## 简介

运用共享技术有效地支持大量细粒度对象。即复用我们内存中已存在的对象，降低系统创建对象实例的性能消耗。

通常分为两种状态：外蕴状态和内蕴状态。

内蕴状态存储在享元内部，不会随环境的改变而有所不同，是可以共享的

外蕴状态是不可以共享的，它随环境的改变而改变的，因此外蕴状态是由客户端来保持（因为环境的变化是由客户端引起的）。

## 特点

复用我们内存中已存在的对象，降低系统创建对象实例的性能消耗。

## 角色

1. 抽象享元角色：为具体享元角色规定了必须实现的方法，而外蕴状态就是以参数的形式通过此方法传入。在Java中可以由抽象类、接口来担当。
2. 具体享元角色：实现抽象角色规定的方法。如果存在内蕴状态，就负责为内蕴状态提供存储空间。
3. 享元工厂角色：负责创建和管理享元角色。要想达到共享的目的，这个角色的实现是关键！
4. 客户端角色：维护对所有享元对象的引用，而且还需要存储对应的外蕴状态。

## 优缺点

优点：

1. 减少对象创建，降低内存开销，提高效率。

缺点

1、提高了系统的复杂度，需要分出内部状态和外部状态，而且外部状态需要有固有化性质，不应随着内部状态的变化而变化，否则会造成系统混乱。

## 使用场景

1. 当我们系统中某个对象类型的实例较多的情况。
2. 并且要求这些实例进行分类后，发现真正有区别的分类很少的情况。
3. 通过使用享元模式后能够提高系统的性能和不会带来更多的复杂度时。

## 参考

http://www.cnblogs.com/hegezhou\_hot/archive/2010/12/12/1903728.html