static{}语句块详解

static{}(即static块)，会在类被加载的时候执行且仅会被执行一次，一般用来初始化静态变量和调用静态方法，下面我们详细的讨论一下该语句块的特性及应用。

**一、在程序的一次执行过程中，static{}语句块中的内容只被执行一次，看下面的示例:**

**示例一**

class Test{

    public static int X=100;

public final static int Y;=200

public Test(){

    System.out.println("Test构造函数执行");

}

static{

    System.out.println("static语句块执行");

}

public static void display(){

    System.out.println("静态方法被执行");

}

public void display\_1(){

    System.out.println("实例方法被执行");

}

}

public class StaticBlockTest{

public static void main(String args[]){

    try{

            Class.forName("Test");

                Class.forName("Test");

    }catch(ClassNotFoundException e){

        e.printStackTrace();

    }

}

}

**结果**:你会发现虽然执行了两条Class.forName("Test")语句，但是，只输出了一条"静态方法被执行"语句；其实第二条Class.forName()语句已经无效了，因为在虚拟机的生命周期中一个类只被加载一次；又因为static{}是伴随类加载执行的，所以，不管你new多少次对象实例，static{}都只执行一次。 --**关于类加载请看本文的附录。**

**二、static{}语句块执行的时机(其实就是附录中类加载的时机)**

**上面说到static{}会在类被加载的时候执行，我们必须准确理解类加载的准确含义，含义如下:**

1、用Class.forName()显示加载的时候，如上面的示例一;

2、实例化一个类的时候，如将main()函数的内容改为:Test t=new Test();//这种形式其实和1相比，原理是相同的，都是显示的加载这个类，读者可以验证Test t=new Test();和Test t=(Test)Class.forName().newInstance();这两条语句效果相同。

3、调用类的静态方法的时候，如将main()函数的内容改为:Test.display();

4、调用类的静态变量的时候，如将main()函数的内容改为:System.out.println(Test.X);

**总体来说就这四种情况，但是我们特别需要注意一下两点:**

1、调用类的静态常量的时候，是不会加载类的，即不会执行static{}语句块，读者可以自己验证一下(将main()函数的内容改为System.out.println(Test.Y);)，你会发现程序只输出了一个200；(这是java虚拟机的规定，当访问类的静态常量时，如果编译器可以计算出常量的值，则不会加载类，否则会加载类)

2、用Class.forName()形式的时候，我们也可以自己设定要不要加载类，如将Class.forName("Test")改为 Class.forName("Test",false,StaticBlockTest.class.getClassLoader())，你会发现程序什么都没有输出，即Test没有被加载，static{}没有被执行。

**三、static{}语句块的执行次序**

**1、**当一个类中有多个static{}的时候，按照static{}的定义顺序，从前往后执行；

**2、**先执行完static{}语句块的内容，才会执行调用语句；

**示例二**

public class TestStatic{  
    static{  
        System.out.println(1);  
    }  
    static {  
        System.out.println(2);  
    }  
    static {  
        System.out.println(3);  
    }  
    public static void main(String args[]){  
        System.out.println(5);  
    }  
    static {  
        System.out.println(4);  
    }  
}  
**结果**:程序会输出1，2，3，4，5

3、如果静态变量在定义的时候就赋给了初值(如 static int X=100)，那么赋值操作也是在类加载的时候完成的，并且当一个类中既有static{}又有static变量的时候，同样遵循“先定义先执行”的原则；

**示例三**

 class Test{  
 public static int X=300;  
 static{  
  System.out.println(X);  
  X=200;  
  System.out.println(X);  
 }  
}

public class StaticBlockTest{  
 public static void main(String args[]){  
  System.out.println(Test.X);  
 }  
}

结果:程序会依次输出300，200，200，先执行完X=300，再执行static{}语句块。

**四、static{}语句块应用**

**1、JDBC中的应用**

熟悉JDBC的读者应该知道，java中有一个DriverManager类，用于管理各种数据库驱动程序、建立新的数据库连接。DriverManager类包含一些列Drivers类，这些Drivers类必须通过调用DriverManager的registerDriver()方法来对自己进行注册，那么注册是什么时候发生的呢？下面会给出答案:

所有Drivers类都必须包含有一个静态方法，利用这个静态方法可以创建该类的实例，然后在加载该实例时向DriverManage类进行注册。我们经常用Class.forName()对驱动程序进行加载，那么注册就发生在这条语句的执行过程中，前面说的Drivers的静态方法是放在static{}中的，当对驱动程序进行加载的时候，会执行该static{}，便完成了注册。

**2、hibernate中的应用**

hibernate中的SessionFactory是一个重量级的类，创建该类的对象实例会耗费比较多的系统资源，如果每次需要时都创建一个该类的实例，显然会降低程序的执行效率，所以经常将对该类的实例化放在一个static{}中，只需第一次调用时执行，提高程序的执行效率，如下:

static {  
     try {  
   configuration.configure(configFile);  
   sessionFactory = configuration.buildSessionFactory();  
  } catch (Exception e) {  
   System.err.println("%%%% Error Creating SessionFactory %%%%");  
   e.printStackTrace();  
  }  
    }

**五、附录**

**类加载:**Java命令的作用是启动虚拟机，虚拟机通过输入流，从磁盘上将字节码文件(.class文件)中的内容读入虚拟机，并保存起来的过程就是类加载。

**类加载特性 :**      \***在虚拟机的生命周期中一个类只被加载一次。**      \***类加载的原则：延迟加载，能少加载就少加载，因为虚拟机的空间是有限的。**      \***类加载的时机：**      1）第一次创建对象要加载类.  
      2）调用静态方法时要加载类,访问静态属性时会加载类。  
      3）加载子类时必定会先加载父类。  
      4）创建对象引用不加载类.  
      5) 子类调用父类的静态方法时  
          (1)当子类没有覆盖父类的静态方法时，只加载父类，不加载子类  
          (2)当子类有覆盖父类的静态方法时，既加载父类，又加载子类  
      6）访问静态常量，如果编译器可以计算出常量的值，则不会加载类,例如:public static final int a =123;否则会加载类,例如:public static final int a = math.PI。