**第一节：UML 设计**

**第二节：数据库框架设计**

目的：

1. 调用层不关系数据库操作细节
2. 不关心表的创建
3. 不关心SQL语句的拼写
4. 只需传入对象，便能存入数据库。

SQLite数据库

调用层(Activity)

BaseDao

IBaseDao接口 （增、删、改、查）

**第三节：网络访问框架**

目标：

1. 支持请求：JSON 文本类型、音频、图片类型、批量下载、上传；
2. 请求各种数据时，调用层不关心上传参数封装，如request.addParams(key value),直接将参数封装为对象传给框架。
3. 获取数据后，调用层不关心JSON数据解析，回调时，调用层只需要知道传入的JSON的对应响应类。
4. 回调响应结果发生在主线程(线程切换)
5. 对下载，上传扩展
6. 支持高并发请求，请求队列依次获取，可以设置最大并发数，设置先请求先执行。

知识点和设计模式：

1. 泛型、请求队列、阻塞队列、线程池拒绝策略；
2. 模板方法模式、单例模式、策略模式、生产者消费者模式。

回调

RequestHolder ：

指定策略Request封装为T对象

指定策略Response封装为M对象

传入URL

传入回调接口：Listener

JsonDealListener

将String->Json->对象

线程切换

IHttpListener接口

onSuccess

(HttpEntity)

请求队列

ImageHttpService 实现execute 具体网络操作

FileHttpService 实现execute 具体网络操作

JsonHttpService 实现execute 具体网络操作

封装成IHttpService接口

抽象方法execute()

一般网络

请求过程:

设置URL

HttpPost.Post

HttpClient()

HttpEntity

获得数据

处理数据

回调

线程池

策略模式，由调用层传入策略：

备注：这是生产者消费者模式，属性动画也是应用了这种模式，见API 18 ValueAnimator源码。

MainActivity

第四章 数据库框架

第五章 图片加载框架

1，根据用户需求可以灵活配置 (建造者模式)

2，支持高并发，图片加载的优先级

3，支持可以选择不同的加载策略，对加载策略进行扩展

4，二级缓存 加载图片时内存中已经加载了，则从内存中加载，不存在去外置卡中加载，外置还不 存在则从网络下载，并对缓存策略可以扩展

1. 支持从加载过程中显示默认加载图片
2. 支持加载失败时显示默认错误图片
3. 图片显示自适应，从网络加载下来的图片经最佳比例压缩后显示
4. 不能失真变形
5. 支持请求转发，下载

用到的模式：

生产者消费者模式

建造者模式

单例模式

模板方法模式

策略模式

用到的知识点：

内存缓存LruCache技术

硬盘缓存技术 DiskLruCache技术

图片下载时请求转发

初始化

缓存策略配置

加载策略配置

线程个数

请求队列

其他配置

根据用户关注点，分为：

固定屏幕 🡪 “先进先加载”

滑动屏幕(GridView/ListView) 🡪“后进先加载”

不断从队列中获取请求

请求转发

在UI显示

使用缓存策略

获取对应的加载策略

解析图片地址(分为file:// 和 http://)

用户UI

将请求添加到请求队列

发起请求

转发器

(消费者)

从队列中拿请求进行消费

继承Tread，一直运行。

1.判断Url开头：http、https、file；

2.判断加载策略；

3.下载图片，完成后调用缓存策略：

🡪 内存缓存、硬盘缓存；

4.显示图片 🡪 显示正确/错误时的图片；

display(url, image) 🡪 生产者

BitmapRequest

(Url, ImageView, ImageListener)

请求队列

特点：阻塞式队列；线程安全；支持优先级。 🡪 PriorityBlockingQueue

第六章 RXJava

简洁

**链式调度**

**事件变换**