一：环境搭建

**搭建Truffle框架**

Truffle（智能合约的编译与部署）：https://truffleframework.com

基于以太坊的Solidity语言的一套开发框架,内置智能合约编译，连接，部署等功能。

（1）:安装Truffle ，命令行：npm install -g truffle

（2）：Solc编译器，truffle使用solc编译器编译智能合约，但是truffle会自动预装一个solc编译器

创建truffle项目时，目录结构对应的含义：

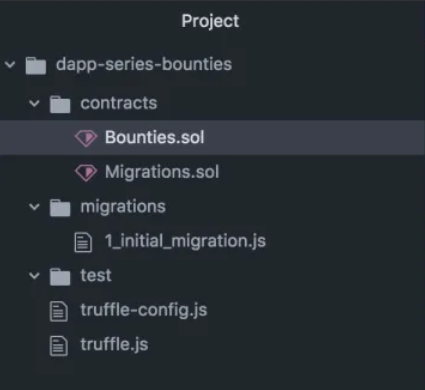
**contracts/:** 存储智能合约的原始代码。我们将把Bounties.sol文件放在这里。

**migrations/:** 在“contracts”文件夹中部署智能合约的说明。

**test/:** 对智能合约的测试，truffle支持用Javascript和Solidity编写的测试，学习编写测试内容在下一章。

**truffle.js:** 配置文件

**truffle-config.js:** Windows用户的配置文档()。



**Ganache**

因为在以太坊上部署和测试智能合约都是要花费以太币的，也就是油费（Gas）。而Ganache可以在本地创建一个区块链网络来测试程序，不需要消耗真实的油费。

工作原理：创建一个Ganachi虚拟的区块链网络，给我们分配10个外部账号（节点），每个账户都有100个假以太币。

**ACCOUNTS：**账号界面，显示了自动生成所有的账号及其余额。

**BLOCKS：**区块界面，显示了再本地区块链网络上挖掘的每一个区块，及其Gas成本和包含的交易。

**TRANSACTIONS**：交易页面，列出了再本地区块链上发生的所有交易

**CONTRACTS：**合约页面

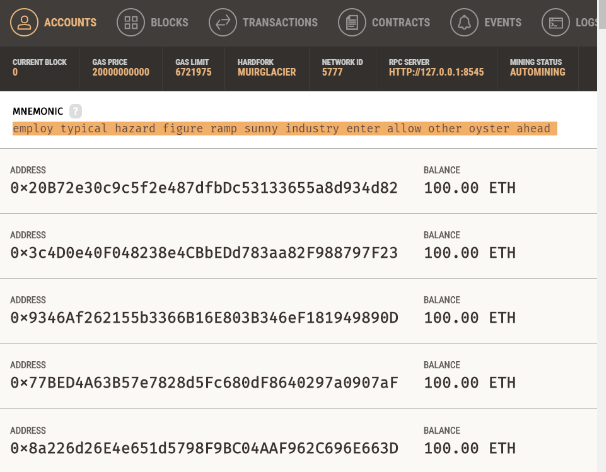
**EVENTS：**事件页面

**LOGS：**日志页面

**SERVER：**服务器设置页面，管理关于网络连接的详细信息，比如网络id，端口，主机名和自动挖掘状态。

**ACCOUNTS & KEYS：**账户和密钥页，设置自动生成账户及其余额。

CHAIN 链页，可以设置Gas限制和Gas价格。



3.**solidity**（现阶段没学太多，下个月的时候重点输出这一部分的文章，这个月的输出主要是以合约框架为主）

二．开发步骤

**1.初始化项目：**

用truffle init初始化项目，初始化成功之后再点开mydapp这个文件

**contracts目录：** 顾名思义，是智能合约目录，现在里面已经有了一个**Migrations.sol文件：**功能是迁移/部署/升级智能合约。

**migrations目录：**迁移文件，里面的js脚本都是帮助我们将只能合约部署到区块链上的。

**test目录：**测试代码目录。

**truffle-config.js文件：**可以在里面配置网络。

**2.添加package.json文件：**

{

"name": "ethereum-demo",

"version": "1.0.0",

"description": "以太坊demo",

"main": "truffle-config.js",

"directories": {

"test": "test"

},

"scripts": {

"dev": "lite-server",

"test": "echo \"Error: no test specified\" && sexit 1"

},

"author": "kevinhwu@qikegu.com",

"license": "ISC",

"devDependencies": {

"@truffle/contract": "^4.0.33",

"dotenv": "^8.1.0",

"lite-server": "^2.5.4",

"truffle-hdwallet-provider": "^1.0.17"

}

}

**package.json概述：**每个项目的根目录下面，一般都有一个package.json文件，定义了这个项目所需的各种模块，以及项目的配置信息（比如：名称，版本，许可证等元素），npm install命令，根据这个配置文件，自动下载所需要的模块，也就是项目所需要的运行和开发环境。

**3.添加智能合约源文件**

在contracts目录下创建一个新文件MyContract.sol，给出代码：

// 声明solidity版本

pragma solidity ^0.5.0;

// 声明智能合约MyContract，合约的所有代码都包含在花括号中。

contract MyContract {

// 声明一个名为value的状态变量

string value;

// 合约构造函数，每当将合约部署到网络时都会调用它。

// 此函数具有public函数修饰符，以确保它对公共接口可用。

// 在这个函数中，我们将公共变量value的值设置为“myValue”。

constructor() public {

value = "myValue";

}

// 本函数读取值状态变量的值。可见性设置为public，以便外部帐户可以访问它。

// 它还包含view修饰符并指定一个字符串返回值。

function get() public view returns(string memory ) {

return value;

}

// 本函数设置值状态变量的值。可见性设置为public，以便外部帐户可以访问它。

function set(string memory \_value) public {

value = \_value;

}

}

（这个智能合约的功能是可以获取值和设置值）

使用truffle compile命令编译项目：

编译完成之后，再打开项目的根目录，可以发现多了build目录

这个文件是智能合约的ABI文件，代表“抽象二进制接口”，主要有以下两个作用：

A.作为可以在以太坊虚拟机（EVM）运行的可执行文件

B.包含智能合约函数的JSON表示，以便外部客户端调用这些函数

**4.更新配置文件**

修改网络配置连接到本地区块链网络（Ganache）

打开位于根目录下的truffle-config.js文件，修改内容如下：

module.exports = {

networks: {

development: {

host: "127.0.0.1", // ip地址

port: 8545, // 端口

network\_id: "\*" // Match any network id

}

},

solc: {

optimizer: {

enabled: true,

runs: 200

}

}

}

注意：ip地址，端口号等，应该与Ganache匹配

**5.创建迁移脚本**

在migrations目录中创建脚本2\_deploy\_contracts，这个脚本是告诉Truffle如何部署智能合约，一定要标序号，作用是让Truffle知道它们的执行顺序。

var MyContract = artifacts.require("./MyContract.sol");

// 用require将合约赋值给一个MyContract变量

module.exports = function(deployer) {

deployer.deploy(MyContract); // 部署该合约

};

执行迁移命令truffle migrate

部署成功，并消耗了以太币，使用truffle console（控制台）访问智能合约

控制台启动成功

MyContract.deployed().then((instance) => { app = instance })

输入指令，运行成功

接着试试调用get方法，输入app.get()

调用成功