

课程目录



1

Activity 跳转

实现一个新的 Activity

1. 在 XML 中定义 **Layout**
2. 在 Java class 中定义 **Activity**
 - 继承 AppCompatActivity
3. 将 Activity 和 Layout 连接起来
 - 在 **onCreate()** 方法中设置内容视图
4. 在安卓 manifest 文件中 **声明 Activity**



1.在 XML 中定义 Layout



```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Let's Shop for Food!" />
</RelativeLayout>
```



2.在 Java class 中定义 Activity

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```



3. 将 Activity 和 Layout 连接起来

通过R.资源类型.资源名称进行访问的

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

4. 在 manifest 文件中声明 Activity

```
<activity android:name=".MainActivity">
```

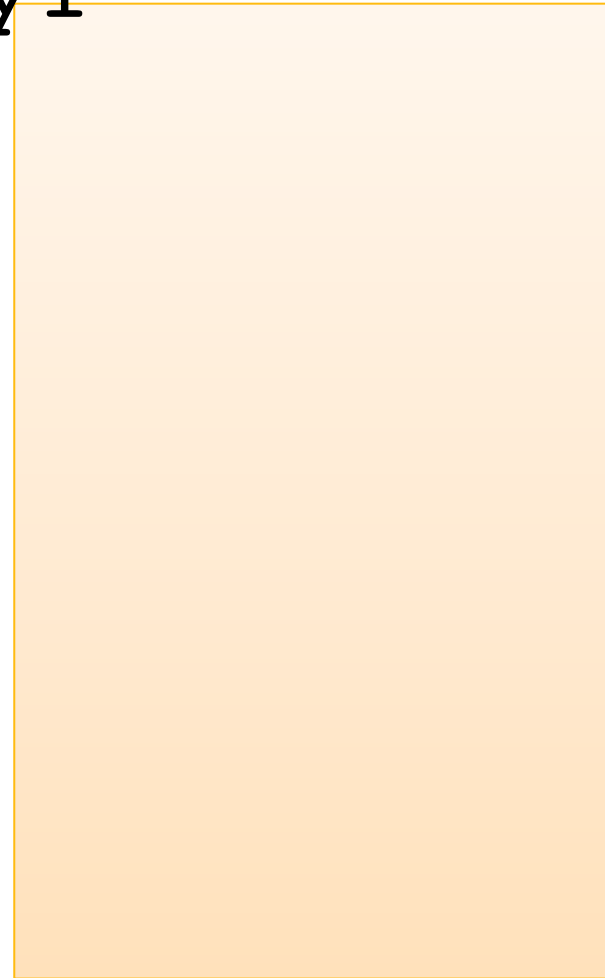


Activity跳转



浙江大学
ZHEJIANG UNIVERSITY

MainActivity1



MainActivity2



按钮添加点击事件

`android:onClick="startActivity2"`

```
// 跳转到MainActivity2
public void startActivity2(View view) {
    startActivity(new Intent(this, MainActivity2.class));
}
```



把数据返回给原先的 Activity



1. 使用 `startActivityForResult()` 来启动接收数据Activity
2. 从第二个 Activity 当中返回数据
 - 创建一个新的 `Intent`
 - 使用 `putExtra()` 方法将回应的数据放在 `Intent` 中
 - 将结果设置为 `Activity.RESULT_OK`
 - 如果用户取消了，设置为 `Activity.RESULT_CANCELED`
 - 调用 `finish()` 来关闭 Activity
3. 在第一个 Activity 中实现 `onActivityResult()`



startActivityForResult()



startActivityForResult(intent, requestCode);

- 根据 **Intent** 启动 Activity，将其标识符设定为 **requestCode**
- 通过 Intent **extras** 来传递数据
- 结束之后回到先前的 Activity，执行 **onActivityResult()** 回调函数 来处理返回的数据
- 使用 requestCode 来标识返回的是哪一个 Activity



1. startActivityForResult()

```
public static final int TEXT_REQUEST = 1;

Intent intent = new Intent(this, TextActivity.class);

startActivityForResult(intent, TEXT_REQUEST);
```

2. 从第二个 Activity 当中返回数据

```
// 创建一个 intent
Intent replyIntent = new Intent();
// 把 data 放入 extra
replyIntent.putExtra(EXTRA_REPLY, reply);
// 把 activity 的 result 设置为 RESULT_OK
setResult(RESULT_OK, replyIntent);
// 结束当前 activity
finish();
```



3.实现 onActivityResult()



```
public void onActivityResult(int requestCode,
                             int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == TEXT_REQUEST) { // Identify activity
        if (resultCode == RESULT_OK) { // Activity succeeded
            String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
            // ... do something with the data
        }
    }
}
```



使用新接口 Activity Result API

```
private ActivityResultLauncher<Intent> activityResultLauncher;  
  
activityResultLauncher = registerForActivityResult(  
    new ActivityResultContracts.StartActivityForResult(),  
    result -> {  
        String reply = result.getData().getStringExtra(SecondActivity.EXTRA_REPLY);  
        // ... do something with the data  
    }  
);
```

Activity2中返回代码无需修改



课程目录



2

OKHttp

OkHttp使用

由Square公司贡献的一个处理网络请求的开源项目,是目前Android使用最广泛的网络框架。从Android4.4开始HttpURLConnection的底层实现采用的是OkHttp。

- 支持HTTP/2并允许对同一主机的所有请求共享一个套接字
- 如果非HTTP/2, 则通过连接池减少了请求延迟
- 默认请求GZip压缩数据
- 响应缓存,避免了重复请求的网络
-



OKHttp使用

1. 在app的build.gradle中添加依赖

```
dependencies {  
.....  
    //OKHttp  
    implementation("com.squareup.okhttp3:okhttp:4.9.0")  
}
```

2. 在AndroidManifest.xml中添加权限

```
<uses-permission android:name="android.permission.INTERNET" />
```



同步请求

```
public void getSync(View view) {  
    new Thread() {  
        @Override  
        public void run() {  
            Request request = new Request.Builder().url(Url).build();  
            Call call = okHttpClient.newCall(request);  
            try {  
                Response response = call.execute();  
                Log.i(TAG, "getSync:" + response.body().string());  
            } catch (IOException e) {  
                throw new RuntimeException(e);  
            }  
        }  
    }.start();  
}
```



OKHttp使用

异步请求

```
public void getAsync(View view) {  
    Request request = new Request.Builder().url(Url).build();  
    Call call = okHttpClient.newCall(request);  
    call.enqueue(new Callback() {  
        @Override  
        public void onFailure(@NonNull Call call, @NonNull IOException e) {  
  
        }  
  
        @Override  
        public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {  
            if (response.isSuccessful()) {  
                Log.i(TAG, "getAsync: " + response.body().string());  
            }  
        }  
    });  
}
```

