Java 语法特性之静态导入

作者：leap
原文链接：https://ld246.com/article/1500737846506
来源网站：链滴

什么是静态导入?

JDK1.5引入了static import机制,借助这一机制，可以用略掉所在的类或接口名的方式来使用静态成。为什么要这么做呢？因为我们知道在Java中常量和函数是不能单独存在的，必须要依附于类或者接。在这种情况下我们访问常量和函数时必须带上类或者接口名来做限定。静态导入或者叫静态成员导就是为了解决这一问题，让我们可以像访问本类中的变量和函数一样访问外部变量和函数。

使用方法

精确导入

精确的导入一个静态成员的方法，是在源文件的开头部分（任何类或接口的定义之前），加上类似这的声明：

import static 包名.类或接口名.静态成员名;

==注意尽管这个机制的名称叫做“static import”，但是在这里的次序却是正好相反的“import stat c”。==

```
//精确的导入Math.sin和Math.PI
import static java.lang.Math.sin;
import static java.lang.Math.PI;

public class staticImportSampleA {
    public static void main(String[] args) {
          System.out.println(sin(PI/2));//输出“1.0”
  }
 }
```

按需导入

static import机制也支持一种不必逐一指出静态成员名称的导入方式。这时，要采用这样的语法：

import static 包名.类或接口名.*;

==注意这种方式只是指出遇到来历不明的成员时，可以到这个类或接口里来查找，并不是把这个类接口里的所有静态成员全部导入。==

```
//声明遇到来历不明的成员时到java.lang.Math中去寻找
import static java.lang.Math.*;

public class staticImportSampleB {
    public static void main(String[] args) {
       System.out.println(sin(PI/2));//输出“1.0”
    }
}
```

原文链接：Java 语法特性之静态导入

可以导入一切静态成员

使用import static语句，可以导入一个类里的一切被static修饰的东西，包括变量、常量、方法和内类。

```
package com.example.p3;

public class staticImportee {
    //静态变量
    public static int one = 1;
    public static final int TWO = 2;
    //静态方法
    public static int three() {
        return 3;
    }
    //内部类
    public static class Four {
        public int value() {
            return 4;
        }
    }
}
```

```
package com.example.p3;
import static com.example.p3.staticImportee.*;

public class staticImporter {
    public static void main(String[] args) {
        System.out.println(one);
        System.out.println(TWO);
        System.out.println(three());
        System.out.println(new Four());
    }
}
```

不受影响的访问控制

static import不能突破Java语言中原有的访问控制机制的限制，不过也并不在这方面增加新的约束。来有权限访问的静态成员，都可以被导入和使用；而原来无权限访问的静态成员，用了这个方法之后仍然是不能访问。

使用原则

外部导入冲突

不同的类（接口）可以包括名称相同的静态成员。因此，在进行static import的时候，可能会出现“个语句导入同名的静态成员”的情况。

在这种时候，JDK1.5会这样来加以处理：

- 如果两个语句都是精确导入的形式，或者都是按需导入的形式，那么会造成编译错误。
- 如果一个语句采用精确导入的形式，一个采用按需导入的形式，那么采用精确导入的形式的一个有

。

==注意，如果两个同名的静态成员一个是属性，而另一个是方法，那么因为使用时的写法有差异，会造成冲突。==

本地导入冲突

有时候，导入的东西还可能和本地的东西相冲突，这种情况下的处理规则是“本地优先”。

总结

在编译期间，所有因static import的存在而简化了的名字，都会被编译器打回原型。因此在性能方面，tatic import没有任何影响。但是名字简化却可能造成一些维护方面的问题。

去掉静态成员前面的类型名，固然有助于在频繁调用时显得简洁，但是同时也失去了关于“这个东西哪里定义”的提示信息，增加了阅读理解的麻烦。如果导入的来源很著名（比如java.lang.Math），者来源的总数比较少，这个问题并不严重；但是在不属于这两种的情况下，这就不是基本可以忽略的题了。

借助JDK 1.5里提供的static import机制，可以用一种更简单的方式，来访问类和接口的静态成员。过，使用这一机制并不是没有代价的，在使用不当的时候可能给维护工作带来一定的困扰。因此，在体使用之前，还要作一些权衡。

ps:普遍做法还是直接通过类名或接口名访问，这样可读性更好。

在实际项目中没有遇到过使用这一特性，在这里仅作为一个知识点总结下。

参考 理解Java的static import静态引入机制