



IntelliJ IDEA 2023.2 ▼

[Compilation and build...](#) / [Build to...](#) / [Mav...](#) / [Maven tutor...](#) / [Troubleshooting common Maven](#)

# Troubleshooting common Maven issues

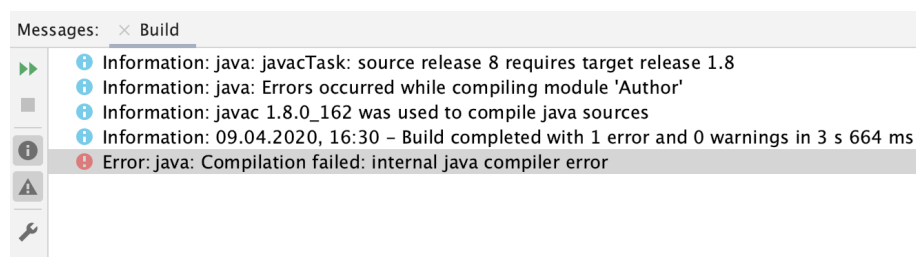
Last modified: 24 August 2023

If you encounter problems working with your Maven project you can check to see if the following solutions and workarounds can help you solve your issues.

## How to fix compiler version problems in Maven projects

In some cases when you import a Maven project, it might have compiler settings that will not match the expected settings in IntelliJ IDEA and when you compile your code, you might encounter a problem.

For example, you can get the following error:




This error usually indicates a problem with the compiler version compatibility, and you can check few places to fix it.

For example, you can edit your POM and configure Maven compiler plugin [↗](#) to compile your Java code. You should set the compiler level explicitly, so it won't revert to the default settings when you re-import your project.

## Set the compiler level in POM




1. Open your POM in the editor.
2. Change the configuration for the [Maven compiler plugin](#) [↗](#).

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.7.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>
  </plugins>
</build>
```

3. Click  to import your changes. Note that the configurations specified in your POM override any configurations specified in your project structure. So, now after this project is imported the language level should be picked up.

Check the Java compiler settings to see if the bytecode versions match.

## Check the compiler settings

1. In the **Settings** dialog (  ), go to **Build, Execution, Deployment | Compiler | Java Compiler**.

2. On the page that opens, check if **Project bytecode version** and **Target bytecode version** match, or leave the **Target bytecode version** option blank so it can be determined from JDK.

**Build, Execution, Deployment** > **Compiler** > **Java Compiler** For current project

Use compiler: javac

☒ Use '--release' option for cross-compilation (Java 9 and later)

Project bytecode version: Same as language level

Per-module bytecode version:

Module	Target bytecode version
Author	Same as language level
Book	1.8
BookStore	1.8
Genre	1.8
MyGradle	1.8

Java Options

☒ Use compiler from module target JDK when possible

☒ Generate debugging info

☒ Report use of deprecated features

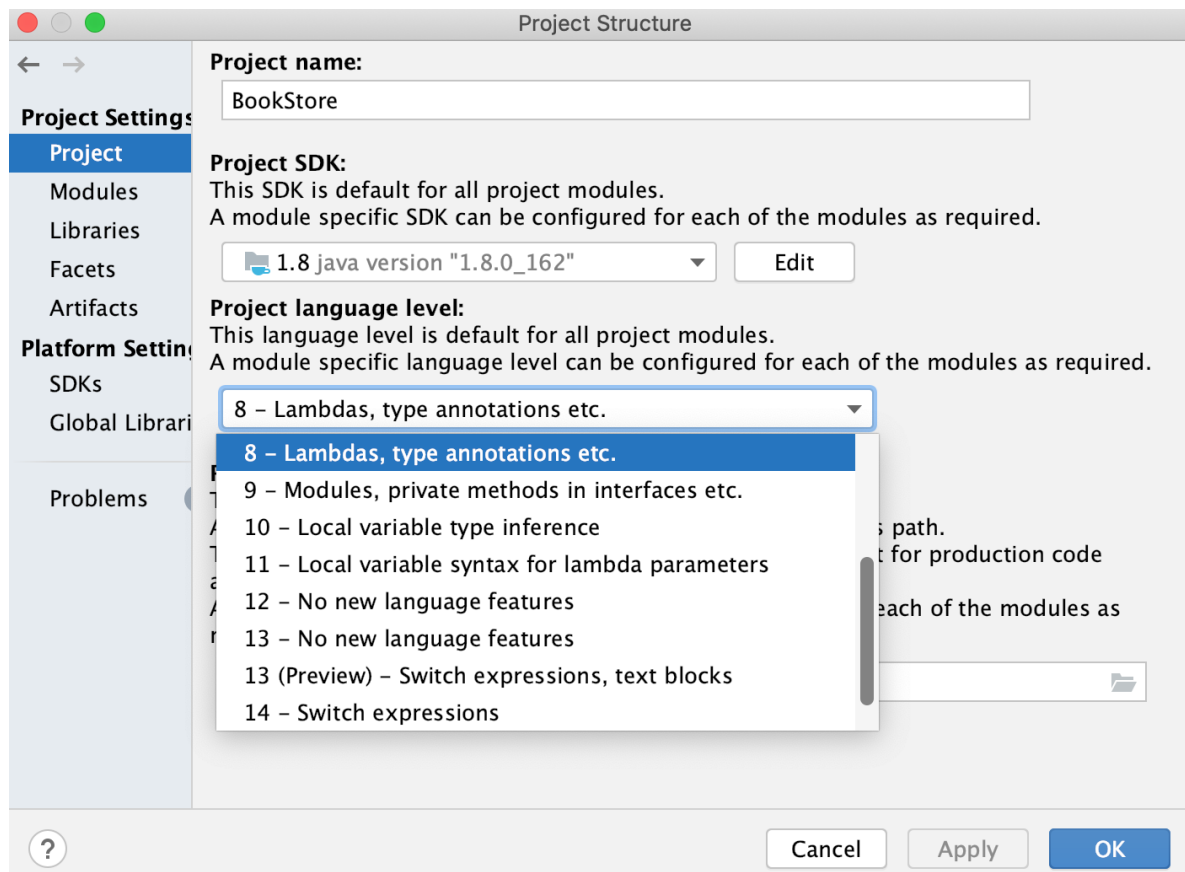
☐ Generate no warnings

Additional command line parameters: ('/' recommended in paths for cross-platform configurations)

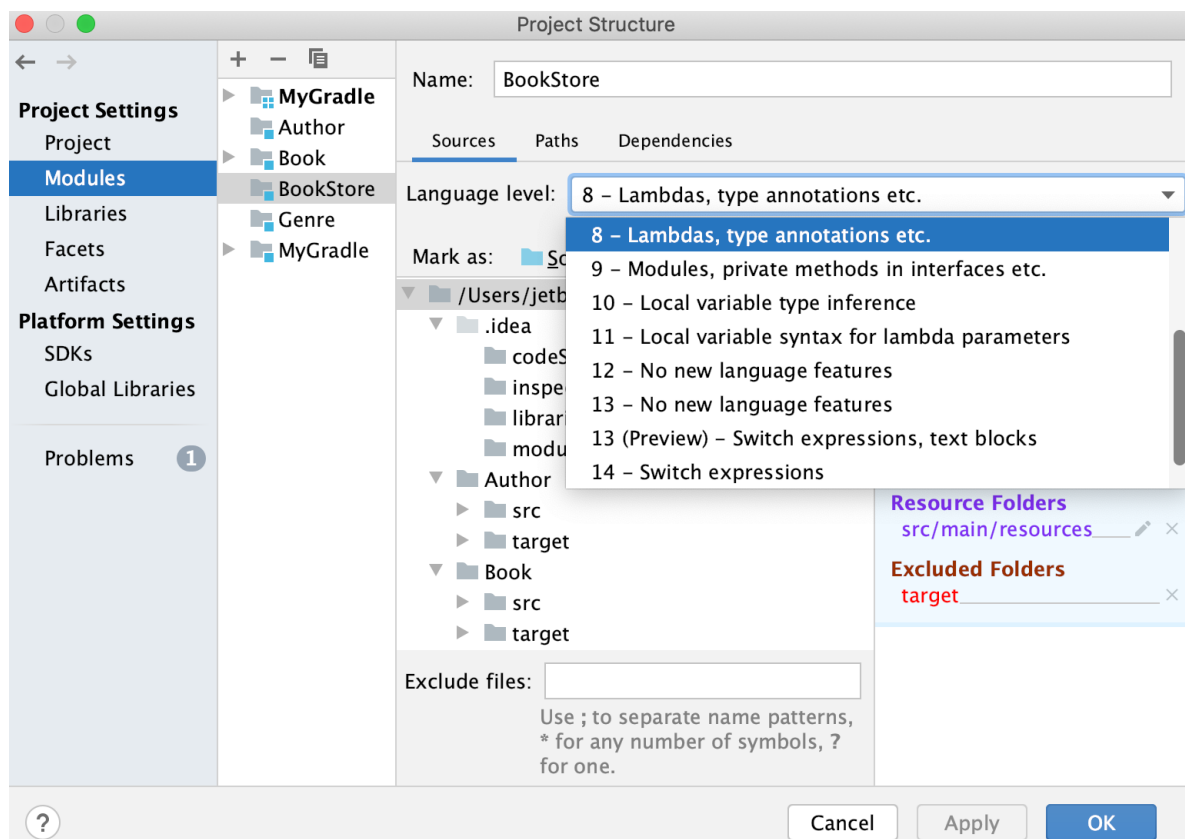
If you have imported a multi-level project, you can check project structure settings for source language level configuration.

## Check the language level in the project structure

1. Open **Project Structure** dialog and select **Project** from the options on the left.
2. Check the source language level for your project.



3. Check the source language level for each module (click the **Sources** tab).




# How to fix problems with Maven projects that won't start

To fix issues that are related to the Maven projects that won't start or import, you can perform one of the following actions.

If you received the `OutOfMemory` error, try to increase the heap size for the Maven importer.

1. In the **Settings** dialog (**Ctrl** **Alt** **S**), go to **Build, Execution, Deployment | Build Tools | Maven | Importing**.



You can click the  icon in the **Maven** tool window to open the Maven settings.

2. On the **Importing** page, in the **VM options for importer** field, increase heap size for the Maven importer.
3. Also, in the **JDK for importer** field, increase IDE heap size ↗.

**Build, Execution, Deployment > Build Tools > Maven > Importing** For current project

☐ Keep project files in:

☐ Store generated project files externally

☒ Detect compiler automatically

☒ Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)

☐ Create module groups for multi-module Maven projects

☒ Keep source and test folders on reimport

☒ Exclude build directory (%PROJECT\_ROOT%/target)

☒ Use Maven output directories

Generated sources folders:

Phase to be used for folders update:

IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.  
Note that all test-\* phases firstly generate and compile production sources.

Automatically download: ☐ Sources ☐ Documentation ☐ Annotations

Dependency types:   
Comma separated list of dependency types that should be imported

**VM options for importer:**

**JDK for importer:**

If you need to use more heap, switch to 64-bit Java and specify the same 64-

bit JVM for Maven **JDK for importer**.

If you received the `Operation timed out` error or IDE connection failure to the Maven process, try to edit the **hosts** file.

### Edit the hosts file

- On some systems you need to edit the **hosts** file so that **localhost** resolves correctly. Try to have `127.0.0.1 localhost` in the **etc/hosts** file. Also, make sure there are no other IP addresses mapped to **localhost**.

If the error indicates the Maven repository issue, such as the `Failed to update Maven indices` error, try to check if Maven repositories were indexed correctly.


IntelliJ IDEA works with repository indexes. The indexes are fetched remotely from remote repositories. Some repositories do not provide indexes, or do not keep an updated index, for example, repositories from [Bintray ↗](#), in this case you can ignore the error.

If you have an indexed repository, but still get a Maven repository error, check the following options:

### Check the user settings file

- In the **Settings** dialog (`Ctrl Alt S`), go to **Build, Execution, Deployment | Build Tools | Maven**.



You can click the  icon in the **Maven** tool window to open the Maven settings.

- On the **Maven** page, in the **User settings file** field, check if you defined proper credentials for the server in **settings.xml**.

**Build, Execution, Deployment** > **Build Tools** > **Maven** For current project [Reset](#)

☐ Work offline

☐ Use plugin registry

☒ Execute goals recursively

☐ Print exception stack traces

☐ Always update snapshots

☒ Update Incices on project open

Output level:

Checksum policy:

Multiproject build fail policy:

Thread count  -T option

Maven home directory:

(Version: 3.6.1)

User settings file:  ☒ **Override**

Local repository:  ☐ **Override**

You can try to restart IntelliJ IDEA and update Maven repositories.

## Update Maven repositories

1. In the **Settings** dialog (  ), go to **Build, Execution, Deployment | Build Tools | Maven | Repositories**.



You can click the  icon in the **Maven** tool window to open the Maven settings.

2. Select **Repositories** from options on the left.
3. On the **Repositories** page, click **Update** to update Maven repositories.

Build, Execution, Deployment &gt; Build Tools &gt; Maven &gt; Repositories

For current project

Indexed Maven Repositories:

URL	Type	Updated	
https://repo.maven.apache.org/maven2	Remote	Never	
/Users/jetbrains/.m2/repository	Local	19.03.2...	

Update

4. After the update is finished, click **OK**.

## Maven dependencies imported incorrectly

If the dependencies weren't imported correctly (IntelliJ IDEA highlights them), try to perform the following actions:

- You can check your local maven repository in the [Maven | Repositories](#) settings and try to update it.
- You can check the **jar** file of the local `.m2` repository to see if it was downloaded correctly.
- You can check the effective POM to determine which Maven repository was used as an [origin of the dependency](#).
- You can select the **Always update snapshots** option in [Maven](#) settings. In this case, IntelliJ IDEA checks the latest version of the downloaded dependency and updates it accordingly.



**Build, Execution, Deployment** > **Build Tools** > **Maven** For current project Reset

☐ Work offline

☐ Use plugin registry

☒ Execute goals recursively

☐ Print exception stack traces

☒ Always update snapshots

☒ Update indices on project open

Output level:

Checksum policy:

Multiproject build fail policy:

Plugin update policy:  ignored by Maven 3+

Thread count:  -T option

Maven home directory:  ...  
(Version: 3.6.1)

User settings file:  ... ☐ Override

Local repository:  ... ☐ Override

Cancel Apply OK