[XUtils3框架的基本使用方法(一)](http://blog.csdn.net/a1002450926/article/details/50341173)

今天给大家带来XUtils3的基本介绍，本文章的案例都是基于XUtils3的API语法进行的演示。相信大家对这个框架也都了解过，下面简单介绍下XUtils3的一些基本知识。   
XUtils3一共有4大功能：注解模块，网络模块，图片加载模块，**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/a1002450926/article/details/_blank)**模块。   
使用XUtils只需要在libs文件夹里面加入一个jar包，如果对服务器返回的数据进行封装的话，还需要导入一个Gson的jar包。   
所需权限:

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

### 注解模块

Activity的注解   
1.在Application的oncreate方法中加入下面代码:   
x.Ext.init(this);   
2.在Activity的oncreate方法中加入下面代码:   
x.view().inject(this);   
3.加载当前的Activity布局需要如下注解:   
@ContentView加入到Activity的上方   
4.给View进行初始化需要如下注解:   
@InjectView   
5.处理控件的各种响应事件需要如下注解:   
@Event   
示例如下:

@ContentView(R.layout.activity\_main)

public class MainActivity extends ActionBarActivity {

@ViewInject(R.id.btn\_get)

Button btn\_get;

@ViewInject(R.id.btn\_post)

Button btn\_post;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

x.view().inject(this);

btn\_get.setText("发送get请求");

btn\_post.setText("发送post请求");

}

//等同于@Event(value={R.id.btn\_get,R.id.btn\_post},type=View.OnClickListener.class)

@Event(value={R.id.btn\_get,R.id.btn\_post})

private void getEvent(View view){

switch(view.getId()){

case R.id.btn\_get:

Toast.makeText(MainActivity.this, btn\_get.getText().toString().trim(), 0).show();

break;

case R.id.btn\_post:

Toast.makeText(MainActivity.this, btn\_post.getText().toString().trim(), 0).show();

break;

}

}

通过**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/a1002450926/article/details/_blank)**，我们发现当点击btn\_get按钮时，弹出了“发送get请求”的效果。到这里，大家算是对XUtils的注解有个一个基本的认识，下面我在强调一点的是，@Event这个注解默认情况下type属性为View.OnClickListener.class.如果你想实现其余点击事件效果，只需要把type值进行修改即可。   
另外需要注意的一点是按钮的点击事件必须用private进行修饰。

Fragment的注解:

@ContentView(R.layout.fragment\_first)

public class FirstFragment extends Fragment{

private MyAdapter adapter;

private List<Person> list=new ArrayList<>();

private List<String> listUrl=new ArrayList<>();

private List<String> listName=new ArrayList<>();

@ViewInject(R.id.btn\_test)

Button btn\_test;

@ViewInject(R.id.listView)

ListView listView;

@Override

public View onCreateView(LayoutInflater inflater,

@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

return x.view().inject(this, inflater, container);

}

ViewHolder的注解:

public class MyAdapter extends BaseAdapter{

private Context context;

private List<Person> list;

private LayoutInflater mInflater;

private ImageOptions options;

public ViewHolder holder;

public MyAdapter(Context context, List<Person> list) {

this.context = context;

this.list = list;

this.mInflater=LayoutInflater.from(context);

options=new ImageOptions.Builder().setLoadingDrawableId(R.drawable.ic\_launcher)

.setLoadingDrawableId(R.drawable.ic\_launcher).setUseMemCache(true).setCircular(true).build();

}

@Override

public int getCount() {

return list.size();

}

@Override

public Object getItem(int position) {

return list.get(position);

}

@Override

public long getItemId(int position) {

return position;

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

holder=null;

if(convertView==null){

convertView=mInflater.inflate(R.layout.itemone, null);

holder=new ViewHolder();

**x.view().inject(holder,convertView);**

convertView.setTag(holder);

}

else{

holder=(ViewHolder) convertView.getTag();

}

Person bean=list.get(position);

holder.tv\_name.setText(bean.getName());

x.image().bind(holder.iv\_image, bean.getImgUrl(), options);

return convertView;

}

class ViewHolder{

**@ViewInject(R.id.tv\_name)**

private TextView tv\_name;

**@ViewInject(R.id.iv\_image)**

private ImageView iv\_image;

}

上面的代码是注解ViewHolder的标准模式，相信大家完全能够看明白。这里我加载图片使用到了XUtils3的加载网络图片的方法。后面我会详细进行讲解。

### 网络模块

XUtils的网络请求方法和一些网络请求框架的用法非常类似，我进行了一些简单的封装。   
封装一共有3个文件，分别是网络请求工具类XUtil,请求响应数据的解析类，还有一个就是一个请求成功的一个接口回调类。   
代码如下:

public class XUtil {

/\*\*

\* 发送get请求

\* @param <T>

\*/

public static <T> Cancelable Get(String url,Map<String,String> map,CommonCallback<T> callback){

RequestParams params=new RequestParams(url);

if(null!=map){

for(Map.Entry<String, String> entry : map.entrySet()){

params.**addQueryStringParameter**(entry.getKey(), entry.getValue());

}

}

Cancelable cancelable = x.http().get(params, callback);

return cancelable;

}

/\*\*

\* 发送post请求

\* @param <T>

\*/

public static <T> Cancelable Post(String url,Map<String,Object> map,CommonCallback<T> callback){

RequestParams params=new RequestParams(url);

if(null!=map){

for(Map.Entry<String, Object> entry : map.entrySet()){

params.addParameter(entry.getKey(), entry.getValue());

}

}

Cancelable cancelable = x.http().post(params, callback);

return cancelable;

}

/\*\*

\* 上传文件

\* @param <T>

\*/

public static <T> Cancelable UpLoadFile(String url,Map<String,Object> map,CommonCallback<T> callback){

RequestParams params=new RequestParams(url);

if(null!=map){

for(Map.Entry<String, Object> entry : map.entrySet()){

params.addParameter(entry.getKey(), entry.getValue());

}

}

params.**setMultipart**(true);

Cancelable cancelable = x.http().post(params, callback);

return cancelable;

}

/\*\*

\* 下载文件

\* @param <T>

\*/

public static <T> Cancelable DownLoadFile(String url,String filepath,CommonCallback<T> callback){

RequestParams params=new RequestParams(url);

//设置断点续传

params.**setAutoResume**(true);

params.**setSaveFilePath**(filepath);

Cancelable cancelable = x.http().get(params, callback);

return cancelable;

}

}

public class JsonResponseParser implements ResponseParser {

//检查服务器返回的响应头信息

@Override

public void checkResponse(UriRequest request) throws Throwable {

}

/\*\*

\* 转换result为resultType类型的对象

\*

\* @param resultType 返回值类型(可能带有泛型信息)

\* @param resultClass 返回值类型

\* @param result 字符串数据

\* @return

\* @throws Throwable

\*/

@Override

public Object parse(Type resultType, Class<?> resultClass, String result) throws Throwable {

return new Gson().fromJson(result, resultClass);

}

}

public class MyCallBack<ResultType> implements Callback.CommonCallback<ResultType>{

@Override

public void onSuccess(ResultType result) {

//可以根据公司的需求进行统一的请求成功的逻辑处理

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

//可以根据公司的需求进行统一的请求网络失败的逻辑处理

}

@Override

public void onCancelled(CancelledException cex) {

}

@Override

public void onFinished() {

}

}

1.发送get请求示例:

//本get请求来自于免费测试接口：<http://www.k780.com/api/entry.baidu>

String url="http://api.k780.com:88/?app=idcard.get";

Map<String,String> map=new HashMap<>();

map.put("appkey", "10003");

map.put("sign", "b59bc3ef6191eb9f747dd4e83c99f2a4");

map.put("format", "json");

map.put("idcard", "110101199001011114");

XUtil.Get(url, map, new MyCallBack<PersonInfoBean>(){

@Override

public void onSuccess(PersonInfoBean result) {

super.onSuccess(result);

Log.e("result", result.toString());

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

});

2.发送post请求

String url="http://api.k780.com:88/?app=idcard.get";

Map<String,Object> map=new HashMap<>();

map.put("appkey", "10003");

map.put("sign", "b59bc3ef6191eb9f747dd4e83c99f2a4");

map.put("format", "json");

map.put("idcard", "110101199001011114");

XUtil.Post(url, map, new MyCallBack<PersonInfoBean>(){

@Override

public void onSuccess(PersonInfoBean result) {

super.onSuccess(result);

Log.e("result", result.toString());

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

});

3.上传文件

/\*\*

\* 上传文件(支持多文件上传)

\*/

private void uploadfile() {

//图片上传地址

String url="";

Map<String,Object> map=new HashMap<>();

//传入自己的相应参数

//map.put(key, value);

//map.put(key, value);

XUtil.UpLoadFile(url, map, new MyCallBack<String>(){

@Override

public void onSuccess(String result) {

super.onSuccess(result);

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

});

}

4.下载文件

private void downloadfile() {

//文件下载地址

String url="";

//文件保存在本地的路径

String filepath="";

XUtil.DownLoadFile(url, filepath,new MyCallBack<File>(){

@Override

public void onSuccess(File result) {

super.onSuccess(result);

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

});

5.下载文件带进度条

private void downloadprogressfile() {

//文件下载地址

String url="";

//文件保存在本地的路径

String filepath="";

XUtil.DownLoadFile(url, filepath,new MyProgressCallBack<File>(){

@Override

public void onSuccess(File result) {

super.onSuccess(result);

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

@Override

public void onLoading(long total, long current,

boolean isDownloading) {

super.onLoading(total, current, isDownloading);

}

});

}

6.发送get请求(服务器以xml格式返回)

private void getxml() {

String url="http://flash.weather.com.cn/wmaps/xml/china.xml";

XUtil.Get(url, null, new MyCallBack<String>(){

@Override

public void onSuccess(String xmlString) {

super.onSuccess(xmlString);

try{

XmlPullParserFactory factory = XmlPullParserFactory.newInstance();

XmlPullParser xmlPullParser = factory.newPullParser();

xmlPullParser.setInput(new StringReader(xmlString));

int eventType = xmlPullParser.getEventType();

while (eventType != XmlPullParser.END\_DOCUMENT) {

switch (eventType) {

case XmlPullParser.START\_TAG:

String nodeName = xmlPullParser.getName();

if ("city".equals(nodeName)) {

String pName = xmlPullParser.getAttributeValue(0);

Log.e("TAG", "city is " + pName);

}

break;

}

eventType = xmlPullParser.next();

}

}catch(Exception e){

e.printStackTrace();

}

}

@Override

public void onError(Throwable ex, boolean isOnCallback) {

super.onError(ex, isOnCallback);

}

});

}

### 

### 图片加载模块

用法:

x.image().bind(imageView, url, imageOptions);

x.image().bind(imageView, "file:///sdcard/test.gif", imageOptions);

x.image().bind(imageView, "assets://test.gif", imageOptions);

x.image().bind(imageView, url, imageOptions, new Callback.CommonCallback<Drawable>() {...});

x.image().loadDrawable(url, imageOptions, new Callback.CommonCallback<Drawable>() {...});

x.image().loadFile(url, imageOptions, new Callback.CommonCallback<File>() {...});

Xutils3的api还是比较简单的,相信大家都可以看懂，第一个参数传入一个View，第二个参数传入一个图片的网络地址，第三个参数一般是加载图片的配置。   
下面看一下ImageOptions这个类吧:

ImageOptions options=new ImageOptions.Builder()

//设置加载过程中的图片

.setLoadingDrawableId(R.drawable.ic\_launcher)

//设置加载失败后的图片

.setFailureDrawableId(R.drawable.ic\_launcher)

//设置使用缓存

.setUseMemCache(true)

//设置显示圆形图片

.setCircular(true)

//设置支持gif

.setIgnoreGif(false).build();

其余的一些配置请参考源码

如果需要对加载的图片进行操作的话，可以使用:

x.image().loadDrawable(url, imageOptions, new Callback.CommonCallback<Drawable>() {...});

通过返回的Drawable对象进行图片的处理，满足项目的个性化的要求.