Volley的使用

开源库之volley volley 异步网络请求 图片加载框架

Volley简介和特点

- 从并发、效率和性能上都具有较高水准的框架
- Android平台上的网络通信请求库
  +通信更快，更简单，可以提高开发效率，提高模块的的稳定性
- Get Post网路请求及网络图像的高效异步处理请求
- 对网络请求优先级进行排序
- 网络请求缓存
- 多级别取消请求
- 和Activity生命周期联动避免退出程序后还进行网络请求，可提高App性能和用户体验度
- 不适合数据的上传和下载，当有该需求时要使用其他框架

为什么使用Volley

功能上

- 高效的Get/Post方式的数据请求交互
- 网络图片加载和缓存

Markdown 语法

其他

- 谷歌官方推荐
- 性能稳定和强劲

Volley框架的使用

Volley的get和post请求方式的使用

- Get和Post请求接口数据的使用：
  首先挑选合适的对象：StringRequest(适合于返回数据不确定类型时)、JsonObjectRequest(返回数据类型为JsonObject)、JsonArrayRequest(返回数据为JsonArray)
  1.Get请求

a.StringRequest

```
private void volley_Get(){
    String url="";
    StringRequest request=new StringRequest(Method.GET,url,new Lister
<String>(){
    @Override
    public void onResponse(String arg0){
        //成功时调用
                Toast.makeText(this,arg0,Toast.LENGTH_LONG).show();
    }
    },new Resonse.ErrorListener(){
    @Override
    public void onErrorResponse(VolleyError arg0){
        //请求失败时调用
        Toast.makeText(this,arg0.toString(),Toast.LENGTH_LONG).show();
    }
    }
    );
    request.setTag("设置一个Tag标签");//f方便在请求队列中查找
    MyApplication.getHttpQueues().add(request);//将请求加入到全局队列中

}
```

b.JsonObjectRequest

```
private void volley_Get(){
    String url="";
    //JsonObjectRequest 的构造方法有5个参数
Method.GET,url,JsonRequest,lister,error;其中第三个参数JsonRequest为附带请求
参数，已经携带到Url中（因为是get请求，若为post请求则还需设置），故此处设为null
    sonObjectRequest request=new
JsonObjectRequest(Method.GET,url,null,new Lister <JsonObject>(){
    @Override
    public void onResponse(JsonObject arg0){
        //成功时调用
                Toast.makeText(this,arg0,Toast.LENGTH_LONG).show();
    }
    },new Resonse.ErrorListener(){
    @Override
    public void onErrorResponse(VolleyError arg0){
        //请求失败时调用
        Toast.makeText(this,arg0.toString(),Toast.LENGTH_LONG).show();
    }
    }
    );
    request.setTag("设置一个Tag标签");//f方便在请求队列中查找
    MyApplication.getHttpQueues().add(request);//将请求加入到全局队列中
}
```

2.Post请求

a.StringRequest

```
public void volley_Post(){
    String url="http://xxxxxxxx.xxxxx?"}
    StringRequest request=new StringRequest(Method.POST,url,new Lister
<String>(){
    @Override
    public void onResponse(String arg0){
        //成功时调用
                Toast.makeText(this,arg0,Toast.LENGTH_LONG).show();
    }
    },new Resonse.ErrorListener(){
    @Override
    public void onErrorResponse(VolleyError arg0){
        //请求失败时调用
        Toast.makeText(this,arg0.toString(),Toast.LENGTH_LONG).show();
    }
    }
    ){
    @Override
    protected Map<String,String> getParams()throws AuthFailureError{
    //将传递的参数添加到Map中
    Map<String,String> hashMap=new HashMap<String,String>();
    hashMap.put("name","qdh");
    hashMap.put("age","23");
    return hashMap;
    }};
    request.setTag("设置一个Tag标签");//f方便在请求队列中查找
    MyApplication.getHttpQueues().add(request);//将请求加入到全局队列中

}
```

b. JsonObjectRequest

```
private void volley_Post(){
    String url="";
    HashMap<String,String> map=new HashMap<String.String>();
    map.put("name","qdh");
    map.put("age","23");
JsonObject jsonrequest=new JsonObject(map);
//jsonRequest为url请求携带参数
    sonObjectRequest request=new
JsonObjectRequest(Method.GET,url,jsonRequest,new Lister <JsonObject>(){
    @Override
    public void onResponse(JsonObject arg0){
        //成功时调用
                Toast.makeText(this,arg0,Toast.LENGTH_LONG).show();
    }
    },new Resonse.ErrorListener(){
    @Override
    public void onErrorResponse(VolleyError arg0){
        //请求失败时调用
        Toast.makeText(this,arg0.toString(),Toast.LENGTH_LONG).show();
    }
    }
    );
    request.setTag("设置一个Tag标签");//f方便在请求队列中查找
    MyApplication.getHttpQueues().add(request);//将请求加入到全局队列中
}
```

- 回调的使用：
  如请求成功，请求失败，请求重试

Volley的网络请求队列建立和取消队列请求

- 建立请求队列
  首先要建立全局的请求队列，然后将请求加入到全局队列
  1.在Application建立全局的请求队列

```
//要将该Application添加到清单文件中，同时加上网络请求权限
public class MyApplication extends Applicatin{
    public staric RequestQueue queues;

    public void onCreate(){
    super.onCreate();
    queues=Volley.newRequestQueue(getApplicationContext)
    }

    public static RequestQueue getHttpQueues(){
    return queues
    }
}
```

- 取消某个请求

Volley与Activity生命周期的联动

- 可以在Activityx销毁的时候，同时关闭请求
- 和Activity生命周期关联时需要设置Tag标签(便于查找)，onStop()里执行取消请求

```
@Override
protected void onStop(){
    super.onStop();
    MyApplication.getHttpQueues().cancleAll("XxxxTag")//设置的Tag标签
    }
```

Volley的简单的二次回调封装

```
public calss VolleyRequest{

    public static StringRequest stringRequest;
    public static Context context;
     //Get请求
     //第四个参数为请求成功或者请求失败的回调接口
     public static void RequestGet(Context mContext,String url,String
tag,VolleyInterface vif){
         MyApplication.getHttpQueues().cancaleAll(tag);//在请求之前先取消
该tag对应的请求，已免造成不必要的系统资源消耗
         stringRequest=new
StringRequest(Method.GET,url,vif.loadingListener(),vif.erroLisener())
         stringRequest。setTag(tag);
         MyApplication.getHttpQueues().add(stringRequest);
         MyApplication.getHttpQueues().start();
     }
     //Post请求
     public static void RequestPost(Context mContext,String url,String
tag,final Map<String,String> params,VolleyInterface vif){

    MyApplication.getHttpQueues().cancaleAll(tag);//在请求之前先取消该tag
对应的请求，已免造成不必要的系统资源消耗
         stringRequest=new
StringRequest(Method.POST,url,vif.loadingListener(),vif.erroLisener()){
         @Override
         protected Map<String,String> getparams() throw
AuthFailureError{
         return params;
         }
         };
         stringRequest.setTag(tag);
         MyApplication.getHttpQueues().add(stringRequest);
         MyApplication.getHttpQueues().start();
     }


}
```

建立一个抽象类来封装请求成功或失败

```
public adstract class VolleyInterface{
    public Context mContext;
    public static Lisener<String> mLisener;
    public ststic ErrorLisener mEeeorLisener;
    public VolleyInterface(Context mContext,Lisener<String>
lisener,ErrorLisener errorLisener){
    this.mContext=context;
    this.mErrorLisener=errorLisener;
    this.mLisener=lisener;
    }
    public abstract void onMySuccess(String result);
    public abstract void onMyError(VolleyError error);
    //监听请求成功的方法
    public Listen<String> loadingListener(){
        mLiener=new Lisener<String>(){
        @Override
        public void onResponse(String arg0){
            //弹出加载中
            onMySuccess(arg0);
        }
        };
        return mLisener;
    }
    //监听请求失败的方法
    public Errorlisener errorlisener(){
        mErroelisener=new ErrorLisener(){
            @Override
            public void onErrorResponse(VolleyError arg0){
            onMyErro(arg0);
            //提示请求失败
            }};
            return mErrorLisener;
    }
}
```

二次封装后的使用

```
private void volley_Get(){
    String url="";
    VolleyRequest.RequestGet(this,url,"abcTag",new
VolleyInterface(this,VolleyINterface.mLisener,VolleyInterface.mErrorLis
ener){
    @Override
    public void onMySuccess(String result){
    //请求成功
    Toast.makeText(this,result,Toast.LENGTH_LONG).show();
    }
    @Override
    public void onMyError(VolleyError error){
    //请求失败
    Toast.makeText(this,arg0.toString(),Toast.LENGTH_LONG).show();
    }})
}
```

Volley加载图片的用法

缓存功能简单介绍

- LruCache
- ImageCache

```
public class BitmapCache implements ImageLoader.ImageCache {

    public BitmapCache() {
        cache = new LruCache<String, Bitmap>(max) {
            @Override
            protected int sizeOf(String key, Bitmap value) {
                return value.getRowBytes() * value.getHeight();
            }
        };

    }

    public LruCache<String, Bitmap> cache;
    public int max = 10 * 1024 * 1024;//定义最大缓存量为10M

    @Override
    public Bitmap getBitmap(String url) {

        return cache.get(url);
    }

    @Override
    public void putBitmap(String url, Bitmap bitmap) {
        cache.put(url, bitmap);
    }
  }
```

加载网络图片及监听

- ImageRequest

```
/**
     * 通过imageRequest来加载网络图片
     */
    public void Volley_imgReq() {
        ImageRequest imageRequest = new ImageRequest(url, new
Response.Listener<Bitmap>() {
            @Override
            public void onResponse(Bitmap response) {
                imageViewReq.setImageBitmap(response);
            }
        }, 0, 0, Bitmap.Config.RGB_565, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                imageViewReq.setImageResource(R.mipmap.ic_launcher);//
加载默认图片
            }
        });
        MyApplication.getHttpQueues().add(imageRequest);
    }
```

- ImageLoader

```
/**
     * 通过imageRequest来加载网络图片
     */
    public void Volley_imgReq() {
        ImageRequest imageRequest = new ImageRequest(url, new
Response.Listener<Bitmap>() {
            @Override
            public void onResponse(Bitmap response) {
                imageViewReq.setImageBitmap(response);
            }
        }, 0, 0, Bitmap.Config.RGB_565, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                imageViewReq.setImageResource(R.mipmap.ic_launcher);//
加载默认图片
            }
        });
        MyApplication.getHttpQueues().add(imageRequest);
    }
```

- NetworkImageView

```
    /**
     * 通过NetWorkImageView
     */
    public void volley_netWorkImageView(){


        ImageLoader imageLoader = new
ImageLoader(MyApplication.getHttpQueues(),
                new BitmapCache());
        networkImageView.setDefaultImageResId(R.mipmap.ic_launcher);
        networkImageView.setErrorImageResId(R.mipmap.ic_launcher);
        networkImageView.setImageUrl(url,imageLoader);
    }
```

在xml文件中

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/iv_volley_Req"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <com.android.volley.toolbox.NetworkImageView
        android:id="@+id/iv_network"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

在代码中声明绑定控件

```
    @Bind(R.id.iv_volley_Req)
    ImageView imageViewReq;

    @Bind(R.id.iv_network)
    NetworkImageView networkImageView;
```

在onCreate()方法中绑定

```
Butterknife.bind(this);
```

Demo

volley示例

更多了解

Volley源码解析