Notatki - Maven - Intellij

Spis treści

Integracja Maven z Intellij

Najpierw sam Maven, bez Intellij. Dodajmy teraz w pliku pom.xml poniższą konfigurację.

```
<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>pl.zajavka</groupId>
    <artifactId>java-maven-examples</artifactId>
    <version>1.0.0</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>1.14.2</version>
        </dependency>
    </dependencies>
</project>
```

Pamiętajmy jednocześnie o podejściu **Convention over configuration**.

W głównym katalogu projektu (czyli tam, gdzie jest plik pom.xml) umieszczamy folder: src/main/java i dopiero w nim umieszczamy katalog pl/zajavka. W nim umieszczamy plik MavenCompilingJsoupExamplesRunner.java. Czyli struktura wygląda teraz tak:

```
główny_katalog_projektu
  - pom.xml
  - src
    - main
```

```
      - java
        - pl
          - zajavka
            MavenCompilingJsoupExamplesRunner.java
  - target
```

Teraz (w terminalu) będąc w folderze **główny_katalog_projektu** możemy uruchomić polecenie:

```
mvn compile
```

Na ekranie wydrukowane zostanie coś podobnego do:

```
[INFO] Scanning for projects...
[INFO]
[INFO] -------------------< pl.zajavka:java-maven-examples >-------------------
[INFO] Building java-maven-examples 1.0.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ java-maven-examples ---
...
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ java-maven-examples ---
[INFO] Changes detected - recompiling the module!
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.148 s
[INFO] Finished at: 2021-09-30T13:48:40+02:00
[INFO] ------------------------------------------------------------------------
```

Jeżeli natomiast chcemy aby `Maven` wydrukował o wiele więcej informacji to napiszemy:

```
mvn compile -X
```

Informacja **BUILD SUCCESS** mówi nam o tym, że `Maven` z powodzeniem zakończył swoje działanie. Nie oznacza to natomiast, że zrobił dokładnie to co chcieliśmy bo zawsze mogliśmy popełnić gdzieś błąd.

Maven compiler plugin

Co oznacza w konfiguracji `pom.xml` taki zapis:

```
<properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
</properties>
```

Bardzo dobrze wyjaśnia to dokumentacja

Czyli jeżeli chcemy używać funkcji, które zostały dodane w Java 17, dodajemy opcję `source` z

parametrem 17. Jeżeli chcemy, aby skompilowane klasy były kompatybilne z JVM w wersji 17, dodajemy zapis `target`. Dla innych wersji Javy upewnij się proszę z dokumentacją zanim dokonasz zmian ☺.

Jeżeli teraz zmienię zmienię zawartość pliku `.java` i dodam do niego metodę `List.of()`, której nie ma w Javie 7, oraz ustawię **source** i **target** na **1.7**:

```
package pl.zajavka;

import org.jsoup.Jsoup;

import java.io.IOException;
import java.time.LocalDateTime;
import java.util.List;

public class MavenCompilingJsoupExamplesRunner {

    public static void main(String[] args) throws IOException {
        System.out.println(Jsoup.connect("https://app.zajavka.pl").get().title());

        List.of("");
    }
}
```

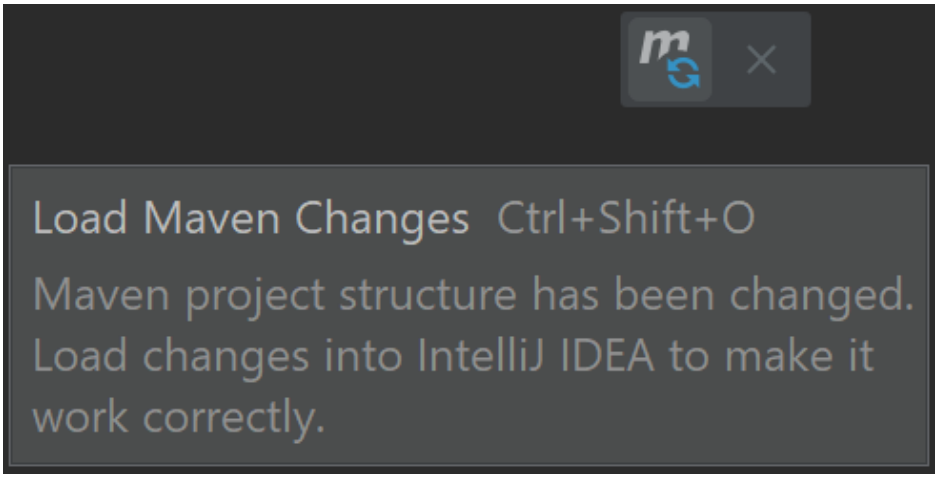
Dostanę poniższy błąd:

```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile
    (default-compile) on project java-maven-examples: Compilation failure
[ERROR] .../pl/zajavka/MavenCompilingJsoupExamplesRunner.java:[14,13]
    static interface method invocations are not supported in -source 7
[ERROR]   (use -source 8 or higher to enable static interface method invocations)
[ERROR]
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Czyli widzimy już, że `Maven` pomaga nam automatycznie 'zaciągać' zależności - biblioteki zewnętrzne.

Dołóżmy do tego Intellij

Jeżeli dopiszemy `dependencies` w pliku `pom.xml` oraz mamy poprawnie skonfigurowany projekt, nie musimy nawet wywoływać z terminala komendy `mvn compile`, albo innej. W prawym górnym rogu Intellij wyświetli nam ikonkę, która oznacza możliwość odświeżenia dopisanych zależności.



Obraz 1. Load Maven Changes

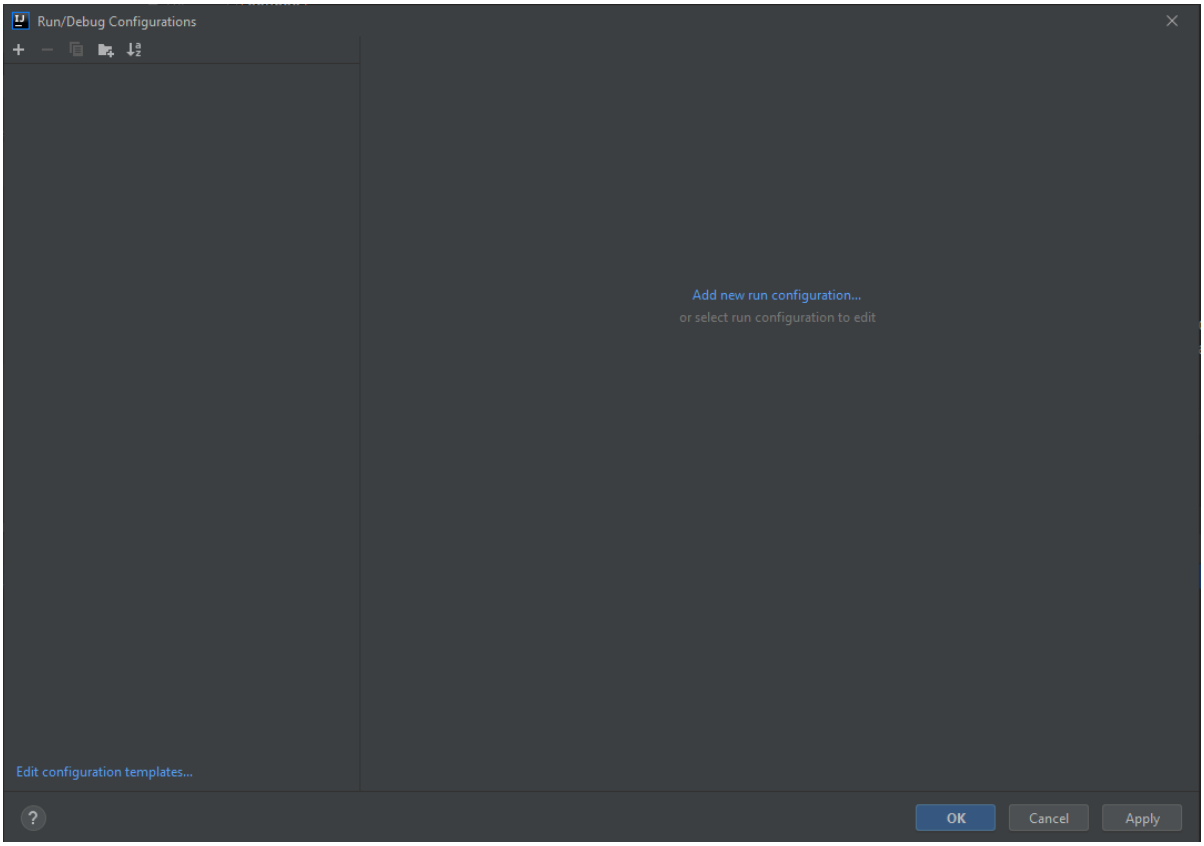
W tym momencie następuje pobranie wymaganych zależności. Jeżeli teraz chcemy uruchomić nasz projekt, wystarczy kliknąć zieloną strzałkę, którą już doskonale znamy. Intellij dodaje nam lokalizację bibliotek sam, do komendy wywołującej program.

```
C:\...\java.exe
    ...
    -classpath
        C:\Users\karol\java-pl\zajavka\examples-maven\target\classes;
        C:\Users\karol\.m2\repository\org\jsoup\jsoup\1.14.2\jsoup-1.14.2.jar
    pl.zajavka.MavenCompilingJsoupExamplesRunner
```

W ten sposób wiemy już jak działa automatyzacja związana z integracją maven i Intellij. Ale to nie wszystko.

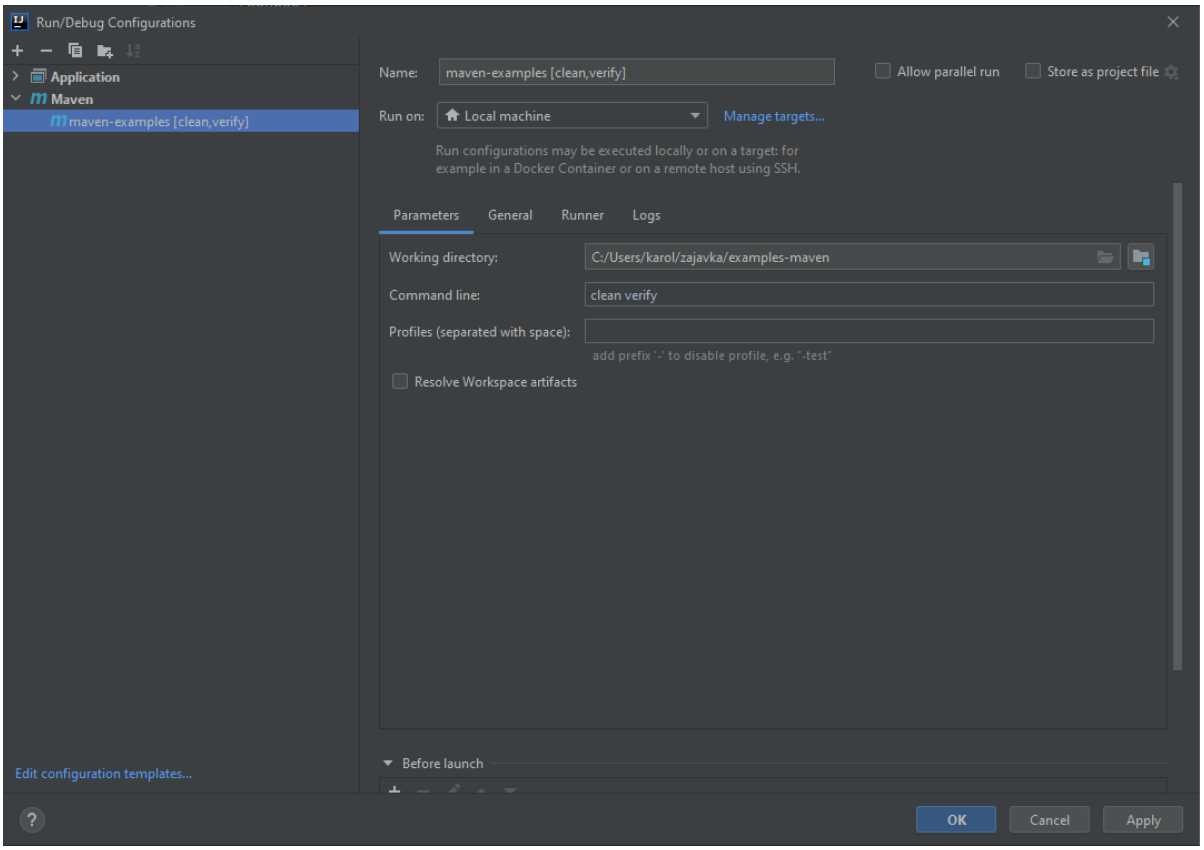
Uruchamianie poleceń Maven z Intellij

Jeżeli wciśniesz teraz dwukrotnie `Shift` i wpiszesz `edit configurations` pokaże Ci się okno, którym można definiować zadania uruchomieniowe. Dokładnie to samo robi Intellij gdy uruchamiamy nasz program z metody main. Dodawana jest wtedy konfiguracja uruchomieniowa programu. Możemy również sami zdefiniować taką konfigurację w przypadku Maven.



Obraz 2. Tworzenie konfiguracji uruchomienia Maven z Intellij

Następnie możemy wybrać '+' po lewej stronie i wybrać konfigurację `Maven`. Możemy wtedy stworzyć taką konfigurację:



Obraz 3. Własna konfiguracja Maven z Intellij

Możemy wybrać teraz konfigurację o nazwie `maven-examples [clean,verify]` i uruchomić Maven z poziomu Intellij.

To co Intellij tak na prawdę zrobi to uruchomi polecenie, które można zobaczyć jako pierwsze w oknie z konsolą, gdzie drukowany jest rezultat wywołania `Maven`.

Okno Maven po prawej stronie ekranu

Maven również dodaje nam możliwość zobaczenia drzewo `lifecycle`, `phase` i `goals` w oknie po prawej stronie ekranu. Okno to pokazuje również zdefiniowane przez nas konfiguracje. Możemy też tutaj zobaczyć biblioteki, które zostały pobrane do nas na maszynę.