|  |  |
| --- | --- |
| Plugin | * Zapewnia kompilacje kodu projektu Mavena * Cele:  compile - kompiluje pliki katalogu main testCompile - kompiluje pliki testowe   <**plugin**>      <**artifactId**>maven-compiler-plugin</**artifactId**>      <**version**>3.8.1</**version**>      <**configuration**>          ...      </**configuration**>  </**plugin**>  [https://search.maven.org/classic/#search%7Cga%7C1%7Cg%3A%22org.apache.maven.plugins%22%20AND%20a%3A%22maven-compiler-plugin%22](https://search.maven.org/classic/" \l "search|ga|1|g:"org.apache.maven.plugins" AND a:"maven-compiler-plugin") |
| Java Version | * Domyślnie Maven używa Java 5   <**properties**>      <**maven.compiler.source**>1.8</**maven.compiler.source**>      <**maven.compiler.target**>1.8</**maven.compiler.target**>  </**properties**>   * Od Java 8, wersja Javy nie musi być poprzedzona zapisem 1.x   <**configuration**>      <**source**>9</**source**>      <**target**>9</**target**>  </**configuration**>  Lub zapisu  <**properties**>      <**maven.compiler.source**>9</**maven.compiler.source**>      <**maven.compiler.target**>9</**maven.compiler.target**>  </**properties**>   * Od Java 9 należy użyć wersji Mavena min. 3.8.0   <**build**>      <**plugins**>          <**plugin**>              <**groupId**>org.apache.maven.plugins</**groupId**>              <**artifactId**>maven-compiler-plugin</**artifactId**>              <**version**>3.8.0</**version**>              <**configuration**>                  <**source**>9</**source**>                  <**target**>9</**target**>              </**configuration**>          </**plugin**>      </**plugins**>  </**build**> |
| Configuration | * Można również skonfigurować argumenty javac, używając tagu <compilerArgs> wewnątrz, której można użyć tagu <arg>   <**configuration**>      <!-- other configuration -->      <**compilerArgs**>          <**arg**>-Xlint:unchecked</**arg**>      </**compilerArgs**>  </**configuration**> |
| Parent | * Za pomocą tagu <parent> można dodać   <**parent**>      <**groupId**>org.springframework.boot</**groupId**>      <**artifactId**>spring-boot-starter-parent</**artifactId**>      <**version**>2.0.5.RELEASE</**version**>  </**parent**> |
| Moduły | W intellij klikamy prawy przycisk myszki na projekcie i wybieramy “new -> module”.  Przy wielu modułach projektu w pliku pom.xml dodajemy:  <**packing**>pom</**packing**>  Wtedy program wie że ten artefakt nie ma być .jar albo .war ale jest ogólnie definiującym POMem. W takim POMie można dodać dependacies które potem przejmą wszystkie niższe pomy:  <**dependencyManagment**>  ...  </**dependencyManagment**>  tutaj możemy definiować werjse dependancies itd żeby potem poniższe POMy przejęły te dependency:  <**dependency**>      <**groupId**>zgodne z głownym POM dla danej dep.</**groupId**>      <**artifactId**>jak wyżej</**artifactId**>  </**dependency**>  Przy wielomodułowych projektach w głównym POMie oznaczamy moduły projektu::  <**modules**>      <**module**>nazwa modułu</**module**>  </**modules**>  Potem dany moduł musi mieć artifactId zgodne z tym co zapisane w głównym POM:    <**artifactId**>modul1</**artifactId**>  Definiowanie properties (używamy ich jak zmiennych w pliku POM), np.:    <**properties**>      <**logback.version**>1.2.3</**logback.version**>  </**properties**>  <**version**>${**logback.version**}</**version**> |