**无聊，谈一些面向对象设计思想的高级话题**

**元类，类的模板**  
  
**1 什么是元类**  
  
      我们说，面向对象本质上是过程化方法的高级阶段，其目的还是为了重用代码。或者说，同过程化相比，面向对象是更高一级的抽象，它把相似的对象归为一个类型，描述的是一种通用的“泛化”关系。那么，更高的一个层次上，我们说，如果是相似的类呢？那又应该如何处理？  
      面向对象概念中从来不缺乏描述类与类之间相似性的机制，比如继承和接口、比如原型，但是JavaScript中有没有一种机制，可以描述类的“创建型”泛化关系呢？答案是有的，它就是我们所说的“元类”。  
      所谓的元类，是一种特殊的类，它的作用是构造一组类，这组类都具有同样的特征。元类和类的关系就像类和对象的关系一样，是一种创建型的泛化关系。  
  
**2 元类：构造类的类**  
  
      之前我们已经知道，JavaScript中的函数也是对象，它们是Function类型的实例。而在JavaScript的面向对象中，函数本身被作为类型。因此，从关系上来讲，Function就是一个元类，它是由JavaScript语言本身提供的。  
      但是，从Function的实现角度来讲，它又不适合作为创造类的元类，因为它的构造方法十分繁琐，是通过字符串来直接生成函数的，例如：  
        //TODO:（因为太无聊了，所以连这个也省略了）  
  
      于是，我们考虑其他的方式。还是回想第7章的那个规则——“构造函数通常没有返回值，它们只是初始化由this值传递进来的对象，并且什么也不返回。如果函数有返回值，被返回的对象就成了new表达式的值”——如果我们在构造函数里返回一个函数或者闭包，那么会怎么样？答案是令人吃惊的，那样我们就创建了一个用来构造类的类，也就是我们所说的“元类”。  
      例如：  
  
   提示：您可以先修改部分代码再运行  
  
在上面的例子里，我们创建了一个叫做Singleton的元类，它抽象的是一种设计模式，我们用它来构造了类SessionFactory，它遵循了单例模式，因此不允许被实例化，当我们尝试去实例化它时，系统就报告了异常。  
  
**3 为什么要用元类**  
  
      单纯从上面的例子来看，似乎使用元类的理由还不够充分，但是仔细思考一下，当你使用元类思想处理一些足够复杂的问题时，它带来好处就显而易见了，你可以像普通对象一样继承你的类，你也可以给你的元类指定原型，元类的静态方法对应于对象的类方法，必要的时候你可以派生一系列的类（而不是对象），再用这些类去创造形形色色的对象。总而言之，元类思想提升了你抽象建模的能力，它是一种非常强大的高层次的抽象方法，下面一个小节里，通过一些稍微复杂的例子演示这种思想能够的创造的奇迹：  
       （尤其无聊，干脆连这个也省略了 ^^）  
  
[[i] 本帖最后由 月影 于 2007-4-15 13:53 编辑 [/i]]

一个稍微复杂一点的例子 看起来很像C++模板的Group类

**QUOTE:**

function ClassA()  
{  
    this.constructor=ClassA;  
    this.v="!!";  
    this.run=function(){  
        alert(this.v);  
    }  
    this.clone=function(){  
        var r=new ClassA();  
        r.v=this.v;  
        return r;  
    }  
}  
var a=new ClassA();  
var b=new ClassA();  
var ClassAGroup=new Template\_Group(ClassA);  
var ag=new ClassAGroup();  
a.v="a";  
b.v="b";  
ag.add(a);  
ag.add(b);  
var ag2=ag.clone().clone().clone();  
ag2.run();

Template\_Group从一个类（构造器）构造一个新的类  
这个新的类是原来的类的对象的集合 可以通过add 和remove为它添加新的成员或者移除成员  
这个新的类拥有跟原来的类相同的方法  调用它的方法时 它会调用每个成员的方法  
如果成员的方法有返回值 并且返回值是对象类型的话 它会返回一个Group