

Загрузка классов в Java. Теория

Java*



Одной из основных особенностей платформы Java является модель динамической загрузки классов, которая позволяет загружать исполняемый код в JRE не перезагружая основное приложение. Такая особенность широко используется в серверах приложений, получивших последнее время высокую популярность.

В статье рассмотрены базовые понятия, аспекты и принципы модели динамической загрузки кода. В следующей статье будет рассмотрена реализация собственного загрузчика классов, как основного механизма приложения с плагино-модульной архитектурой.

Введение

Любой класс (экземпляр класса `java.lang.Class` в среде и `.class` файл в файловой системе), используемый в среде исполнения был так или иначе загружен каким-либо загрузчиком в Java. Для того, чтобы получить загрузчик, которым был загружен класс `A`, необходимо воспользоваться методом `A.class.getClassLoader()`.

Классы загружаются по мере надобности, за небольшим исключением. Некоторые базовые классы из `rt.jar` (`java.lang.*` в частности) загружаются при старте приложения. Классы расширений (`$JAVA_HOME/lib/ext`), пользовательские и большинство системных классов загружаются по мере их использования.

Виды загрузчиков

Различают 3-и вида загрузчиков в Java. Это — базовый загрузчик (`bootstrap`), системный загрузчик (`System Classloader`), загрузчик расширений (`Extension Classloader`).

Bootstrap — реализован на уровне JVM и не имеет обратной связи со средой исполнения. Данным загрузчиком загружаются классы из директории `$JAVA_HOME/lib`. Т.е. всеми любимый `rt.jar` загружается именно базовым загрузчиком. Поэтому, попытка получения загрузчика у классов `java.*` всегда заканчивается `null`ом. Это объясняется тем, что все базовые классы загружены базовым загрузчиком, доступа к которому из управляемой среды нет.

Управлять загрузкой базовых классов можно с помощью ключа `-Xbootclasspath`, который позволяет переопределять наборы базовых классов.

System Classloader — системный загрузчик, реализованный уже на уровне JRE. В Sun JRE — это класс `sun.misc.Launcher$AppClassLoader`. Этим загрузчиком загружаются классы, пути к которым указаны в переменной окружения `CLASSPATH`.

Управлять загрузкой системных классов можно с помощью ключа `-classpath` или системной опцией `java.class.path`.

Extension Classloader — загрузчик расширений. Данный загрузчик загружает классы из директории `$JAVA_HOME/lib/ext`. В Sun JRE — это класс `sun.misc.Launcher$ExtClassLoader`.

Управлять загрузкой расширений можно с помощью системной опции `java.ext.dirs`.

Понятия

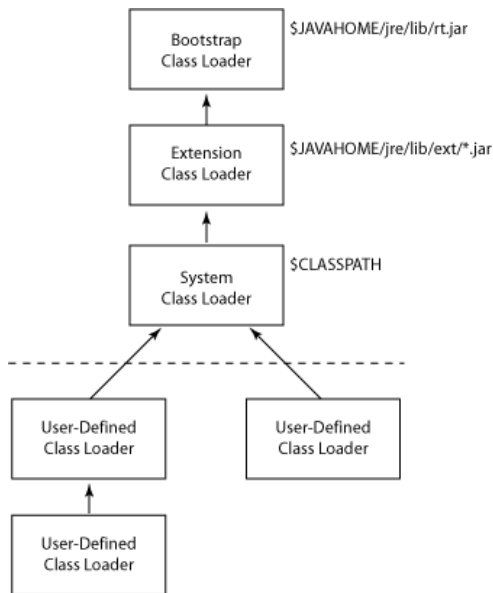
Различают текущий загрузчик (`Current Classloader`) и загрузчик контекста (`Context Classloader`).

Current Classloader — это загрузчик класса, код которого в данный момент выполняется. Текущий загрузчик используется по умолчанию для загрузки классов в процессе исполнения. В частности, при использовании метода `Class.forName("")/ClassLoader.loadClass("")` или при любой декларации класса, ранее не загруженного.

Context Classloader — загрузчик контекста текущего потока. Получить и установить данный загрузчик можно с помощью методов `Thread.getContextClassLoader()/Thread.setContextClassLoader()`. Загрузчик контекста устанавливается автоматически для каждого нового потока. При этом, используется загрузчик родительского потока.

Модель делегирования загрузки

Начиная с версии Java 2 Platform, Standard Edition, v1.2 загрузчики классов образуют иерархию. Корневым является базовый (у него предка нет). Все остальные загрузчики при инициализации инстанцируют ссылку на родительский загрузчик. Такая иерархия необходима для модели делегирования загрузки. В общем случае, иерархия выглядит следующим образом.



Право загрузки класса рекурсивно делегируется от самого нижнего загрузчика в иерархии к самому верхнему. Такой подход позволяет загружать классы тем загрузчиком, который максимально близко находится к базовому. Так достигается максимальная область видимости классов. Под областью видимости подразумевается следующее. Каждый загрузчик ведет учет классов, которые были им загружены. Множество этих классов и называется областью видимости.

Рассмотрим процесс загрузки более детально. Пусть в систем исполнении встретилась декларация переменной пользовательского класса `Student`.

- 1) Системный загрузчик попытается поискать в кеше класс `Student`.
 - _1.1) Если класс найден, загрузка окончена.
 - _1.2) Если класс не найден, загрузка делегируется загрузчику расширений.
- 2) Загрузчик расширений попытается поискать в кеше класс `Student`.
 - _2.1) Если класс найден, загрузка окончена.
 - _2.2) Если класс не найден, загрузка делегируется базовому загрузчику.
- 3) Базовый загрузчик попытается поискать в кеше класс `Student`.
 - _3.1) Если класс найден, загрузка окончена.
 - _3.2) Если класс не найден, базовый загрузчик попытается его загрузить.
 - _3.2.1) Если загрузка прошла успешно, она закончена ;)
 - _3.2.2) Иначе управление передается загрузчику расширений.
 - _3.3) Загрузчик расширений пытается загрузить класс.
 - _3.3.1) Если загрузка прошла успешно, она закончена ;)
 - _3.3.2) Иначе управление передается системному загрузчику.
 - _3.4) Системный загрузчик пытается загрузить класс.
 - _3.4.1) Если загрузка прошла успешно, она закончена ;)
 - _3.4.2) Иначе генерируется исключение `java.lang.ClassNotFoundException`.

Если в системе присутствуют пользовательские загрузчики, они должны

- а) расширять класс `java.lang.ClassLoader`;

б) поддерживать модель динамической загрузки.

Inside

Запустим простейшее приложение с ключем `-verbose:class`.

```
public class A { }

public class B extends A { }

public class C extends B { }

public class Main {

    public static void main(String args[]) {

        C c = new C();
        B b = new B();
        A a = new A();

    }

}
```

* This source code was highlighted with Source Code Highlighter.


Вывод показывает, что классы были загружены не в том порядке в котором были использованы. Это обусловлено наследованием.

[Loaded Main from file:/C:/devel/CL/bin/]

[Loaded A from file:/C:/devel/CL/bin/]

[Loaded B from file:/C:/devel/CL/bin/]

[Loaded C from file:/C:/devel/CL/bin/]

 java, classloading