河北师范大学软件学院
Software College of Hebei Normal University

JPush实战

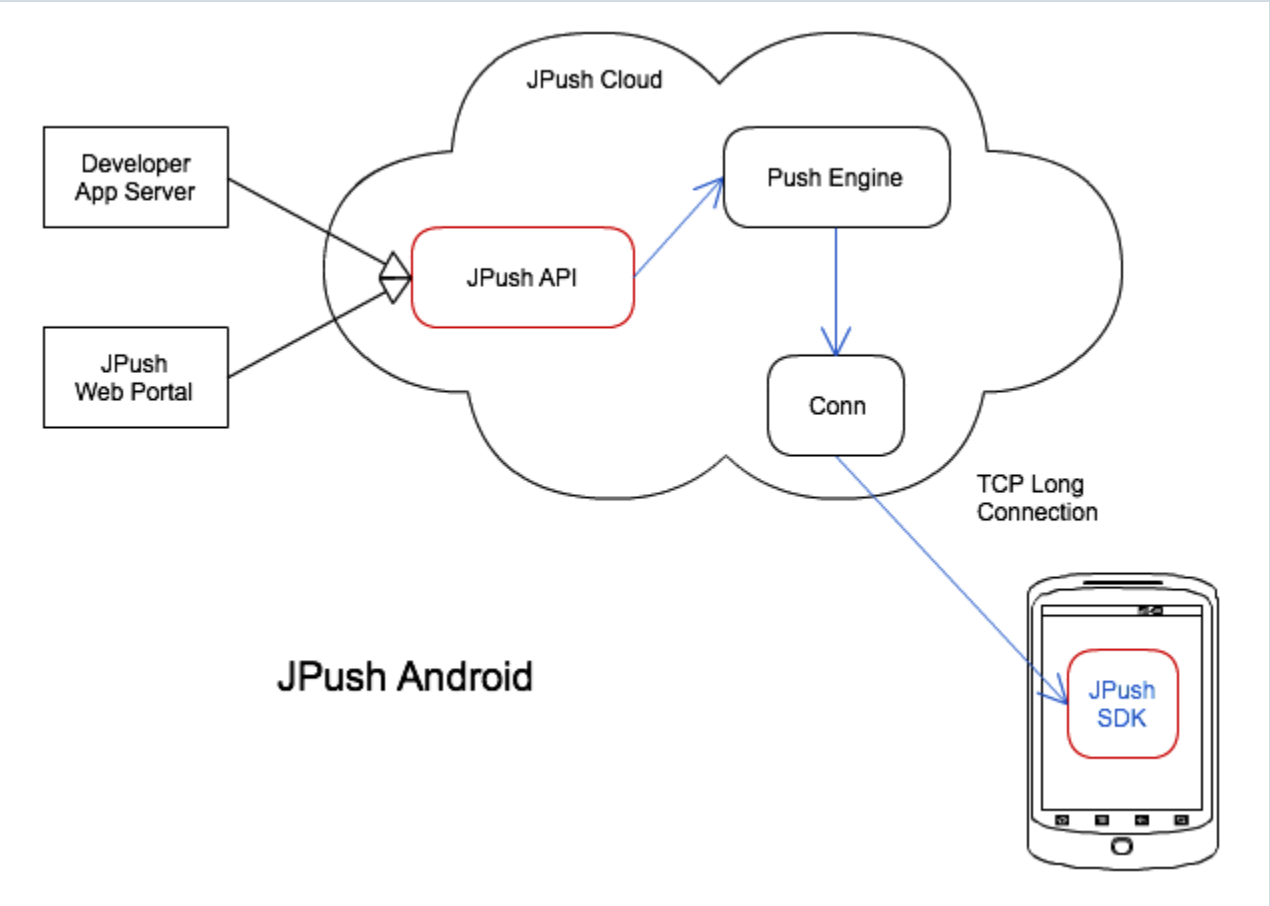
Android教研室

- JPush概述
- 集成JPush SDK
- 控制台推送消息
- 接收推送消息Receiver

JPush概述

- 极光推送（JPush）是一个端到端的推送服务，使得服务器端消息能够及时地推送到终端用户手机上。
- JPush是常用的APP推送平台， 开发者集成SDK后，可以通过调用API推送消息。同时，JPush还提供可视化的web端控制台发送通知，统计分析推送效果。
- 目前SDK只支持Android 2.3或以上版本的手机系统。

JPush概述


JPush Cloud
Developer
App Server
JPush
Web Portal
JPush API
Push Engine
Conn
TCP Long
Connection
JPush
SDK
JPush Android

JPush概述

- 主要功能
  - 保持与服务器的长连接，以便消息能够即时推送到达客户端。
  - 接收通知与自定义消息，并向开发者App 传递相关信息。
- 官方帮助文档
  - https://docs.jiguang.cn/jpush/client/Android/android_sdk/

- JPush概述
- 集成JPush SDK
- 控制台推送消息
- 接收推送消息Receiver

集成JPush SDK

- 下载Android SDK：https://docs.jiguang.cn/jpush/resources/
- 解压缩 jpush-android--3.x.x-release.zip 集成压缩包。
- 复制 libs/jcore-android-1.x.x.jar 到工程 libs/ 目录下。
- 复制 libs/jpush-android-3.x.x.jar 到工程 libs/ 目录下。
- 复制 libs/(cpu-type)/libjcore1xy.so 到你的工程中存放对应cpu类型的目录下。
- 复制 res/ 中drawable-hdpi, layout, values文件夹中的资源文件到你的工程中 res/ 对应同名的目录下。

集成JPush SDK

- 在 build.gradle 中设置so文件路径，在module的gradle配置中的android节点下添加以下配置：

```
sourceSets {
        main {
            jniLibs.srcDirs = ['libs']
        }
    }
```

```
1  apply plugin: 'com.android.application'
2
3  android {
4      sourceSets {
5          main {
6              jniLibs.srcDirs = ['libs']
7          }
8      }
```

集成JPush SDK

- 根据压缩包中示例文件，配置 AndroidManifest.xml
  - 复制备注为 “Required” 的部分。
  - 将标注为“您应用的包名”的部分，替换为当前应用程序的包名。
  - 将标注为“您应用的Appkey”的部分，替换为在Portal上注册该应用的Key。

集成JPush SDK

➢ 必须权限说明

权限	用途
JPUSH_MESSAGE	官方定义的权限，允许应用接收JPUSH内部代码发送的广播消息。
RECEIVE_USER_PRESENT	允许应用可以接收点亮屏幕或解锁广播。
INTERNET	允许应用可以访问网络。
WAKE_LOCK	允许应用在手机屏幕关闭后后台进程仍然运行
READ_PHONE_STATE	允许应用访问手机状态。
WRITE_EXTERNAL_STORAGE	允许应用写入外部存储。
READ_EXTERNAL_STORAGE	允许应用读取外部存储。
WRITE_SETTINGS	允许应用读写系统设置项。
VIBRATE	允许应用震动。
MOUNT_UNMOUNT_FILESYSTEMS	允许应用挂载/卸载 外部文件系统。
ACCESS_NETWORK_STATE	允许应用获取网络信息状态，如当前的网络连接是否有效。

创建极光推送开发者账号

- 极光推送开发者官网：https://www.jiguang.cn/push

创建应用

- 使用注册账号登陆，进入极光控制台后，点击“创建应用”按钮。填写应用程序的名称，完成应用创建。

创建应用

- 创建完成后即可看见AppKey，如果要使用推送业务，还需要先完成推送设置。（推送设置在后面进行设置的说明）

集成JPush SDK

- 初始化推送服务

  public static void init(Context context)

  - context 应用的 ApplicationContext

- 设置调试模式，该方法需在init方法之前调用，避免出现部分日志没打印的情况。多进程情况下建议在自定义的Application中onCreate中调用。

  public static void setDebugMode(boolean b)

集成JPush SDK

- 定制一个本应用程序 Application 类。需要在 AndoridManifest.xml 里配置。

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        JPushInterface.setDebugMode(true);
        JPushInterface.init(this);
    }
}
```

集成JPush SDK

- 定制一个本应用程序 Application 类。需要在 AndoridManifest.xml 里配置。

```
<application
    android:name=".MyApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name “
    ……>
    ……
</application>
```

- JPush概述
  - 集成JPush SDK
  - 控制台推送消息
  - 接收推送消息Receiver

创建应用

- 推送设置


应用信息
推送设置
IM 设置
短信设置
周报设置
推送设置
使用推送业务或 IM 业务需先完成推送设置
开发者认证
使用短信业务需先完成开发者认证

创建应用

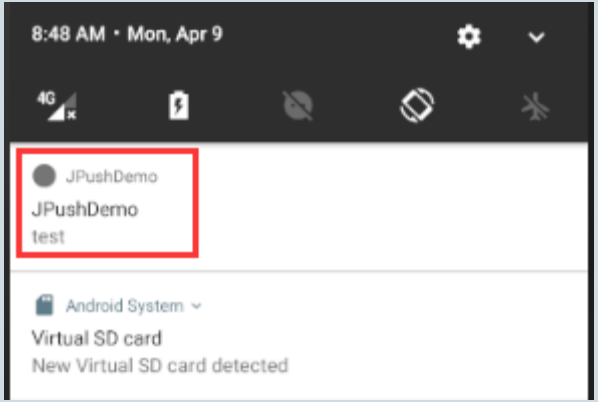
- 填写应用程序包名，点击保存按钮，完成推送设置。

发送通知

- 在控制台选择应用，点击推送，选择发送通知选项卡，可以在输入框中输入通知的内容，选择Android平台并发送。

接收通知

- 如果安装app的手机能够收到该条通知，即为配置成功。



推送消息的形式

- JPush提供四种消息形式：通知，自定义消息，富媒体和本地通知。

- 注意：富媒体信息流功能需Android3.0或以上版本的系统。

推送消息的形式

- 通知（Push Notification），指在手机的通知栏（状态栏）上会显示的一条通知信息。
- 通知主要用于提示用户。一条通知，简单的填写纯文本的通知内容即可。
- 应用加上通知功能，有利于提高应用的活跃度。

推送消息的形式

- 自定义消息主要用于应用的内部业务逻辑，可能没有任何界面。
- 自定义消息主要用于应用的内部业务逻辑和特殊展示需求。
- 图文并茂的通知，从而更好的传达信息，带来更丰富的用户互动。
- 本地通知API不依赖于网络，无网条件下依旧可以触发。

- JPush概述
  - 集成JPush SDK
  - 控制台推送消息
  - 接收推送消息Receiver

接收推送消息Receiver

- JPush SDK 收到推送，通过广播的方式，转发给开发者App，这样开发者就可以灵活地进行处理。
- 默认情况下当集成了JPush SDK的APP收到推送消息时：
  - 接收到推送的自定义消息，并没有被处理。
  - 可以正常收到通知，当用户点击通知时打开应用主界面。

接收推送消息Receiver

- 如果全部类型的广播都接收，则需要在AndroidManifest.xml 里注册receiver时，添加如下的action信息：
  - cn.jpush.android.intent.REGISTRATION：SDK 向 JPush Server 注册所得到的注册 ID
  - cn.jpush.android.intent.MESSAGE_RECEIVED：接收自定义消息 Push 。
  - cn.jpush.android.intent.NOTIFICATION_RECEIVED：接收推送通知
  - cn.jpush.android.intent.NOTIFICATION_OPENED：用户点击通知
  - cn.jpush.android.intent.CONNECTION：JPush服务的连接状态发生改变

接收推送消息Receiver

```
<receiver
    android:name="你的应用包名.MyReceiver"
    android:enabled="true"
    android:exported="false">
    <intent-filter>
        <action android:name="cn.jpush.android.intent.REGISTRATION" />
        <action android:name="cn.jpush.android.intent.MESSAGE_RECEIVED" />
        <action android:name="cn.jpush.android.intent.NOTIFICATION_RECEIVED" />
        <action android:name="cn.jpush.android.intent.NOTIFICATION_OPENED" />
        <action android:name="cn.jpush.android.intent.CONNECTION" />
        <category android:name="你的应用包名" />
    </intent-filter>
</receiver>
```

接收推送消息Receiver

- 可以通过定义 Receiver 类来处理程序接收到的广播。创建MyReceiver类，继承自BroadcastReceiver，并重写其onReceive方法。

```
public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        ……
    }
}
```

接收推送消息Receiver

```
public void onReceive(Context context, Intent intent) {
    if (JPushInterface.ACTION_MESSAGE_RECEIVED
                    .equals(intent.getAction())) {
        //接收到推送下来的自定义消息
    } else if (JPushInterface.ACTION_NOTIFICATION_RECEIVED
                    .equals(intent.getAction())) {
        //接收到推送下来的通知
    } else if (JPushInterface.ACTION_NOTIFICATION_OPENED
                    .equals(intent.getAction())) {
        //用户点击打开了通知
    }
}
```

详见帮助文档：https://docs.jiguang.cn/jpush/client/Android/android_api/

Thank You!