



常用技术及工具介绍 (2)

2025/09/23



移动开发

ANDROID



Peking
University



- 1. 开发工具 (IDE)
- 2. 开发语言
- 3. 项目结构
- 4. 其他



Peking
University

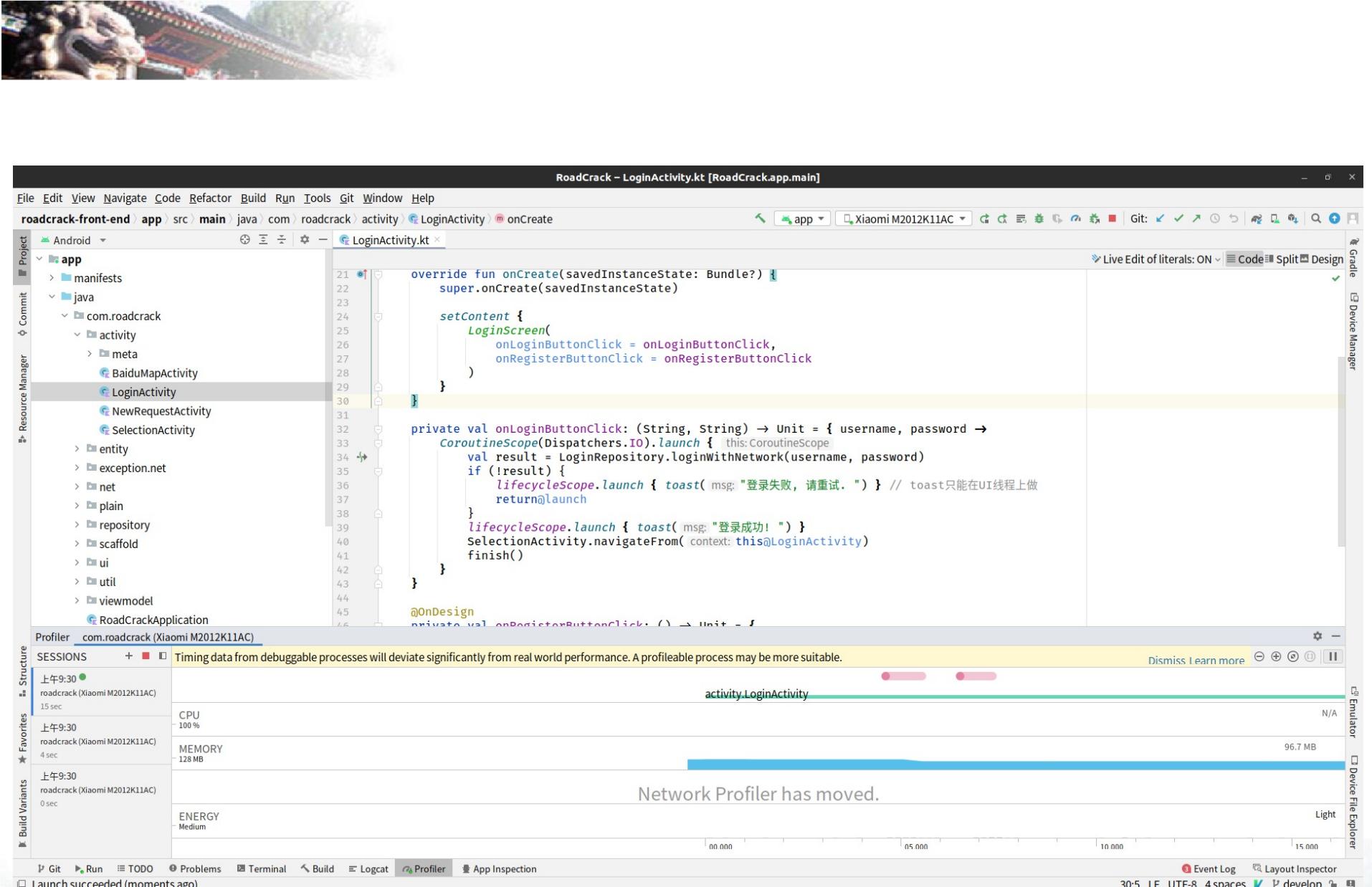


开发工具 Android Studio

- Android Studio是一个为Android平台开发程序的集成开发环境。2013年5月16日在Google I/O上发布，可供开发者免费使用。
- 基于JetBrains IntelliJ IDEA，为Android开发特殊定制，并在Windows、OS X和Linux平台上均可运行
- 当前最新版本为2025.1.3.7，已经较为稳定、成熟
- 下载和使用可以参考其官网：
<https://developer.android.google.cn/studio/>
- 官方学习资源：
<https://developer.android.google.cn/>
- 微信公众号：谷歌开发者



Peking
University



RoadCrack – LoginActivity.kt [RoadCrack.app.main]

File Edit View Navigate Code Refactor Build Run Tools Git Window Help

roadcrack-front-end > app > src > main > java > com > roadcrack > activity > LoginActivity > onCreate

Project

Android

Manifests

Java

com.roadcrack

activity

BaiduMapActivity

LoginActivity

NewRequestActivity

SelectionActivity

entity

exception.net

net

plain

repository

scaffold

ui

util

viewmodel

RoadCrackApplication

Live Edit of literals: ON

Code Split Design

Device Manager

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setContent {
        LoginScreen(
            onLoginButtonClick = onLoginButtonClick,
            onRegisterButtonClick = onRegisterButtonClick
        )
    }
}

private val onLoginButtonClick: (String, String) -> Unit = { username, password ->
    CoroutineScope(Dispatchers.IO).launch { this:CoroutineScope
        val result = LoginRepository.loginWithNetwork(username, password)
        if (!result) {
            lifecycleScope.launch { toast( msg: "登录失败, 请重试. " ) } // toast只能在UI线程上做
            return@launch
        }
        lifecycleScope.launch { toast( msg: "登录成功! " ) }
        SelectionActivity.navigateFrom( context: this@LoginActivity )
        finish()
    }
}

@OnDesign
private val onRegisterButtonClick: () -> Unit = {
```

Profiler com.roadcrack (Xiaomi M2012K11AC)

SESSIONS + Dismiss Learn more

Timing data from debuggable processes will deviate significantly from real world performance. A profileable process may be more suitable.

上午9:30 roadcrack (Xiaomi M2012K11AC) 15 sec

CPU 100 %

MEMORY 128 MB

ENERGY Medium

Favorites

Build Variants

Emulator

Device File Explorer

Git Run TODO Problems Terminal Build Logcat Profiler App Inspection

Launch succeeded (moments ago)

Event Log Layout Inspector

30:5 LF UTF-8 4 spaces develop



Peking
University

Why Kotlin

Modern,
concise and safe
programming language

Easy to pick up, so you can create powerful
applications immediately.

Get started →

Concise Safe Expressive Interoperable Multiplatform

```
data class Employee(  
    val name: String,  
    val email: String,  
    val company: String  
) // + automatically generated equals(), hashCode(), toString(), and copy()  
  
object MyCompany {  
    const val name: String = "MyCompany"  
}  
  
fun main() {  
    val employee = Employee("Alice",  
        "alice@mycompany.com", MyCompany.name)  
    println(employee)  
}
```

// Function at the top level
// No `new` keyword

[Open in Playground →](#)

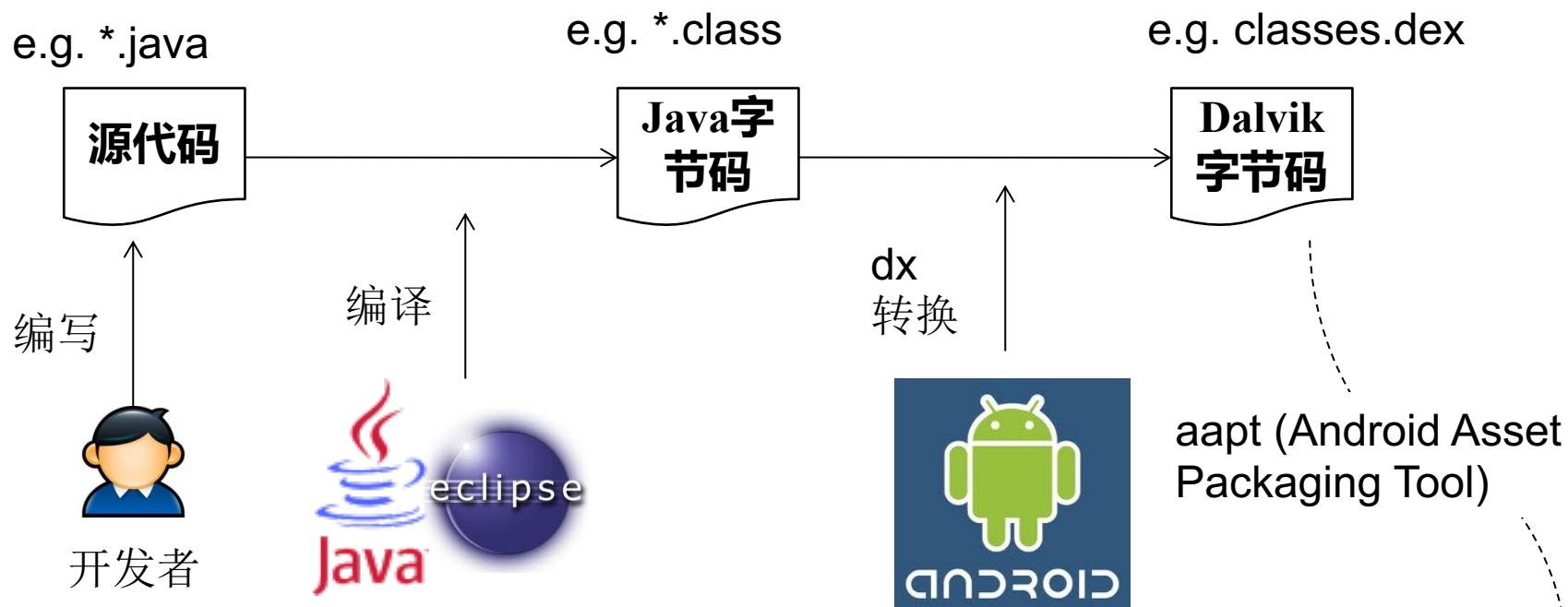
Target: JVM Running on v.1.7.10



Peking
University



开发语言-编译过程



名称	大小	压缩后大小	类型	修改时间
..			Folder	
assets			Folder	2011/6/14 4:02
lib			Folder	2011/6/14 4:08
META-INF			Folder	2011/6/14 4:08
res			Folder	2011/6/14 4:05
AndroidManifest.xml	4,180	1,371	XML Document	2011/6/14 4:08
classes.dex	475,104	206,465	文件 dex	2011/6/14 4:08
resources.arsc	51,108	11,180	文件 arsc	2011/6/14 4:08

Android应用程序组成——APK视图

Android应用(.apk)

AndroidManifest.xml

Activities

Views

Intents

Services

BroadcastReceivers

ContentProviders

代码组成(classes.dex)

主资源文件(resources.arsc)

Assets

lib

META-INF

res



名称	大小	压缩后大小	类型
..	4,180	1,371	Folder XML Docu
assets	475,104	206,465	Folder 文件 dex
lib	51,108	11,180	Folder 文件 arsc
META-INF			
res			
AndroidManifest.xml			
classes.dex			
resources.arsc			



Peking
University



Android应用程序组成——源码视图



案例：开发一个天气预报App: Sunny Weather

口功能需求：

➤ Sunny Weather应该具备的功能有：

- 可以搜索全球大多数国家的各个城市数据；
- 可以查看全球绝大多数城市的天气信息。



Peking
University



□ 开发步骤：

- 注册彩云天气账号，获取天气API访问权限；
- 在Android Studio中创建项目，并在GitHub上托管；
- 设计并确定项目的架构；
- 添加相关依赖和图片等资源；
- 编写逻辑层代码；
- 实现UI层代码；
- 签名、生成APK文件。



Peking
University



案例：开发一个天气预报App: Sunny Weather

注册彩云天气账号，获取天气API访问权限

➤ 注册地址：<https://dashboard.caiyunapp.com/>

```
https://api.caiyunapp.com/v2/place?query=北京&token={token}&lang=zh_CN
```

```
https://api.caiyunapp.com/v2.5/{token}/116.4073963,39.9041999 realtime.json
```

```
{"status": "ok", "query": "北京",  
"places": [  
{"name": "北京市", "location": {"lat": 39.9041999, "lng": 116.4073963},  
"formatted_address": "中国北京市"},  
{"name": "北京西站", "location": {"lat": 39.89491, "lng": 116.322056},  
"formatted_address": "中国 北京市 丰台区 莲花池东路118号"},  
{"name": "北京南站", "location": {"lat": 39.865195, "lng": 116.378545},  
"formatted_address": "中国 北京市 丰台区 永外大街车站路12号"},  
{"name": "北京站(地铁站)", "location": {"lat": 39.904983, "lng": 116.427287},  
"formatted_address": "中国 北京市 东城区 2号线"}]
```

```
{  
    "status": "ok",  
    "result": {  
        "realtime": {  
            "temperature": 23.16,  
            "skycon": "WIND",  
            "air_quality": {  
                "aqi": { "chn": 17.0 }  
            }  
        }  
    }  
}
```

我的申请信息 3. 完成

彩云天气 API 彩云小译 API

SunnyWeather

如果您开发的应用中使用到开放平台的数据接口，且已在iOS/安卓应用市场上架，请提交应用所在商店地址（如果是web应用请提供网址，如果是pc端应用，请提供下载地址）

应用链接：

应用开发情况：

第一行代码中的实战项目开发

如果应用仍处于开发过程中，请在此栏位里说明，暂可不需要提供相关文件，待上线后再提交即可。

提交

 University

案例：开发一个天气预报App: Sunny Weather

在Android Studio中创建项目，并在GitHub上托管

Owner: guolindev / Repository name: SunnyWeather

Great repository names are short and memorable. Need inspiration? How about bookish-octo-pancake?

Description (optional):

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer.

Add .gitignore: Android | Add a license: Apache License 2.0 | ⓘ

Create repository

创建github项目

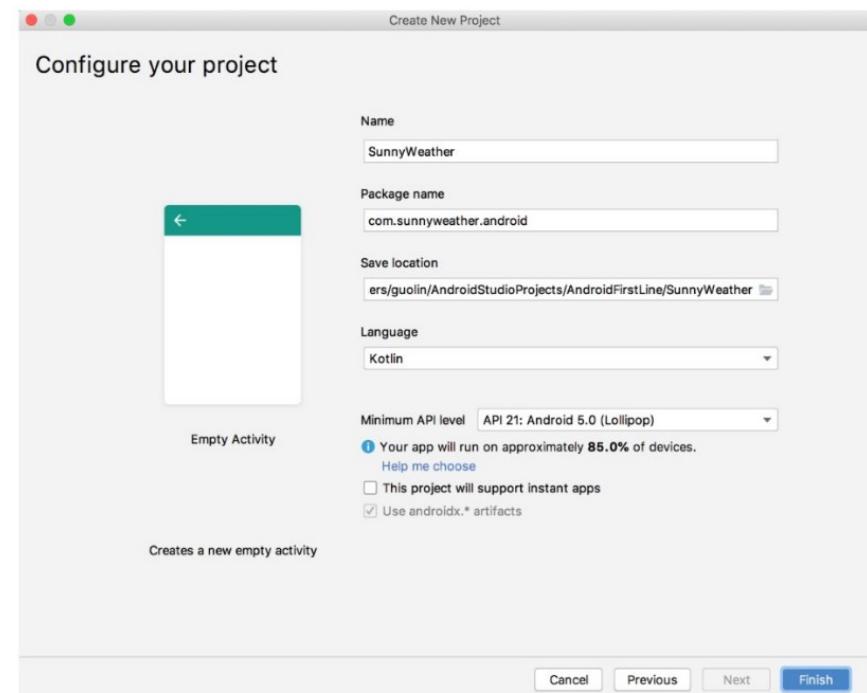
Create new file | Upload files | Find file | Clone or download ▾

Clone with HTTPS ⓘ | Use SSH

Use Git or checkout with SVN using the web URL.
https://github.com/guolindev/SunnyWe... | ⌂

Open in Desktop | Download ZIP

复制项目链接



在Android Studio中创建项目

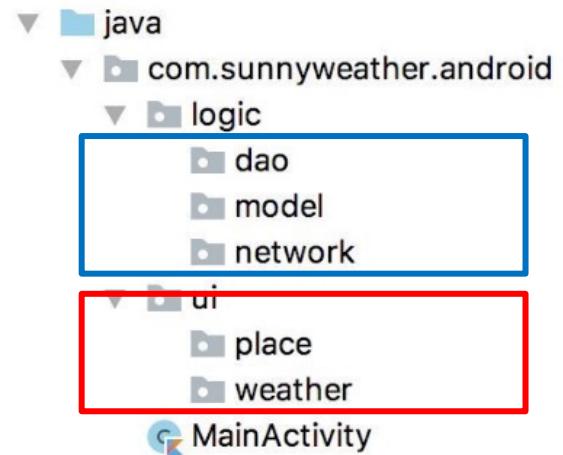
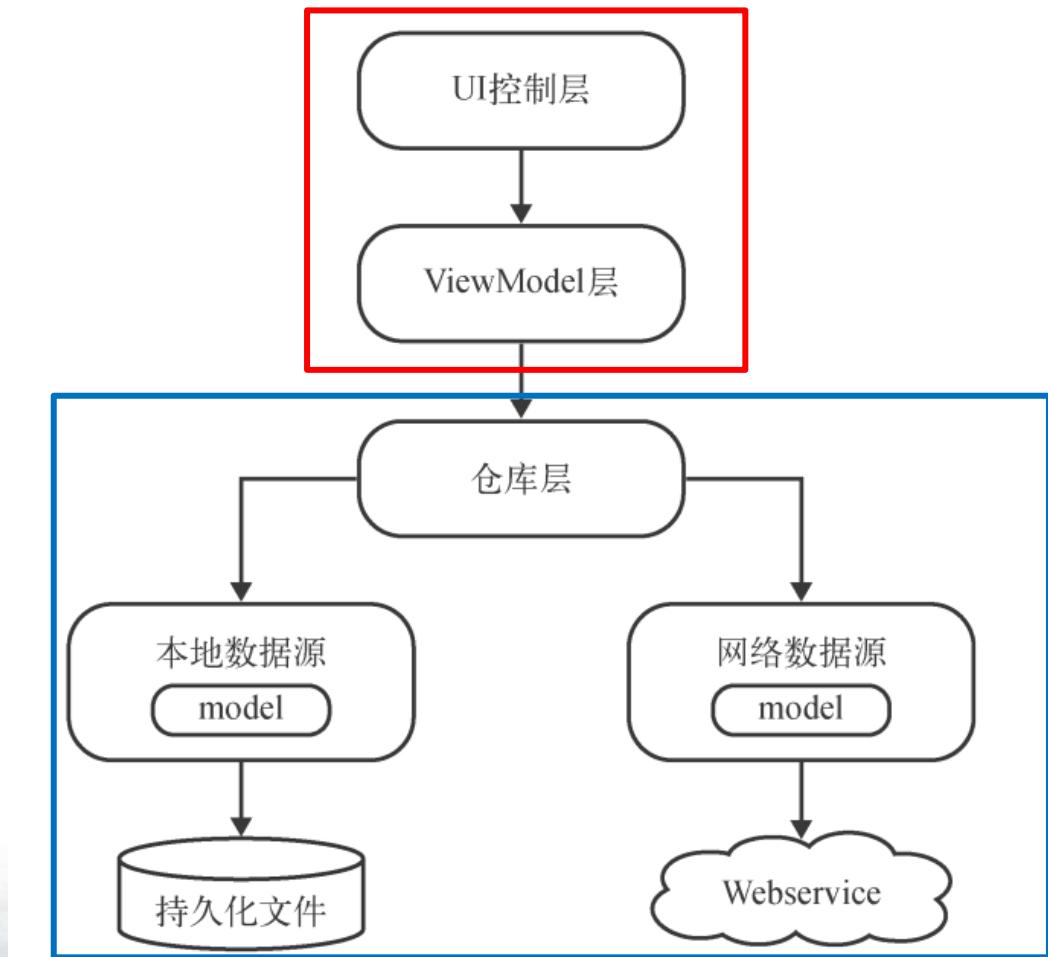
创建项目后执行：

```
git clone https://github.com/guolindev/SunnyWeather.git  
git commit -m "first commit"  
git push origin master (输入GitHub账号和Token)
```



案例：开发一个天气预报App: Sunny Weather

设计并确定项目的架构



Peking
University



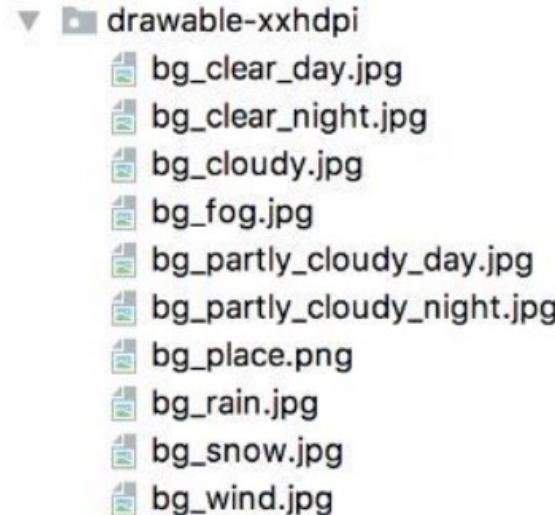
案例：开发一个天气预报App: Sunny Weather

添加相关依赖和图片等资源

```
dependencies {  
    ...  
    implementation 'androidx.recyclerview:recyclerview:1.0.0'  
    implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"  
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.2.0"  
    implementation 'com.google.android.material:material:1.1.0'  
    implementation? "androidx.swiperefreshlayout:swiperefreshlayout:1.0.0"  
    implementation 'com.squareup.retrofit2:retrofit:2.6.1'  
    implementation 'com.squareup.retrofit2:converter-gson:2.6.1'  
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0"  
    implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.1.1"  
}
```

↑ 在 **build.gradle** 中添加依赖项

在**res**中添加图片资源 →



案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

➤ 创建Application

- 在com.sunnyweather.android包下创建名为SunnyWeatherApplication的类（继承 android.app.Application），并在 AndroidManifest.xml 中指定

```
1 package com.sunnyweather.android
2
3 import android.annotation.SuppressLint
4 import android.app.Application
5 import android.content.Context
6
7 class SunnyWeatherApplication : Application() {
8
9     companion object {
10
11         const val TOKEN = "" // 填入你申请到的令牌值
12
13         @SuppressLint("StaticFieldLeak")
14         lateinit var context: Context
15     }
16
17     override fun onCreate() {
18         super.onCreate()
19         context = applicationContext
20     }
21 }
22 }
```

在 SunnyApplication 中添加 context 和 TOKEN 字段，以便APP整个生命周期均可引用

在 AndroidManifest.xml 中指定 Application
(第7行↓)

```
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   <uses-permission android:name="android.permission.INTERNET" />
4
5   <application
6       android:name=".SunnyWeatherApplication"
7       android:allowBackup="true"
8       android:icon="@mipmap/ic_launcher"
9       android:label="@string/app_name"
10      android:roundIcon="@mipmap/ic_launcher_round"
11      android:supportsRtl="true"
12      android:theme="@style/AppTheme">
13
14         <activity android:name=".ui.weather.WeatherActivity"></activity>
15         <activity android:name=".MainActivity">
16             <intent-filter>
17                 <action android:name="android.intent.action.MAIN" />
18
19                 <category android:name="android.intent.category.LAUNCHER" />
20             </intent-filter>
21         </activity>
22     </application>
23
24 
```



案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

➤ 定义API访问接口：使用Retrofit包

- 定义一个访问API的接口 (`logic.network.PlaceService`)
- 定义若干单例对象，用于创建服务和访问接口

```
1 package com.sunnyweather.android.logic.network
2
3     import com.sunnyweather.android.SunnyWeatherApplication
4     import com.sunnyweather.android.logic.model.PlaceResponse
5     import retrofit2.Call
6     import retrofit2.http.GET
7     import retrofit2.http.Query
8
9     interface PlaceService {
10
11         @GET("v2/place?token=${SunnyWeatherApplication.TOKEN}&lang=zh_CN")
12         fun searchPlaces(@Query("query") query: String): Call<PlaceResponse>
13
14     }
15
16     data class PlaceResponse(val status: String, val places: List<Place>)
```

使用Retrofit与API交互
整个接口定义了访问网页的服务；
圈出的函数定义了交互的方法：
调用方法时，访问注解中的API

```
object ServiceCreator {
    private const val BASE_URL = "https://api.caiyunapp.com/"

    private val retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    fun <T> create(serviceClass: Class<T>): T = retrofit.create(serviceClass)

    inline fun <reified T> create(): T = create(T::class.java)
}
```

创建 retrofit-service 的工厂

```
10     object SunnyWeatherNetwork {
11
12         private val weatherService = ServiceCreator.create(WeatherService::class.java)
13
14         suspend fun getDailyWeather(lng: String, lat: String) = weatherService.getDailyWeather(lng, lat).await()
15
16         suspend fun getRealtimeWeather(lng: String, lat: String) = weatherService.getRealtimeWeather(lng, lat).await()
17
18         private val placeService = ServiceCreator.create(PlaceService::class.java)
19
20         suspend fun searchPlaces(query: String) = placeService.searchPlaces(query).await()
21
22     }
23
24     private suspend fun <T> Call<T>.await(): T { ... }
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

使用工厂创建service，包装成函数供上层调用

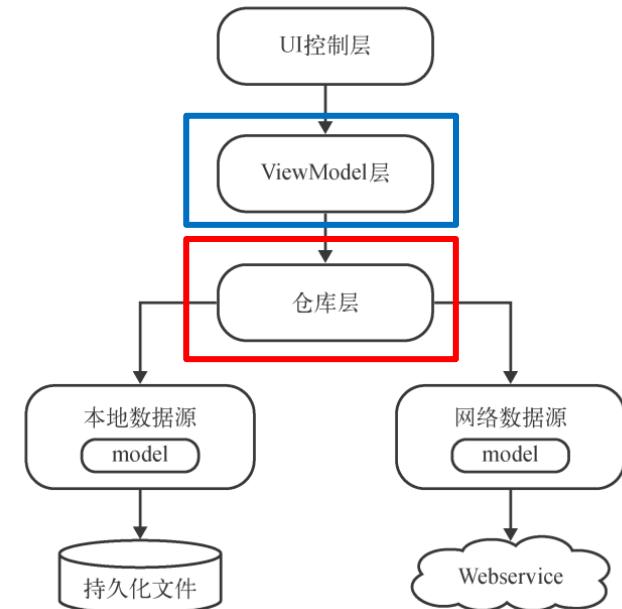
案例：开发一个天气预报App: Sunny Weather

编写逻辑层代码

- 定义Repository：用于封装统一入口
- 在 ViewModel 层使用接口 (ui.place.PlaceViewModel)

```
13  <object Repository {  
14  
15      fun searchPlaces(query: String) = fire(Dispatchers.IO) {  
16          val placeResponse = SunnyWeatherNetwork.searchPlaces(query)  
17          if (placeResponse.status == "ok") {  
18              val places = placeResponse.places  
19              Result.success(places)  
20          } else {  
21              Result.failure(RuntimeException("response status is ${placeResponse.status}"))  
22          }  
23      }  
24  }  
25  ...  
26 }  
27 }
```

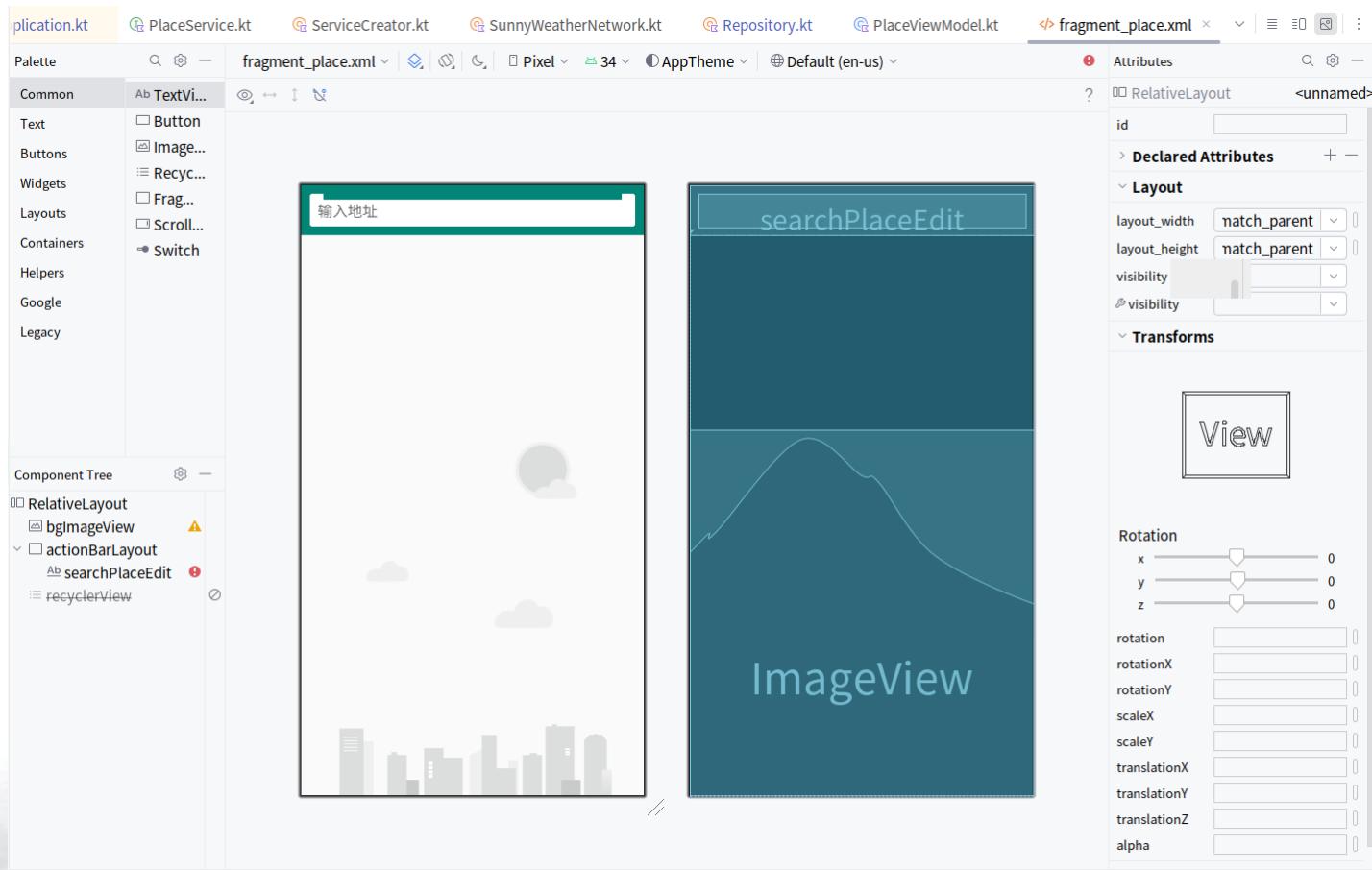
```
7  class PlaceViewModel : ViewModel() {  
8  
9      private val searchLiveData = MutableLiveData<String>()  
10  
11     val placeList = ArrayList<Place>()  
12  
13     val placeLiveData = Transformations.switchMap(searchLiveData) { query →  
14         Repository.searchPlaces(query)  
15     }  
16  
17     fun searchPlaces(query: String) {  
18         searchLiveData.value = query  
19     }  
20  
21     fun savePlace(place: Place) = Repository.savePlace(place)  
22  
23     fun getSavedPlace() = Repository.getSavedPlace()  
24  
25     fun isPlaceSaved() = Repository.isPlaceSaved()  
26 }  
27 }
```



案例：开发一个天气预报App: Sunny Weather

实现UI层代码

- 定义Fragment，并添加到Activity中
- 细节工作：和ViewModel的绑定、Fragment的Java/Kotlin实现等





案例：开发一个天气预报App: Sunny Weather

实现UI层代码

- 定义Fragment，并添加到Activity中
- 细节工作：和ViewModel的绑定、Fragment的Java/Kotlin实现等

```
1 package com.sunnyweather.android.ui.place
2
3 > import ...
18
19 <> <class PlaceFragment : Fragment() {
20
21     val viewModel by lazy { ViewModelProviders.of(fragment: this).get(PlaceViewModel::class.java) }
22
23     private lateinit var adapter: PlaceAdapter
24
25 @+< override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {
26         return inflater.inflate(R.layout.fragment_place, container, attachToRoot: false)
27     }
28
29 @+> override fun onActivityCreated(savedInstanceState: Bundle?) [...]
30
31 }
32 }
```

```
<> activity_weather.xml <
1 <?xml version="1.0" encoding="utf-8"?>
2 <@+> <androidx.drawerlayout.widget.DrawerLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/drawerLayout"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.swiperefreshlayout.widget.SwipeRefreshLayout ... >
9
10    <FrameLayout
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"
13        android:layout_gravity="start"
14        android:clickable="true"
15        android:focusable="true"
16        android:background="@color/colorPrimary">
17
18        <fragment
19            android:id="@+id/placeFragment"
20            android:name="com.sunnyweather.android.ui.place.PlaceFragment"
21            android:layout_width="match_parent"
22            android:layout_height="match_parent"
23            android:layout_marginTop="25dp"/>
24
25    </FrameLayout>
26
27 </androidx.drawerlayout.widget.DrawerLayout>
```

↑ 和界面关联的 Fragment
(Fragment 是比 Activity 更细粒度管理界面的类)

← 在 activity 中添加 Fragment

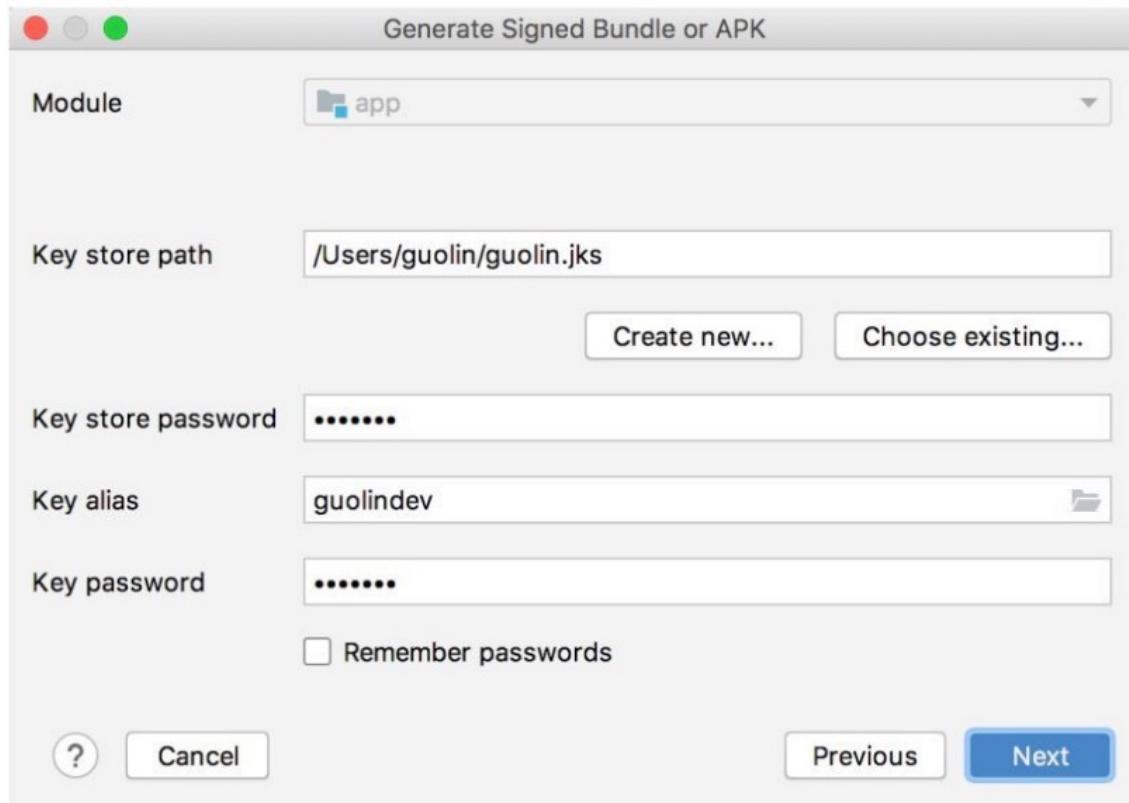


Peking
University

案例：开发一个天气预报App: Sunny Weather

签名、生成APK文件

➤ 使用Android Studio生成：Build→Generate Signed Bundle/APK

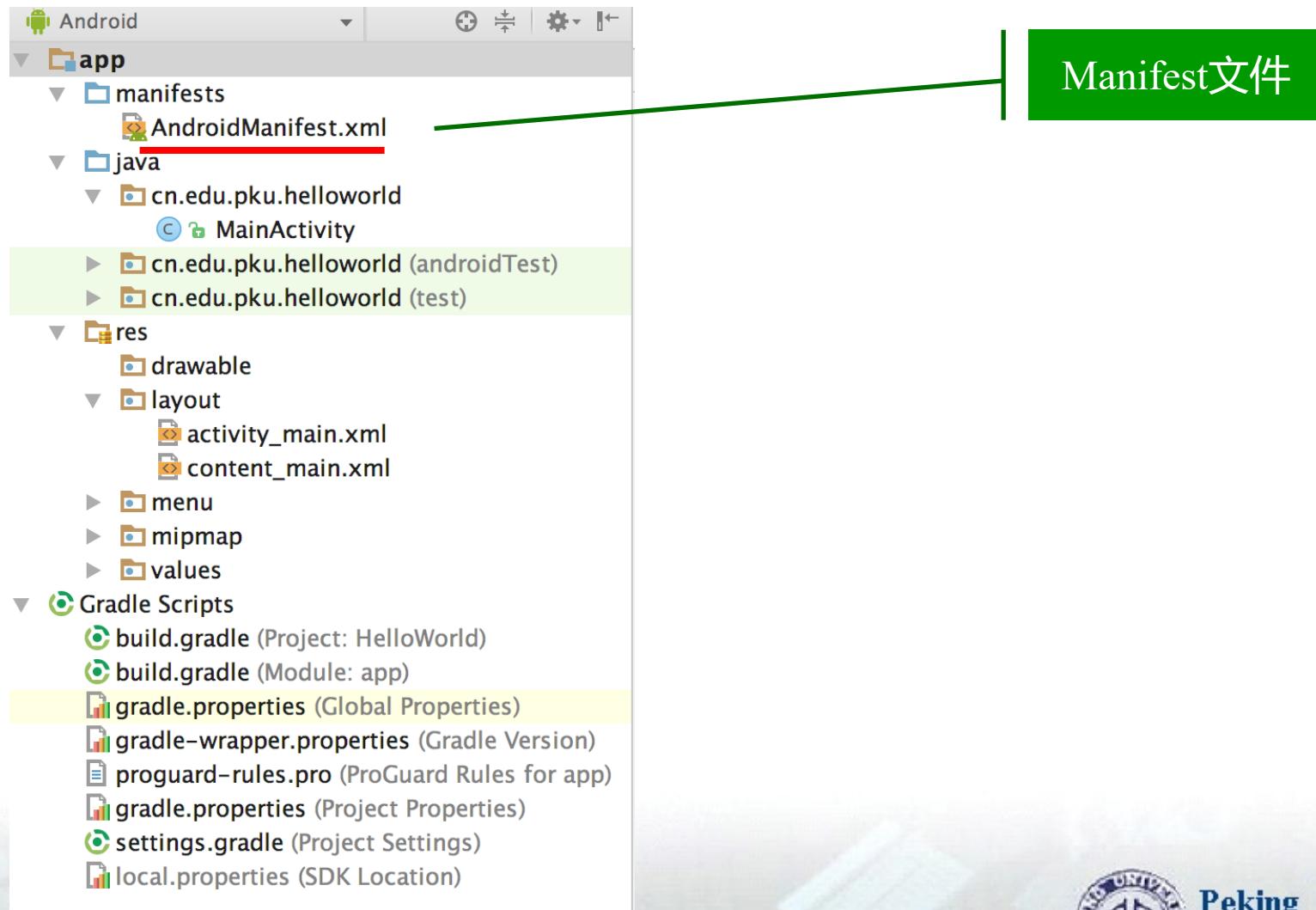


ⓘ Generate Signed APK
APK(s) generated successfully for module 'app'
with 1 build variant:
Build variant 'release': [locate or analyze](#) the APK.

名称	修改日期	大小	种类
app-release.apk	下午1:12	4.9 MB	文稿
output.json	下午1:12	234 字节	纯文本文稿



Android应用程序组成——源码视图



Peking
University

□ Android应用的部署描述符文件

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.edu.pku.helloworld">
    <uses-permission android:name="android.permission.INTERNET" />

    <application>
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="HelloWorld"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
            <activity>
                <uandroid:name=".MainActivity"
                    android:label="HelloWorld"
                    android:theme="@style/AppTheme.NoActionBar">
                    <intent-filter>
                        <action android:name="android.intent.action.MAIN" />
                        <category android:name="android.intent.category.LAUNCHER" />
                    </intent-filter>
                </activity>
            </application>
    </manifest>
```

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

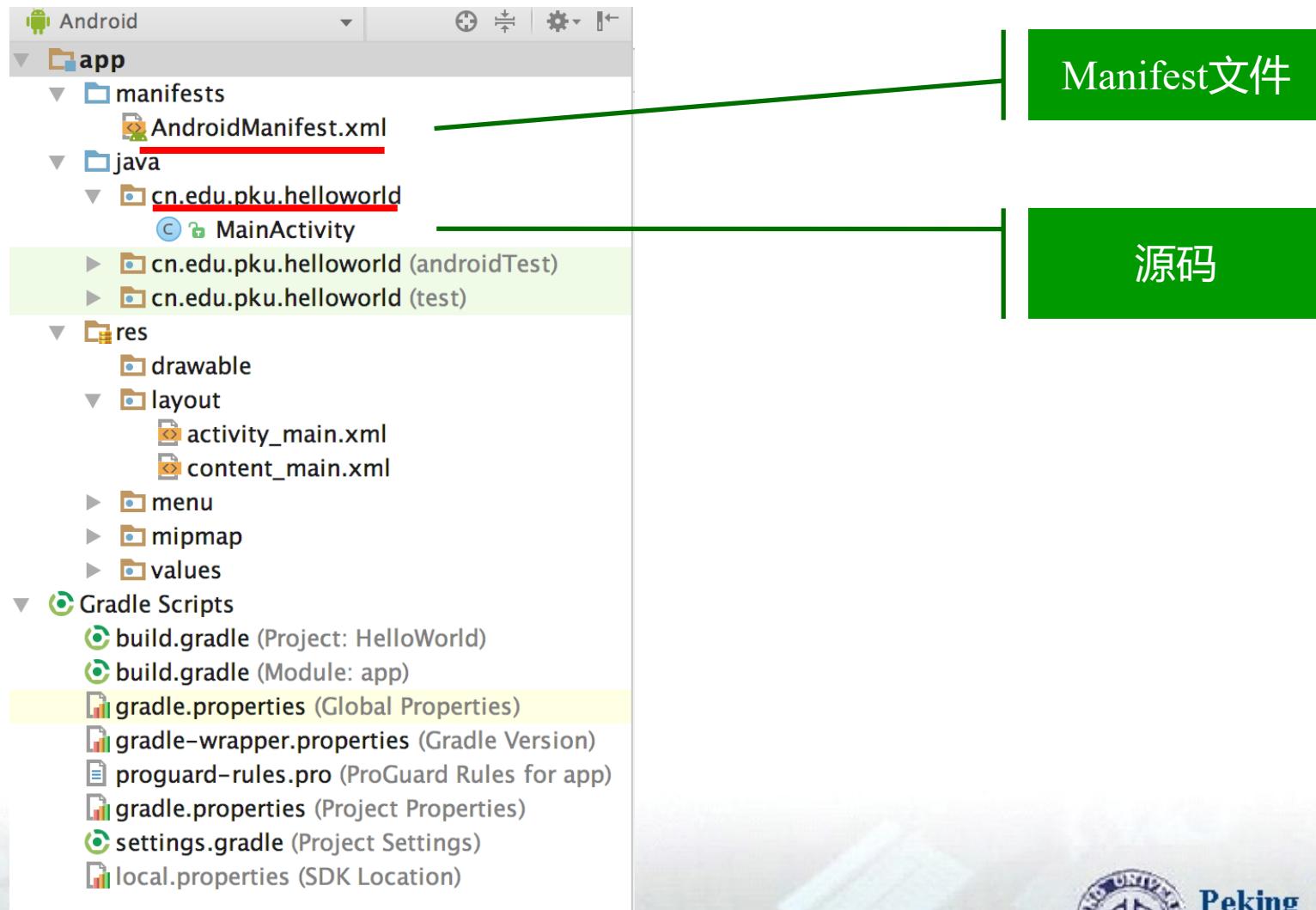
□权限举例 (参考`android.manifest.permission`)

权限名称	权限描述
接收短信	android.permission.RECEIVE_SMS
拨打电话	android.permission.CALL_PHONE
系统启动完毕通知	android.permission.RECEIVE_BOOT_COMPLETED
读取联系人信息	android.permission.READ_CONTACTS
修改联系人信息	android.permission.WRITE_CONTACTS

注意：Android 6.0后权限管理的变化(`targetSdkVersion >= 23`)



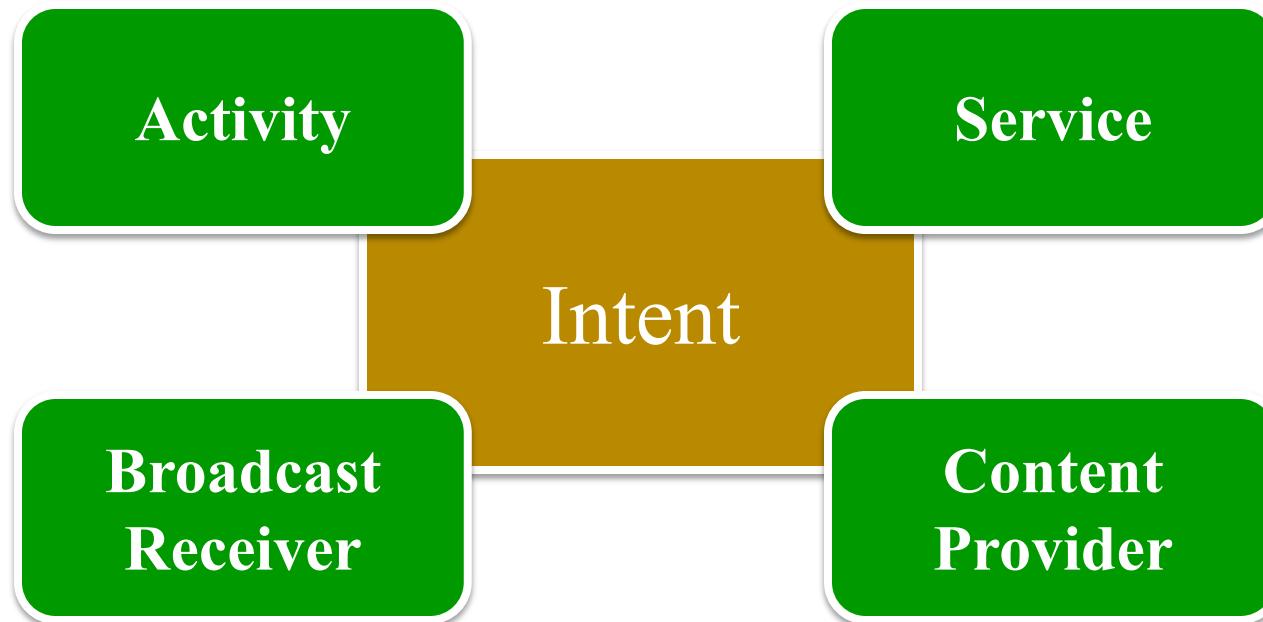
Android应用程序组成——源码视图



Peking
University

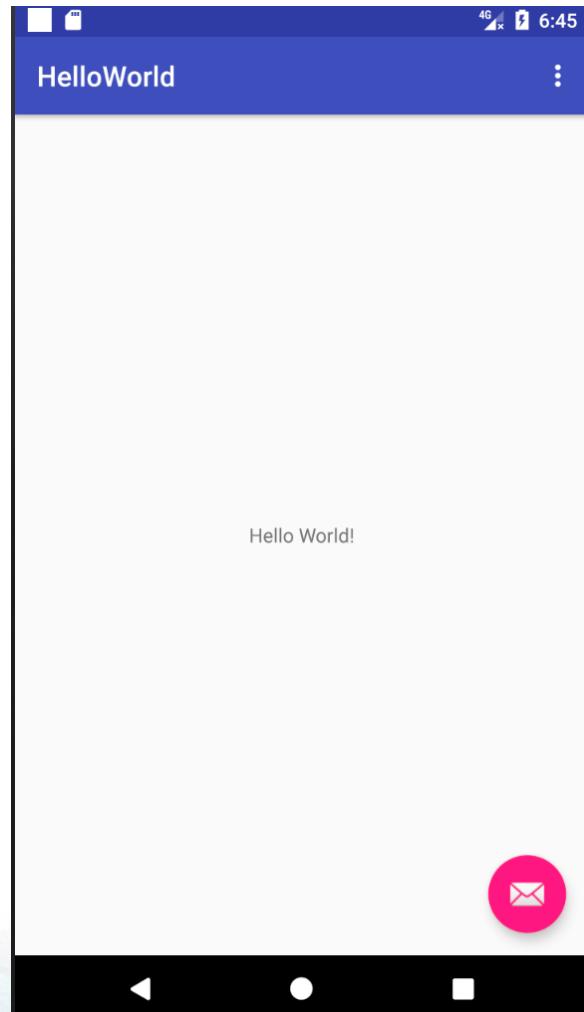


组成Android应用程序的基本构件



Activity(活动)

- 最基本的应用程序构件
- 应用程序中，一个Activity通常可理解为一个单独的屏幕
- Activity 主要作用是对用户在屏幕上的UI界面的输入进行处理并做出响应（更新UI、启动其他Activity等）
- 继承android.app.Activity 类，并覆盖相应的方法
 - onCreate
 - ...





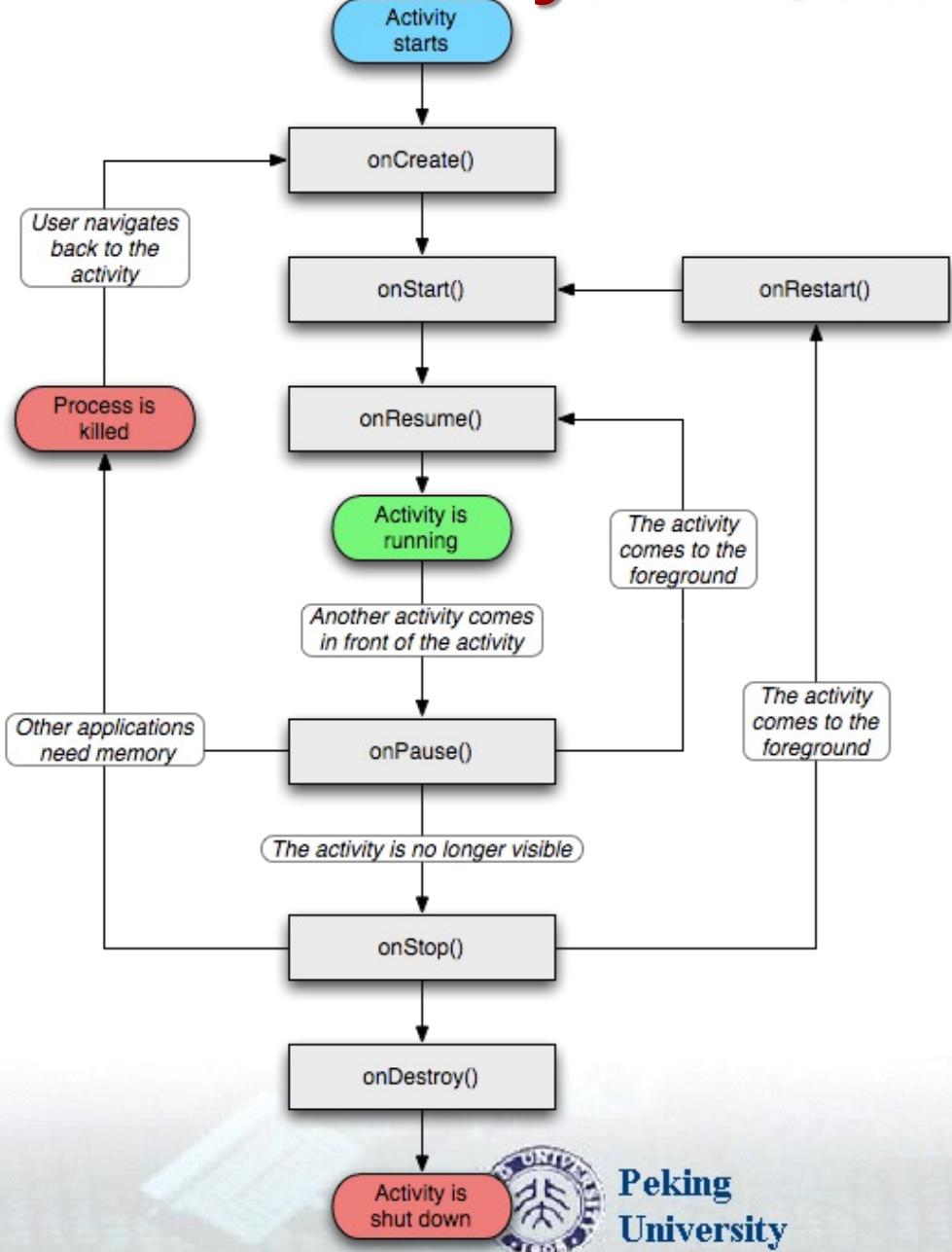
Activity的代码结构与配置文档

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
        setSupportActionBar(toolbar);  
  
        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
        fab.setOnClickListener(view -> {  
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
                .setAction("Action", null).show();  
        });  
    }  
  
<activity  
    android:name=".MainActivity"  
    android:label="HelloWorld"  
    android:theme="@style/AppTheme.NoActionBar">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```



Activity的生命周期

- onCreate**
- onStart**
- onResume**
- onPause**
- onStop**
- onDestroy**



□ TextView extends View

- `setText(String text)`
- `getText()`

□ EditText extends View

- `getText()`

□ ImageView extends View

□ Button extends View

□ View

- `setOnClickListener(View.OnClickListener listener)`
- `setVisibility(int visibility)`



Service (服务)

□什么样的逻辑程序需要实现为Service?

- 执行长时间的计算任务
- 且不与用户直接进行界面交互

□Service是做“地下工作的”



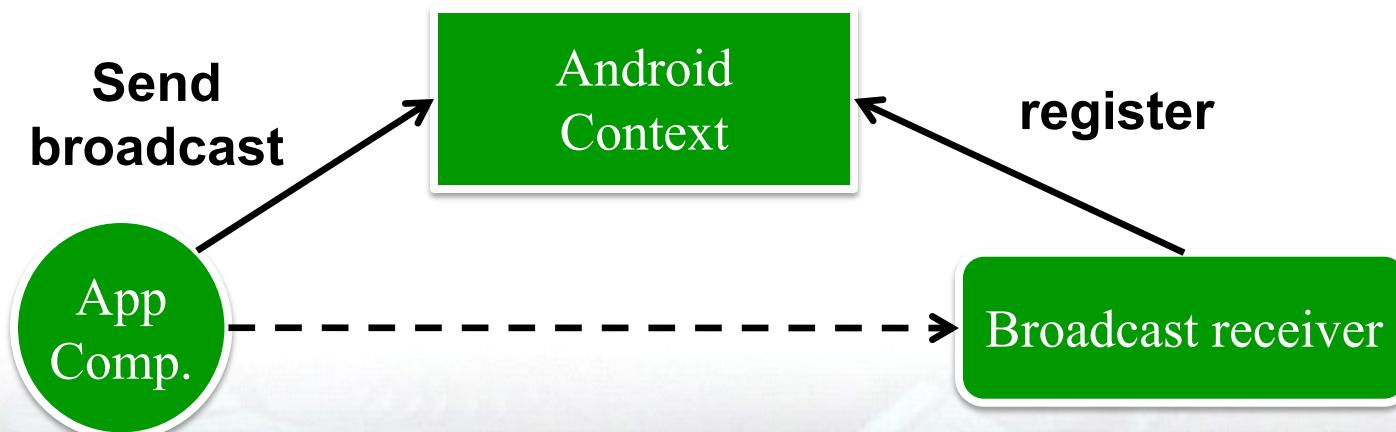


Broadcast Receiver (广播接收器)

□ 接收和处理Android的广播消息

□ Android的广播机制

- Receiver (Observer)向系统注册
- 构件广播Intent
 - Activity/Service/Broadcast Receiver/ContentProvider
- 系统触发相应的Receiver来使其接收Intent

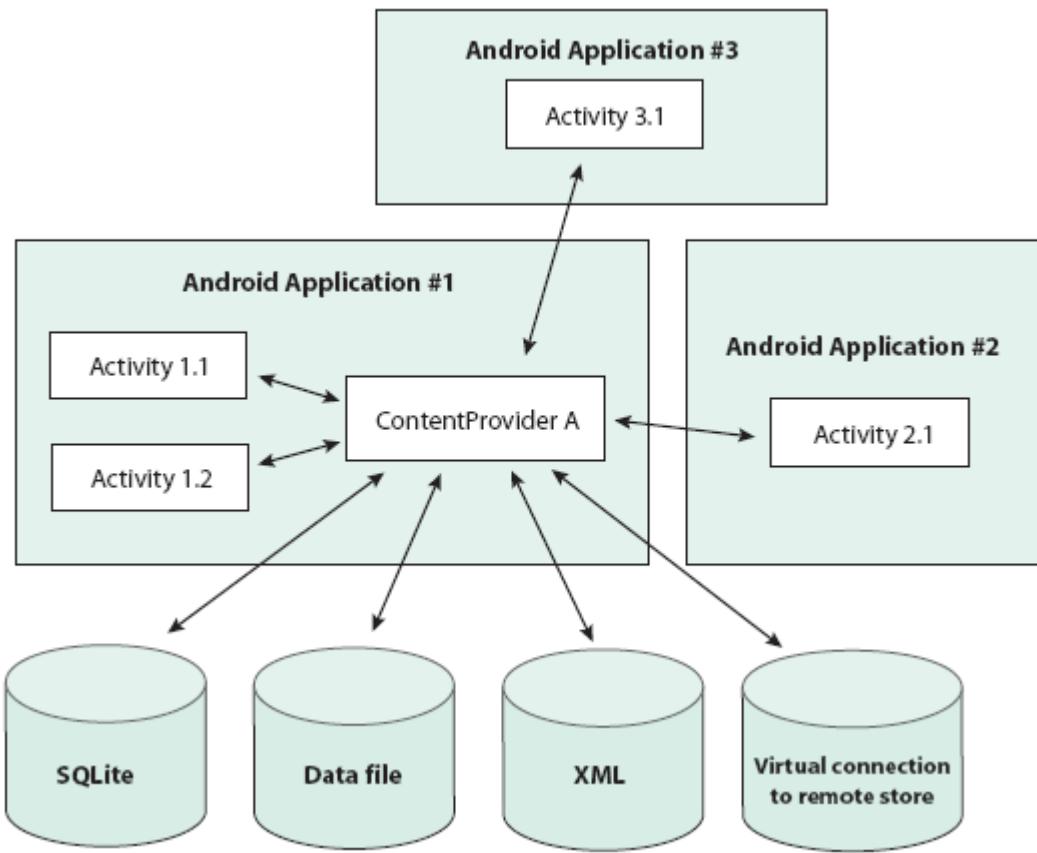




Content Provider (内容供应器)

- Android中的跨应用的数据访问机制
- 为何需要content provider?

- Android中每一个app的资源是私有的
- app通过content provider实现和其他apps的私有数据共享
 - Preference
 - Contact
 - ...

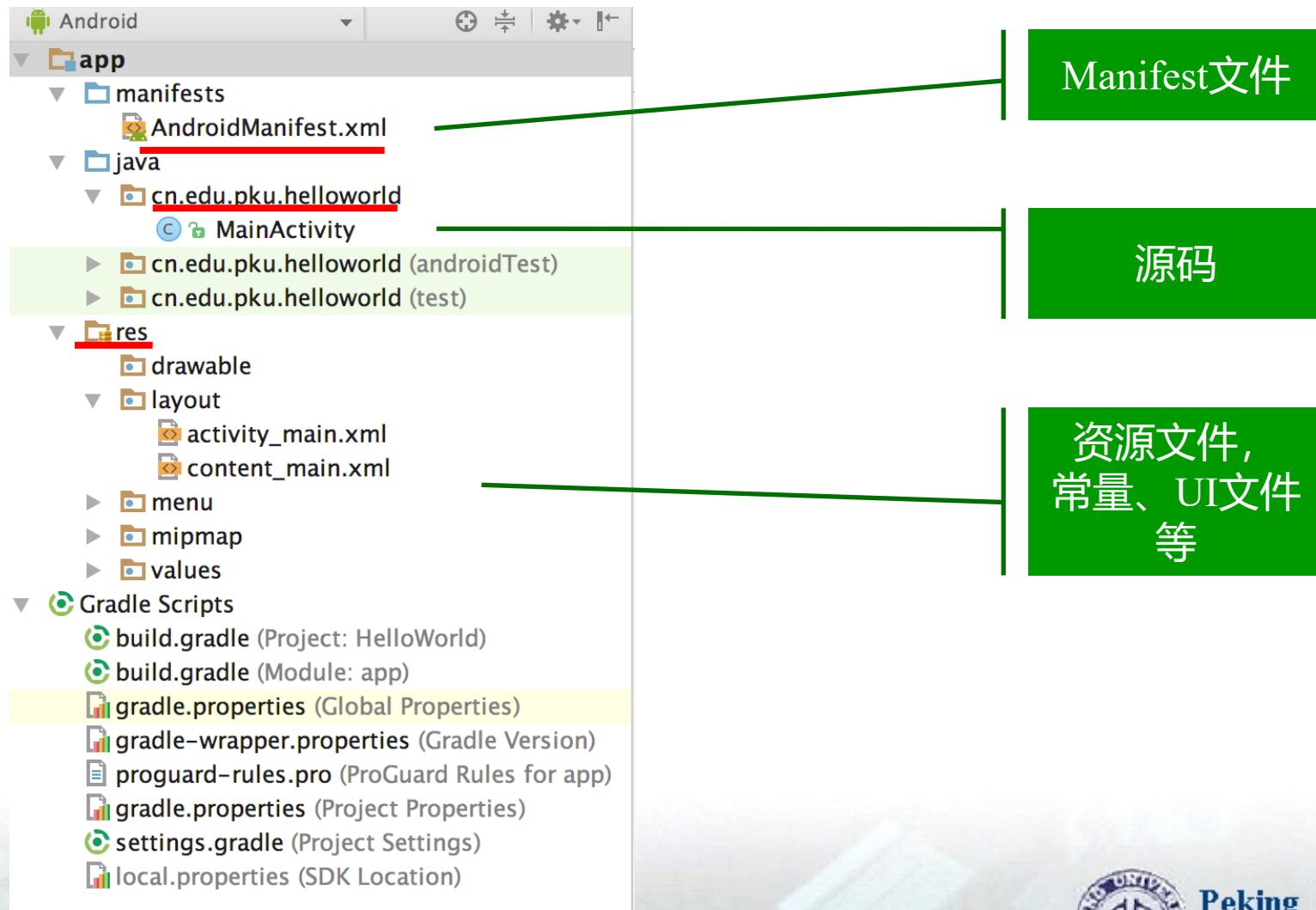


- 可理解为带请求协助意味的消息或事件
- 关联Activity、Service、BroadcastReceiver、ContentProvider的桥梁

```
public class LoginActivity extends AppCompatActivity {  
  
    private int mUserId;  
  
    @Override  
    protected void onCreate(@Nullable Bundle savedInstanceState) {...}  
  
    private void gotoMain() {  
        Intent intent = new Intent(LoginActivity.this, MainActivity.class);  
        intent.putExtra("id", mUserId);  
        startActivity(intent);  
    }  
}
```



Android应用程序组成——源码视图



□ 资源文件

□ 一般有 anim、assets、drawable、layout、menu、mipmap、values（包括 colors、dimens、strings、styles）

□ 通过自动生成的 R 类访问

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
    setSupportActionBar(toolbar);  
  
    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
    fab.setOnClickListener((view) -> {  
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
            .setAction("Action", null).show();  
    });  
}
```

□ 或者在别的 xml 文件中直接引用

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="@color/blue_1">  
  
    <LinearLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:orientation="vertical"  
        android:padding="16dp">
```

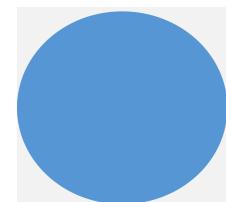
```
<color name="blue_6">#2C82C9</color>  
<color name="blue_5">#7BBBD6</color>  
<color name="blue_4">#A4D4E8</color>  
<color name="blue_3">#C3EAF1</color>  
<color name="blue_2">#E1F2F9</color>  
<color name="blue_1">#F8F9FB</color>
```

colors.xml



□ 使用 xml 的方式绘图

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="@color/google_blue" />
</shape>
```



```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M10,20v-6h4v6h5v-8h3L12,3 2,12
    </vector>
```



Layout

The screenshot shows the Android Studio interface with the project 'HelloWorld' open. The left sidebar displays the project structure, including the app module with its manifest, Java files (MainActivity), and various resource folders like res/layout and res/values. The main workspace shows the XML code for 'activity_main.xml' and its corresponding preview.

activity_main.xml (Text View)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="cn.edu.pku.helloworld.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

        <include layout="@layout/content_main" />

        <android.support.design.widget.FloatingActionButton
            android:id="@+id/fab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|end"
            android:layout_margin="16dp"
            app:srcCompat="@android:drawable/ic_dialog_email" />

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

Preview (Design View)

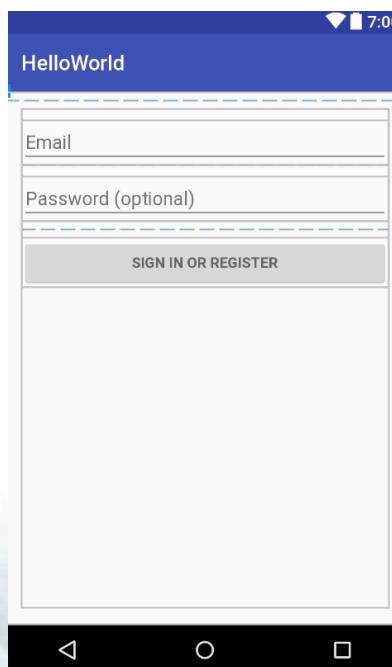
The preview window shows a smartphone screen with a blue header bar labeled '7:00'. Below it is a white content area with the text 'Hello World!'. At the bottom is a black navigation bar with three icons: a triangle (left), a circle (center), and a square (right). A small red circular icon with a white envelope symbol is positioned in the bottom right corner of the content area.

Bottom Navigation Bar

At the very bottom of the screen, there is a navigation bar with several icons: Messages, Terminal, Android Monitor, Problems, Run, and TODO.

□ 设置元素属性

- Application
- Activity
- Views

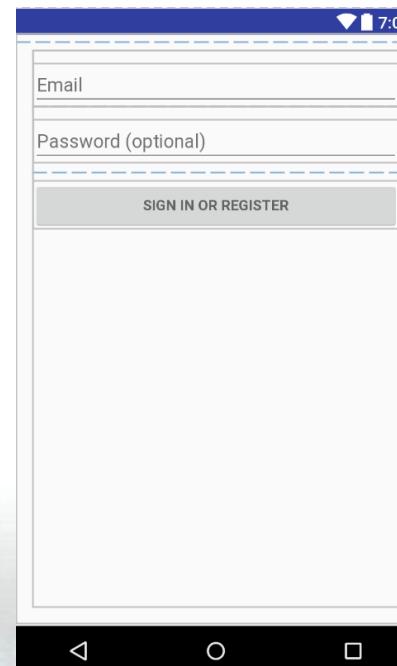


```
<activity  
    android:name=".activity.TagInfoActivity"  
    android:theme="@style/AppTheme.NoActionBar" />
```

Manifest.xml

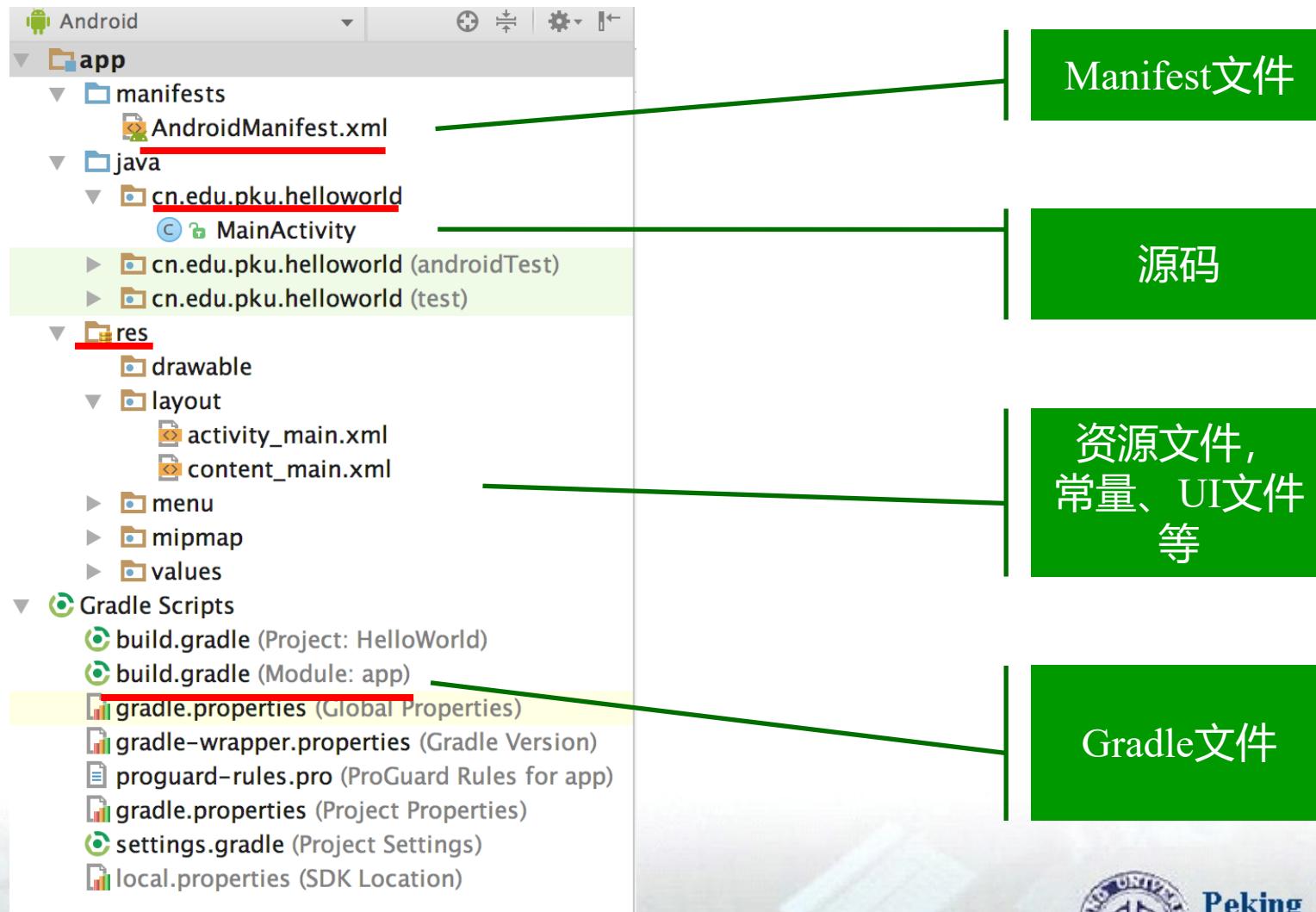
```
<style name="AppTheme.NoActionBar">  
    <item name="windowActionBar">false</item>  
    <item name="windowNoTitle">true</item>  
</style>
```

styles.xml





Android应用程序组成——源码视图



- 对你的项目进行编译、运行、签名、打包、依赖管理等一系列功能的合集
- 如设定 编译 sdk 版本、设置运行时 sdk 最低/最高的版本号、设置应用版本号、设置签名文件、管理依赖关系等
- 一般首选 Gradle 导入第三方依赖库，也支持 Maven

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
        exclude group: 'com.android.support', module: 'support-annotations'  
    })  
    compile 'com.android.support:appcompat-v7:25.3.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.1'  
    compile 'com.android.support:design:25.3.1'  
    testCompile 'junit:junit:4.12'  
}
```

- 网络请求: retrofit、volley、okhttp
- 序列化: Gson、Moshi、Jackson
- 图片载入: picasso、fresco、Glide
- 依赖注入: butterknife、dagger、hilt
- 事件总线: EventBus
- 响应式编程: LiveData, RxJava, RxAndroid,
Koroutine
- Socket: SocketIO
- <https://github.com/wasabeef/awesome-android-libraries>

□ 推送

- 小米、极光、友盟、腾讯信鸽、LeanCloud、GCM等

□ IM

- 环信、极光、融云、LeanCloud、网易等

□ 地图

- 百度、高德、腾讯等

□ 异常上报、应用统计

- 腾讯Bugly、友盟、LeanCloud等

□ 对象存储（图片等）

- 阿里云OSS、七牛云等

□



□ Google开发的Java序列化与反序列化工具。即将Json字符串转换为Java对象，或将Java对象转换为Json字符串。

String To Java Object

Java Object To String

```
class User {  
    private String id;  
    private String name;  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
}  
  
String json = "{ \"id\": \"user-id\", \"name\": \"root\" }";  
User user = new Gson().fromJson(json, User.class);  
  
User user = new User();  
String json = new Gson().toJson(user);
```



Peking
University

□ 配置

```
public static Retrofit getRetrofit() {  
    if (retrofit == null)  
        retrofit = new Retrofit.Builder()  
            // 设置域名  
            .baseUrl(mBaseUrl)  
            // 配合json使用  
            .addConverterFactory(GsonConverterFactory.create())  
            .build();  
    return retrofit;  
}
```

□ 定义接口

```
public interface GitHubService {  
    @GET("users/{user}/repos")  
    Call<List<Repo>> listRepos(@Path("user") String user);  
}
```

□ 直接调用

```
GitHubService service = getRetrofit().create(GitHubService.class);  
  
Call<List<Repo>> call = service.listRepos("username", 0);  
call.enqueue(new Callback<List<Repo>>() {  
    @Override  
    public void onResponse(Call<List<Repo>> call, Response<List<Repo>> response) {  
        List<Repo> repoList = response.body();  
        // do something  
    }  
    @Override  
    public void onFailure(Call<List<Repo>> call, Throwable t) {  
        t.printStackTrace(); // 错误处理  
    }  
});
```



Retrofit2+Koroutine

```
1 package com.roadcrack.net.meta
2
3 import ...
4
5 object MyRetrofit { ... }
6
7
8     package com.roadcrack.net
9
10    import ...
11
12    interface RoadService {
13
14        @POST("/road/id")
15        suspend fun fetchRoadId(
16            @Query("districtCode") districtCode: Long,
17            @Query("roadName") roadName: String,
18            @Body /*("districtInfo")*/ districtInfo: RequestBody
19        ): WebResult<Long>
20
21
22        companion object {
23            operator fun invoke() = MyRetrofit<RoadService>()
24        }
25    }
26
27
28    suspend fun retrieveRoadId(loc: BDLocation, roadName: String): Long {
29        val districtCode: Long = loc.adCode.toLong()
30        val formBody = FormBody.create(
31            okhttp3.MediaType.parse("application/json; charset=utf-8"),
32            // 目前只需要这3个信息
33            mapOf( ... ).toJson()
34        )
35
36        return roadSrv.fetchRoadId(districtCode, roadName, formBody).validIt()
37            .also { BriefInfo.roadId.value = it }
38    }
39
40 }
```



□ A powerful image downloading and caching library for Android

```
Picasso.with(getApplicationContext())
    .load(url)
    .placeholder(R.drawable.error_image)
    .error(R.drawable.error_image)
    .into(imageView);
```

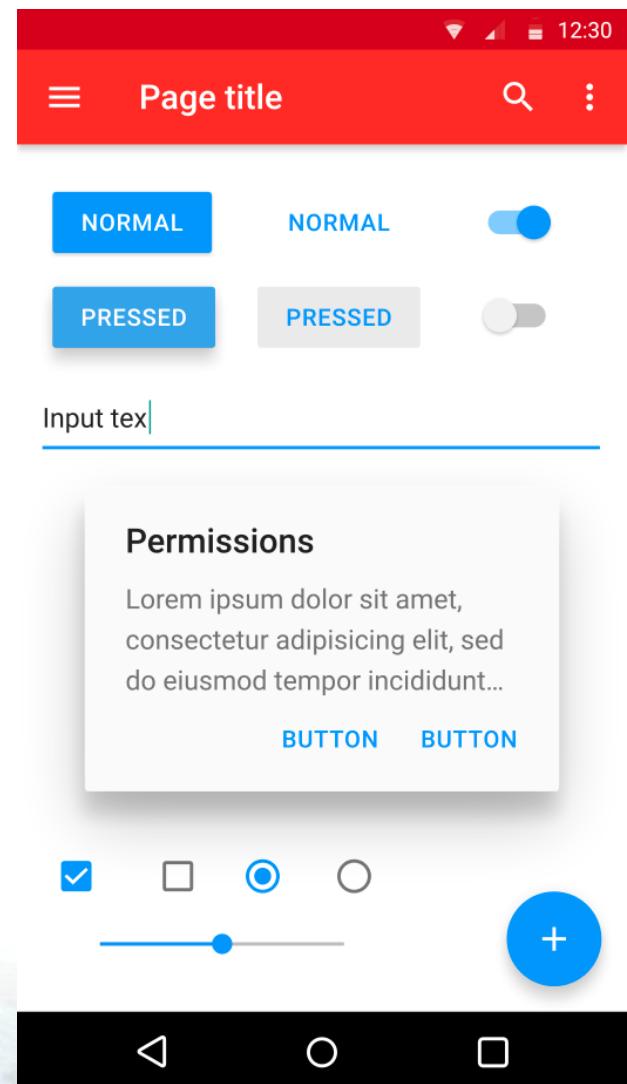


界面设计：Material Design

□ Material Design 是由 Google 开发的设计语言。扩展于 Google 即时的“卡片”设计，其基于网格的布局、响应动画与过渡、填充、深度效果（如光线和阴影）。

□ 可参考官网的介绍：

□ <https://developer.android.google.cn/design/material/>



□ JUnit

- 单元测试

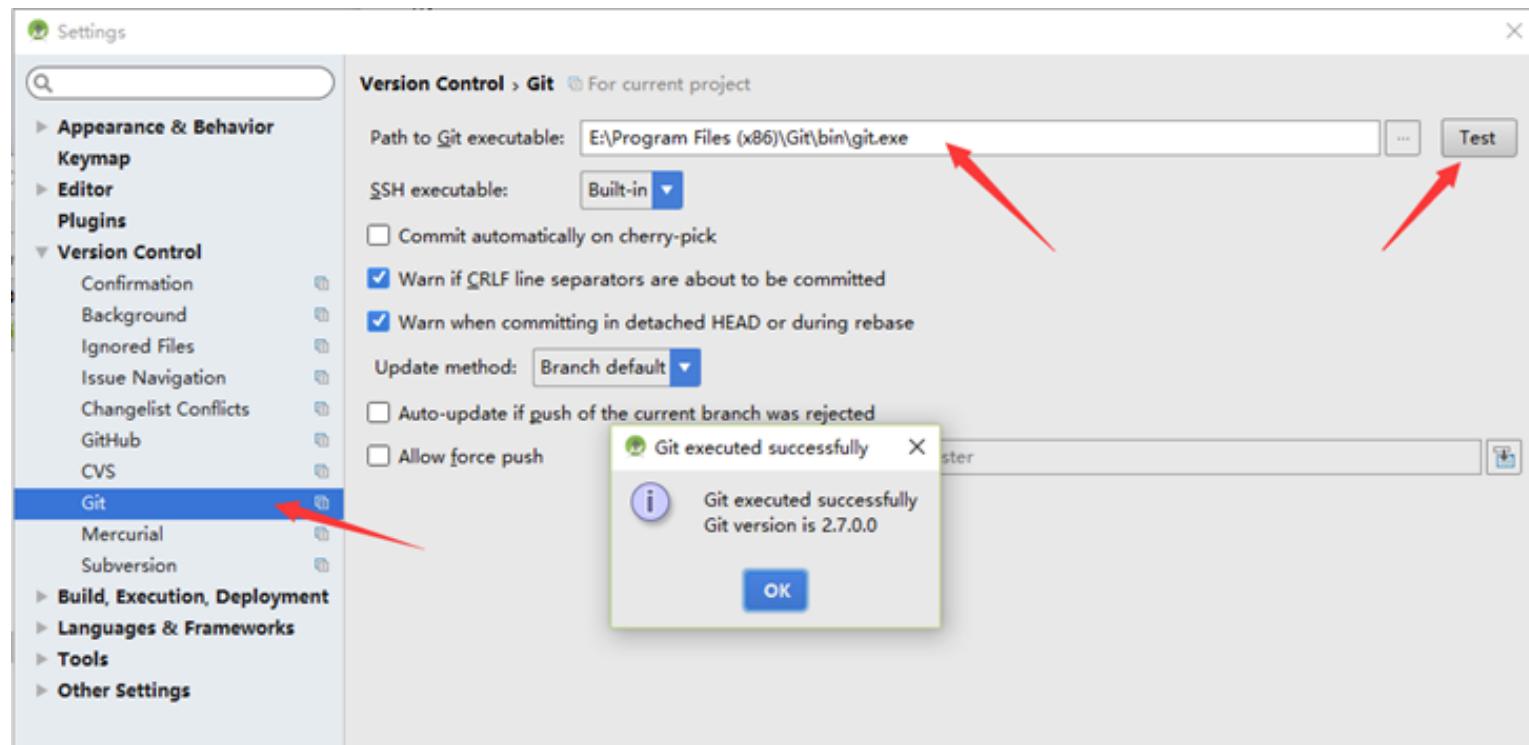
□ Espresso

- UI测试
- AS新建的项目中自动集成了Espresso
- <https://developer.android.com/training/testing/espresso/index.html>

□ Monkey Test

- 随机UI测试
- 脚本录制回放
- 鲁棒性测试





Android Studio中可以方便地进行Git操作

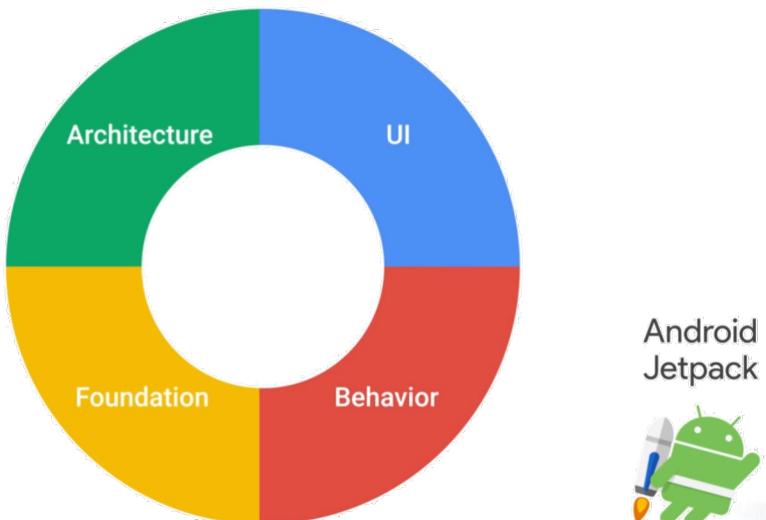


Android 新技术

库	说明
activity *	访问基于 Activity 构建的可组合 API。
appcompat *	允许在平台的旧版 API 上访问新 API（很多使用 Material Design）。
camera *	构建移动相机应用。
compose *	使用描述界面形状和数据依赖项的可组合函数，以编程方式定义界面。
databinding *	使用声明性格式将布局中的界面组件绑定到应用中的数据源。
fragment *	将您的应用细分为在一个 Activity 中托管的多个独立屏幕。
hilt *	扩展了 Dagger Hilt 的功能，以实现 androidx 库中某些类的依赖项注入。
lifecycle *	构建生命周期感知型组件，这些组件可以根据 Activity 或 Fragment 的当前生命周期状态调整行为。
Material Design 组件*	适用于 Android 的模块化、可自定义 Material Design 界面组件。
navigation *	构建和组织应用内界面，处理深层链接以及在屏幕之间导航。
paging *	在页面中加载数据，并在 RecyclerView 中呈现。
room *	创建、存储和管理由 SQLite 数据库支持的持久性数据。

□ Jetpack

□ **Jetpack是一个由多个库组成的套件，可帮助开发者遵循最佳做法、减少样板代码并编写可在各种 Android 版本和设备中一致运行的代码，让开发者可将精力集中于真正重要的编码工作。**



<https://developer.android.google.cn/jetpack?hl=zh-cn>



Android-基于Web开发

```
<WebView  
    android:id="@+id/wv_webview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

```
public class WebViewActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_web_view);  
        //获得控件  
        WebView webView = (WebView) findViewById(R.id.wv_webview);  
        //访问网页  
        webView.loadUrl("http://www.baidu.com");  
        //系统默认会通过手机浏览器打开网页，为了能够直接通过WebView显示网页，则必须设置  
        webView.setWebViewClient(new WebViewClient(){  
            @Override  
            public boolean shouldOverrideUrlLoading(WebView view, String url) {  
                //使用WebView加载显示url  
                view.loadUrl(url);  
                //返回true  
                return true;  
            }  
        });  
    }  
}
```

WebViewDemo



百度一下

文库 图片 知道 新闻 百科
应用 地图 贴吧 hao123 更多

小说 游戏 下载

下载： 百度客户端 百度应用 地图

Baidu 京ICP证030173号



Peking
University

□ Android

➤ kotlin

- guolindev/SunnyWeather: 这个例子的最初来源是《第一行代码-第三版》，作者郭霖。这是一个天气预报的例子，非常小巧但涉及到的知识点很多，比如如何写界面（传统方法）、如何做网络交互、如何管理多个界面（Activity）和应用的状态。作者本人没有将代码放到github上，所以有一些人读完书后将自己实践的效果放了上来，同时用了一些google的新技术，作者在写书的时候这些技术还没出来。代码质量优秀。编译结果可运行
- android/Sunflower: 这个例子直接来自google。上一个例子中写界面用的是传统的XML，但安卓团队最近建议大家使用更加贴近底层的 jetpack-compose，直接用Kotlin的代码写界面；此外，还展示了如何使用 material design。这是一个展示各种向日葵种类的APP



□ Android

➤ java

- cachecats/LikeMeiTuan: 这是一个仿美团的开源项目，展示了如何使用Java做原生开发。除了网络请求的封装，该有的功能都有了。作者还给出了详细的开发流程。现在开发Android不建议使用Java，Android Studio对纯Java的项目支持也不是很好了（主要和Gradle版本有关）
- Yalantis/uCrop: 这个例子展示了如何使用 Java 编写一个简单的裁剪图片的APP，其中包含了一些Java使用原生共享库的例子。这个例子可能不适合用来给学生作为编写一个独立应用的全面的例子，但可以作为一个很好的补充
- 其他例子: 可以找到其他Android官方的例子——这些都不是完整的项目，是为开发中的概念编写的小例子（包括Kotlin）



推荐书目

口第一行代码（第三版）作者：郭霖

基于Kotlin语言，针对Android10，讲解了近些年开发的Android的主要技术和最新变化，以及Android项目的组织架构的最佳实践。



Peking
University



移动开发

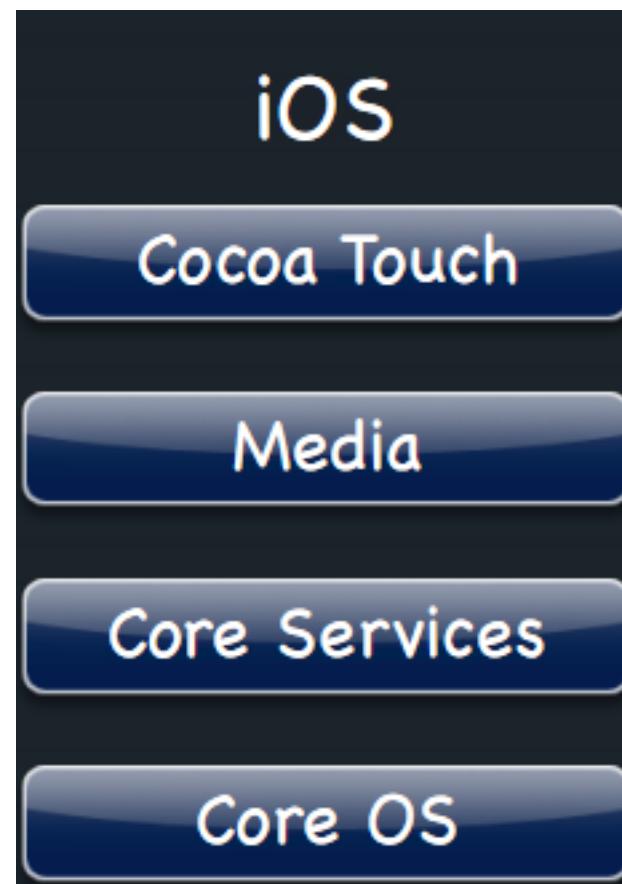
IOS



Peking
University



系统结构



从上到下iOS分为4层，理解这4层的作用对于我们编程的作用相当大。

Core OS: 该层主要包含一些操作系统的内核、文件系统、安全性、证书之类的，一般我们很少接触该层。

Core Services: 该层主要包含跟网络相关的东西，还有数据库、线程、电话簿、网络服务等，该层为核心服务层。

Media: 看到这名字也知道，该层主要是跟多媒体相关的，包含视频、音频、图片、pdf、OpenAL、OpenGL等。

Cocoa Touch: 该层为操作系统的最顶层，该层包含了构建iOS程序的关键framework，还有多任务啊、标准的view controller啊等等。



- 近几年由于移动市场红利的消失，国内移动开发的技术资料更新速度有所下降，所以最佳的学习资料推荐是apple developer的官方网站，以及WWDC每一年的各种视频，会及时更新iOS新特性对应的开发资料
- 入门学习资料可以参考
<https://www.raywenderlich.com/ios>
- 尽量不要从中文博客中学习，因为随着swift的推出和成熟，iOS的开发方式和最佳实践都在不断更新

- 0.选择Swift语言
- 1.快速搭建一个基础App，找到成就感
- 2.UI上优先使用SwiftUI
- 3.合理运用MVC等架构
- 4.合理运用开源库



- 目前Xcode11结合SwiftUI已经解决了曾经iOS相对于Android开发的痛点：包管理与界面版本管理



The screenshot shows the Xcode IDE with a SwiftUI project named "macOS Namerizer". The code editor displays "Content.swift" with the following content:

```
1 //  
2 // Content.swift  
3 //  
4  
5 import SwiftUI  
6  
7 struct Content : View {  
8  
    @State var model = Themes.listModel  
9  
    var body: some View {  
        List(model.items, action: model.selectItem)  
            .Image(item.image)  
            .VStack(alignment: .leading) {  
                Text(item.title)  
                Text(item.subtitle).color(.gray)  
            }  
    }  
}
```

The Xcode interface includes a "Views" library sidebar on the right, a preview window showing a list of items with images and titles, and a "Text" inspector panel for the selected "Text" view.

Views Library:

- Z Stack
- Spacer
- List
- Text
- Button
- Toggle
- Date Picker
- Slider

Preview Window:

Category	Name	Count
Mountains	7 Names	7
Big Cats	9 Names	9
Farms	5 Names	5
Food	4 Names	4
Beaches	10 Names	10
Deserts	3 Names	3
Bridges	13 Names	13

Text Inspector (for Bridges item):

- Font: Inherited
- Weight: Inherited
- Color: Gray
- Alignment: Inherited
- Line Limit: - inherited +
- Padding: Inherited



Swift vs. Objective-C

□ Swift

- 新的编程模式
- 更友好的语法
- 更精简
- 不断加入的新功能
- ...

□ Objective-C

- Runtime
- 遍布代码的指针
- 跨平台
- ...



Peking
University



- 对于初学者来说，推荐从Swift入手，其语法比较简单，和常见的高级语言类似，并且Apple提供了很多开发学习的工具。但是从长远来讲，Objective-C仍然没有被淘汰。
- 二者各自都有优势，目前跨平台或是与C++混编仍然需要通过Objective-C桥接，但是大方向上来讲Apple是在逐渐用Swift替换Objective-C的。

- UI设计需要符合iOS设计理念，iOS都有自己的一套完整的设计理念与设计要求，其中包括素材大小等都有着严格规范，在App实际上线的时候都会进行严格要求。
- 此处不详细介绍，详情可见官方开发者中心的设计板块：

<https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/>



□ --UI related

- DZNEmptyDataSet
- ChameleonFramework
- MGSwipeTableCell
- SVProgressHUD/MBProgressHUD





推荐的三方库

□--Utility related

- MagicalRecord
- SwiftyJSON
- MJExtension
- CocoaLumberjack
- KeychainAccess

□--Network related

- Alamofire
- SDWebImage



Peking
University

□ iOS

- [awosome-iOS](#): iOS生态下各种功能的开源仓库集合
- [open-source-ios-apps](#): 开源的iOS软件集合
- [Swift-30-Projects](#): 小型软件项目集合
- [awesome-python-webapp](#): 博客网站App





常用技术 跨平台开发



Peking
University

□ 何谓跨平台？

- 代码共享：一次编写，多平台运行，帮助更快构建软件产品
- （多数具有）近native的表现与性能

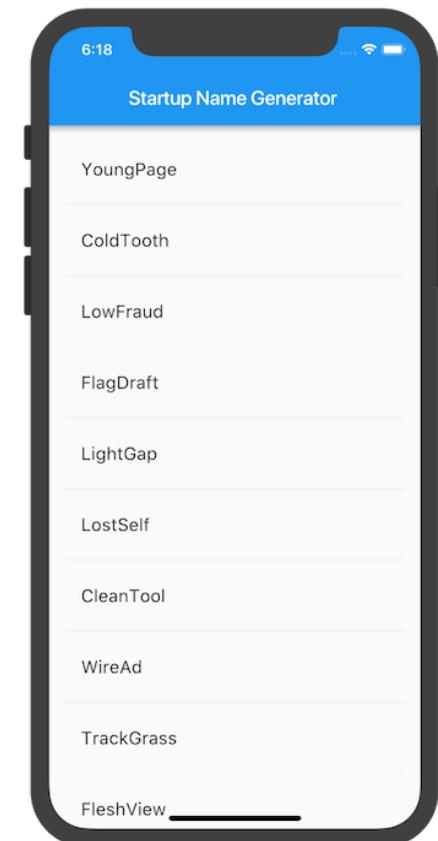
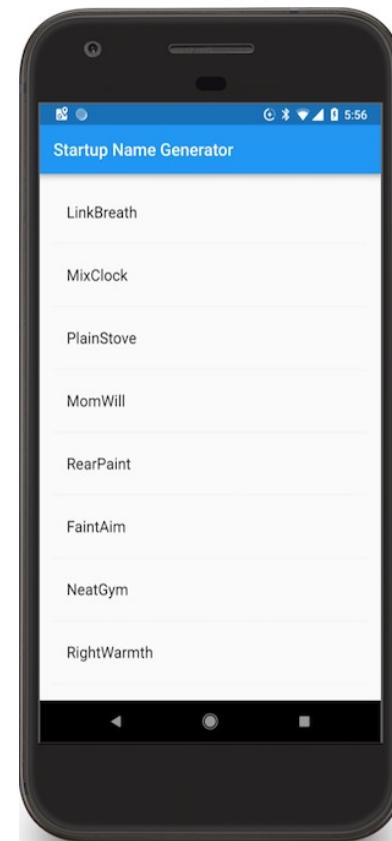
□ 选择跨平台的原因？

- 需要（想要）快速开发多平台移动应用
- 更熟悉的技术栈：JavaScript、.NET.....
- 想要尝试新技术
-



□ Flutter: Google天选之子

- 漂亮与流畅: UI漂亮、可媲美原生性能
- 响应式框架、开发热加载: 开发简便
- 能够访问原生代码: 易于复用
- 使用语言
 - Dart: Google开发
 - 广泛用于web、服务器、移动应用等领域



口万物皆Widget

- 使用嵌套的Widget编写界面：对界面掌控能力更好
- 基本由无状态Widget和有状态Widget组成

口使用State管理状态

- 为有状态的组件编写State类、管理其状态，逻辑清晰
- 类似react的开发模式，有前端经验更容易上手

```
void main() => runApp(MyApp());  
  
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Welcome to Flutter',  
      home: RandomWords(),  
    ); // MaterialApp  
  }  
}
```

```
class RandomWords extends StatefulWidget {  
  @override  
  _RandomWordsState createState() => _RandomWordsState();  
}  
  
class _RandomWordsState extends State<RandomWords> {  
  final _suggestions = <WordPair>[];  
  final _biggerFont = TextStyle(fontSize: 18.0);  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Startup Name Generator'),  
      ), // AppBar  
      body: _buildSuggestions(),  
    ); // Scaffold  
  }  
}
```

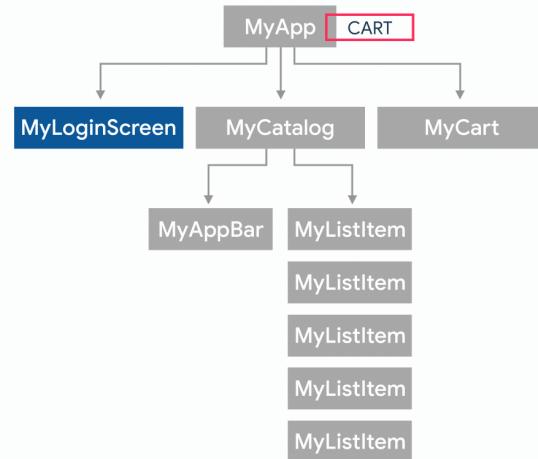
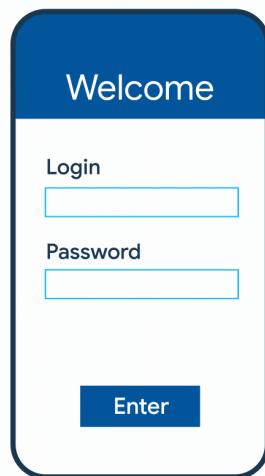
Flutter开发逻辑

口 声明式框架 flutter

- “UI” 是APP内“状态”的反映
- 改变状态、用户界面随之改变

口 命令式框架 (原生android、ios)

- UI先行，用xml等手段设计UI
- 为UI背后添加命令式代码逻辑



UI = f(state)

The layout
on the screen

Your build
methods

The application state



Peking
University

□ 开发环境

- Android: Android Studio
- iOS: Xcode

□ 编辑器

- 可以直接使用IDE
- 也可选择使用Visual Studio Code (更轻量)



□ 单元测试

- Flutter自带单元测试框架，对方法、类进行单独测试
- 引用测试包：`import 'package:test/test.dart'`；书写测试用例，可在VSCode、terminal中运行测试
- 参考：[An introduction to unit testing – Flutter](#)

□ 组件测试

- Flutter自带，对Flutter中的Widget进行测试，可类比UI测试
- 引用`flutter_test`依赖，使用`WidgetTester`进行测试
- 参考：[An introduction to widget testing – Flutter](#)

□ 集成测试

- Flutter自带，对整个app或app中一个或几个模块进行集成测试
- 引用`flutter_driver`依赖（作用类似webdriver），驱动app进行测试
- 参考：[An introduction to integration testing - Flutter](#)

□ 官方文档

- <https://flutter.dev/docs>

□ Dart语言学习

- [Bootstrap into Dart – Flutter](#)

□ 应用示例库

- [Flutter Gallery](#)



Flutter常见第三方库

□ 网络相关

- dio
- http

□ UI相关

- Toast: fluttertoast
- Webview:
flutter_webview_plugin
- 富文本编辑器: zefyr
- 下拉刷新:
flutter_easyrefresh
- 图表: fl_chart
- 图片缓存:
cached_network_image
- 动画: flutter_spinkit

□ Utility相关

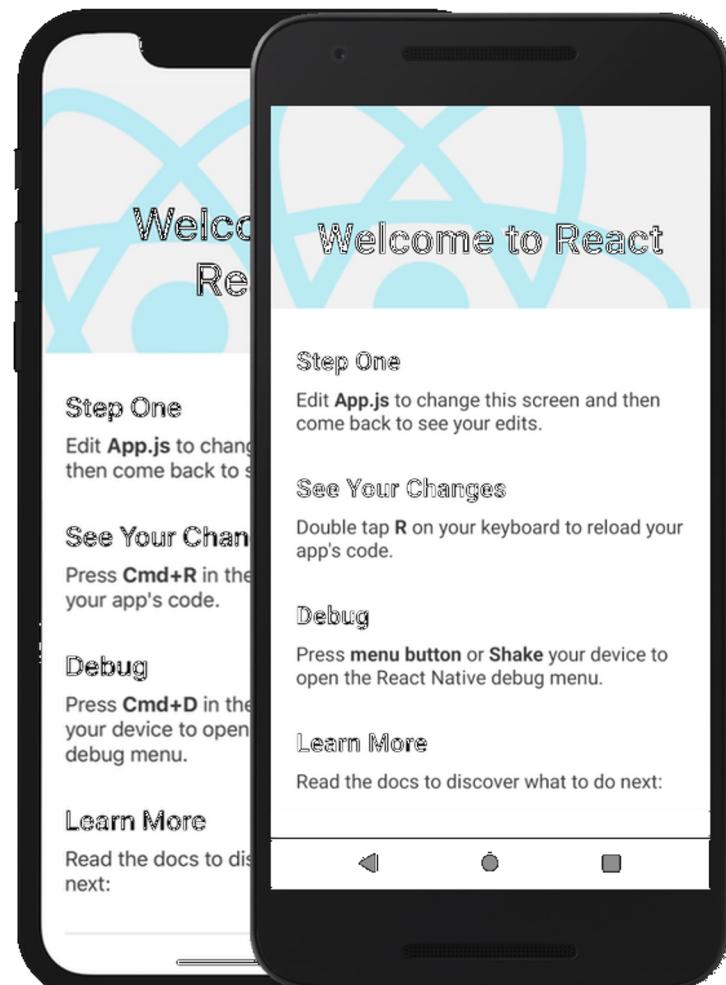
- 状态管理: provide、provider
- 消息传递:
event_bus
- 权限管理:
simple_permissions
- 二维码: qr_flutter



Peking
University

□ React-Native: Facebook钦定之人

- 更好地支持原生组件：
标记元素都被转化为平
台UI组件
- 熟悉的跨平台开发体验：
类似React
- 使用语言： JavaScript
- 依托node的丰富生态



React-Native开发模式

□ 同样的声明式框架

- 继承Component类
- 使用JSX语法声明组件
- 通过props向子组件传递参数
- 通过state管理组件状态
- React中的dom组件被替换为原生组件

```
// ReactJS Counter Example Using Hooks!
import React, { useState } from 'react';

function App() {
  const [count, setCount] = useState(0);

  return (
    <div className="container">
      <p>You clicked {count} times</p>
      <button
        onClick={() => setCount(count + 1)}
        title='Click me'
      ></button>
    </div>
  );
}

// CSS
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}
```

```
// React Native Counter Example Using Hooks!
import React, { useState } from 'react';
import {
  View, Text, Button, StyleSheet
} from 'react-native';

function App() {
  const [count, setCount] = useState(0);

  return (
    <View styles={styles.container}>
      <Text>You clicked {count} times</Text>
      <Button
        onPress={() => setCount(count + 1)}
        title='Click me'
      />
    </View>
  );
}

// React Native Styles
const styles = StyleSheet.create({
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center'
})
```



Peking
University

□ 与原生平台更好的结合

- React-Native中的 component是对原生 android、ios系统基础组件的包装，React-Native的产出是真正的“移动应用”
- React-Native完美兼容使用 Objective-C、Java或是 Swift编写的原生组件。
 - 可以针对应用的某一部分特别优化，中途换用原生代码编写
 - 可以实现应用的一部分用原生，一部分用React-Native

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';
import { TheGreatestComponentInTheWorld } from './your-native-code';

class SomethingFast extends Component {
  render() {
    return (
      <View>
        <TheGreatestComponentInTheWorld />
        <Text>
          上面这个TheGreatestComponentInTheWorld组件完全可以使用原生Objective-C、Java或是Swift来编写 - 开发流程并无二致。
        </Text>
      </View>
    );
  }
}
```



□ 开发环境

- Android: Android Studio (其实主要是Android SDK)
- iOS: Xcode
- Js项目开发环境: npm、node、react-native-cli

□ 编辑器

- Visual Studio code
- WebStorm



口 测试框架

- React-Native主体都由js code构成
- 可以使用js测试框架Mocha或Jest进行测试
- Mock
 - Mock对React-Native的测试更加重要，因为部分组件可能是原生组件，必须被mock掉
 - 可以考虑使用Mock.js等mock手段

口 组件测试

- React-Native提供React Native Testing Library为其中的组件提供测试的支持
- [Getting Started | React Native Testing Library \(callstack.github.io\)](#)



口官方文档

- [React Native • A framework for building native apps using React \(reactnative.dev\)](#)

口社区内容

- [The most insightful stories about React Native – Medium](#)
- [Reactnative – DEV](#)

口示例库

- [React Native Example for Android and iOS \(reactnativeexample.com\)](#)



□ UI相关

- 导航组件: react-native-router-flux
- 抽屉: react-native-drawer
- 滑动组件: react-native-swiper
- 下拉刷新: react-native-refreshable-listview
- 滑动删除: react-native-swipe-list-view
- Material Design: mrn

□ 其它

- 可使用node中的一系列库





常用技术 后端技术



Peking
University

□ 选择编程语言

- 选择你熟悉的语言
- 选择简单易上手的语言
- 选择社区支持丰富的语言

□ 确定编程框架

□ 选择数据库

□ 设计接口规范

- 统一接口命名风格
- 统一参数传输形式
- 统一返回值格式
- 统一异常处理格式



- PHP
- Java
- Python
- Node.js
- 其它的
 - Golang, Erlang, Ruby, C++ , C#



□ PHP——号称世界上最好的语言

- 最早出现是用来替代cgi，可以嵌入到html中
- 曾经后端的霸主，当然现在依然是后端使用人数最多的
- 目前来看，PHP正在被逐渐淘汰，但是随着PHP7.0的推出，可能会迎来第二春

➤ 框架

- Yii
- Laravel
- Symfony



Symfony



Peking
University

□ Java——使用人数最多的语言

➤ Java EE框架

- Spring
- Struts 2
- Hibernate
- Spring MVC
- MyBatis



➤ 优势：架构成熟，要啥有啥，运行较快

➤ 缺点：概念、规范、架构太多太臃肿，初学者往往无所适从



spring+springMVC+MyBatis

□ spring

- 比EJB更加轻量级和简单的JAVA EE编程模型
- 核心概念：依赖注入（IOC）、面向切面编程（AOP）
- 由spring容器统一管理JavaBean

□ springMVC

- 由spring提供的MVC框架
- 核心是带有@Controller注解的控制器类
- 通过@ResponseBody注解和引入JSON、xml依赖，抛弃视图层，实现Restful API

□ MyBatis

- 流行的ORM框架，也可用Hibernate或其它ORM框架

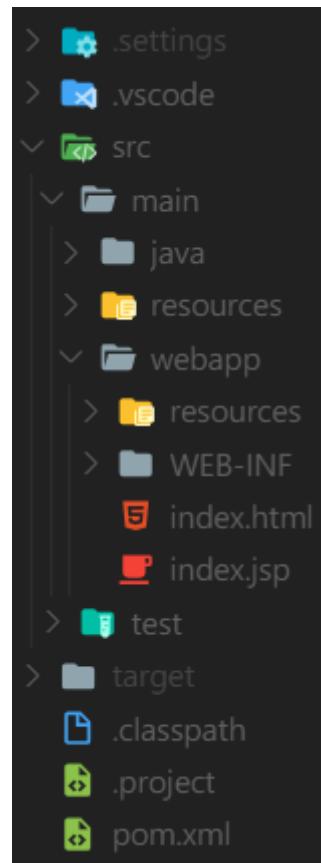
介绍的一系列框架里，只有这一套还得自己部署到服务器才能跑（如Tomcat）

□一般开发步骤

➤ 一般使用maven或gradle构建项目

➤ 以maven为例：

- 使用mvn archetype:generate指令生成项目结构（选择javaweb项目）
- 在pom.xml文件中管理项目依赖
- 在web.xml中配置spring的DispatcherServlet
- 在DispatcherServlet指定路径下通过xml文件配置beans (IOC)
 - 当然现在也可以用java注解配置了，能简单一点
- 编写带有@Controller注解的类，实现业务逻辑
 - 使用Mybatis等ORM框架时，还需要一些ORM配置xml
- 执行maven生命周期，构建项目
 - 编译、测试、导出war包等
- 把war包拖到tomcat服务器里，配置路由，运行tomcat，it should work



```
@Controller  
@RequestMapping("/su")  
public class AreaController {  
    Logger logger = LoggerFactory.getLogger(AreaController.class);  
    @Autowired private AreaService areaService;  
    @RequestMapping(value = "/listarea", method = RequestMethod.GET)  
    @ResponseBody private Map<String, Object> listArea() {  
        logger.info("==start==");  
        long startTime=System.currentTimeMillis();  
        Map<String, Object> modelMap = new HashMap<>();  
        List<Area> areas = new ArrayList<>();  
        try {  
            areas = areaService.getAreaList();  
            modelMap.put("rows", areas);  
            modelMap.put("total", areas.size());  
        } catch (Exception e) {  
            logger.error(e.toString());  
            e.printStackTrace();  
            modelMap.put("success", false);  
            modelMap.put("errMsg", e.toString());  
        }  
        long endTime=System.currentTimeMillis();  
        logger.debug("costTime:[{}ms]", endTime - startTime);  
        logger.info("==end==");  
        return modelMap;  
    }  
}
```

□ spring boot借助以下特性极大简化了spring开发

- Spring boot starter: 整合常用依赖
- 自动配置: 自动化配置需要的JavaBean
- CLI: 使用 groovy语言、可以轻松使用 spring boot CLI运行web应用程序，无需进一步配置和部署到web服务器
- Actuator: 增加管理特性

□ Spring和spring boot

- 有点像手动挡和自动挡.....
- Spring boot明显更加新手友好
- 但并不是说不需要spring知识



```
Hi.groovy
1  @RestController
2  class Hi {
3      @RequestMapping("/")
4      String hi() {
5          return "Hi!"
6      }
7  }
```

□ Spring自带测试框架

➤ 声明依赖: org.springframework:spring-test

□ 可能还需要mockito等框架创建mock对象

□ 使用MockMvc模拟对Controller的请求进行测试

```
@Test
public void shouldProcessRegistration() throws Exception {
    SpitterRepository mockRepository = mock(SpitterRepository.class);
    Spitter unsaved = new Spitter("jbauer", "24hours", "Jack", "Bauer", "jbauer@ctu.gov");
    Spitter saved = new Spitter(24L, "jbauer", "24hours", "Jack", "Bauer", "jbauer@ctu.gov");
    when(mockRepository.save(unsaved)).thenReturn(saved);

    SpitterController controller = new SpitterController(mockRepository);
    MockMvc mockMvc = standaloneSetup(controller).build();

    mockMvc.perform(post("/spitter/register")
        .param("firstName", "Jack")
        .param("lastName", "Bauer")
        .param("username", "jbauer")
        .param("password", "24hours")
        .param("email", "jbauer@ctu.gov"))
        .andExpect(redirectedUrl("/spitter/jbauer"));

    verify(mockRepository, atLeastOnce()).save(unsaved);
}
```

□ spring + springMVC + mybatis, spring boot

- [Yin-Hongwei/music-website](#): 这是一个音乐网站。相比于下一个例子，它足够专注：主要部分只涉及到 SpringBoot, Spring MVC, Vue, MyBatis。代码量相对小一些，适合初学者练习
- [macrozheng/mall](#): 这是一个商城的例子，包括前台商城系统及后台管理系统，基于SpringBoot+MyBatis实现，采用Docker容器化部署。还包含了很多Java技术栈的其他组件，如 ElasticSearch、Redis、Docker、Vue等。作者将项目分成若干个子项目，如首页门户、商品推荐、商品搜索、商品展示、购物车、订单流程、会员中心、客户服务、帮助中心等模块。可以参考自己的需求，针对性的查看其中的子项目



□ Python——人生苦短，我用Python

- 上手快，易读，易维护，无比强大的扩展性
- 在python3和python2的兼容性上存在一些问题，现在导致python2成为了一个包袱，建议直接选择python3
- 限制：解释型语言，线程不能利用多CPU，速度慢
- 框架
 - Django
 - Flask
 - Tornado



□ Django：面向相对大型的应用，提供模块更加全面，开发较方便

- **django-admin startproject mysite**自动创建项目
- **进入项目目录，运行python .\manage.py runserver**
就能打开开发服务器

```
> python .\manage.py runserver
Watching for file changes with StatReloader
Performing system checks ...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 14, 2019 - 14:52:02
Django version 2.2.6, using settings 'mysite.settings'
Quit the server with CTRL-BREAK.
Not Found: /echo/hello
[14/Oct/2019 14:52:25] "GET /echo/hello?echo=wdnmd HTTP/1.1" 404 2146
```



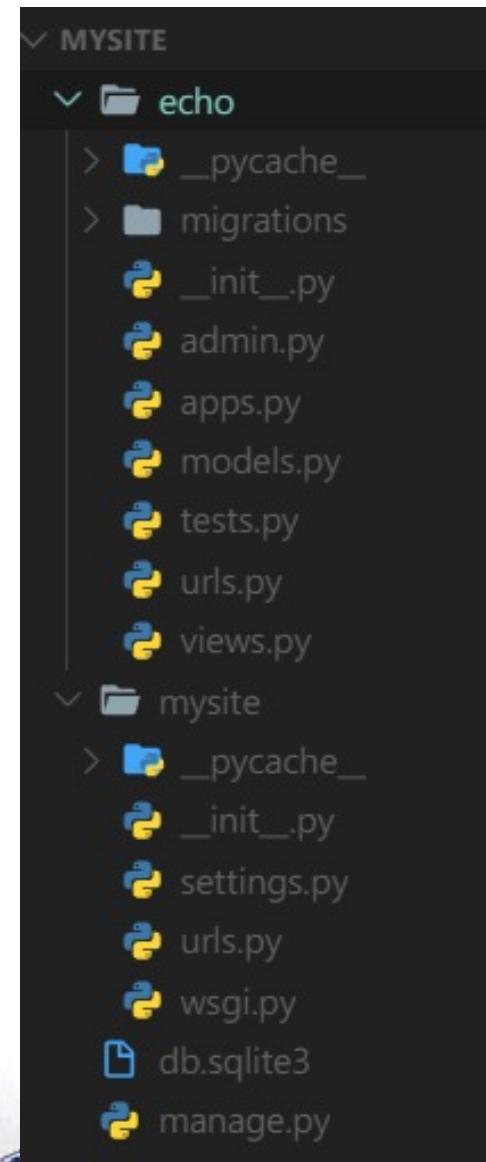
口项目结构

➤ 最开始仅有manage.py和mysite目录

- manage.py: 一个命令行工具，用于与 Django 进行不同方式的交互脚本，和 java 中 maven 作用有点像
- urls.py: 项目的一级路由逻辑
- settings: 项目设置

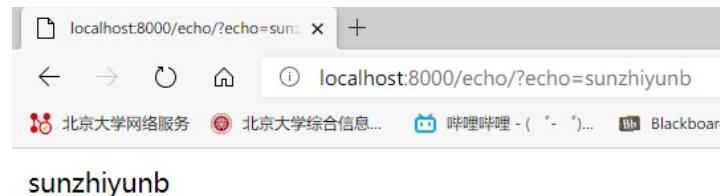
➤ 通过 python .\manage.py startapp echo 命令创建 app

- app 代表着 django 中的功能模块划分
- 在 views.py 编写业务逻辑
- 可以在 app 中同样编写一个 urls.py 负责项目的二级路由逻辑



口业务逻辑编写的常见方法

- 使用主项目中的urls.py做一级路由转发
- 使用app中的urls.py做二级路由转发
- 以app的views.py为起点编写业务逻辑



```

tests.py
urls.py
settings.py
mysite > urls.py > ...
16 from django.contrib import admin
17 from django.urls import path,
18     include
19 urlpatterns = [
20     path('admin/',
21         admin.site.urls),
22     path('echo/', include
23         ('echo.urls'))
24 ]

```

```

urls.py
echo > urls.py > ...
1 from django.urls import path
2
3 from . import views
4
5 urlpatterns = [
6     path('', views.hello,
7          name='hello'),
8 ]

```

```

views.py
echo > views.py > ...
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3
4 # Create your views here.
5
6 def hello(request):
7     if request.method == 'GET':
8         echo_message =
9             request.GET.get('echo',
10                 default='hello world')
11         return HttpResponseRedirect(
12             echo_message)
13     return HttpResponseRedirect("hello
14         world")

```

口使用django自带测试库django.test

- 在app的tests.py中编写测试
- 用django.test.TestCase构建测试用例
- 用django.test.Client实现 get 或 post 内容，检查一个网址返回的网页源代码
- 使用python manage.py test app指令对app执行自动化测试

```
尹航@ZEPHYRUS-DELAVE > C:\workspace\pythonTrain\mysite
```

```
> python .\manage.py test echo  
Creating test database for alias 'default' ...  
System check identified no issues (0 silenced).  
.
```

```
Ran 1 test in 0.008s
```

```
OK
```

```
Destroying test database for alias 'default' ...
```

```
from django.test import TestCase, Client  
  
# Create your tests here.  
  
class EchoTests(TestCase):  
    def test_echo(self):  
        c = Client()  
        response = c.get('/echo/',  
{'echo' : 'hello'})  
  
        self.assertEquals(bytes.decode(  
response.content), 'hello')
```

□ Flask：面向小型应用的微框架，开发上手极其简单，应用伸缩性好

- 极简情况下，一个文件就能写完
- 你甚至可以把它直接整合到已有Python项目里，直接包装成一个服务



口 开发流程

- 导入必要的包
- 创建一个app
- 写个函数做处理逻辑
- app.run()
- 执行py文件就能跑起来了，太简单

```
from flask import Flask  
from flask import jsonify  
from flask import request
```



```
app = Flask(__name__)
```



```
@app.route('/asso', methods=["POST"])  
def get_synonyms():  
    ret = 'failed'  
    try:  
        req = simplejson.loads(request.get_data().decode("utf-8"))  
        synonyms = server_func.associate(req["keyword"])  
        ret = jsonify(synonyms)  
    except Exception:  
        ret = 'failed'  
    return ret
```



```
if __name__ == '__main__':  
    app.run(host='0.0.0.0', port=3000)
```

□ 测试方法

- 使用flask的测试扩展 Flask-Testing
- 通过继承 flask_testing.TestCase 构建测试用例
 - 覆盖TestCase的 create_app 方法来获取测试的app
 - 增加测试方法来进行单元测试
- 使用unittest.main()执行自动化测试
 - 同样执行文件即可进行测试

```
from flask import Flask, jsonify
from flask_testing import TestCase
import unittest
import server
from server import app

class MyTest(TestCase):
    def create_app(self):
        app.config['TESTING'] = True
        return app

    def test_hello(self):
        response = self.client.get('/')
        self.assertEquals(response.json, dict(success=True))

if __name__ == "__main__":
    unittest.main()
```



□ Django

- <http://www.liujiangblog.com/>
- <https://docs.djangoproject.com/en/2.2/>

□ Flask

- <https://dormousehole.readthedocs.io/en/latest/>

□ 其它

- 建议使用Pipenv管理你们的依赖
 - <https://pipenv.readthedocs.io/en/latest/>



□ Nodejs——异步IO，前后统一

- Node.js 是一个基于Google V8引擎建立的一个平台，用来方便地搭建快速的，易于扩展的网络应用
- Node.js 借助事件驱动，非阻塞 I/O 模型变得轻量和高效，非常适合数据密集型的准实时应用
- 便于书写前后端统一的代码
- 框架
 - Express.js
 - Koajs



□ Express：常用的web应用程序开发框架

- 上手跟flask差不多简单
- 实际使用时还是会分着写路由文件和控制器文件
- 注意别把node_modules传到仓库中

□ 测试：同属js语言，测试同前端测试类似，不赘述

```
var express = require('express')
var http = require('http')
var app = express()
app.get('/index', function(req,res){
    res.send('hello')
})
app.listen(1337, '127.0.0.1')
```



□ 关系型数据库

- 基于关系表的存储方式
 - 关系表容易理解、且易于维护
- 支持使用SQL在多个表之间做非常复杂的查询
- 支持事务，能够维护数据的一致性
- 适用于：大多数场景，要求事务一致性、数据的持久存储的场景

□ 非关系型数据库（NoSQL数据库）

- 基于键值对/文档等存储方式，而非关系表，不使用SQL
- 性能通常较高，能满足高并发的读写需求
- 不基于关系表，数据的灵活性和拓展性更高
- 适用于：对一致性要求不高，而对读写性能要求很高



□ MySQL

- 主流
- 不要钱

□ PostgreSQL

- “世界上功能最强大的开源数据库”
- 也不要钱
- 支持大部分 SQL 标准时提供了许多其他现代特性
 - 复杂查询、外键、触发器、视图、事务完整性、MVCC
- 提供许多方法拓展
 - 增加新的数据类型、函数、操作符、聚集函数、索引



口对象-关系映射 (Object/Relation Mapping, 简称ORM)

- 让程序员不再关注数据库细节，专心在业务逻辑上，程序员可以不懂数据库就可以开发系统。
- 让数据库迁移变的非常方便，如果系统需要更改使用的数据库，直接改配制就好了。
- 防SQL注入
- 省时，可快速开发，因为不需要自己写复杂的SQL语句，不需要封装复杂的数据底层，这样可以节省很多时间。

口流行框架

- Sequelize(nodejs)
- Peewee(Python)
- Hibernate, MyBatis(Java)



□ MongoDB

- 基于文档 (BSON) 的存储方式
- 存储对象更加直观，且字段可以动态变化，比关系表灵活
- 不使用SQL，需要一定学习成本
- 适用于：
 - web应用等对读写性能要求高的应用
 - 关系数据库的缓存
 - 日志存储系统

关系数据库 (MySQL)	MongoDB
database	database
table	collection
row	document/object

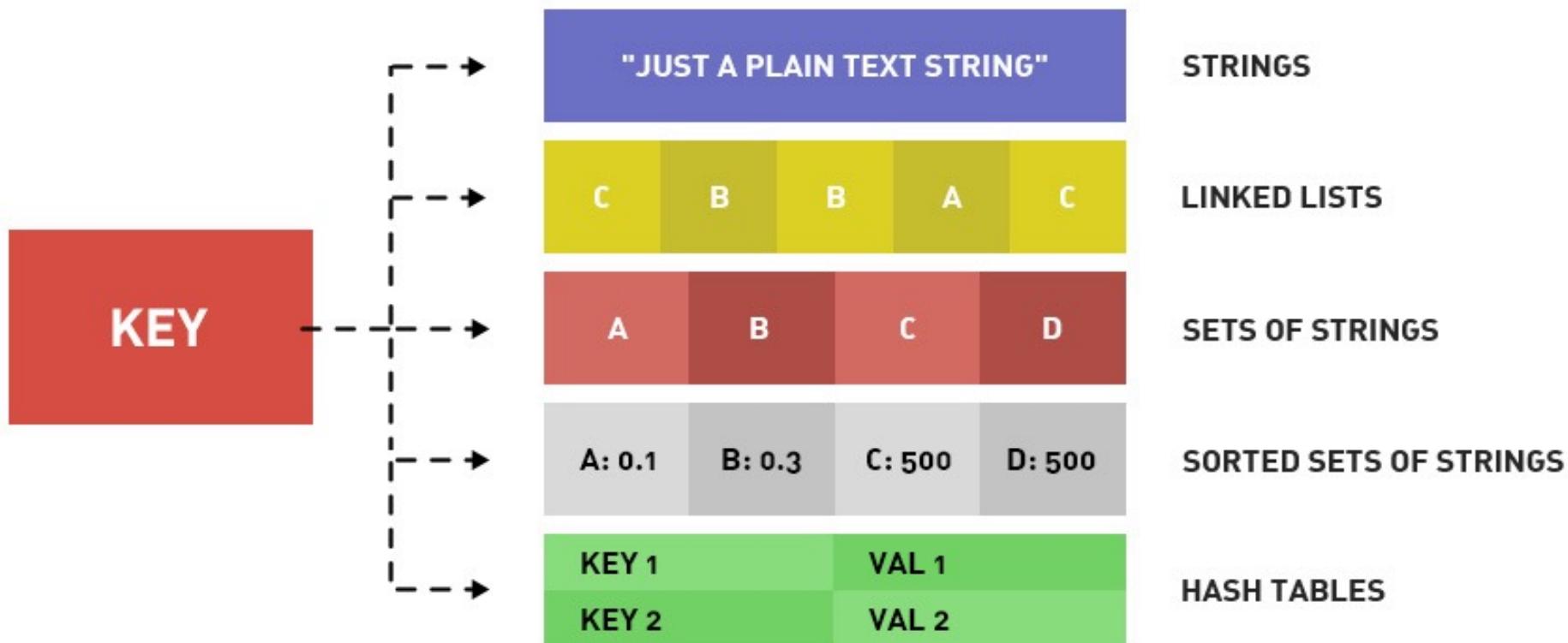
关系数据库中最基本的单元是行，而 MongoDB 中最基本存储单元是 document，典型结构如下所示。

```
{  
  "_id" : ObjectId("51e0c391820fdb628ad4635a"),  
  "author" : { "name" : "Jordan", "email" : "Jordan@123.com" },  
  "postcontent" : "jordan is the god of basketball",  
  "comments" : [  
    { "user" : "xiaoming", "text" : "great player"},  
    { "user" : "xiaoliang", "text" : "nice action" }  
  ]  
}
```



□ Redis

- 基于键值对的内存数据库，一般用作缓存
 - 真的很快
- 需要先看看自己内存够不够
- 使用file snapshotting持久化策略时，数据可能丢失



口推荐使用postman进行测试

- 支持自定义HTTP Request
- 支持Pretty Response
- 支持Collection (分类收藏+共享)

The screenshot displays the Postman application interface. At the top, there is a navigation bar with 'GET' selected, the URL 'https://postman-echo.com/headers', and buttons for 'Params', 'Send', and 'Save'. Below the URL, there are tabs for 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. The 'Headers (1)' tab is active, showing a key-value pair: 'my-sample-header' with the value 'Lorem ipsum dolor sit amet'. There is also a 'New key' placeholder. Below this, there are tabs for 'Body', 'Cookies (1)', 'Headers (14)', and 'Test Results (3/3)'. The 'Body' tab is active, showing a JSON response with the following content:

```
1  {
2   "headers": {
3     "host": "postman-echo.com",
4     "accept": "*/*",
5     "accept-encoding": "gzip, deflate",
6     "cache-control": "no-cache",
7     "my-sample-header": "Lorem ipsum dolor sit amet",
8     "postman-token": "efe2ab01-9b18-41cb-8a61-5b0c90913580",
9     "user-agent": "PostmanRuntime/7.1.1",
10    "x-forwarded-port": "443",
11    "x-forwarded-proto": "https"
12  }
13 }
```





大型网站使用技术比较

Websites	Popularity (unique visitors per month) ^[1]	Front-end (Client-side)	Back-end (Server-side)	Database	Notes
Google.com ^[2]	1,600,000,000	JavaScript	C, C++, Go, ^[3] Java, Python	BigTable, ^[4] MariaDB ^[5]	The most used search engine in the world
Facebook.com	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D, ^[6] Xhp, ^[7] Haskell ^[8]	MariaDB, MySQL, ^[9] HBase Cassandra ^[10]	The most visited social networking site
YouTube.com	1,100,000,000	JavaScript	C, C++, Python, Java, ^[11] Go ^[12]	Vitess, BigTable, MariaDB ^[5] ^[13]	The most visited video sharing site
Yahoo	750,000,000	JavaScript	PHP	MySQL, PostgreSQL ^[14]	Yahoo is presently ^[when?] transitioning to Node.js ^[15]
Amazon.com	500,000,000	JavaScript	Java, C++, Perl ^[16]	Oracle Database ^[17]	Popular internet shopping site
Wikipedia.org	475,000,000	JavaScript	PHP, Hack	MySQL ^[citation needed] , MariaDB ^[18]	"MediaWiki" is programmed in PHP, runs on HHVM; free online encyclopedia
Twitter.com	290,000,000	JavaScript	C++, Java, Scala, Ruby on Rails ^[19]	MySQL ^[20]	140 characters social network
Bing	285,000,000	JavaScript	ASP.NET	Microsoft SQL Server	
eBay.com	285,000,000	JavaScript	Java, ^[21] JavaScript, ^[22] Scala ^[23]	Oracle Database	Online auction house
MSN.com	280,000,000	JavaScript	ASP.NET	Microsoft SQL Server	An email client, for simple use. Mostly known as "messenger".
Microsoft	270,000,000	JavaScript	ASP.NET	Microsoft SQL Server	Software company
Linkedin.com	260,000,000	JavaScript	Java, JavaScript, ^[24] Scala	Voldemort ^[25]	World's largest professional network
Pinterest	250,000,000	JavaScript	Django (a Python framework), ^[26] Erlang	MySQL, Redis ^[27]	
WordPress.com	240,000,000	JavaScript	PHP, JavaScript ^[28] (Node.js)	MariaDB, MySQL	





一个后端案例：Flask

□ [x1204604036/flask backend](#): 这是一个以 Flask 框架为基础搭建的 web 后端，每一次提交都是以 Flask 中一个或几个知识点为基础，方便学习者按照提交的过程进行学习，项目体量比较小，容易上手

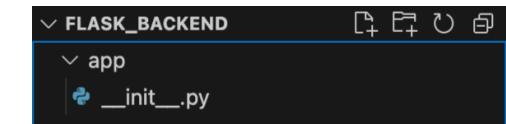


Peking
University



搭建项目

- 安装flask: pip install flask=2.3.2
- 创建项目
- 添加初始化代码
- 运行 flask run
- 浏览器访问: 127.0.0.1:5000



```
from flask import Flask

def create_app():
    app = Flask(__name__)

    @app.route("/")
    def hello():
        return "<h1>hello world</h1>"

    return app
```

hello world



Peking
University



□ Blueprint和请求数据

○ Blueprint定义和注册

- 蓝图 (Blueprint) 是一种组织路由和视图函数的方式，可以用于构建模块化的应用
- 在bp_user.py中定义bp： 路由、 处理逻辑等
- 在__init__.py中注册bp

○ 讲解

The screenshot shows a code editor with two tabs: __init__.py and bp_user.py. The __init__.py tab is active, displaying the following code:

```
# app/user/bp_user.py
from flask import Blueprint, request
bp = Blueprint("user", __name__, url_prefix="/user")
@bp.route("/login", methods=("POST", "GET"))
def login():
    # username = request.form["username"]
    # password = request.form["password"]
    return {"code": 0, "msg": "success"}
```

The bp_user.py tab shows the file content:

```
from flask import Blueprint
bp = Blueprint("user", __name__, url_prefix="/user")
```

A red annotation "定义bp" is placed above the first line of the __init__.py code. A red annotation "路由" is placed above the @bp.route line. A red annotation "处理逻辑" is placed above the return statement.

```
from .user import bp_user
app.register_blueprint(bp_user.bp)
```

注册bp



Peking
University



□ session 会话

- 在Web开发中，Session是一种服务器端存储，用于跟踪用户的状态。它允许Web应用在多个页面请求或访问之间记住用户的信息。Session可以存储用户的登录状态、偏好设置等信息
- 首先在__init__.py中设置密钥

```
def create_app():
    app = Flask(__name__)
    app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
```





□ Session会话的三个接口

- **登录接口：**根据传入的用户名和密码进行验证登录操作，并将登录信息写入 session

```
session.clear()  
session["user_id"] = user.id
```

- **用户信息接口：**登录之后才可访问，在获取用户信息之前 session 中是否有登录信息

```
def login_required(func):  
    def wrapped(*args, **kwargs):  
        if session.get("user_id") is None:  
            return {"code": -1, "msg": "请先登录系统"}  
        return func(*args, **kwargs)  
    return wrapped  
@bp.route("/user_info", methods=("POST",))  
@login_required  
def get_user_info():  
    user_id = session.get("user_id")  
    user_info = {"user_id": user_id} # get user info from db  
    return {"code": 0, "msg": "success", "user_info": user_info}
```

- **退出接口：**注销接口，删除 session 中的登录信息

```
def logout():  
    session.clear()
```



□连接使用数据库

- 安装数据库
- 安装相关依赖: sqlalchemy, pymysql
- 定义相关配置信息和用户数据模型
- 连接和引入数据库
- 改写接口, 从数据库中写入和获取数据



Peking
University

database.py

```
engine_str = "%s:// %s: %s@%s: %s/%s? charset=utf8" % (  
    config.MYSQL_CONNECT_TYPE,  
    config.MYSQL_USERNAME,  
    config.MYSQL_PASSWORD,  
    config.MYSQL_HOST,  
    config.MYSQL_PORT,  
    config.MYSQL_DB_NAME,  
)  
    读取配置信息  
  
engine = create_engine(engine_str)  
        创建引擎  
db_session = scoped_session(  
    sessionmaker(  
        autocommit=False,  
        autoflush=False,  
        bind=engine,  
)  
  
Base = declarative_base()  
Base.query = db_session.query_property()  
  
def init_db(app):  
    import app.models.user  
    Base.metadata.create_all(bind=engine)
```

创建表

```
app > models > user.py > User  
1   from sqlalchemy import Column, Integer, String  
2   from app.database import Base  
3  
4  
5   class User(Base):  
6       __tablename__ = "user"          定义用户数据模型  
7  
8       id = Column(Integer, primary_key=True, autoincrement=True)  
9       username = Column(String(50), unique=True)  
10      email = Column(String(120), default="", comment="邮箱")  
11      password = Column(String(300))
```

__init__.py

```
from . import database  
database.init_db(app)  引入数据库
```

bp_user.py

```
user = User.query.filter(User.username == username).first()  
if not user:  
    return {"code": -1, "msg": "用户不存在"}          查询数据库数据  
  
if not check_password_hash(user.password, password):  
    return {"code": -1, "msg": "密码验证失败"}
```



□ 异常处理

- 异常处理函数

□ 日志操作

- 打印日志和异常

□ 中间件操作

- 在处理请求之前和之后执行一些公共逻辑



Peking
University



常用技术

LLM工具辅助软件开发实践

借鉴：大语言模型辅助的软件开发实践指南，毛新军，杨康，2025.2.



Peking
University

口为什么使用 LLM 工具？

- 更快的开发速度、更高的开发质量
- GitHub Copilot 报告：提升 55% 的开发速度
- 贯穿软件工程的全周期：需求分析、设计、编码、调试、测试、维护
- LLM 不仅帮你“写代码”，而是帮你“更快更好地思考”

口如何对待 LLM 工具？

- LLM 可能出错、生成无法运行的代码、引入安全漏洞
- 理解，检查并验证 LLM 的生成内容
- “你不是被AI取代，而是被会用AI的同事取代”





□ 大模型工具可以辅助学生完成以下几个方面的软件开发实践活动，包括需求获取与分析、软件设计、代码编写、软件测试、运行维护等。

□ 1、需求获取与分析

- 目标:确定软件项目，构思和明确软件需求，定义软件功能和非功能需求。
- 项目选题和需求构思
 - 辅助创意生成：可以利用大模型，针对特定应用领域或技术栈，生成软件项目的需求创意。
 - 例如，输入“假设你是一个富有创意的软件工程师，请你帮助我生成一个基于AI的高校校园应用创意”，deepseek给出如下结果

应用名称：EduPal - 你的AI校园共生伙伴

核心理念：EduPal 不再是一个功能堆砌的“超级App”，而是一个以AI为核心驱动的、具有预测性、主动性和个性化的智能体。它旨在与学生的学术和校园生活“共生”，提前预见需求，无缝解决问题，并促进学生全面成长。



Deepseek从四个方面对这个创意进行了详细介绍：

一、核心AI驱动功能亮点

➤ 1. 学术智能体 (Academic Agent)

- **个性化学习路径规划**: AI通过分析学生的专业培养方案、当前成绩、选课历史和学习行为（如图书馆借阅、视频观看停留点），动态生成并调整专属学习路径。例如：“根据你的线性代数期中成绩，建议你本周优先复习第二章，并完成我为你筛选的这三道典型习题。”
- **AI课程表+**: 不仅是静态课表。它能与校园系统联动，自动提示：“下节课的教室从A栋换到了B栋，已为你规划最优路径，预计步行8分钟，现在出发刚好。”
- **智能笔记与知识图谱生成**: 学生上传课堂录音或拍摄板书照片，AI自动进行语音转文字、提取关键概念、生成结构化笔记，并进一步构建本课程的**可视化知识图谱**，帮助学生看清知识点之间的联系。
- **“救命”模式 (考前冲刺)** : 输入考试日期和范围，AI自动生成一个冲刺时间表，整合了课程重点、往届试题和相关的学习资料（如MO视频链接），并提供自测题。



Peking
University



➤ 2. 生活智能体 (Life Agent)

- **上下文感知的校园导航**: 不仅告诉你怎么走，还能结合实时数据：“去第三食堂？根据当前人流数据，建议你去第二食堂，排队时间少5分钟。”
- **社交联结者**: AI基于兴趣社团、选课重叠度、图书馆借阅相似度等，**匿名且隐私保护地**推荐“可能志同道合”的同学。例如：“发现一位同学和你在同一时间都在图书馆研究‘元宇宙与建筑设计’，并有3个共同社团兴趣，是否愿意匿名打个招呼？”
- **心理健康伙伴**: 内置一个基于大型语言模型的、经过安全训练的聊天机器人。它可以进行日常情绪疏导，识别压力过大的语言模式，并主动推荐校园心理资源（预约咨询、正念冥想课程等）。

➤ 3. 行政与效率智能体 (Admin Agent)

- **智能文档助手**: 帮助学生自动填写各种重复性的行政表格（如奖学金申请、活动报备），只需核对信息即可。
- **一站式问答中心**: 用自然语言询问任何校园相关问题：“奖学金什么时候申请？” “体育馆今晚开门到几点？” “我的校园网为什么断了？” AI直接给出精准答案或操作步骤，无需在不同公众号或网站上寻找。
- **消费与补贴洞察**: 连接校园卡消费数据，AI分析你的消费习惯，并主动提示：“检测到你本月餐饮消费较低，符合‘阳光助餐’补贴的申请条件，是否立即办理？”



Peking
University



二、技术栈构想（工程师视角）

➤ 后端：

- **框架：** Python (FastAPI / Django) 或 Go (Gin)
- **AI核心：** LangChain/LlamaIndex 用于构建AI智能体框架，集成多种大语言模型（如GPT-4, LLaMA 3）和专用小模型（语音转文字、OCR）
- **数据存储：** PostgreSQL (主数据库), Redis (缓存), Pinecone/Chroma (向量数据库，用于存储知识图谱和语义搜索)
- **API集成：** 通过校园开放平台API对接教务、一卡通、图书馆等系统

➤ 前端：

- **移动端：** React Native (跨平台) 或 Swift/Kotlin (原生)
- **设计理念：** 极简主义，对话式UI，大量运用卡片和自然语言交互，减少层级菜单。

➤ 基础设施：

- **云服务：** AWS/Azure/GCP
- **容器化：** Docker + Kubernetes
- **MLOps：** 用于持续训练和部署AI模型（如消费模式识别模型）



Peking
University



三、商业模式与可持续性

- **To C (学生)**: 基础功能完全免费。高级功能（如更详细的生涯规划报告、个性化深度分析）可采用订阅制（EduPal Pro）。
- **To B (学校)**: 向高校销售SaaS平台和部署服务，帮助学校数字化升级，费用基于学生人数和功能模块。这是主要的盈利模式。
- **生态合作**: 与校园内外的书店、打印店、咖啡厅合作，提供基于位置的智能优惠券推送（“你常去的图书馆旁边的咖啡店，今天拿铁半价”），从中获得佣金。

四、为什么这个创意能成功？

- **真痛点，非痒点**: 解决了学生“信息孤岛”、“选择困难”和“效率低下”的核心痛点。
- **AI不是点缀，是核心**: AI驱动从“人找信息”变为“信息找人”，创造了前所未有的流畅体验。
- **平台化潜力**: 可以成为高校数字生态的统一入口，价值巨大。
- **高度可扩展**: 初始版本可以聚焦核心功能（课表、导航、问答），后续逐步迭代加入更复杂的智能体。

希望这个名为 EduPal 的创意能激发您的灵感。我愿意继续深入讨论任何一个功能模块的技术细节或实现方案。



➤ 项目选题和需求构思

- **市场调研辅助**: 模型可以帮助分析某个领域的市场需求或技术趋势，提供相关背景信息，帮助团队确定项目的可行性和创新性。(纯生成式大模型工具不合适直接用于市场调研，建议结合有智能搜索功能的大模型工具，例如纳米 AI 搜索、腾讯元宝、豆包等)

➤ 细化和分析软件需求，生成软件需求文档

- **用户故事生成**: 根据用户角色和场景，生成符合实际应用场景的用户故事。
 - 例如：输入“作为顾客，想要搜索商品，以便找到想要购买的商品，请你根据顾客的需求生成用户故事”，大模型工具可以生成详细的用户故事，包括前置条件、后置条件、正常流程、异常流程等。
- **需求澄清与角色扮演**: 让大模型扮演不同的客户和用户角色，通过模拟对话的方式，帮助学生澄清模糊的需求，完善需求细节。





- 例如：
 - 场景 1：同学提出“网站要美观”，大模型可以扮演设计师，询问具体的设计风格、配色方案、字体选择等，帮助学生明确“美观”的具体含义。
 - 场景 2：同学提出“商品搜索功能”，大模型可以扮演电商平台商家，询问搜索算法的精准度、排序规则、筛选条件等，帮助学生优化搜索功能的设计。
- 需求文档生成：利用大模型工具，自动生成初步的需求文档模板。
 - 例如：输入“假设你是一个资深的软件需求分析师，我正在开发一个在线购物网站，请你帮我生成包含功能列表、用户角色以及用况图等内容的需求文档框架，包括功能需求、非功能需求(如性能、安全性等)。”



Peking
University



□2. 软件设计

➤ 目标：逐步明确其解决方案，将需求转化为软件设计方案，包括体系结构、用户界面、数据等方面

➤ 软件体系结构设计

- 辅助架构设计：利用大模型，根据需求描述，生成初步的软件体系结构图，并推荐合适的设计模式和框架。
- 示例 1：输入“假设你是一个经验丰富的软件架构师，我正在开发一个在线购物网站，请你帮我设计一个可扩展、高性能的系统架构，并推荐合适的技术栈。” 大模型可以生成包含前端、后端、数据库、缓存等组件的架构图，并推荐使用 Spring Boot、React、Redis 等技术。
- 示例 2：输入“我正在开发一个基于微服务架构的社交网络应用，请你帮我设计服务划分方案，并推荐合适的通信协议。” 大模型可以生成用户服务、内容服务、消息服务等微服务划分方案，并推荐使用 RESTfulAPI 或 gRPC 进行通信。





➤ 用户界面设计

- **辅助界面原型设计：**利用大模型根据用户角色和功能需求，生成初步的界面 原型图，并提供设计建议。
- **示例 1：**输入“假设你是一个资深 UI 设计师，我正在开发一个在线教育平台，请你为老师和学生角色分别设计课程管理界面和学习界面，并提供设计建议。”大模型可以生成包含课程列表、视频播放、在线测试等元素的界面原型，并建议使用简洁明了的布局和符合教育主题的配色方案。
- **示例 2：**输入“我正在开发一个移动端电商应用，请你设计商品详情页的界面布局，并考虑用户体验和交互设计。”大模型可以生成包含商品图片、价格、购买按钮等元素的界面原型，并建议使用大图展示、滑动查看、一键购买等交互方式。

➤ 数据设计：

- **辅助数据库设计：**利用大模型，根据需求描述，生成初步的数据 库模型，并推荐合适的数据库类型和数据结构。



- **示例 1：**输入“假设你是一个数据库专家，我正在开发一个博客系统，请你帮我设计数据库表结构，并考虑文章的存储、分类、标签等功能。”大模型可以生成包含用户表、文章表、分类表、标签表等表结构的 ER 图，并推荐使用 MySQL 或 PostgreSQL 数据库。
- **示例 2：**输入“我正在开发一个实时聊天应用，请你设计消息存储方案，并考虑消息的实时性、可靠性和可扩展性。”大模型可以生成使用 NoSQL 数据库(如 MongoDB)存储消息的方案，并建议使用消息队列(如 Kafka)实现消息的异步处理。

➤ 详细设计

- **辅助详细设计文档生成：**利用大模型（例如豆包），根据体系结构设计和用户界面设计，生成类图、时序图等详细设计文档，并提供代码实现的思路和建议。
- **示例 1：**输入“假设你是一个资深软件工程师，我正在开发一个在线购物网站，请你根据之前设计的系统架构，生成用户登录模块的类图和时序图，并提供代码实现的思路。”大模型可以生成包含用户类、认证服务类等类图，以及用户登录流程的时序图，并建议使用 Spring Security 框架实现用户认证功能。





- **示例 2：**输入“我正在开发一个基于微服务架构的社交网络应用，请你生成用户关注功能的 API 接口文档，并提供代码实现的示例。”大模型可以生成包含请求方法、URL、参数、返回值等信息的 API 接口文档，并提供使用 Spring Boot 和 RESTful API 实现用户关注功能的代码示例。

□3、编写代码

➤ **目标：将软件设计方案转化为可执行的代码。**

➤ **代码生成**

- **辅助代码生成：**利用大模型，根据设计文档和自然语言描述，生成代码片段或完整函数。
 - **示例 1：**输入“假设你是一个资深 Python 开发工程师，我正在开发一个在线购物网站，请你根据之前设计的用户登录模块，生成用户注册功能的 Python 代码。”大模型可以生成包含用户注册表单验证、密码加密、数据库操作等功能的代码片段。
 - **示例 2：**输入“我正在开发一个基于 React 的移动端应用，请你生成一个商品列表组件，并实现下拉刷新和上拉加载更多功能。”大模型可以生成包含商品列表展示、下拉刷新、上拉加载更多等功能的 React 组件代码。





➤ 代码适配

- 辅助代码适配：从开源网站、问答社区或其他代码资源中找到可复用的代码片段后，利用大模型将其修改为适配当前项目上下文的代码。
- 示例 1：输入“我从 GitHub 上找到了一段用于用户认证的 Python 代码，但它是基于 Flask 框架的，而我的项目使用的是 Django 框架。请你帮我将这段代码适配到 Django 框架中。”大模型可以将 Flask 的认证逻辑转换为 Django 的认证逻辑，并生成适配后的代码。

示例 2：输入“我从 Stack Overflow 上找到了一段用于处理文件上传的 JavaScript 代码，但它是基于原生 JavaScript 的，而我的项目使用的是 React。请你帮我将这段代码适配到 React 组件中。”大模型可以将原生 JavaScript 代码转换为 React 组件代码，并确保其与项目的状态管理和事件处理机制兼容。





➤ 代码优化

- 辅助代码优化：利用大模型，分析代码性能瓶颈，提供优化建议，并重构代码以提高代码可读性和可维护性。
- 示例 1：输入“假设你是一个性能优化专家，请你分析以下 Python 代码的性能瓶颈，并提供优化建议。”大模型可以分析代码性能瓶颈，例如循环嵌套过深、数据库查询次数过多等，并提供优化建议，例如使用缓存、优化算法等。
- 示例 2：输入“我正在开发一个大型软件项目，请你重构以下代码，提高代码的可读性和可维护性。”大模型可以重构代码，例如提取函数、消除重复代码、添加注释等，以提高代码的可读性和可维护性。



Peking
University



➤ 代码注释生成

- **辅助代码注释生成：**利用大模型自动生成代码注释，解释代码功能和逻辑。
- **示例1：**输入“请你为以下 Python 函数生成注释，解释函数的功能和参数。”大模型可以生成详细的代码注释，解释函数的功能、参数、返回值等信息。
- **示例 2：**输入“我正在开发一个开源项目，请你为以下 Java 类生成注释，解释类的功能和成员变量。”大模型可以生成详细的代码注释，解释类的功能、成员变量、方法等信息。





□4 软件测试

- **目标：**通过一系列的测试，尽可能地发现并修复程序代码中的缺陷，提高软件质量。
- **测试用例生成**
 - **辅助测试用例生成：**利用大模型，根据需求文档、设计文档或代码逻辑，自动生成测试用例。
 - **示例1：**输入“假设你是一个资深测试工程师，我正在开发一个在线购物网站，请你根据用户登录功能的需求文档，生成测试用例，包括正常登录、密码错误、用户名不存在等场景。”大模型可以生成详细的测试用例，包括测试步骤、预期结果、实际结果等。
 - **示例2：**输入“我正在开发一个基于微服务架构的社交网络应用，请你根据用户关注功能的 API 接口文档，生成测试用例，包括关注成功、关注失败、重复关注等场景。”大模型可以生成详细的测试用例，包括请求参数、预期响应、实际响应等。





➤ 测试脚本生成

- **辅助测试脚本生成：利用大模型，根据测试用例，自动生成测试脚本。**
- **示例1：**输入“假设你是一个资深测试工程师，请你根据以下用户登录功能的测试用例，生成 Python 的单元测试脚本。你可以使用 unittest 或 pytest 框架的单元测试脚本。”
- **示例2：**输入“我正在开发一个 Web 应用，请你根据以下商品搜索功能的测试用例，生成 Selenium 的自动化测试脚本。”
大模型可以生成使用 Selenium 的自动化测试脚本，模拟用户操作浏览器进行测试。





➤ 测试报告生成

- **辅助测试报告生成：**利用大模型，帮助分析测试结果，识别潜在问题并提供改进建议。例如，输入测试失败日志，模型可以分析失败原因并建议修复方法。自动生成测试报告。
- **示例：**输入“我正在开发一个移动端应用，请你根据以下自动化测试结果，生成测试报告，包括测试覆盖率、性能指标、用户体验评分等信息。并提供优化建议。”大模型可以生成全面的测试报告，帮助团队评估软件质量，并针对性的优化。





□ 5 运行维护

- **目标：**将软件部署到目标环境，确保其稳定运行，并根据用户反馈和运行数据持续改进软件。
- **部署与配置**
 - **脚本生成：**利用大模型，根据目标环境，自动生成部署脚本，提供目标环境的配置建议，例如服务器配置、网络配置、安全配置等。
 - **示例 1：**输入“假设你是一个 DevOps 工程师，我正在开发一个基于 Docker 的 Web 应用，请你生成一个 Docker Compose 文件，用于部署应用的前端、后端和数据库。”大模型可以生成包含服务定义、网络配置、环境变量等内容的 Docker Compose 文件。
 - **示例 2：**输入“假设你是一个系统管理员，我正在部署一个高并发的 Web 应用，请你提供服务器配置建议，包括 CPU、内存、磁盘等。”大模型可以根据应用的需求，提供服务器配置建议，例如使用多核 CPU、大内存、SSD 磁盘等。





➤ 故障排查与修复

- **故障排查与修复：**利用大模型，分析系统日志，识别故障原因，提供修复建议。
- **示例：**输入“假设你是一个故障排查专家，我正在排查一个 Web 应用的 500 错误，请你帮我分析以下日志文件，识别故障原因。”大模型可以分析日志文件，识别故障原因，例如数据库连接失败、代码逻辑错误等，并提供修复建议。



Peking
University



➤ 用户反馈分析与功能更新

- **辅助用户反馈分析：**利用大模型，分析用户反馈，识别软件缺陷和改进点。
- **示例：**输入“我正在分析一个在线教育平台的用户反馈，请你识别用户最常提到的问题和改进建议。”大模型可以分析用户反馈，识别常见问题，例如课程加载慢、界面不友好等，并提供改进建议。
- **辅助功能更新与迭代：**利用大模型，根据用户需求和市场趋势，提供功能更新建议，并生成功能更新文档。
- **示例：**输入“假设你是一个产品经理，我正在规划一个电商应用的下一版本，请你根据用户反馈和市场趋势，提供功能更新建议。”大模型可以提供功能更新建议，例如增加直播带货功能、优化推荐算法、生成功能更新文档，帮助团队明确开发目标和优先级。



Peking
University



如何应用LLMs工具辅助软件开发

➤ 在与大模型或其他专业人员交互时，如何组织问题、明确需求以及有效沟通是至关重要的。

➤ 1. 明确问题

- 先明确你要解决的核心问题是什么。避免模糊或过于宽泛的问题。
- 反例：如何开发好一个项目？
- 正例：在开发一个校园管理系统时，如何设计一个安全且高效的用户认证模块？

➤ 2. 交代背景和领域

- 角色预设：假设大模型是一个某某领域非常有经验的专家。示例：“假设你 是一位资深的后端开发专家，专注于高并发系统的设计与优化。”



Peking
University



- **项目背景:** 提供项目的背景信息，包括目标、用户群体、技术栈等。
 - **示例:** “我们正在开发一个在线考试系统，目标用户是大学生和教师，技术栈是 React 前端和 Spring Boot 后端。”
- **领域知识:** 如果问题涉及特定领域(如机器学习、区块链等)，简要说明相关背景知识。
 - **示例:** “我们正在开发一个基于机器学习的推荐系统，使用的是 Python 和 TensorFlow。”

➤ 3. 问题的组织与分解

- **问题分解:** 将复杂问题分解为多个子问题，step-by-step 逐步解决。
 - **示例:** 对于“如何设计一个在线考试系统？”可以分解为：“如何设计用户认证模块？如何实现考试创建和管理功能？如何确保系统的安全性和稳定性？”
- **优先级排序:** 根据问题的紧急程度或重要性排序，优先解决关键问题。
 - **示例:** 在开发初期，优先解决用户认证和考试管理功能，后期再优化性能和安全性。





➤ 4. 向大模型提问(Prompt设计)

- **结构化提问：**使用清晰、结构化的语言提问，确保问题易于理解。虽然它是机器，但是可以将它想象成一个专业人员，考虑它是否能够清晰的理解。
 - **示例：**“在开发一个在线考试系统时，如何设计一个支持高并发的用户认证模块？我们目前使用的是 Spring Security，但担心性能问题。”
- **提供上下文：**在提问时提供足够的上下文信息，避免对方需要猜测你的需求。如果可能，提供具体的示例或代码片段，帮助对方更好地理解问题。
 - **示例：**“我们正在开发一个网购平台，使用微服务架构，目前遇到的问题是订单服务的响应时间较慢。以下是我们当前的代码，看起来是循环嵌套太多，如何优化？”

➤ 5. 汇总回答后进一步提问

- **总结回答：**在获得回答后，先复述总结对方提供的信息，确保你理解正确。示例：“你建议我们使用缓存来优化订单服务的性能，具体可以使用 Redis，对吗？”



Peking
University



- **追问细节：**如果回答不够详细或存在疑问，可以进一步追问。
 - **示例：**“关于使用 Redis 缓存，你能提供一些具体的实现示例吗？”
- **建议验证：**如果对方提供了解决方案，但其超出了我们的知识范围，可以让大模型工具提供方案验证其可行性或请求更多参考资料。
 - **示例：**“你上面提到的微服务架构优化方案，是否有相关的案例研究或文档可以参考？”

➤ 6. 总结梳理

- **多轮对话过程梳理总结：**根据和大模型的沟通交互，不断地明确共识和分歧，多次迭代后，最终达成共识，将最终的方案总结梳理出来。
 - **示例：**在与大模型的多轮对话后，“经过与你的多反讨论，我们确定了以下解决方案：(1)使用 Redis 缓存优化订单服务的性能。(2)采用分布式锁解决数据一致性问题。(3)使用消息队列异步处理订单，进一步提高系统吞吐量。你看看还有什么其他的建议吗？没有问题我们就按照这个方案实施了。”

□ GitHub Copilot

- GitHub Copilot 是由 GitHub (微软) 与 OpenAI 联合推出的 AI 编码助手，内嵌于 VS Code 等主流 IDE
- 免费订阅计划限制每月 2000 次代码补全，完成 GitHub 教育认证可以获得无限制代码补全
- 在 VSCode 中，安装 GitHub Copilot 与 GitHub Copilot Chat 扩展，登录与授权后即可开始使用





□ GitHub Copilot

- 自动代码生成功能，Tab 补全

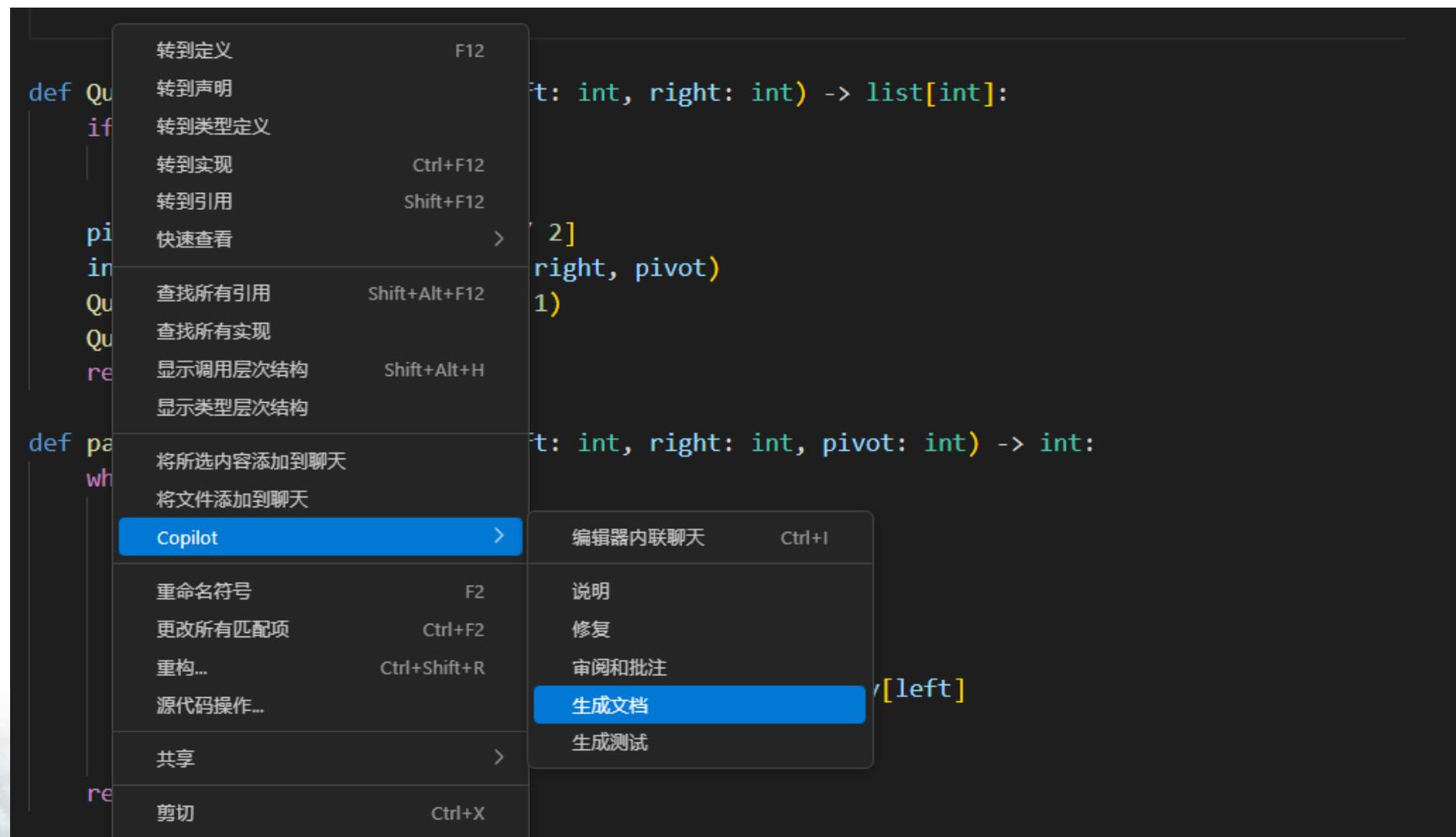
```
# 示例：开始编写一个排序函数
def QuickSort(array: list[int], left: int, right: int) -> list[int]:
    if left >= right:
        return array

    pivot = array[(left + right) // 2]
    index = partition(array, left, right, pivot)
    QuickSort(array, left, index - 1)
    QuickSort(array, index, right)
    return array
```



□ GitHub Copilot

➤ 自动生成文档





□ GitHub Copilot

➤ 自动生成文档

```
def QuickSort(array: list[int], left: int, right: int) -> list[int]:
    """
    This module implements the QuickSort algorithm for sorting a list of integers.
    Functions:
        QuickSort(array: list[int], left: int, right: int) -> list[int]:
            Recursively sorts the input list using the QuickSort algorithm.
        partition(array: list[int], left: int, right: int, pivot: int) -> int:
            Partitions the input list around a pivot value, ensuring that all elements
            less than the pivot are on the left and all elements greater than the pivot
            are on the right.
    Usage:
        - Call the `QuickSort` function with the list to be sorted and the indices of
          the range to sort (typically 0 and len(array) - 1).
        - The `partition` function is used internally by `QuickSort` to rearrange the
          elements around the pivot.
    """
    if left >= right:
        return array

    pivot = array[(left + right) // 2]
    index = partition(array, left, right, pivot)
    QuickSort(array, left, index - 1)
```



□ GitHub Copilot Chat

- Ask模式：用于一般性问题解答和代码解释
- Edit模式：专注于多文件代码编辑和重构
- Agent模式：可以自主完成复杂编码任务

The screenshot shows the GitHub Copilot Chat interface. On the left is a code editor with Python code for a QuickSort algorithm. On the right is a panel with analysis results.

Code Editor Content:

```
1
2
3
4
5
6 def QuickSort(array: list[int], left: int, right: int) -> list[int]:
7     """
8         This module implements the QuickSort algorithm for sorting a list of integers.
9     Functions:
10        QuickSort(array: list[int], left: int, right: int) -> list[int]:
11            Recursively sorts the input list using the QuickSort algorithm.
12            partition(array: list[int], left: int, right: int, pivot: int) -> int:
13                Partitions the input list around a pivot value, ensuring that all elements
14                less than the pivot are on the left and all elements greater than the pivot
15                are on the right.
16        Usage:
17            - Call the `QuickSort` function with the list to be sorted and the indices of
18            the range to sort (typically 0 and len(array) - 1).
19            - The `partition` function is used internally by `QuickSort` to rearrange the
20            elements around the pivot.
21        """
22
23    if left >= right:
24        return array
25
26    pivot = array[(left + right) // 2]
27    index = partition(array, left, right, pivot)
28    QuickSort(array, left, index - 1)
29    QuickSort(array, index, right)
30    return array
31
32 def partition(array: list[int], left: int, right: int, pivot: int) -> int:
33     while left <= right:
34         while array[left] < pivot:
35             left += 1
36         while array[right] > pivot:
37             right -= 1
38         if left >= right:
39             break
40         array[left], array[right] = array[right], array[left]
```

Analysis Panel:

- Explain the structure of this file.
- sort.py
- 已使用 1 个引用
- 该文件实现了快速排序 (QuickSort) 算法，结构如下：

 - QuickSort 函数：主排序函数，递归地对列表进行排序。参数包括待排序列表 `array`，排序范围的左、右索引 `left` 和 `right`。核心逻辑是选取中间元素为 pivot，通过 `partition` 函数分区后递归排序左右两部分。
 - partition 函数：辅助分区函数。根据 pivot 值，将列表分为左侧小于 pivot、右侧大于 pivot 的两部分，并返回新的分区点索引。
 - 文档字符串：在 `QuickSort` 函数顶部，详细说明了模块用途、函数功能和使用方法。

- 整体结构清晰，分为排序主逻辑和分区辅助逻辑，适合用于整数列表的快速排序。

□ Vibe Coding 氛围编程

- 将代码编写的主导权完全移交给AI
- 开发者通过自然语言指令“指挥”AI，观察和验证程序的运行结果，并再向AI提供反馈
- 其核心工具生态包括 GitHub Copilot、Cursor Chat 等IDE集成与编辑器插件
- 最理想的应用场景是快速原型开发。它能够迅速地将一个想法变为可交互的原型，实现快速迭代
- 但在生产环境中直接使用Vibe Coding是很危险的，其中潜在的漏洞、性能瓶颈等问题仍需人工评审



口其他流行的 LLM 工具

- Cursor: 无缝衔接 VSCode 的 IDE，在自动编辑、生成代码之外，运行用户配置外部文档作为知识库，以及加入内置的 System Prompt
- Trae: 与 Cursor 类似的 AI 原生 IDE，其主要特性为以 AI 为主导的多轮迭代式协作。AI 不仅响应指令，而且主动提问，与用户共同决策
- aiXcoder: 北大李戈团队开发的 AI 集成 IDE，更强的工程语义理解能力，能够理解整个项目的架构、类关系、调用链，深度支持大型复杂工程
- ChatGPT, DeepSeek, 豆包、Qwen 等通用 LLM：在调试复杂报错、设计软件系统架构、理解算法/数据结构原理、帮助学习新技术框架等方面具有优势。

大模型工具推荐

工具名称	特点	适用场景	优点
IDE 插件	Cursor <ul style="list-style-type: none">-深度集成到 IDE 中，可用于代码生成、代码适配、代码优化和注释生成等开发任务-通过实时提示和建议，优化开发体验	<ul style="list-style-type: none">-编码实现阶段：代码生成、适配、优化、注释生成等开发任务-运维与维护阶段：对现有代码进行优化。	<ul style="list-style-type: none">-集成 IDE，开发体验流畅，不需要从聊天界面来回复制粘贴
	aiXcoder <ul style="list-style-type: none">-中文交互能力优秀-支持多种编程语言	<ul style="list-style-type: none">-编码实现阶段：补全或生成代码片段、生成代码注释。-测试阶段：生成单元测试并做 Bug 修复。-维护阶段：对当前代码库进行代码解释	<ul style="list-style-type: none">-国产开源免费使用-集成 IDE，开发体验流畅
	GitHub Copilot/ Copilot <ul style="list-style-type: none">-支持代码生成、代码优化、注释生成以及跨文件和项目级代码理解任务。-基于 GitHub 仓库的强大支持，可提供项目上下文相关的高质量建议	<ul style="list-style-type: none">-编便码实现阶段：快速生成复杂代码逻辑维护阶段：对代码库进行理解和重构。	<ul style="list-style-type: none">-GitHub 网页版，跨文件、项目级程序理解能力强-集成 IDE 版，开发体验良好

大模型工具推荐

LLMs 工具	Claude	-理解能力强，可生成高质量文档内容	-需求分析阶段:生成用户故事、需求文档模板及澄清模糊需求。 -测试阶段:生成测试用例和测试报告。	-理解能力强，生成质量高，适合各种软件文档的生成，英文交互效果更好
	DeepSeek	-轻量级代码生成、文档生成功能，中文能力好。 -对文档处理和简单代码生成任务的支持。	-需求分析阶段：分析市场需求或生成项目创意 -文档整理阶段：生成需求文档或初步设计文档。	-中文支持好 -适合处理多文档和轻量级任务
	豆包	-轻量级代码生成、文档生成功能，中文能力好。 -对文档处理和简单代码生成任务的支持 -对图的生成支持比较好	-需求分析阶段：分析市场需求或生成项目创意 -设计阶段：辅助支持生成分析和设计模型 -文档整理阶段：生成需求文档或初步设计文档。	-中文支持好 -适合处理多文档和轻量级任务 -辅助支持生成分析和设计模型
	KiMi	-适用于文档总结处理、网页总结等需要联网搜索的任务。 -擅长提取和总结海量文本信息。	-需求分析阶段：生成项目背景调研报告或市场分析总结。 -文档处理阶段：快速总结用户反馈或需求说明	-对文档和网页总结效果好，适合综合信息处理任务 -中文支持出色，适合本地化需求



完

欢迎提问！



Peking
University