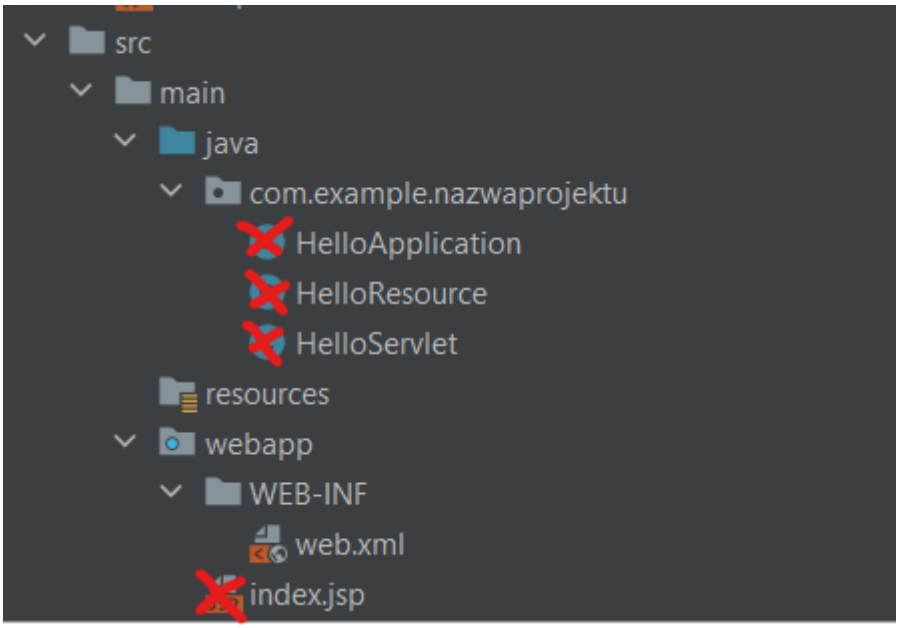
Znajdujemy plik pom.xml i podmieniamy tamtejszy tag <dependencies> … </dependencies> na poniższy:

```
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.containers</groupId>
            <artifactId>jersey-container-servlet</artifactId>
            <version>2.31</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.media</groupId>
            <artifactId>jersey-media-json-jackson</artifactId>
            <version>2.31</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.inject</groupId>
            <artifactId>jersey-hk2</artifactId>
            <version>2.31</version>
        </dependency>
        <dependency>
            <groupId>org.glassfish.jersey.core</groupId>
            <artifactId>jersey-client</artifactId>
            <version>2.31</version>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
            <version>2.3.3</version>
        </dependency>

    </dependencies>
```

Po zmianach w pom.xml pamiętamy, żeby zsynchronizować:



Usuwamy te pliki:



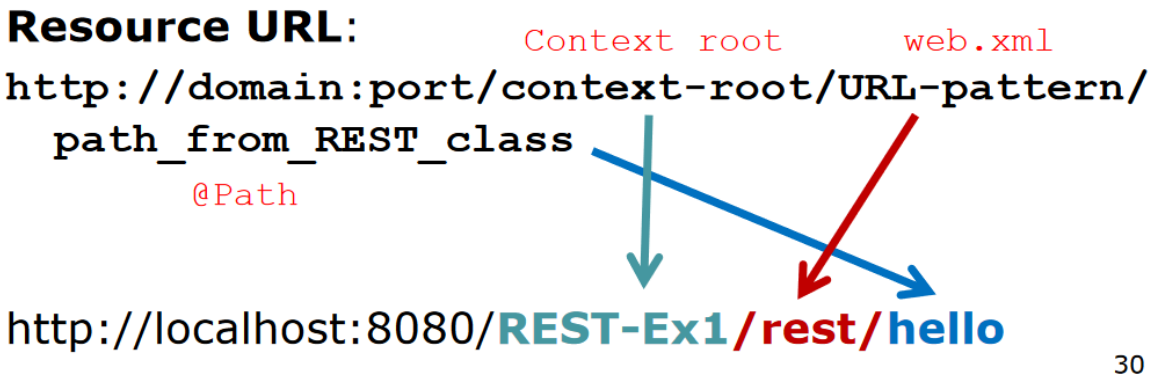
I kontynuujemy wg. instrukcji

Ćwiczenie 1:

Uwaga, żeby w web.xml wpisać nazwę swojego package'u, a nie "services.rest"

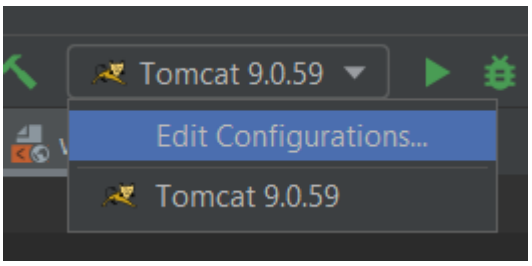
```
<init-param>
    <param-name>jersey.config.server.provider.packages</param-name>
    <param-value>com.example.temp</param-value>
</init-param>
```

Co do tego:

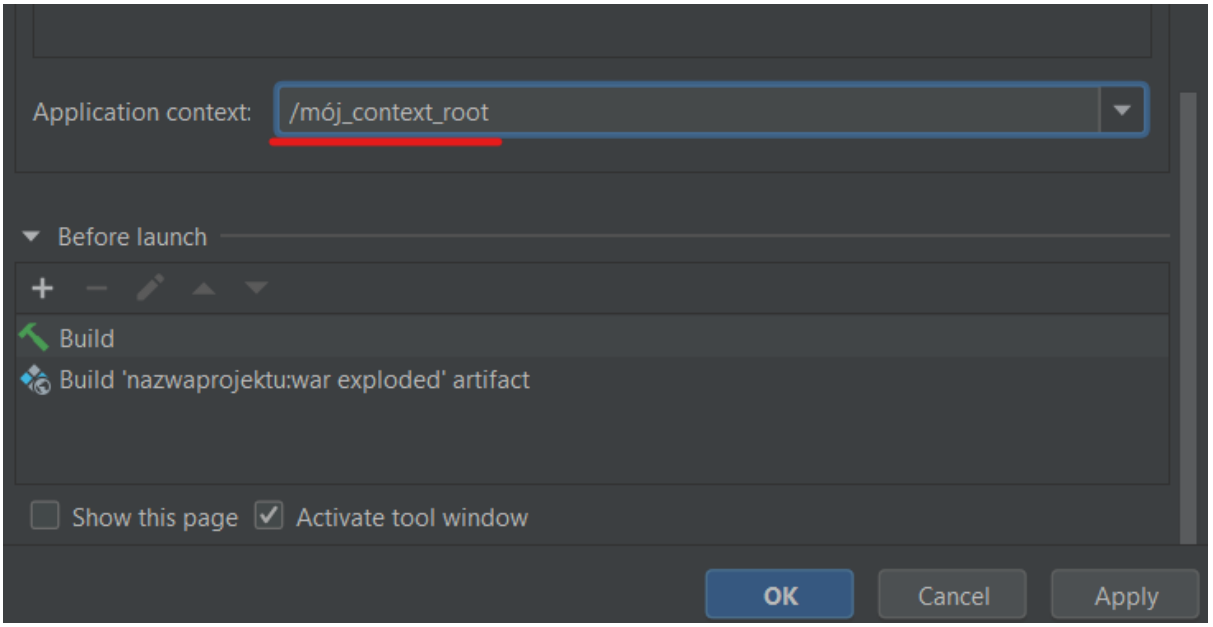


30

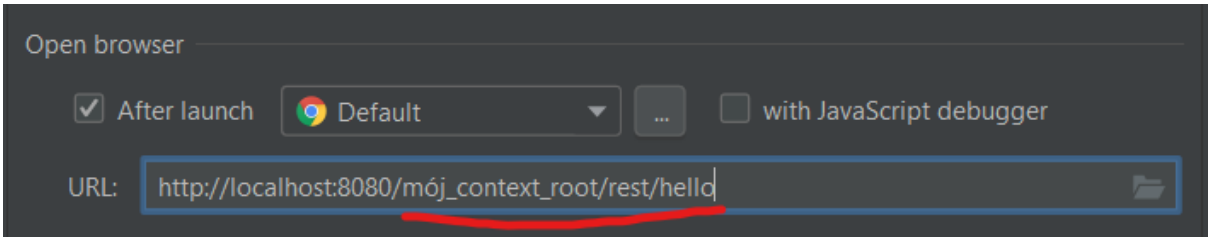
"Context root" to jest to co wpisujemy w edycji konfiguracji:



Wchodzimy w zakładkę “Deployment”, scrollujemy na dół i uzupełniamy “Application context”



Warto też w zakładce “Server” ustawić adres otwieranej strony



Reszta identycznie z instrukcją

Ćwiczenie 2, 3, 4:

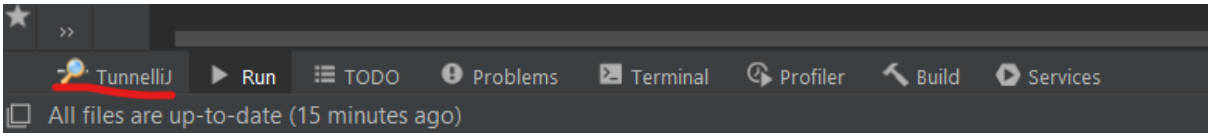
Identycznie z instrukcją

Jeśli po stronie serwera System.out.println nic nie wypisuję polecam spróbować ustawić breakpointa po princie i uruchomienia serwera ikonką “żuczka” zamiast “play”

Ćwiczenie 5:

File -> Settings -> Plugins -> wyszukujemy i instalujemy TunnelliJ

Po restarcie IDE wchodzimy w zakładkę TunnelliJ



Po prawej wpisujemy port na którym działa serwer, po lewej dowolny inny port (tutaj 4444)

Modyfikujemy klientów, żeby wysyłali na port 4444

```
    public static void main(String[] args) {
        Client client = ClientBuilder.newClient();
//          WebTarget target = client.target(getBaseURI());
        WebTarget target = client.target(URI.create("http://localhost:4444/mój_context_root"));
```

Pamiętamy, żeby podać ten context root podany w “edit configuration” i ten port ustawiony w TunnelliJ

Klikamy “play” w zakładce TunnelliJ, uruchamiamy ponownie serwer i powinniśmy mieć możliwość podglądania przesyłanych danych w zakładce TunnelliJ