Android课设报告-Android博客阅读App

1.选题的目的与意义

1.1 面临的问题

随着CSDN，掘金，开源中国，知乎，简书，微信公众号，博客园等多个博客平台的发展，以及部分有名有水平的开发者的个人微信公众号与个人博客网站的发展，博客平台的发展可谓是百花齐放，这使得Android开发人员与程序设计初学者有十分丰富的渠道与途径去了解或学习相关技术知识。但从另一个方面来说，平台很多，文章和博客也很多，也带来了以下问题：

- 需要花费大量时间与精力去寻找优质博客文章与优质开源项目
- 大部分的文章都是零零散散，个人难以构成知识体系的整理博客
- 各个平台收藏的文章难以统一整理
- 对于个人技术层面的拓展与深入难以开展，缺乏知识扩散与小方面专攻的文章收集

1.2 开发目的与意义

Android博客阅读App就是为了解决上述一系列由于博客平台很多的问题所开发，集成上述的所有博客平台的优质博客与开源项目的资源，并将资源分门别类的整理。此外还可以收藏自己喜欢的博客，达到一个账号，收集所有平台的文章与开源项目的效果。

1.3 需完成的功能

通过使用 玩安卓网站官方开放的部分API,实现如下功能：

- 实现与自行搭建后端交互，包括登录，注册，获取指定用户的所有收藏的文章，删除一篇指定用户所收藏的文章。
- 首页实现每日文章推荐与常用导航Tab轮播图
- 导向页面实现知识体系，实现按照知识体系查找文章，比如开发环境，四大组件，常用控件，网络访问，图片加载，数据存储等44类博客文章。
- 导向页面实现拓维导航，收集了安卓开发学习中的常用网站，比如公司博客，开发社区，常用工具等28类网页。
- 项目页面实现按照项目类别分别推荐不同的博客及其开源项目，比如完整项目，资源聚合类，动画，RV列表动效，二维码等26种开源项目。
- 我的界面实现团队图片轮播，6个网站跳转链接：个人博客网站，个人码云首页，个人CSDN博客首页等
- 在首页/导向/项目页面中的博客均可查看与收藏，并在我的界面实现对收藏的博客进行查看与取消收藏处理。

2.技术方案论证

2.1 开发环境简介

2.1.1移动端

Android博客阅读App使用Android Studio作为开发工具，使用Java作为开发语言。

2.1.2服务器后端

Android博客阅读App后端使用IDEA作为开发工具，使用Java作为开发语言。

后端数据库使用常用的关系型数据库MySQL。

2.2 开发技术简介

2.2.1 网络与解析相关相关技术：

- Retrofit：一个基于okHttp封装的网络请求库，在该Android博客阅读App中用于用户登录，注册和获取收藏的文章信息，获取app中的首页文章首推，导向页面中的所有分类标签信息，项目页面中的所有项目分类信息及其每个分类下对应的文章信息，我的页面中收藏的文章信息。
- Glide：一个源自Google的网络图片加载库，在该Android博客阅读App中用于加载项目图片，首页轮播图的加载。
- agentweb： 一个基于 Android WebView 高度封装的功能强大的web组件库 ，在该Android博客阅读App中用于加载每一篇博客或者开源项目对应的网页
- GSON：源自google的json解析库，在该Android博客阅读App中用于解析后端响应的JSON格式的字符串。

2.2.2 UI界面相关技术：

- tabviewlibrary：一个开源的底部导航栏与fragment碎片适配方案。
- banner：图片轮播库，在该Android博客阅读App中用于加载首页轮播图。
- SmartRefreshLayout：一个开源下拉刷新特效布局，具有多种可自定义的刷新UI效果。
- flowlayout：流式布局，在该Android博客阅读App中用于加载导向页面中的所有标签
- butterknife：控件绑定框架，在该Android博客阅读App中用于绑定绝大部分的UI控件，简化代码。
- design库：安卓原生material design库，使用其中的cardView，recycleView，FloatingActionButton等
- cardView： 安卓原生material design库中的UI组件，适用于实现卡片式布局效果的重要控件
- recycleView：安卓原生material design库中的UI组件，Android 5.0推出的，是support-v7包中的新组件,它被用来代替ListView和GridView，并且能够实现瀑布流的布局，更加高级并且更加灵活，提供更为高效的回收复用机制，同时实现管理与视图的解耦合。在该Android博客阅读App中用于展示首页中每日文章首推的文章，导向页面的标签分类，项目页面中项目等
- listView：安卓原生的UI组件，以列表形式展示其具体数据内容的UI组件，在该Android博客阅读App中用于加载收藏的文章。

2.2.3 移动端数据存储相关技术

SQlite：Android移动设备集成的一种轻量级数据库，在该Android博客阅读App中用于存储用户收藏的博客信息，包括收藏博客的网址，收藏博客的标题。

2.2.4 后端相关技术

- SpringBoot： 简化Spring开发的，当下最流行的基于Java的web开发框架。
- MyBatis：一种支持定制化SQL，存储过程以及高级映射的持久层框架。
- MySQL： 一个开源的关系型数据库管理系统。
- Maven： 项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的项目管理工具。

3.App所使用API接口

3.1 自行搭建的后端提供的API接口

在使用SpringBoot+MyBatis+MySQL所编写的后端，运行至阿里云学生服务器中，提供了如下的接口：

接口名称	接口地址（含参数的为接口举例）	请求方法	参数说明
用户登录接口	http://101.201.238.193:8081/user/login	POST	String类型账号userNumber；String类型密码password
用户注册接口	http://101.201.238.193:8081/user/register	POST	String类型帐号userNumber；String类型密码password
添加指定用户对应的一条收藏文章数据接口	http://101.201.238.193:8081/user/addArticle	POST	String类型账号userNumber；String类型收藏文章网址collectUrl；String类型收藏文章标题collectTitles
获取指定用户对应的所有收藏文章数据接口	http://101.201.238.193:8081/user/findAllArticle	POST	String类型账号userNumber
删除指定用户对应的一条收藏文章数据接口	http://101.201.238.193:8081/user/deleteArticle	POST	String类型账号userNumber；String类型收藏文章网址collectUrl

3.2 所使用的开放API接口

在玩安卓官方网站: https://www.wanandroid.com/) 所使用的部分开放API如下：

接口名称	接口地址（含参数的为接口举例）	请求方法	参数说明
首页轮播图banner接口	https://www.wanandroid.com/banner/json	GET	无
置顶文章接口	https://www.wanandroid.com/article/top/json	GET	无
知识体系数据接口	https://www.wanandroid.com/tree/json	GET	无
获取知识体系下的数据接口	https://www.wanandroid.com/article/list/0/json?cid=60	GET	cid 分类的id；page：页码,拼接在链接上,从0开始(均为int类型)
导航数据标题接口	https://www.wanandroid.com/navi/json	GET	无
获取项目类别标题	https://www.wanandroid.c	GET	无

接口	om/project/tree/json	请求	无
项目列表数据，某个项目类别下的文章接口	接口地址（含参数的为接口举例） https://www.wanandroid.com/project/list/1/json?cid=294	请求方法 GET	参数说明 cid:分类的id；page：页码，拼接在链接上,从0开始(均为int类型)

4.数据库设计

4.1 服务器端MySQL中数据表设计

数据库名 food

表名 user

属性名	数据类型	长度	备注
username	varchar	50	主键，用户名
password	varchar	50	密码

表名 collect_article

属性名	数据类型	长度	备注
collectUrl	varchar	300	收藏博客的网址
collectTitles	varchar	100	收藏博客的标题
username	varchar	50	对应的用户名
id	varchar	10	收藏记录ID，主键且自增

4.2 Android端SQLite中数据表设计

本程序只有在收藏博客时需要使用到SQLite数据库，所以只需要设计一张收藏博客文章的数据表即可。

在我的界面中，收藏的博客以ListView组件呈现，在该组件中收藏的博客以字符串的形式呈现。

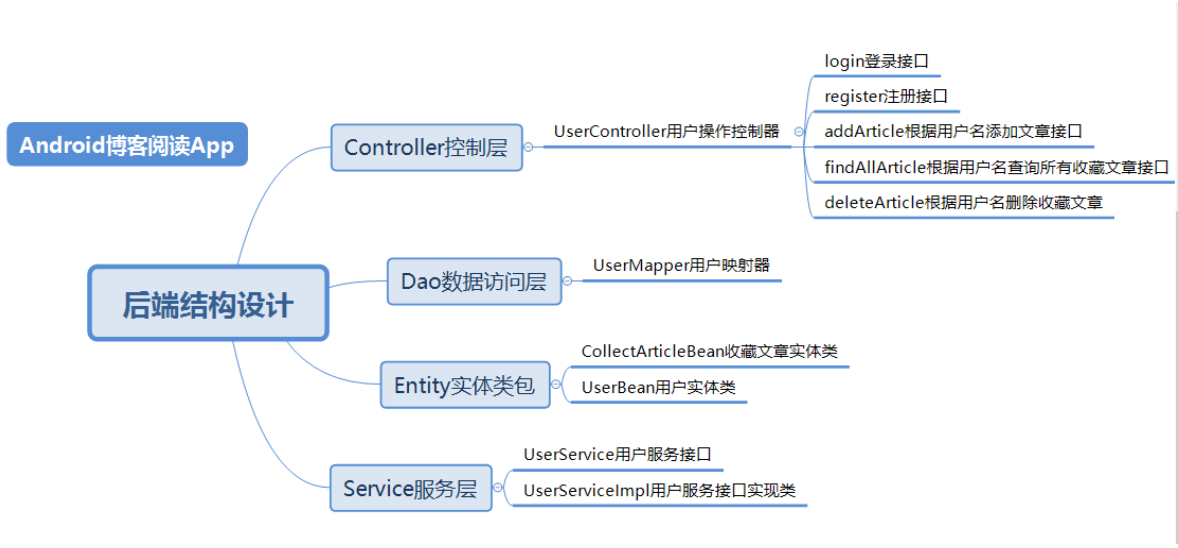
收藏博客文章数据表如下所示：数据库名 article 表名 collect_article

属性名	数据类型	长度	备注
collectUrl	varchar	300	收藏博客的网址
collectTitles	varchar	100	收藏博客的标题

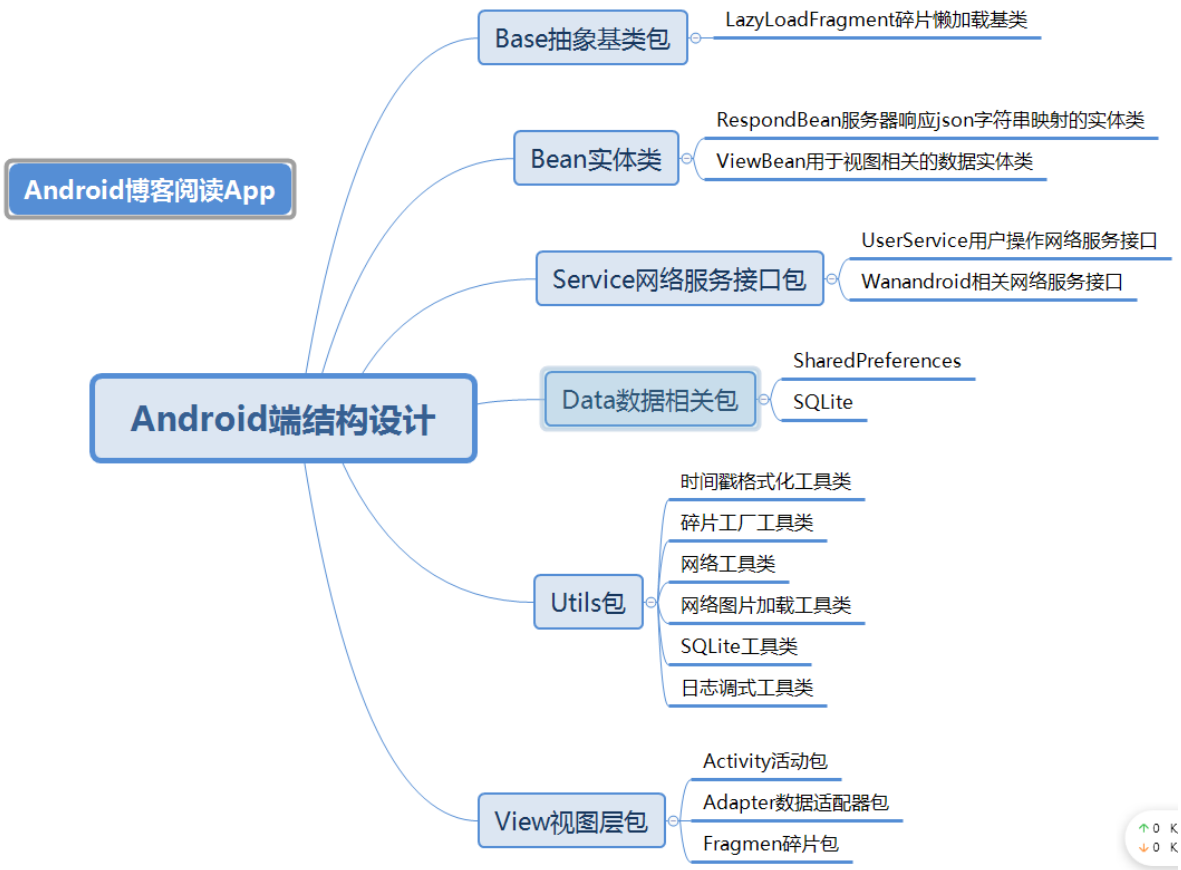
5.系统整体设计

5.1 系统工程设计

5. 1.1后端结构设计



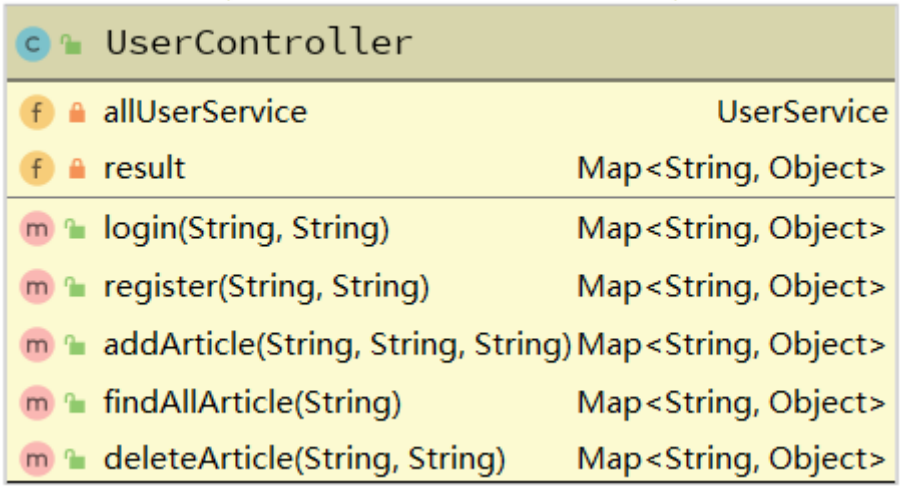
5.1.2 Android端结构设计

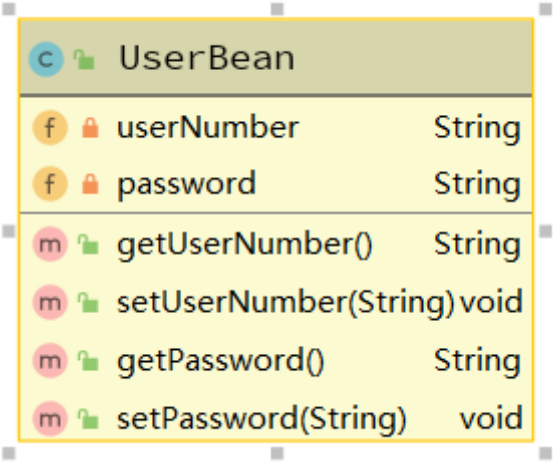


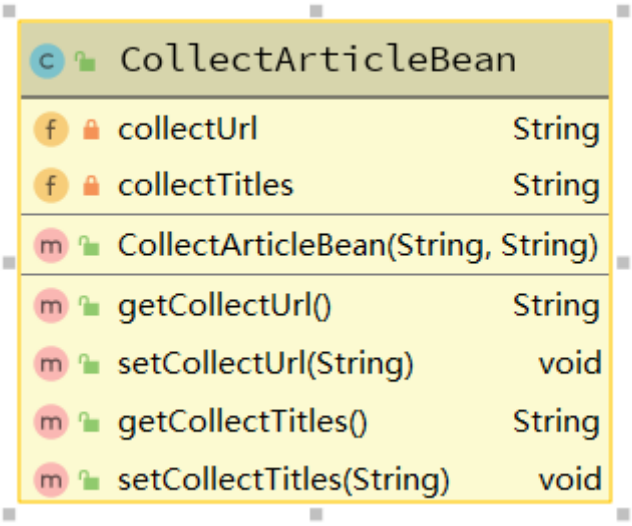
5.2 系统类图设计

5.2.1 后端主要类图

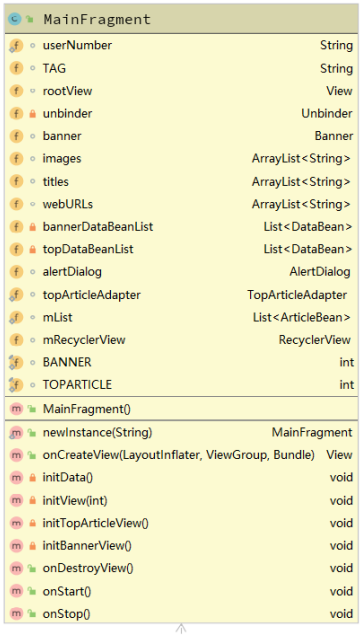
在后端中主要类图为Controller控制层的userControler类，Entity实体类CollectArticleBean和UserBean类。

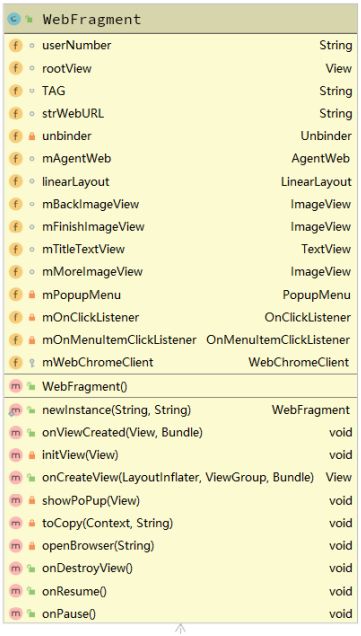


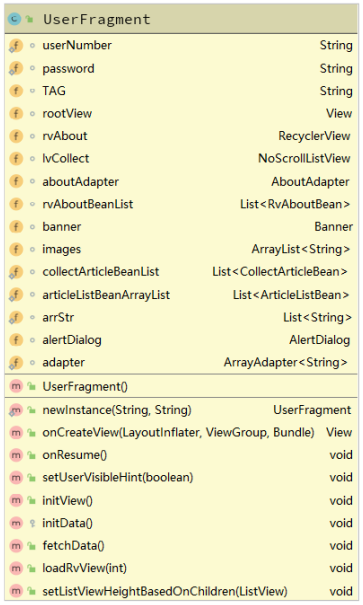


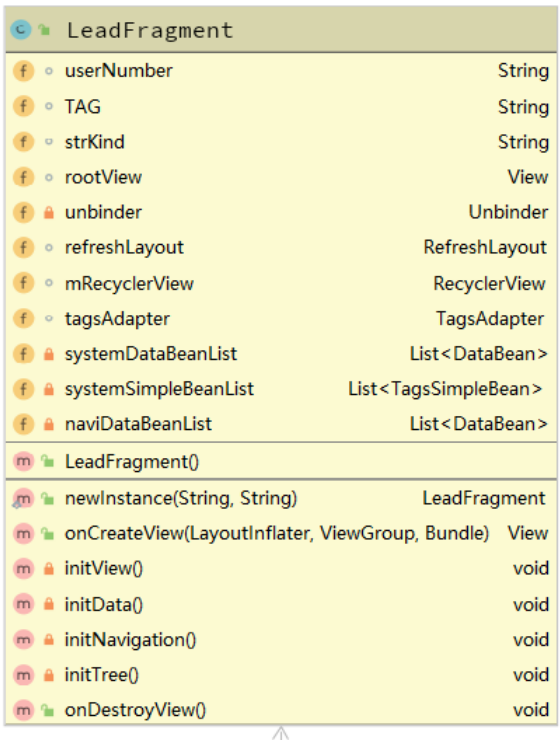


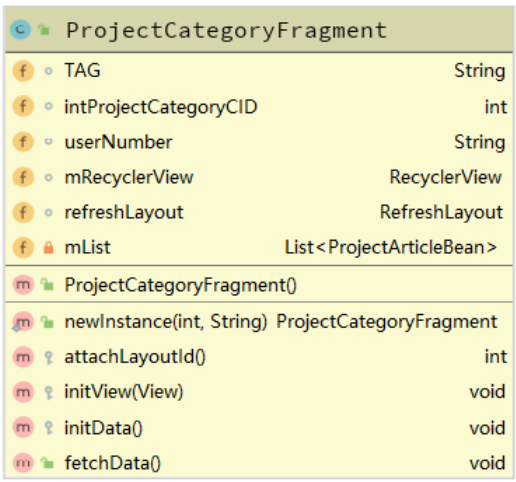
5.2.2 Android端主要类图

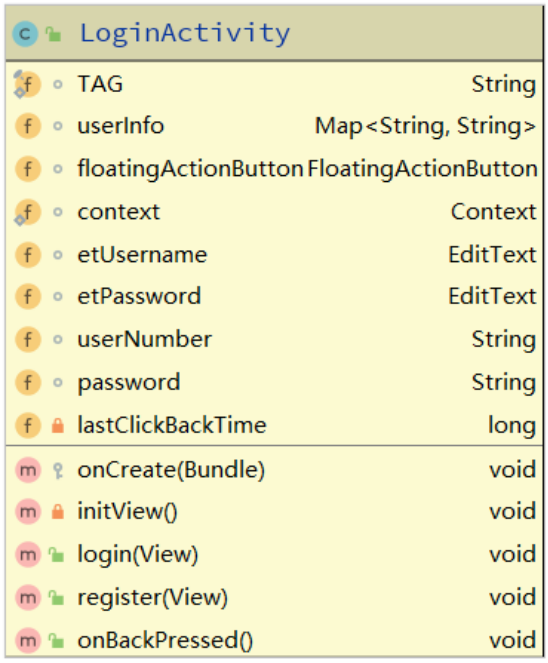

MainFragment
userNumber String
TAG String
rootView View
unbinder Unbinder
banner Banner
images ArrayList<String>
titles ArrayList<String>
webURLs ArrayList<String>
bannerDataBeanList List<DataBean>
topDataBeanList List<DataBean>
alertDialog AlertDialog
topArticleAdapter TopArticleAdapter
mList List<ArticleBean>
mRecyclerView RecyclerView
BANNER int
TOPARTICLE int
MainFragment()
newInstance(String) MainFragment
onCreateView(LayoutInflater, ViewGroup, Bundle) View
initData() void
initView(int) void
initTopArticleView() void
initBannerView() void
onDestroyView() void
onStart() void
onStop() void

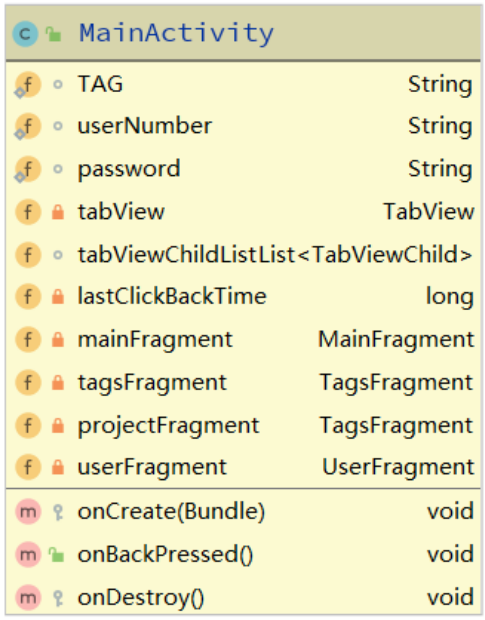

WebFragment
userNumber String
rootView View
TAG String
strWebURL String
unbinder Unbinder
mAgentWeb AgentWeb
linearLayout LinearLayout
mBackImageView ImageView
mFinishImageView ImageView
mTitleTextView TextView
mMoreImageView ImageView
mPopupMenu PopupMenu
mOnClickListener OnClickListener
mOnMenuItemClickListener OnMenuItemClickListener
mWebChromeClient WebChromeClient
WebFragment()
newInstance(String, String) WebFragment
onViewCreated(View, Bundle) void
initView(View) void
onCreateView(LayoutInflater, ViewGroup, Bundle) View
showPoPup(View) void
toCopy(Context, String) void
openBrowser(String) void
onDestroyView() void
onResume() void
onPause() void

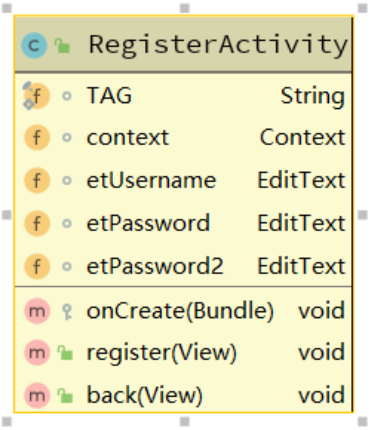

UserFragment
userNumber String
password String
TAG String
rootView View
rvAbout RecyclerView
lvCollect NoScrollListView
aboutAdapter AboutAdapter
rvAboutBeanList List<RvAboutBean>
banner Banner
images ArrayList<String>
collectArticleBeanList List<CollectArticleBean>
articleListBeanArrayList List<ArticleListBean>
arrStr List<String>
alertDialog AlertDialog
adapter ArrayAdapter<String>
UserFragment()
newInstance(String, String) UserFragment
onCreateView(LayoutInflater, ViewGroup, Bundle) View
onResume() void
setUserVisibleHint(boolean) void
initView() void
initData() void
fetchData() void
loadRvView(int) void
setListViewHeightBasedOnChildren(ListView) void

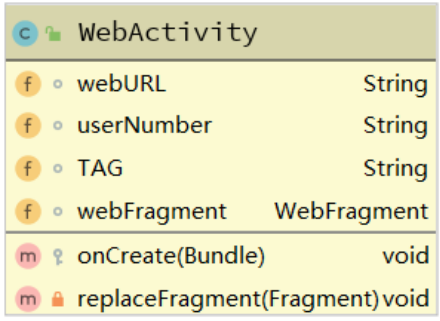

LeadFragment
userNumber String
TAG String
strKind String
rootView View
unbinder Unbinder
refreshLayout RefreshLayout
mRecyclerView RecyclerView
tagsAdapter TagsAdapter
systemDataBeanList List<DataBean>
systemSimpleBeanList List<TagsSimpleBean>
naviDataBeanList List<DataBean>
LeadFragment()
newInstance(String, String) LeadFragment
onCreateView(LayoutInflater, ViewGroup, Bundle) View
initView() void
initData() void
initNavigation() void
initTree() void
onDestroyView() void


ProjectCategoryFragment
TAG String
intProjectCategoryCID int
userNumber String
mRecyclerView RecyclerView
refreshLayout RefreshLayout
mList List<ProjectArticleBean>
ProjectCategoryFragment()
newInstance(int, String) ProjectCategoryFragment
attachLayoutId() int
initView(View) void
initData() void
fetchData() void



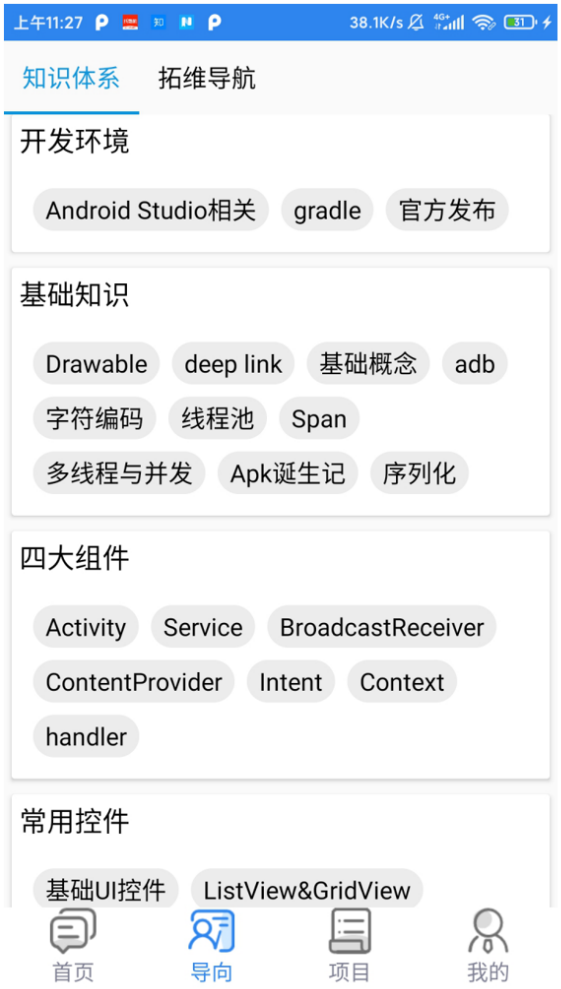


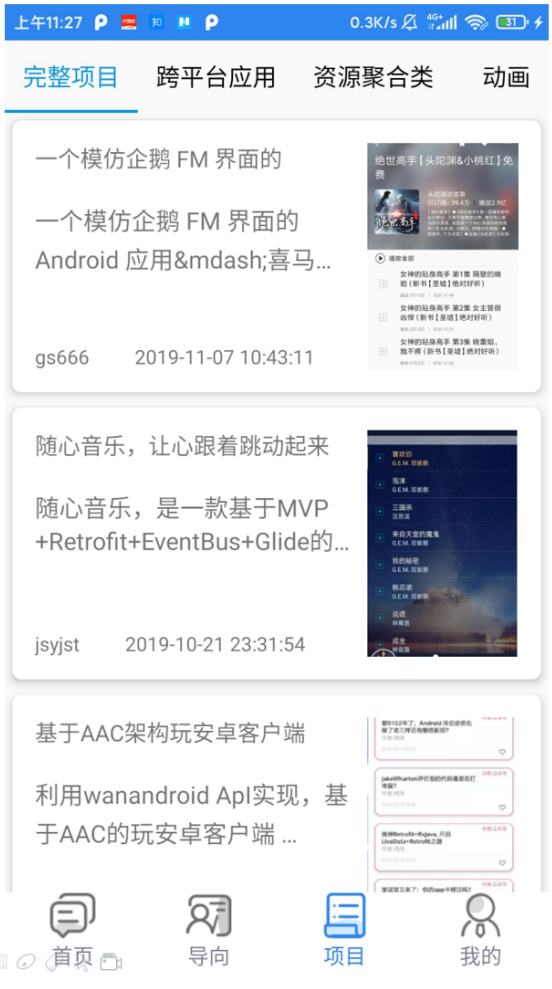


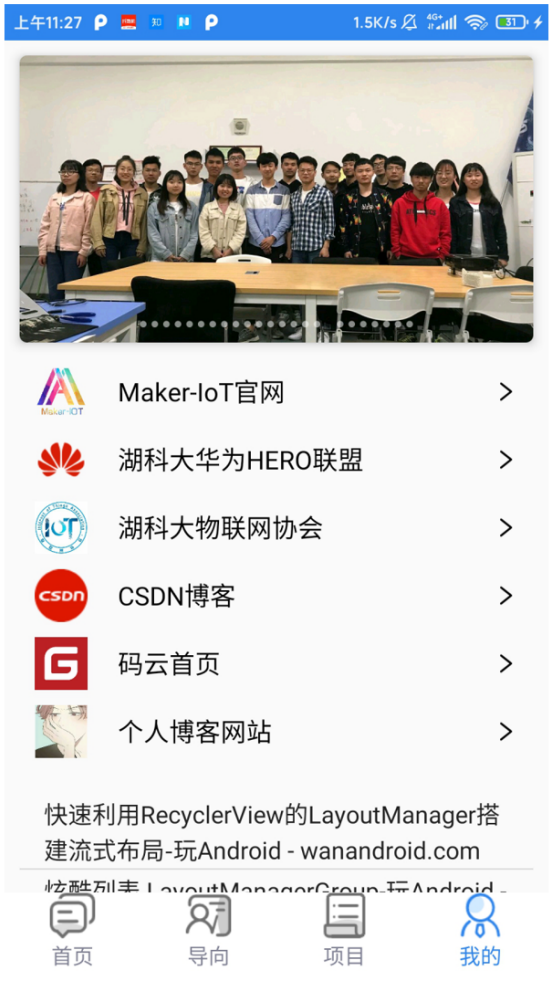


5.3 Android端UI界面设计


上午11:26
7.1K/s
首页
搜索android博客
怎么又出Bug了呀
BUG如风，常伴吾身！
每日文章首推
拉勾
干货资源
阿里 P7 都需要掌握哪些技术?
2020-01-01 00:00:00
xiaoyang
官方
每日一问 | 线程池中的线程是如何复用的?
2019-12-12 00:47:08
首页
导向
项目
我的


上午11:27
38.1K/s
知识体系
拓维导航
开发环境
Android Studio相关
gradle
官方发布
基础知识
Drawable
deep link
基础概念
adb
字符编码
线程池
Span
多线程与并发
Apk诞生记
序列化
四大组件
Activity
Service
BroadcastReceiver
ContentProvider
Intent
Context
handler
常用控件
基础UI控件
ListView&GridView
首页
导向
项目
我的


上午11:27
0.3K/s
完整项目
跨平台应用
资源聚合类
动画
一个模仿企鹅 FM 界面的
一个模仿企鹅 FM 界面的 Android 应用—喜马...
gs666
2019-11-07 10:43:11
随心音乐，让心跟着跳动起来
随心音乐，是一款基于MVP +Retrofit+EventBus+Glide的...
jsyjst
2019-10-21 23:31:54
基于AAC架构玩安卓客户端
利用wanandroid Apl实现，基于AAC的玩安卓客户端 ...
首页
导向
项目
我的


上午11:27
1.5K/s
Maker-IoT官网
湖科大华为HERO联盟
湖科大物联网协会
CSDN博客
码云首页
个人博客网站
快速利用RecyclerView的LayoutManager搭建流式布局-玩Android - wanandroid.com
首页
导向
项目
我的

6.关键技术实现及代码说明

6.1 利用SharedPreferences实现自动保存账号密码

创建SharedPreferences数据处理类，包括保存数据方法saveAccountInfo()和获取数据方法getUserInfo()。

```
public class SaveAccount {
      public static boolean saveAccountInfo(Context context, String userNumber, String password) {
            SharedPreferences sp = context.getSharedPreferences("data", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = sp.edit();
            editor.putString("userNumber", userNumber);
            editor.putString("password", password);
            editor.commit();
            return true;
      }

      public static Map<String, String> getUserInfo(Context context) {
            SharedPreferences sp = context.getSharedPreferences("data", Context.MODE_PRIVATE);
            String userNumber = sp.getString("userNumber", null);//缺省值为null
            String password = sp.getString("password", null);//缺省值为null
            Map<String, String> userMap = new HashMap<>();
            userMap.put("userNumber", userNumber);
            userMap.put("password", password);
            return userMap;
      }
}
```

在登录活动中，登录按钮监听事件函数中的登录成功的条件下，执行如下操作：

```
SaveAccount.saveAccountInfo(context, userNumber, password);//保存账号密码
```

而在登录活动中生命周期方法onCreate()中加载完视图后执行入操作：

```
 Map<String, String> userInfo;
userInfo = SaveAccount.getUserInfo(this);//获取SharedPreferences中的数据
```

这样就实现了当SharedPreferences保存数据，拿到Map<String, String>数据后即可获取其中的数据并将其加载到EditText中，以此实现保存账号密码的功能。

6.2利用Retrofit请求自行搭建后端提供的API接口

导入Retrofit与okhttp依赖包

```
//网络请求 Retrofit库(基于okHttp封装的网络请求库)
      implementation 'com.squareup.okhttp3:okhttp:4.1.0'
      implementation 'com.squareup.retrofit2:retrofit:2.0.2'
      implementation 'com.squareup.retrofit2:converter-gson:2.0.2' // 用Gson解析json的转换器
```

AndroidManifest.xml申明网络权限

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

编写后端登陆注册接口对应的请求接口类，其中MessageBean为服务器响应的json字符串对应的Java实体对象类，这里只包含一个String类型的message属性，因此省略其代码。

```
public interface UserService {
```

```
    //表示提交表单数据，@Field注解键名
    //适用于数据量少的情况
    @POST("login")
    @FormUrlEncoded
    Call<MessageBean> login(@Field("userNumber") String userNumber, @Field("password") String
password);

    @FormUrlEncoded
    @POST("register")
    Call<MessageBean> register(@Field("userNumber") String userNumber, @Field("password") String
password);

    @FormUrlEncoded
    @POST("addArticle")
    Call<MessageBean> addArticle(@Field("userNumber") String userNumber, @Field("collectUrl") String
collectUrl, @Field("collectTitles") String collectTitles);

    @FormUrlEncoded
    @POST("findAllArticle")
    Call<UserCollectArticleBean> findAllArticle(@Field("userNumber") String userNumber);

    @FormUrlEncoded
    @POST("deleteArticle")
    Call<MessageBean> deleteArticle(@Field("userNumber") String userNumber, @Field("collectUrl")
String collectUrl);
}
```

初始化请求对象，即实例化上述接口对应的对象。

```
        retrofit = new Retrofit.Builder()
                .baseUrl("http://101.201.238.193:8081/user/")
                .addConverterFactory(GsonConverterFactory.create()) // 设置数据解析器 JSON-
>JavaBean
                .build();
        foodService = retrofit.create(FoodService.class);
```

发送请求，并处理服务器响应的结果，以登录接口为例：参数为String类型 用户名username,String类型 密码password

```
Call<MessageBean> call = HttpUtil.getUserService().login(username, password);
        call.enqueue(new Callback<MessageBean>() {
            @Override
            public void onResponse(Call<MessageBean> call, Response<MessageBean> response) {
                String strBack=response.body().getMessage();
                if (strBack.equals("loginSuccess")){
                    Toast.makeText(context, "登录成功!", Toast.LENGTH_LONG).show();
                    startActivity(new Intent(context, MainActivity.class));
                    finish();
                }else {
                    Toast.makeText(context, "登录失败!"+strBack, Toast.LENGTH_LONG).show();
                }
            }
            @Override
            public void onFailure(Call<MessageBean> call, Throwable t) {
                Toast.makeText(context, t.getMessage(), Toast.LENGTH_LONG).show();
                Log.d(TAG, "onResponse: error");
                Log.d(TAG,t.getMessage());
```

```
            }
        });
```

6.3利用Retrofit请求玩安卓网站提供的开放API接口

与请求自行搭建后端提供的API接口类似，其服务接口类定义如下：

```
//玩安卓服务接口 baseURL:https://www.wanandroid.com/
public interface WanAndroidService {

    //1.2首页banner
    @GET("banner/json")
    Call<ResponseBody> getBannerData();

    // 1.5 置顶文章
    @GET("article/top/json")
    Call<ResponseBody> getTopArticle();

    // 2.1 知识体系数据
    @GET("tree/json")
    Call<ResponseBody> getTree();

    // 2.2获取知识体系下的数据
    @GET("article/list/{page}/json")
    Call<ResponseBody> getTreeData(@Path("page") int page,@Query("cid") int cid);

    // 3.1 导航数据 标题
    @GET("navi/json")
    Call<ResponseBody> getNavigation();

    // 4.1 获取项目类别
    @GET("project/tree/json")
    Call<ResponseBody> getProjectCategory();

   // 4.2 项目列表数据 某个项目类别下的文章
    @GET("project/list/{page}/json")
    Call<ResponseBody> getProjectArticle(@Path("page") int page, @Query("cid") int cid);
}
```

```
        //初始化WanAndroidService对应的Retrofit实体类
        retrofitWanAndroid = new Retrofit.Builder()
                .baseUrl("https://www.wanandroid.com/")
                .build();
        wanAndroidService = retrofitWanAndroid.create(WanAndroidService.class);
```

```
```

```
Call<ResponseBody> callTopArticle = HttpUtil.getWanAndroidService().getTopArticle();
        callTopArticle.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {
                String strBack = null;
```

```
                        try {
                              strBack = response.body().string();
                              //处理数据
                              JSONObject jsonObject = null;
                              jsonObject = new JSONObject(strBack);
                              Gson gson = new Gson();
                              TopArticleBean topArticleBean;
                              topArticleBean = gson.fromJson(jsonObject.toString(),
TopArticleBean.class);
                              topDataBeanList = topArticleBean.getData();
                              //初始化mList
                              for (int i = 0; i < topDataBeanList.size(); i++) {
                                    //文章所需数据 mList
                                    ArticleBean articleBean = new ArticleBean(
                                                "" + topDataBeanList.get(i).getAuthor(),
                                                "" + topDataBeanList.get(i).getTitle(),
                                                "" +
DateUtil.timeStampDate(topDataBeanList.get(i).getPublishTime() + ""),
                                                "" + topDataBeanList.get(i).getChapterName(),
                                                "" + topDataBeanList.get(i).getLink()
                                          );
                                          mList.add(articleBean);
                                    }
                              //获取完数据后UI操作
                              //主线程将alertDialog提示隐藏
                              Objects.requireNonNull(getActivity()).runOnUiThread(() -> {
                                    initView(TOPARTICLE);
                                    alertDialog.hide();
                              });

                        } catch (IOException | JSONException e) {
                              e.printStackTrace();
                        }
                  }
                   @Override
                   public void onFailure(Call<ResponseBody> call, Throwable t) {

                  }
            });
```

6.4 利用SQlite存储与删除收藏的博客数据

创建SQLiteOpenHelper子类，用于创建数据库及其里面的数据表。

```
public class MyHelper extends SQLiteOpenHelper {
      public MyHelper(@Nullable Context context) {
            super(context, "article", null, 1);
      }

      //数据库第一次创建时调用 // 1.博客url(主键) 2. 博客Z文章标题
      @Override
      public void onCreate(SQLiteDatabase sqLiteDatabase) {
            sqLiteDatabase.execSQL("CREATE TABLE collect_article(collectUrl VARCHAR(300) PRIMARY KEY
,collectTitles VARCHAR(100) )");
      }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    }
}
```

编写SQLite数据库工具类，用于整合对SQLite数据库的数据操作，包括插入一条美食数据，删除一条美食数据，查询所有美食数据。

```
//操作SQLite中的数据1.插入数据2.删除一数据3.查所有博客数据
public class SqliteUtils {
    private static String TAG = "SqliteUtils";
    private static MyHelper myHelper;
    private static SQLiteDatabase sqLiteDatabase;
    private static SqliteUtils sqliteUtils;
    //获取唯一可用的对象
    public static SqliteUtils getInstance(Context context) {
        return sqliteUtils = new SqliteUtils(context);
    }
    //将构造函数私有 用于实现单例 使得外部无法使用构造函数实例化
    // 参数activity的上下文context ： 用于初始化MyHelper
    private SqliteUtils(Context context) {
        myHelper = new MyHelper(context);//初始化MyHelper
        sqLiteDatabase = myHelper.getWritableDatabase();//由myHelper获取可写的SQLiteDatabase对象
    }
    //插入一条美食数据
    public static String insert(String collectUrl, String collectTitles) {
        try {
            sqLiteDatabase.execSQL("INSERT into collect_article (collectUrl,collectTitles)" +
                    "values (?,?)", new Object[]{collectUrl, collectTitles});
            Log.d(TAG, "insert: 收藏博客成功 " + collectTitles);
            return "收藏博客成功！";
        } catch (Exception e) {
            return "您已收藏过该博客了哦！";
        }
    }
    //删除一条美食数据（参数封装为FoodBean对象的类，具体数据从类中获取）
    public static String delete(String collectUrl) {
        try {
            sqLiteDatabase.execSQL("DELETE FROM collect_article WHERE collectUrl=?",
                    new Object[]{collectUrl});
            Log.d(TAG, "insert: 删除成功 " + collectUrl);
            return "删除博客成功！";
        } catch (Exception e) {
            return "删除博客失败！";
        }
    }
    //查所有美食数据 无参数
    //结果使用Cursor保存
    public static List<CollectArticleBean> queryAll() {
        //创建游标对象
        Cursor cursor = sqLiteDatabase.rawQuery("SELECT * FROM collect_article WHERE collectUrl !=
?; ", new String[]{"-1"});
        List<CollectArticleBean> collectArticleBeanList=new ArrayList<>();
        //利用游标遍历所有数据对象
        while (cursor.moveToNext()) {
            String collectUrl = cursor.getString(cursor.getColumnIndex("collectUrl"));
            String collectTitles = cursor.getString(cursor.getColumnIndex("collectTitles"));
```

```
                CollectArticleBean collectArticleBean=new
CollectArticleBean(collectUrl,collectTitles);
                collectArticleBeanList.add(collectArticleBean);
                Log.d(TAG, "queryAll: " +collectArticleBean.toString());
            }
            cursor.close();//关闭游标
            Log.d(TAG, "queryAll: 查询成功");
            return collectArticleBeanList;//返回
        }
        //关闭数据库
        public static void closeSQLite(){
            sqLiteDatabase.close();
        }
}
```

初始化上述工具类

```
SqliteUtils.getInstance(getBaseContext());
```

通过上述代码后，只需在需要操作SQLite数据库的地方调用SqliteUtils工具类中的方法即可，以查询所有收藏的文章为例。

```
List<CollectArticleBean> collectArticleBeanList = new ArrayList<>();//空初始化收藏的文章集合
collectArticleBeanList = SqliteUtils.queryAll();
```

7.技术总结与心得体会

通过这次课程设计进一步加深了对Android移动端开发的很多方面的认识，比如ListView与RecycleView的数据适配，与服务端进行GET/POST网络请求，使用常用的数据存储技术SharedPreferences和SQLite。除此之外也使用Springboot和MyBatis进行了简单的后端服务的搭建。在技术层面上总结来说，各方面的知识涵盖得比较广，加深了对各个知识点的理论认知，提高了实践编码水平。完成了在选题与需求分析阶段所有需完成的功能。当然，本次课程设计仍有不足的地方需要完善与改进，比如app中缺少搜索功能，比如偶尔会有界面内容加载不完全的情况等，这些都需要在后面进一步去查漏补缺，使项目更为完整。