这里所有的东西都确认一遍,并将确认后的结果写到该文档中!

1. 企业详情界面
   1. 企业招聘界面【搜索功能干什么的？】【列表的Api呢】

**搜索功能：在招聘列表范围内搜索职位**

**Api: ListCompanyRecruitRequest**

* 1. 企业联系方式【公司的经纬度地址呢？】【企业的各个联系数据呢】

**没有经纬度字段，根据详细地址**

**联系数据：GetPopCompanyRequest**

**对应返回结果：contact,phoneNo, mailBox, detailedAddr**

* 1. 企业简介【数据Api呢】

**这些数据都是一次获取的GetPopCompanyRequest**

**对应返回结果：basicBenefits,** **companyDes**

* 1. 企业评价【获取特定企业的评价接口呢？】

**ListCompanyCommentsRequest**

1. 首页界面
   1. 首页Banner【Api呢】

**Api：还没整**

* 1. 热门职业界面 【有热门职业的数据吗？能加点吗？】【点击跳到哪？】

**有三条数据，点击跳到TabSearchFragment 在找职位中显示**

* 1. 搜索按钮【点击跳到哪？】

**SearchInterfaceActivity**

1. 伯乐模块
   1. 我的机遇【列表的Api呢】

**Api：还没有**

* 1. 市场行情界面
     1. 同行竞技【数据Api呢】【按钮干什么的？】
     2. 同行薪酬【数据Api呢】【按钮干什么的？】
     3. 行业统计【数据Api呢】【按钮干什么的？】

**点击按钮界面已经加上，这些api 还没有**

1. 搜索模块
   1. 搜索界面【数据Api呢】【刷选条件如何加入】

**Api:还没有**

* 1. 职位详情【数据Api呢】

**Api:还没有**

1. 我的模块
   1. 评论界面【评论的回复如何关联？】

**EvaluationProfessionalInformationActivity**

**中回复企业：Api:NewProfessionalEvaluationReplyRequest**

* 1. 身份实名认证Api呢

**Api：还没有**

1. 消息模块
   1. 各个消息界面的数据Api呢
   2. 相关内部的操作Api呢

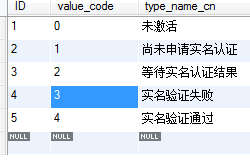
**面试通知Api:没有**

**面试印象Api: GetInterviewEvaluationRequest**

**录用通知Api;没有**

**浏览记录Api: ListByReadingRecordsRequest**

**系统通知Api：没有**



tbl\_dic\_account\_status

getuserprofile判断account\_status

**认证结果为value\_code 为2,4是要给提示**

**企业简介:福利显示问题**

# Android 属性动画

**ValueAnimator.ofFloat(0f,1f) valueAnimator 最核心类 只能改变值**

**ObjectAnimator 可以改变任何对象的属性值，继承自ValueAnimator**

**组合动画 AnimatorSet.play(Animator ani) 返回Builder call after() before() with()**

**添加动画监听器 addListener(new AnimatorListener(){}) 也可以使用**

**new AnimatorAdapterListener(){}**

**XML 编写动画**

**在res 下新建文件夹animator**

**标签<animtor> <objectAnimatro> <set>**

1. **<set** xmlns:android="http://schemas.android.com/apk/res/android"
2. android:ordering="sequentially" **>**
4. **<objectAnimator**
5. android:duration="2000"
6. android:propertyName="translationX"
7. android:valueFrom="-500"
8. android:valueTo="0"
9. android:valueType="floatType" **>**
10. **</objectAnimator>**
12. **<set** android:ordering="together" **>**
13. **<objectAnimator**
14. android:duration="3000"
15. android:propertyName="rotation"
16. android:valueFrom="0"
17. android:valueTo="360"
18. android:valueType="floatType" **>**
19. **</objectAnimator>**
21. **<set** android:ordering="sequentially" **>**
22. **<objectAnimator**
23. android:duration="1500"
24. android:propertyName="alpha"
25. android:valueFrom="1"
26. android:valueTo="0"
27. android:valueType="floatType" **>**
28. **</objectAnimator>**
29. **<objectAnimator**
30. android:duration="1500"
31. android:propertyName="alpha"
32. android:valueFrom="0"
33. android:valueTo="1"
34. android:valueType="floatType" **>**
35. **</objectAnimator>**
36. **</set>**
37. **</set>**
39. **</set>**
40. Animator animator = AnimatorInflater.loadAnimator(context, R.animator.anim\_file);
41. animator.setTarget(view);
42. animator.start();

**布局动画（Layout Animation）**

**LayoutTransition setAnimator(Mode,Animator)**

**Mode :LayoutTransitoin.APPEARING CHANGE\_APPEARING DISAPPEARING CHANGE\_DISAPPEARING CHANGE (动画是对自身还是其他谁设置)**

**直接对view.animate()**

# view 事件分发机制

**dispatchTouchEvent -> setOnTouchEventListener onTouch -> onTouchEvent**

# ViewGroup的事件分发机制

**如果有子view将事件传递给子view ,自己的onTouchEvent不执行**

**可以重写onInterceptTouchEvent（）返回true进行事件拦截 不让子view执行**

**执行自己的onTouchEvent()**

**service 和子线程没关系 运行在主线程上 在后台运行不依赖UI 如果耗时做操作ANR异常 所以在service里在开启线程**

**greenDao数据库ORM 框架 基本知识**

**用一个java 工程 generate DaoMaster DaoSession entityDao entity**

**Git 与github 的交互使用 安装msysGit(直接下载Git)和tortoiseGit图形化工具**

**stage暂存区 —》local repostory —》remote repostory**

**setting add commit push pull**

# universal-image-loader--------

**配置ImageLoaderConfiguration**

1. ImageLoaderConfiguration config = **new** ImageLoaderConfiguration
2. .Builder(context)
3. .memoryCacheExtraOptions(480, 800) // max width, max height，即保存的每个缓存文件的最大长宽
4. .discCacheExtraOptions(480, 800, CompressFormat.JPEG, 75, **null**) // Can slow ImageLoader, use it carefully (Better don't use it)/设置缓存的详细信息，最好不要设置这个
5. .threadPoolSize(3)//线程池内加载的数量
6. .threadPriority(Thread.NORM\_PRIORITY - 2)
7. .denyCacheImageMultipleSizesInMemory()
8. .memoryCache(**new** UsingFreqLimitedMemoryCache(2 \* 1024 \* 1024)) // You can pass your own memory cache implementation/你可以通过自己的内存缓存实现
9. .memoryCacheSize(2 \* 1024 \* 1024)
10. .discCacheSize(50 \* 1024 \* 1024)
11. .discCacheFileNameGenerator(**new** Md5FileNameGenerator())//将保存的时候的URI名称用MD5 加密
12. .tasksProcessingOrder(QueueProcessingType.LIFO)
13. .discCacheFileCount(100) //缓存的文件数量
14. .discCache(**new** UnlimitedDiscCache(cacheDir))//自定义缓存路径
15. .defaultDisplayImageOptions(DisplayImageOptions.createSimple())
16. .imageDownloader(**new** BaseImageDownloader(context, 5 \* 1000, 30 \* 1000)) // connectTimeout (5 s), readTimeout (30 s)超时时间
17. .writeDebugLogs() // Remove for release app
18. .build();//开始构建
19. // Initialize ImageLoader with configuration.
20. ImageLoader.getInstance().init(config);//全局初始化此配置

**ImageLoader imageLoader = ImageLoader.getInstance();**

**imageLoader. displayImage(imageUrl, imageView, options);**

**添加加载监听器，第一次加载时动画监听**

**DisplayImageOptions options = new DisplayImageOptions.Builder()**

**.cacheOnDisk(true)**

**.cacheInMemory(true)**

**.showImageForEmptyUri(R.drawable.icon\_avatar\_default)**

**.showImageOnLoading(R.drawable.icon\_avatar\_default)**

**.showImageOnFail(R.drawable.icon\_avatar\_default)**

**.imageScaleType(ImageScaleType.EXACTLY\_STRETCHED)**

**.displayer(new RoundedBitmapDisplayer(DimenUtils.dip2px(Global.getAn45fairApplication(), 50)))**

**.bitmapConfig(Bitmap.Config.RGB\_565).build();**

**清除缓存：imageLoader.clearMemeoryCache() clearDiscCache()**

# MessagePack

**以二进制格式传递数据**

**需要依赖包：json-simple javassist msgpack junit**

***pack.register(object.getClass)***

***byte[] data = pack.write(object)***

**public <E> E read4DataDir(Class<E> eClass, String key) {**

**if (eClass == null || TextUtils.isEmpty(key)) return null;**

**FileInputStream in = null;**

**try {**

***pack.register(eClass);***

**File dir = FileHelper.getInternalDataDir(mApplication);**

**File saveFile = new File(dir, md5(key));**

**in = new FileInputStream(saveFile);**

***return pack.read(in, eClass);***

**} catch (Throwable e) {**

**return null;**

**} finally {**

**if (in != null) try {**

**in.close();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

# MD5

**private String md5(String key) {**

**try {**

**MessageDigest digest = MessageDigest.getInstance("MD5");**

**digest.update(key.getBytes());**

**StringBuilder builder = new StringBuilder();**

**for (byte b : digest.digest()) {**

**builder.append(Integer.toHexString((b >> 4) & 0xf));**

**builder.append(Integer.toHexString(b & 0xf));**

**}**

**return builder.toString().toLowerCase();**

**} catch (NoSuchAlgorithmException e) {**

**return key;**

**}**

**}**

# SizeFile

**public static long sizeFile(File file) {**

**if (file == null || !file.exists()) return 0;**

**if (file.isDirectory()) {**

**File[] fileList = file.listFiles();**

**long size = 0;**

**for (File item : fileList) {**

**size += sizeFile(item);**

**}**

**return size;**

**} else {**

**return file.length();**

**}**

**}**

# DelDir

**public static boolean delDir(File file) throws Throwable {**

**if (file != null && file.exists()) {**

**if (file.isFile()) {**

**return delFile(file);**

**} else {**

**String[] paths = file.list();**

**if (paths == null) return true;**

**File temp = null;**

**boolean flag = true;**

**for (int i = 0; i < paths.length; i++) {**

**temp = new File(file, paths[i]);**

**if (temp.isFile()) {**

**if (!delFile(temp)) flag = false;**

**} else if (temp.isDirectory()) {**

**if (!delDir(temp)) flag = false;**

**}**

**}**

**return flag;**

**}**

**}**

**return true;**

**}**

# BinaryUtils

**public class BinaryUtils {**

**private static long[] sizeUnit = new long[]{**

**1024l,1048576l,1073741824l,1099511627776l,1125899906842624l**

**};**

**private static String[] showUnit = new String[]{**

**"B","KB","MB","GB","TB"**

**};**

**private static DecimalFormat df = new DecimalFormat("0.0");**

**public static String scaleUnit(long size){**

**for (int i = 0; i < sizeUnit.length; i++) {**

**if(size<sizeUnit[i]){**

**double now = size/((double)(i<1?1:sizeUnit[i-1]));**

**String util = showUnit[i];**

**return df.format(now)+util;**

**}**

**}**

**return "";**

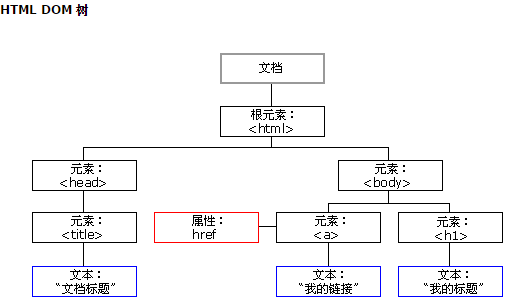
**}**

**}**

# JavaScript 学习

HTML DOM

网页加载后创建文档对象模型 ，被构造成对象的树



查找HTML 元素 然后操作，getElementById () getElementsByTagName()(html标签)

**改变HTML 元素样式 doucment.getElementById(id).style.property=new style**

HTML DOM 可以对HTML 元素分配事件

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

function displayDate(){

document.getElementById("demo").innerHTML=Date();

}

</script>

# webview javascript 与 android 交互

1. **<body>**
2. **<a>**js中调用本地方法**</a>**
3. **<script>**
5. function funFromjs(){
6. document.getElementById("helloweb").innerHTML="HelloWebView,i'm from js";
7. }
8. var aTag = document.getElementsByTagName('a')[0];
9. aTag.addEventListener('click', function(){
10. //调用android本地方法
11. myObj.fun1FromAndroid("调用android本地方法fun1FromAndroid(String name)！！");
12. return false;
13. }, false);
14. **</script>**
15. **<p></p>**
16. **<div** id="helloweb"**>**
18. **</div>**
19. **</body>**
20. //设置编码
21. mWebView.getSettings().setDefaultTextEncodingName("utf-8");
22. //支持js
23. mWebView.getSettings().setJavaScriptEnabled(**true**);
24. //设置背景颜色 透明
25. mWebView.setBackgroundColor(Color.argb(0, 0, 0, 0));
26. //设置本地调用对象及其接口
27. mWebView.addJavascriptInterface(**new** JavaScriptObject(mContext), "myObj");
28. //载入js
29. mWebView.loadUrl("file:///android\_asset/test.html");
31. //点击调用js中方法
32. mBtn1.setOnClickListener(**new** View.OnClickListener() {
34. @Override
35. **public** **void** onClick(View v) {
36. mWebView.loadUrl("javascript:funFromjs()");
37. Toast.makeText(mContext, "调用javascript:funFromjs()", Toast.LENGTH\_LONG).show();
38. }
39. });
40. **public** **class** JavaScriptObject {
41. Context mContxt;
42. @JavascriptInterface //sdk17版本以上加上注解
43. **public** JavaScriptObject(Context mContxt) {
44. **this**.mContxt = mContxt;
45. }
47. **public** **void** fun1FromAndroid(String name) {
48. Toast.makeText(mContxt, name, Toast.LENGTH\_LONG).show();
49. }
51. **public** **void** fun2(String name) {
52. Toast.makeText(mContxt, "调用fun2:" + name, Toast.LENGTH\_SHORT).show();
53. }
54. }

# SQLite

public class DatabaseHelper extends SQLiteOpenHelper{

private static final String DB\_NAME = "myData.db";

private static final int version = 1;

public DataBaseHelper(Context context){

super(context,DB\_NAME,null,version);

}

 @Override

    public void onCreate(SQLiteDatabase db) {

        String sql = "create table user(username varchar(20) not null , password varchar(60) not null );";

        db.execSQL(sql);

    }

@Override

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        // TODO Auto-generated method stub

     }

}

**public class MainActivity extends Activity{**

1. @Override
2. **public** **void** onCreate(Bundle savedInstanceState) {
3. **super**.onCreate(savedInstanceState);
4. DatabaseHelper helper = new DatabaseHelper(this);
5. SQLiteDatabase db = null;
6. db = helper.getWritableDatabase();
7. }

**}**

**数据的添加**

1.使用insert方法

|  |  |
| --- | --- |
| 1  2  3 | ContentValues cv = new ContentValues();//实例化一个ContentValues用来装载待插入的数据cv.put("username","Jack Johnson");//添加用户名  cv.put("password","iLovePopMusic"); //添加密码  db.insert("user",null,cv);//执行插入操作 |

2.使用execSQL方式来实现

|  |  |
| --- | --- |
| 1  2 | String sql = "insert into user(username,password) values ('Jack Johnson','iLovePopMuisc');//插入操作的SQL语句  db.execSQL(sql);//执行SQL语句 |

**数据的删除**

同样有2种方式可以实现

|  |  |
| --- | --- |
| 1  2  3 | String whereClause = "username=?";//删除的条件  String[] whereArgs = {"Jack Johnson"};//删除的条件参数  db.delete("user",whereClause,whereArgs);//执行删除 |

使用execSQL方式的实现

|  |  |
| --- | --- |
| 1  2 | String sql = "delete from user where username='Jack Johnson'";//删除操作的SQL语句  db.execSQL(sql);//执行删除操作 |

**数据修改**

同上，仍是2种方式

|  |  |
| --- | --- |
| 1  2  3  4  5 | ContentValues cv = new ContentValues();//实例化ContentValues  cv.put("password","iHatePopMusic");//添加要更改的字段及内容  String whereClause = "username=?";//修改条件  String[] whereArgs = {"Jack Johnson"};//修改条件的参数  db.update("user",cv,whereClause,whereArgs);//执行修改 |

使用execSQL方式的实现

|  |  |
| --- | --- |
| 1  2 | String sql = "update [user] set password = 'iHatePopMusic' where username='Jack Johnson'";//修改的SQL语句  db.execSQL(sql);//执行修改 |

**数据查询**

数据查询相对前面几种方法就复杂一些了，因为查询会带有很多条件

通过query实现查询的

public Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Cursor c = db.query("user",null,null,null,null,null,null);//查询并获得游标  if(c.moveToFirst()){//判断游标是否为空      for(int i=0;i<c.getCount();i++){          c.move(i);//移动到指定记录          String username = c.getString(c.getColumnIndex("username");          String password = c.getString(c.getColumnIndex("password"));      }  } |

通过rawQuery实现的带参数查询

|  |  |
| --- | --- |
| 1  2  3  4 | Cursor c = db.rawQuery("select \* from user where username=?",new Stirng[]{"Jack Johnson"});  if(cursor.moveToFirst()) {      String password = c.getString(c.getColumnIndex("password"));  } |

# SQLite ORM 框架 GreenDao

GreenDAO整个运行的逻辑是通过配置其提供的JavaSE代码，自动在一个文件夹下生成需要Bean、DAO、DaoMaster、DaoSession,依赖两个jar包greendao-generator.jar和freemarker.jar如果数据库要升级，这里是修改代码后全量重新创建

GitHub 上下载示例工程，在引入JavaSE工程MyDaoGenerator来实现关键代码的自动创建

打开MyDaoGenerator工程中的MyDaoGenerator.java

配置里面的代码，如示例:

public static void main(String args[]) throws Exception {

Schema schema = new Schema( 1, "com.xxx.bean");

// 1: 数据库版本号

// com.xxx.bean:自动生成的Bean对象会放到/java-gen/com/xxx/bean中

schema.setDefaultJavaPackageDao("com.xxx.dao");

// DaoMaster.java、DaoSession.java、BeanDao.java会放到/java-gen/com/xxx/dao中

// 上面这两个文件夹路径都可以自定义，也可以不设置

initUserBean(schema); // 初始化Bean了

new DaoGenerator().generateAll(schema, args[0]);// 自动创建

}

private static void initUserBean(Schema schema) {

Entity userBean = schema.addEntity("UserBean");// 表名

userBean.setTableName("user"); // 可以对表重命名

userBean.addStringProperty("id").primaryKey().index();// 主键，索引

userBean.addStringProperty("phone");

userBean.addStringProperty("profile\_picture");

userBean.addStringProperty("client\_id");

userBean.addStringProperty("name");

userBean.addStringProperty("location");

userBean.addStringProperty("gender");

}

假设要给user表增加一个”age”属性，并且增加school表的做法是：

把数据库版本号修改为2

在initUserBean方法中加入userBean.addStringProperty("age");

新建initSchoolBean方法

当明白这个文件的写法之后就要运行这个文件了，进到AndroidStudio的Gradel面板中执行run任务

初始化代码示例:

THDevOpenHelper helper = new THDevOpenHelper(context, "my-db", null);

SQLiteDatabase db = helper.getWritableDatabase();

DaoMaster daoMaster = new DaoMaster(db);

daoSession = daoMaster.newSession();

**避免升级数据库时候删除原来的表格数据，对OpenHelper封装**

public class THDevOpenHelper extends DaoMaster.OpenHelper {

public THDevOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory) {

super(context, name, factory);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

switch (oldVersion) {

case 1:

//创建新表，注意createTable()是静态方法

// SchoolDao.createTable(db, true);

// 加入新字段

// db.execSQL("ALTER TABLE 'moments' ADD 'audio\_path' TEXT;");

// TODO

break;

}

}

}

# weight

原始尺寸加上剩余尺寸占比

# EventBus

2、基本使用

（1）自定义一个类，可以是空类，比如：

1. **public** **class** AnyEventType {
2. **public** AnyEventType(){}
3. }

（2）在要接收消息的页面注册：

1. eventBus.register(**this**);

（3）发送消息

1. eventBus.post(**new** AnyEventType event);

（4）接受消息的页面实现(共有四个函数，各功能不同，这是其中之一，可以选择性的实现，这里先实现一个)：

1. **public** **void** onEvent(AnyEventType event) {}
2. onEventMainThread
3. onEventBackgroundThread
4. onEventAsync

（5）解除注册

1. eventBus.unregister(**this**);

**onEvent:**如果使用onEvent作为订阅函数，那么该事件在哪个线程发布出来的，onEvent就会在这个线程中运行，也就是说发布事件和接收事件线程在同一个线程。使用这个方法时，在onEvent方法中不能执行耗时操作，如果执行耗时操作容易导致事件分发延迟。  
**onEventMainThread:**如果使用onEventMainThread作为订阅函数，那么不论事件是在哪个线程中发布出来的，onEventMainThread都会在UI线程中执行，接收事件就会在UI线程中运行，这个在Android中是非常有用的，因为在Android中只能在UI线程中跟新UI，在onEvnetMainThread方法中是不能执行耗时操作的。  
**onEventBackground:**如果使用onEventBackgrond作为订阅函数，那么如果事件是在UI线程中发布出来的，那么onEventBackground就会在子线程中运行，如果事件本来就是子线程中发布出来的，那么onEventBackground函数直接在该子线程中执行。  
**onEventAsync：**使用这个函数作为订阅函数，那么无论事件在哪个线程发布，都会创建新的子线程在执行onEventAsync.

**消息的接收是根据参数中的类名来决定执行哪一个的**