JAVA与合约交互

使用库依赖（Maven）

```
<dependency>
    <groupId>org.web3j</groupId>
    <artifactId>core</artifactId>
    <version>4.9.8</version>
</dependency>
```

下载web3js工具命令（window版本）

```
Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).Dow
nloadString('https://raw.githubusercontent.com/web3j/web3j-installer/master/installer.ps1'))
```

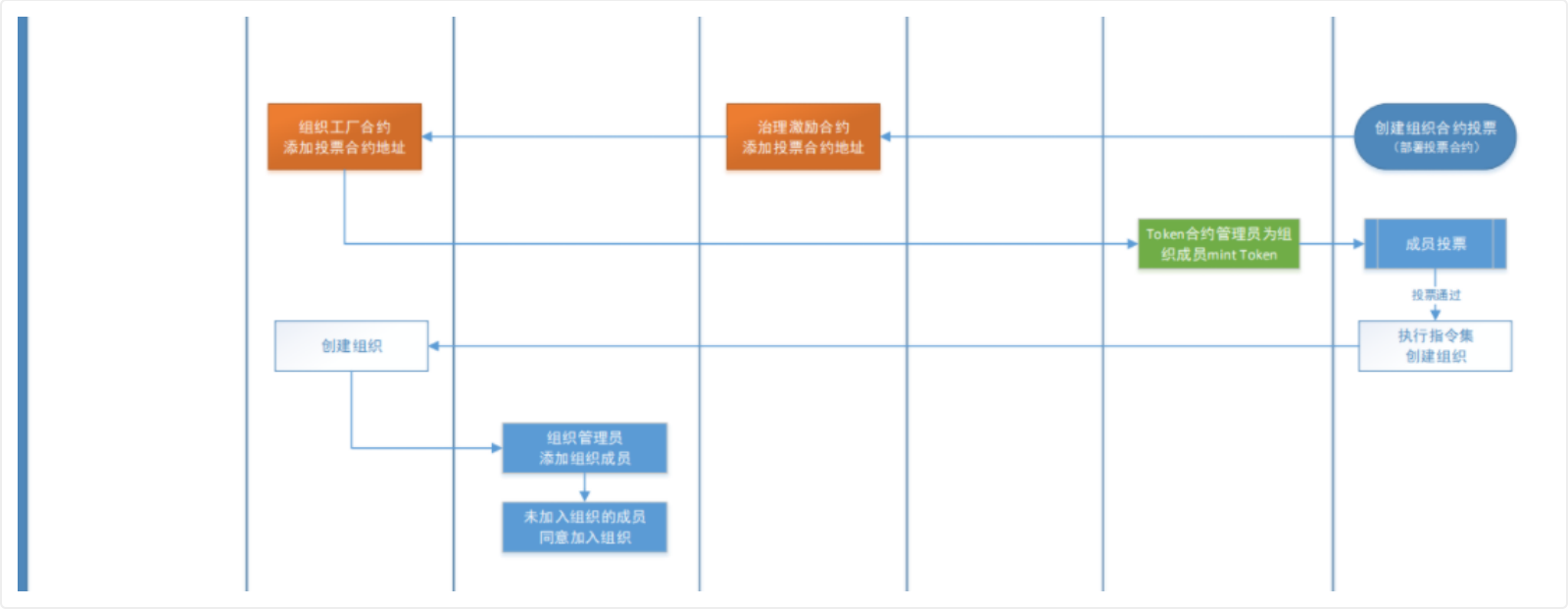
Web3J参考文档

```
https://docs.web3j.io/4.9.8/
```

根据abi文件 与 bin文件 生成 .java

```
web3j generate solidity -b BallotFactory.bin -a BallotFactory.abi -o . -p com.example.dfh.dt
o.contracts
```

交互例子（以创建组织投票为例子）



这里一共有三个步骤：

1. 监听投票事件
2. 若有事件进入，触发操作
    1. 治理激励合约添加投票合约地址
    2. 组织工厂合约添加投票合约地址

代码实现思路：

- 在构造函数中，创建WebSocketService对象，并使用它来初始化Web3j对象。这里使用了以太坊的Alchemy API连接以太坊网络。
- 接着，在listen方法中，通过EventEncoder.encode()方法将CREATEBALLOT_EVENT事件的签名转换成字符串形式，并创建一个EthFilter对象，过滤器用于只监听与该事件相关的区块数据。
- 然后，利用web3j.ethLogFlowable()方法，订阅符合过滤器条件的区块数据，并在回调函数中对接收到的区块数据进行处理。
- 当监听到CREATEBALLOT_EVENT事件时，通过contract.getCreateBallotEvents()方法解析出事件参数，包括新建投票合约的ID、地址和发送者等信息。
- 借助addVotingContractToExcitation和addVotingContractToOrganizationFactory两个方法，分别将新建投票合约的地址添加到治理激励合约和组织工厂合约中。

实现代码

```
package com.example.dfh.listen;

import com.example.dfh.constants.ContractConstants;
import com.example.dfh.dto.contracts.BallotFactory;
```

```
import com.example.dfh.dto.contracts.Excitation;
import com.example.dfh.dto.contracts.OrganizationFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Component;
import org.web3j.abi.EventEncoder;
import org.web3j.crypto.Credentials;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.DefaultBlockParameterName;
import org.web3j.protocol.core.methods.request.EthFilter;
import org.web3j.protocol.core.methods.response.TransactionReceipt;
import org.web3j.protocol.websocket.WebSocketService;
import org.web3j.tx.RawTransactionManager;
import org.web3j.tx.TransactionManager;
import org.web3j.tx.gas.DefaultGasProvider;

import java.io.IOException;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;


@Component
public class BallotFactoryEventListener {

    private final Web3j web3j;
    private final BallotFactory contract;

    private static final int THREAD_POOL_SIZE = 10;
    private ExecutorService executor = Executors.newFixedThreadPool(THREAD_POOL_SIZE);

    @Autowired
    public BallotFactoryEventListener() throws IOException {
        String endpoint = "wss://polygon-mumbai.g.alchemy.com/v2/" + ContractConstants.ALCHE
MY_API_KEY;
        WebSocketService webSocketService = new WebSocketService(endpoint, true);
        webSocketService.connect();
        this.web3j = Web3j.build(webSocketService);
//        this.contract = BallotFactory.load(ContractConstants.BALLOT_FACTORY_ADDR, web3j, C
redentials.create(ContractConstants.MUMBAI_PRIVATE_KEY), new DefaultGasProvider());
        Credentials credentials = Credentials.create(ContractConstants.MUMBAI_PRIVATE_KEY);
        TransactionManager transactionManager = new RawTransactionManager(web3j, credential
s, 80001);
        this.contract = BallotFactory.load(ContractConstants.BALLOT_FACTORY_ADDR, web3j, tra
nsactionManager, new DefaultGasProvider());
    }

    @Async
    public void listen() throws Exception {
```

```
        System.out.println("BallotFactoryEventListener.listen");
        String ballotCreatedEventSignature = EventEncoder.encode(contract.CREATEBALLOT_EVEN
T);
        EthFilter filter = new EthFilter(DefaultBlockParameterName.LATEST, DefaultBlockParam
eterName.LATEST, contract.getContractAddress())
                .addSingleTopic(ballotCreatedEventSignature);

        web3j.ethLogFlowable(filter).subscribe(log -> {
            List<String> topics = log.getTopics();
            if (topics.size() > 0 && topics.get(0).equals(ballotCreatedEventSignature)) {
                BallotFactory.CreateBallotEventResponse response = contract.getCreateBallotE
vents(log).get(0);
                String id = response._id;
                String addr = response._ballotAddr;
                String sender = response._sender;
                // 在这里添加处理逻辑
                System.out.println("Ballot created. ID: " + id + ", address: " + addr.toStri
ng() + ", sender: " + sender.toString());

                // 使用线程池异步执行这个任务
//                executor.execute(() -> {
//                    try {
//                        addVotingContractToExcitation(addr.toString());
//                        addVotingContractToOrganizationFactory(addr.toString());
//                    } catch (Exception e) {
//                        e.printStackTrace();
//                    }
//                });
                try {
                    addVotingContractToExcitation(addr.toString());
                    addVotingContractToOrganizationFactory(addr.toString());
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * 治理激励合约添加投票合约地址
     * @param votingContractAddress
     * @throws Exception
     */
    public void addVotingContractToExcitation(String votingContractAddress) throws Exception
{
        Credentials credentials = Credentials.create(ContractConstants.MUMBAI_PRIVATE_KEY);
        TransactionManager transactionManager = new RawTransactionManager(web3j, credential
s, 80001);
        Excitation contract = Excitation.load(ContractConstants.EXCITATION_ADDR, web3j, tran
```

```
sactionManager, new DefaultGasProvider());
        // 调用激励合约的addVotingContract方法
        TransactionReceipt result = contract.setBallotAddr(votingContractAddress).send();
        if(result.isStatusOK()){
            System.out.println("Voting contract address added to Excitation contract.");
        } else {
            System.out.println("Failed to add voting contract address to Excitation contrac
t.");
        }
    }

    /**
     * 组织工厂合约添加投票合约地址
     * @param votingContractAddress
     * @throws Exception
     */
    public void addVotingContractToOrganizationFactory(String votingContractAddress) throws
Exception {
        Credentials credentials = Credentials.create(ContractConstants.MUMBAI_PRIVATE_KEY);
        TransactionManager transactionManager = new RawTransactionManager(web3j, credential
s, 80001);
        OrganizationFactory contract = OrganizationFactory.load(ContractConstants.ORG_FACTOR
Y_ADDR, web3j,transactionManager, new DefaultGasProvider());
        // 调用组织工厂合约的addVotingContract方法
        TransactionReceipt result = contract.setBallotAddr(votingContractAddress).send();
        if(result.isStatusOK() && contract.ballotAddrs(votingContractAddress).send()) {
            System.out.println("Voting contract address added to OrganizationFactory contrac
t.");
        } else {
            System.out.println("Failed to add voting contract address to OrganizationFactory
contract.");
        }
    }

}
```

优化点

在listen方法中，当接收到CREATEBALLOT_EVENT事件时，程序会调用addVotingContractToExcitation和addVotingContractToOrganizationFactory方法来向治理激励合约和组织工厂合约中添加新建的投票合约地址。因为这两个方法没有执行的先后顺序，并且若同步执行也比较耗时，为了避免这种情况，我们可以将这两个方法放入一个线程池中异步执行，以避免阻塞主线程的执行。（参考代码注释部分）