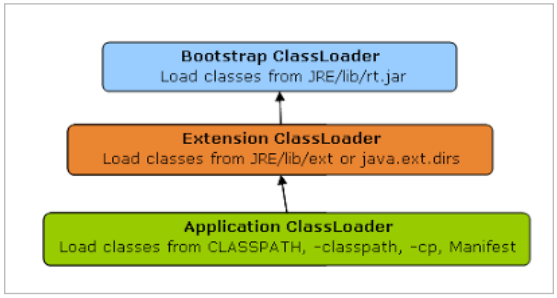
SATURDAY, DECEMBER 8, 2012

How ClassLoader Works in Java

Java class loaders are used to load classes at runtime. ClassLoader in Java works on three principle: `delegation`, `visibility` and `uniqueness`. Delegation principle forward request of class loading to parent class loader and only loads the class, if parent is not able to find or load class. Visibility principle allows child class loader to see all the classes loaded by parent ClassLoader, but parent class loader can not see classes loaded by child. Uniqueness principle allows to load a class exactly once, which is basically achieved by delegation and ensures that child ClassLoader doesn't reload the class already loaded by parent. Correct understanding of class loader is must to resolve issues like NoClassDefFoundError in Java and java.lang.ClassNotFoundException, which are related to class loading. ClassLoader is also an important topic in advanced Java Interviews, where good knowledge of working of Java ClassLoader and How classpath works in Java is expected from Java programmer. I have always seen questions like, **Can one class be loaded by two different ClassLoader in Java** on various Java Interviews. In this Java programming tutorial, we will learn what is ClassLoader in Java, How ClassLoader works in Java and some specifics about Java ClassLoader.

What is ClassLoader in Java

ClassLoader in Java is a class which is used to load class files in Java. Java code is compiled into class file by `javac` compiler and JVM executes Java program, by executing byte codes written in class file. ClassLoader is responsible for loading class files from file system, network or any other source. There are three default class loader used in Java, **Bootstrap** , **Extension** and **System or Application class loader**. Every class loader has a predefined location, from where they loads class files. Bootstrap ClassLoader is responsible for loading standard JDK class files from `rt.jar` and it is parent of all class loaders in Java. Bootstrap class loader don't have any parents, if you call `String.class.getClassLoader()` it will return `null` and any code based on that may throw NullPointerException in Java. Bootstrap class loader is also known as **Primordial ClassLoader** in Java. Extension ClassLoader delegates class loading request to its parent, `Bootstrap` and if unsuccessful, loads class form `jre/lib/ext` directory or any other directory pointed by `java.ext.dirs` system property. Extension ClassLoader in JVM is implemented by `sun.misc.Launcher$ExtClassLoader`. Third default class loader used by JVM to load Java classes is called System or Application class loader and it is responsible for loading application specific classes from CLASSPATH environment variable, `-classpath` or `-cp` command line option, `Class-Path` attribute of Manifest file inside JAR. Application class loader is a child of Extension ClassLoader and its implemented by `sun.misc.Launcher$AppClassLoader` class. Also, except Bootstrap class loader, which is implemented in native language mostly in C, all Java class loaders are implemented using `java.lang.ClassLoader`.



In short here is the location from which Bootstrap, Extension and Application ClassLoader load Class files.

1) Bootstrap ClassLoader - JRE/lib/rt.jar

2) Extension ClassLoader - JRE/lib/ext or any directory denoted by `java.ext.dirs`

3) Application ClassLoader - CLASSPATH environment variable, `-classpath` or `-cp` option, Class-Path attribute of Manifest inside JAR file.

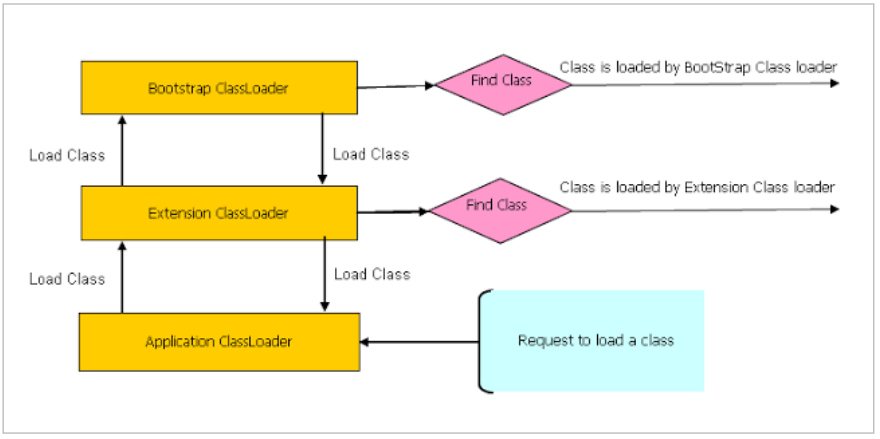
How ClassLoader works in Java

As I explained earlier Java ClassLoader works in three principles : `delegation`, `visibility` and `uniqueness`. In this section we will see those rules in detail and understand working of Java ClassLoader with example. By the way here is a diagram which explains How ClassLoader load class in Java using delegation.

Recommended Books for Java Developer
9 Must Read Books for Java Developers
3 Good Books to Learn Java 8
6 Books to Learn and Master Programming
2 Books to Prepare Oracle Java Certification
5 Books to Learn Spring Framework for Java Programmers
5 Books to Learn Design Patterns and OOP
5 Free Scala Programming Books PDF
5 Books to Learn Hibernate for Java Developers
Top 5 Java Performance Tuning Books
Top 5 jQuery Books for Java Web Developer
5 Java Unit Testing Book Every Developer Should Read

Followers
Join this site
with Google Friend Connect
Members (3172) More »
Already a member? Sign in

Blog Archive
► 2015 (55)
► 2014 (106)
► 2013 (136)
▼ 2012 (217)
▼ December (52)
Top 10 Oracle Interview Question and Answer - Dat...
How to compare Arrays in Java – Equals vs deepEqua...
How to append text into File in Java – FileWriter ...
Difference between equals method and "==" operator...
How to add, subtract days, months, years, hours fr...
Difference between Primary key vs Foreign key in t...
How to fix java.lang.classcastexception cannot be ...
Difference between Class and Object in Java and OO...
How to attach source in eclipse for Jars, debuggin...
How to find middle element of LinkedList in Java i...
Recursion in Java with example – Programming Techn...
What is Referential Integrity in Database or SQL -...
How to create thread safe Singleton in Java - Java...
Oracle 10g Pagination Query - SQL Example for Java...
How to convert milliseconds to Date in Java - tuto...
3 Example to print array values in Java - toString...
How to check if a number is a palindrome or not in...
Inner class and nested Static Class in Java with



Delegation principles

As discussed on when a class is loaded and initialized in Java, a class is loaded in Java, when its needed. Suppose you have an application specific class called Abc.class, first request of loading this class will come to Application ClassLoader which will delegate to its parent Extension ClassLoader which further delegates to Primordial or Bootstrap class loader. Primordial will look for that class in rt.jar and since that class is not there, request comes to Extension class loader which looks on jre/lib/ext directory and tries to locate this class there, if class is found there than Extension class loader will load that class and Application class loader will never load that class but if its not loaded by extension class-loader than Application class loader loads it from Classpath in Java. Remember Classpath is used to load class files while PATH is used to locate executable like javac or java command.

Visibility Principle

According to visibility principle, Child ClassLoader can see class loaded by Parent ClassLoader but vice-versa is not true. Which mean if class Abc is loaded by Application class loader than trying to load class ABC explicitly using extension ClassLoader will throw either java.lang.ClassNotFoundException. as shown in below Example

```
package test;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Java program to demonstrate How ClassLoader works in Java,
 * in particular about visibility principle of ClassLoader.
 *
 * @author Javin Paul
 */

public class ClassLoaderTest {

    public static void main(String args[]) {
        try {
            //printing ClassLoader of this class
            System.out.println("ClassLoaderTest.getClass().getClassLoader() : "
                                + ClassLoaderTest.class.getClassLoader());



            //trying to explicitly load this class again using Extension class loader
            Class.forName("test.ClassLoaderTest", true
                          ,  ClassLoaderTest.class.getClassLoader().getParent());
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(ClassLoaderTest.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

Output:

```
ClassLoaderTest.getClass().getClassLoader() : sun.misc.Launcher$AppClassLoader@601bb1
16/08/2012 2:43:48 AM test.ClassLoaderTest main
SEVERE: null
```

E...
How to add, modify and drop column with default va...
What is Type Casting in Java - Casting one Class ...
How to create auto incremented identity column in ...
How to parse or convert String to long in Java - 4...
How to fix java.io.NotSerializableException: org.a...
How to initialize ArrayList with Array in Java | C...
How to Create and Evaluate XPath Expression in Jav...
How to Split String in SQL Server and Sybase
How to get current date, month, year and day of we...
How to create and modify Properties file form Java...
How to Count Occurrences of a Character in String ...
How to read input from command line in Java using ...
How to remove duplicates elements from ArrayList i...
How to find second highest or maximum salary of Em...
XPath Tutorial - How to select elements in XPATH b...
Constructor Chaining in Java - Calling one constru...
How to escape String literal in Java using Eclipse...
Invalid initial and maximum heap size in JVM - How...
SQL query to copy, duplicate or backup table in My...
How to sort HashMap by key and value in Java - Has...
Does Java pass by value or pass by reference - Int...
Mapping network drive in Windows XP and 7 – net us...
Inversion of Control and Dependency Injection desi...
Java Error : 'javac' is not recognized as an inter...
How to find duplicate records in a table on databa...
java.lang.ClassNotFoundException: oracle.jdbc.driv...
What is Constructor in Java with Example – Constru...
How to comment uncomment single line and block of ...
Union and Intersection of two Set in Java - Google...
How ClassLoader Works in Java
Top 10 JDBC Interview questions answers for Java p...
What is Object in Java Programming and OOPS - Exam...
BlockingQueue in Java – ArrayBlockingQueue vs Link...
Why getter and setter are better than public field...

► November (8)
► October (14)
► September (8)
► August (9)
► July (9)
► June (12)
► May (10)
► April (14)
► March (28)

```
java.lang.ClassNotFoundException: test.ClassLoaderTest
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:190)
        at sun.misc.Launcher$ExtClassLoader.findClass(Launcher.java:229)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:306)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:247)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:247)
        at test.ClassLoaderTest.main(ClassLoaderTest.java:29)
```

Uniqueness Principle

According to this principle a class loaded by Parent should not be loaded by Child ClassLoader again. Though its completely possible to write class loader which violates Delegation and Uniqueness principles and loads class by itself, its not something which is beneficial. You should follow all class loader principle while writing your own ClassLoader.

How to load class explicitly in Java

Java provides API to explicitly load a class by `Class.forName(classname)` and `Class.forName(classname, initialized, classloader)`, remember JDBC code which is used to load JDBC drives we have seen in Java program to Connect Oracle database. As shown in above example you can pass name of ClassLoader which should be used to load that particular class along with binary name of class. Class is loaded by calling `loadClass()` method of `java.lang.ClassLoader` class which calls `findClass()` method to locate bytecodes for corresponding class. In this example Extension ClassLoader uses `java.net.URLClassLoader` which search for class files and resources in JAR and directories. any search path which is ended using "/" is considered directory. If `findClass()` does not found the class than it throws java.lang.ClassNotFoundException and if it finds it calls `defineClass()` to convert bytecodes into a .class instance which is returned to the caller.

Where to use ClassLoader in Java

ClassLoader in Java is a powerful concept and used at many places. One of the *popular example of ClassLoader* is `AppletClassLoader` which is used to load class by `Applet`, since `Applets` are mostly loaded from internet rather than local file system, By using separate ClassLoader you can also loads same class from multiple sources and they will be treated as different class in JVM. J2EE uses multiple class loaders to load class from different location like classes from WAR file will be loaded by Web-app ClassLoader while classes bundled in EJB-JAR is loaded by another class loader. Some web server also supports hot deploy functionality which is implemented using ClassLoader. You can also use ClassLoader to load classes from database or any other persistent store.

That's all about **What is ClassLoader in Java** and **How ClassLoader works in Java**. We have seen delegation, visibility and uniqueness principles which is quite important to debug or troubleshoot any ClassLoader related issues in Java. In summary knowledge of How ClassLoader works in Java is must for any Java developer or architect to design Java application and packaging.

Other **Java Tutorials** from Javarevisited you may like

How HashMap works in Java

How volatile variable works in Java

Why multiple Inheritance is not supported in Java

Top 10 JDBC best practices for Java programmer

10 OOPS design principle Java programmer should know

Posted by Javin Paul at 3:55 AM

Labels: core java , core java interview question , programming
Location: United States

7 comments :

Ivan Bezyazychnyy said...

Hi!

References

Java API documentation JDK 6

Spring framework doc

Struts

JDK 7 API

MySQL

Linux

Eclipse

jQuery

Thanks for the article. I have a question about the sentence: "Java provides API to explicitly load a class by Class.forName(classname) and Class.forName(classname, initialized, classloader), remember JDBC code which is used to load JDBC drives we have seen in Java program to Connect Oracle database."
I cannot find the explicit loading of class in that article. Do I misunderstand something?

December 10, 2012 at 12:25 PM

sajjad paracha said...

Good Article , I would like to add one more exception/error caused by ClassLoading problems.
NoSuchMethodError , this happens mostly in web applications or J2EE applications where you have the latest jar file of a utility such as apache common utilities in lib folder of your application but the server has an older version of same jar file.When your application needs the class it is is already loaded by the parent class loader (your server's class loader) but when the application wants to call a method which is available in new jar but not in the old jar loaded by parent class loader you will face NOSuchMethod Error exception. i recently went through all these problems trying to deploy a JPA2.0 application on weblogic10.3.3 server (which is JPA1.0 compliant which means you will have old jar files loaded in your praent class laoder) and only after understanding the ClasLoader working I was able to resolve all the issues.
In My below blog piost I posted the solution as well
http://javaiscoool.blogspot.com/2012/12/deploy-jpa20-application-on-weblogic1033.html

December 24, 2012 at 9:10 PM

Riddhi said...

is there any change in working of ClassLoader on Java 5, 6 or Java 7 ? I also come to know that certain method on JDK executed by immediate classloader rather than delegation e.g. DriverManager.getConnection() and Class.forName(), is that true, if yes what issue they cause and what precaution Java programmer should take ?

February 12, 2013 at 10:22 PM

Anonymous said...

I have just few questions regarding class loaders :

When to create your own ClassLoader in Java?
What security consideration is required while implmenting ClassLoader in Java?
Can a untrusted code e.g. from Applet contains a ClasSLoader? Can it load classes from filesystem or network at same time?
What kind of ClassLoaders are bundled in JDK?
Can a ClassLoader load classes compiled in different Java version than class loader itself?

February 12, 2013 at 10:59 PM

Anonymous said...

Can we load our own class using bootstrap class loaders?

April 17, 2013 at 8:39 PM

jeet said...

As you said in the delegation principle "first request of loading this class will come to Application ClassLoader".If the order of loading classes is bootstrap loaders, then extension class loader then application class loader then why does the request first go to Application class loader.Should't it start from the bootstrap loader.Why is it so?

September 4, 2013 at 9:23 PM

govind said...

I have one query regarding this sentence
"By using separate ClassLoader you can also loads same class from multiple sources and they will be treated as different class in JVM"

This means there will be two version of same class and anyone of them will be picked up during runtime ?

March 23, 2014 at 11:54 PM

Post a Comment

Enter your comment...

Comment as: Google Accour

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)