Java: How to resolve java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException

Asked 3 years, 7 months ago Active 2 days ago Viewed 709k times

919

262

I have some code that uses JAXB API classes which have been provided as a part of the JDK in Java 6/7/8. When I run the same code with Java 9, at runtime I get errors indicating that JAXB classes can not be found.

The JAXB classes have been provided as a part of the JDK since Java 6, so why can Java 9 no longer find these classes?

java jaxb java-9 java-11 java-10

edited Oct 6 at 8:46
Promise Preston
4,921 ●3 ●21 ●46

asked Apr 23 '17 at 17:40
Andy Guibert
32.4k ●7 ●32 ●53

2 The additional part in this answer relates to the migration of these API's. – Naman Oct 21 '17 at 7:05

9 building with Java 8 will get your code to compile yes, but when you try to run that compiled code on Java 9+ it will fail because JAX-B is not present. – Andy Guibert Oct 24 '18 at 12:07

2 For Java 11, this article's solution is up to date: crunchify.com/java-11-and-javax-xml-bind-jaxbcontext – Eric Wang Jan 11 '19 at 9:33

see wiki.bitplan.com/index.php/Java8 – Wolfgang Fahl Oct 10 '19 at 13:41

34 Answers

Active Oldest Votes

1 2 Next

1345

The JAXB APIs are considered to be Java EE APIs and therefore are no longer contained on the default classpath in Java SE 9. In Java 11, they are completely removed from the JDK.

Java 9 introduces the concepts of modules, and by default, the `java.se` aggregate module is available on the classpath (or rather, module-path). As the name implies, the `java.se` aggregate module does *not* include the Java EE APIs that have been traditionally bundled with Java 6/7/8.

Fortunately, these Java EE APIs that were provided in JDK 6/7/8 are still in the JDK, but they just aren't on the classpath by default. The extra Java EE APIs are provided in the following

```
java.activation
java.corba
java.transaction
java.xml.bind  << This one contains the JAXB APIs
java.xml.ws
java.xml.ws.annotation
```

Quick and dirty solution: (JDK 9/10 only)

To make the JAXB APIs available at runtime, specify the following command-line option:

`--add-modules java.xml.bind`

But I still need this to work with Java 8!!!

If you try specifying `--add-modules` with an older JDK, it will blow up because it's an unrecognized option. I suggest one of two options:

1. You can set any Java 9+ only options using the `JDK_JAVA_OPTIONS` environment variable. This environment variable is automatically read by the `java` launcher for Java 9+.
2. You can add the `-XX:+IgnoreUnrecognizedVMOptions` to make the JVM silently ignore unrecognized options, instead of blowing up. But beware! Any other command-line arguments you use will no longer be validated for you by the JVM. This option works with Oracle/OpenJDK as well as IBM JDK (as of JDK 8sr4).

Alternate quick solution: (JDK 9/10 only)

Note that you can make all of the above Java EE modules available at run time by specifying the `--add-modules java.se.ee` option. The `java.se.ee` module is an aggregate module that includes `java.se.ee` as well as the above Java EE API modules. Note, this **doesn't work on Java 11** because `java.se.ee` was removed in Java 11.

Proper long-term solution: (JDK 9 and beyond)

The Java EE API modules listed above are all marked `@Deprecated(forRemoval=true)` because they are scheduled for removal in Java 11. So the `--add-module` approach will no longer work in Java 11 out-of-the-box.

What you will need to do in Java 11 and forward is include your own copy of the Java EE APIs on the classpath or module path. For example, you can add the JAX-B APIs as a Maven dependency like this:

```
<!-- API, java.xml.bind module -->
<dependency>
```

```
</dependency>

<!-- Runtime, com.sun.xml.bind module -->
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.2</version>
</dependency>
```

See the JAXB Reference Implementation page for more details on JAXB.

For full details on Java modularity, see JEP 261: Module System

For Gradle or Android Studio developer: (JDK 9 and beyond)

Add the following dependencies to your `build.gradle` file:

```
dependencies {
    // JAX-B dependencies for JDK 9+
    implementation "jakarta.xml.bind:jakarta.xml.bind-api:2.3.2"
    implementation "org.glassfish.jaxb:jaxb-runtime:2.3.2"
}
```

edited Sep 8 at 11:47
Naman
63.8k ●21 ●161 ●268

answered Apr 23 '17 at 17:40
Andy Guibert
32.4k ●7 ●32 ●53

9 So if the Java EE API modules are marked deprecated does that mean it is a possibility that in Java 10 JAXB will no longer be available at runtime in Java 10? That seems like a step backwards. We will have to go back to the pre-6 practice of including JAXB as a dependency. – Michael Sep 22 '17 at 17:35

4 Using --add-modules java.se.ee or --add-modules ALL-SYSTEM as a workaround is not recommended according to migration guide here docs.oracle.com/javase/9/migrate under section Modules Shared with Java EE Not Resolved by Default --> point 1 – justMe Oct 20 '17 at 14:32

6 With Java 10 officially released, we can confirm that the add-modules method will still work. The `javax.xml.bind` and other JavaEE classes are scheduled for removal in Java 11, per JEP-320. – Joep Weijers Mar 21 '18 at 11:02

11 And now Java 11 is released and the `java.se.ee` module has been removed, so the `--add-modules` solution does not work anymore. Use the recommended solution instead: add JAXB as a separate dependency. – Jesper Sep 30 '18 at 12:30

18 i've added these dependencies and it is still giving me the same error. any ideas why? – João Vieira Jul 8 '19 at 17:16

285

In my case (spring boot fat jar) I just add the following to pom.xml.

```
<dependency>
```

```
    <version>2.3.1</version>
</dependency>
```

edited Sep 21 '19 at 12:33

answered Nov 21 '17 at 12:07
jdev
3,834 ● 1 ● 12 ● 18

11 Just for reference github.com/spring-projects/spring-boot/wiki/… – Tuno Mar 8 '18 at 10:08

9 Adding gradle dependency like this `testCompile('javax.xml.bind:jaxb-api')` worked for me. – pamcevoy May 17 '18 at 15:07

5 Like @pamcevoy mentioned, no need to specify the version of jaxb-api when using Spring Boot. Boot manages the version automatically. – Marcel Overdijk Jul 12 '18 at 13:32

3 I suggest to use `<scope>runtime</scope>` for such case – VladS Sep 18 '18 at 22:43

6 @Tuno's link didn't work for me, fixed link is: github.com/spring-projects/spring-boot/wiki/… – Francisco Mateo Dec 12 '18 at 1:01

71

None of these solutions worked fine for me in the recent JDK 9.0.1.

I found that this list of dependencies is enough for a proper functioning, so you **don't need** to explicitly specify `--add-module` (though it is specified within these dependencies's pom's). The only you need is to specify this list of dependencies:

```
<dependencies>
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.0</version>
    </dependency>
    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.3.0</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
        <version>2.3.0</version>
    </dependency>
    <dependency>
        <groupId>javax.activation</groupId>
        <artifactId>activation</artifactId>
        <version>1.1.1</version>
    </dependency>
</dependencies>
```

edited Jun 4 '18 at 11:18

answered Jan 7 '18 at 12:06
Andremoniy

2 For JDK 8, remove the jaxb-core and jaxb-impl from above. – foo Jan 19 '18 at 4:48

3 @Anil this is a `pom.xml` file of the Maven configuration. If you don't know what is that, then it is better to start from beggining – Andremoniy Feb 13 '18 at 13:16

8 An illegal reflective access operation has occurred WARNING: Illegal reflective access by com.sun.xml.bind.v2.runtime.reflect.opt.Injector (file:/C:/Users/eis/.m2/repository/com/sun/xml/bind/jaxb-impl/2.3.0/jaxb-impl-2.3.0.jar) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int) WARNING: Please consider reporting this to the maintainers of com.sun.xml.bind.v2.runtime.reflect.opt.Injector WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations WARNING: All illegal access operations will be denied in a future release – Stefan Feb 19 '18 at 10:02

1 This worked for me on JDK 9.0.4 (I was calling JAXB related code through a Maven plugin with Maven 3.5.3). Although I would use `<dependency>             <groupId>javax.activation</groupId> <artifactId>javax.activation-api</artifactId>             <version>1.2.0</version> </dependency>` as last dependency. – scrutari Apr 20 '18 at 22:58

2 Awesome. I had a situation where - for some reason - a spring boot app would run in intellij CE but not eclipse on mac, and in eclipse but not intellij CE on win10. Being able to work in one IDE on two platforms is an advantage. – kometen May 25 '18 at 13:22

Clean solution for all JDKs >= 9

57

You need to add two dependencies to your build

- the jaxb-api
- a jaxb implementation

As an implementation I chose to use the reference implementation by glassfish to get rid of old com.sun classes / libraries. So as a result I added in my maven build

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>

<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
  <version>2.3.1</version>
</dependency>
```

Note that from version 2.3.1 you don't need to add the javax.activation any longer. (see https://github.com/eclipse-ee4j/jaxb-ri/issues/1222)

edited Sep 2 at 17:07
TuGordoBello
3,707 ● 7 ● 34 ● 57

answered Oct 18 '18 at 10:37
Sebastian Thees
1,574 ● 1 ● 9 ● 17

@k.liakos I am not sure. The jaxb-runtime jar and the api-jar do not share the same classes/packages. I guess it depends on your code. If your code does not use the classes from the package 'javax.xml.bind' then you probably don't need it. The topic of this thread is that 'javax/xml/bind/JAXBException' cannot be found; this class is only in the jaxb-api. – Sebastian Thees Nov 2 '18 at 9:30

2 Works perfectly with multi-module project in java 12. – Heril Muratovic Jun 24 '19 at 12:23

42

This worked for me:

```
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.7.0</version>
</dependency>
```

Update

As @Jasper suggested, in order to avoid depending on the entire EclipseLink library, you can also just depend on EclipseLink MOXy:

Maven

```
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>org.eclipse.persistence.moxy</artifactId>
    <version>2.7.3</version>
</dependency>
```

Gradle

```
compile group: 'org.eclipse.persistence', name: 'org.eclipse.persistence.moxy',
version: '2.7.3'
```

As dependencies for my Java 8 app, which produces a *.jar which can be run by both JRE 8 or JRE 9 with no additional arguments.

In addition, this needs to be executed somewhere before JAXB API will be used:

```
System.setProperty("javax.xml.bind.JAXBContextFactory",
"org.eclipse.persistence.jaxb.JAXBContextFactory");
```

edited Oct 1 '18 at 1:10

answered Sep 27 '17 at 18:40
Mikhail Kholodkov
15k ● 13 ● 47 ● 65

5 adding `org.eclipse.persistence:eclipselink` just to get JAXB APIs is a very heavy-weight dependency, unless you are already using eclipselink? – Andy Guibert Sep 27 '17 at 18:59

4 Yes it is heavy (~9mb) and yes I've been using that already. I've mentioned that this is simply an alternative workaround for those who, maybe temporary, will have to use both 8 and 9 JREs for the same jar/war without providing command-line arguments. – Mikhail Kholodkov Sep 27 '17 at 19:33

2 for the sake of interop between JDK 8 & 9, I would recommend using the `-XX:+IgnoreUnrecognizedVMOptions` command line option (updated my answer with details) – Andy Guibert Sep 27 '17 at 19:55

System.setProperty("javax.xml.bind.JAXBContextFactory", "org.eclipse.persistence.jaxb.JAXBContextFactory"); does not work for me – David Brossard Feb 16 '18 at 18:00

1 To avoid depending on the entire EclipseLink library, you can also just depend on EclipseLink MOXy: groupId `org.eclipse.persistence`, artifactId `org.eclipse.persistence.moxy`. – Jesper Sep 30 '18 at 13:14

36

it´s because java version if you are using jdk 9 or a later version just add this to your pom

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.0</version>
</dependency>
```

edited Jun 23 '19 at 22:36
Jared Burrows
49.1k ● 21 ● 136 ● 174

answered Oct 29 '18 at 6:34
Cesar Rodriguez T
559 ● 4 ● 7

1 I run into this all the time with the Spring Boot guides... Thanks a ton. – masterxilo Nov 7 '18 at 15:39

2 @Cesar Rodriguez T, I tried this with a swagger example and compiling worked but running gave errors. I used the selected answer which included more dependencies and that worked. – PatS Nov 13 '18 at 13:24

On pom.xml file of your project – Cesar Rodriguez T Nov 13 '19 at 4:34

23

To solve this, I have imported some JAR files in my project:

- javax.activation-1.2.0.jar

http://search.maven.org/remotecontent?

http://search.maven.org/remotecontent?filepath=javax/xml/bind/jaxb-api/2.3.0/jaxb-api-2.3.0.jar

- jaxb-core-2.3.0.jar

http://search.maven.org/remotecontent?filepath=com/sun/xml/bind/jaxb-core/2.3.0/jaxb-core-2.3.0.jar

- jaxb-impl-2.3.0.jar

http://search.maven.org/remotecontent?filepath=com/sun/xml/bind/jaxb-impl/2.3.0/jaxb-impl-2.3.0.jar

1. Download above files and copy them into libs folder in the project
2. Add the imported JAR files in Java Build Path

edited Apr 23 '18 at 2:51
Cœur
30.6k ● 16 ● 168 ● 219

answered Jan 23 '18 at 14:53
Fábio Nascimento
2,112 ● 1 ● 16 ● 25

4 Note that the `com.sun.xml.bind` artifacts are old and provided only for backward compatibility. You should use the equivalent `org.glassfish.jaxb` artifacts instead, as mentioned in some of the other answers. – Jesper Sep 30 '18 at 12:59

This didn't work for me. It threw an error, and said that it couldn't find a particular class. – RamenChef Oct 15 '18 at 2:15

Worked for me when I put them in tomcat9/lib folder under Mint 19.2 (Ubuntu 18.04 base), when deploying a Grails 3.4.10 application. – Mohamad Fakih Dec 5 '19 at 20:53

18

At the time of compilation as well as run time, add the switch `--add-modules java.xml.bind`

```
javac --add-modules java.xml.bind <java file name>

java --add-modules java.xml.bind <class file>
```

A good introduction of the `JDK 9` modules can also be found at : https://www.youtube.com/watch?v=KZfbRuvv5qc

edited Nov 3 '17 at 5:20
Tarunn
966 ● 2 ● 19 ● 40

answered Apr 25 '17 at 4:35
Pallavi Sonal
2,671 ● 12 ● 18

This worked for me. Adding only jaxb-api wasn't enough.

```
            <version>${jaxb-api.version}</version>
        </dependency>
        <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>${jaxb-api.version}</version>
        </dependency>
        <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-core</artifactId>
            <version>${jaxb-api.version}</version>
        </dependency>
```

answered Dec 4 '17 at 18:24
Mr Jedi
27.2k ● 7 ● 23 ● 36

And what was jaxb-api.version set to? – MiguelMunoz Dec 11 '17 at 3:42

@MiguelMunoz I used 2.2.7 – Mr Jedi Dec 11 '17 at 11:41

2 Note that the `com.sun.xml.bind` artifacts are old and provided only for backward compatibility. You should use the equivalent `org.glassfish.jaxb` artifacts instead, as mentioned in some of the other answers. – Jesper Sep 30 '18 at 12:58

11

Go to Your Build.gradle and add below dependencies for both Java 9 or Java 10.

```
sourceCompatibility = 10 // You can also decrease your souce compatibility to 1.8

//java 9+ does not have Jax B Dependents

    compile group: 'javax.xml.bind', name: 'jaxb-api', version: '2.3.0'
    compile group: 'com.sun.xml.bind', name: 'jaxb-core', version: '2.3.0'
    compile group: 'com.sun.xml.bind', name: 'jaxb-impl', version: '2.3.0'
    compile group: 'javax.activation', name: 'activation', version: '1.1.1'
```

answered Jul 8 '18 at 18:51
Kumar Abhishek
2,306 ● 22 ● 26

11

You can use `--add-modules=java.xml.bind` JVM option to add xml bind module to JVM run-time environment.

Eg: `java --add-modules=java.xml.bind XmlTestClass`

edited Sep 19 '18 at 22:48
Willi Mentzel

answered Oct 10 '17 at 19:27
Jayesh Jayanthivasan

11

Update April 2019

Changelong for JAXB releases is at https://javaee.github.io/jaxb-v2/doc/user-guide/ch02.html

excerpts:

```
    4.1. Changes between 2.3.0.1 and 2.4.0

         JAXB RI is now JPMS modularized:

            All modules have native module descriptor.

            Removed jaxb-core module, which caused split package issue on JPMS.

            RI binary bundle now has single jar per dependency instead of shaded fat
jars.

            Removed runtime class weaving optimization.

    4.2. Changes between 2.3.0 and 2.3.0.1

          Removed legacy technology dependencies:

            com.sun.xml.bind:jaxb1-impl

            net.java.dev.msv:msv-core

            net.java.dev.msv:xsdlib

            com.sun.xml.bind.jaxb:isorelax

    4.3. Changes between 2.2.11 and 2.3.0

          Adopt Java SE 9:

            JAXB api can now be loaded as a module.

            JAXB RI is able to run on Java SE 9 from the classpath.

            Addes support for java.util.ServiceLoader mechanism.

            Security fixes
```

Authoritative link is at https://github.com/eclipse-ee4j/jaxb-ri#maven-artifacts

> Maven coordinates for JAXB artifacts
>
> jakarta.xml.bind:jakarta.xml.bind-api: API classes for JAXB. Required to compile against JAXB.
>
> org.glassfish.jaxb:jaxb-runtime: Implementation of JAXB, runtime used for serialization and deserialization java objects to/from xml.
>
> JAXB fat-jar bundles:

> In contrast to org.glassfish.jaxb artifacts, these jars have all dependency classes included inside. These artifacts does not contain JPMS module descriptors. In Maven projects org.glassfish.jaxb artifacts are supposed to be used instead.

org.glassfish.jaxb:jaxb-runtime:jar:2.3.2 pulls in:

```
[INFO] +- org.glassfish.jaxb:jaxb-runtime:jar:2.3.2:compile
[INFO] |  +- jakarta.xml.bind:jakarta.xml.bind-api:jar:2.3.2:compile
[INFO] |  +- org.glassfish.jaxb:txw2:jar:2.3.2:compile
[INFO] |  +- com.sun.istack:istack-commons-runtime:jar:3.0.8:compile
[INFO] |  +- org.jvnet.staxex:stax-ex:jar:1.8.1:compile
[INFO] |  +- com.sun.xml.fastinfoset:FastInfoset:jar:1.2.16:compile
[INFO] |  \- jakarta.activation:jakarta.activation-api:jar:1.2.1:compile
```

Original Answer

Following Which artifacts should I use for JAXB RI in my Maven project? in Maven, you can use a profile like:

```
<profile>
    <id>java-9</id>
    <activation>
        <jdk>9</jdk>
    </activation>
    <dependencies>
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
            <version>2.3.0</version>
        </dependency>
        <dependency>
            <groupId>javax.activation</groupId>
            <artifactId>activation</artifactId>
            <version>1.1.1</version>
        </dependency>
    </dependencies>
</profile>
```

Dependency tree shows:

```
[INFO] +- org.glassfish.jaxb:jaxb-runtime:jar:2.3.0:compile
[INFO] |  +- org.glassfish.jaxb:jaxb-core:jar:2.3.0:compile
[INFO] |  |  +- javax.xml.bind:jaxb-api:jar:2.3.0:compile
[INFO] |  |  +- org.glassfish.jaxb:txw2:jar:2.3.0:compile
[INFO] |  |  \- com.sun.istack:istack-commons-runtime:jar:3.0.5:compile
[INFO] |  +- org.jvnet.staxex:stax-ex:jar:1.7.8:compile
[INFO] |  \- com.sun.xml.fastinfoset:FastInfoset:jar:1.2.13:compile
[INFO] \- javax.activation:activation:jar:1.1.1:compile
```

To use this in Eclipse, say Oxygen.3a Release (4.7.3a) or later, Ctrl-Alt-P, or right-click on the

edited Apr 20 '19 at 7:59 answered May 6 '18 at 3:19
JasonPlutext
14k ● 3 ● 36 ● 74

Thanks for showing that a dependency for `javax.xml.bind` > `jaxb-api` that I have seen elsewhere is actually redundant. The glassfish dependency pulls it is. I just tried that, and it does indeed work. – Basil Bourque Jul 23 '18 at 5:42

9

I also stumpled accross the ClassNotFoundException:javax.xml.bind.DatatypeConverter using **Java 11** and

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

I tried all this stuff around adding javax.xml.bind:jaxb-api or spring boot jakarta.xml.bind-api .. I found a hint for fixes in jjwt version 0.10.0 .. but most importantly, the jjwt package is now split !

Thus, check this reference: https://github.com/jwtk/jjwt/issues/510

Simply, if you use

> Java11 and jjwt 0.9.x and you face the ClassNotFoundException:javax.xml.bind.DatatypeConverter issue,

go for

jjwt version 0.11.x, but use the splitted packages: https://github.com/jwtk/jjwt#install

You maven wont find a higher version for jjwt dependency, since they split the packages.

Cheers.

edited Jun 20 at 9:12 answered Apr 3 at 9:32
Community ♦ rico_s
1 ● 1 151 ● 1 ● 5

8

For Java Web Start Execution we can use Andy Guibert's suggestion like this:

```
<j2se version="1.6+"
      java-vm-args="-XX:+IgnoreUnrecognizedVMOptions --add-modules=java.se.ee"/>
```

Note the extra "=" in the --add-modules. See this OpenJDK Ticket or the last note in "Understanding Runtime Access Warnings" of the Java Platform, Standard Edition Oracle JDK 9 Migration Guide.

edited Sep 29 '17 at 11:01 answered Sep 29 '17 at 8:34
mvw
4,737 ● 1 ● 23 ● 31

8

add javax.xml.bind dependency in pom.xml

```
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.0</version>
    </dependency>
```

answered Oct 17 '18 at 19:02
malith vitha
335 ● 3 ● 4

8

Since JavaEE is now governed by https://jakarta.ee/, the new Maven coordinates as of 2.3.2 are:

https://eclipse-ee4j.github.io/jaxb-ri/#maven-artifacts

The first released jaxb.version is 2.3.2.

```
<properties>
  <jaxb.version>2.3.2</jaxb.version>
</properties>

<dependency>
  <groupId>jakarta.xml.bind</groupId>
  <artifactId>jakarta.xml.bind-api</artifactId>
  <version>${jaxb.version}</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>${jaxb.version}</version>
</dependency>
```

edited Dec 6 '19 at 10:28 answered Apr 11 '19 at 6:55
dschulten
2,142 ● 1 ● 20 ● 34

7

```
   <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.0</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>javax.activation</groupId>
    <artifactId>javax.activation-api</artifactId>
    <version>1.2.0</version>
</dependency>
```

answered Sep 15 '18 at 16:12
itsraghz
575 ● 7 ● 20

7

This solved my problems with dependencies running Apache Camel 2.24.1 on Java 12:

```
    <dependency>
        <groupId>javax.activation</groupId>
        <artifactId>activation</artifactId>
        <version>1.1.1</version>
    </dependency>

    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.1</version>
    </dependency>

    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-core</artifactId>
        <version>2.3.0.1</version>
    </dependency>

    <dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.3.0.1</version>
    </dependency>
```

answered Aug 25 '19 at 6:32
kachanov
2,378 ● 2 ● 18 ● 16

in my case i need to add tomcat dependency in pom file – GvSharma Sep 15 '19 at 17:04

6

I encountered the same issue using Spring Boot `2.0.5.RELEASE` on Java 11.

Adding `javax.xml.bind:jaxb-api:2.3.0` alone did not fix the problem. I also had to update Spring Boot to the latest Milestone `2.1.0.M2`, so I assume this will be fixed in the next official release.

answered Sep 29 '18 at 4:25
Javide
1,853 ● 2 ● 32 ● 50

This does not sound related to me. there are several solutions in this thread that work regardless of using spring boot 2. (I also use spring boot 2.0.5.RELEASE btw). Maybe in Spring 2.1.0.M2 there is already a jaxb runtime included. – Sebastian Thees Nov 2 '18 at 9:40

It seems that with Spring Boot 2.1.0.RELEASE, JAXB is not necessary anymore - github.com/spring-projects/spring-boot/releases – Burrich Nov 4 '18 at 14:53

5

You need to add JAX-B dependencies when using JDK 9+. For Android Studio user, you'll need to add this to your `build.gradle`'s `dependencies {}` block:

```
// Add missing dependencies for JDK 9+
if (JavaVersion.current().ordinal() >= JavaVersion.VERSION_1_9.ordinal()) {
    // If you're using @AutoValue or any libs that requires javax.annotation (like
Dagger)
    compileOnly 'com.github.pengrad:jdk9-deps:1.0'
    compileOnly 'javax.annotation:javax.annotation-api:1.3.2'

    // If you're using Kotlin
    kapt "com.sun.xml.bind:jaxb-core:2.3.0.1"
    kapt "javax.xml.bind:jaxb-api:2.3.1"
    kapt "com.sun.xml.bind:jaxb-impl:2.3.2"

    // If you're using Java
    annotationProcessor "com.sun.xml.bind:jaxb-core:2.3.0.1"
    annotationProcessor "javax.xml.bind:jaxb-api:2.3.1"

    testAnnotationProcessor "com.sun.xml.bind:jaxb-core:2.3.0.1"
    testAnnotationProcessor "javax.xml.bind:jaxb-api:2.3.1"
}
```

edited Mar 4 at 10:21
Malachiasz
6,502 ● 2 ● 29 ● 46

answered Feb 6 at 3:07
Hieu Rocker
840 ● 9 ● 17

I have modified your answer to work with Unit tests as well. – Malachiasz Mar 4 at 10:22

Not an answer, but an addendum: I got because running `groovysh` (Groovy 2.4.13) if JAVA_HOME points to a Java 9 installation (`java version "9.0.1"` to be precise) fails abysmally:

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
java.lang.reflect.InvocationTargetException
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native
Method)
        at
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.ja

        at
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccesso

        at java.base/java.lang.reflect.Method.invoke(Method.java:564)
        at org.codehaus.groovy.tools.GroovyStarter.rootLoader(GroovyStarter.java:107)
        at org.codehaus.groovy.tools.GroovyStarter.main(GroovyStarter.java:129)
Caused by: java.lang.NoClassDefFoundError: Unable to load class
groovy.xml.jaxb.JaxbGroovyMethods due to missing dependency javax/xml/bind/JAXBContext
        at org.codehaus.groovy.vmplugin.v5.Java5.configureClassNode(Java5.java:400)
        at org.codehaus.groovy.ast.ClassNode.lazyClassInit(ClassNode.java:277)
        at org.codehaus.groovy.ast.ClassNode.getMethods(ClassNode.java:397)
        ...
        ..
        .
        ..
        ...
        at org.codehaus.groovy.tools.shell.Groovysh.<init>(Groovysh.groovy:135)
        at
org.codehaus.groovy.vmplugin.v7.IndyInterface.selectMethod(IndyInterface.java:232)
        at org.codehaus.groovy.tools.shell.Main.<init>(Main.groovy:66)
        at
org.codehaus.groovy.vmplugin.v7.IndyInterface.selectMethod(IndyInterface.java:232)
        at org.codehaus.groovy.tools.shell.Main.main(Main.groovy:163)
... 6 more
```

The solution was to:

- Go to the JAXB Project at github.io (*"JAXB is licensed under a dual license - CDDL 1.1 and GPL 2.0 with Class-path Exception"*)
- Download `jaxb-ri-2.3.0.zip`
- Unzip wherever you put your java infrastructure files (in my case, `/usr/local/java/jaxb-ri/`). Other solution may exist (maybe via SDKMAN, I dunno)
- Make sure the jars in the lib subdirectory are on the `CLASSPATH`. I do it via a script started on bash startup, called `/etc/profile.d/java.sh`, where I added (among many other lines) the following loop:

Packed into a function...

```
function extend_qzminynshg {
   local BASE="/usr/local/java"
   for LIB in jaxb-api.jar  jaxb-core.jar  jaxb-impl.jar  jaxb-jxc.jar  jaxb-xjc.jar;
do
      local FQLIB="$BASE/jaxb-ri/lib/$LIB"
      if [[ -f $FQLIB ]]; then
         export CLASSPATH=$FQLIB:$CLASSPATH
      fi
```

```
extend_qzminynshg; unset extend_qzminynshg
```

And it works!

answered Jan 4 '18 at 20:46
David Tonhofer
10.5k ● 3 ● 43 ● 39

1 I don't understand the downvotzes. Apparently people want to faff around with command-line options instead of actually getting the jars? Suit yourself. – David Tonhofer Jan 20 '18 at 17:50

7 Java developers typically use build tools like Gradle or Maven to manage dependencies rather than manually downloading jars. That is probably the reason for the down votes. – Joshua Davis Mar 24 '18 at 13:55

3

You only need 1 dependency:

```
dependencies {
    implementation ("jakarta.xml.bind:jakarta.xml.bind-api:2.3.2")
```

answered Feb 18 at 11:13
Dmitry Kaltovich
829 ● 6 ● 12

3

As the official documentation states:

> When upgrading you may face the following:
>
> `java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException`
>
> Hibernate typically requires JAXB that's no longer provided by default. You can add the java.xml.bind module to restore this functionality with Java9 or Java10 (even if the module is deprecated).
>
> As of Java11, the module is not available so your only option is to add the JAXB RI (you can do that as of Java9 in place of adding the java.xml.bind module:

Maven

```
<dependency>
```

Gradle (build.gradle.kts):

```
implementation("org.glassfish.jaxb:jaxb-runtime")
```

Gradle (build.gradle)

```
implementation 'org.glassfish.jaxb:jaxb-runtime'
```

If you rather specify a specific version, take a look here: https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime

edited Sep 7 at 14:35

answered Jul 11 at 15:40
Willi Mentzel
19.5k ● 12 ● 85 ● 97

2

OK, I have been having the same kind of issue, but I was using Java 8, and kept getting this error, I tried most of the solutions. but it turns out that my maven was still pointing to java 9 even-though I set the global Java version to 8, as soon as I fixed that it all worked.

For anybody who might have this kind of problem, check out How to fix Maven to use default Java

edited Apr 8 '19 at 12:03
Samuel Liew ♦
65.4k ● 41 ● 133 ● 222

answered Feb 22 '18 at 11:39
Ipkiss
535 ● 1 ● 5 ● 15

2

Old answer "Problem resolved by switching to amazoncorretto" News answer: I used corretto latest , but is similar jdk 1.8. so anyway we need add dependencies manually

edited Apr 23 '19 at 10:53

answered Apr 22 '19 at 7:48
Armen Arzumanyan
1,469 ● 1 ● 22 ● 50

2 The Amazon Corretto distribution for JDK 11 does not provide javax.xml.bind classes. If the problem was resolved after switching to Correto, it was because you downgraded to JDK 8. – Andy Guibert Apr 22 '19 at 13:38

strange, i will check, in the docker i used correto latest – Armen Arzumanyan Apr 22 '19 at 18:34

Yea, `amazoncorretto:latest` currently gives JDK 8, not 11. Many Docker images are still based on JDK 8, precisely because of the compatibility issues caused by the API removal between JDK 8 --> 11 – Andy Guibert Apr 22 '19 at 19:26

2

application in Java 8, 11, and 12 JREs.

```
        <!-- replace dependencies that have been removed from JRE's starting with Java
v11 -->
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.2.8</version>
        </dependency>
        <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-core</artifactId>
            <version>2.2.8-b01</version>
        </dependency>
        <dependency>
            <groupId>com.sun.xml.bind</groupId>
            <artifactId>jaxb-impl</artifactId>
            <version>2.2.8-b01</version>
        </dependency>
        <!-- end replace dependencies that have been removed from JRE's starting with
Java v11 -->
```

answered Jun 8 '19 at 20:01
Chris
431 ●4 ●6

2

For me in Java 11 and gradle this is what worked out:

```
plugins {
      id 'java'
}

dependencies {
      runtimeOnly 'javax.xml.bind:jaxb-api:2.3.1'
}
```

answered Aug 22 '19 at 7:08
silver_mx
576 ●6 ●12

where exactly do we put that? – nyxee Jan 8 at 9:41

In your build.gradle file if you are using gradle. – silver_mx Jan 9 at 10:10

1

You need to add jaxb dependancies to maven. The glassfish implementation version 2.3.2 is perfectly compatible with new jakarta EE jaxb api version 2.3.2.

```
    <version>2.3.2</version>
</dependency>

<!-- Runtime -->
<dependency>
    <groupId>org.glassfish.jaxb</groupId>
    <artifactId>jaxb-runtime</artifactId>
    <version>2.3.2</version>
</dependency>
```

answered Aug 24 '19 at 14:23
krishna Telgave
157 ● 1 ● 11

1

i had similar issues after upgrading my project to java 11, then what fixed it was upgrading to spring boot 2.1.1 which apparently has support for java 11, this helped

answered Oct 19 '19 at 10:44
FreakAtNs
105 ● 9

please consider adding at least the part of the solution in your answer since link-only answers will become invalid if the url changes in the future. – yukashima huksay Oct 19 '19 at 11:13

1

I encountered this issue when working on a **Java Project** in **Debian 10**.

Each time I start the appliction it throws the error in the log file:

> java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException

Here's how I solved it:

The issue is often caused when **JAXB** library (Java Architecture for XML Binding) is missing in the classpath. JAXB is included in Java SE 10 or older, but it is removed from Java SE from Java 11 or newer –moved to Java EE under Jakarta EE project.

So, I checked my Java version using:

```
java --version
```

And it gave me this output

So I was encountering the **JAXBException** error because I was using **Java 11**, which does not have the JAXB library (Java Architecture for XML Binding) is missing in the classpath. JAXB is included in it.

To fix the issue I had to add the **JAXB API** library to the **lib** (`/opt/tomcat/lib`) directory of my tomcat installation:

```
sudo wget https://repo1.maven.org/maven2/javax/xml/bind/jaxb-api/2.4.0-
b180830.0359/jaxb-api-2.4.0-b180830.0359.jar
```

Then I renamed it from `jaxb-api-2.4.0-b180830.0359.jar` to `jaxb-api.jar`:

```
sudo mv jaxb-api-2.4.0-b180830.0359.jar jaxb-api.jar
```

Note: Ensure that you change the permission allow tomcat access the file and also change the ownership to `tomcat`:

```
sudo chown -R tomcat:tomcat /opt/tomcat
sudo chmod -R 777 /opt/tomcat/
```

And then I restarted the tomcat server:

```
sudo systemctl restart tomcat
```

Resources: [Solved] java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException

That's all.

I hope this helps

edited 2 days ago

answered Oct 6 at 8:45
Promise Preston
4,921 ● 3 ● 21 ● 46

1 2 Next

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.