Android 天气预报简单记录（二）

主要内容

- 使用RecyclerView显示所有城市和未来4天天气
- 使用SearchView来实现查询
- 基于百度地图API实现定位，更新天气
- ViewPager实现介绍界面

使用RecyclerView显示所有城市和未来4天天气

RecyclerView是support.v7包中的控件，和ListView类似，十分的灵活，在有限窗口展示大量数据集。因为ListView性能问题（迫使我们不得不使用自定义 ViewHolder 和 convertView 一起来完成复用优化工作）以及仅支持竖向滚动的局限性，Google官方重新定义了一个滚动控件，也就是RecyclerView。

关键点在于：

- 需要重写 RecyclerView.Adapter 和 RecyclerView.ViewHolder
- 设置布局管理器，控制布局效果（决定item的展示方向，以及复用不可见的item）

也就是，ViewHolder 的编写规范化了；
RecyclerView 复用 Item 的工作 Google 全帮你搞定，不再需要像 ListView 那样自己调用 setTag；
RecyclerView 需要多出一步 LayoutManager 的设置工作；

接下来，贴出的代码都带有比较详细的注释。

第一步，先在grade文件里面添加依赖：（后面的版本号根据自身项目的SDK选择）

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

第二步，布局文件XML引入控件，以及对应的item布局的构造（这里不再赘述）

```
<android.support.v7.widget.RecyclerView
        android:id="@+id/city_list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:color/background_light"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/city_sv" />
```

第三步，实现RecyclerView.Adapter

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder>{
    private List<City> list; //数据

    public MyAdapter(List<City> list) { // 构造函数
        this.list = list;
    }

    // 实现ViewHolder，这里每一个item里面展示 省份，地区，以及城市代码
    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView cityProvince;
        TextView cityName;
        TextView cityCode;
        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            cityProvince = itemView.findViewById(R.id.city_province);
            cityName = itemView.findViewById(R.id.city_name_tv);
            cityCode = itemView.findViewById(R.id.city_code_tv);
        }
    }

    // 实例化View，实例化viewHolder
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View v = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.city_item
        ViewHolder viewHolder = new ViewHolder(v);
        return viewHolder;
    }

    // 绑定数据
    @Override
    public void onBindViewHolder(@NonNull ViewHolder viewHolder, final int i) {
        viewHolder.cityProvince.setText(list.get(i).getProvince());
        viewHolder.cityName.setText(list.get(i).getCity());
        viewHolder.cityCode.setText(list.get(i).getNumber());
        viewHolder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mOnItemClickListener != null){
                    mOnItemClickListener.onItemClick(v, i);
                }
            }
        });
    }

    @Override
    public int getItemCount() {
        return list == null ? 0 : list.size();
    }
}
```

第四步，写好对应的Activity部分代码：

```
public class SelectCity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_select_city);

        initData();
        initView();
    }
    // 数据准备，包括布局管理器设置
    private void initData() {
        MyApplication myApplication = (MyApplication) getApplication();
        mLayoutManager = new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, fa
        list = myApplication.getList();
        mAdapter = new MyAdapter(list);
    }
    // 初始化视图以及各种点击事件
    private void initView() {
        // 得到当前城市名称
        titleCityName = findViewById(R.id.title_name_tv);
        Intent intent = getIntent();
        final String name = intent.getStringExtra("cityName");
        Log.d("intent", "initView: " + name );
        titleCityName.setText("当前城市：" + name);

        // 左上角回退
        myBack = findViewById(R.id.title_back_img);
        myBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

        // 点击事件
        mAdapter.setOnItemClickListener(new MyAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(View view, int position) {
                City city = list.get(position);
                Intent intent = new Intent();
                intent.putExtra("cityCode", city.getNumber());
                intent.putExtra("cityName", city.getCity());
                intent.putExtra("province",city.getProvince());
                setResult(RESULT_OK, intent);
                finish();
            }
        });
        // RecyclerView设置布局效果，绑定带数据的适配器，添加滚动方向
        mRecyclerView = findViewById(R.id.city_list);
        mRecyclerView.setLayoutManager(mLayoutManager);
        mRecyclerView.setAdapter(mAdapter);
        mRecyclerView.addItemDecoration(new DividerItemDecoration(this, DividerItemDecc
    }
}
```

这就是RecyclerView使用的三板斧了，这里还出现了对于item的点击操作，需要注意这里并没有像ListView一样有供外接调用的API，需要我们自己去实现item的点击处理。这里我的思路是，ViewHolder绑定数据时设置监听，,然后利用Adapter进行回调。

```
// 在前面出现过的MyAdapter类里添加

private OnItemClickListener mOnItemClickListener;

public interface OnItemClickListener {
        void onItemClick(View view, int position);
}

// 回调监听
public void setOnItemClickListener(MyAdapter.OnItemClickListener listener) {
        mOnItemClickListener = listener;
}

@Override
public void onBindViewHolder(@NonNull ViewHolder viewHolder, final int i) {
        viewHolder.cityProvince.setText(list.get(i).getProvince());
        viewHolder.cityName.setText(list.get(i).getCity());
        viewHolder.cityCode.setText(list.get(i).getNumber());
        // 设置监听
        viewHolder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mOnItemClickListener != null){
                    mOnItemClickListener.onItemClick(v, i);
                }
            }
        });
}


// 然后在对应的Activity里面设置对应的setOnItemClickListener即可
// 例如:
mAdapter.setOnItemClickListener(new MyAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(View view, int position) {
                City city = list.get(position);
                Intent intent = new Intent();
                intent.putExtra("cityCode", city.getNumber());
                intent.putExtra("cityName", city.getCity());
                intent.putExtra("province",city.getProvince());
                setResult(RESULT_OK, intent);
                finish();
            }
});
```

那么现在我们已经利用RecyclerView实现了城市列表的展示，同理横向展示未来四天的数据就可以依样画葫芦了（未来四天天气数据的获取前一篇已介绍），下面直接贴出相应代码：

> item的布局文件

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent">


    <TextView
        android:id="@+id/each_day_week"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        android:text="TextView"
        android:textColor="@android:color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/each_day_type"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/each_day_week"
        app:srcCompat="@drawable/biz_plugin_weather_yin" />

    <TextView
        android:id="@+id/each_day_temperature"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:text="TextView"
        android:textColor="@android:color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/each_day_type" />

    <TextView
        android:id="@+id/each_day_climate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
```

```
        android:layout_marginTop="8dp"
        android:text="TextView"
        android:textColor="@android:color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/each_day_temperature" />

    <TextView
        android:id="@+id/each_day_wind"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="TextView"
        android:textColor="@android:color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/each_day_climate" />
</android.support.constraint.ConstraintLayout>
```

> Adapter

```
public class DayOfWeekForecastAdapter extends RecyclerView.Adapter<DayOfWeekForecastAda

    private List<ForecastWeather> list;
    private Map<String, Integer> imgSrcForWeather;
    private Calendar calendar;

    public DayOfWeekForecastAdapter(List<ForecastWeather> list, Map<String, Integer> im
        this.list = list;
        this.imgSrcForWeather = imgSrcForWeather;
        calendar = Calendar.getInstance();
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.day_week_weather_item, viewGroup, false);
        ViewHolder viewHolder = new ViewHolder(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
        String type = list.get(i).getType();
        viewHolder.eachDayWeek.setText((calendar.get(Calendar.MONTH) + 1) + "月" + list
        viewHolder.eachDayTemperature.setText(list.get(i).getLow() + " ~ " + list.get(i
        viewHolder.eachDayClimate.setText(type);
        viewHolder.eachDayWind.setText(list.get(i).getFengxiang());
        viewHolder.eachDayType.setImageResource(imgSrcForWeather.get(type));
    }

    @Override
    public int getItemCount() {
        return list == null ? 0 : list.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {

        TextView eachDayWeek;
        ImageView eachDayType;
        TextView eachDayTemperature;
        TextView eachDayClimate;
        TextView eachDayWind;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            eachDayWeek = itemView.findViewById(R.id.each_day_week);
            eachDayType = itemView.findViewById(R.id.each_day_type);
            eachDayTemperature = itemView.findViewById(R.id.each_day_temperature);
            eachDayClimate = itemView.findViewById(R.id.each_day_climate);
            eachDayWind = itemView.findViewById(R.id.each_day_wind);
        }
    }
}
```

> Activity

```
mLayoutManager = new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL, false);
mRecyclerView = findViewById(R.id.day_week_weather);
mRecyclerView.setLayoutManager(mLayoutManager);

mAdapter = new DayOfWeekForecastAdapter(info.getDayOfWeekForecastWeather(), imgSrcForWe
mRecyclerView.setAdapter(mAdapter);
```

结果

效果如图：

未来4天的天气

可选城市列表

使用SearchView来实现查询

直接讲一下我的思路，利用SearchView来作为搜索的控件，优点在于有内置的监听函数可以调用，方便实现，在每次得到控件内容时，对城市列表进行筛选操作（在这里实现对中文，拼音，首字母的筛

选），得到符合要求的有序的城市列表之后，重新传递给Adapter对象去展现。看起来十分的简单，下面就是具体的实现：

- 在选择城市视图界面添加上控件SearchView

```
<SearchView
        android:id="@+id/city_sv"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="@android:color/background_light"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/title_name_tv" />
```

- 在选择城市的Activity里面注册SearchView，并实现过滤函数

```
mSearchView = findViewById(R.id.city_sv);
mSearchView.setOnQueryTextListener(new SearchView.OnQueryTextListener(){
        @Override
        public boolean onQueryTextSubmit(String query) {
            return false;
        }
        // 当监听到SearchView里面的内容发生变化，
        // 调用过滤函数得到新的城市数据集，并更新Adapter适配器
        // 然后RecyclerView加载新的适配器展示城市列表
        @Override
        public boolean onQueryTextChange(String newText) {
            filterData(newText);
            mRecyclerView.setAdapter(mAdapter);
            return true;
        }
});
// 过滤函数
private void filterData(String newText) {
        filterDataList = new ArrayList<>();
        // 如果当前输入为空，展示全部数据
        if (TextUtils.isEmpty(newText)) {
            mAdapter = new MyAdapter(list);
        }else {
            // 清空过滤数据，因为每次输入的改变都会造成数据的变化
            filterDataList.clear();
            // 遍历列表里的城市，分别对城市名，全拼，首字母拼音进行筛选，就是判断当前输入是否为字符串
            for (City c : list) {
                if (c.getCity().contains(newText)) {
                    filterDataList.add(c);
                } else if (c.getAllPY().indexOf(newText.toUpperCase().toString()) == 0)
                    filterDataList.add(c);
                } else if (c.getAllFirstPY().indexOf(newText.toUpperCase().toString())
                    filterDataList.add(c);
                }
            }
            // 返回过滤数据的适配器
            mAdapter = mAdapter.updataView(filterDataList);
        }
        mAdapter.setOnItemClickListener(new MyAdapter.OnItemClickListener() {
            @Override
            public void onItemClick(View view, int position) {
                City city = filterDataList.get(position);
                Intent intent = new Intent();
                intent.putExtra("cityCode", city.getNumber());
                setResult(RESULT_OK, intent);
                finish();
            }
        });
}
```

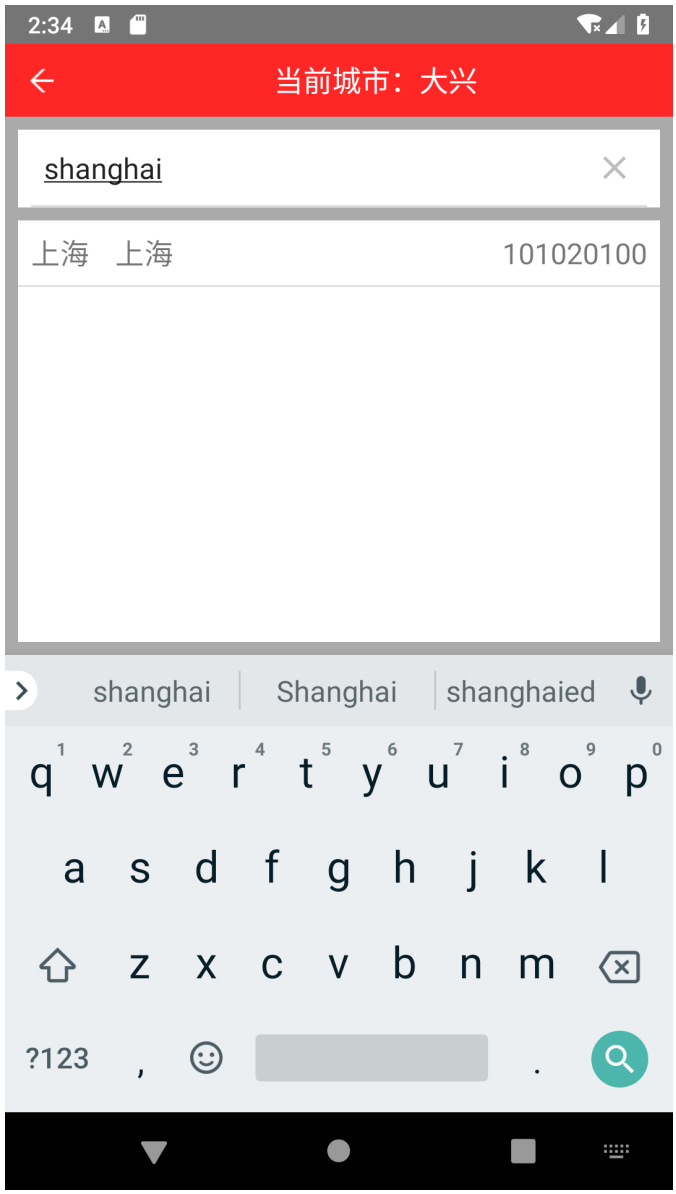
- 为了搜索后让列表有序排列，尽可能让我们希望的城市出现在上面，这里默认是字典排序，主要做法就是，对城市bean实现compareTo方法，调用Collection.sort()即可

```
public class City implements Comparable<City>{

    // ......

    @Override
    public int compareTo(@NonNull City o) {
        return o.getCity().compareTo(this.getCity()); // 用城市名进行比较
    }
}

public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder>{

    // ......

    public MyAdapter updataView(List<City> filterDataList) {
        Collections.sort(filterDataList);
        return new MyAdapter(filterDataList);
    }
}
```

结果

效果如图：

可选城市列表



搜索

基于百度地图API实现定位，更新天气

百度地图API实现定位的配置其实只要根据官方文档就够了，这里就不在描述什么申请key，在AndroidManifest.xml里面添加权限和服务等等，给出官方文档。安装要求操作即可。

同样走一遍实现思路，配置好要求，导入好SDK之后，我们就可以使用百度的定位API。如何实现我们想要的定位效果呢？构造定位服务对象，注册监听函数，配置相关选项（这里主要是设置详细地址为true），实现BDAbstractLocationListener接口，得到返回的定位信息去更新城市。当然具体细节，比如注销点击事件，动画旋转等等也需要考虑。具体代码如下：

```
// onCreate里面 构造对象，注册监听函数，对定位控件设置点击事件
locationClient = new LocationClient(getApplicationContext());
locationClient.registerLocationListener(new MyLocationListener());
locationImg = findViewById(R.id.title_location_img);
locationImg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        locationImg.setClickable(false);
        initLocation();
        updateAnimation(MainActivity.this, locationImg);
        locationClient.start();
    }
});


private void initLocation() {
        LocationClientOption option = new LocationClientOption();
        option.setIsNeedAddress(true); //需要地址信息
        locationClient.setLocOption(option);
}

// 设置需要旋转的图片对象
private void updateAnimation(Context context, ImageView update) {
        Animation circle_anim = AnimationUtils.loadAnimation(context, R.anim.anim_rounc
        LinearInterpolator interpolator = new LinearInterpolator();  //设置匀速旋转，在xm
        circle_anim.setInterpolator(interpolator);
        update.startAnimation(circle_anim);  //开始动画
}

// 实现BDAbstractLocationListener接口
public class MyLocationListener extends BDAbstractLocationListener {
        @Override
        public void onReceiveLocation(BDLocation bdLocation) {
            String city = bdLocation.getCity();
            String province = bdLocation.getProvince();
            city = city.substring(0, city.length() - 1); // 返回数据的是北京市这种格式，去掉

            for (City c : list) {
                if (c.getCity().equals(city)) {
                    cityCode = c.getNumber();
                    Log.d("Location: ", city + "  " + province + "  " + cityCode);
                    break;
                }
            }
            myUpdate.setClickable(false);
            updateAnimation(MainActivity.this, myUpdate);
            requestWeatherByCode(cityCode);//调用更新天气方法
            locationImg.clearAnimation();
            locationClient.stop();
            locationImg.setClickable(true);
        }
}
```

结果

效果如图：

原先天气信息

点击定位之后(返回北京市)

ViewPager实现介绍界面

介绍一下实现思路，核心就是ViewPager的addOnPageChangeListener方法，实现Adapter和对应的Fragment。这里有一个新的概念Fragment，就是片段，作为一个组件来嵌入到Activity里面，更多信息参考Google官方文档。

首先我先创建了一个新的Activity，选择的是Tabbed Activity。生成后，有三个文件，分别是一个Activity，一个activity的xml布局，一个fragment的xml布局。

- activity的xml布局，主要就是一个ViewPager，下面是三个button按钮和圆形的指示器

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context=".IntroductionActivity">

    <android.support.v4.view.ViewPager
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"

        />

    <View
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:layout_gravity="bottom"
        android:layout_marginBottom="?attr/actionBarSize"
        android:alpha="0.12"
        android:background="@android:color/white" />

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:layout_gravity="bottom"
        android:paddingEnd="@dimen/activity_horizontal_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingStart="@dimen/activity_horizontal_margin">


        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imageButtonPre"
            style="@style/Widget.AppCompat.Button.Borderless"
            android:contentDescription="pre"
            android:layout_gravity="start|center"
            android:padding="@dimen/activity_horizontal_margin"
            android:src="@drawable/ic_left_white_24dp"
            android:visibility="gone"
            android:tint="@android:color/white" />

        <!--圆形指示器-->
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:orientation="horizontal">
```

```
            <ImageView
                android:id="@+id/imageViewIndicator0"
                android:layout_width="8dp"
                android:layout_height="8dp"
                android:background="@drawable/introduction_indicator_selected" />

            <ImageView
                android:id="@+id/imageViewIndicator1"
                android:layout_width="8dp"
                android:layout_height="8dp"
                android:layout_marginEnd="8dp"
                android:layout_marginRight="8dp"
                android:layout_marginLeft="8dp"
                android:layout_marginStart="8dp"
                android:background="@drawable/introduction_indicator_unselected" />

            <ImageView
                android:id="@+id/imageViewIndicator2"
                style="@style/Widget.AppCompat.Button.Borderless"
                android:layout_width="8dp"
                android:layout_height="8dp"
                android:background="@drawable/introduction_indicator_unselected" />

        </LinearLayout>

        <android.support.v7.widget.AppCompatButton
            android:id="@+id/buttonFinish"
            style="@style/Widget.AppCompat.Button.Borderless"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end|center"
            android:text="Finish"
            android:contentDescription="Finish"
            android:textColor="@android:color/white"
            android:visibility="gone" />

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            style="@style/Widget.AppCompat.Button.Borderless"
            android:id="@+id/imageButtonNext"
            android:contentDescription="Next"
            android:layout_gravity="end|center"
            android:padding="@dimen/activity_horizontal_margin"
            android:src="@drawable/ic_right_white_24dp"
            android:tint="@android:color/white" />

    </FrameLayout>
</android.support.design.widget.CoordinatorLayout>
```

- fragment 的xml布局，就是里面放一个ImageView控件

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/constraintLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".IntroductionActivity$PlaceholderFragment">

    <ImageView
        android:id="@+id/introductionImg"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="?attr/actionBarSize"
        />
</FrameLayout>
```

- 最后就是Activity ， 主要就是切换fragment，改变背景颜色和图片，细节注释在代码里面：

```
public class IntroductionActivity extends AppCompatActivity {


    private SectionsPagerAdapter mSectionsPagerAdapter;


    private ViewPager mViewPager;
    private int bgColors[];

    private int currentPosition;

    private AppCompatButton buttonFinish;
    private ImageButton buttonPre;
    private ImageButton buttonNext;

    private ImageView[] indicators;

    private static Map<Integer, Integer> introduction;


    @SuppressLint("UseSparseArrays")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 作为一个引导性的界面，只有在应用开启的第一次才会出现，所以判断是不是第一次启动
        final SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(this

        if (sp.getBoolean("first_launch", true)){
            setContentView(R.layout.activity_introduction);

            // 得到引导界面的三张图片
            introduction = new HashMap<>();
            introduction.put(1,R.drawable.screenshot_1);
            introduction.put(2,R.drawable.screenshot_2);
            introduction.put(3,R.drawable.screenshot_3);

            // 初始化控件
            initView();
            // 得到引导界面的3种颜色
            bgColors = new int[]{ContextCompat.getColor(this, R.color.colorPrimary),
                    ContextCompat.getColor(this, R.color.cyan_500),
                    ContextCompat.getColor(this, R.color.light_blue_500)};

            // ViewPager的切换fragment的监听函数
            mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {

                // 滑动切换颜色
                @Override
                public void onPageScrolled(int i, float v, int i1) {
                    int colorUpdate = (Integer) new ArgbEvaluator().evaluate(v, bgColor
                    mViewPager.setBackgroundColor(colorUpdate);
                }
                // 不同fragment ，更新fragment里面的按钮和圆形指示器
```

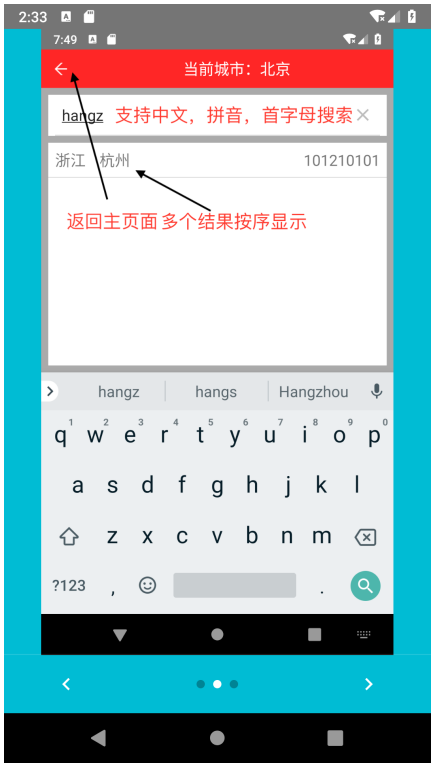
```
            @Override
            public void onPageSelected(int i) {
                currentPosition = i;
                updateIndicators(i);
                mViewPager.setBackgroundColor(bgColors[i]);
                buttonPre.setVisibility(i == 0 ? View.GONE : View.VISIBLE);
                buttonNext.setVisibility(i == 2 ? View.GONE : View.VISIBLE);
                buttonFinish.setVisibility(i == 2 ? View.VISIBLE : View.GONE);
            }

            @Override
            public void onPageScrollStateChanged(int i) {

            }
        });

        // finish 登记已启动信息，打开主页面
        buttonFinish.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SharedPreferences.Editor ed = sp.edit();
                ed.putBoolean("first_launch", false);
                ed.apply();
                navigateToMainActivity();

            }
        });

        // 切换fragment
        buttonNext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                currentPosition += 1;
                mViewPager.setCurrentItem(currentPosition, true);
            }
        });

        buttonPre.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                currentPosition -= 1;
                mViewPager.setCurrentItem(currentPosition, true);
            }
        });


    }else {
        navigateToMainActivity();
        finish();
    }

}
// 跳转到主页面
private void navigateToMainActivity() {
    Intent i = new Intent(this, MainActivity.class);
```

```
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(i);
    }

    // 初始化控件
    private void initView(){
        mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());
        mViewPager = findViewById(R.id.container);
        mViewPager.setAdapter(mSectionsPagerAdapter);
        buttonFinish = findViewById(R.id.buttonFinish);
        buttonFinish.setText("finish");
        buttonFinish.setEnabled(true);
        buttonNext = findViewById(R.id.imageButtonNext);
        buttonPre = findViewById(R.id.imageButtonPre);
        indicators = new ImageView[] {
                findViewById(R.id.imageViewIndicator0),
                findViewById(R.id.imageViewIndicator1),
                findViewById(R.id.imageViewIndicator2) };
    }

    // 更新圆形指示器，选择亮或者不亮
    private void updateIndicators(int position) {
        for (int i = 0; i < indicators.length; i++) {
            indicators[i].setBackgroundResource(
                    i == position ? R.drawable.introduction_indicator_selected :
                            R.drawable.introduction_indicator_unselected
            );
        }
    }



    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {

        private static final String ARG_SECTION_NUMBER = "section_number";

        public PlaceholderFragment() {
        }

        // 保存下sectionNumber进行传递
        public static PlaceholderFragment newInstance(int sectionNumber) {
            PlaceholderFragment fragment = new PlaceholderFragment();
            Bundle args = new Bundle();
            args.putInt(ARG_SECTION_NUMBER, sectionNumber);
            fragment.setArguments(args);
            return fragment;
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                 Bundle savedInstanceState) {
```

```
            // 初始化fragment里面的控件
            View rootView = inflater.inflate(R.layout.fragment_introduction, container,
            int num = getArguments().getInt(ARG_SECTION_NUMBER);
            // 根据当前的sectionNumber选择图片
            ImageView imageView = rootView.findViewById(R.id.introductionImg);
            imageView.setImageResource(introduction.get(num));
            return rootView;
        }
    }

    // 适配器
    public class SectionsPagerAdapter extends FragmentPagerAdapter {

        public SectionsPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {

            return PlaceholderFragment.newInstance(position + 1);
        }

        @Override
        public int getCount() {
            // Show 3 total pages.
            return 3;
        }
    }
}
```

结果

效果如图：（丑是丑了点，但道理就是这样的）

		
引导页1	引导页2	引导页3

总结

到这里全部就结束了，其实主要明白思路是怎么样，很多东西就迎刃而解，其他的东西，比如具体控件怎么使用，去官方文档或者一些博客了解一下API接口也就很快能实现。总的来说，只要思路清晰，有比较好的学习能力，剩下的需要就是实现功能的代码能力（优美而可读）。