Difference Between Java 6 and Java 8

Features of java 8

www.javabrahman.com/java-8/tutorials‎

* forEach() method in Iterable interface.
* default and static methods in Interfaces.
* Functional Interfaces and Lambda Expressions.
* Java Stream API for Bulk Data Operations on Collections.
* Java Time API.
* Collection API improvements.
* Concurrency API improvements.
* Java IO improvements.

Features of java 6

www.roseindia.com

**Changes in jar and zip**Jar and zip support has been enhanced in JDK 6. In this, two new compressed streams have been added. These are java.util.zip.DeflaterInputStream and java.util.zip.InflaterOutputStream.

**Collections Framework Enhancement**In Collection framework, we are able to improve the performance hashing function that is used by java.util.HashMap. It provides some new Collection interfaces also.

**Scripting for the Java Platform**By using this Features developers integrate Java technology and scripting languages by defining a standard framework and application programming interface (API)

**Changes in I/O**This is a new feature added in Java SE 6, which has the ability to read text from a terminal without having it echo on the screen through java.io.Console.

\*\*\*\*to upgrade the existing standalone java application from **Java 1.6** to **Java 1.8**.

[www.stackoverflow.com/www.reddit.com](http://www.stackoverflow.com/www.reddit.com)

Technically, Java is completely backwards-compatible.

first we need to identify whether the used Tools/Frameworks are compatible with Java 1.8.

first, update the project's JDK and rebuild it targeting the new bytecode's version. There's a chance you'll have to upgrade both Ant and (if that's your IDE of choice) Eclipse.

Second, you'll have to check for compilation errors, which will most likely lead to update libraries conditionally to get them fixed. With those solved, we MUST run app and see if it's running as intended; remember that compilation problems are just the top of the iceberg when the subject are dependencies.

Carefully check the app's logs looking for exceptions of any kind but mainly the ones related to class loading exceptions such as ClassCastException, ClassNotFoundException, NoClassDefFoundException, UnsatisfiedLinkError and others. If any apear, we'll have to pinpoint one by one and search for the specific solution of the specific troublemaker library.

🡨-------------------------------------------------------------------------------------------------------------🡪

1. To build an AEM project with Java8 as runtime environment, following configuration is required in pom.xml. The Compiler Plugin is added to the parent POM, it is used to compile the sources of your project.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.1</version>  <inherited>true</inherited>  <configuration>  <source>1.8</source>  <target>1.8</target>  </configuration>  </plugin> |

Default source and target setting is 1.5. If you want to change it to work with Java 8, you should set the source and target to 1.8.

2. If you want to use scr annotations in your project, you have to use maven-scr-plugin. And its version >= 1.17.0 for Java8

|  |  |
| --- | --- |
| 1  2  3  4  5 | <plugin>    <groupId>org.apache.felix</groupId>    <artifactId>maven-scr-plugin</artifactId>    <version>1.17.0</version>  </plugin> |

3. For SCR annotations, following dependency should be added in the project with the version >= 1.9.8 for Java8

|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>org.apache.felix</groupId>      <artifactId>org.apache.felix.scr.annotations</artifactId>      <version>1.9.8</version>  </dependency> |

🡨--------------------------------------------------------------------------------------------------------🡪

1. / importing StringJoiner class
2. import java.util.StringJoiner;
3. public class StringJoinerExample {
4. public static void main(String[] args) {
5. StringJoiner joinNames = new StringJoiner(","); // passing comma(,) as delimiter
7. // Adding values to StringJoiner
8. joinNames.add("Rahul");
9. joinNames.add("Raju");
10. joinNames.add("Peter");
11. joinNames.add("Raheem");
13. System.out.println(joinNames);
14. }
15. }

Output:

Rahul,Raju,Peter,Raheem