

# 软件设计文档

——TripJournal

## Object-Oriented Programming

Java 本来就是面向对象的语言，所有的代码都写在类中，所以很自然有面向对象编程。

举例：

代码来自 MyMap.java

```
public class MyMap extends Activity implements OnMapLoadedCallback {  
    TextureMapView mMapView;  
    BaiduMap mBaiduMap;  
    MapStatus ms;  
    private ClusterManager<MyItem> mClusterManager;  
    private SQLiteDatabase mDatabase;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        @Override  
        protected void onPause() {  
            mMapView.onPause();  
            super.onPause();  
        }  
  
        @Override  
        protected void onResume() {  
            mMapView.onResume();  
            super.onResume();  
        }  
  
        @Override  
        protected void onDestroy() {  
            mMapView.onDestroy();  
            super.onDestroy();  
        }  
}
```

这部分是实现在 APP 中展现地图，并在地图上标记写过日记的地点，以地图为类，地图相关的操作都写在该类中。

## Service-Oriented Architecture

在 MyPicture.java 模块中使用到取色器，把这部分的代码抽取出来，写成

一个独立的模块，然后开放接口，在 MyPicture 模块中直接使用取色器服务提供的接口，取色器内部的修改不会影响到提供的接口，保持相互独立，实现了松耦合。

```
public class ColorPickerView extends View{

    private String TAG="colorPicker";
    private boolean DBG=false;
    private Context mContext;
    private Paint mRightPaint;           //画笔
    private int mHeight;                 //view高
    private int mWidth;                  //view宽
    private int[] mRightColors;
    private int LEFT_WIDTH;
    private Bitmap mLeftBitmap;
    private Bitmap mLeftBitmap2;
    private Paint mBitmapPaint;
    private PointF mLeftSelectPoint;
    private OnColorChangeListenerD mChangeListenerD;
    private boolean mLeftMove = false;
    private float mLeftBitmapRadius;
    private Bitmap mGradualChangeBitmap;
    private Bitmap bitmapTemp;
```

## Design Patterns

在 MyPicture 模块中，由于生成图片比较耗时，所以另外开启一个线程来处理图片的生成和保存，使用到了命令模式。

线程创建 Handler 和 Runnable 实例

```
void newThread() {
    mHandler = new Handler();

    mRunnable = new Runnable() {
        @Override
        public void run() {
            outPic.measure(View.MeasureSpec.makeMeasureSpec(0, View.MeasureSpec.UNSPECIFIED),
                View.MeasureSpec.makeMeasureSpec(0, View.MeasureSpec.UNSPECIFIED));
            outPic.layout(0, 0, outPic.getMeasuredWidth(), outPic.getMeasuredHeight());
            outPic.buildDrawingCache();
            outPic.buildDrawingCache();
            Bitmap bitmap;
            try {
                bitmap = outPic.getDrawingCache();
                uri = functions.saveImage(MyPicture.this, bitmap);
                Intent intent = new Intent("TheStringUsedAsStaticFilter");
                intent.putExtra("uri", uri.toString());
                sendBroadcast(intent);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
}
```

需要保存图片的时候 `mHandler.post(mRunnable);` 就会运行

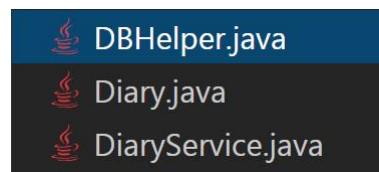
mRunnable 中的 run 函数保存图片。

## Aspect-Oriented Programming

AOP 则是针对业务处理过程中的切面进行提取，它所面对的是处理过程中的某个步骤或阶段，以获得逻辑过程中各部分之间低耦合性的隔离效果。在本项目中，实现了多种功能，但功能的各个部分是单独实现的。因此具有较低的耦合性和较高的可扩展性。

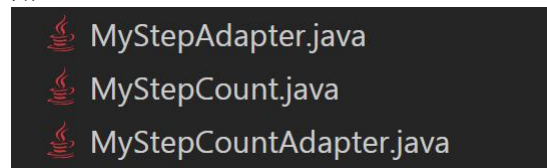
项目实现了多种功能：

### 1. 写日记功能



这三个文件实现的是日记功能。其中 DBHelper 提供数据库，DiaryService 主要实现对数据库的增删查改，Diary 主要是获取每篇日记的信息。

### 2. 计步器功能



这个三个文件实现的是计步器功能。其中 MyStepCountAdapter 主要实现获取步数记录，MyStepCount 主要实现的是新建一个计步器、开启后台等。

### 3. 图片功能



这个文件主要实现的就是生成图片的功能。而在这个文件中还会引用两

个文件：ColorPickerDialog.java 和 ColorPickerView.java。这两个文件是关于取色和设置图片样式的。

以上就是我们实现的功能，各个部分是单独进行的。调用过程中各个功能不会有交叉，这样保证了一个功能不能实现的情况下不会牵扯到其他功能的修改。

MainActivity 中我们是这样实现各个功能的：点击不同按钮，会跳转到相应功能里。

跳转到地图：

```
mapButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent();  
        intent.setClass(MainActivity.this, MyMap.class);  
        startActivity(intent);  
    }  
});
```

跳转到计步器页面：

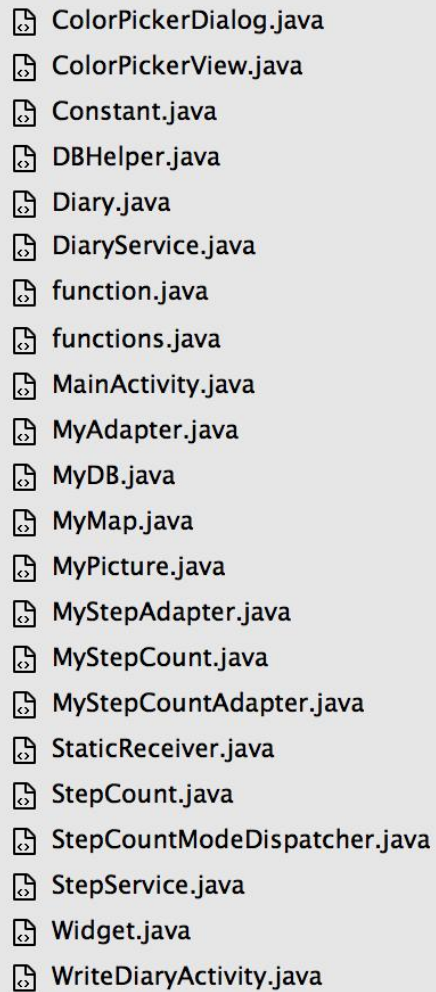
```
stepButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent();  
        intent.setClass(MainActivity.this, MyStepCount.class);  
        startActivity(intent);  
    }  
});
```

跳转到制作图片页面：

```
picButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent();  
        intent.setClass(MainActivity.this, MyPicture.class);  
        startActivity(intent);  
    }  
});
```

而在每一个功能中，还有自己的子逻辑。比如可能从写日记页面跳转到制作图片页面，此时也只需要调用制作图片的接口就可以实现了。

## Structure Programming SD



- ColorPickerDialog.java
- ColorPickerView.java
- Constant.java
- DBHelper.java
- Diary.java
- DiaryService.java
- function.java
- functions.java
- MainActivity.java
- MyAdapter.java
- MyDB.java
- MyMap.java
- MyPicture.java
- MyStepAdapter.java
- MyStepCount.java
- MyStepCountAdapter.java
- StaticReceiver.java
- StepCount.java
- StepCountModeDispatcher.java
- StepService.java
- Widget.java
- WriteDiaryActivity.java

单个组件单独实现，不同的功能分成不同的 java 文件实现，比如我们分工时，陈舒仪负责步数记录和主页的 coding，卢诗娟负责 Mini 日记和足迹 coding，陈娅负责数据库建立，写日记和界面的 coding，那我们就根据自己要实现的功能编写不同的 java 文件然后整合到一起。

功能实现代码和界面代码也在不同的目录下，不同的目录各尽其责：

