## Class

三种得到Class对象的方式

|  |  |  |  |
| --- | --- | --- | --- |
| 实现方式 | 说明 | 执行一次的耗时(毫秒) | 重复调用一百万次的耗时(毫秒) |
| 类型.class | JVM只装入该类，但不做初始化操作(静态初始化) | 42 | 48 |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 实现方式 | 说明 | 执行一次的耗时(毫秒) | 重复调用一百万次的耗时(毫秒) |
| 类型.class | 只将该类装入JVM，但不做初始化操作(静态初始化) | 0.805 | 6.869 |
| Class.forname() | 将该类装入JVM，同时执行初始化操作(初始化静态成员，执行static代码块) | 0.93 | 670.198 |
| 对象.getClass() |  | 0.93 | 3.895 |

由此可见通过 对象.getClass()的方式得到Class的性能是最好的

调用一个方法一百万次，方法中什么都不做，耗时0.935538毫秒。可以以这个时间衡量上面三种方式的耗时。

需要注意的是，使用Class.forname()时，是先装入JVM，紧接着做静态初始化，如果该类在静态初始化时(比如调用static代码块)依赖其他东西，可能需要将其依赖提前装入JVM

## Field和Method

使用java反射操作Field和Method耗时主要是调用class.getDeclaredField()获得Field和class.getMethod()获得Method时比较耗时。而调用field.get()和field.set()以及method.invoke()这些方法并不是很耗时。经过一百万数据量的测试，发现操作Field比操作Method稍快一些。所以使用反射时尽量操作Field。

|  |  |  |
| --- | --- | --- |
| 一百万级别的测试 | |  |
| 操作方式 | 耗时(ms) | 比普通方式慢百分之多少 |
| 普通方式(对象.方法) | 210.2903 |  |
| 操作Field | 235.5063 | 慢11% |
| 操作Method | 250.9776 | 慢19% |

但操作field**需要注意**

getFields() 获取所有public字段,包括父类字段

getDeclaredFields() 获取所有字段,public和protected和private,但是**不包括父类字段**

如果需要查找父类中的非public字段可以参考Spring中的工具类org.springframework.util.ReflectionUtils提供的findField方法

public static Field findField(Class<?> clazz, String name, Class<?> type) {

Assert.notNull(clazz, "Class must not be null");

Assert.isTrue(name != null || type != null, "Either name or type of the field must be specified");

Class<?> searchType = clazz;

while (!Object.class.equals(searchType) && searchType != null) {

// 这里一次性获取了类中的所有元素

Field[] fields = searchType.getDeclaredFields();

for (Field field : fields) {

if ((name == null || name.equals(field.getName())) && (type == null || type.equals(field.getType()))) {

return field;

}

}

// 获取所有父类，接着遍历父类中的元素。

searchType = searchType.getSuperclass();

}

return null;

}