如何编写智能合约(Smart Contract)? (III)建立标准代币部落币「BLC」

X

块

链

学

习

框

架

冬



golang|Java|Python|Node.js

区块链技术概况

区块链进阶

密码学基础

比特币区块链开发

以太坊智能合约开发

超级账本 - Fabric

IPFS - 分布式存储

算法



讲师:黎跃春

简介:曾就职于中国石油东方地球物理公司、北京中友瑞飞公司移动业务部 担任 hybridApp 研发工程师; iOS资深讲师, React系全栈工程师; 15年接触数字货币, 16年初开始关注区块链技术的发展, 16年中旬开始专注于区块链技术的研究。16年12月创办孔壹学院,旨在专注于『区块链+内容』产品的研发以及对区块链技术的推广和普及。

博客: http://liyuechun.org

Github: http://github.com/liquechun

微博:黎跃春-追时间的人

公众号: 区块链部落



在上一篇中,我们我们<u>如何编写智能合约? (II) 建立简易的加密代币</u>,但是它存在很多安全问题,在本章中,我们将一步步带领大家创建一个能够放到 以太币钱包的加密代币。

创建项目

有别于之前使用 truffle init 指令来初始化项目,在 Truffle 推出 Boxes 功能之后,我们可以直接套用称作 react-box 的样板,此样板已经整合 create-react-app,可以直接用它来开发 react web,省下项目设置的时间。

liyuechun:BloggerCoin yuechunli\$ pwd

/Users/liyuechun/Desktop/SmartContractDemo/BloggerCoin liyuechun:BloggerCoin yuechunli\$ truffle unbox react-box

Downloading... Unpacking... Setting up...

Unbox successful, Sweet!

Commands:

Compile: truffle compile
Migrate: truffle migrate
Test contracts: truffle test

Test dapp: npm test

Run dev server: npm run start
Build for production: npm run build
liyuechun:BloggerCoin yuechunli\$

```
liyuechun:BloggerCoin yuechunli$ pwd
/Users/liyuechun/Desktop/SmartContractDemo/BloggerCoin
liyuechun:BloggerCoin yuechunli$ truffle unbox react-box
Downloading...
Unpacking...
Setting up...
```

目录结构:

```
SimpleStorage.sol
                                                                             Project
BloggerCoin
                                                                                                                                                                                                                                                               pragma solidity ^0.4.2;
  > 💼 config

→ im contracts

                                                                                                                                                                                                                                                               contract SimpleStorage {
                   Migrations.sol
                                                                                                                                                                                                                                                                             uint storedData;
    > node_modules
                                                                                                                                                                                                                                                                                         storedData = x;
  > public
  > 💼 scripts
 Y 🛅 src
           > 💼 css
                                                                                                                                                                                                                                                                                  function get() constant returns (uint) {
              > in fonts
                                                                                                                                                                                                                                                                                           return storedData;
              > 🛅 utils
                      index.js

→ intest

           box-img-lg.png

    box-img-sm.png
    box-img-sm.png
    box-img-sm.png
    contact    contact
```

● /contracts :存放智能合约原始码的地方,可以看到里面已经有放两个 sol 文件。我们开发的

BloggerCoin.sol 也会放在这里。

- /migrations :这是 Truffle 用来部署智能合约的功能,待会我们会修改 2_deploy_contracts.js 来部署 BloggerCoin.sol。
- /test:测试智能合约的代码放这目录,支持 js 与 sol 测试。
- /public 、/src:存放 react web 的地方,后面用到会再说明。
- truffle.js: Truffle 的设置文件。

开发前的准备

- 1. 打开终端, 启动 testrpc, 继续通过 testrpc 模拟以太坊区块链测试环境。
- 2. 创建的代币如果想要能够通过以太币钱包来进行转账和收帐,必须兼容于以太坊的 ERC20 标准, ERC20 定义了支持钱包所必需的合约界面。
- 3. 在本篇文章中,我们将安装 OpenZeppelin 来简化加密钱包开发的过程。 OpenZeppelin 是一套能够给我们方便提供编写加密合约的函数库,同时里面也提供了兼容 ERC20 的智能合约。

liyuechun:BloggerCoin yuechunli\$ npm install zeppelin-solidity

```
■ BloggerCoin — node « npm TERM_PROGRAM=Apple_Terminal ANDROID_HOME=/Users/liyuechun/Library/Android/sdk SHELL=/bin/bash — 80×2<sup>2</sup>
lliyuechun:SmartContractDemo yuechunli$ ls
EncryptedToken HelloWorld
liyuechun:SmartContractDemo yuechunli$ mkdir BloggerCoin
liyuechun:SmartContractDemo yuechunli$ cd BloggerCoin/
liyuechun:BloggerCoin yuechunli$ truffe init
-bash: truffe: command not found
livuechun:BloggerCoin vuechunli$ truffle init
Downloading project...
Project initialized.
  Documentation: http://truffleframework.com/docs
Commands:
  Compile: truffle compile
  Migrate: truffle migrate
  Test: truffle test
liyuechun:BloggerCoin yuechunli$ ls
                 migrations
contracts
                                  test
                                                    truffle.is
liyuechun:BloggerCoin yuechunli$ npm install zeppelin-solidity
                    ) = extract:coinstring: sill extract qs@6.5.1
```

Atom打开项目查看zeppelin-solidity安装结果

通过Atom打开项目,在 node_modules 中的最后一个文件夹就是 zeppelin-solidity 的内容。

```
> m wrap-ans
                                        "webpack-dev-server": "1.16.2",
 > m wrappy
 > 🖮 write
                                        "webpack-manifest-plugin": "1.1.0",
 > 🛅 xhr
                                        "whatwg-fetch": "1.0.0"
 > m xml-char-classes
 > m xml-name-validator
                                     "dependencies": {
 > m xmlhttprequest
                                       "dotenv": "^2.0.0",
 > 📺 xtend
 > 🛅 y18n
                                        "react": "^15.4.2",
 > 💼 yallist
                                        "react-dom": "^15.4.2",
 > 📺 yargs
                                      "zeppelin-solidity": "^1.2.0"
 > args-parse
 > zeppelin-solidity
 public
                                     "scripts": {
                                       "start": "node scripts/start.js",
Y 🛅 src
 > 🖿 css
                                        "build": "node scripts/build.js",
 > in fonts
                                       "test": "node scripts/test.js --env=jsdom"
 > 🛅 utils
  App.css
  App.test.js
                                       "collectCoverageFrom": [
  index.js
                                         "src/**/*.{js,jsx}"
∨ 💼 test
                                       "setupFiles": [
 .DS_Store
                                         "<rootDir>/config/polyfills.js"
 box-img-lg.png
 box-img-sm.png
                                       "testMatch": [
                                          "<rootDir>/src/**/__tests__/**/*.js?(x)",
 truffle-config.js
                                          "<rootDir>/src/**/?(*.)(spec|test).js?(x)"
 truffle.js
```

创建标准的「BLC」代币合约

在 contracts/目录下建立一个 BloggerCoin.sol 文件。也可以使用以下命令来创建文件:

```
liyuechun:BloggerCoin yuechunli$ truffle create contract BloggerCoin
liyuechun:BloggerCoin yuechunli$ ls
contracts node_modules test
migrations package-lock.json truffle.js
liyuechun:BloggerCoin yuechunli$ cd contracts/
liyuechun:contracts yuechunli$ ls
BloggerCoin.sol ConvertLib.sol MetaCoin.sol Migrations.sol
liyuechun:contracts yuechunli$
```

BloggerCoin.sol 代码如下:

```
BloggerCoin
                                                                                                                                              pragma solidity ^0.4.4;
> 🖿 config
                                                                                                                                               import "zeppelin-solidity/contracts/token/StandardToken.sol";
  contracts
              BloggerCoin.sol
                                                                                                                                               contract BloggerCoin is StandardToken {
            Migrations.sol
            SimpleStorage.sol
                                                                                                                                                        string public name = "BloggerCoin";
                                                                                                                                                        string public symbol = "BLC";
  > node_modules
  > public
                                                                                                                                                        uint8 public decimals = 2;
                                                                                                                                                        uint256 public INITIAL_SUPPLY = 666666;

✓ 

mathrid

mat
                                                                                                                                                        function BloggerCoin() {
       > 💼 css
       > in fonts
                                                                                                                                                                  totalSupply = INITIAL_SUPPLY;
       > 🛅 utils
                                                                                                                                                                   balances[msg.sender] = INITIAL_SUPPLY;
           App.css
           App.test.is
           index.css
           index.js
   test
           simplestorage.js
     .DS_Store
     box-img-lg.png
      box-img-sm.png
     package-lock.json
     package.json
      truffle-config.js
      truffle.is
```

```
pragma solidity ^0.4.4;
import "zeppelin-solidity/contracts/token/StandardToken.sol";
contract BloggerCoin is StandardToken {
    string public name = "BloggerCoin";
    string public symbol = "BLC";
    uint8 public decimals = 4;
    uint256 public INITIAL_SUPPLY = 666666;
    function BloggerCoin() {
        totalSupply = INITIAL_SUPPLY;
        balances[msg.sender] = INITIAL_SUPPLY;
    }
}
```

代码解释

```
pragma solidity ^0.4.4;
```

第一行代表 solidity 的版本,不同的版本编译的字节码不一样,个代表向上兼容,不过版本不能超过 0.5.0。

```
import "zeppelin-solidity/contracts/token/StandardToken.sol";
```

这句代码是通过 import 来导入我们需要使用到的 StandardToken 合约。

```
contract BloggerCoin is StandardToken {
    ...
}
```

建立 BloggerCoin 合约时,让 BloggerCoin 合约直接继承自 StandardToken 。 is 既是继承。因此 BloggerCoin 继承了 StandardToken 所有的状态数据和方法。

当我们继承了 StandardToken 合约,也就支持了以下 ERC20 标准中规定的函数。

函数	方法	
totalSupply()	代币发行的总量	
balanceOf(A)	查询A帐户下的代币数目	
transfer(A,x)	发送x个代币到A帐户	
transferFrom(A,x)	从A帐户提取x个代币	
approve(A,x)	同意A帐户从我的帐户中提取代币	
allowance(A,B)	查询B帐户可以从A帐户提取多少代币	

和之前一样,后面验证时会用到 balanceOf 和 transfer 两个函数。因为 StandardToken 合约中已 经帮我们实现了这些函数,因此我们不需要自己从头再写一次。

```
string public name = "BloggerCoin";
string public symbol = "BLC";
uint8 public decimals = 4;
uint256 public INITIAL_SUPPLY = 666666;
```

这边设定参数的目的是指定这个代币的一些特性。以人民币为例,人民币的名称(name)是 RMB , 美元的代号为 ¥ , 拿 100元 去找零最小可以拿到零钱是一分,也就是 0.0001 元。因为 1元 最小可分割到小数点 后4位 (0.0001) , 因此最小交易单位(decimals)为 4 。

这里将这个加密代币取名(name)为BloggerCoin(部落币),代币的代号(symbol)为BLC,最小分割单位是4(最小可以找0.0001个部落币)。

以下为人民币, 比特币, 以太币, 部落币的对照表供参考:

name	symbol	decimals
RMB	¥	4
Bitcoin	ВТС	8
Ethereum	ETH	18
BloggerCoin	BLC	4

最后也定义了初始代币数目INITIAL_SUPPLY。这里选择了一个吉祥数字 666666 。另外,当我们把全局变量设为 public (公开),编译时就会自动新增一个读取公开变量的 ABI接口,我们在 truffle console 中也可以读取这些变量。

```
function BloggerCoin() {
   totalSupply = INITIAL_SUPPLY;
   balances[msg.sender] = INITIAL_SUPPLY;
}
```

和合约同名的 BloggerCoin 方法,就是 BloggerCoin 合约的 构造函函数(constructor)。在构造函数里指定了 totalSupply 数目,并将所有的初始代币 [NITIAL_SUPPLY] 都指定给 msg.sender 帐号,也就是用来部署这个合约的帐号。 totalSupply 定义于 [ERC20Basic.sol 中, balances 定义于 [BasicToken.sol 中。

```
▼ ERC20Basic.sol
                                  pragma solidity ^0.4.11;
zeppelin-solidity
> 🛅 audit

✓ ■ contracts

  > 💼 crowdsale
  > iii lifecycle
  > math
  > mownership
  > i payment

▼ ERC20Basic.sol

                               contract ERC20Basic {

    □ LimitedTransferToken.sol

                                   uint256 public totalSupply;
     MintableToken.sol
                                     function balanceOf(address who) constant returns (uint256);
                                      function transfer(address to, uint256 value) returns (bool);
     StandardToken.sol
                                      event Transfer(address indexed from, address indexed to, uint256 va
     TokenOption.sol
     TokenTimelock.sol
   DayLimit.sol

    □ LimitBalance.sol

> 🖿 docs
> migrations
> in scripts
> 🛅 test
 coveralls.yml
 ishintrc
```

```
BasicToken.sol
        Proiect
zeppelin-solidity
> 🛅 audit

→ im contracts

  > in crowdsal
                                  contract BasicToken is ERC20Basic {
  > 🛅 lifecycle
  > 🖿 math
                                       using SafeMath for uint256;
  > m ownership
  > 💼 payment
   💼 token
                                        mapping(address => uint256) balances;
    BasicToken.sol

▼ ERC20Basic.sol

    □ LimitedTransferToken.sol

     MintableToken.sol
     StandardToken.sol
     TokenOption.sol
                                        function transfer(address _to, uint256 _value) returns (bool) {
     TokenTimelock.sol

▼ VestedToken.sol

                                             balances[msg.sender] = balances[msg.sender].sub(_value);
                                             balances[_to] = balances[_to].add(_value);
                                             Transfer(msg.sender, _to, _value);
   ECRecovery.sol

→ LimitBalance.sol

                                             return true;
   ReentrancyGuard.sol
> in docs
> migrations
> m scripts
> 🛅 test
 jshintro
  npmignore
 solcover.js
                                        function halanceOf(address owner) constant returns (uint256 halanceOf(address))
```

```
pragma solidity ^0.4.11;
import './ERC20Basic.sol';
import '../math/SafeMath.sol';
/**
* @title Basic token
* @dev Basic version of StandardToken, with no allowances.
contract BasicToken is ERC20Basic {
    using SafeMath for uint256;
    mapping(address => uint256) balances;
  /**
  * @dev transfer token for a specified address
  * @param _to The address to transfer to.
  * @param _value The amount to be transferred.
  */
    function transfer(address _to, uint256 _value) returns (bool) {
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        Transfer(msg.sender, _to, _value);
        return true;
    }
```

```
/**
 * @dev Gets the balance of the specified address.
 * @param _owner The address to query the the balance of.
 * @return An uint256 representing the amount owned by the passed address.
 */
   function balanceOf(address _owner) constant returns (uint256 balance) {
     return balances[_owner];
   }
}
```

进一步追去看·BasicToken.sol 合约的内容,可以发现 BasicToken.sol 合约中导入了 SafeMath.sol 合约。 SafeMath`对各种数值运算加入了必要的验证,让合约中的数字计算更安全。

如此一来,我们已写好一个可通过以太币钱包交易的新加密代币合约。这个合约一经部署,就可以一直存在于以太坊区块链上,世界上从此也就多了一种新的加密代币。只要你能找到人想拥有这种代币,这种代币就有交易的价值。

编译、部署、验证

在migrations/目录下建立一个3_deploy_bloggerchain.js文件,内容如下:

现在执行compile与migrate命令

备注: 确保 testrpc 处于运行状态。

• truffle compile

```
/Users/liyuechun/Desktop/SmartContractDemo/BloggerCoin
liyuechun:BloggerCoin yuechunli$ truffle compile
Compiling ./contracts/BloggerCoin.sol...
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/SimpleStorage.sol...
Compiling zeppelin-solidity/contracts/math/SafeMath.sol...
Compiling zeppelin-solidity/contracts/token/BasicToken.sol...
Compiling zeppelin-solidity/contracts/token/ERC20.sol...
Compiling zeppelin-solidity/contracts/token/ERC20Basic.sol...
Compiling zeppelin-solidity/contracts/token/StandardToken.sol...
Writing artifacts to ./build/contracts
liyuechun:BloggerCoin yuechunli$
```

truffle migrate

```
liyuechun:BloggerCoin yuechunli$ truffle migrate
Using network 'development'.
Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xac35fdd655a7b8916d5a43fb608227f1827aa666e4d4aa7b4d50347f8883de8a
 Migrations: 0x5c7102091425e16998b8bed1cd6634f499ab3684
Saving successful migration to network...
  ... 0x1131a209a1ca27cadbec4ef8f84cecbe322e59d01b2b584f3e0ddada5a7a53d8
Saving artifacts...
Running migration: 2 deploy contracts.js
  Deploying BloggerCoin...
  ... 0xc23199c5fe72206a5d74ad09797c9df17deb361c56ee1cb14b816ee0d874d5e2
  BloggerCoin: 0xbacb9b3da2e3140df11516be2244c4ea230d6d39
Saving successful migration to network...
  ... 0x32bf4f5299bb4d260cc86da76591d9564376a82c4b8122261043d74a70c57b9e
Saving artifacts...
Running migration: 3_deploy_bloggerchain.js
  Replacing BloggerCoin...
  ... 0x87e8c7a24727a06da750a2c9f3b4ea1bc4b87c8c3e9c8a9219c3dada911e0991
  BloggerCoin: 0x5262d2b6de1a1187abdd203cb726b387bcd6140f
Saving successful migration to network...
  ... 0x75166d7f6ee595437718df960d9a3bc76466bd890988a92b1aac1a396dc7f018
Saving artifacts...
liyuechun:BloggerCoin yuechunli$
```

验证

```
liyuechun:BloggerCoin yuechunli$ truffle console
truffle(development)> let contract
undefined
truffle(development)> BloggerCoin.deployed().then(instance => contract = instance)
.....
truffle(development)> contract.balanceOf(web3.eth.coinbase)
{ [String: '66666'] s: 1, e: 5, c: [ 666666 ] }
truffle(development)> contract.balanceOf(web3.eth.accounts[1])
{ [String: '600000'] s: 1, e: 0, c: [ 0 ] }
truffle(development)> contract.transfer(web3.eth.accounts[1], 600000)
truffle(development)> contract.balanceOf(web3.eth.coinbase)
{ [String: '66666'] s: 1, e: 4, c: [ 66666 ] }
truffle(development)> contract.balanceOf(web3.eth.accounts[1])
{ [String: '600000'] s: 1, e: 5, c: [ 600000 ] }
truffle(development)>
```

验证过程中具体方法的讲解,请看这篇文章:如何编写智能合约?(II)建立简易的加密代币

结语

我们用到 OpenZeppelin 来简化我们加密代币的开发,当然在正式的系统中,建议大家看看 OpenZeppelin 源码,检查一下是否还有缺陷,同时也可以从这个开源库中学到不少东西。

打赏地址

比特币: 1FcbBw62FHBJKTiLGNoguSwkBdVnJQ9NUn

以太坊: 0xF055775eBD516e7419ae486C1d50C682d4170645

技术交流

● 区块链技术交流QQ群: 348924182

• 「区块链部落」官方公众号





长按, 识别二维码, 加关注

参考资料

- [1] http://solidity.readthedocs.io/en/latest/index.html
- [2] https://ethereum.github.io/browser-solidity/
- [3] http://truffleframework.com/
- [4] https://github.com/iurimatias/embark-framework

- [5] https://github.com/ethereum/ens
- [6] https://github.com/ethereumjs/testrpc
- [7] https://github.com/ethereumjs/ethereumjs-vm
- [8] http://web3js.readthedocs.io/en/1.0/index.html