

今日历史APP系统结构与制作过程

APP简介

APP功能结构

模块结构与制作过程

老黄历

今日历史

详细事件

个人主页

笔记

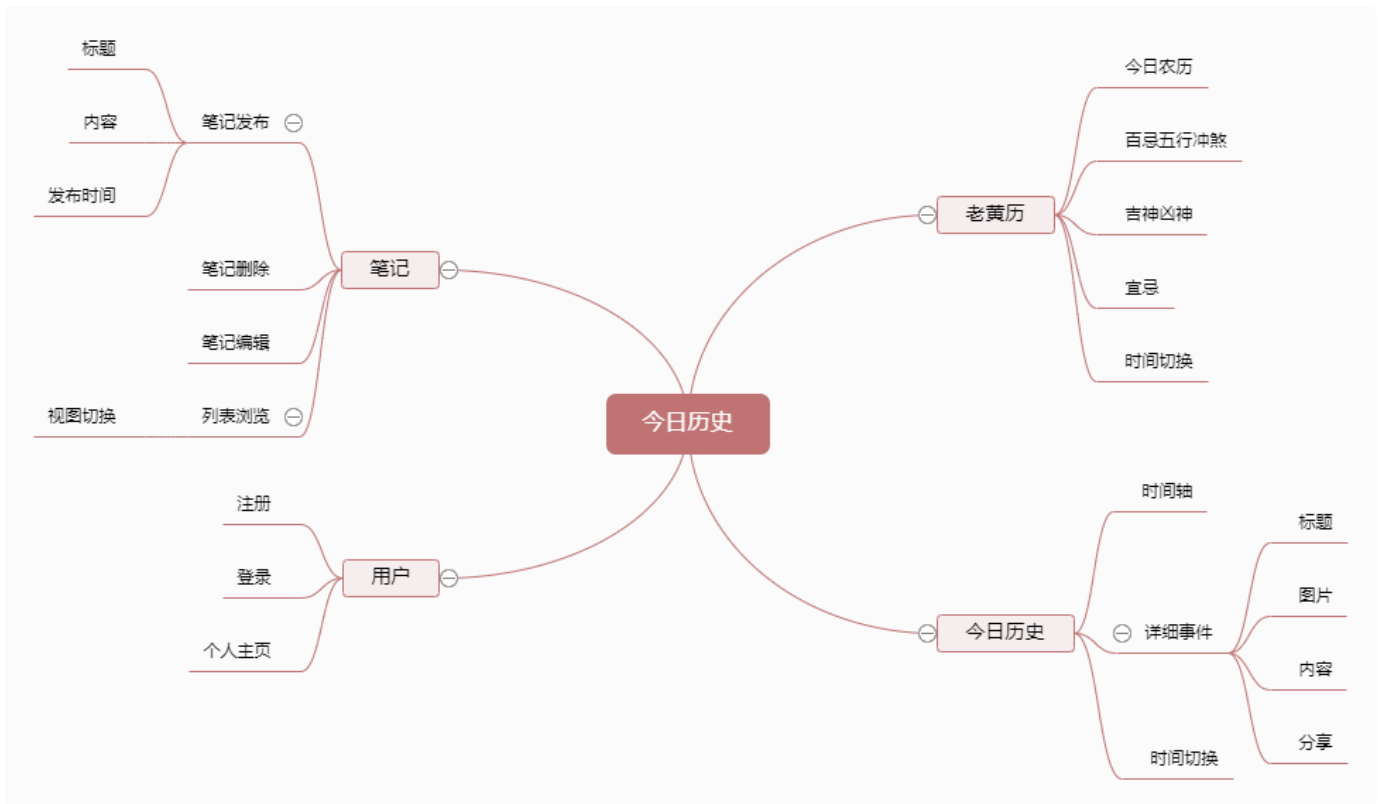
张菲娅 2019141480011

APP简介

今日历史APP是面向中国传统文化与世界历史爱好者的多功能应用，用户可便捷地获取当日的黄历信息、历史上的今日发生的详细事件；并可以在APP中随时记录知识与感悟，方便日后查看复习。APP针对不同的用户群体也做出了细节设计：

1. 针对老年群体，首页黄历放大加粗，可作为电子日历使用。
2. 针对历史爱好者，APP提供日期选择功能，用户可查看不同日期的历史；APP还提供了分享功能，用户可以将自己感兴趣的历史事件分享于他人。
3. 针对笔记爱好者，APP提供简洁纯净的笔记功能，用户可快速记录、编辑或删除笔记。

APP功能结构



模块结构与制作过程

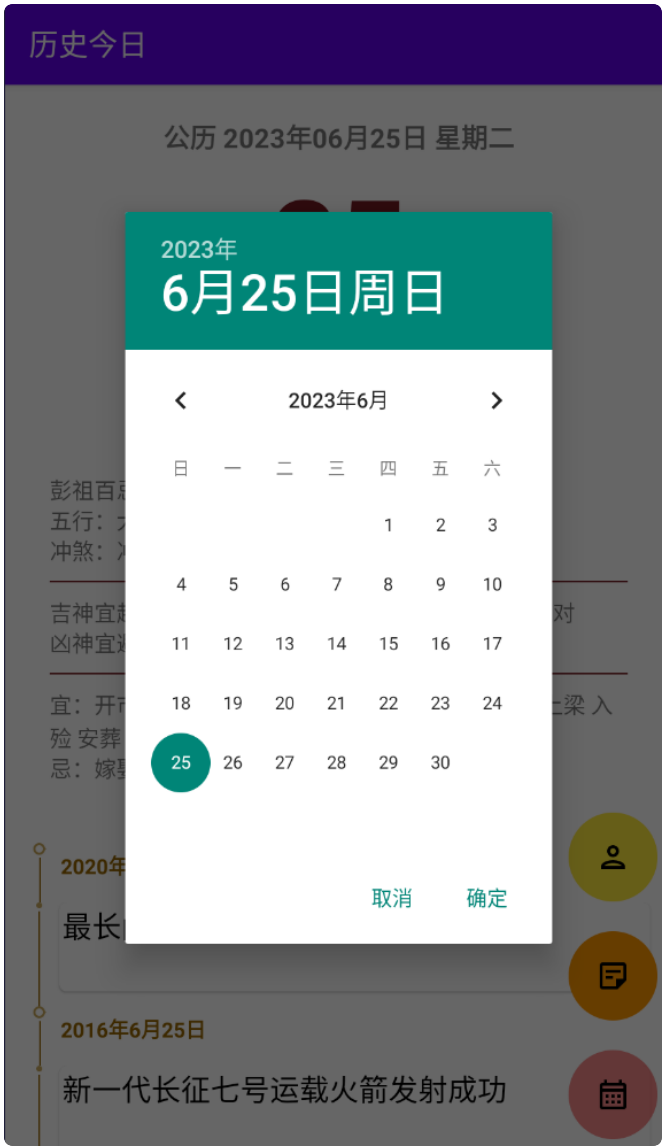
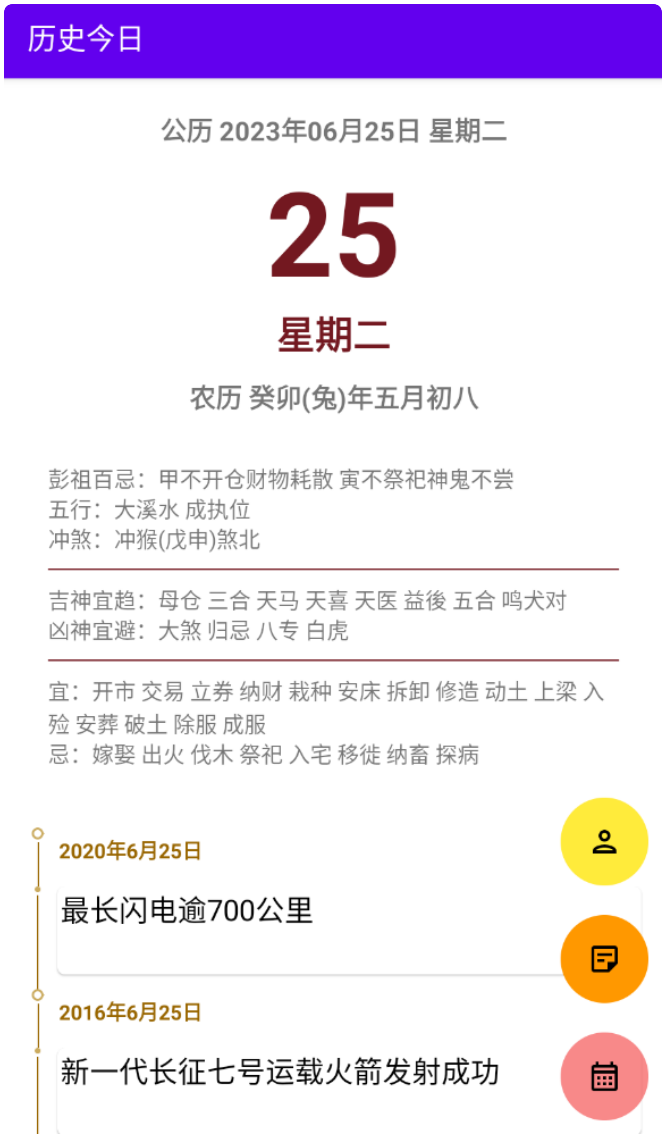
老黄历

1. 核心原理

依赖OkHttp在网络API进行数据请求与接收，从而获取查询日期的黄历信息数据包。利用提前建立好的实体化对象将数据整理传入对应的页面控件，对控件进行渲染更新。

2. 页面实现： `main_headerview.xml`

线性布局排列各个 `TextView` 。



3. 功能实现

- 时间获取： `Calendar` 默认设置当前时间，可通过系统组件调整时间

```
1  //获取日历对象
2      mCalendar = Calendar.getInstance();
3      mDate = new Date();
4      mCalendar.setTime(mDate);
5      int month = mCalendar.get(Calendar.MONTH) + 1;
6      int day = mCalendar.get(Calendar.DAY_OF_MONTH);
7  //获取星期
8  private String getWeek(int year, int month, int day) {
9      Calendar calendar = Calendar.getInstance();
10     calendar.set(year, month, day);
11     String[] weeks = {"星期日", "星期一", "星期二", "星期三", "星期四",
12     "星期五", "星期六"};
13     int index = calendar.get(Calendar.DAY_OF_WEEK) - 1;
14     return weeks[index];
15 }
16 //通过点击日历控件弹出对话框，调整日历日期重载
17 private void popCalendarDialog() {
18     Calendar calendar = Calendar.getInstance();
19     DatePickerDialog datePickerDialog = new DatePickerDialog(this,
20     new DatePickerDialog.OnDateSetListener() {
21         @Override
22         public void onDateSet(DatePicker view, int year, int month
23         , int dayOfMonth) {
24             // 改变老黄历的显示内容
25             String time = year + "-" + (month + 1) + "-" + dayOfMo
26             nth;
27             String url = ContentURL.getLaoHuangLiURL(time);
28             loadHeaderData(url);
29             // 今日历史的数据
30             String date = (month + 1) + "/" + dayOfMonth;
31             String url2 = ContentURL.getTodayHistoryURL(date);
32             loadData(url2);
33         }
34     }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),
35     calendar.get(Calendar.DAY_OF_MONTH));
36     datePickerDialog.show();
37 }
```

- 数据请求与接收：依赖OkHttp建立客户端与服务器的网络连接、数据请求与响应。

```
▼ BaseActivity.java Java | 复制代码

1 public class BaseActivity extends AppCompatActivity implements Callbac
  k {
2     // 网络加载用的Okhttp
3     private OkHttpClient mOkHttpClient;
4     @Override
5     protected void onCreate(@Nullable Bundle savedInstanceState) {...}
9     public void loadData(String url) {...}
14    @Override
15    public void onFailure(Call call, IOException e) {...}
18    @Override
19    public void onResponse(Call call, Response response) throws IOExce
    ption {...}
22    private void loadDataFailed() {...}
30 }
```

异步加载数据，通过 `handler` 切到主线程，渲染数据。

```
▼ MainActivity.java Java | 复制代码

1 private void loadHeaderData(String url) {
2     OkHttpClient client = new OkHttpClient.Builder().build();
3     Request request = new Request.Builder().url(url).build();
4     Call call = client.newCall(request);
5     call.enqueue(new Callback() {
6         @Override
7         public void onFailure(Call call, IOException e) {
8             }
9         @Override
10        public void onResponse(Call call, Response response) throw
    s IOException {
11            mDatas.clear();
12            LaoHuangLiBean laoHuangLiBean = new Gson().fromJson(re
    sponse.body().string(), LaoHuangLiBean.class);
13            resultBean = laoHuangLiBean.getResult();
14            Message message = new Message();
15            message.what = MSG_WHAT2;
16            mHandler.sendMessage(message); //handler将控制老黄历渲染数
    据
17        }
    }
```

- 调用API：接口来自聚合数据 (<http://v.juhe.cn/laohuangli/d>) 。

▼ ContentURL.java

Java

📄 复制代码

```
1 public static String getLaoHuangLiURL(String data) {
2     String url = "http://v.juhe.cn/laohuangli/d?date=" + data +
3         "&key=6dea892b8e9e5dcf10e3608164111eab";
4     return url;
5 }//参数key是我自己申请的访问钥匙, data是传入日期
```

- 数据包封装

▼ LaoHuangLiBean.java

Java

📄 复制代码

```
1 public static class ResultBean {
2     private String id;
3     private String yangli;
4     private String yinli;
5     private String wuxing;
6     private String chongsha;
7     private String baiji;
8     private String jishen;
9     private String yi;
10    private String xiongshen;
11    private String ji;
12    //...
13 }
```

- 控件更新

```
1 private void setLaoHuangLi() {
2     yinliTv.setText("农历 " + resultBean.getYinli());
3     String[] yangliArr = resultBean.getYangli().split("-");
4     String week = getWeek(Integer.parseInt(yangliArr[0]), Integer.
    parseInt(yangliArr[1]),
5         Integer.parseInt(yangliArr[2]));
6     yangliTv.setText("公历 " + yangliArr[0] + "年" + yangliArr[1] +
    "月" + yangliArr[2] + "日 " + week);
7     dayTv.setText(yangliArr[2]);
8     weekTv.setText(week);
9     baijiTv.setText("彭祖百忌: " + resultBean.getBaiji());
10    wuxingTv.setText("五行: " + resultBean.getWuxing());
11    chongshaTv.setText("冲煞: " + resultBean.getChongsha());
12    jishenTv.setText("吉神宜趋: " + resultBean.getJishen());
13    xiongshenTv.setText("凶神宜避: " + resultBean.getXiongshen());
14    yiTv.setText("宜: " + resultBean.getYi());
15    jiTv.setText("忌: " + resultBean.getJi());
16 }
```

今日历史

1. 核心原理

依赖OkHttp在网络API进行数据请求与接收，从而获取查询日期的今日历史数据包。利用提前建立好的实体化对象将数据整理，再通过适配器 `HistoryAdapter` 传入时间轴 `ListView`，对控件进行渲染更新。

2. 页面实现

- 时间轴列表通过 `ListView` 实现：`activity_main.xml`。
- 列表项目通过线性布局实现：`item_main_timeline.xml`。



3. 功能实现

- 数据请求同老黄历。首页接收数据后，倒时序传入5条进入适配器，异步加载更新。


```
MainActivity.java Java 复制代码
1 public void onResponse(Call call, Response response) throws IOException {
2     mData.clear();
3     historyBean = new Gson().fromJson(response.body().string(), HistoryBean.class);
4     List<HistoryBean.ResultBean> list = historyBean.getResult();
5     for (int i = list.size() - 1; i > list.size() - 6; i--) {
6         mData.add(list.get(i));
7     }
8     Message message = new Message();
9     message.what = MSG_WHAT1;
10    mHandler.sendMessage(message); //handler将控制适配器更新数据
11 }
12 class MyHandler extends Handler {
13     @Override
14     public void handleMessage(@NonNull Message msg) {
15         switch (msg.what) {
16             case MSG_WHAT1:
17                 mAdapter.notifyDataSetChanged(); //mAdapter属于HistoryAdapter类
18                 break;
19             case MSG_WHAT2:
20                 setLaoHuangLi();
21         }
22     }
23 }
24 }
```

- 调用API：接口来自聚合数据 (<http://v.juhe.cn/laohuangli/d>) 。

```
ContentURL.java Java 复制代码
1 public static String getTodayHistoryURL(String data) {
2     String url = "http://v.juhe.cn/todayOnhistory/queryEvent.php?date=" + data +
3         "&key=dc0be1f476d191405e3377fc26595cb2";
4     return url;
5 }
```

- 适配器：实现视图的生成与传入，使用了 `convertView` 进行优化，避免 `View` 的重复创建。

```
1  //提前准备好视图
2  class ViewHolder {
3      TextView timeView, titleView;
4      LinearLayout timeLayout;
5
6      public ViewHolder(View itemView) {
7          timeView = itemView.findViewById(R.id.item_main_time);
8          titleView = itemView.findViewById(R.id.item_main_title);
9          timeLayout = itemView.findViewById(R.id.item_main_ll);
10     }
11 }
12 public class HistoryAdapter extends BaseAdapter {
13     Context mContext;
14     private List<HistoryBean.ResultBean> mDatas;
15     public HistoryAdapter(Context context, List<HistoryBean.ResultBean>
16 > datas) {...}
19     @Override
20     public int getCount() {...}
23     @Override
24     public Object getItem(int position) {...}
27     @Override
28     public long getItemId(int position) {...}
31     @Override
32     public View getView(int position, View convertView, ViewGroup parent) {
33         ViewHolder viewHolder;
34         if (convertView == null) {
35             convertView = LayoutInflater.from(mContext).inflate(R.layout.item_main_timeline, parent, false);
36             viewHolder = new ViewHolder(convertView);
37             convertView.setTag(viewHolder);
38         } else {
39             viewHolder = (ViewHolder) convertView.getTag();
40         }
41         //如果当前历史的发生日期与上一个相同, 则不显示该历史事件
42         //以保证每个日期 (年份) 只显示一个历史事件
43         HistoryBean.ResultBean resultBean = mDatas.get(position);
44         if (position != 0) {
45             HistoryBean.ResultBean lastResultBean = mDatas.get(position - 1);
46             if (resultBean.getDate() == lastResultBean.getDate()) {
47                 viewHolder.timeLayout.setVisibility(View.GONE);
48             } else {
49                 viewHolder.timeLayout.setVisibility(View.VISIBLE);
50             }
51         }
52     }
53 }
```

```

51         } else {
52             viewHolder.timeLayout.setVisibility(View.VISIBLE);
53         }
54         viewHolder.timeView.setText(resultBean.getDate());
55         viewHolder.titleView.setText(resultBean.getTitle());
56
57         return convertView;
58     }

```

- 数据包封装

▼ HistoryBean.java Java [复制代码](#)

```

1 public static class ResultBean implements Serializable {
2     private String day;
3     private String date;
4     private String title;
5     private String e_id;
6     //...
7 }

```

- 查看更多：由于首页只显示5条数据，可以通过点击 **footer** 切换到更多历史的页面进一步浏览时间轴，该时间轴显示API获取到的全部数据。

▼ MainActivity.java Java [复制代码](#)

```

1 //footer提前设置了标签，如果显示数据不为空，intent将通过bundle携带其进入“更多历史”事件
2 String tag = (String) v.getTag();
3 if (tag.equals("footer")) {
4     Intent intent = new Intent(this, MoreHistoryActivity.class);
5     if (historyBean != null) {
6         Bundle bundle = new Bundle();
7         bundle.putSerializable("history", historyBean);
8         intent.putExtras(bundle);
9     }
10    startActivity(intent);
11 }
12 }

```

```
1 try {  
2     Bundle bundle = getIntent().getExtras();  
3     historyBean = (HistoryBean) bundle.getSerializable("history"  
4         y");  
5     mDatas.addAll(historyBean.getResult());  
6     mAdapter.notifyDataSetChanged();  
7 } catch (Exception e) {  
8     emptyTv.setVisibility(View.VISIBLE);  
9 }
```

详细事件

1. 核心原理

在首页或者更多历史页面点击时间轴的 `item` 后，获取该 `item` 的 `e_id`，通过 `intent` 将变量传 `e_id` 入“具体历史”的事件。再调用API请求该 `e_id` 为参数值的数据包，接受、渲染。

2. 页面实现： `activity_history_detail.xml`

未知高度滑动布局。

第二次大沽之战发生



在164年前的今天，1859年6月25日(农历1859年5月...日)，第二次大沽之战发生。

第二次鸦片战争期间有3次大沽战斗。

第一次发生在1858年5月20日，英法联军集大小舰艇26艘、侵略军2700名，陈兵白河之外，向直隶总督谭...出照会，限2小时内交出炮台，接着发炮轰击。大沽...门户，南北两岸有4座炮台，守军奋起反击，伤敌舰3艘，毙伤敌军八、九十名，清官兵400余人殉国，由于谭廷襄逃跑，炮台失陷。侵略军进逼天津，清朝被迫签订中英、中法《天津条约》。联军退去。



想要了解最长闪电逾700公里详情吗？快来下载今日历史这个app吧



3. 功能实现

- 传入 `e_id`

▼ MainActivity.java Java 复制代码

```
1 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
2     Intent intent = new Intent(MainActivity.this, HistoryDetailActivity.class);
3     // 这里因为添加了headerView导致position与item对不上了
4     // 通过parent的getAdapter().getItem(position)就能取得正确的元素了
5     // 当listView有headerView的时候, getAdapter()会返回一个HeaderViewListAdapter
6     // 这个Adapter其实包装了一开始传进去的adapter, 能够返回正确的listView item
7     String e_id = ((HistoryBean.ResultBean) parent.getAdapter().getItem(position)).getE_id();
8     intent.putExtra("id", e_id);
9     startActivity(intent);
10 }
```

- 调用API

▼ ContentURL.java Java 复制代码

```
1 public static String getHistoryDetail(String uid) {
2     String url = "http://v.juhe.cn/todayOnhistory/queryDetail.php?"
3     + "key=dc0be1f476d191405e3377fc26595cb2&e_id=" + uid;
4     return url;
5 }
```

- 数据包封装

▼ HistoryDetailBean.java Java 复制代码

```
1 public static class ResultBean {
2     private String e_id;
3     private String title;
4     private String content;
5     private String picNo;
6     private List<PicUrlBean> picUrl;
7     //...
8 }
```

- 请求接收渲染

```
1  //请求
2      String e_id = getIntent().getStringExtra("id");
3      String url = ContentURL.getHistoryDetail(e_id);
4      loadData(url);
5
6  //异步加载
7  public void onResponse(Call call, Response response) throws IOException {
8      HistoryDetailBean historyDetailBean = new Gson().fromJson(response.body().string(), HistoryDetailBean.class);
9      resultBean = historyDetailBean.getResult().get(0);
10     Message message = new Message();
11     message.what = MSG_WHAT;
12     mHandler.sendMessage(message);
13 }
14 class MyHandler extends Handler {
15     @Override
16     public void handleMessage(@NonNull Message msg) {
17         if (msg.what == MSG_WHAT) {
18             titleTv.setText(resultBean.getTitle());
19             contentView.setText(resultBean.getContent());
20             //如果图片空不显示, 图片有依赖Picasso加载
21             if (resultBean.getPicUrl().size() != 0) {
22                 String picUrl = resultBean.getPicUrl().get(0).getUrl();
23                 if (TextUtils.isEmpty(picUrl)) {
24                     picture.setVisibility(View.GONE);
25                 } else {
26                     picture.setVisibility(View.VISIBLE);
27                     Picasso.get().load(picUrl).into(picture);
28                 }
29             } else {
30                 picture.setVisibility(View.GONE);
31             }
32         }
33     }
34 }
```

- 点击分享按钮, 隐式 Intent 短信分享

```
1 public void onClick(View v) {
2     switch (v.getId()) {
3         case R.id.history_detail_back:
4             finish();
5             break;
6         case R.id.history_detail_share:
7             String text = "我发现一款好用的软件--今日历史，快来一起探索这
            个app吧";
8             if (resultBean != null) {
9                 text = "想要了解" + resultBean.getTitle() + "详情吗？
            快来下载今日历史这个app吧";
10            }
11            Intent intent = new Intent(Intent.ACTION_SEND);
12            intent.setType("text/plain");
13            intent.putExtra(Intent.EXTRA_TEXT, text);
14            startActivity(intent);
15            break;
16        }
17    }
```

个人主页

1. 核心原理

通过SQLite创建用户数据库，登录实现数据库的查询功能，注册实现数据库的增加，登录成功后将用户名与密码通过 `PreferenceManager` 保存，个人主页从 `PreferenceManager` 调用相关信息，渲染 `TextView` 控件。

2. 页面实现： `activity_login.xml`， `activity_register.xml`， `activity_person.xml`

线性布局。



3. 功能实现

- 用户数据库管理

通过 `UserDbOpenHelper` 调用数据库。

▼ UserDbOpenHelper.java Java 复制代码

```
1 //数据库创建
2 private static final String DB_NAME = "Mysqlite.db";
3 private static final String create_users = "create table users(name varchar(32),password varchar(32))";
4 public void onCreate(SQLiteDatabase db) { db.execSQL(create_users); }
5 //注册
6 public long register(UserBean u){
7     SQLiteDatabase db = getWritableDatabase();
8     ContentValues cv = new ContentValues();
9     cv.put("name",u.getUsername());
10    cv.put("password",u.getPassword());
11    long users = db.insert("users",null,cv);
12    return users;
13 }
14 //登录, 条件查询数据库中符合username字段的记录, 再依次查询记录对应的密码字段是否符合
15 public boolean login(String name,String password){
16     SQLiteDatabase db1 = getWritableDatabase();
17     boolean result = false;
18     Cursor users = db1.query("users",null,"name like ?",new String
19 []{name},null,null,null);
20 if(users!=null){
21     while(users.moveToNext()){
22         String password1 = users.getString(1);
23         result = password1.equals(password);
24     }
25 }
26 return result;
27 }
```

- 用户信息对象封装

▼ UserBean.java Java 复制代码


```
1 public class UserBean {
2     String username;
3     String password;
4     //...
5 }
```

- 具体事件调用数据库

登录成功后, 数据传入 `PreferenceManager` , 随后在个人主页事件中渲染相关控件

LoginActivity.java


Java

 复制代码

```
1 public void submitlog(View view) {
2     String s1 = username.getText().toString();
3     String s2 = password.getText().toString();
4     boolean result = useropenhelper.login(s1,s2);
5     if(result){
6         ToastUtil.toastShort(this, "登录成功");
7         SharedPreferences.Editor editor = PreferenceManager.getDefaultSharedPreferences(this).edit();
8         editor.putString("USERNAME", s1);
9         editor.apply();
10        editor.putString("PASSWORD", s2);
11        editor.apply();
12        Intent intent = new Intent(this,MainActivity.class);
13        startActivity(intent);
14    }else{
15        ToastUtil.toastShort(this, "登录失败");
16    }
17 }
```

PersonActivity.java

Java

 复制代码

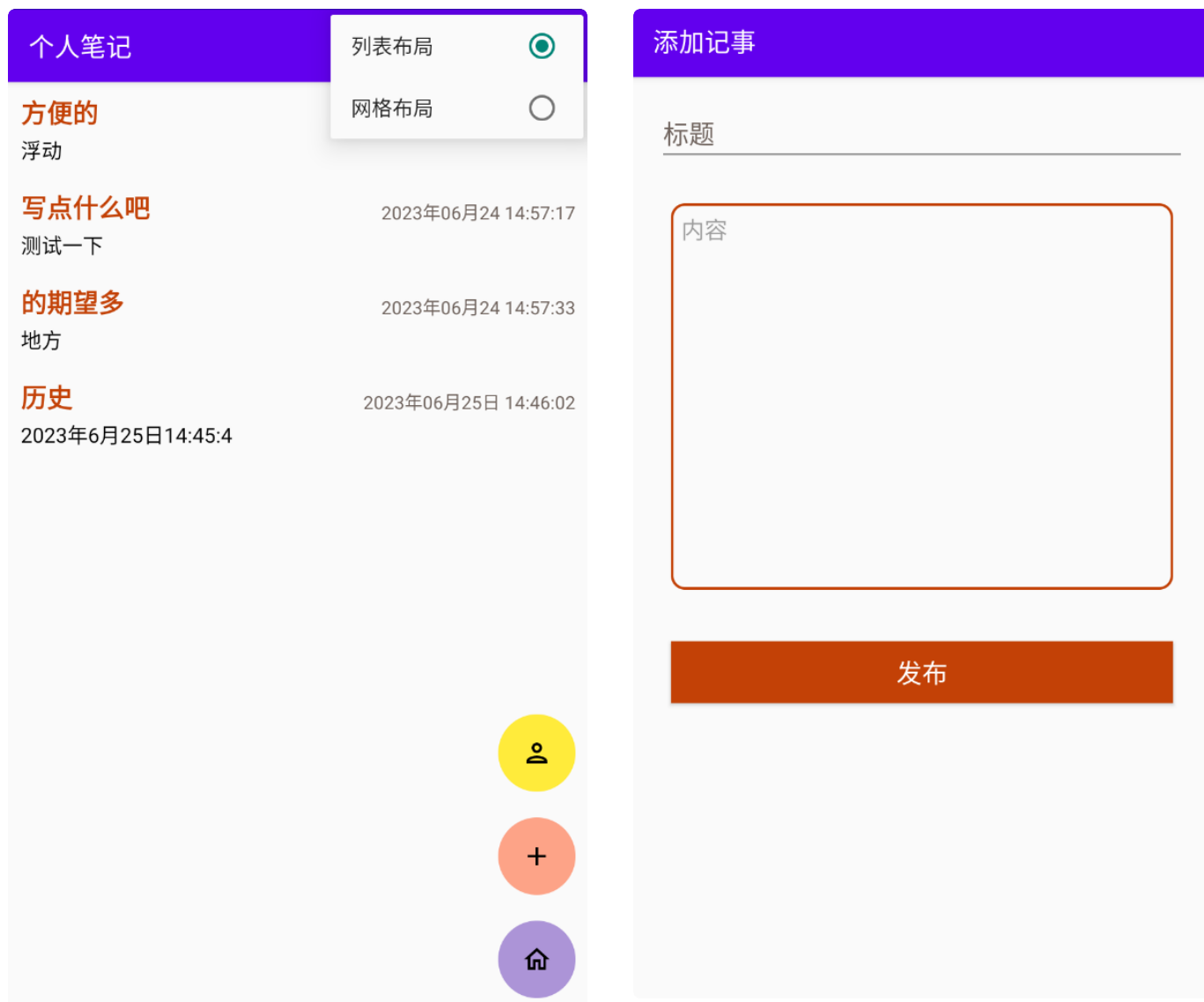
```
1 public class PersonActivity extends AppCompatActivity {
2     TextView username,password;
3     @Override
4     protected void onCreate(Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         setContentView(R.layout.activity_person);
7         username = findViewById(R.id.username);
8         password = findViewById(R.id.password);
9         username.setText(PreferenceManager.getDefaultSharedPreferences(this).getString("USERNAME", ""));
10        password.setText(PreferenceManager.getDefaultSharedPreferences(this).getString("PASSWORD", ""));
11    }
```

笔记

1. 核心原理

通过SQLite创建笔记本数据库，并可通过点击相关控件进行数据库的增删改查操作，每次操作后自动刷新数据库的查询结果。因此笔记页面可实时显示所有记录，并可切换视图的显示模式。

2. 页面实现: `activity_note.xml`, `activity_add.xml`, `activity_edit.xml`, `list_item_dialog.xml`, `list_item_layout.xml`, `list_item_grid_layout.xml`, `menu_main.xml`
- `activity_note.xml` 使用 `RecyclerView`, 可进行布局切换, 视图滚动



- `activity_add.xml`, `activity_edit.xml` 类似, 未知高度 `ScrollView` 布局
- `list_item_dialog.xml` 设置编辑/删除弹窗的布局, `list_item_layout.xml` 上设置笔记排列的线性垂直布局, `list_item_grid_layout.xml` 设置笔记排列的网格布局。



- `menu_main.xml` 中 `androidx.appcompat.widget.SearchView` 设置搜索栏拓展, `gro`
`up` 设置下拉视图切换的菜单

3. 功能实现

- 笔记数据库管理

```
1  //数据库创建
2  private static final String DB_NAME = "noteSQLite.db";
3  private static final String TABLE_NAME_NOTE = "note";
4  private static final String CREATE_TABLE_SQL = "create table " + TABLE
    _NAME_NOTE + " (id integer primary key autoincrement, title text, cont
    ent text, create_time text)";
5  public void onCreate(SQLiteDatabase db) {db.execSQL(CREATE_TABLE_SQL);
    }
6  //数据插入
7  public long insertData(Note note) {
8      SQLiteDatabase db = getWritableDatabase();
9      ContentValues values = new ContentValues();
10     values.put("title", note.getTitle());
11     values.put("content", note.getContent());
12     values.put("create_time", note.getCreateTime());
13     return db.insert(TABLE_NAME_NOTE, null, values);
14 }
15 //具体数据删除
16 public int deleteFromDbById(String id) {
17     SQLiteDatabase db = getWritableDatabase();
18     return db.delete(TABLE_NAME_NOTE, "id like ?", new String[]{id
    });
19 }
20 //具体数据修改
21 public int updateData(Note note) {
22     SQLiteDatabase db = getWritableDatabase();
23     ContentValues values = new ContentValues();
24     values.put("title", note.getTitle());
25     values.put("content", note.getContent());
26     values.put("create_time", note.getCreateTime());
27     return db.update(TABLE_NAME_NOTE, values, "id like ?", new Str
    ing[]{note.getId()});
28 }
29 //获取全部数据, 生成Note对象传入列表
30 public List<Note> queryAllFromDb() {
31     SQLiteDatabase db = getWritableDatabase();
32     List<Note> noteList = new ArrayList<>();
33     Cursor cursor = db.query(TABLE_NAME_NOTE, null, null, null, nu
    ll, null, null);
34     if (cursor != null) {
35         while (cursor.moveToNext()) {
36             String id = cursor.getString(cursor.getColumnIndexOrTh
    row("id"));
37             String title = cursor.getString(cursor.getColumnIndexOrTh
    row("title"));
```

```

38         String content = cursor.getString(cursor.getColumnIndexOrThrow("content"));
39         String createTime = cursor.getString(cursor.getColumnIndexOrThrow("create_time"));
40         Note note = new Note();
41         note.setId(id);
42         note.setTitle(title);
43         note.setContent(content);
44         note.setCreateTime(createTime);
45         noteList.add(note);
46     }
47     cursor.close();
48 }
49 return noteList;
50 }
51 //根据标题查询数据, 生成Note对象传入列表
52 public List<Note> queryFromDbByTitle(String title) {
53     if (TextUtils.isEmpty(title)) {
54         return queryAllFromDb(); }
55     SQLiteDatabase db = getWritableDatabase();
56     List<Note> noteList = new ArrayList<>();
57     Cursor cursor = db.query(TABLE_NAME_NOTE, null, "title like ?",
58     , new String[]{"%" + title + "%"}, null, null, null);
59     if (cursor != null) {
60         while (cursor.moveToNext()) {
61             String id = cursor.getString(cursor.getColumnIndexOrThrow("id"));
62             String title2 = cursor.getString(cursor.getColumnIndexOrThrow("title"));
63             String content = cursor.getString(cursor.getColumnIndexOrThrow("content"));
64             String createTime = cursor.getString(cursor.getColumnIndexOrThrow("create_time"));
65             Note note = new Note();
66             note.setId(id);
67             note.setTitle(title2);
68             note.setContent(content);
69             note.setCreateTime(createTime);
70             noteList.add(note);
71         }
72         cursor.close();
73     }
74     return noteList;
75 }

```

- 笔记信息对象封装

▼ Note.java

Java | 复制代码

```
1 public class Note implements Serializable{
2     private String title;
3     private String content;
4     private String createTime;
5     private String id;
6     //...
7 }
```

- 点击具体笔记跳转编辑页, `intent` 传递具体的 `Note` 对象

▼ NoteActivity.java

Java | 复制代码

```
1 private void bindMyViewHolder(MyViewHolder holder, int position) {
2     Note note = mBeanList.get(position);
3     holder.mTvTitle.setText(note.getTitle());
4     holder.mTvContent.setText(note.getContent());
5     holder.mTvTime.setText(note.getCreateTime());
6     holder.rlContainer.setOnClickListener(new View.OnClickListener
7     () {
8         @Override
9         public void onClick(View v) {
10             Intent intent = new Intent(mContext, EditActivity.class);
11             intent.putExtra("note", note);
12             mContext.startActivity(intent);
13         }
14     });
15 }
```

- 长按跳转编辑/删除


```
1  holder.rlContainer.setOnLongClickListener(new View.OnLongClickListener() {
2      @Override
3      public boolean onLongClick(View v) {
4          // 弹出弹窗
5          Dialog dialog = new Dialog(mContext, android.R.style.ThemeOverlay_Material_Dialog_Alert);
6          View dialogView = mLayoutInflater.inflate(R.layout.list_item_dialog_layout, null);
7          TextView tvDelete = dialogView.findViewById(R.id.tv_delete);
8          TextView tvEdit = dialogView.findViewById(R.id.tv_edit);
9          //删除点击监听
10         tvDelete.setOnClickListener(new View.OnClickListener() {
11             @Override
12             public void onClick(View v) {
13                 int row = mNoteDbOpenHelper.deleteFromDbById(note.getId());
14                 if (row > 0) {
15                     removeData(position);
16                 }
17                 dialog.dismiss();
18             }
19         });
20         //编辑点击监听
21         tvEdit.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View v) {
24                 Intent intent = new Intent(mContext, EditActivity.class);
25                 intent.putExtra("note", note);
26                 mContext.startActivity(intent);
27                 dialog.dismiss();
28             }
29         });
30         dialog.setContentView(dialogView);
31         dialog.setCanceledOnTouchOutside(true);
32         dialog.show();
33         return true;
34     }
35 });
36 }
```

- 切换视图

```
1  //通过viewType控制布局模式
2  public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
3      if(viewType == TYPE_LINEAR_LAYOUT){
4          View view = mLayoutInflater.inflate(R.layout.list_item_layout, parent, false);
5          MyViewHolder myViewHolder = new MyViewHolder(view);
6          return myViewHolder;
7      }else if(viewType == TYPE_GRID_LAYOUT){
8          View view = mLayoutInflater.inflate(R.layout.list_item_grid_layout, parent, false);
9          MyGridViewHolder myGridViewHolder = new MyGridViewHolder(view);
10         return myGridViewHolder;
11     }
12
13     return null;
14 }
15 //通过currentListLayoutMode保存当前选中布局，在后续操作中依然保持该布局
16 public boolean onOptionsItemSelected(@NonNull MenuItem item) {
17     item.setChecked(true);
18     switch (item.getItemId()) {
19         case R.id.menu_linear:
20             setToLinearList();
21             currentListLayoutMode = MODE_LINEAR;
22             SpfUtil.saveInt(this, KEY_LAYOUT_MODE, MODE_LINEAR);
23             return true;
24         case R.id.menu_grid:
25             setToGridList();
26             currentListLayoutMode = MODE_GRID;
27             SpfUtil.saveInt(this, KEY_LAYOUT_MODE, MODE_GRID);
28             return true;
29         default:
30             return super.onOptionsItemSelected(item);
31     }
32 }
33 public boolean onPrepareOptionsMenu(Menu menu) {
34     if (currentListLayoutMode == MODE_LINEAR) {
35         MenuItem item = menu.findItem(R.id.menu_linear);
36         item.setChecked(true);
37     } else {
38         menu.findItem(R.id.menu_grid).setChecked(true);
39     }
40     return super.onPrepareOptionsMenu(menu);
41 }
```

```
42 }
```

- 搜索

提交标题关键字查询数据库，刷新笔记列表显示。

```
▼ NoteActivity.java Java | 复制代码

1 public boolean onCreateOptionsMenu(Menu menu) {
2     getMenuInflater().inflate(R.menu.menu_main, menu);
3     SearchView searchView = (SearchView) menu.findItem(R.id.menu_s
    earch).getActionView();
4
5     searchView.setOnQueryTextListener(new SearchView.OnQueryTextLi
        stener() {
6         @Override
7         public boolean onQueryTextSubmit(String query) {
8             return false;
9         }
10
11         @Override
12         public boolean onQueryTextChange(String newText) {
13             mNotes = mNoteDbOpenHelper.queryFromDbByTitle(newText)
14             ;
15             //刷新数据库
16             mMyAdapter.refreshData(mNotes);
17             return true;
18         }
19     });
20     return super.onCreateOptionsMenu(menu);
}
```