[xUtils框架的介绍（一）](http://blog.csdn.net/lanyeming2012/article/details/14043633)

 微信账号申请终于通过了，这是我们第一次Android干货分享。想来是第一次，要对得起“干货”二字。今天我要为大家推荐的是一个Android基于快速开发的一个框架——xUtils，它是在aFinal基础上进行重构和扩展的框架，相比aFinal有很大的改善。同时，如果如果你的应用是基于网络的，那么只要处理得当，它会让你彻底的摆脱各种工具类和重复代码的困扰。好的，废话不多说，程序猿们还是代码说起。

    xUtils 包含了很多实用的android工具。xUtils 源于Afinal框架，对Afinal进行了大量重构，使得xUtils支持大文件上传，更全面的http请求协议支持，拥有更加灵活的ORM，更多的事件注解支持且不受混淆影响。同时需要注意的是，xUitls最低兼容android 2.2 (api level 8)  。今天我们的主题是整体介绍下xUtils，主要介绍它重要的四大组件。具体各个组件的使用，会在之后几天陆续为大家奉上。下面开始：

一、ViewUtils

        你受够了重复冗长的findViewById了嘛？你受够了各种监听事件的绑定了嘛？在这里，你只需要一句注解，如@ViewInject、@OnClick，就能轻松摆脱小白似的代码，大大的上了一个档次。

二、HttpUtils

       支持的HTTP七种请求方式，非常便捷的满足你的接口请求的需要。同时还支持大文件上传下载，以及同步异步请求。

三、BitmapUtils

       你的程序因OOM强制关闭过嘛？你在为加载网络图片头疼嘛？有了组件，你将永久摆脱前面的问题。

四、DbUtils

       简单易用又出色的ORM框架，真的是谁用谁知道，直接轻松存储各种对象到sqlite数据库中，同时也能非常方便的进行各种条件查询，甚至分页查询，还有对表中数据的更新删除等操作，真正的实现。一行代码就可以进行增删改查。并且可通过注解自定义表名，列名，外键，唯一性约束，NOT NULL约束，CHECK约束等，支持事务。

    由于xUtils是基于aFinal的，这个开源框架是国内的某位大神写的，所以了解了aFinal之后再回头看xUtils，才会更有收获。同时，也要向这位大神以及众多的开源贡献者致敬，有了他们的奉献和开源的精神，才涌现出一个个耳熟能详的更加优秀的更加稳定的框架。我们众所周知的Linux就是这么诞生的。

# [xUtils框架的介绍（二）](http://blog.csdn.net/lanyeming2012/article/details/14043685)

昨天对xUtils整体上做了一个简单的介绍，今天咱们就代码码起，真刀实枪的也看看，看看如何快速便捷的把xUtils给集成到大家的项目中去。xUtils中有四大组件可以供我们使用，分别是ViewUtils、HttpUtils、BitmapUtils以及DbUtils。如果你没能先读一下我的[上一篇文章](http://blog.csdn.net/lanyeming2012/article/details/14043633)，那么请你移步过去先整体了解一下，再回过头来看这篇文章，相信你回更有体会的。

下面依次开始介绍这些组件具体的使用。

一、ViewUtils  android中得ioc（控制反转）框架，可以完全使用注解的方式来完成UI的绑定和事件绑定。简单的说，ViewUtils的功能就是做这个的，但是可以说，就这么个功能确是能极大的简化我们的代码。下面我们看下具体的代码,顺便对比下注解的方式绑定ID和findViewById之间的差别。

1. @ViewInject(R.id.btn)
2. **private** Button btn;
4. @ViewInject(R.id.img)
5. **private** ImageView img;
7. @ViewInject(R.id.list)
8. **private** ListView list;
10. @Override
11. **protected** **void** onCreate(Bundle savedInstanceState) {
12. **super**.onCreate(savedInstanceState);
13. setContentView(R.layout.activity\_second);
14. ViewUtils.inject(**this**);}

1. @Override
2. **protected** **void** onCreate(Bundle savedInstanceState) {
3. **super**.onCreate(savedInstanceState);
4. setContentView(R.layout.activity\_second);
6. btn = (Button) findViewById(R.id.btn);
7. img = (ImageView) findViewById(R.id.img);
8. list = (ListView) findViewById(R.id.list);
10. }

如果项目中得Activity中的控件相当多，那么想象一下代码中累积的那一大坨就真心受不了。通过xUtils的简单注解，就能轻松摆脱无尽坏味道的代码。

**注意：在使用注解绑定控件的时候，一定记得在onCreate中调用ViewUtils.inject(this);**

下面我们再对比下Android中事件绑定的区别。

1. @OnClick({ R.id.btn, R.id.img })
2. **public** **void** clickMethod(View v) {
3. Toast.makeText(SecondActivity.**this**, "you clicked button!",
4. Toast.LENGTH\_SHORT).show();
5. }
7. @OnItemClick(R.id.list)
8. **public** **void** itemClick(AdapterView<?> parent, View view, **int** position,**long** id) {
9. Toast.makeText(SecondActivity.**this**, "position--->" + position,
10. Toast.LENGTH\_SHORT).show();
11. }

1. btn.setOnClickListener(**this**);
2. list.setOnItemClickListener(**new** OnItemClickListener() {
3. @Override
4. **public** **void** onItemClick(AdapterView<?> parent, View view,
5. **int** position, **long** id) {
6. Toast.makeText(SecondActivity.**this**, "position--->" + position,
7. Toast.LENGTH\_SHORT).show();
8. }
9. });
10. @Override
11. **public** **void** onClick(View v) {
12. **switch** (v.getId()) {
13. **case** R.id.btn:
14. Toast.makeText(SecondActivity.**this**, "you clicked button!",
15. Toast.LENGTH\_SHORT).show();
16. **break**;
17. **default**:
18. **break**;
19. }
20. }

原本绑定Button的监听事件要么用丑陋的内部类，要么Activity实现OnClickListener，在复写的onClick方法中去根据id。而xUtils只要通过简单的一句注解就能实现监听事件的功能，而且可以实现多个控件共用一个监听方法。同时xUtils提供onClick、onItemClick、onLongClick等15种事件监听注解。

**注意：在使用注解监听事件的时候，监听方法名是自定义的，但是一定要保证方法的访问修饰符为public，同时方法的参数要与Android原来的监听方法参数一致，不仅参数类型，而且要保证参数的顺序。**

二、BitmapUtils  加载网络或本地bitmap的时候无需担心再遇到OOM的现象，管理bitmap的内存采用了LRU算法，同时也能避免列表滑动过程中发生图片错位等得现象。加载网络图片时，还可以配置运行线程的数量，缓存路径等。。。通过BitmapUtils的各种构造器，可以很方便的创建出本地缓存路径和缓存的大小，以及内存缓存的大小。

1. BitmapUtils utils = **new** BitmapUtils(**this**);
2. BitmapDisplayConfig config = **new** BitmapDisplayConfig(**this**);
3. config.setLoadingDrawable(getResources().getDrawable(R.drawable.loading));
4. config.setLoadFailedDrawable(getResources().getDrawable(R.drawable.failed));
5. config.setImageLoadCallBack(**new** ImageLoadCallBack() {
6. @Override
7. **public** **void** loadFailed(ImageView imageView, Drawable drawable) {
8. }
9. @Override
10. **public** **void** loadCompleted(ImageView imageView, Drawable drawable,
11. BitmapDisplayConfig config) {
12. }
13. });
14. config.setBitmapMaxWidth(480);
15. config.setBitmapMaxHeight(720);
16. /       utils.display(img, "http://img1.gtimg.com/news/pics/hv1/63/26/1451/94357968.jpg");
17. utils.display(img, "http://img1.gtimg.com/news/pics/hv1/63/26/1451/94357968.jpg", config);

BitmapUtils在用来加载网络图片时，可以配置加载图片尺寸的大小，加载成功和失败的回调以及加载过程中图片的配置。同时也可以选择不配置。

1. //bitmapUtils.display(testImageView, "/sdcard/test.jpg"); //支持加载本地图片
3. // 使用ListView等容器展示图片时可通过PauseOnScrollListener控制滑动和快速滑动过程中时候暂停加载图片
4. listView.setOnScrollListener(**new** PauseOnScrollListener(bitmapUtils, **false**, **true**));
5. listView.setOnScrollListener(**new** PauseOnScrollListener(bitmapUtils, **false**, **true**), customListener);

明天再讲剩下的两个组件：DbUtils和HttpUtils。

[xUtils框架的介绍（三）](http://blog.csdn.net/lanyeming2012/article/details/14147975)

接上回，继续介绍xUtils的最后两个模块：DbUtils和HttpUtils。首先先介绍第一个SQLite数据库操纵的简单ORM框架，只要能理解xUtils为我们提供的api,相信你也能熟练的把DbUtils用到项目中去。

操纵数据库的工具类，无论多牛X，总离不开最根本的crud，即创建，查询，更新和删除。下面从这四个角度依次介绍xUtils是如何简便持久化数据的。大家都知道，在Android里面如果要存储一个对象，我们需要创建一个SQLiteOpenHelper，然后还得创建一张对应对象各个属性的表，还得继续把我们的对象转换成ContentValues，进而去存储。真心是麻烦的不能再麻烦了，我们现在介绍的DbUtils就能让你轻松解脱麻绳一样的代码。DbUtils在进行save操作的时候，会根据java反射反射出对象的各个字段，然后去查询数据库中是否存在这个对象类型对应的表，如果表已经存在，直接进行插入操作；如果不存在，就先动态的创建的一张对应我们对象的表，再进行插入处理。直接上代码，大家看。

1. @OnClick(R.id.insert)
2. **public** **void** insert(View v) {
3. Student stu = **null**;
4. **for** (**int** i = 0; i < 20; i++) {
5. stu = **new** Student();
6. stu.setAge(10 + i);
7. stu.setName("jack" + i);
8. mList.add(stu);
9. **try** {
10. dbUtils.save(stu);
11. } **catch** (DbException e) {
12. e.printStackTrace();
13. }
14. }
15. }

**注意：并不是所有的实体对象都快可以通过这种方式去存储，一定要保证对象的类型中有int类型的id或者\_id的属性，这就对应数据库表中的主键字段。如果类型中没有id字段，可以通过@Id注解去指定一个int类型的字段作为主键。如果表中的又字段不想被存储在数据库中，也可以通过@Transient去实现忽略。如果直接存储一个对象的列表，这样也是被允许的，达到批量存储的目的。**

DbUtils可以帮助对SQL语句不是很熟悉的同学快速的实现查询，而不用去写sql查询语句，而且可以对查询结果进行排序和分页，使用简单，功能强大。大家可以看下，下面的几行代码就能实现复杂的查询功能

1. dbUtils.findAll(Selector.from(Student.**class**)
2. .where("\_id", "<", 10).and("age", ">", 10).orderBy("\_id")
3. .limit(pageSize).offset(pageSize \* pageIndex));

同样的，也可以对数据库中得数据进行便捷的更新。下面演示的是更新Student对应的表中的第一条记录的age这个字段。这个比较简单，就直接上代码看吧。

1. @OnClick(R.id.update)
2. **public** **void** update(View v){
3. **try** {
4. List<Student> stus = dbUtils.findAll(Selector.from(Student.**class**));
5. Student stu = stus.get(0);
6. stu.setAge(20);
7. dbUtils.update(stu);
8. } **catch** (DbException e) {
9. e.printStackTrace();
10. }
11. }

最后一个就是数据的删除。一个实体对象，一组实体对象，根据条件删除，删除表，删除整个数据库，这些操作都可以通过一句简单的代码来实现，看了代码就明白了。

1. @OnClick(R.id.delete)
2. **public** **void** delete(View v){
3. **try** {
4. List<Student> stus = dbUtils.findAll(Selector.from(Student.**class**));
5. dbUtils.delete(stus.get(0));
6. dbUtils.deleteAll(stus);
7. dbUtils.deleteById(Student.**class**, WhereBuilder.b("age", "==", 20));
8. dbUtils.dropTable(Student.**class**);
9. dbUtils.dropDb();
10. } **catch** (DbException e) {
11. e.printStackTrace();
12. }
13. }

[xUtils框架的介绍（四）](http://blog.csdn.net/lanyeming2012/article/details/14209455)

今天介绍xUtils的最后一个模块——HttpUtils，拖了那么久，终于要结束了。另外，码字不易，如果大家有什么疑问和见解，欢迎大家留言讨论。HttpUtils是解决日常工作过程中繁杂的上传下载文件以及各种Get和post请求的必备工具类，通过这个类，开源非常方便关注接口的业务，不必再再写那么冗长的代码。下面全文都围绕着四个中心点去依次展开，分别是：HttpGet请求,HttpPost请求，下载文件和上传文件。

**HttpGet请求**。相信从事Android网络应用的开发的同学对这个一定不会陌生，长期的开发会让大家积累下一个可以复用的工具类。而xUtils就是帮助了我们把那些工具类给抽象整合成一个更具扩展性的帮助类。比如HtttpGet请求，这里只简要介绍下关键的应用方法，具体的细节还请大家自己去下载xUtils的源码去研究研究了，我们介绍的足够应用到我们的应用开发中去了。通常的HttpGet请求会把一系列的请求参数挂在请求地址的后面，拖出一节长长的尾巴，十分的惹人厌，这里可以通过像HttpPost请求的参数那样组成一个类似BasicNameValuePair的对象直接封装到请求方法中去，省去拼接url的麻烦，同时也可以设置超时时间。另外请求的方法中提供了一个回调类，这个类中有处理不同请求结果的回调方法，比如说请求过程中的回调，请求成功的回调以及请求出现错误时的回调。下面直接上代码看下。

1. RequestParams params = **new** RequestParams();
2. params.addQueryStringParameter("method", "info");
3. params.addQueryStringParameter("access\_token",
4. "3.1042851f652496c9362b1cd77d4f849b.2592000.1377530363.3590808424-248414");
6. HttpUtils http = **new** HttpUtils();
7. http.configCurrentHttpGetCacheExpiry(1000 \* 10);
8. http.send(HttpRequest.HttpMethod.GET,  "https://pcs.baidu.com/rest/2.0/pcs/quota",   params, **new** RequestCallBack<String>() {
9. @Override
10. **public** **void** onStart() {
11. resultText.setText("conn...");
12. }
14. @Override
15. **public** **void** onLoading(**long** total, **long** current) {
16. resultText.setText(current + "/" + total);
17. }
19. @Override
20. **public** **void** onSuccess(String result) {
21. resultText.setText("response:" + result);
22. }

25. @Override
26. **public** **void** onFailure(HttpException error, String msg) {
27. resultText.setText(msg);
28. }
29. });

**HttpPost请求。**为了统一请求的风格，HttpPost请求的方式和HttpGet的几乎可以说是一模一样，一样提供了各种对应不同结果的回调方法，大家自己看下面的代码就晓得了，真是一目了然。

1. RequestParams params = **new** RequestParams();
2. params.addQueryStringParameter("method", "mkdir");
3. params.addQueryStringParameter("access\_token", "3.1042851f652496c9362b1cd77d4f849b.2592000.1377530363.3590808424-248414");
4. params.addBodyParameter("path", "/apps/测试应用/test文件夹");
6. HttpUtils http = **new** HttpUtils();
7. http.send(HttpRequest.HttpMethod.POST,
8. "https://pcs.baidu.com/rest/2.0/pcs/file",
9. params,
10. **new** RequestCallBack<String>() {
12. @Override
13. **public** **void** onStart() {
14. resultText.setText("conn...");
15. }
17. @Override
18. **public** **void** onLoading(**long** total, **long** current) {
19. resultText.setText(current + "/" + total);
20. }
22. @Override
23. **public** **void** onSuccess(String result) {
24. resultText.setText("upload response:" + result);
25. }

28. @Override
29. **public** **void** onFailure(HttpException error, String msg) {
30. resultText.setText(msg);
31. }
32. });

下面为大家介绍一个非常非常实用的功能，就是通过Http协议去下载文件，再也不必为Android中下载文件而写下一大坨一大坨的代码，如此长的代码调试起来真是能气死人。还记得以前做过一个项目，有个需求就是能下载视频的，而且还要能支持断点下载，只写那么一个下载工具类就写了一两天，再加上调试，真心把人都给逼疯了。要是xUtils早点在那个时候面世，想那时做那个需求也不必那么通过。HttpUtils为开发者提供了非常方便的下载api，可以通过简单的几个参数来实现下载，甚至断点下载的功能。上代码。

1. HttpHandler  handler = http.download( downloadAddrEdit.getText().toString(),   "/sdcard/fileexplorer.apk",

**true**, // 如果目标文件存在，接着未完成的部分继续下载。

1. **true**, // 如果从请求返回信息中获取到文件名，下载完成后自动重命名。
2. **new** RequestCallBack<File>() {
4. @Override
5. **public** **void** onStart() {
6. resultText.setText("conn...");
7. }
9. @Override
10. **public** **void** onLoading(**long** total, **long** current) {
11. resultText.setText(current + "/" + total);
12. }
14. @Override
15. **public** **void** onSuccess(File result) {
16. resultText.setText("downloaded:" + result.getPath());
17. }
19. @Override
20. **public** **void** onFailure(HttpException error, String msg) {
21. resultText.setText(error.getExceptionCode() + ":" + msg);
22. }
23. });

**注意：下载过程中如果需要暂停下载，也只需简单的一行代码来实现：mHandler.stop()，如果设置断点下载的话，下次会重新开始的话，会自动从上次下载的断点处继续下载。**

最后介绍的功能就是上传文件了，这个也是在项目中也是非常常见的。比如用户上传头像，再比如网盘应用需要把本地文件上传到云端等等。同时HttpUtils也同时为开发者提供了上传过程中和上传结果的各个回调接口。大家在使用HttpUtils上传文件的时候，只要仿照下面的代码去码代码就快可以基本满足业务的需要了。

1. <strong> </strong>RequestParams params = **new** RequestParams();
2. params.addQueryStringParameter("method", "upload");
3. params.addQueryStringParameter("path", "/apps/测试应用/test.zip");
4. // 请在百度的开放access\_tokenapi测试页面找到自己的access\_token
5. params.addQueryStringParameter("access\_token", "3.1042851f652496c9362b1cd77d4f849b.2592000.1377530363.3590808424-248414");
6. params.addBodyParameter("file", **new** File("/sdcard/test.zip"));
8. HttpUtils http = **new** HttpUtils();
9. http.send(HttpRequest.HttpMethod.POST,
10. "https://pcs.baidu.com/rest/2.0/pcs/file",
11. params,
12. **new** RequestCallBack<String>() {
14. @Override
15. **public** **void** onStart() {
16. resultText.setText("conn...");
17. }
19. @Override
20. **public** **void** onLoading(**long** total, **long** current) {
21. resultText.setText(current + "/" + total);
22. }
24. @Override
25. **public** **void** onSuccess(String result) {
26. resultText.setText("upload response:" + result);
27. }

30. @Override
31. **public** **void** onFailure(HttpException error, String msg) {
32. resultText.setText(msg);
33. }
34. });

好了，xUtils的介绍就到此为止了，后面我会为大家介绍点Java基础，Java内存处理以及国外广受欢迎的Android各种的框架，欢迎大家的关注。如果觉得对你有所帮助，欢迎大家订阅我的微信公众账号——Android干货分享