1. 下载Ethereum-Wallet钱包

<https://github.com/ethereum/mist/releases/tag/v0.11.1>

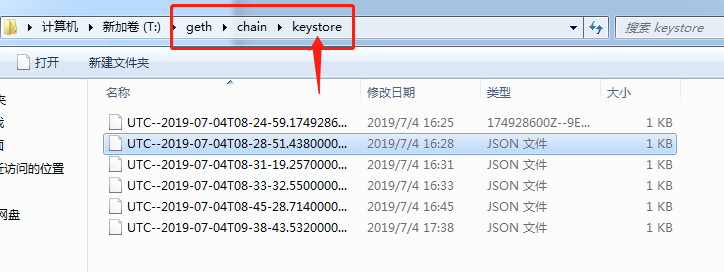
1. 下载Geth客户端

<https://geth.ethereum.org/downloads/>

1. 运行Geth客户端（注意方式）
2. geth --identity "mainnet etherum" --datadir "./chain" --rpc --rpcapi "admin,web3,eth,personal,miner,net" --rpcaddr 0.0.0.0 --rpcport 8545 --cache=1024 console
3. 参数解释

<https://www.jianshu.com/p/e2630bc1d510>

1. 运行Ethereum-Wallet
2. 如果一直卡在创建中应该就是Geth没运行起来
3. 创建钱包后会存储在Geth中的keystore目录下



1. 后台使用web3j调用接口
2. Demo
   1. 导入jar

<dependency>  
 <groupId>org.web3j</groupId>  
 <artifactId>core</artifactId>  
 <version>4.2.0</version>  
</dependency>

<!-- 或者下面，稍微不同就是解析钱包文件时多了个参数 -->

<dependency>  
 <groupId>org.web3j</groupId>  
 <artifactId>web3j-spring-boot-starter</artifactId>  
 <version>1.6.0</version>  
</dependency>

* 1. 代码

import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.test.context.junit4.SpringRunner;  
import org.web3j.crypto.Credentials;  
import org.web3j.crypto.RawTransaction;  
import org.web3j.crypto.TransactionEncoder;  
import org.web3j.crypto.WalletUtils;  
import org.web3j.protocol.Web3j;  
import org.web3j.protocol.core.DefaultBlockParameterName;  
import org.web3j.protocol.core.Response;  
import org.web3j.protocol.core.methods.response.EthSendTransaction;  
import org.web3j.protocol.http.HttpService;  
import org.web3j.utils.Numeric;  
  
import java.io.File;  
import java.math.BigInteger;  
import java.util.Objects;  
  
*/\*\*  
 \* <p>  
 \* 以太坊钱包接口调用  
 \* </p>  
 \*  
 \** ***@author*** *郭海斌  
 \** ***@since*** *2019/7/4  
 \*/*@RunWith(SpringRunner.class)  
@SpringBootTest  
public class EthTest {  
  
 @Value("${ethereum.url}")  
 private String url;  
 @Value("${ethereum.password}")  
 private String password;  
 @Value("${ethereum.directory}")  
 private String directory;  
  
 @Test  
 public void test() throws Exception {  
 Web3j web3j = connected(url);  
 if (Objects.*isNull*(web3j)) {  
 System.*out*.println(" 连接失败...");  
 return;  
 }  
 //地址都是0x开头，拼接上文件中的address  
 ethGetBalance(web3j, "0x93ee6ceca353c50cf7ae058d21077edc579d2a2a", DefaultBlockParameterName.*EARLIEST*);  
 generateWalletAddress(password, directory);  
 trade(web3j, password,  
 "UTC--2019-07-04T08-31-19.257000000Z--93ee6ceca353c50cf7ae058d21077edc579d2a2a.json",  
 "0x93ee6ceca353c50cf7ae058d21077edc579d2a2a",  
 BigInteger.*valueOf*(666));  
 }  
  
 */\*\*  
 \* 交易  
 \** ***@param*** *web3j  
 \* 相当于用一个钱包存了所有的ETH（不安全可能）  
 \** ***@param*** *password 文件密码 - 个人拥有  
 \** ***@param*** *fileName 钱包文件 - 个人拥有  
 \** ***@param*** *to 收款方 - 外部传递  
 \** ***@param*** *value 交易数量  
 \** ***@throws*** *Exception  
 \*/* public void trade(Web3j web3j, String password, String fileName, String to, BigInteger value) throws Exception {  
 System.*out*.println("开始交易...");  
 //转账金额不能为0  
 if (value.intValue() < 0) {  
 System.*out*.println(" 转账金额不能小于或等于0！");  
 return;  
 }  
 //加载此钱包文件  
 Credentials credentials = WalletUtils.*loadCredentials*(password, directory + fileName);  
 //获取该钱包余额  
 BigInteger nonce = web3j.ethGetTransactionCount(credentials.getAddress(), DefaultBlockParameterName.*LATEST*).send().getTransactionCount();  
 if (nonce.compareTo(value) < 0) {  
 System.*out*.println(" 余额不足，无法交易！");  
 return;  
 }  
 // create our transaction - 创建交易  
 RawTransaction rawTransaction = RawTransaction.*createEtherTransaction*(nonce, BigInteger.*ONE*, BigInteger.*ONE*, to, value);  
 // sign & send our transaction - 签名和部署？交易  
 byte[] signedMessage = TransactionEncoder.*signMessage*(rawTransaction, credentials);  
 String hexValue = Numeric.*toHexString*(signedMessage);  
 //创建交易  
 EthSendTransaction send = web3j.ethSendRawTransaction(hexValue).send();  
 Response.Error error = send.getError();  
 //error不为null说明交易失败  
 if (!Objects.*isNull*(error)) {  
 System.*out*.println(" 交易失败，" + error.getMessage());  
 } else {  
 System.*out*.println(" 交易成功！");  
 }  
 System.*out*.println("交易结束...");  
 }  
  
 //返回钱包地址，Ethereum-Wallet钱包其地址是无法改变的  
 public void returnWalletAddress() {  
 System.*out*.println("此处yml中配置一个钱包地址即可...");  
 }  
  
 */\*\*  
 \* 生成钱包（这是生成钱包，不是钱包地址，钱包地址是固定的）  
 \** ***@param*** *password 密码  
 \** ***@param*** *directory 钱包文件目录  
 \** ***@throws*** *Exception  
 \*/* public void generateWalletAddress(String password, String directory) throws Exception {  
 //远程操作时，需要使用一个ftp或者其他远程创建/读取文件的东西  
 System.*out*.println("生成钱包地址...");  
 String newWalletFile = WalletUtils.*generateNewWalletFile*(password, new File(directory), false);  
 Credentials loadCredentials = WalletUtils.*loadCredentials*(password, directory + newWalletFile);  
 String address = loadCredentials.getAddress();  
 System.*out*.println(" 钱包地址：" + address);  
 }  
  
 */\*\*  
 \* 获取某钱包余额（钱包余额）  
 \** ***@param*** *web3j  
 \** ***@param*** *address 钱包地址  
 \** ***@param*** *defaultBlockParameterName  
 \** ***@throws*** *Exception  
 \*/* public void ethGetBalance(Web3j web3j, String address, DefaultBlockParameterName defaultBlockParameterName) throws Exception {  
 System.*out*.println("获取钱包余额...");  
 BigInteger balance = web3j.ethGetBalance(address, defaultBlockParameterName).send().getBalance();  
 System.*out*.println(" 余额：" + balance);  
 }  
  
 //连接geth节点  
 public Web3j connected(String url) {  
 return Web3j.*build*(new HttpService(url));  
 }  
  
}

* 1. Yml

ethereum:  
 *# 节点url - geth节点  
# url: http://192.168.1.15:8545/* url: http://192.168.1.14:8545/  
 *# 密码 - 钱包密码* password: 123456789  
 *# 钱包文件存放目录 - 钱包文件存放目录* directory: T:\geth\chain\keystore\  
*# directory: E:\Geth\chain\keystore\*

1. 总结
2. 又搞了两天，以为很简单，结果又陷入坑里面了，但是总算搞完，下面小结
3. 如果是调客户端的API，首先把客户端玩一下
4. 玩大概知道什么个形式后，再开始找如何通过代码的方式去实现
5. 不要一上来就直接代码软件一起搞，特别费时间
6. 不过终究算搞懂了，哎，继续努力了，加油！