链滴

# Java 以太坊开发 Dapp（三）

作者：pencilso

原文链接：https://ld246.com/article/1529898371056

来源网站：链滴

# Java接入SDK

**毕竟都是一个体系的 其实Java Android API差别不大**

**Java Version 8+**

```
Maven
<dependency>
  <groupId>org.web3j</groupId>
  <artifactId>core</artifactId>
  <version>3.4.0</version>
</dependency>

Gradle
compile ('org.web3j:core:3.4.0')
```

# Android 接入SDK

```
Maven
<dependency>
  <groupId>org.web3j</groupId>
  <artifactId>core</artifactId>
  <version>3.3.1-android</version>
</dependency>

Gradle
compile ('org.web3j:core:3.3.1-android')

//引入RxAndroid
compile 'io.reactivex:rxandroid:1.2.1'
```

# 初始化项目

**无论是Java Android 都有一个Web3j的接口**

**先初始化这个接口**

```
//SERVICE_IP 这个是测试网地址  或者你的私有链地址
//私有链的话 一般带上端口号比如我的私有链是:http://127.0.0.1:8545
//如果是从infura.io注册的测试网的话  不需要加端口号 例如 https://ropsten.infura.io/your-token

Java
Web3j web3j = Web3j.build(new HttpService(SERVICE_IP));

Android
Web3j web3j = JsonRpc2_0Web3j(new HttpService(SERVICE_IP));
```

# 同步调用API

**这里举例获取ETH 客户端版本API**

**同步调用的话 Java Android都一样**

```
//两种方式 都是同步 具体细节没有去研究过
Web3ClientVersion web3ClientVersion = web3j.web3ClientVersion().send();
Web3ClientVersion web3ClientVersion = web3j.web3ClientVersion().sendAsync().get();

String version = web3ClientVersion.getWeb3ClientVersion();
```

# 异步调用

得益于RxJava 让我们异步调用更简洁 更方便

因为Android的特性 子线程不可更改UI 所以 Android方面 有点不一样

```
Java:
 web3j.web3ClientVersion()
         .observable()
         .observeOn(Schedulers.io())
         .subscribe(new Subscriber<Web3ClientVersion>() {
             @Override
             public void onStart() {
                 super.onStart();
                 System.out.println("onStart:" + Thread.currentThread());
             }

             @Override
             public void onCompleted() {
                 System.out.println("onCompleted:" + Thread.currentThread());
             }

             @Override
             public void onError(Throwable e) {
                 e.printStackTrace();
                 System.out.println("onError:" + Thread.currentThread());
             }

             @Override
             public void onNext(Web3ClientVersion web3ClientVersion) {
                 System.out.println("onNext:" + Thread.currentThread());
                 System.out.println("web3ClientVersion:" + web3ClientVersion.getWeb3ClientVers
on());
             }
         });
    //最后输出
    onStart:Thread[main,5,main]    //主线程
    onNext:Thread[RxIoScheduler-2,5,main]    //IO线程
    web3ClientVersion:Geth/v1.8.3-stable/linux-amd64/go1.10    //ETH客户端版本
    onCompleted:Thread[RxIoScheduler-2,5,main]    //IO线程

Android:
```

```
web3j.web3ClientVersion()
        .observable()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<Web3ClientVersion>() {
            @Override
            public void onStart() {
                super.onStart();
                System.out.println("onStart:" + Thread.currentThread());
            }

            @Override
            public void onCompleted() {
                System.out.println("onCompleted:" + Thread.currentThread());
            }

            @Override
            public void onError(Throwable e) {
                e.printStackTrace();
                System.out.println("onError:" + Thread.currentThread());
            }

            @Override
            public void onNext(Web3ClientVersion web3ClientVersion) {
                System.out.println("onNext:" + Thread.currentThread());
                System.out.println("web3ClientVersion:" + web3ClientVersion.getWeb3ClientVers
on());
            }
        });

//subscribeOn(Schedulers.io()) 用IO线程执行耗时操作
//observeOn(AndroidSchedulers.mainThread()) 在观察者回调的时候 使用Android 主线程回调
```

# 获取账户余额

```
//address 账户地址
EthGetBalance send = web3j.ethGetBalance(address, DefaultBlockParameterName.LATEST).se
d();
// 默认获取到的单位是WEI  转换为ETH
BigDecimal balance = Convert.fromWei(send.getBalance().toString(), Convert.Unit.ETHER);
```

# 区块链高度

```
 BigInteger blockNumber = web3j.ethBlockNumber().send().getBlockNumber();
```

# 区块详情

```
 DefaultBlockParameterNumber defaultBlockParameterNumber = new DefaultBlockParamete
Number(blockNumber);
EthBlock send = web3j.ethGetBlockByNumber(defaultBlockParameterNumber, true).send();
```

```
EthBlock.Block block = send.getBlock();
```

# 创建以太坊地址

```
String newWalletFile = WalletUtils.generateNewWalletFile(password, walletFolder, false);
//password 钱包文件密码
//walletFolder 钱包文件存储目录
//false 不进行加密  （如果加密  内存开销很大）
```

# 加载凭证

凭证是指 一个账户地址的凭证
可以通过私钥直接加载 或者通过密码 和私钥文件加载 （私钥文件是指keystore目录下的加密文件）

私钥加载凭证

```
 Credentials credentials = Credentials.create(privateKey);
```

私钥文件加载凭证

```
//password 密码  filePath私钥路径
Credentials credentials = WalletUtils.loadCredentials(password, filePath);
```

# 通过凭证 取出地址和私钥

```
ECKeyPair ecKeyPair = credentials.getEcKeyPair();
String address = "0x" + ecKeyPair.getPublicKey().toString(16);
String privateKey = "0x"+ecKeyPair.getPrivateKey().toString(16);

//address 以太坊地址
//privateKey 私钥
```

其他的API 自己摸索吧