中山大学移动信息工程学院本科生实验报告

(2017年秋季学期)

课程名称：移动应用开发

年级	专业方向	学号	姓名
1501	移动（互联网）	15352005	蔡景韬

一、实验题目

1.1 项目简介

- 期末项目我们组做的主要是，实现一个简单的便签软件
  - 主界面实现双视图预览——单列显示、瀑布流显示
  - 主界面可以对文本进行批量删除、加锁、按文本内容搜索等功能
  - 主界面可以将本地数据库同步到云端服务器的数据库上，也可以拉取云端服务器的数据库文本内容
  - 而富文本编辑界面，可以编写文本，并实现加粗、斜体等功能，也可以插入图片，记录当前修改时间，并保存到本地数据库；实现转发、删除、加锁；
  - 筛选界面可以根据写作日期，展示一年内几个月写了多少篇文章，并通过勾选筛选出相应的文本在主界面显示

1.2 本人担任的部分

- 我主要实现富文本界面的编写工作

二、实现内容

1. 富文本界面的编辑文本功能，并记录当前修改时间
2. 转发功能
3. 加锁功能
4. 删除文本

三、课堂实验结果

A. 实验截图

B. 实验步骤以及关键代码

B.1 RichEditor的实现

- 富文本编辑需求主要有两种实现思路
  - EditText + Span 的实现方式
  - WebView + JavaScript 的实现方式
- 本次项目主要使用`'jp.wasabeef:richeditor-android:1.2.2'`依赖实现RichEditor，该实现是采用的第二种思路：WebView + JavaScript
- `RichEditor` 是一个继承自 `WebView` 的自定义 view，枚举类型 `Type` 定了它所支持的排版格式

```
public enum Type {
    BOLD,
    ITALIC,
    SUBSCRIPT,
    SUPERSCRIPT,
    STRIKETHROUGH,
    UNDERLINE,
    H1,
    H2,
    H3,
    H4,
    H5,
    H6,
    ORDEREDLIST,
    UNORDEREDLIST,
    JUSTIFYCENTER,
    JUSTIFYFULL,
    JUSTUFYLEFT,
    JUSTIFYRIGHT
}
```

- 首先在构造函数中加载一个 html 文件
- 其中 html 文件 - `editor.html` 加载了两个 css 文件 - `normalize.css` & `style.css`，一个 js 文件 - `rich_editor.js`。

14 lines (13 sloc) | 422 Bytes

```html
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="user-scalable=no">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="normalize.css">
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id="editor" contenteditable="true"></div>
<script type="text/javascript" src="rich_editor.js"></script>
</body>
</html>
```

- `rich_editor.js` 中定义了很多排版功能的 API，`RichEditor` 类似 proxy，对外提供了 Java API

```
RE.getHtml = function() {
    return RE.editor.innerHTML;
}

RE.getText = function() {
    return RE.editor.innerText;
}

RE.setBaseTextColor = function(color) {
    RE.editor.style.color  = color;
}

RE.setBaseFontSize = function(size) {
    RE.editor.style.fontSize = size;
}
```

- 其中 `exec` 用于执行 js 代码

```
public void setEditorFontColor(int color) {
  String hex = convertHexColorString(color);
  exec("javascript:RE.setBaseTextColor('" + hex + "');");
}
```

- 总之，编辑器的核心功能由 js 实现，RichEditor 封装了 js 的功能，为上层提供了 java 接口。

B.2 RichEditor相应功能的实现

1. 居中、加粗、斜体、中划线、撤销操作的实现

```
findViewById(R.id.centerBtn).setOnClickListener((view) -> {
        if ( !ifSetCenter ){
            richEditor.setAlignCenter();
            ifSetCenter = true ;
        }
        else {
            richEditor.setAlignLeft();
            ifSetCenter = false ;
        }

});
findViewById(R.id.boldBtn).setOnClickListener((view) -> {
        richEditor.setBold();
});
findViewById(R.id.italicBtn).setOnClickListener((view) -> {
        richEditor.setItalic();
});
findViewById(R.id.biggerBtn).setOnClickListener((view) -> {
        richEditor.setStrikeThrough();
});
findViewById(R.id.undo).setOnClickListener((view) -> {
        richEditor.undo();
});
```

   居中功能需要判断是否点击多次，点击一次居中，点击两次则为正常文档流——居左

   而加粗、中划线、斜体在执行js代码时会实现偶次数执行代码取消相应的效果

2. 插入图片功能的实现
    - 插入图片是实现富文本编辑功能的难点
    - 基本思路是：

从相册或者照相获取图片——裁剪图片——保存图片到本地（使用保存时间作为文件名）——使用本地链接显示到富文本编辑器上

- 从相册或照相获取图片

1. 相册Intent

```
Intent intent = new Intent() ;
intent.setType("image/*") ;
// 调用系统相册
intent.setAction(Intent.ACTION_GET_CONTENT) ;
startActivityForResult(intent,1);
```

```
// 打开相册的回调函数
if ( requestCode==1 ){
    if ( data!=null ){
        Uri uri = data.getData();    // 获取返回的uri
        cutImg(uri);     // 调用剪切函数
    }
}
```

2. 照相Intent

```
Intent intent = new Intent();
// 调用系统的照相机
intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
imageUri = Uri.parse("file://" + "/" +
                     Environment.getExternalStorageDirectory().getPath()
                     + "/" + "tmp.jpg");
// 将返回的数据直接保存到tmp.jpg，并将其uri保存为imageUri
intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
startActivityForResult(intent, 3);
```

```
 // 打开相机的回调函数
if ( requestCode == 3 ){
    cutImg(imageUri);    // 将保存的tmp.jpg转化的Uri传给剪切函数
}
```

- 剪切图片

通过调用系统剪切Intent进行剪切动作

```
// 裁剪图片
void cutImg(Uri uri) {
    if (uri != null) {
        Intent intent = new Intent("com.android.camera.action.CROP");
        intent.setDataAndType(uri, "image/*");
        //true:出现裁剪的框
        intent.putExtra("crop", "true");
        //裁剪宽高时的比例
        intent.putExtra("aspectX", 1);
        intent.putExtra("aspectY", 1);
```

```
        //裁剪后的图片的大小
        intent.putExtra("outputX", 150);
        intent.putExtra("outputY", 150);
        /**
         * 此方法返回的图片只能是小图片（sumsang测试为高宽160px的图片）
         * 故将图片保存在Uri中，调用时将Uri转换为Bitmap
         * 此方法还可解决miui系统不能return data的问题
         */
        //uritempFile为Uri类变量，实例化uritempFile
        uriTempFile = Uri.parse("file://" + "/" +
                                Environment.getExternalStorageDirectory().getPath()
                                + "/" + "small.jpg");
        intent.putExtra(MediaStore.EXTRA_OUTPUT, uriTempFile);
        intent.putExtra("outputFormat", Bitmap.CompressFormat.JPEG.toString());
        startActivityForResult(intent, 2);
    }
    else { return; }
}
```

```
 // 打开系统裁剪界面的回调函数
 if (requestCode == 2) {
     if (data != null) {
         //将Uri图片转换为Bitmap
         Bitmap bitmap = null;
         try {
             bitmap = BitmapFactory.decodeStream(getContentResolver()
                                     .openInputStream(uriTempFile));
         } catch (FileNotFoundException e) { e.printStackTrace(); }
        //保存到图片到本地
        saveImg(bitmap);
     } else { return; }
}
```

- 保存图片

```
// 保存图片
void saveImg(Bitmap bitmap) {
    try {
        // 创建文件夹路径
        File destDir = new File(fileBasePath);
        if (!destDir.exists()) {    // 如果不存在该文件夹，则创建该文件夹
            destDir.mkdirs();
        }
        // 创建图片文件，路径为上文的文件夹子目录，将此时系统时间作为图片文件名
        File f = new File(fileBasePath + System.currentTimeMillis()+ ".png");
        f.createNewFile();
        //输出流
        FileOutputStream out = new FileOutputStream(f);
        /** mBitmap.compress 压缩图片
         *  Bitmap.CompressFormat.PNG   图片的格式
         *   100  图片的质量（0-100）
         *   out  文件输出流
```

```
         */
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
        out.flush();
        out.close();
        // 将该图片插入富文本编辑器中
        richEditor.insertImage(filePath,"png");
    } catch (IOException e) { e.printStackTrace(); }
}
```

- 使用`Dialog`实现一个从底部向上弹出的菜单栏，用于选择图片来源，相册或者相机

```
// 选择相册或相机的dialog
void setPictureMenuDialog(){
    pictureMenu = new Dialog(this,R.style.bottom_dialog) ;
    //填充对话框的布局
    final LinearLayout linearLayout =
                (LinearLayout) LayoutInflater.from(this)
                                    .inflate(R.layout.layout_picture_menu, null);
    //将布局设置给Dialog
    pictureMenu.setContentView(linearLayout);
    //按键初始化
    cameraBtn = (Button) linearLayout.findViewById(R.id.cameraBtn);
    albumBtn = (Button) linearLayout.findViewById(R.id.albumBtn);
    //获取当前Activity所在的窗体
    Window dialogWindow = pictureMenu.getWindow();
    //设置Dialog从窗体底部弹出
    dialogWindow.setGravity(Gravity.BOTTOM);
    // 设置Dialog动画
    dialogWindow.setWindowAnimations(R.style.dialog_style);
    //获得窗体的属性
    WindowManager.LayoutParams lp = dialogWindow.getAttributes();
    // 窗体距离底部的距离
    lp.y = 150+20;  // 150为虚拟机虚拟键盘的高度
    // 窗体的高度、宽度和透明度
    lp.width = getResources().getDisplayMetrics().widthPixels-50; // 宽度
    linearLayout.measure(0, 0);
    lp.height = linearLayout.getMeasuredHeight(); // 获取layout的高度
    lp.alpha = 9f;       // 设置窗体的透明度
    // 将属性设置给窗体
    dialogWindow.setAttributes(lp);
    // 点击外界面时撤除Diglog
    pictureMenu.setCanceledOnTouchOutside(true);
    // 显示Diglog
    pictureMenu.show();
    // 按键响应
    PictureMenuRespond() ;
}
```

Diglog的基本style属性`R.style.bottom_dialog`

```xml
<style name="bottom_dialog">
    <item name="android:windowNoTitle">true</item>       <!-- 全屏显示 -->
    <item name="android:backgroundDimEnabled">true</item><!-- 弹出Dialog时背景变暗 --
>
</style>
```

Diglog的动画属性`R.style.dialog_style`

```xml
<style name="dialog_style" parent="android:Animation">
    <item name="@android:windowEnterAnimation">@anim/dialog_enter</item>
    <item name="@android:windowExitAnimation">@anim/dialog_exit</item>
</style>
```

```xml
<!-- dialog_enter -->
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromYDelta="100%p"
        android:duration="300"/>
</set>
```

```xml
<!-- dialog_exit -->
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:toYDelta="100%p"
        android:duration="300"/>
</set>
```

3. RichEditor功能选择，左右滑动的实现

使用`HorizontalScrollView`，其中`android:scrollbars="none"`，取消滚动条

4. 保存html代码到本地数据库，读取html代码到富文本编辑器

```
// 获得编辑器的html代码
richEditor.getHtml();
// 读取html代码到编辑器中
richEditor.setHtml(htmlCode);
```

5. 实现监听软键盘弹出收起——弹出时显示编辑器工具（撤销、插图等），收起时显示便签工具（转发、加锁等）并取消编辑器焦点

   实现监听类：根据根视图在屏幕中显示的大小判断是否弹出软键盘

```
public class SoftKeyBoardListener {

    private View rootView;//activity的根视图
    int rootViewVisibleHeight;//纪录根视图的显示高度
    private OnSoftKeyBoardChangeListener onSoftKeyBoardChangeListener;

    public SoftKeyBoardListener(Activity activity) {
        //获取activity的根视图
        rootView = activity.getWindow().getDecorView();

        //监听视图树中全局布局发生改变或者视图树中的某个视图的可视状态发生改变
        rootView.getViewTreeObserver().addOnGlobalLayoutListener(() -> {
                //获取当前根视图在屏幕上显示的大小
                Rect r = new Rect();
                rootView.getWindowVisibleDisplayFrame(r);

                int visibleHeight = r.height();
                System.out.println(""+visibleHeight);
                if (rootViewVisibleHeight == 0) {
                    rootViewVisibleHeight = visibleHeight;
                    return;
                }

                //根视图显示高度没有变化，可以看作软键盘显示/隐藏状态没有改变
                if (rootViewVisibleHeight == visibleHeight) {
                    return;
                }

                //根视图显示高度变小超过200，可以看作软键盘显示了
                if (rootViewVisibleHeight - visibleHeight > 200) {
                    if (onSoftKeyBoardChangeListener != null) {
                        onSoftKeyBoardChangeListener.keyBoardShow(rootViewVisibleHeight - visibleHeight);
                    }
                    rootViewVisibleHeight = visibleHeight;
                    return;
                }

                //根视图显示高度变大超过200，可以看作软键盘隐藏了
                if (visibleHeight - rootViewVisibleHeight > 200) {
                    if (onSoftKeyBoardChangeListener != null) {
                        onSoftKeyBoardChangeListener.keyBoardHide(visibleHeight - rootViewVisibleHeight);
                    }
                    rootViewVisibleHeight = visibleHeight;
                    return;
                }

        });
    }

    private void setOnSoftKeyBoardChangeListener(OnSoftKeyBoardChangeListener onSoftKeyBoardChangeListener) {
        this.onSoftKeyBoardChangeListener = onSoftKeyBoardChangeListener;
    }

    public interface OnSoftKeyBoardChangeListener {
        void keyBoardShow(int height);
        void keyBoardHide(int height);
    }

    public static void setListener(Activity activity, OnSoftKeyBoardChangeListener onSoftKeyBoardChangeListener) {
        SoftKeyBoardListener softKeyBoardListener = new SoftKeyBoardListener(activity);
        softKeyBoardListener.setOnSoftKeyBoardChangeListener(onSoftKeyBoardChangeListener);
    }
}
```

   实现监听事件

```
// 软键盘弹出收起的监听响应
void softKeyBoardListener(){
    SoftKeyBoardListener.setListener(
                          NoteActivity.this,
                        new SoftKeyBoardListener.OnSoftKeyBoardChangeListener() {
        @Override
        public void keyBoardShow(int height) {
//           Toast.makeText(NoteActivity.this,"键盘显示",Toast.LENGTH_SHORT).show();
             editTool.setVisibility(View.VISIBLE);
             noteTool.setVisibility(View.GONE);
        }
        @Override
        public void keyBoardHide(int height) {
//          Toast.makeText(NoteActivity.this,"键盘隐藏",Toast.LENGTH_SHORT).show();
            richEditor.clearFocus() ;
            editTool.setVisibility(View.GONE);
            noteTool.setVisibility(View.VISIBLE);
        }
    });
}
```

B.3 便签工具相应功能的实现

- 转发功能：直接调用系统的转发功能

```
findViewById(R.id.shareBtn).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent=new Intent(Intent.ACTION_SEND);
        intent.putExtra(Intent.EXTRA_TEXT, richEditor.getHtml());
        intent.setType("text/plain");
        startActivity(Intent.createChooser(intent,"分享给"));
    }
});
```

- 删除功能

```
findViewById(R.id.delBtn).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(NoteActivity.this);
        alertDialog.setTitle("是否删除")
                .setPositiveButton("是", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // 如果原先存在该便签，则从数据库中删除
                        if( id>0 ){
                            db_helper.getWritableDatabase()
                              .execSQL("DELETE FROM Article WHERE id =" + id + " ");
                        }
                        // 否则直接结束编辑界面即可
                        finish();
```

```
                    }
                })
                .setNegativeButton("否", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {}
                })
                .create().show();
    }
});
```

- 加锁功能

```
findViewById(R.id.lockBtn).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // 自定义对话框:LayoutInflater
        LayoutInflater layoutInflater = LayoutInflater.from(NoteActivity.this);
        View alertDialogLayout = layoutInflater.inflate(R.layout.layout_dialog,null);
        final TextView passwordET = (TextView)
                                alertDialogLayout.findViewById(R.id.passwordET) ;
        final TextView repetitionET = (TextView)
                                alertDialogLayout.findViewById(R.id.repetitionET) ;
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(NoteActivity.this);
        alertDialog.setView(alertDialogLayout)
                .setTitle("修改/添加密码")
                .setPositiveButton("保存", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        String password = passwordET.getText().toString() ;
                        String repetition = repetitionET.getText().toString() ;
                        if( !password.equals(repetition) )
                            Toast.makeText(NoteActivity.this,
                                           "重复密码有误",Toast.LENGTH_SHORT).show();
                        else{
                            // 使用MD5加密密码
                            pass = MD5Utils.md5Password(password) ;
                        }
                    }
                })
                .setNegativeButton("放弃", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {}
                })
                .create().show();
    }
});
```

- 回传数据保存到数据库中（返回键的操作跟下面一样）

```
//重写返回键的操作
@Override

public void onBackPressed(){
```

```
    if ( id>0 ){     // 如果是修改文本
        // 修改文本：回写到数据库
        String sql = "UPDATE Article set article='" + richEditor.getHtml() + "'" ;
        // 如果过程有加密，则回传时将标签locked设为1，并将密码赋给password属性
        if ( !pass.equals("") ) sql += ", locked="+1+", password='"+pass+"'" ;
        sql += " WHERE id="+id ;
        db_helper.getWritableDatabase().execSQL(sql);
    }
    else{    // 如果是新建文本
        // 获取当前时间—年月日
        Calendar calendar=Calendar.getInstance();
        ContentValues values = new ContentValues() ;
        values.put("article",richEditor.getHtml());
        values.put("year",calendar.get(Calendar.YEAR));
        values.put("month",calendar.get(Calendar.MONTH)+1);
        values.put("day",calendar.get(Calendar.DAY_OF_MONTH));
        // 之前保存的进入编辑器的事件
        values.put("time",time);
        // 判断是否有加锁
        if ( pass.equals("") ) values.put("locked",0);
        else{
            values.put("locked",1);
            values.put("password",pass);
        }
        // 插入数据库新文本
        db_helper.getWritableDatabase().insert("Article",null,values) ;
    }
    finish() ;
}
```

至此，关于富文本编辑器的内容实现结束