软件设计文档

（说明：本项目来源于系统分析与设计大作业）

**目录**

[1 技术选型 3](#_Toc521463250)

[1.1框架选择 3](#_Toc521463251)

[1.2后台技术 3](#_Toc521463252)

[1.3数据库连接工具 3](#_Toc521463253)

[2构架设计 4](#_Toc521463255)

[2.1项目包图 4](#_Toc521463256)

[2.2项目部署图 4](#_Toc521463257)

[3模块划分 5](#_Toc521463259)

[3.1用例图 5](#_Toc521463260)

[3.2登录和注册模块 5](#_Toc521463261)

[3.3预定电影票模块 6](#_Toc521463265)

[3.4支付模块 6](#_Toc521463266)

[4软件设计技术 7](#_Toc521463269)

[4.1Object-Oriented Programming 7](#_Toc521463270)

[4.2 Aspect-Oriented Programming 7](#_Toc521463271)

[4.3Design Patterns 9](#_Toc521463272)

[5.接口设计. 13](#_Toc521463259)

[6.数据库设计. 13](#_Toc521463259)

1、技术选型

**1.1框架选择**：Spring MVC

**理由**：

1.Spring是一个轻量级的DI和AOP容器框架。且Spring是非侵入式的，基于spring开发的应用一般不依赖于spring的类。对于这个不算大型的项目比较适合。

2.考虑到另一个框架Structs2，spring会稍微比struts快。spring mvc是基于方法的设计，而sturts是基于类，每次发一次请求都会实例一个action，每个action都会被注入属性，而spring基于方法，粒度更细。

3.Spring有强大而直接的配置方式：将框架类和应用程序累都能作为JavaBean配置，支持跨多个context的引用。

4.可重用的业务代码：可以使用现有的业务对象作为命令或表单对象，而不需要去扩展某个特定框架的基类。

5.可定制的handler mapping和view resolution：[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "_blank)提供从最简单的URL映射，到复杂的、专用的定制策略。与某些web MVC框架强制开发人员使用单一特定技术相比，Spring显得更加灵活。

**数据库选择：**Mysql

**理由：**

MySQL是一个小型关系型数据库管理系统，操作简便，不像其他数据库那样有复杂繁琐的操作，MySQL一直都奉行简单易用的原则。而且有较高的性能和稳定性。

**1.2后台技术：**JavaWeb

**理由：**

1. 适用性强，一般来说几乎所有浏览器都支持java无需特别为一款浏览器特别建立一个网站
2. java web可维护性强，具有开放性，可以增加模块而无需改变许多代码，只需要接入预留的接口就可以。
3. 代码复用率多，无需写许多代码

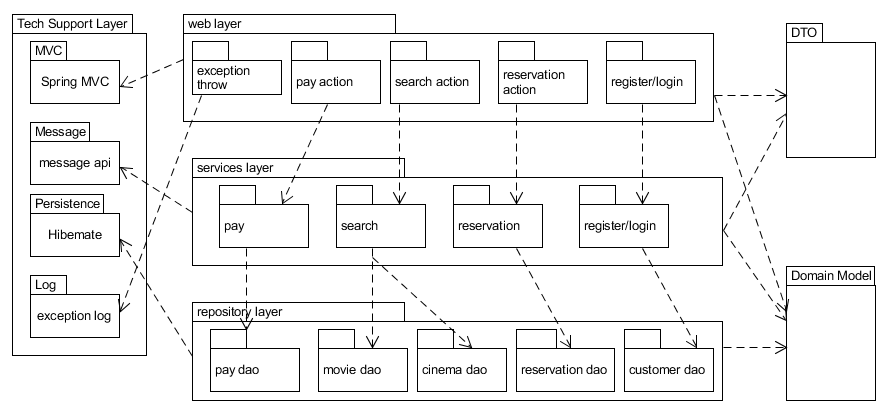
**1.3数据库连接工具**：JDBC

**理由：**

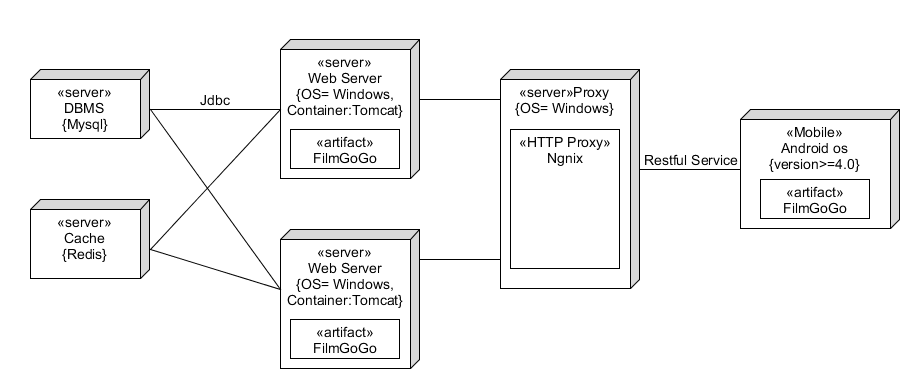
直接底层操作，提供了很简单、便捷的访问数据库的方法，跨平台性比较强。灵活性比较强，可以写很复杂的SQL语句。Java 具有坚固、安全、易于使用、易于理解和可从网络上自动下载等特性，是编写数据库应用程序的杰出语言。所需要的只是 Java应用程序与各种不同数据库之间进行对话的方法。而 JDBC 正是作为此种用途的机制。

2.架构设计

2.1项目包图：

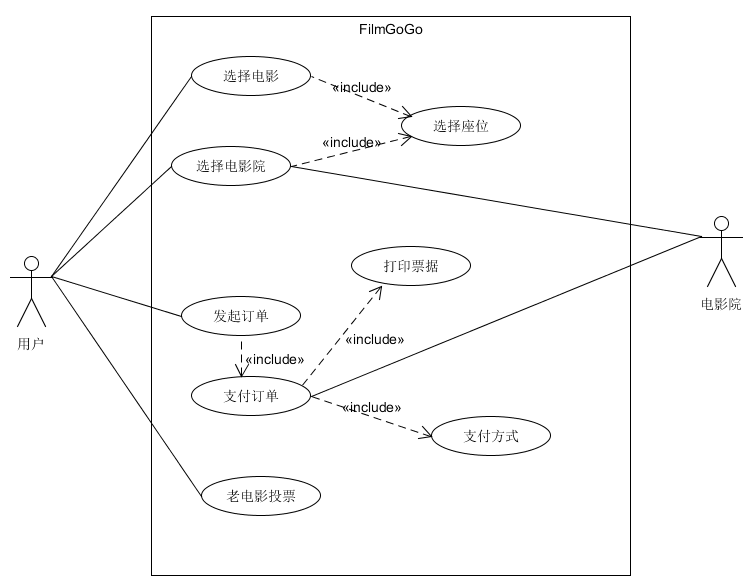


2.2项目部署图：

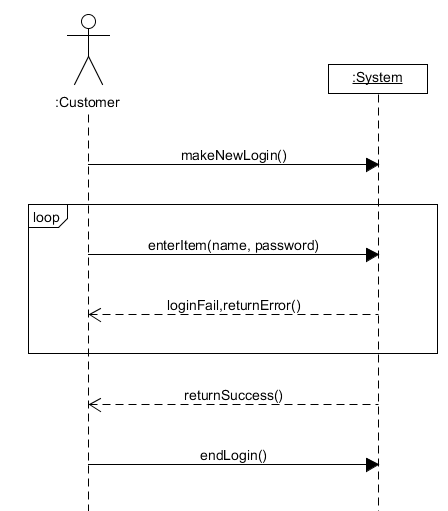
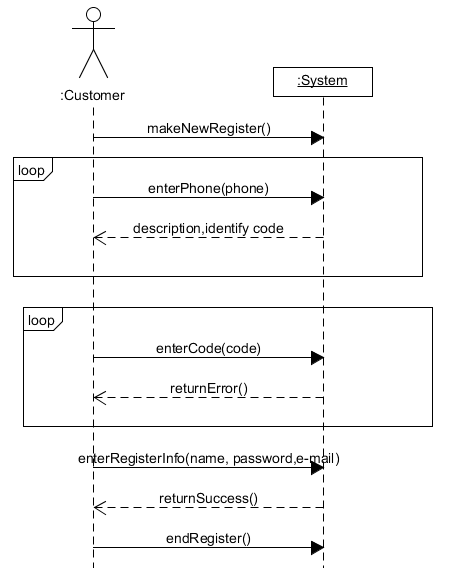


3.模块划分

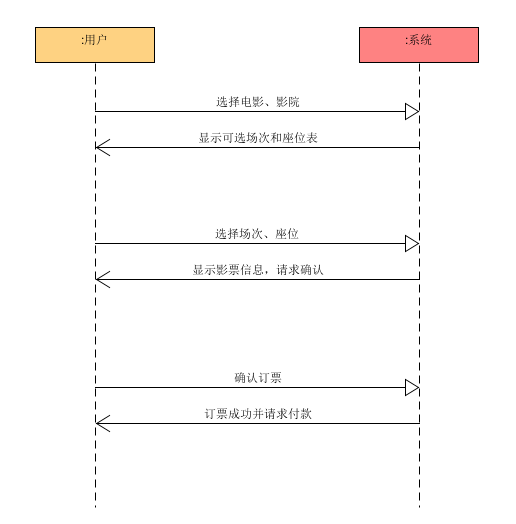
3.1用例图



3.2登录和注册模块

3.3预定电影票模块



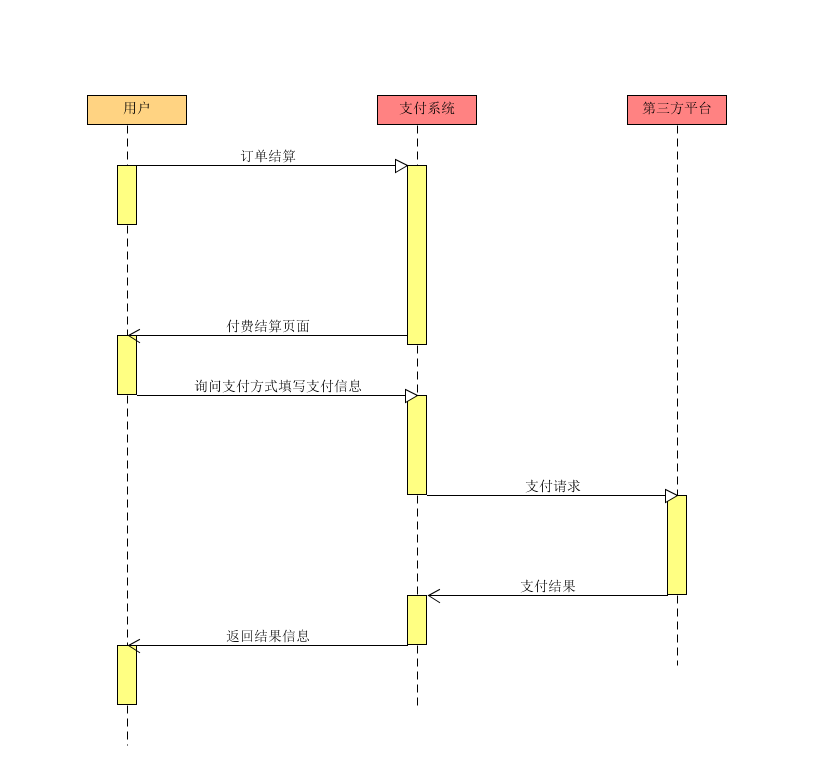
后置条件：

1、创建了Reservation实例R

2、R被关联到对应ticket

3、对应的ticket的isAvailable被置为false，并且将定单锁定。

3.4支付模块



契约操作后置条件

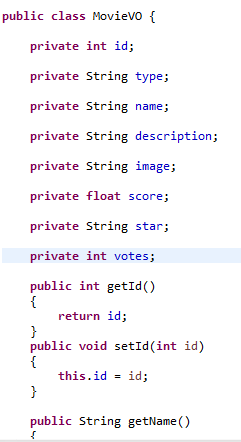
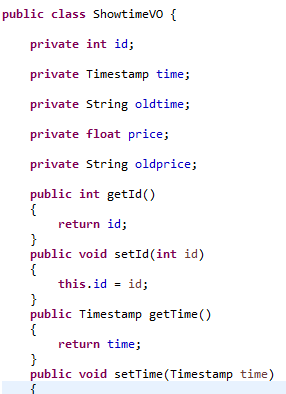
1. 返回支付结果信息后票单信息
2. 锁定对应的订单，不能再修改订单的信息。

4.软件设计技术

**4.1 Object-Oriented Programming**

Java是一门面向对象语言。

在VO中定义各种基础概念类，movie, cinema, showtime, seat, reservation等

**4.2 Aspect-Oriented Programming**  
 AOP的实现主要分为静态植入和动态注入两种方式。

动态注入一句设计模式动态代理模式重写父类（原类）方法hack逻辑，利用截取消息的方式，对该消息进行装饰，以取代原有对象行为的执行。在市面上的常见AOP框架都会依托IOC框架来处理引用。比如**spring的拦截机制**



1. **DispatcherServlet**

SpringMVC具有统一的入口DispatcherServlet，所有的请求都通过DispatcherServlet。  
 DispatcherServlet是前置控制器，配置在web.xml文件中的。拦截匹配的请求，Servlet拦截匹配规则要自已定义，把拦截下来的请求，依据某某规则分发到目标Controller来处理。  所以我们现在web.xml中加入以下配置：

****

1. 执行被拦截的Controller，这里以登录为例

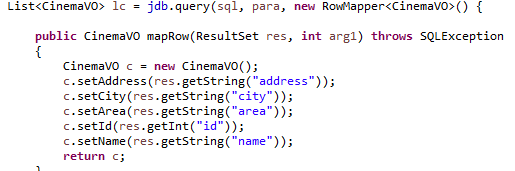
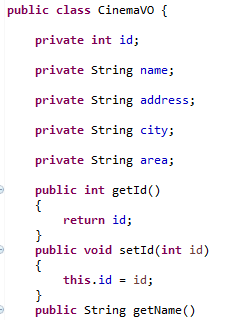


（3）Spring以[XML文件](https://www.baidu.com/s?wd=XML%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPW0knjDsuWnLuynLrjFh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWfvP1TvrjmLnWnYPjTYnj0d" \t "_blank)描述对象逻辑关系的方式，简单实用，有效地将原本紧密耦合的对象分解为[松散耦合](https://www.baidu.com/s?wd=%E6%9D%BE%E6%95%A3%E8%80%A6%E5%90%88&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPW0knjDsuWnLuynLrjFh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWfvP1TvrjmLnWnYPjTYnj0d" \t "_blank)的bean



**4.3 Design Patterns**

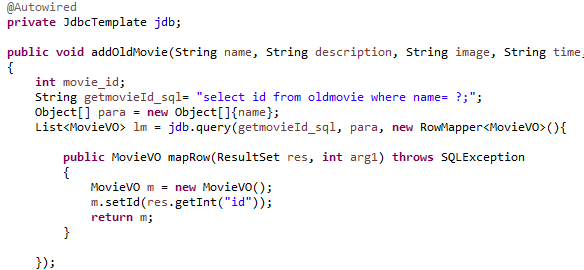
**4.3.1原型模式：**将一个对象作为原型，对其进行复制、克隆，产生一个和原对象类似的新对象。



**4.3.2桥接模式（Bridge）**

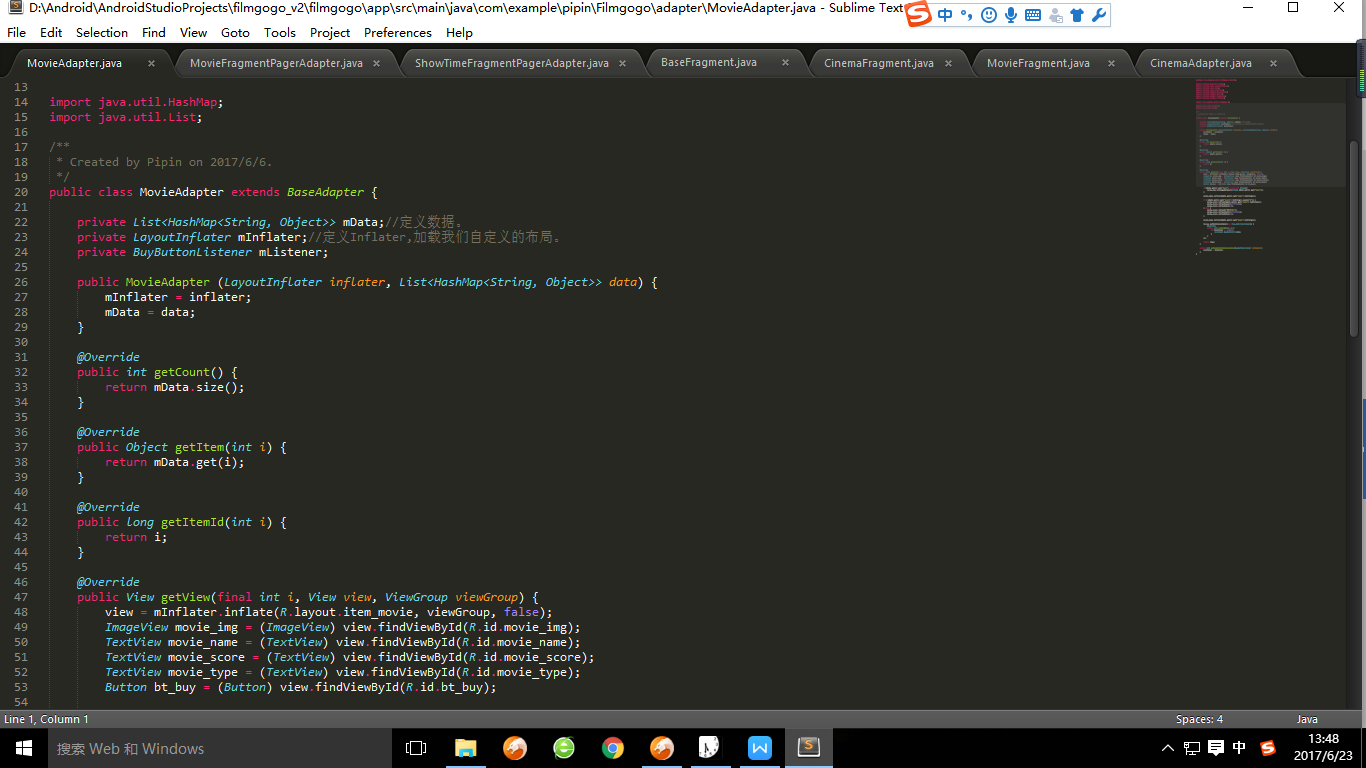
JDBC进行连接数据库的时候，在各个数据库之间进行切换

****

****

**4.3.3**适配器模式

适配器模式将某个类的接口转换成客户端期望的另一个接口表示，目的是消除由于接口不匹配所造成的类的兼容性问题。主要分为三类：类的适配器模式、对象的适配器模式、接口的适配器模式。



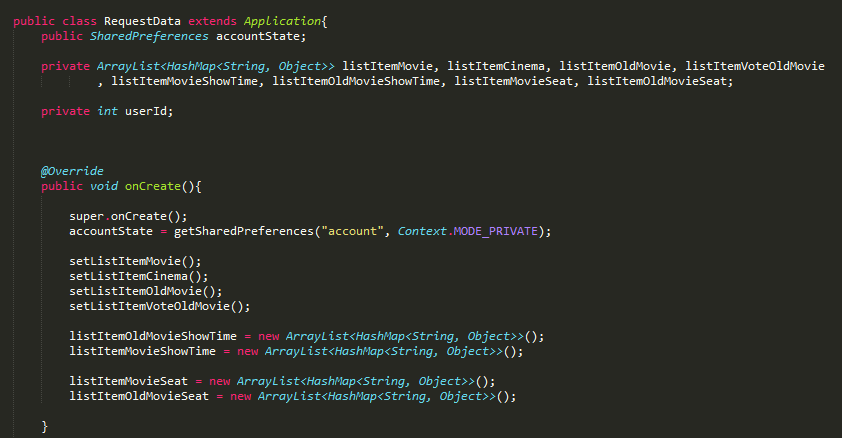
**4.3.4**模板方法模式

我们在Android开发中接触的比较早的设计模式。新建一个Actiivty时都会覆盖onCreate、onResume等生命周期方法，这些方法是在父类中定义好的模板，会由系统自动回调，我们可以通过重写来替换掉这些方法，却无法改变他们的调用顺序。这是典型的模板方法模式。

**4.3.5**单例模式

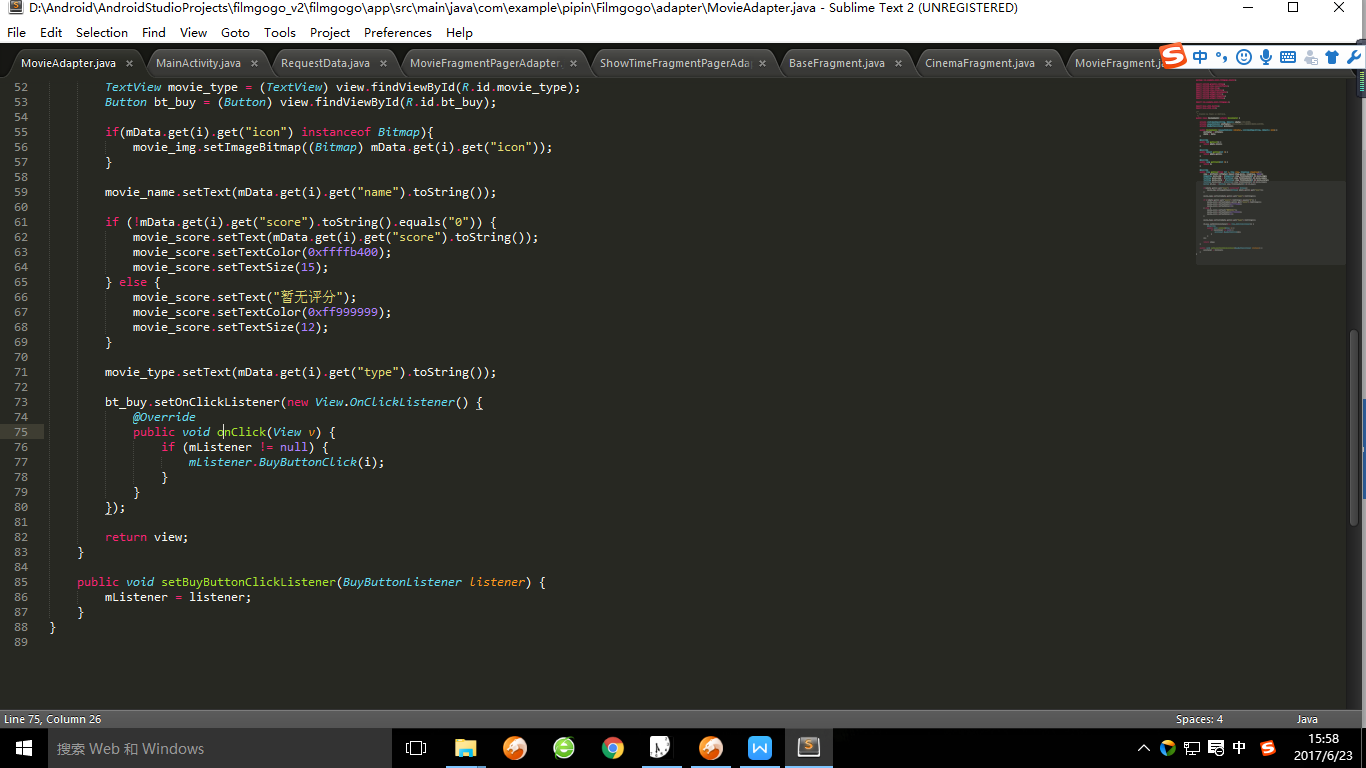
Android中最常见的单例是Application类，它全局只有一个。不过Application的构造函数并非是私有的，也就是说我们可以new一个新的Application对象，因此Application这个类并不符合单例的规范。但是，即使我们重新new出来一个Application，它并没有绑定相关的上下文，也会是无效的，也就保证了全局只有一个有效的Application对象。





**4.3.6**观察者模式

观察者模式在Android中无处不在，Android中的各种Listener，单击、双击、触摸事件的监听处理，都是观察者模式。



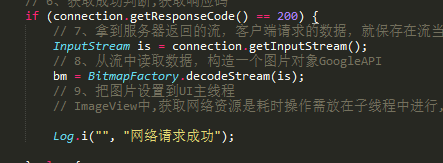
**4.3.7**命令模式

Android中两个典型的使用命令模式的类是Handler和ExecutorService。Handler类提供了post方法发送命令，ExecutorService类通过submit方法提交命令。



**4.3.8**工厂方法模式

工厂方法模式在Android中比较明显的使用就是BitmapFactory，通过各种decodeXXX()方法就可以用不同方式获得Bitmap对象。



5.接口设计

举两个例子，具体看接口文档：

5.1关于登录和注册

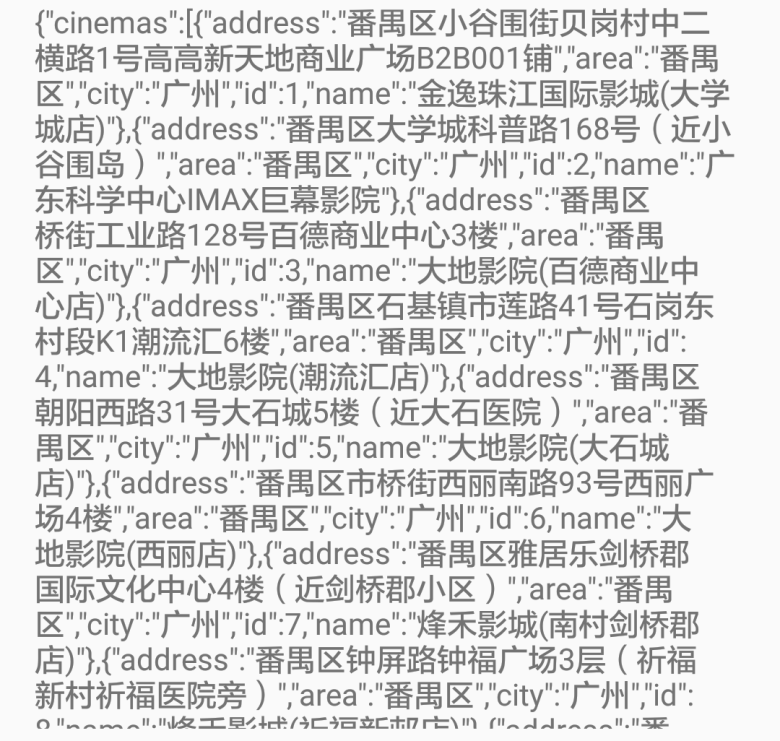
接口：<http://172.18.71.17:8080/FilmGoGo/login>

接口：<http://172.18.71.17:8080/FilmGoGo/register>

5.2关于cinema的请求：

接口：<http://172.18.71.17:8080/FilmGoGo/cinema>

返回的json数据如下：



6.数据库设计

举部分例子：

