WeBank

WeBASE实践

2019年9月

课程目的

- 了解WeBASE的组件和各个组件的作用
- 掌握WeBASE的搭建
- 掌握WeBASE基本功能、原理和运用
- 掌握使用WeBASE进行智能合约开发
- 掌握使用WeBASE进行FISCO BCOS系统管理

课程目录

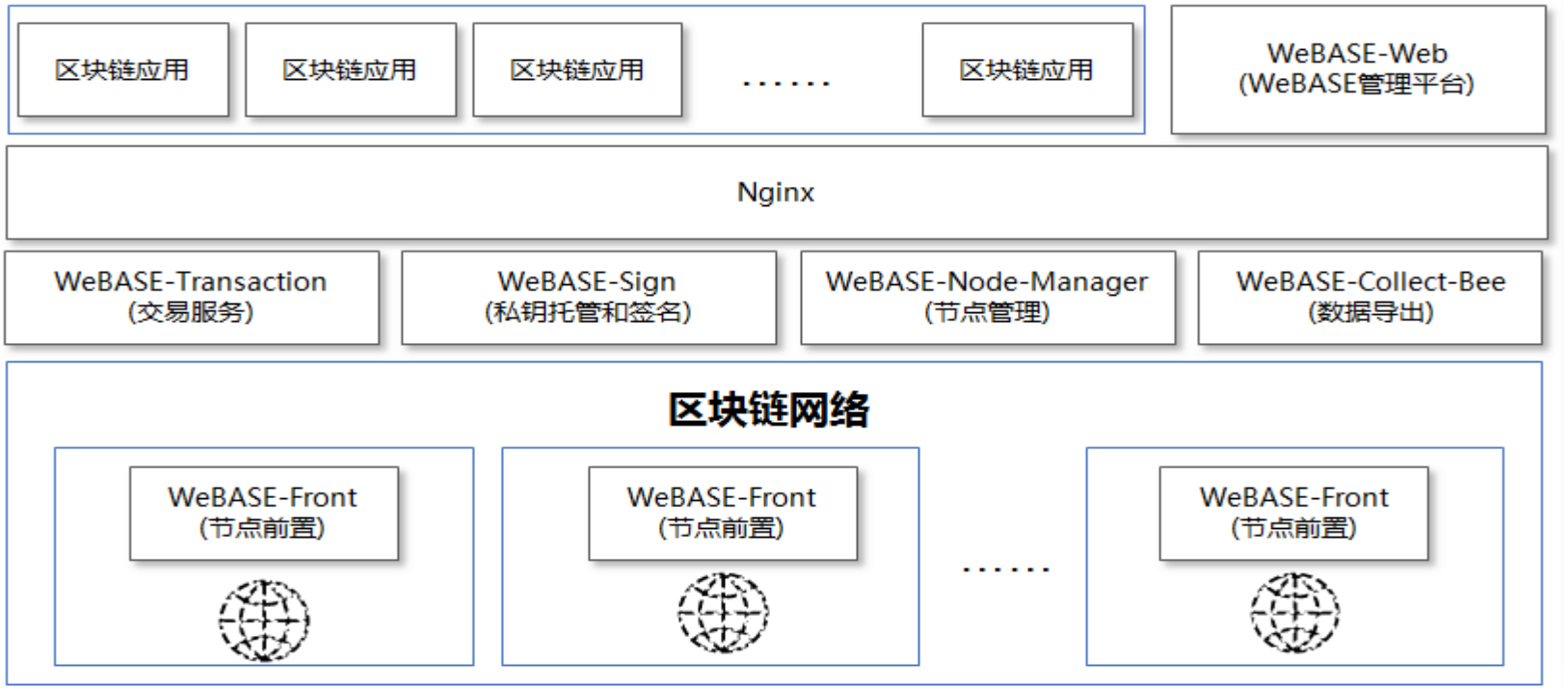
1. WeBASE是什么？
2. WeBASE环境搭建
   - 快速入门搭建
   - 一键部署
3. 学习WeBASE基本功能、原理和运用
   - 合约管理
   - 私钥管理
   - 交易审计
   - 交易立体展示
4. 学习使用WeBASE进行智能合约开发
   - Asset合约
   - 存证合约
5. 学习使用WeBASE进行FISCO BCOS系统管理
   - 节点管理
   - 配置管理
   - CRUD

01

WeBASE是什么？

整体架构

WeBASE（WeBank BlockChain Application Software Extension）是在区块链应用和FISCO-BCOS节点之间搭建的一套通用组件。



功能介绍

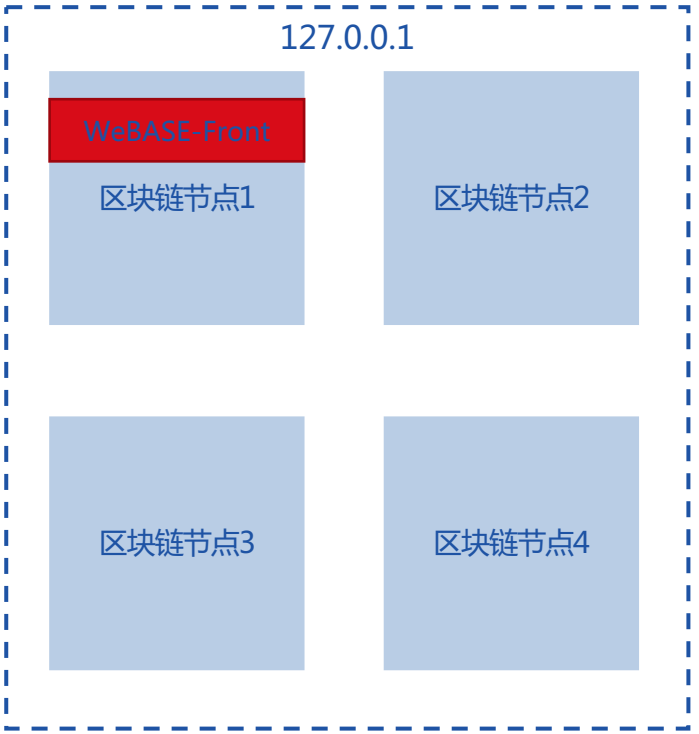
从可视化，智能合约，交易，数据四个维度设计各个中间件，各模块主要功能如下：

02

WeBASE环境搭建

快速入门搭建：

- 快速入门只需要搭建节点和节点前置服务(WeBASE-Front)，就可通过WeBASE-Front的合约编辑器进行合约的编辑，编译，部署，调试。
- 参考链接：快速入门搭建



快速入门搭建： FISCO-BCOS搭建实操：

- 参考文档
  - https://fisco-bcos-documentation.readthedocs.io/zh_CN/release-2.0/docs/installation.html
- 环境准备
  - 建议使用CentOS 7和Ubuntu 16.04 64bit
  - sudo yum install -y openssl curl（若为CentOS，将下面命令中的yum替换为apt执行即可。macOS执行brew install openssl curl即可。）
  - cd ~ && mkdir -p fisco && cd fisco
  - curl -LO https://github.com/FISCO-BCOS/FISCO-BCOS/releases/download/`curl -s https://api.github.com/repos/FISCO-BCOS/FISCO-BCOS/releases | grep "\"v2\.[0-9]\.[0-9]\"" | sort -u | tail -n 1 | cut -d \" -f 4`/build_chain.sh && chmod u+x build_chain.sh

快速入门搭建： FISCO-BCOS搭建实操：

- 搭建区块链系统
  - 在单台服务器上，搭建四个节点的区块链系统
    - ./build_chain.sh –l "127.0.0.1:4" -p 30300,20200,8545
- 启动区块链节点
  - cd nodes/127.0.0.1
  - ./start_all.sh
- 检查区块链节点运行状态
  - ps -ef | grep fisco-bcos

快速入门搭建： WeBASE-Front搭建实操

- 参考文档
  - https://webasedoc.readthedocs.io/zh_CN/latest/docs/WeBASE-Install/developer.html
  - 依赖JAVA，JAVA_HOME
- 下载安装包
  - wget https://www.fisco.com.cn/cdn/WeBASE/release/download/v1.0.2/webase-front.zip
- 拷贝证书文件
  - 将节点所在目录nodes/${ip}/sdk下的ca.crt、node.crt和node.key文件拷贝到conf下
- 服务启停
  - **启动: bash start.sh**
  - 停止: bash stop.sh
  - 检查: bash status.sh
- 访问
  - http://{deployIP}:{frontPort}/WeBASE-Front
  - 示例：http://localhost:5002/WeBASE-Front

一键部署

- 一键部署可以在 **同机** 快速搭建WeBASE管理台环境，方便用户快速体验WeBASE管理平台的交易审计，交易展示，系统管理等功能
- 参考链接：一键部署

一键部署

- 参考文档
  - https://webasedoc.readthedocs.io/zh_CN/latest/docs/WeBASE/install.html
- 环境准备
  - 建议使用CentOS 7和Ubuntu 16.04 64bit
  - Java
  - MySQL
  - Python
  - MySQL-python

一键部署（环境准备）

- Java
  - 安装包下载（下载地址）
  - 解压
    - mkdir /software
    - tar -zxvf jdkXXX.tar.gz /software/
  - 配置环境变量
    - sudo vi /etc/profile
    - JAVA_HOME=/nemo/jdk1.8.0_181
      PATH=$PATH:$JAVA_HOME/bin
      CLASSPATH=.:$JAVA_HOME/lib
      export JAVA_HOME CLASSPATH PATH
    - source /etc/profile
  - java -version

一键部署（环境准备）

- MySQL
  - sudo yum install -y mariadb*
  - 启动：sudo systemctl start mariadb.service
    停止：sudo systemctl stop mariadb.service
  - sudo systemctl enable mariadb.service
  - 初始化
    - 执行以下命令：
      sudo mysql_secure_installation
      以下根据提示输入：
      Enter current password for root (enter for none):<–初次运行直接回车
      Set root password? [Y/n] <– 是否设置root用户密码，输入y并回车或直接回车
      New password: <– 设置root用户的密码
      Re-enter new password: <– 再输入一次你设置的密码
      Remove anonymous users? [Y/n] <– 是否删除匿名用户，回车
      Disallow root login remotely? [Y/n] <–是否禁止root远程登录，回车
      Remove test database and access to it? [Y/n] <– 是否删除test数据库，回车
      Reload privilege tables now? [Y/n] <– 是否重新加载权限表，回车
  - mysql -uroot -p -h localhost -P 3306
  - mysql > GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
    mysql > flush PRIVILEGES;
  - mysql > GRANT ALL PRIVILEGES ON *.* TO 'test'@localhost IDENTIFIED BY '123456' WITH GRANT OPTION;
    mysql > flush PRIVILEGES;

一键部署（环境准备）

- Python
  - CentOs
    - sudo yum install -y python-requests
  - Ubuntu
    - sudo apt-get install -y python-requests
- MySQL-python
  - CentOs
    - sudo yum install -y MySQL-python
  - Ubuntu
    - sudo apt-get install -y python-pip
      sudo -H pip install MySQL-python

一键部署

- 下载安装包
  - wget https://github.com/WeBankFinTech/WeBASELargeFiles/releases/download/v1.0.2/webase-deploy.zip
- 解压安装包
  - unzip webase-deploy.zip
- 进入目录
  - cd webase-deploy

一键部署

- **修改配置（默认只需要修改数据库配置，其他配置修改请参考文档：一键部署）**
  - mysql.ip（数据库ip）：sed -i "s%localhost%${your_db_ip}%g" common.properties
  - mysql.port（数据库端口）：sed -i "s%3306%${your_db_port}%g" common.properties
  - mysql.user（数据库用户名）：sed -i "s%dbUsername%${your_db_account}%g" common.properties
  - mysql.password（数据库密码）：sed -i "s%dbPassword%${your_db_password}%g" common.properties
  - mysql.database（数据库名称）：sed -i "s%webasenodemanager%${your_db_name}%g" common.properties
- **部署**
  - **python deploy.py installAll**
  - python deploy.py stopAll
- **访问**
  - http://{deployIP}:{webPort}
    示例：http://localhost:5000

03

WeBASE基本功能、原理和运用

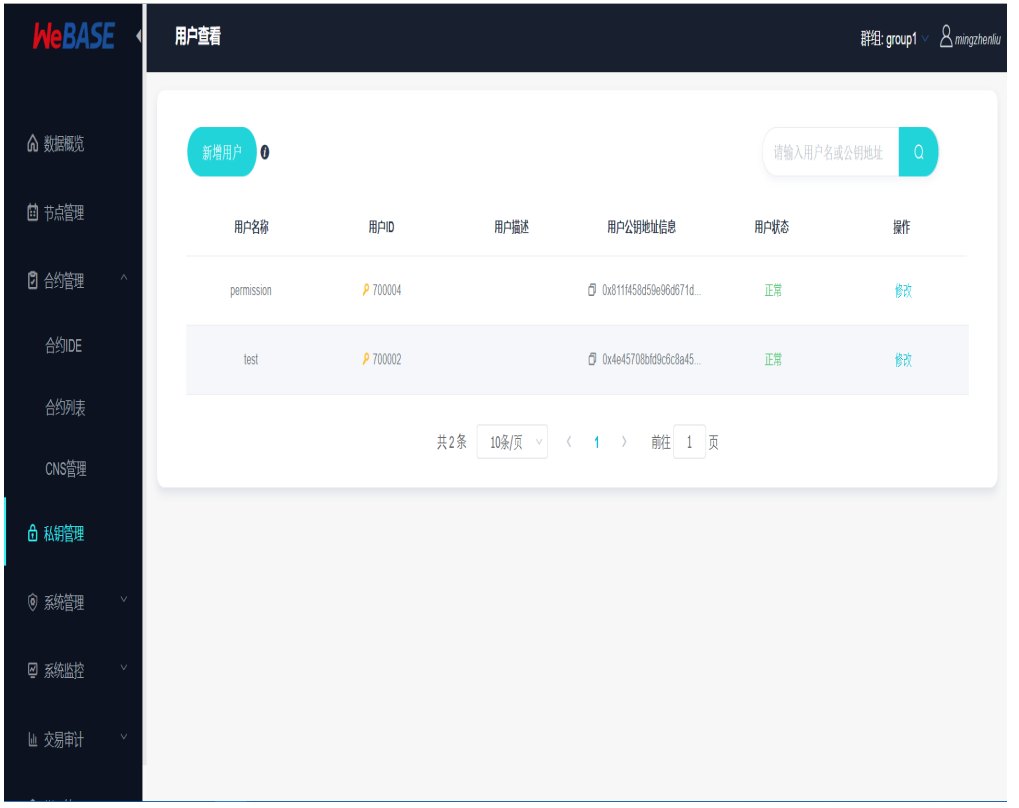
私钥管理

私钥管理主要功能：
- 私钥生产
- 公钥导入

私钥管理保存了系统用的公私钥。FISCO-BCOS使用基于椭圆曲线加密的椭圆曲线数字签名算法（ECDSA）。特定的椭圆曲线称为secp256k1。

私钥主要用途：
- 交易签名
- 用户关联

公钥主要用途：
- 用户关联



合约管理

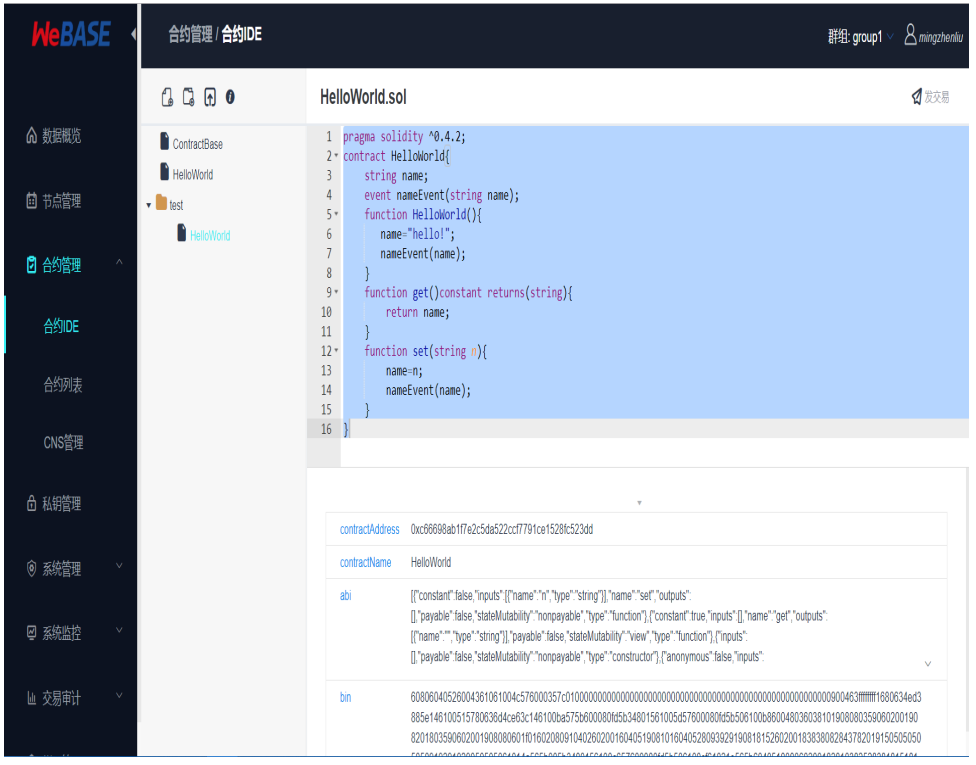
合约管理主要功能：

- 合约导入
- 合约编译
- 合约部署
- 合约调用

合约管理保存了合约的原文，还保存了合约编译后的ABI和BIN。

ABI和BIN是交易解析，交易审计的基础。

ABI编码规则可以参考：solidity文档。
在线编码器可以参考： abi.hashex.org 。

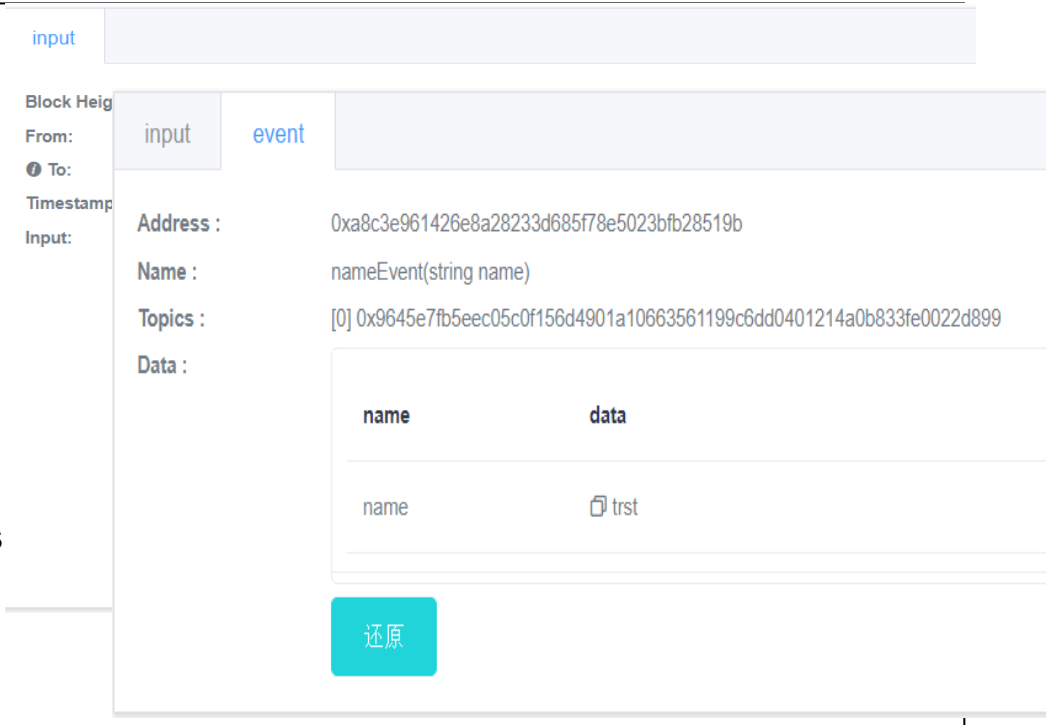


交易立体展示

交易立体展示主要功能：

- 构造函数Input展示
- 函数调用Input展示
- Output展示
- Event展示

- 构造函数Input编码规则
  - DeployBin+Params
- 函数调用Input编码规则
  - MethodID+Params
  - MethodID为函数签名的Keccak hash的前4bytes
- Output编码规则
  - Params
- Event展示
  - topics[n]+data
  - topic[0]为 keccak(EVENT_NAME+"("+EVENT_ARGS.map(canonical_type_of).join(",")+")")
  - 如果是匿名event，则没有topic[0]
  - Topic[n]是indexed的参数
  - Data是non indexed 参数



交易审计

- 背景

联盟链中各个机构按照联盟链委员会制定的规章在链上共享和流转数据。这些规章往往是字面的，大家是否遵守缺乏监管和审计。因此为了规范大家的使用方式，避免链的计算资源和存储资源被某些机构滥用，急需一套服务来辅助监管和审计链上的行为。

- 目标

提供可视化的去中心化合约部署和交易监控、审计功能，方便识别链资源被滥用的情况，为联盟链治理提供依据。

关键业务功能	业务功能描述
用户交易总量数量统计	监控链上各个外部交易账号的每日交易量
用户子类交易数量统计	监控链上各个外部交易账号的每种类型的每日交易量
异常交易用户监控	监控链上出现的异常交易用户（没在区块链中间件平台登记的交易用户）
异常合约部署监控	监控链上合约部署情况，非白名单合约（没在区块链中间件平台登记的合约）记录

系统管理

- 权限管理
  - 权限管理主要就是对底层权限管理模块的可视化。
- 系统配置管理
  - 主要是系统配置的展示、添加、修改。
- 节点管理
  - 主要是节点的展示、添加、修改。

04

学习使用WeBASE进行智能合约开发

实操：Asset合约


```
1  pragma solidity ^0.4.21;
2
3  contract Asset {
4      address public issuer;
5      mapping (address => uint) public balances;
6
7      event Sent(address from, address to, uint amount);
8
9      constructor() {
10         issuer = msg.sender;
11     }
12
13     function issue(address receiver, uint amount) public {
14         if (msg.sender != issuer) return;
15         balances[receiver] += amount;
16     }
17
18     function send(address receiver, uint amount) public {
19         if (balances[msg.sender] < amount) return;
20         balances[msg.sender] -= amount;
21         balances[receiver] += amount;
22         emit Sent(msg.sender, receiver, amount);
23     }
24
25 }
26
```

只有issuer才有权限发行资产

声明了key为address、value为uint的mapping，存储各个账户的余额

通知账户变动的事件，由send()调用

发行

转账接口

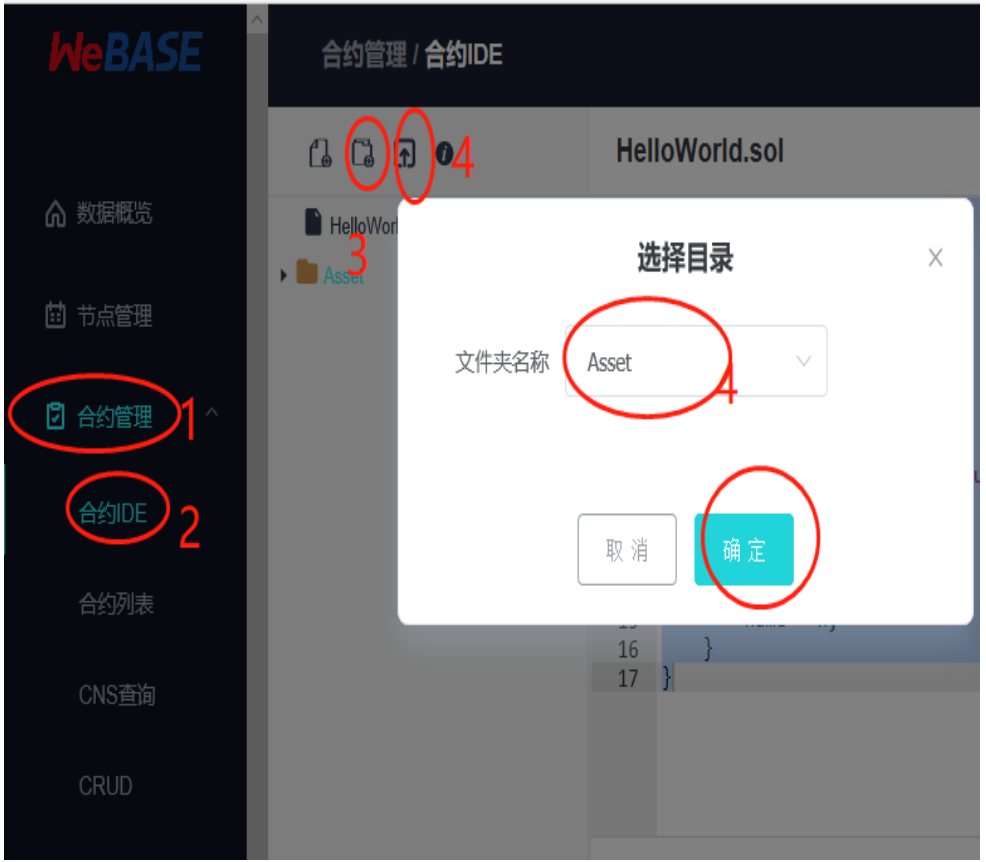
实操步骤

```
pragma solidity ^0.4.21;

contract Asset {
    address public issuer;
    mapping (address => uint) public balances;
    event Sent(address from, address to, uint amount);

    constructor() {
        issuer = msg.sender;
    }

                                                        Bob
    function issue(address receiver, uint amount) public {
        if (msg.sender != issuer) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }

}
```

1.导入合约
2.在私钥管理里新建三个演示账号：Issuer， Alice ，Bob
3.使用Issuer账号部署Asset
4.使用Issuer账号调用issue接口给Alice发放100资产
5.查询Alice的余额，此时应该是100
6.使用Alice的账号，调用send接口，给bob转账10
7.查询Alice的余额，此时应该是90
8.查询Bob的余额，此时应该是10

实操步骤：登陆管理台并导入合约

- 一键搭建管理台后，使用浏览器登陆
  - http://X.X.X.X:5000/#/login：X.X.X.X为机器ip
  - 默认用户名和密码分别是：admin/Abcd1234
- 登陆成功后使用合约管理功能，导入合约
  1. 合约管理
  2. 合约IDE
  3. 新建文件夹
  4. 上传文件

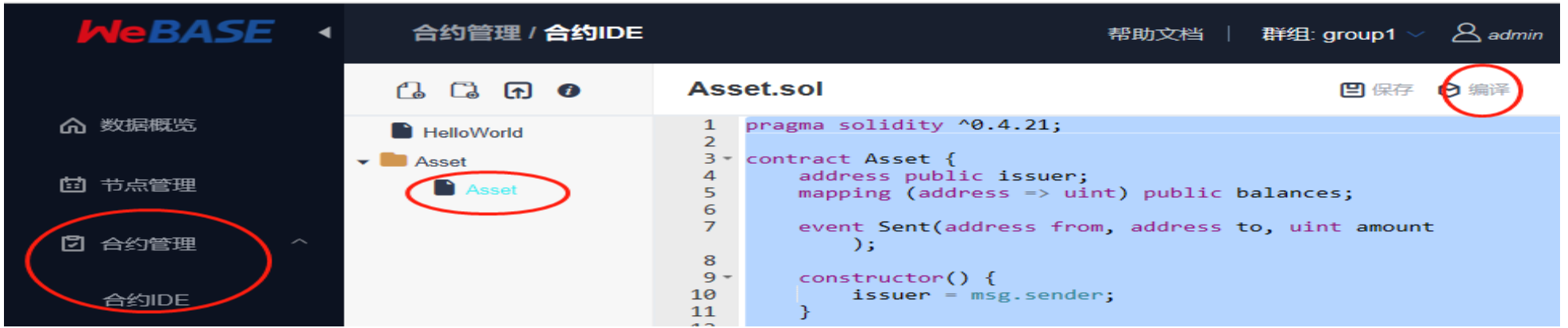


实操步骤：创建演示私钥用户

- 在私钥管理里新建三个演示账号：Issuer，Alice ，Bob
  - Issuer代表资产发行方
  - Alice和Bob代表交易转账的双方

实操步骤：编译合约

- 编译合约



实操步骤：部署合约

- 部署合约

实操步骤：合约调用

- 使用Issuer账号调用issue接口给Alice发放100资产

实操：存证智能合约

Evidence.sol　　EvidenceFactory.sol

- **完善的防篡改机制**：使用区块链技术保全证据，进一步加强了证据的不可篡改性
- **证据效力得到机构认可**：司法机构作为链上节点，对上链数据参与认可和签名，事后可从链上确认数据的真实有效性
- **服务持续有效**：数据被多方共识上链后，即使有部分共识方退出也不会造成数据的丢失或失效

实操：存证工厂合约

```
pragma solidity ^0.4.4;
import "Evidence.sol";

contract EvidenceFactory{
        address[] signers;
        event newEvidenceEvent(address addr);

        constructor(address[] evidenceSigners){}

        function newEvidence(string evi)
        public returns(address) {}

        function getEvidenceInfo(address addr)
        public constant returns(string){}

        function getEvidence(address addr)
        public constant returns(string,address[],address[]){}

        function addSignatures(address addr) public returns(bool) {}

        function verify(address addr)public constant returns(bool){}

        function getSigner(uint index)public constant returns(address){}

        function getSignersSize() public constant returns(uint){}

        function getSigners() public constant returns(address[]){}

}
```

存证工厂合约，用来生产具体的存证实例

指定存证需要哪些人的签名

新存证的事件通知

存证工厂生产一个新的存证实例，具体证据信息为evi

获取存证（addr）的具体信息

实操：存证合约

```
pragma solidity ^0.4.4;

contract EvidenceSignersDataABI{}

contract Evidence{
    string evidence;
    address[] signers;
    address public factoryAddr;

    event addSignaturesEvent(string evi);
    event newSignaturesEvent(string evi, address addr);
    event errorNewSignaturesEvent(string evi, address addr);
    event errorAddSignaturesEvent(string evi, address addr);
    event addRepeatSignaturesEvent(string evi);
    event errorRepeatSignaturesEvent(string evi);

    function CallVerify(address addr) public constant returns(bool) {}

    constructor(string evi, address addr)  {}

    function getEvidenceInfo() public constant returns(string){}

    function getEvidence() public constant returns(string,address[],address[]){}

    function addSignatures() public returns(bool) {}

    function getSigners()public constant returns(address[]){}
}
```

存证实例

存证的信息，一般是一个hash

调用存证工厂合约验证是否是存证的有效签名者

获取存证详情

增加签名确认

实操步骤

```
pragma solidity ^0.4.4;
import "Evidence.sol";

contract EvidenceFactory{
        address[] signers;
        event newEvidenceEvent(address addr);

        constructor(address[] evidenceSigners){}

        function newEvidence(string evi)
        public returns(address) {}

        function getEvidenceInfo(address addr)
        public constant returns(string){}

        function getEvidence(address addr)
        public constant returns(string,address[],address[]){}

        function addSignatures(address addr) public returns(bool) {}

        function verify(address addr)public constant returns(bool){}

        function getSigner(uint index)public constant returns(address){}

        function getSignersSize() public constant returns(uint){}

        function getSigners() public constant returns(address[]){}

}
```

1.导入合约
2.在私钥管理里新建三个演示账号：Arbitrator代表仲裁机构，Depositor代表存证机构，User代表用户
3.使用Arbitrator账号部署存证工厂合约EvidenceFactory,并且把三个演示账号的地址传入构造函数初始化存证系统
4.使用Depositor账号,新建存证(存证地址从交易返回event中获取)
5.查询存证信息，签名个数为1个
6.使用User账号,增加签名
7.查询存证信息,签名个数为2个
8.使用Arbitrator账号,增加签名
9.查询存证信息,签名个数为3个,存证完成

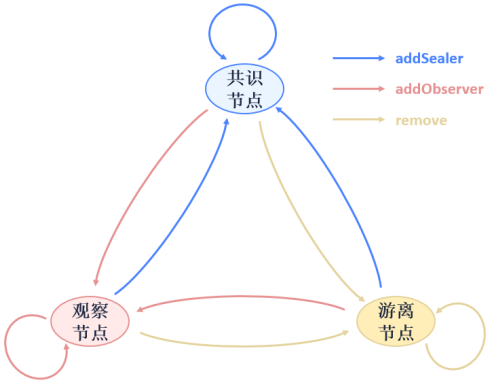
05

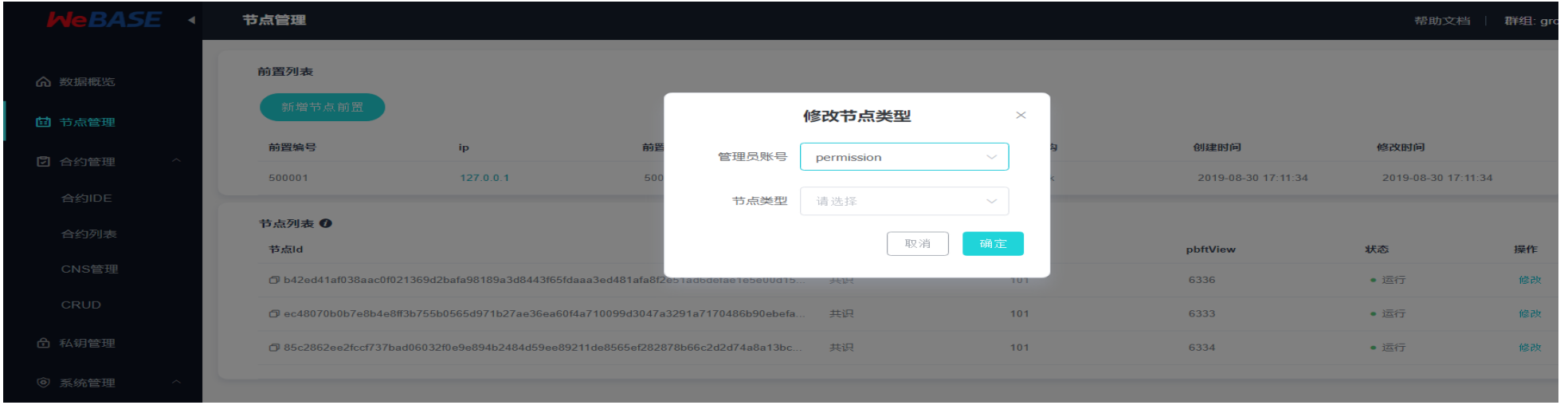
学习使用WeBASE进行FISCO BCOS系统管理

节点管理

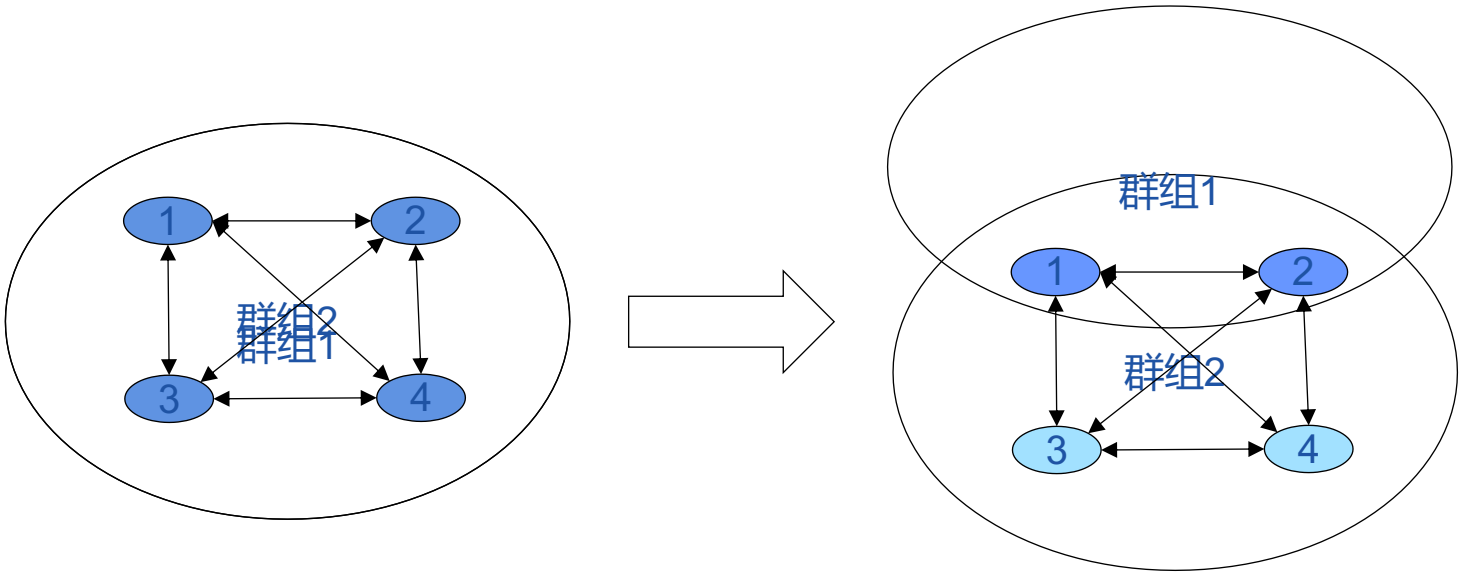
联盟链的FISCO BCOS使用**群组准入机制，来对各个节点行为做限制。**

共识节点：参与共识出块和交易/区块同步，
观察节点：只参与区块同步。
游离节点：完成网络准入但没有加入群组的节点。**游离节点尚未通过群组准入，不参与共识和同步。**





节点管理(练习)



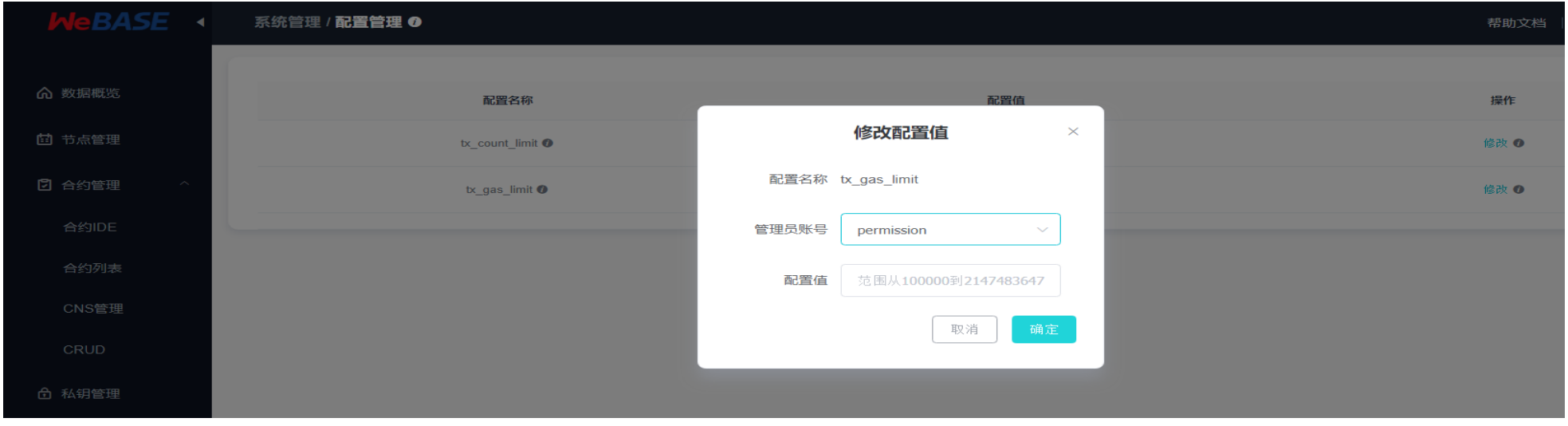
WeBASE-Web练习：

1． 将链的状态由1变迁到2：把3节点和4节点移出群组1

2． 将链的状态由2还原成1：把3节点和4节点重新加入群组1

配置管理

系统参数	默认值	含义
tx_count_limit	1000	一个区块中可打包的最大交易数目
tx_gas_limit	300000000	一个交易最大gas限制



配置管理练习

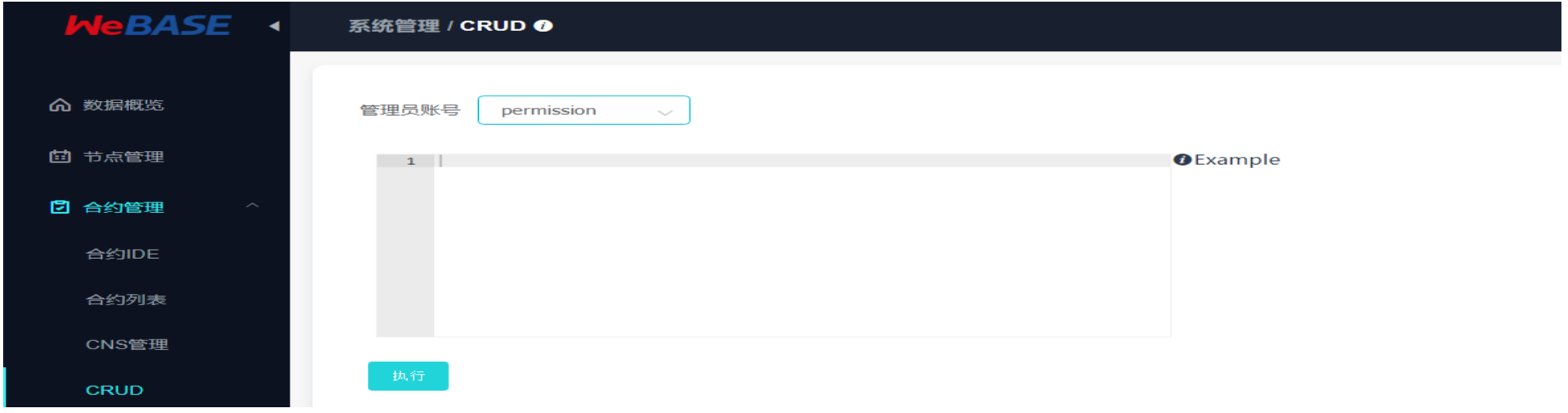
WeBASE-Web练习：

1. 把tx_gas_limit调整到100000。
2. 构造一个交易gas超过上限
3. 修改tx_gas_limit调整到2147483647。
4. 重发交易，交易成功。

CRUD

FISCO BCOS 2.0新增符合CRUD接口的合约接口规范，简化了将主流的面向SQL设计的商业应用迁移到区块链上的成本。

- 与传统业务开发模式类似，降低了合约开发学习成本；
- 合约只需关心核心逻辑，存储与计算分离，方便合约升级；
- CRUD底层逻辑基于预编译合约实现，数据存储采用分布式存储，效率更高；



CRUD练习

WeBASE-Web练习：

1. create table t_demo(name varchar, item_id varchar, item_name varchar, primary key(name))
2. insert into t_demo (name, item_id, item_name) values (fruit, 1, apple1)
3. select * from t_demo where name = fruit
4. update t_demo set item_name = orange where name = fruit and item_id = 1
5. delete from t_demo where name = fruit and item_id = 1
6. desc t_demo

微众银行，版权所有

WeBank

谢谢！