**2024《区块链技术与应用》**

**（微众银行）**

**实验3：在区块链上构建应用系统**

1. **实验背景与目标**

Spring-boot-crud框架作为一个轻量级的Java开发框架，提供了快速构建RESTful服务的能力，非常适合用来构建与区块链交互的服务接口。通过在命令行发起HTTP请求调用服务接口，可以模拟客户端与服务端的交互过程，验证服务接口的正确性和可用性。

**实验目标**

1. **掌握智能合约编译成Java接口代码的过程**：通过将智能合约编译成Java接口代码，参与者将了解智能合约与Java应用交互的机制。
2. **学会使用Spring-boot-crud框架构建服务调用区块链**：通过使用Spring-boot-crud框架构建服务调用区块链，参与者将掌握如何快速构建RESTful服务，并实现与区块链的交互。
3. **学会在命令行发起HTTP请求调用服务接口**：通过在命令行发起HTTP请求调用服务接口，参与者将学会如何测试和验证服务接口的正确性和可用性。

**实验步骤概述**

1. **将智能合约编译成Java接口代码**：学习智能合约编译成Java接口代码的方法和工具。
2. **使用Spring-boot-crud框架构建服务调用区块链**：使用Spring-boot-crud框架构建服务，实现对ERC20Test智能合约的调用和交互。
3. **在命令行发起HTTP请求调用服务接口**：在命令行发起HTTP请求调用Spring-boot-crud服务接口，验证服务接口的正确性和可用性。
4. **将智能合约编译成Java接口文件**

在控制台console提供了工具，可以将solidity合约生成出调用该合约java工具类。本例中使用console将Solidity合约生成出做调用该合约java工具类为例子。

下载及配置控制台在实验1已经有所提及，在此处不再赘述。

**使用命令生成智能合约的Java工具类**

|  |
| --- |
| *# 当前目录~/fisco/console*  $ bash contract2java.sh solidity -p org.com.fisco -s ./contracts/solidity/HelloWorld.sol  *# 以上命令中参数“org.com.fisco”是指定产生的java类所属的包名。*  *# 通过命令./contract2java.sh -h可查看该脚本使用方法* |

得到返回

\*\*\* Compile solidity HelloWorld.sol\*\*\*

INFO: Compile **for** solidity HelloWorld.sol success.

\*\*\* Convert solidity to java **for** HelloWorld.sol success \*\*\*

运行成功之后，将会在console/contracts/sdk目录生成java、abi和bin目录，如下所示。

|  |
| --- |
| |-- abi *# 编译生成的abi目录，存放solidity合约编译的abi文件*  | |-- HelloWorld.abi  |-- bin *# 编译生成的bin目录，存放solidity合约编译的bin文件*  | |-- HelloWorld.bin  |-- java *# 存放编译的包路径及Java合约文件*  | |-- org  | |-- com  | |-- fisco  | |-- HelloWorld.java *# Solidity编译的HelloWorld Java文件* |

Java目录下生成了org/com/fisco/包路径目录。包路径目录下将会生成Java合约文件HelloWorld.java。其中HelloWorld.java是Java应用所需要的Java合约文件。

**生成的Java文件代码结构**

下面以生成的HelloWorld.java的接口列表为例，简要说明代码结构。

|  |
| --- |
| **public** **class** **HelloWorld** **extends** Contract {  *// 构造函数*  **protected** HelloWorld(String contractAddress, Client client, CryptoKeyPair credential);  *// 根据CryptoSuite获取合约的code，若是国密就返回国密的code*  **public** **static** String getBinary(CryptoSuite cryptoSuite);  *// 获取合约的ABI json字符串*  **public** **static** String getABI();  *// HelloWorld合约get接口*  **public** String get() **throws** ContractException;  *// HelloWorld合约get接口的Function类，记录了输入和返回的类型，可用于ABI解析*  **public** Function getMethodGetRawFunction() **throws** ContractException;  *// HelloWorld合约set接口，输入String类型，返回交易回执*  **public** TransactionReceipt set(String n);  *// HelloWorld合约set接口的Function类，记录了输入和返回的类型，可用于ABI解析*  **public** Function getMethodSetRawFunction(String n) **throws** ContractException;  *// 获取已签名的调用set接口的交易，获取后可以直接发到链上*  **public** String getSignedTransactionForSet(String n);  *// HelloWorld合约set异步接口，输入String类型，返回交易哈希*  **public** String set(String n, TransactionCallback callback);  *// HelloWorld合约set的input解析*  **public** Tuple1<String> getSetInput(TransactionReceipt transactionReceipt);  *// 若链上有已知的HelloWorld合约，可以用改接口直接加载出Java 的HelloWorld类。注意：ABI必须相同，否则会出现调用失败*  **public** **static** HelloWorld load(String contractAddress, Client client, CryptoKeyPair credential);  *// 向链上发起部署合约操作，返回Java 的HelloWorld类。*  **public** **static** HelloWorld deploy(Client client, CryptoKeyPair credential) **throws** ContractException;  } |

1. **使用Spring-boot-crud框架构建服务调用区块链**

Spring-boot-crud示例项目基于Java SDK + Maven + SpringBoot方式来调用智能合约。

1. **获取源码**

# 直接从github克隆代码

git clone https://github.com/FISCO-BCOS/spring-boot-crud.git

# 若网络很慢，可从gitee克隆代码

git clone https://gitee.com/FISCO-BCOS/spring-boot-crud

1. **代码结构**

tree ./src/main/java/com/fisco/app -L 2

./src/main/java/com/fisco/app

├── AppApplication.java # App的总入口

├── client # 连接链的Client实现

│   ├── CRUDClient.java

│   ├── CRUDMultiParameterClient.java

│   └── KVClient.java

├── common # 基类定义

│   ├── CommonClient.java

│   └── CommonClientMultiParameter.java

├── config # 链配置类

│   └── FiscoConfig.java

├── contract # 智能合约Java接口

│   ├── TestCRUD.java

│   ├── TestCRUDMultiParameter.java

│   └── TestKV.java

├── controller # 请求控制，调用Client类实现合约互动

│   ├── CrudController.java

│   ├── CrudMultiParameterController.java

│   └── KVController.java

├── entity # 实体类定义

│   ├── Person.java

│   └── ResponseData.java

└── utils # 工具类

└── SpringUtils.java

8 directories, 16 files

1. **配置节点证书**

将节点所在目录`nodes/${ip}/sdk`下的ca.crt、sdk.crt和sdk.key文件拷贝到项目的`src/main/resources/conf`目录下供SDK使用

设节点路径为`~/fisco/nodes/127.0.0.1`，则可使用如下命令拷贝SDK证书:

# 进入项目路径

$ cd spring-boot-crud

# 创建证书存放路径

$ mkdir src/main/resources/conf

# 拷贝SDK证书

$ cp ~/fisco/nodes/127.0.0.1/sdk/\* src/main/resources/conf/

1. **设置配置文件**

`spring-boot-crud`包括SDK配置文件（位于`src/main/resources/applicationContext.xml`路径）和WebServer配置文件(位于`src/main/resources/application.yml`路径)。

需要根据区块链节点的IP和端口相应配置`applicationContext.xml`的`network.peers`配置项，具体如下：

<property name="network">

<map>

<entry key="peers">

<list>

<value>127.0.0.1:20200</value>

<value>127.0.0.1:20201</value>

</list>

</entry>

<entry key="defaultGroup" value="group0" />

</map>

</property>

WebServer主要配置了监听端口，默认为`45000`，具体如下：

server:

#端口号

port: 45000

1. **编译和安装项目**

可以使用IDEA导入并编译并安装该项目，也可使用提供的`mvnw`脚本在命令行编译项目如下：

# 编译项目

$ bash mvnw compile

# 安装项目，安装完毕后，在target/目录下生成fisco-bcos-spring-boot-crud-0.0.1-SNAPSHOT.jar的jar包

$ bash mvnw install

1. **启动spring-boot-crud服务**

\*\*方法一：\*\*

打开IDEA导入并编译该项目，编译成功后，运行`AppApplication.java`即可启动spring boot服务。

\*\*方法二：\*\*

使用`bash mvnw install`生成的jar包`target/fisco-bcos-spring-boot-crud-0.0.1-SNAPSHOT.jar`启动spring-boot-crud服务：

# 启动spring-boot-crud(启动成功后会输出create client for group success的日志)

$ java -jar ./target/fisco-bcos-spring-boot-crud-0.0.1-SNAPSHOT.jar

1. **访问spring boot web服务**
   1. **访问用户信息上链API(CRUD insert)**

`spring-boot-crud`基于CRUD insert接口实现了用户信息上链的API，将`Person`类型的用户信息上链，API声明如下：

@Data

public class Person {

private String name;

private String age;

private String tel;

}

@PostMapping("/insert")

public ResponseData insert(@RequestBody Person person) {

crudClient.insert(person.getName(),person.getAge(), person.getTel());

return ResponseData.success("新增成功");

}

\*\*使用curl工具访问接口如下\*\*：

# 这里假设WebServer监听端口为45000

# 将用户fisco的信息上链，其中name为fisco, age为6，tel为123456789

$ curl -H "Content-Type: application/json" -X POST --data '{"name":"fisco", "age":"6", "tel":"123456789"}' http://localhost:45000/insert

# 返回新增成功的信息

{"code":200,"msg":"新增成功","data":null}

* 1. **访问链上查询用户信息API(CRUD select)**

`spring-boot-crud`基于CRUD select接口实现了链上查询用户信息的API，基于用户名查询用户信息，API声明如下：

@GetMapping("/query/{name}")

public ResponseData query(@PathVariable("name") String name) throws Exception {

return ResponseData.success(crudClient.query(name));

}

\*\*使用curl工具访问接口如下\*\*：

# 这里假设WebServer监听端口为45000

# 查询用户名为fisco的用户信息

$ curl http://localhost:45000/query/fisco

# 返回用户fisco的具体信息

{"code":200,"msg":null,"data":{"value1":"fisco","value2":"6","value3":"123456789","size":3}}