
```
package main.java.Test1;

/*
   线程上下文类加载器的一般使用模式(获取--使用--还原)
   ClassLoader classLoader=Thread.currentThread.getContextClassLoader();
   try{
      Thread.currentThread.setContextClassLoader(targetTcc1);
      myMethod();
   }catch{
   }finally{
      Thread.currentThread().setContextClassLoader(classLoader);
   }

    myMethod里面调用了Thread.currentThread().getContextClassLoader(),获取当前线程的上下文类加载器做一些事情

   如果一个类由加载器A加载,那么这个类的依赖类也是由相同的类加载器加载(如果该依赖类之前没有被加载过得话)

   ContextClassLoader的作用就是为了破坏Java类加载双亲委托机制

   当高层提供统一的接口让低层去实现,同时又要在高层加载(或实例化)低层类的时候,就必须通过线程上下文类加载器
   来帮助高层的classLoader找到并加载该类
 */
import java.sql.Driver;
import java.util.Iterator;
import java.util.ServiceLoader;

public class TestContextClassloader {
   public static void main(String[] args) {
      //获取所有这个接口的实例
      ServiceLoader<Driver> loader=ServiceLoader.load(Driver.class);
      Iterator<Driver> iterator=loader.iterator();
      while(iterator.hasNext()){
```

```
            Driver driver=iterator.next();

System.out.println("Driver:"+driver.getClass()+",loader:"+driver.getClass().getClassLoader());

        }
        System.out.println("当前线程上下文类加载
器:"+Thread.currentThread().getContextClassLoader());
        System.out.println("ServiceLoader的类加载器:"+loader.getClass().getClassLoader());
    }
}
```