

E1-11 增加搜索城市功能

一、实验目的

对城市选择界面进行优化，添加搜索城市的新功能。

二、实验要求

1. 对城市选择界面进行优化，将在界面中添加一个搜索框可以根据城市名或城市 id 进行搜索，并且显示搜索的结果。
2. 可以进行模糊搜索，自行确定正则式。
3. 可以有删除搜索的内容。

三、实验内容及步骤(注：以下代码仅供参考，涉及变量和 ID 不一样的请根据自己情况修改)

1. 在 Select_city.xml 文件中添加新的组件用来实现搜索框

```
<com.example.administrator.mini_weather.ClearEditText
    android:id="@+id/search_city"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4.0dip"
    android:hint="搜索全国城市(中文)"
    android:drawableLeft="@drawable/magnifying_glass"
    android:drawablePadding="6.0dp"
    android:singleLine="true"
    android:background="@drawable/contact_search_box_edittext_keyword_background"
    android:textColor="#ff000000"
    android:textColorHint="#ffcccccc"
    android:textSize="15.0dp"/>
```

2. 新建一个 java 文件用来处理搜索框的变化

如根据输入字符的长短进行变化，添加删除按钮等

Create New Class

Name: ClearEditText

Kind: Class

Superclass:

Interface(s):

Package: com.example.administrator.mini weather

Visibility: ☒ Public ☐ Package Private

Modifiers: ☒ None ☐ Abstract ☐ Final

☐ Show Select Overrides Dialog

OK Cancel Help

新的 ClearEditText 文件

3. 编写 ClearEditText 文件的内容

```
package com.example.administrator.mini_weather;

import ...

public class ClearEditText extends EditText implements
    OnFocusChangeListener, TextWatcher {
    /**
     * 删除按钮的引用
     */
    private Drawable mClearDrawable;

    public ClearEditText(Context context) { this(context, null); }

    public ClearEditText(Context context, AttributeSet attrs) {
        //这里构造方法也很重要，不加这个很多属性不能再XML里面定义
        this(context, attrs, android.R.attr.editTextStyle);
    }

    public ClearEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init();
    }

    private void init() {
        //获取EditText的DrawableRight,假如没有设置我们就使用默认的图片
        mClearDrawable = getCompoundDrawables()[2];
        if (mClearDrawable == null) {
            mClearDrawable = getResources()
                .getDrawable(R.drawable.emotionstore_progresscancelbtn);
        }
        mClearDrawable.setBounds(0, 0, mClearDrawable.getIntrinsicWidth(), mClearDrawable.getIntrinsicHeight());
        setClearIconVisible(false);
        setOnFocusChangeListener(this);
        addTextChangedListener(this);
    }
}
```

```

/**
 * 因为我们不能直接给EditText设置点击事件，所以我们用记住我们按下的位置来模拟点击事件
 * 当我们按下的位置 在 EditText的宽度 - 图标到控件右边的间距 - 图标的宽度 和
 * EditText的宽度 - 图标到控件右边的间距之间我们就算点击了图标，竖直方向没有考虑
 */
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (getCompoundDrawables()[2] != null) {
        if (event.getAction() == MotionEvent.ACTION_UP) {
            boolean touchable = event.getX() > (getWidth()
                - getPaddingRight() - mClearDrawable.getIntrinsicWidth())
                && (event.getX() < ((getWidth() - getPaddingRight())));
            if (touchable) {
                this.setText("");
            }
        }
    }

    return super.onTouchEvent(event);
}

/**
 * 当ClearEditText焦点发生变化的时候，判断里面字符串长度设置清除图标的显示与隐藏
 */
@Override
public void onFocusChange(View v, boolean hasFocus) {
    if (hasFocus) {
        setClearIconVisible(getText().length() > 0);
    } else {
        setClearIconVisible(false);
    }
}
}

```

```

/**
 * 设置清除图标的显示与隐藏，调用 setCompoundDrawables 为 EditText 绘制上去
 * @param visible
 */
protected void setClearIconVisible(boolean visible) {
    Drawable right = visible ? mClearDrawable : null;
    setCompoundDrawables(getCompoundDrawables()[0],
        getCompoundDrawables()[1], right, getCompoundDrawables()[3]);
}

```

```

/**
 * 当输入框里面内容发生变化的时候回调的方法
 */
@Override
public void onTextChanged(CharSequence s, int start, int count,
    int after) {
    setClearIconVisible(s.length() > 0);
}

```

```

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
    int after) {
}

```

```

@Override
public void afterTextChanged(Editable s) {
}

```

```

/**
 * 设置晃动动画
 */
public void setShakeAnimation() { this.setAnimation(shakeAnimation(5)); }

```

1. 在 Selecify.java 文件中添加新的方法用来完成搜索。

```
mClearEditText = (ClearEditText) findViewById(R.id. search_city);

//根据输入框输入值的改变来过滤搜索
mClearEditText.addTextChangedListener(new TextWatcher() {

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        //当输入框里面的值为空，更新为原来的列表，否则为过滤数据列表
        filterData(s.toString());
        mList.setAdapter(myadapter);
    }

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count,
                                   int after) {

    }

    @Override
    public void afterTextChanged(Editable s) {

    }

});
}
```

```

/**
 * 根据输入框中的值来过滤数据并更新ListView
 * @param filterStr
 */
private void filterData(String filterStr) {
    filterDateList = new ArrayList<City>();

    Log.d("Filter", filterStr);

    if (TextUtils.isEmpty(filterStr)) {
        for (City city : cityList) {

            filterDateList.add(city);

        }
    } else {
        filterDateList.clear();
        for (City city : cityList) {
            if (city.getCity().indexOf(filterStr.toString()) != -1) {
                filterDateList.add(city);
            }
        }
    }

    // 根据a-z进行排序
    // Collections.sort(filterDateList, pinyinComparator);
    myadapter.updateListView(filterDateList);
}

```

结果展示:

