proxy模式：实现了相同接口的类【外界看来调用方式没有改变】，代替执行相关逻辑，并在执行前进行相关初始化等工作

public **interface Subject** {

   void request();

}

public class **ConcreteSubject implements Subject** {

   public void request() {

       System.out.println("执行功能");

  }

}

public class **Proxy implements Subject** {

   private Subject subject;

   public Proxy(Subject subject) {

       this.subject = subject;

  }

public void request() {

sout(“执行额外条件判断，考虑是否要调用request 方法”);

Boolean invoke = true;

If (!invoke) {

Return;

}

       subject.request();

  }

}

代理模式实在是太常见太常见了，2个场景：

（1）懒加载：hibernate、mybatis【Hibernate适用于希望使用对象操作和自动化功能的场景，而MyBatis适用于对SQL掌握较多、需要更多手动控制和灵活性的场景。】，都可以实现懒加载，这个大家应该都知道了，其实也是基于动态代理思想实现的，对刚查询出来的对象做一个代理。然后去调用getX()方法的时候，再实际执行sql查询出来关联的数据。

public **interface Classroom** {

    List<Student> getStudents();

}

public class **ConcreteClassroom implements Classroom** {

    private List<Student> students;

【只加载了classroom 对象，只有当调用 getStudents 方法时，才接着查询对应 students 对象】

    public List<Student> getStudents() {

        return student;

    }

}

public class **ProxyClassroom implements Classroom** {

    private ConcreteClassroom classroom;

    public List<Student> getStudents() {

        // 先查询一下缓存，内存缓存中是否有students数据

        // 如果没有的话，那就发送SQL从数据库中查询出来，然后放到缓存中去

        // 接着返回students数据

        classroom.setStudents(students);

        return classroom.getStudents();

    }

}

通过hibernate/mybatis，查询出来这个classroom的时候，是没有级联将students数据从数据库里查出来的。接下来，如果你要从classroom中获取students集合来使用的时候，你会去调用getStudents()。懒加载的原理，其实就是基于代理模式去做的，刚从数据库中查询出来的是classroom的一个代理对象

现在在互联网公司里，**不提倡说利用这种ORM的方式去开发数据库的操作，就是不要推荐在类中放置其他类的引用，就是面向对象的层级在组织数据了**

（2）远程调用：比如thrift等rpc调用包，都是在本地做一个远程服务的本地代理接口，然后对本地代理接口来调用，然后代理对象发送网络请求调用远程服务的接口

系统要调用另外的一个系统的接口，要走网络请求，远程调用，http接口；**rpc，系统在本地就是调用一个代理类的接口，代理类在接口的代理实现中，是会发送网络请求跟远程的系统通信，调用对方的接口**

public class Proxy implements Interface {

    public void request() {

        // 发送网络请求到远程系统，去调用对方的接口

    }

}

spring cloud，dubbo，分布式系统的架构，分布式系统之间互相的调用，其实就是大量的用了代理模式，代理接口，代理了远程系统的接口

我们这里的代理模式实践，会基于调用外部接口的方式来，比如支付和物流，都要调用第三方公司的接口，那么完全可以做一个代理类，我们自己的系统针对代理类来操作，代理类代理了对外部接口的访问，非常合适的场景

853769620