第二章 创建和销毁对象

2018年7月9日

19:12

**第一条：考虑用静态工厂方法代替构造器**

**静态工场方法与构造器不同的优势：**

* 他们有名称。

当一个类需要多个带有相同签名的构造时，就用静态工厂方法代替构造器。

* 不必每次调用他们的时候都创建一个新对象。

Flyweight模式，实例受控类

* 他们可以返回原返回类型的任何子类型的对象。

基于接口的框架，服务提供者框架

* 在创建参数化类型实例的时候，它们使代码变得更加简洁。

类型推导

**静态工厂方法的缺点：**

* 类如果不含有共有的或者受保护的构造器，就不能被子类化。

多使用复合，少使用继承。

* 他们与其他的静态方法实际上没有任何区别。

静态工厂方法的惯用名称：

* valueOf 类型转换方法
* Of valueOf 的一种简介的替代
* getInstance 返回一个实力
* newInstance 返回唯一的实力
* getType Type表示工厂方法所返回的对象类型

**第二条：遇到多个构造器参数时要考虑用构建器**

* 重叠构造器模式

重叠构造器可行，但是当有很多参数的时候，客户端代码会很难编写。

* JavaBeans模式

JavaBean模式的缺点：

* 构造过程中JavaBean可能处于不一致的状态。
* JavaBean莫斯阻止了把类做成不可变类的可能。
* Builder模式

Builder模式模拟了具名的可选参数。

如果类的构造器或者静态工厂中具有许多个参数，设计这种类的时候，Builder模式就是中不错的选择。

Class.newInstance破坏了编译时的异常检查。

**第三条：用私有构造器或者枚举类型强化Singleton属性**

Singleton是指仅仅被实例化一次的类。

* 公有实例域方法

可能会被通过反射机制调用私有构造器。

* 静态工厂方法：

灵活性，泛型有关。

* 枚举方法

单元素的枚举类型已经成为实现Singleton的最佳方法。

**第四条：通过私有构造器强化不可实例化的能力**

用于工具类等不希望被实例化的类应该构造器私有。

企图通过将类做成抽象类来强制该类不可能被实例化。

**第五条：避免创建不必要的对象**

* 一般来说，最好能重用对象而不是在每次需要的时候就创建一个功能相同的新对象。

Date对象最好是重用。

* 通过延迟初始化可以消除一些不必要的初始化。
* 优先使用基本类型而不是装箱基本类型，要当心无意识的自动装箱。
* 对象是非常重量级的时候，优先使用对象池来避免创建对象

**第六条：消除过期的对象引用**

* 只要类是自己管理内存，程序员应该警惕内存泄漏问题。

栈先增长后收缩，栈还会维护这些对象的过期引用。一旦对象已经过期，立刻清空这些引用。

* 内存泄漏的另一个常见来源是缓存。
* 内存泄漏的第三个常见来源是监听器和其他回调。

**第七条：避免使用终结方法**

终结方法(finalizer)通常是不可预测的，也是很危险的，一般情况下是不必要的。

使用终结党法有一个人非常严重的性能损失。

显式的终结方法通常与try-finally结构结合起来使用。