Java -D 设置系统属性讲解

作者：shirenchuang
原文链接：https://ld246.com/article/1565861658906
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

版权声明：本文为博主原创文章，遵循CC 4.0 by-sa
版权协议，转载请附上原文出处链接和本声明。
本文链接: http://blog.shiyi.online/articles/2019/08/15/1565861655572.html

我们在用Java命令来启动一个项目的时候,会有很多可选的参数,比如-XX用来配置虚拟机的大小,今天们看一看 **-D** 这个选项的用法;

用法说明

-Dkey=value

在虚拟机的系统属性中设置属性名的键值对,运行在此虚拟机上的应用程序可用 System.getProperty "key") 来获取 value

这样一个用法,我们就可以在启动项目的时候指定全局参数了,比如在启动的时候选择是 单机模式,还是群模式启动;

例如 **Nacos** 单机启动的时候

```
sh startup.sh -m standalone
```

我们看看这个startup.sh是怎么启动的

```
# JVM Configuration

if [[ "${MODE}" == "standalone" ]]; then
    JAVA_OPT="${JAVA_OPT} -Xms512m -Xmx512m -Xmn256m"
    JAVA_OPT="${JAVA_OPT} -Dnacos.standalone=true"
else
    JAVA_OPT="${JAVA_OPT} -server -Xms2g -Xmx2g -Xmn1g -XX:MetaspaceSize=128m -XX:
axMetaspaceSize=320m"
    JAVA_OPT="${JAVA_OPT} -XX:-OmitStackTraceInFastThrow -XX:+HeapDumpOnOutOfMe
oryError -XX:HeapDumpPath=${BASE_DIR}/logs/java_heapdump.hprof"
    JAVA_OPT="${JAVA_OPT} -XX:-UseLargePages"

fi
```

看最终**JAVA_OPT** 是

```
JAVA_OPT="${JAVA_OPT} -Xms512m -Xmx512m -Xmn256m"   JAVA_OPT="${JAVA_OPT} -D
acos.standalone=true"
```

这里的 -Dnacos.standalone=true 我们就可以在系统中用System.getProperty("nacos.standalone") 取到的值就是 **true**了;

顺便看下Nacos中是是不是用这个方法获取的Jvm属性呢

```
/**
    * Standalone mode or not
```

```
     */
    public static final boolean STANDALONE_MODE = Boolean.getBoolean(STANDALONE_MO
E_PROPERTY_NAME);
```

这个 Boolean.getBoolean()方法最终也是调用了System.getProperty("nacos.standalone")方法的

```
public static boolean getBoolean(String name) {
        boolean result = false;
        try {
            result = parseBoolean(System.getProperty(name));
        } catch (IllegalArgumentException | NullPointerException e) {
        }
        return result;
    }
```

用法注意

* 如果vaule中有空格,则需要用双引号将该值括起来，如：-Dname='Eleven name'.

* 这个是需要设置JVM的参数而不是program的参数(就是那个args)

* 使用此参数的参数优先级最高，会覆盖项目中配置的此项

Spring中配置优先级

关于上面的第三点, 优先级最高的问题;

在整个Jvm运行期间，我们可以随时随地获取到2个与环境相关的参数：

```
//env是与操作系统相关的参数
Map<String, String> env = System.getenv();
//properties中是Jvm相关的参数
Properties p = System.getProperties();
System.out.println("env :" + env);
System.out.println("properties :" + p);
```

如果没有人为的添加额外信息，System::getEnv获取的数据都与当前的操作系统相关（以下称为“操系统参数”），而System::getProperties获取的内容都与Jvm相关（以下称为“JVM参数”）。

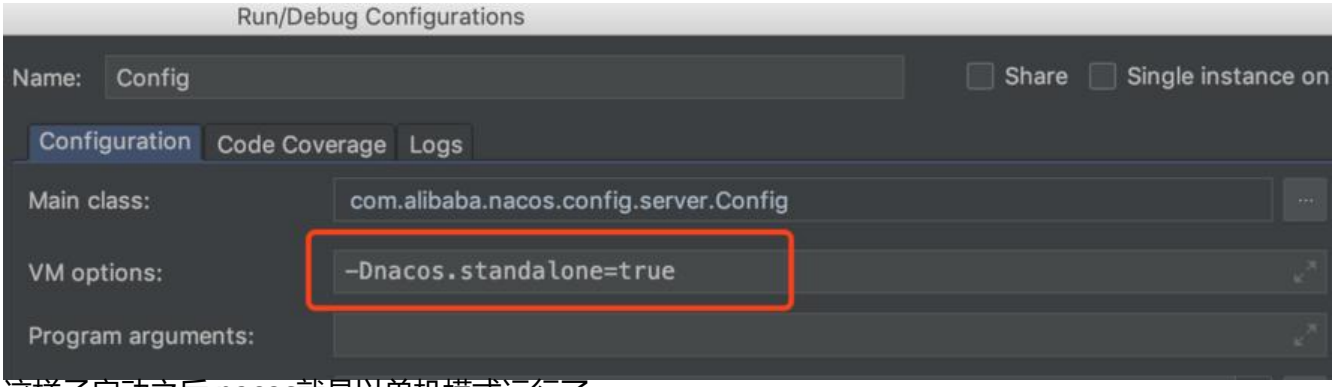
Spring会将操作系统参数和Jvm参数都整合到自己的环境管理接口Environment中

通常情况下，在Environment内部维护了2个PropertySources的实例：一个是操作系统参数，另外一是JVM参数。如果2者有同样的参数，那么我们在调用Environment::getProperty方法时，得到的是JV参数（System::getProperties）,也就是说 Jvm参数具有更高的优先级;

此部分内容引用:Spring核心--资源数据管理

Idea中怎么添加Jvm的属性

以Nacos为例



这样子启动之后,nacos就是以单机模式运行了

原文链接：Java -D 设置系统属性讲解