从零开始使用JDChain SDK

# 前言

有读者反映，通过jdchain-starter项目中的SDKTest.java中的样例来执行，虽然能快速得到结果，但是对其中的前因后果不太清楚，希望能够详细介绍一下SDK的使用。

我们考虑到其他用户可能也有此类困扰，特此编写此文来详细解答这个问题。

针对SDK的操作，我已经制作成视频，下载地址：<https://3.cn/u7KQMus>

为了方便大家使用，我把如下的样例也做成了压缩包：

# SDK使用详解

## 环境准备

1. 采用eclipse或者Idea作为开发环境；
2. 添加必要的maven依赖，如下：

<dependency>

<groupId>com.jd.blockchain</groupId>

<artifactId>sdk-pack</artifactId>

<version>1.0.1.RELEASE</version>

</dependency>

## 数据快速上链

### 服务连接

//创建服务代理

// 生成连接网关的账号

PrivKey privKey = KeyGenCommand.decodePrivKeyWithRawPassword(strPrivKey, passwd);

PubKey pubKey = KeyGenCommand.decodePubKey(strPubKey);

adminKey = new BlockchainKeypair(pubKey, privKey);

final String GATEWAY\_IP = "192.168.151.39";

final int GATEWAY\_PORT = 18081;

final boolean SECURE = false;

GatewayServiceFactory serviceFactory = GatewayServiceFactory.connect(GATEWAY\_IP, GATEWAY\_PORT, SECURE,

adminKey);

// 创建服务代理；

service = serviceFactory.getBlockchainService();

HashDigest[] ledgerHashs = service.getLedgerHashs();

// 获取当前账本Hash

ledgerHash = ledgerHashs[0];

System.out.println("ledgerHash="+ledgerHash.toBase58());

### 用户注册

// 在本地定义注册账号的 TX；

TransactionTemplate txTemp = service.newTransaction(ledgerHash);

//采用KeyGenerator来生成BlockchainKeypair;

BlockchainKeypair user = BlockchainKeyGenerator.getInstance().generate();

System.out.println("new user="+user.getAddress());

txTemp.users().register(user.getIdentity());

// TX 准备就绪；

PreparedTransaction prepTx = txTemp.prepare();

prepTx.sign(adminKey);

// 提交交易；

prepTx.commit();

### 数据账户注册

// 在本地定义注册账号的 TX；

TransactionTemplate txTemp = service.newTransaction(ledgerHash);

//采用KeyGenerator来生成BlockchainKeypair;

BlockchainKeypair dataAccount = BlockchainKeyGenerator.getInstance().generate();

txTemp.dataAccounts().register(dataAccount.getIdentity());

txTemp.dataAccount(dataAccount.getAddress()).setText("key1","value1",-1);

//add some data for retrieve;

this.strDataAccount = dataAccount.getAddress().toBase58();

System.out.println("current dataAccount="+dataAccount.getAddress());

// TX 准备就绪

PreparedTransaction prepTx = txTemp.prepare();

prepTx.sign(adminKey);

// 提交交易；

prepTx.commit();

### 数据账户写入内容

//历史数据账户地址;

this.strDataAccount = "LdeNxNjCiXvSpNADQKW1wyC6bRHHEoGhj1ufq";

TransactionTemplate txTemp = service.newTransaction(ledgerHash);

//add some data for retrieve;

System.out.println("current dataAccount="+this.strDataAccount);

txTemp.dataAccount(this.strDataAccount).setText("cc-fin01-01",

"{\"dest\":\"KA001\",\"id\":\"cc-fin01-01\",\"items\":\"FIN001|5000\",\"source\":\"FIN001\"}",-1);

txTemp.dataAccount(this.strDataAccount).setText("cc-fin02-01",

"{\"dest\":\"KA001\",\"id\":\"cc-fin02-01\",\"items\":\"FIN002|2000\",\"source\":\"FIN002\"}",-1);

// TX 准备就绪

PreparedTransaction prepTx = txTemp.prepare();

prepTx.sign(adminKey);

// 提交交易；

prepTx.commit();