一 简介：

这是一个图片加载框架，图片纯粹，未做回收裁剪等动作，设计面向接口，扩展方便，带给你原生态的感觉。

二:支持功能：

支持加载本地图片

支持加载网络图片

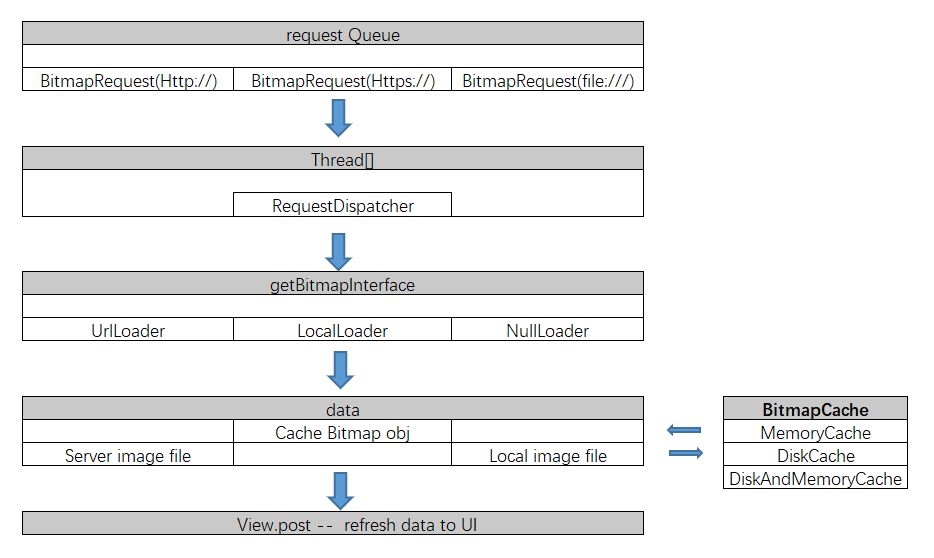
支持图片缓存到内存，缓存到硬盘，缓存到内存和硬盘，也可以不缓存

支持设置加载中图片，支持设置加载失败图片

支持设置执行任务进程数量（*建议不要大于处理器核数  
因为在处理器核数以内线程增长效率是增长，大于后由于存在任务调度等消耗，效率反而会下降）*

支持设置图片加载任务执行先后策略

三：流程说明 流程图如下：



四：使用示例

*//第一步：先检查运行时存储权限，没有就申请。百度一下你就知道，怎么申请。* checkStoragePermission();  
 *//第二步：初始化loader的配置* ImageLoaderConfig imageLoaderConfig = **new** ImageLoaderConfig()  
 *//设置加载失败的默认图片* .setFailImage(R.drawable.***fail***)  
 *//设置加载中的默认图片* .setLoadingImage(R.drawable.***loading***)  
 *//设置缓存策略* .setCache(**new** DiskAndMemoryCache(**this**))  
 *//设置图片加载策略，先加入的请求需要后执行* .setLoadPolicy(**new** ReversePolicy())  
 *//设置执行任务进程数量，建议不要大于处理器核数  
 //因为在处理器核数以内效率是增长，大于后由于存在任务调度等消耗，效率反而会下降* .setThreadCount(4);  
 *//第三步：初始化加载器* SimpleImageLoader.*getInstance*().init(imageLoaderConfig);  
 *//网络图片路径直接Uri即可* String pathTest = **"https://img12.360buyimg.com/red/jfs/t14662/206/484370755/204243/be01c580/5a2f8866N2a94ffe8.jpg"**;  
 *//本地图片示例，需要自己加上schema前缀file://  
// String pathTest2 = "file:///data/data/com.example.luozhenlong.myapplication/a.JPG";  
 //第四步：加载图片* SimpleImageLoader.*getInstance*().displayImage(**mImageview**,pathTest);