java.lang.ClassLoader类API学习

# ClassLoader

## 继承关系

public **abstract** class **ClassLoader** extends Object

java.lang.ClassLoader

Direct Known Subclasses:**SecureClassLoader**

## 功能介绍

### load classes

A class loader is an object that is responsible for **loading classes**. The class **ClassLoader** is an abstract class. Given **the binary name of a class**, a class loader should attempt to locate or generate data that constitutes a definition for the class. A typical strategy is to transform the name into a file name and then read a "class file" of that name from a file system.

### 获取对应的ClassLoader

Every **Class** object contains a **reference** to the **ClassLoader** that defined it.

每一个**Class对象**都拥有定义加载该类的加载器的引用。对应的就是Class中的getClass方法：**ClassLoader** **getClassLoader()**  Returns the class loader for the class.。

同样对于任何一个类都有getClass方法获取Class对象：new Object().getClass.getClassLoader()获取当前类的ClassLoader。

### 对于数组，不需要类加载器

array的ClassLoader就是其元素的ClassLoader，若是基本数据类型，则这个array没有ClassLoader

**Class** objects for **array** classes are not created by class loaders, but are created automatically as required by the Java runtime. The class loader for an array class, as returned by **Class.getClassLoader()** is the same as the class loader for its element type; if the element type is a primitive type, then the array class has no class loader.

Applications implement subclasses of ClassLoader in order to extend the manner in which the Java virtual machine dynamically loads classes.

Class loaders may typically be used by security managers to indicate security domains.

### 双亲委派模型

In environments in which **the delegation model** is not strictly hierarchical, class loaders need to be parallel capable, otherwise class loading can lead to deadlocks because the loader lock is held for the duration of the class loading process (see loadClass methods).

### 类文件的来源：

Normally, the **Java virtual machine** loads classes from **the local file system** in a platform-dependent manner. For example, on UNIX systems, the virtual machine loads classes from the directory defined by the **CLASSPATH** environment variable.

However, some classes may not **originate from** a file; they may originate from **other sources,** such as **the network**, or they could be constructed by an application. The method **defineClass** converts an array of bytes into an instance of class **Class**. Instances of this newly defined class can be created using **Class.newInstance**.

The methods and constructors of objects created by a class loader may reference other classes. To determine the class(es) referred to, the Java virtual machine invokes the **loadClass** method of the class loader that originally created the class.

For example, an application could create a network class loader to download class files from a server. Sample code might look like:

ClassLoader loader = new NetworkClassLoader(host, port);

Object main = loader.loadClass("Main", true).newInstance();

. . .

The network class loader subclass must define the methods **findClass** and **loadClassData** to load a class from the network. Once it has downloaded the bytes that make up the class, it should use the method **defineClass** to create **a class instance**. A sample implementation is:

class **NetworkClassLoader** extends ClassLoader {

String host;

int port;

//覆盖findClass方法，自定义类加载器

public Class **findClass**(String name) {

byte[] b = **loadClassData**(name);

return defineClass(name, b, 0, b.length);

}

private byte[] loadClassData(String name) {

// load the class data from the connection

. . .

}

}

## Binary names

Any class name provided as **a String parameter to methods** in **ClassLoader** must be **a binary name** as defined by The Java™ Language Specification.

Examples of valid class names include:

"java.lang.String"

"javax.swing.JSpinner$DefaultEditor"

"java.security.KeyStore$Builder$FileBuilder$1"

"java.net.URLClassLoader$3$1"

## 重要方法

**defineClass**

protected **Class<?>** **defineClass**(String name, byte[] b, int off, int len, ProtectionDomain protectionDomain)

Converts an array of bytes into an instance of class Class, with an optional ProtectionDomain.

**findClass**

protected Class<?> findClass(String name)

Finds the class with the specified binary name.

**loadClass**

Class<?> **loadClass**(String name)

Loads the class with the specified binary name.

protected **Class**<?> **loadClass**(String name, boolean resolve)

Loads the class with the specified binary name.