2020 区块链大作业：第二阶段

个人信息

字段	值
学院与专业	数据科学与计算机学院软件工程专业
姓名	白家栋
学号	18342001

项目设计说明

根据需求，本次基于 **fisco-bcos** 设计的智能合约需要满足以下的四个需求:

- **功能一**：实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。
- **功能二**：实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。
- **功能三**：利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。
- **功能四**：应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

从背景设定以及需求中，我设计了以下的这两个角色:

一. 角色设计

角色1: Company

Company 在合约中的定义如下:

```
struct Company {
    // 公司名
    string name;
    // 公司地址，哈希值(在后端通过 get_account.sh 生成)
    address _addr;
    // 公司信用等级
    uint256 credit;
    // 公司资产
    uint256 asset;
}
```

同时，Company 还具有 **核心企业/非核心企业** 的区分，保存在一个 `mapping` 中，如下：

```
// 由公司地址到公司级别的映射，表明该公司是否是核心企业
    mapping(address => bool) is_upstream;
```

Company 能够完成如下的任务：

- **发起交易**：类比到供应链中即 **核心企业** 向下游企业采购货物等交易行为
- 创建**应收款账单**：类比到供应链中，即**核心企业**承诺给下游企业1000万来采购轮胎
- 与银行发起**融资**：类比到供应链中，即 **下游企业** 为了组装商品去找银行贷款，从而能够向其他公司购买原材料。

角色2: Bank

Bank 在合约中的定义如下:

```
    struct Bank {
        // 银行名
        string name;
        // 银行地址，哈希值
        address _addr;
    }
```

Bank 具有名称，地址的属性。银行默认资产是充足的，即有能力贷款给任何 Company. Bank 的主要职能是提供融资服务给所有企业，包括 **核心企业**, **下游企业** 等等。

二. 事件设计

除了角色之外，我还设计了3种发生在 Company 和 Bank 之间的事件，如下:

事件1: Bill

Bill 定义了企业之间的 **应收款账单**， Bill 的数据结构设计如下:

```
// 一笔应收账款单
    struct Bill {
        // 账单id
        uint256 bill_id;
        // 申请借钱的公司
        address borrow_company;
        // 借出钱的公司
        address lend_company;
        // 应收金额
        uint256 money;
        // 借款日期
        uint256 start_date;
        // 还款日期
        uint256 end_date;
        // 是否通过审核
        bool is_pass;
        // 是否还款
        bool is_finish;
        // 是否是核心企业的账单
        bool is_from_upstream;
        // 收款公司的信用评级变动
        uint256 credit_growth;
    }
```

对于应收款账单的应用场景，类比到 **供应链中**， 当核心企业，比如宝马向轮胎公司订购一批轮胎时，宝马作为 **核心企业** 就和下游的轮胎公司共同签发了一份 **Bill** ， 即应收账款账单。

同时，Bill 还具有后续的交易凭证的功能，比如轮胎公司能够通过其与宝马签订的 **Bill** 来从轮毂公司采购轮毂，这个过程通过将与宝马签订的 Bill 进行拆分完成的。拆分之后，当还款日期到达，宝马的还款方式从原有的按照最初的 **Bill** 上约定的金额付款给轮胎公司，变成了分别向轮胎公司和轮毂公司转账，这两笔转账之和与最初签订的 Bill 上的金额相等。

拆分 Bill 的函数实现如下:

```
    // 转让一部分应收账单
    function splitBillToAnotherCompany(
        address from, // 从哪家公司转让
        address to, // 转让到哪家公司
        uint256 bid, // 被拆分的账单
        uint256 money, // 转让金额
        uint256 timestamp // 当前时间
    ) public returns (Message memory) {
        Bill memory bill_to_be_split;
        bool flag = false;
        for (uint256 i = 0; i < bills.length; i++) {
            if (bid == bills[i].bill_id) {
                flag = true;
                bill_to_be_split = bills[i];
                break;
            }
        }
        if (!flag) {
            return Message(false, "该账单不存在");
        }

        if (bill_to_be_split.lend_company != from) {
            return Message(false, "该账单并不属于该公司");
        }

        if (bill_to_be_split.borrow_company == to) {
            return Message(false, "该账单不可被转让，因为接收者需要还款");
        }

        if (money > bill_to_be_split.money) {
            return Message(false, "转让金额大于了账单原有金额");
        }

        if (timestamp > bill_to_be_split.end_date) {
            return Message(false, "该账单已经过期");
        }

        if (!bill_to_be_split.is_pass) {
            return Message(false, "该账单还没有通过审核");
        }

        if (bill_to_be_split.is_finish) {
            return Message(false, "该账单已完成还款，不可再被拆分");
        }
        // 鉴权完成，开始转让

        // 首先新增一份订单
        addBill(
            bill_to_be_split.borrow_company,
            to,
            money,
            timestamp,
            bill_to_be_split.end_date,
            false,
            false,
            bill_to_be_split.is_from_upstream,
            0
        );

        // 然后调整原有的账单金额
        adjustMoneyForBill(bid, bill_to_be_split.money - money);
        return Message(true, "转让完成, 等待审核中");
    }
```

当**核心企业** 与 **下游企业** 签订应还账款账单时，下游企业的信用值会根据借款金额增加，增加的规则如下:

```
    // 计算转账后增加的信用等级
    function calculateCredit(
        address borrow_company, // 发起采购/借款的公司
        uint256 money, // 金额
        bool is_from_upstream // 是否来自于核心企业
    ) public view returns (uint256 growth) {
        if (!is_from_upstream) {
            return 0;
        }
        Company memory bc = getCompanyByAddress(borrow_company);
        uint256 prev_credit = bc.credit;
        return prev_credit / 10 + money / 50;
    }
```

事件2: Deal

Deal 是整个供应链中的主体事件，车企向轮胎公司购买轮胎，轮胎公司向轮毂公司购买轮毂都是依靠 Deal 来完成的。

Deal 的数据结构定义如下:

```
    // 交易
    struct Deal {
        // 交易id
        uint256 deal_id;
        // 出钱的一方，买家
        address buyer_company;
        // 出货的一方，卖家
        address seller_company;
        // 交易发起开始日期
        uint256 start_date;
        // 交易是否完成
        bool is_finish;
        // 购买商品描述
        string merchant;
        // 购买金额
        uint256 money;
    }
```

基于 Deal，如果企业之间要发起交易，可以直接通过 `submitDeal` 函数发起。但是交易的能否通过，即 `is_finish` 字段何时设置为 `true` 是有条件的，这个条件的判断方法为:

- 如果 `buyer_company` 为核心企业，则可以通过

- 如果 `buyer_company` 非核心企业，但是 `asset` 充足，也可以通过
- 如果 `buyer_company` 非核心企业，但是其持有 `money` 大于等于**Deal**所要求的购买金额的 **Bill**，也可以通过
- 若上述条件都不满足，则交易无法被通过。

上述的条件判断的实现如下:

```
    function canBuyerSubmitDeal(address buyer_company, uint256 money)
        public
        view
        returns (bool can)
    {
        // 如果买家是核心企业，则可以交易
        if (is_upstream[buyer_company]) {
            return true;
        }

        // 如果买家的资金充足，也可以交易
        if (assets[buyer_company] >= money) {
            return true;
        }

        // 如果买家持有面值较大的账单，也可以交易
        if (canBuyerDealByBill(buyer_company, money)) {
            return true;
        }

        return false;
    }
```

事件3: Loan

Loan 事件是企业向银行贷款时主要使用到的载体，Loan 的数据结构定义如下:

```
    // 一笔向银行的融资
    struct Loan {
        // 融资账单id
        uint256 loan_id;
        // 申请融资的公司
        address request_company;
        // 银行地址
        address bank_address;
        // 发起日期
        uint256 start_date;
        // 还款日期
        uint256 end_date;
        // 是否通过审核
        bool is_pass;
        // 是否还款
        bool is_finish;
        // 融资金额
        uint256 money;
    }
```

发起融资请求后，银行会根据企业的**信用度**和**现有资产**进行贷款资格审核，具体审核规则如下:

- 如果企业的资产大于等于其贷款金额，则可以通过
- 如果企业的资产与其信用值乘50的和大于其贷款金额，则可以通过

实现如下:

```
// 判断是否银行是否能为该公司提供贷款
    function canLoanToThisCompany(
        address company,
        address bank_address,
        uint256 current_credit,
        uint256 money
    ) public returns (bool can) {
        Company memory c = getCompanyByAddress(company);

        // 若贷款公司的资产大于其要借的钱数，则直接同意
        if (c.asset >= money) {
            return true;
        }

        // 若贷款公司的资产与其信用值乘50大于要借的钱数，也直接同意
        if (c.asset + current_credit * 50 >= money) {
            return true;
        }
        return false;
    }
```

功能测试文档

下面的测试方法是基于我在热身报告中搭建的融合 `nodejs-sdk` 的后端服务的，故如果要复现，可以参考我的后端项目的地址.

1. 合约部署

请求

方法	URL	请求体
POST	/contract	{"contract": "SupplyChain","parameters": []}

响应

```
{
    "status": "OK",
    "msg": "部署成功",
    "data": {
        "status": "0x0",
        "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
        "transactionHash":
"0x4df21db4cb28fc3ec87bfe90098f330747d083ba1cb7c770afebfb3e37488b53"
    },
    "time": "2020-12-08T09:05:53.699Z"
}
```

2. 准备账号

下面的账号已经在配置文件中提示准备好

- **注册**
  - 注册宝马
    - 请求

方法	URL	请求体
POST	/transaction	{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "registerCompany","parameters["0x27a28f09ec7accce2eecc27ebcd9453226ed3e52", "宝马"]}

- 响应

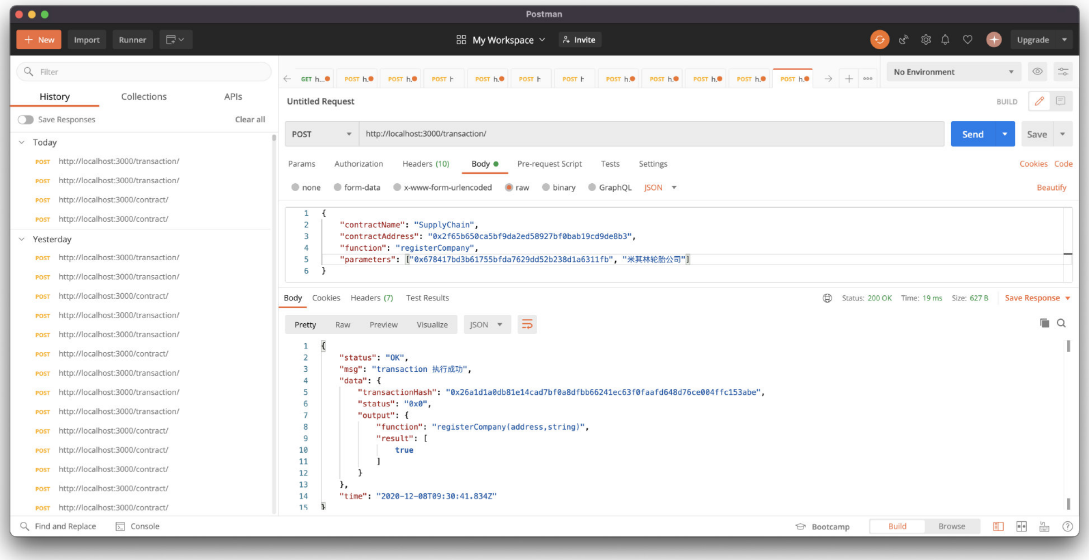
```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0xb24d8985ebf7b608f1ebef5a4da5b426fdaae084947c0f405ef0ad473ec8fc51",
        "status": "0x0",
        "output": {
            "function": "registerCompany(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:25:32.088Z"
}
```

- 注册米其林
    - 请求

方法	URL	请求体
POST	/transaction	{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "registerCompany","parameters: ["0x678417bd3b61755bfda7629dd52b238d1a6311fb", "米其林"]}

    - 响应

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0x26a1d1a0db81e14cad7bf0a8dfbb66241ec63f0faafd648d76ce004ffc153abe",
        "status": "0x0",
        "output": {
            "function": "registerCompany(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:30:41.834Z"
}
```



POST http://localhost:3000/transaction/

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "registerCompany",
    "parameters": ["0x678417bd3b61755bfda7629dd52b238d1a6311fb", "米其林轮胎公司"]
}
```

Status: 200 OK Time: 19 ms Size: 627 B

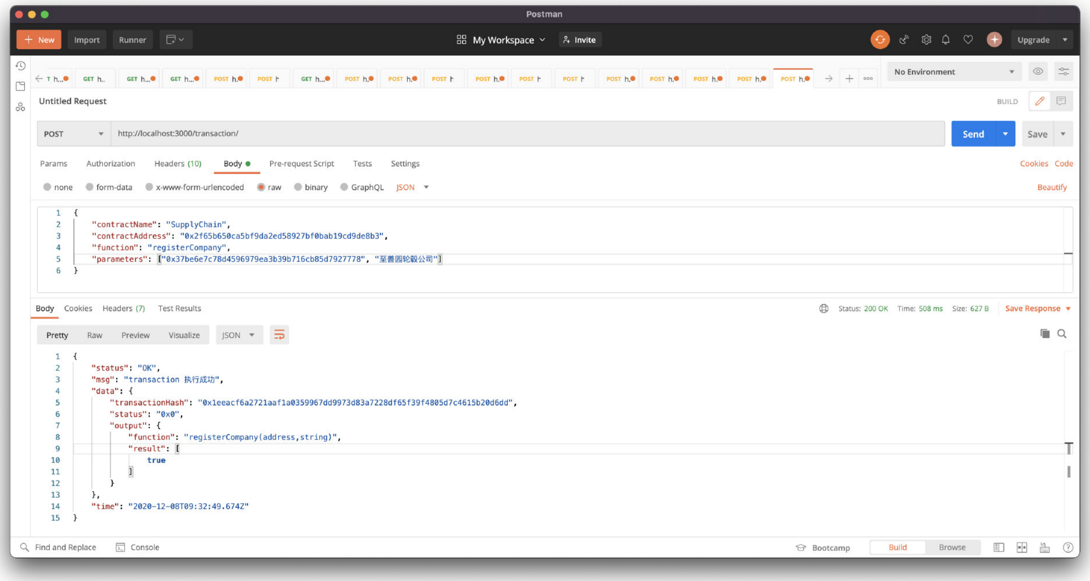
```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x26a1d1a0db81e14cad7bf0a8dfbb66241ec63f0faafd648d76ce004ffc153abe",
        "status": "0x0",
        "output": {
            "function": "registerCompany(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:30:41.834Z"
}
```

- 注册至善园轮毂公司
  - 请求

方法	URL	请求体
POST	/transaction	{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "registerCompany","parameters: ["0x678417bd3b61755bfda7629dd52b238d1a6311fb", "至善园轮毂公司"]}

  - 响应

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0x1eeacf6a2721aaf1a0359967dd9973d83a7228df65f39f4805d7c4615b20d6dd",
        "status": "0x0",
        "output": {
            "function": "registerCompany(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:32:49.674Z"
}
```



Untitled Request

POST http://localhost:3000/transaction/

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "registerCompany",
    "parameters": ["0x37be6e7c78d4596979ea3b39b716cb85d7927778", "至善园轮毂公司"]
}
```

Status: 200 OK Time: 508 ms Size: 627 B

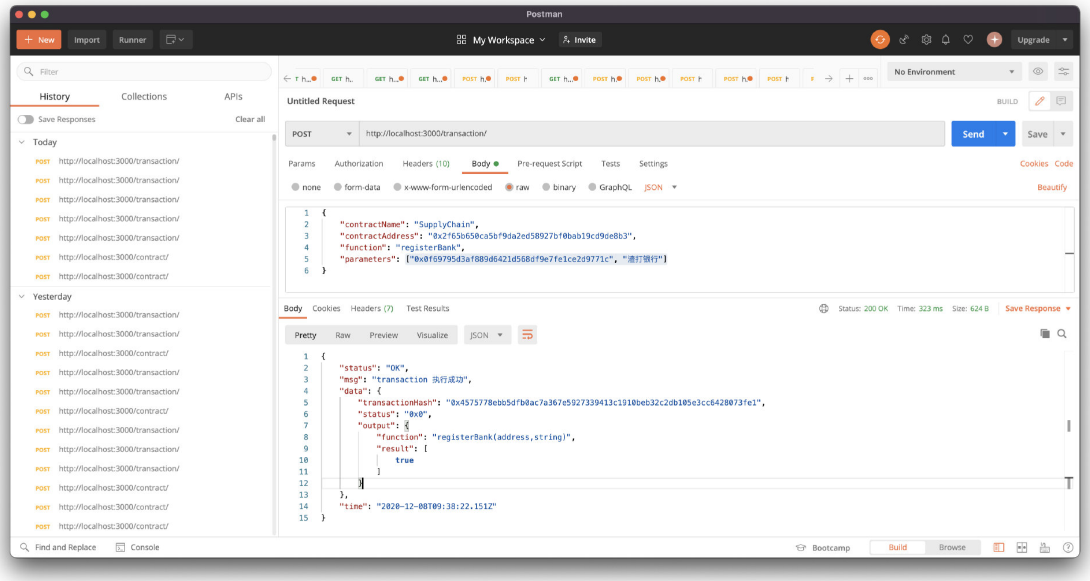
```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x1eeacf6a2721aaf1a0359967dd9973d83a7228df65f39f4805d7c4615b20d6dd",
        "status": "0x0",
        "output": {
            "function": "registerCompany(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:32:49.674Z"
}
```

- 注册渣打银行
  - 请求

方法	URL	请求体
POST	/transaction	{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "registerBank","parameters: ["0x0f69795d3af889d6421d568df9e7fe1ce2d9771c", "渣打银行"]}

  - 响应

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0x4575778ebb5dfb0ac7a367e5927339413c1910beb32c2db105e3cc6428073fe1",
        "status": "0x0",
        "output": {
            "function": "registerBank(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:38:22.151Z"
}
```


POST http://localhost:3000/transaction/

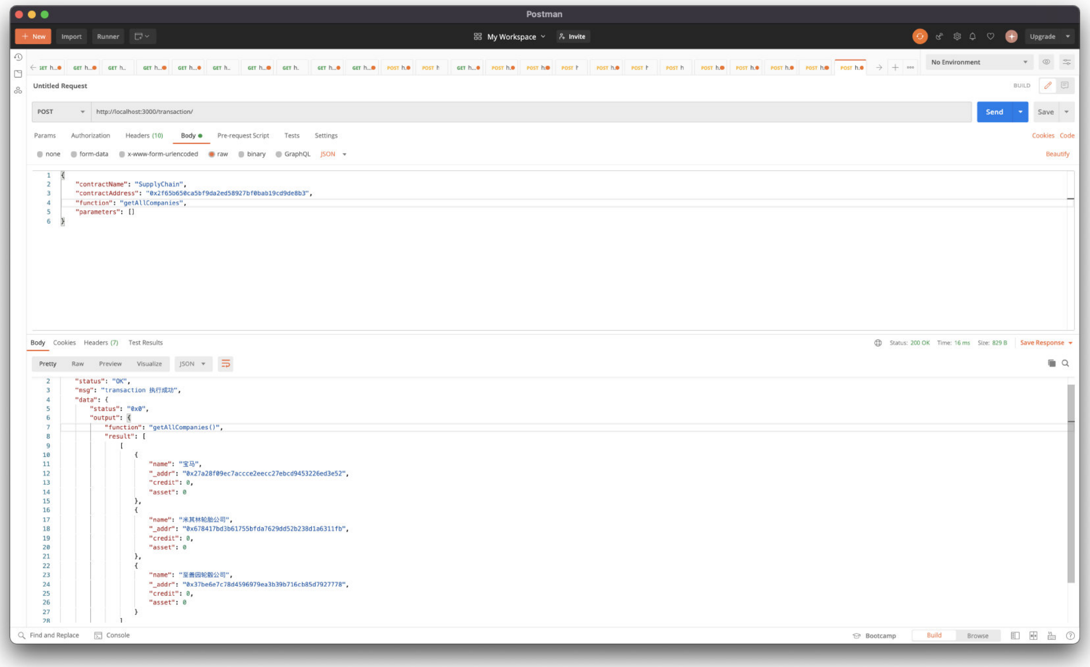
Body: raw JSON

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "registerBank",
    "parameters": ["0x0f69795d3af889d6421d568df9e7fe1ce2d9771c", "渣打银行"]
}
```

Status: 200 OK Time: 323 ms Size: 624 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x4575778ebb5dfb0ac7a367e5927339413c1910beb32c2db105e3cc6428073fe1",
        "status": "0x0",
        "output": {
            "function": "registerBank(address,string)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:38:22.151Z"
}
```

- **检查注册结果**
    - 公司


POST http://localhost:3000/transaction/

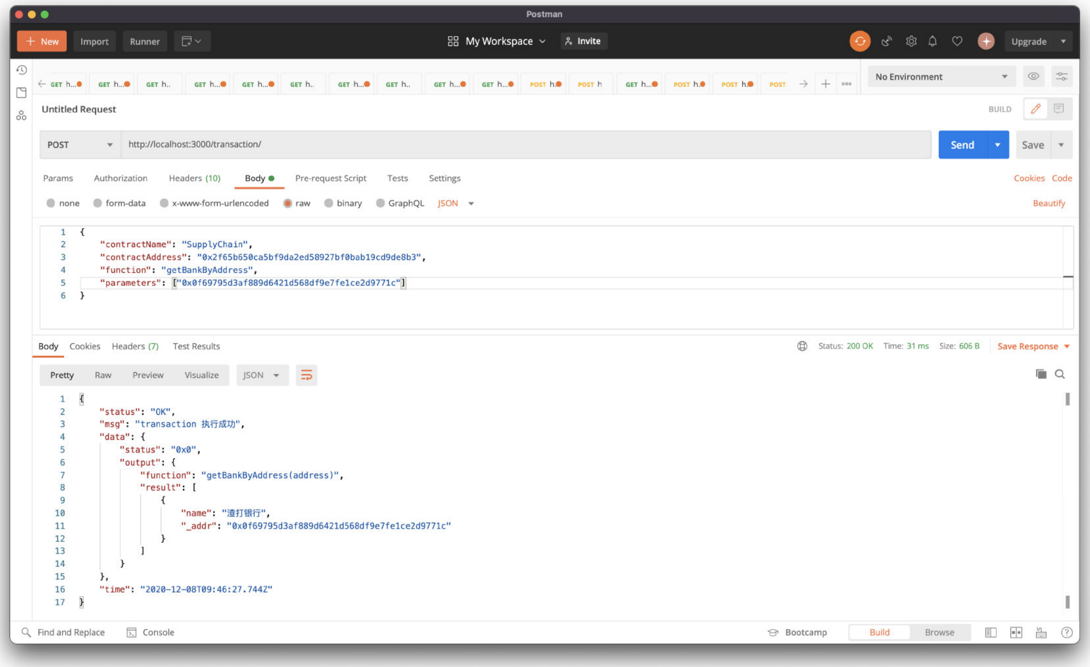
Body: raw JSON

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "getAllCompanies",
    "parameters": []
}
```

Status: 200 OK Time: 16 ms Size: 829 B

```
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getAllCompanies()",
            "result": [
                [
                    {
                        "name": "宝马",
                        "_addr": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                        "credit": 0,
                        "asset": 0
                    },
                    {
                        "name": "米其林轮胎公司",
                        "_addr": "0x678417bd3b61755bfda7629dd52b238d1a6311fb",
                        "credit": 0,
                        "asset": 0
                    },
                    {
                        "name": "至善园轮毂公司",
                        "_addr": "0x37be6e7c78d4596979ea3b39b716cb85d7927778",
                        "credit": 0,
                        "asset": 0
                    }
                ]
```

    - 银行

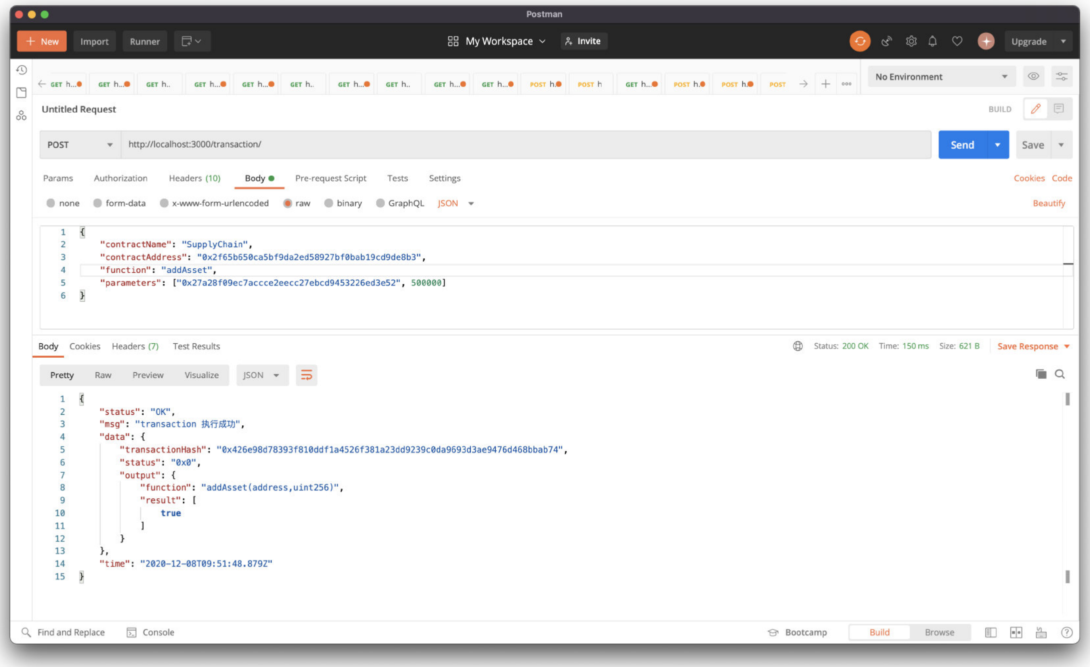

POST http://localhost:3000/transaction/

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "getBankByAddress",
    "parameters": ["0x0f69795d3af889d6421d568df9e7fe1ce2d9771c"]
}
```

Status: 200 OK Time: 31 ms Size: 606 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getBankByAddress(address)",
            "result": [
                {
                    "name": "渣打银行",
                    "_addr": "0x0f69795d3af889d6421d568df9e7fe1ce2d9771c"
                }
            ]
        }
    },
    "time": "2020-12-08T09:46:27.744Z"
}
```

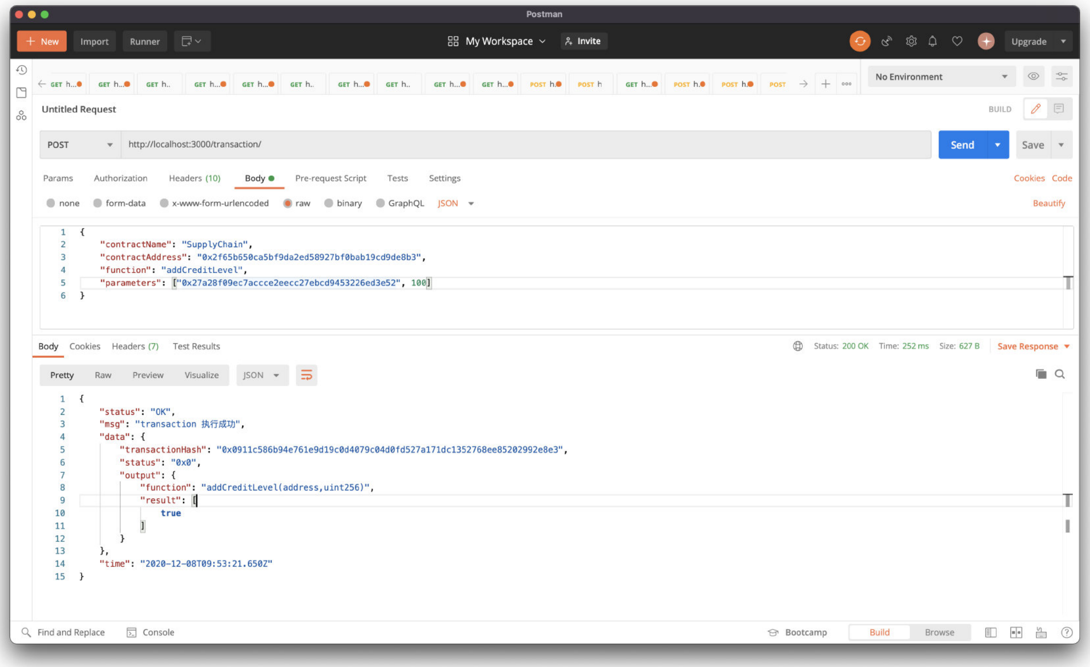
- 初始化资产以及信用
    - **给宝马增加 500000 的资产，100 的信用**


POST http://localhost:3000/transaction/

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "addAsset",
    "parameters": ["0x27a28f09ec7accce2eecc27ebcd9453226ed3e52", 500000]
}
```

Status: 200 OK Time: 150 ms Size: 621 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x426e98d78393f810ddf1a4526f381a23dd9239c0da9693d3ae9476d468bbab74",
        "status": "0x0",
        "output": {
            "function": "addAsset(address,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:51:48.879Z"
}
```



POST http://localhost:3000/transaction/

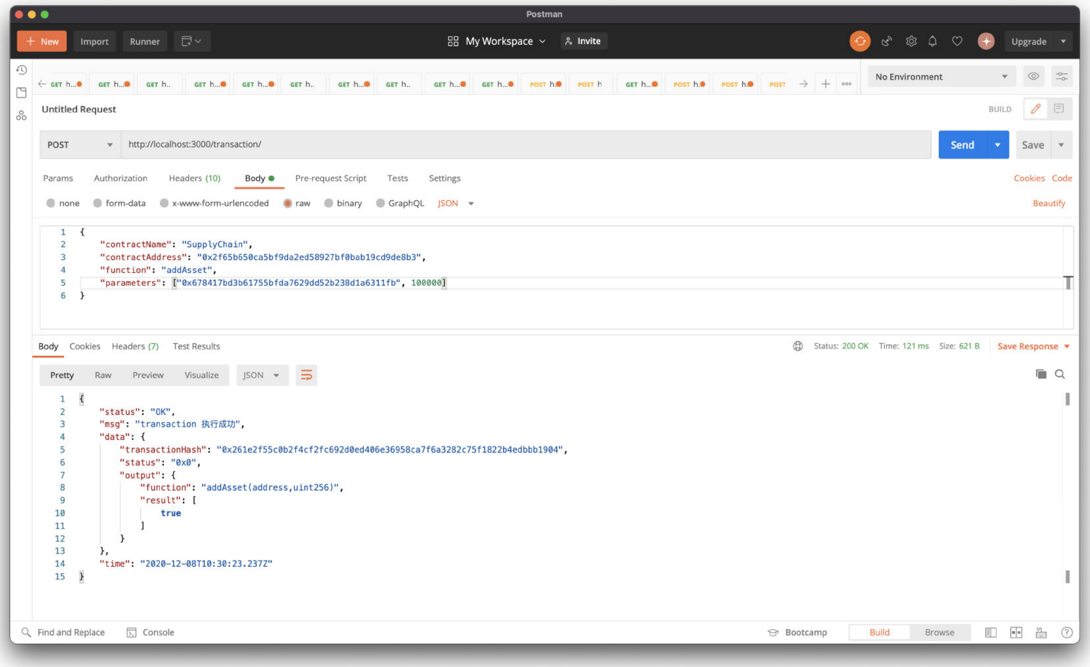
Body: raw JSON

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "addCreditLevel",
    "parameters": ["0x27a28f09ec7accce2eecc27ebcd9453226ed3e52", 100]
}
```

Status: 200 OK Time: 252 ms Size: 627 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x0911c586b94e761e9d19c0d4079c04d0fd527a171dc1352768ee85202992e8e3",
        "status": "0x0",
        "output": {
            "function": "addCreditLevel(address,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T09:53:21.650Z"
}
```

- 给米其林轮胎增加 100000 的资产，10 的信用



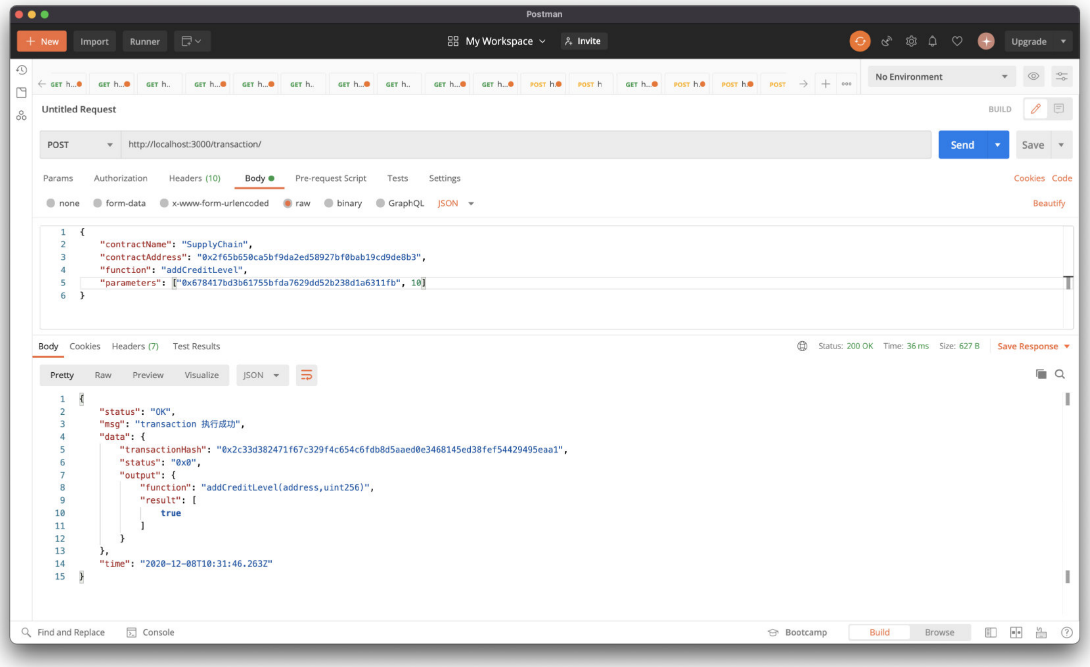
POST http://localhost:3000/transaction/

Body: raw JSON

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "addAsset",
    "parameters": ["0x678417bd3b61755bfda7629dd52b238d1a6311fb", 100000]
}
```

Status: 200 OK Time: 121 ms Size: 621 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x261e2f55c0b2f4cf2fc692d0ed406e36958ca7f6a3282c75f1822b4edbbb1904",
        "status": "0x0",
        "output": {
            "function": "addAsset(address,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T10:30:23.237Z"
}
```



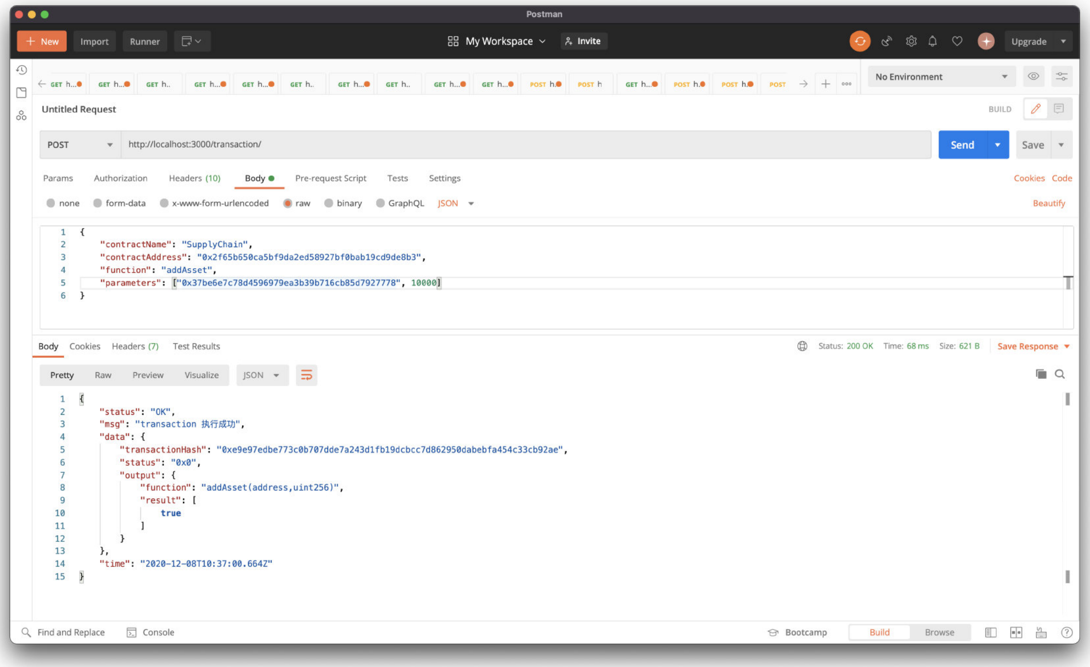
POST http://localhost:3000/transaction/

Status: 200 OK Time: 36 ms Size: 627 B

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "addCreditLevel",
    "parameters": ["0x678417bd3b61755bfda7629dd52b238d1a6311fb", 10]
}
```

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x2c33d382471f67c329f4c654c6fdb8d5aaed0e3468145ed38fef54429495eaa1",
        "status": "0x0",
        "output": {
            "function": "addCreditLevel(address,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T10:31:46.263Z"
}
```

- **给至善园轮毂公司增加10000的资产，5的信用**



POST http://localhost:3000/transaction/

Status: 200 OK Time: 68 ms Size: 621 B

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "addAsset",
    "parameters": ["0x37be6e7c78d4596979ea3b39b716cb85d7927778", 10000]
}
```

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0xe9e97edbe773c0b707dde7a243d1fb19dcbcc7d862950dabebfa454c33cb92ae",
        "status": "0x0",
        "output": {
            "function": "addAsset(address,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T10:37:00.664Z"
}
```

3. 交易测试

下面我们分别根据需求对于合约进行测试

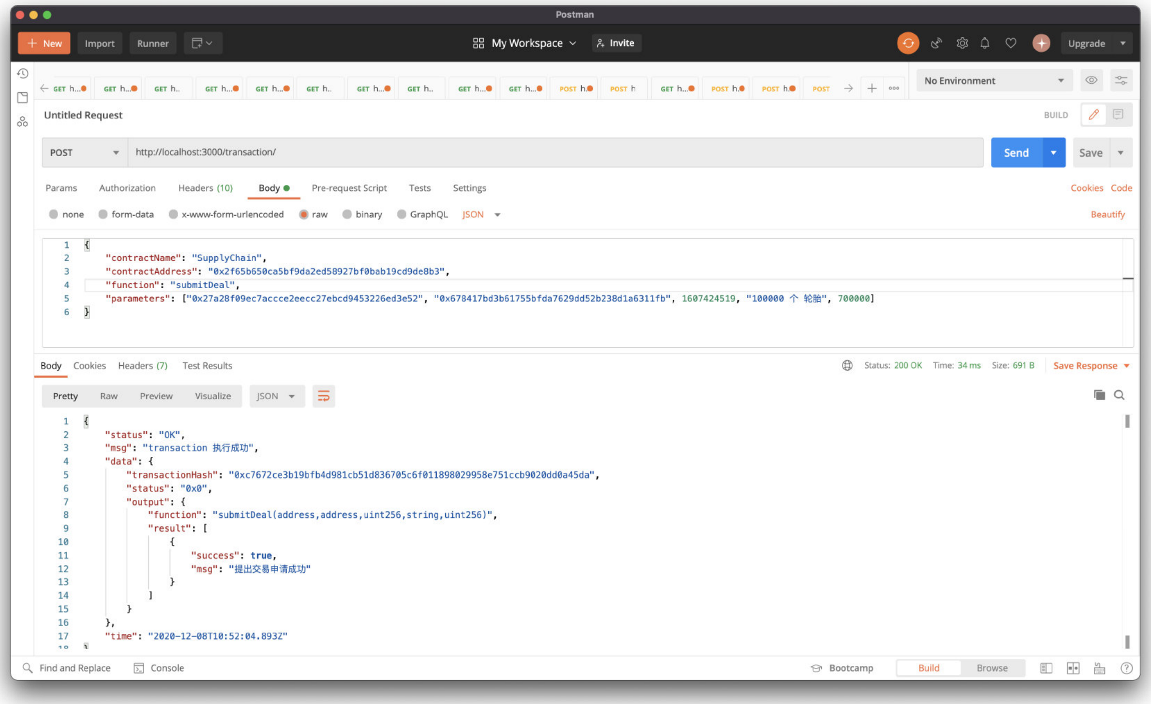
测试1: 采购商品

操作： **宝马公司** 向 **米其林轮胎公司** 订购 10万个轮胎，交易金额为 700000

- 请求

方法	url	请求体
POST	/transaction	{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "submitDeal","parameters": ["0x27a28f09ec7accce2eecc27ebcd9453226ed3e52", "0x678417bd3b61755bfda7629dd52b238d1a6311fb", 1607424519, "100000 个 轮胎", 700000]

- 响应

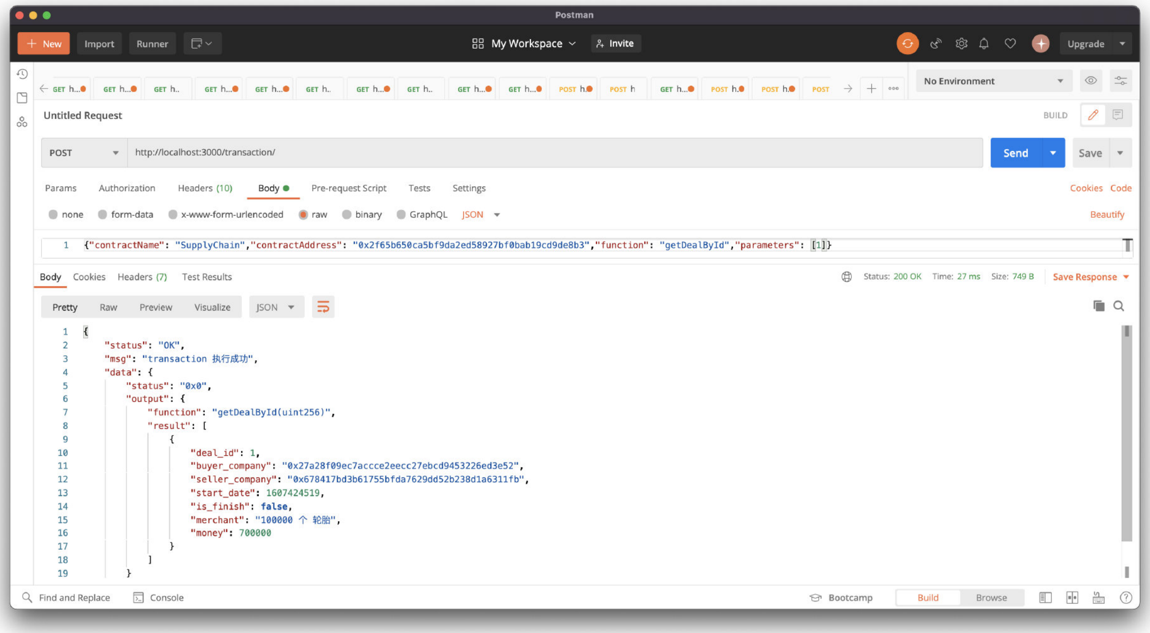


POST http://localhost:3000/transaction/

```
{
    "contractName": "SupplyChain",
    "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3",
    "function": "submitDeal",
    "parameters": ["0x27a28f09ec7accce2eecc27ebcd9453226ed3e52", "0x678417bd3b61755bfda7629dd52b238d1a6311fb", 1607424519, "100000 个 轮胎", 700000]
}
```

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0xc7672ce3b19bfb4d981cb51d836705c6f011898029958e751ccb9020dd0a45da",
        "status": "0x0",
        "output": {
            "function": "submitDeal(address,address,uint256,string,uint256)",
            "result": [
                {
                    "success": true,
                    "msg": "提出交易申请成功"
                }
            ]
        }
    },
    "time": "2020-12-08T10:52:04.893Z"
```

- **查询Deal来验证 Deal 成功产生**

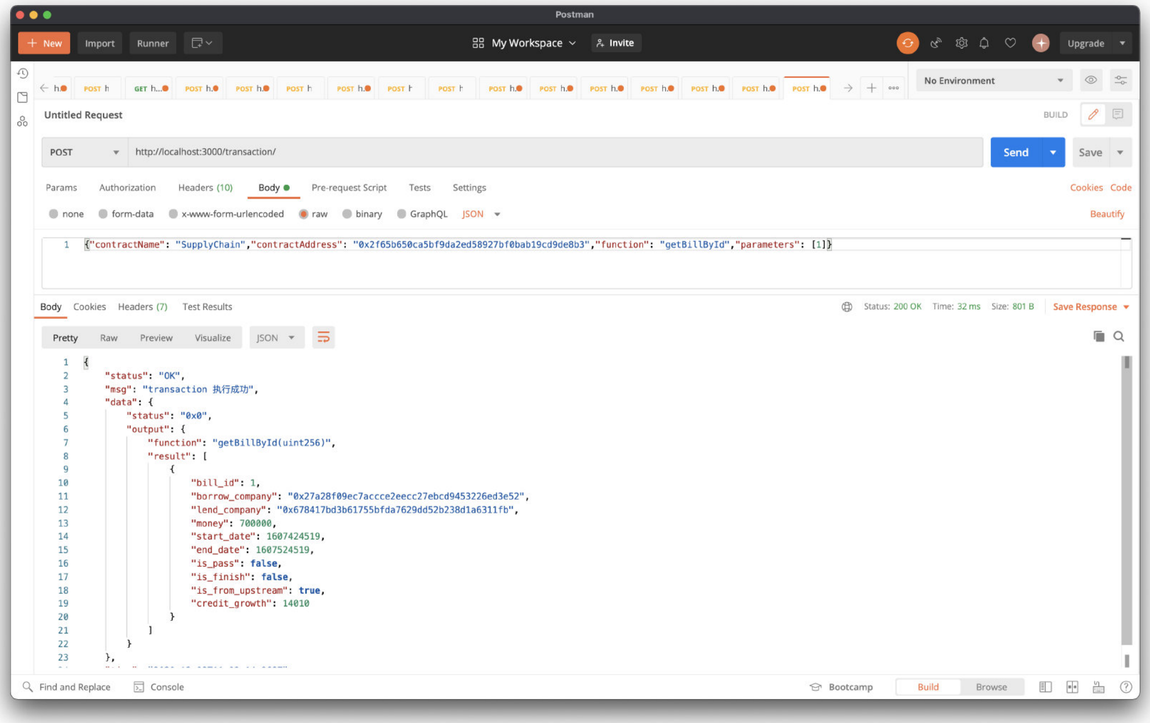


POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "getDealById","parameters": [1]}
```

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getDealById(uint256)",
            "result": [
                {
                    "deal_id": 1,
                    "buyer_company": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                    "seller_company": "0x678417bd3b61755bfda7629dd52b238d1a6311fb",
                    "start_date": 1607424519,
                    "is_finish": false,
                    "merchant": "100000 个 轮胎",
                    "money": 700000
                }
            ]
        }
```

- **查询Bill来验证应收账款账单成功产生**



POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain","contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "getBillById","parameters": [1]}
```

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getBillById(uint256)",
            "result": [
                {
                    "bill_id": 1,
                    "borrow_company": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                    "lend_company": "0x678417bd3b61755bfda7629dd52b238d1a6311fb",
                    "money": 700000,
                    "start_date": 1607424519,
                    "end_date": 1607524519,
                    "is_pass": false,
                    "is_finish": false,
                    "is_from_upstream": true,
                    "credit_growth": 14010
                }
            ]
        }
    },
```

测试2: 应收账款的转让上链

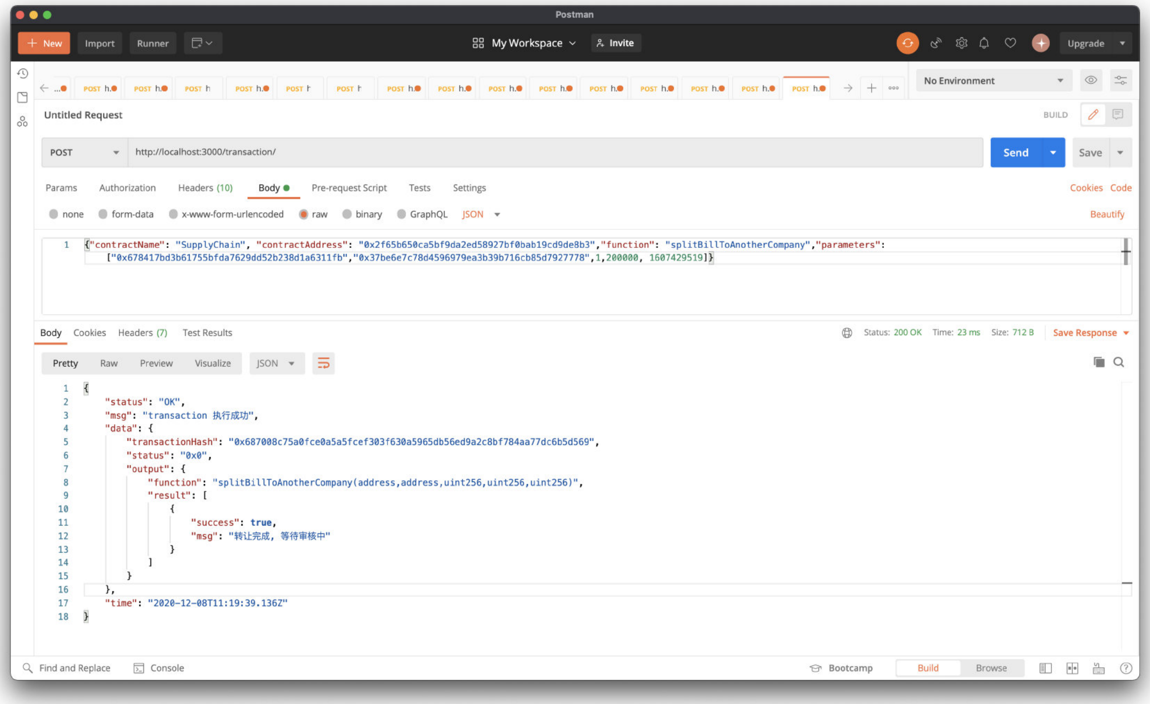
操作：**米其林轮胎公司**将与**宝马公司**之间的账单转让**200000**的金额给**至善园轮毂公司**

- 请求

方法	url	请求体
POST	/transaction	{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "splitBillToAnotherCompany","parameters": ["0x678417bd3b61755bfda7629dd52b238d1a6311fb","0x37be6e7c78d4596979ea3b39b716cb85d7927778",1,200000, 1607429519]}

- 响应

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0x687008c75a0fce0a5a5fcef303f630a5965db56ed9a2c8bf784aa77dc6b5d569",
        "status": "0x0",
        "output": {
            "function":
"splitBillToAnotherCompany(address,address,uint256,uint256,uint256)",
            "result": [
                {
                    "success": true,
                    "msg": "转让完成, 等待审核中"
                }
            ]
        }
    },
    "time": "2020-12-08T11:19:39.136Z"
}
```

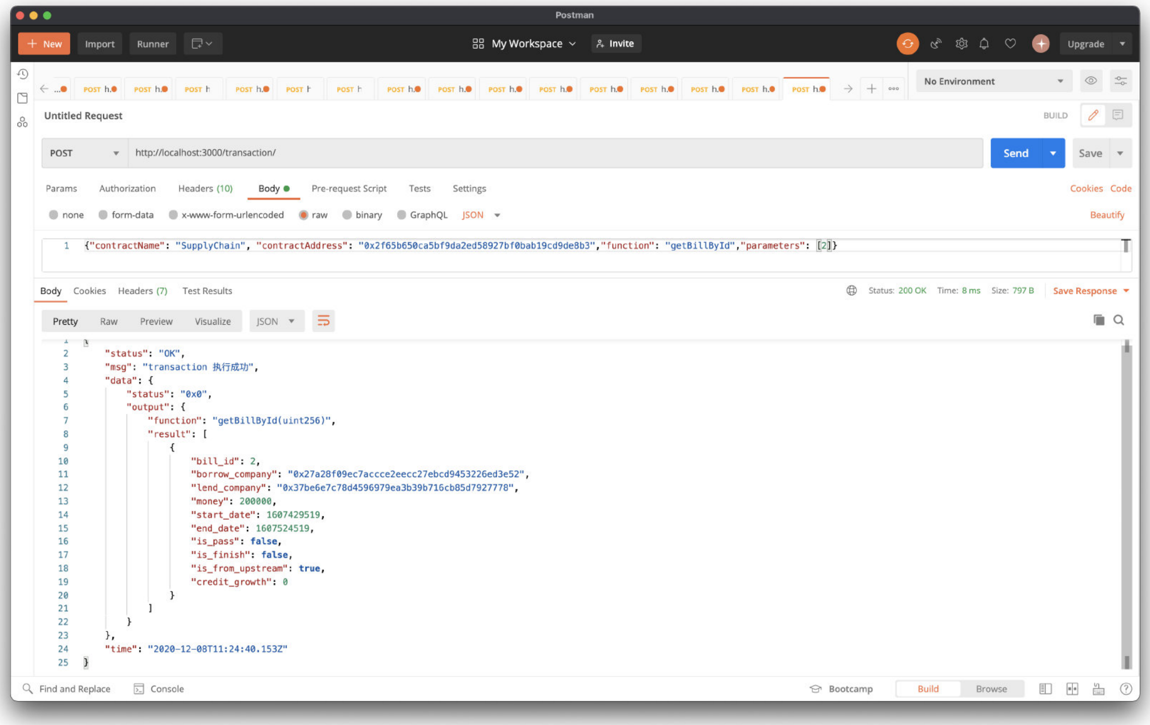

POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "splitBillToAnotherCompany","parameters":
    ["0x678417bd3b61755bfda7629dd52b238d1a6311fb","0x37be6e7c78d4596979ea3b39b716cb85d7927778",1,200000, 1607429519]}
```

Status: 200 OK Time: 23 ms Size: 712 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0x687008c75a0fce0a5a5fcef303f630a5965db56ed9a2c8bf784aa77dc6b5d569",
        "status": "0x0",
        "output": {
            "function": "splitBillToAnotherCompany(address,address,uint256,uint256,uint256)",
            "result": [
                {
                    "success": true,
                    "msg": "转让完成, 等待审核中"
                }
            ]
        }
    },
    "time": "2020-12-08T11:19:39.136Z"
}
```

查询id为2的Bill，来验证成功转让


POST http://localhost:3000/transaction/

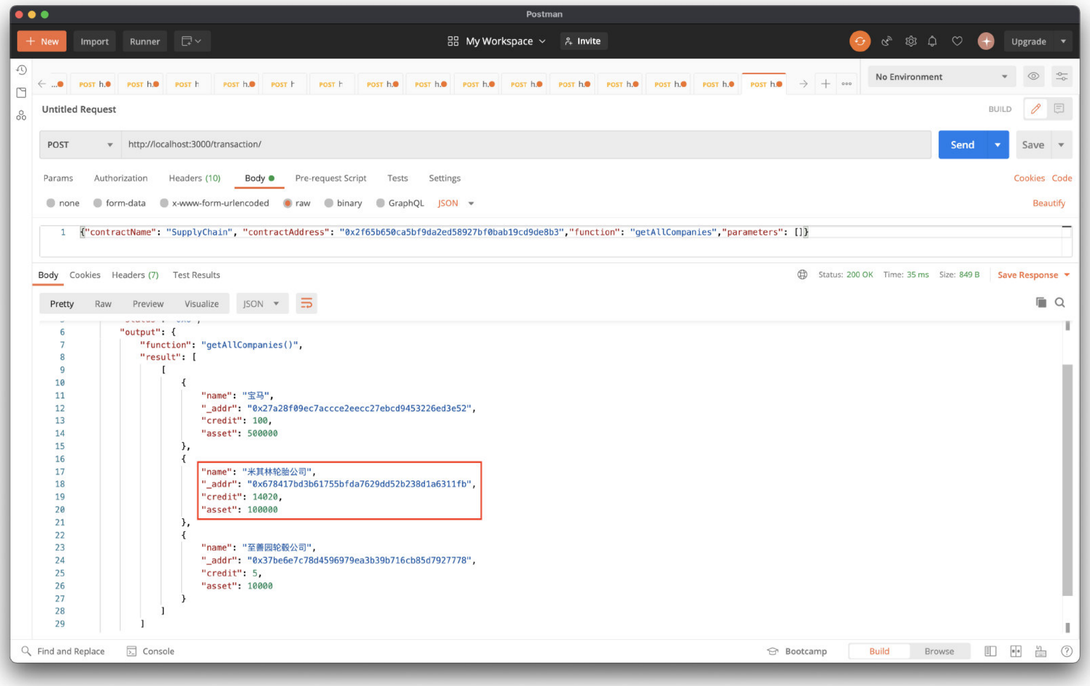
```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "getBillById","parameters": [2]}
```

Status: 200 OK Time: 8 ms Size: 797 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getBillById(uint256)",
            "result": [
                {
                    "bill_id": 2,
                    "borrow_company": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                    "lend_company": "0x37be6e7c78d4596979ea3b39b716cb85d7927778",
                    "money": 200000,
                    "start_date": 1607429519,
                    "end_date": 1607524519,
                    "is_pass": false,
                    "is_finish": false,
                    "is_from_upstream": true,
                    "credit_growth": 0
                }
            ]
        }
    },
    "time": "2020-12-08T11:24:40.153Z"
}
```

测试3: 利用应收账款向银行融资上链

- 操作: 由于持有**宝马公司**的应收款账单，**米其林轮胎公司的信用值上升**，**能够找银行融资更多钱**
    - 查询米其林公司，验证其信用值上升



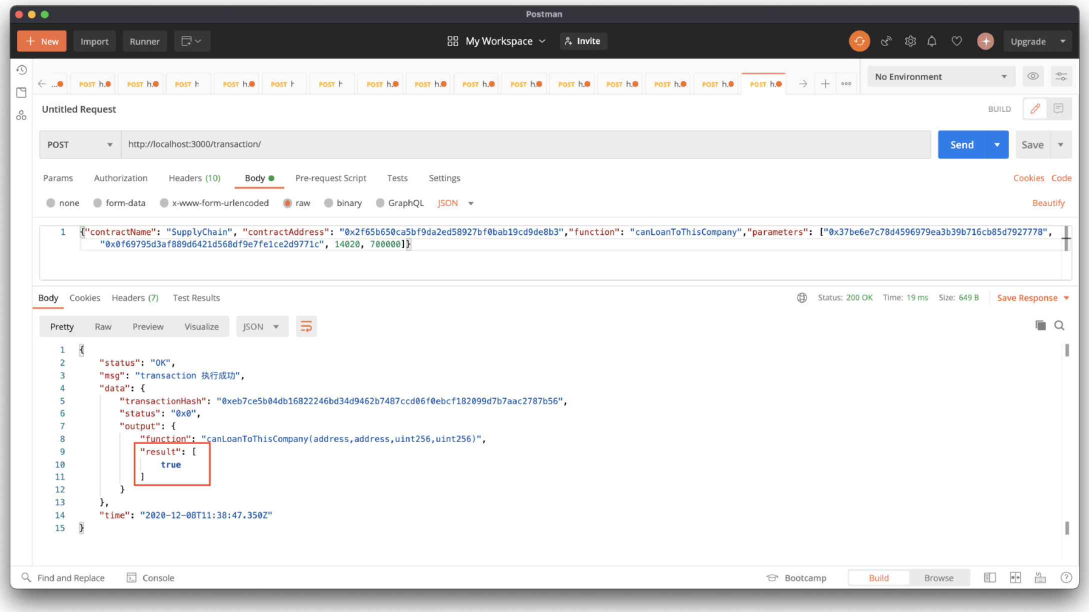
POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "getAllCompanies","parameters": []}
```

Status: 200 OK Time: 35 ms Size: 849 B

```
        "output": {
            "function": "getAllCompanies()",
            "result": [
                [
                    {
                        "name": "宝马",
                        "_addr": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                        "credit": 100,
                        "asset": 500000
                    },
                    {
                        "name": "米其林轮胎公司",
                        "_addr": "0x678417bd3b61755bfda7629dd52b238d1a6311fb",
                        "credit": 14020,
                        "asset": 100000
                    },
                    {
                        "name": "至善园轮毂公司",
                        "_addr": "0x37be6e7c78d4596979ea3b39b716cb85d7927778",
                        "credit": 5,
                        "asset": 10000
                    }
                ]
            ]
```

- 向**渣打银行**验证融资金额能否到达 **700000**



POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "canLoanToThisCompany","parameters": ["0x37be6e7c78d4596979ea3b39b716cb85d7927778",
    "0x0f69795d3af889d6421d568df9e7fe1ce2d9771c", 14020, 700000]}
```

Status: 200 OK Time: 19 ms Size: 649 B

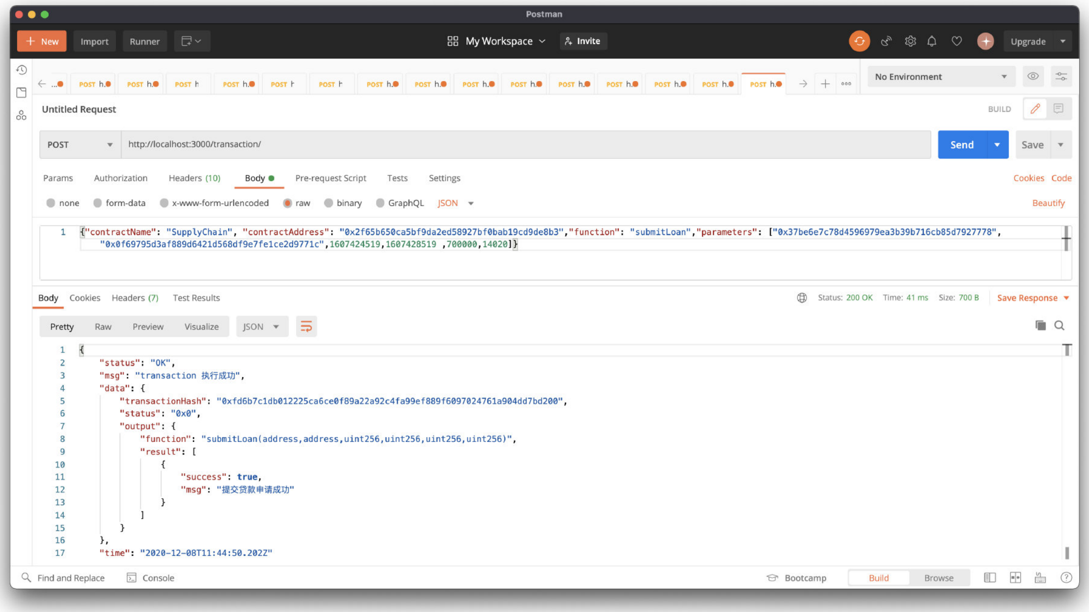
```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0xeb7ce5b04db16822246bd34d9462b7487ccd06f0ebcf182099d7b7aac2787b56",
        "status": "0x0",
        "output": {
            "function": "canLoanToThisCompany(address,address,uint256,uint256)",
            "result": [
                true
            ]
        }
    },
    "time": "2020-12-08T11:38:47.350Z"
}
```

- 向渣打银行申请 **700000** 的融资
    - 请求

方法	url	请求体
POST	/transaction	{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "submitLoan","parameters": ["0x37be6e7c78d4596979ea3b39b716cb85d7927778", "0x0f69795d3af889d6421d568df9e7fe1ce2d9771c",1607424519,1607428519 ,700000,14020]}

- 响应

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash":
"0xfd6b7c1db012225ca6ce0f89a22a92c4fa99ef889f6097024761a904dd7bd200",
        "status": "0x0",
        "output": {
            "function":
"submitLoan(address,address,uint256,uint256,uint256,uint256)",
            "result": [
                {
                    "success": true,
                    "msg": "提交贷款申请成功"
                }
            ]
        }
    },
    "time": "2020-12-08T11:44:50.202Z"
}
```



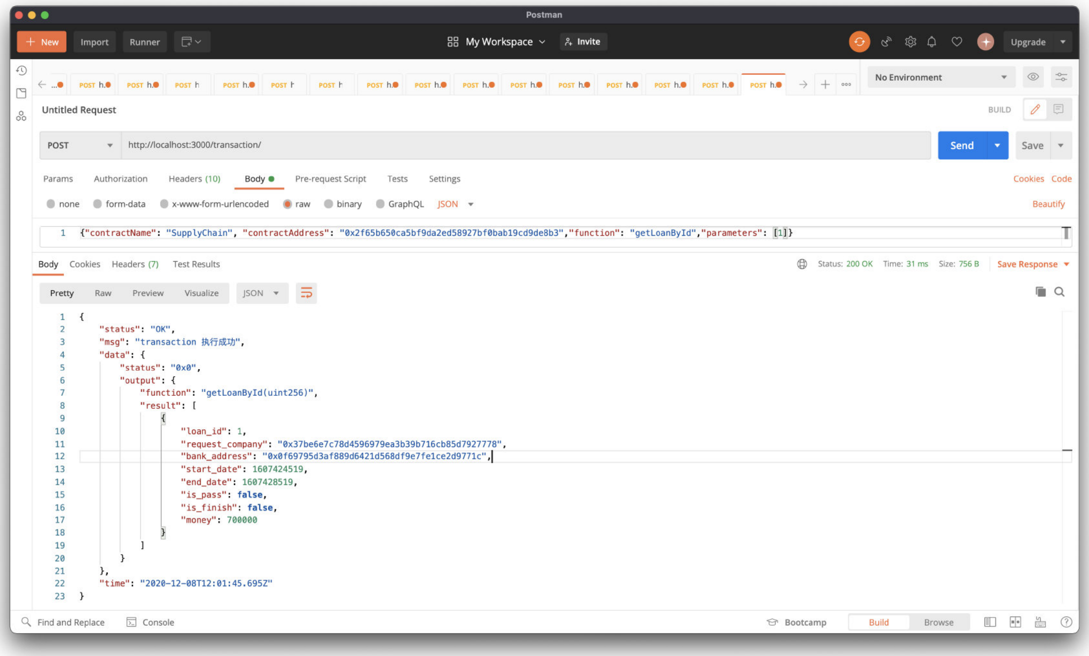
POST http://localhost:3000/transaction/

```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "submitLoan","parameters": ["0x37be6e7c78d4596979ea3b39b716cb85d7927778",
    "0x0f69795d3af889d6421d568df9e7fe1ce2d9771c",1607424519,1607428519 ,700000,14020]}
```

Status: 200 OK Time: 41 ms Size: 700 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "transactionHash": "0xfd6b7c1db012225ca6ce0f89a22a92c4fa99ef889f6097024761a904dd7bd200",
        "status": "0x0",
        "output": {
            "function": "submitLoan(address,address,uint256,uint256,uint256,uint256)",
            "result": [
                {
                    "success": true,
                    "msg": "提交贷款申请成功"
                }
            ]
        }
    },
    "time": "2020-12-08T11:44:50.202Z"
```

- **查询融资账单详情，验证申请融资成功**



POST http://localhost:3000/transaction/

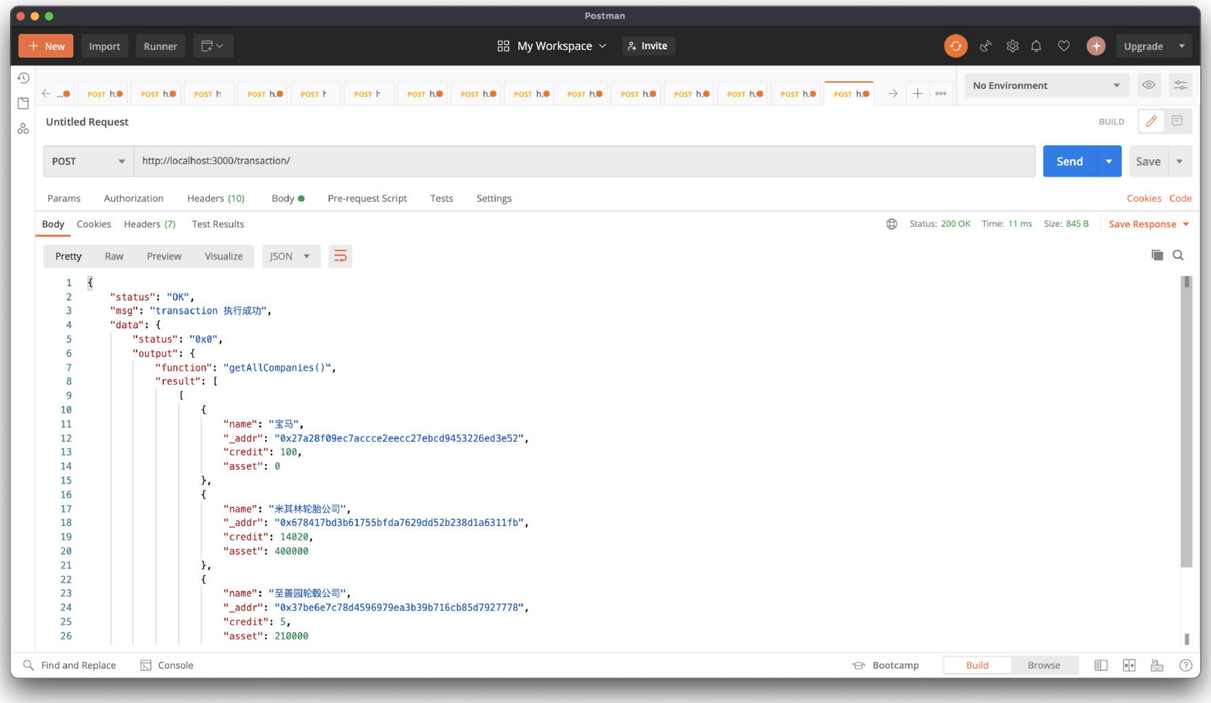
```
{"contractName": "SupplyChain", "contractAddress": "0x2f65b650ca5bf9da2ed58927bf0bab19cd9de8b3","function": "getLoanById","parameters": [1]}
```

Status: 200 OK Time: 31 ms Size: 756 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getLoanById(uint256)",
            "result": [
                {
                    "loan_id": 1,
                    "request_company": "0x37be6e7c78d4596979ea3b39b716cb85d7927778",
                    "bank_address": "0x0f69795d3af889d6421d568df9e7fe1ce2d9771c",
                    "start_date": 1607424519,
                    "end_date": 1607428519,
                    "is_pass": false,
                    "is_finish": false,
                    "money": 700000
                }
            ]
        }
    },
    "time": "2020-12-08T12:01:45.695Z"
}
```

测试4: 应收账款支付结算上链

结算后，宝马的资产应当为0，米其林轮胎公司和至善园轮毂公司的资产应当分别增加到400000和210000



POST http://localhost:3000/transaction/

Status: 200 OK Time: 11 ms Size: 845 B

```
{
    "status": "OK",
    "msg": "transaction 执行成功",
    "data": {
        "status": "0x0",
        "output": {
            "function": "getAllCompanies()",
            "result": [
                [
                    {
                        "name": "宝马",
                        "_addr": "0x27a28f09ec7accce2eecc27ebcd9453226ed3e52",
                        "credit": 100,
                        "asset": 0
                    },
                    {
                        "name": "米其林轮胎公司",
                        "_addr": "0x678417bd3b61755bfda7629dd52b238d1a6311fb",
                        "credit": 14020,
                        "asset": 400000
                    },
                    {
                        "name": "至善园轮毂公司",
                        "_addr": "0x37be6e7c78d4596979ea3b39b716cb85d7927778",
                        "credit": 5,
                        "asset": 210000
```

附录：合约代码

```
pragma solidity ^0.4.24;
pragma experimental ABIEncoderV2;
```

```
contract SupplyChain {
    /** struct defination **/
    struct Message {
        bool success;
        string msg;
    }

    struct Company {
        // 公司名
        string name;
        // 公司地址，哈希值(在后端通过 get_account.sh 生成)
        address _addr;
        // 公司信用等级
        uint256 credit;
        // 公司资产
        uint256 asset;
    }

    struct Bank {
        // 银行名
        string name;
        // 银行地址，哈希值
        address _addr;
    }

    // 一笔应收账款单
    struct Bill {
        // 账单id
        uint256 bill_id;
        // 申请借钱的公司
        address borrow_company;
        // 借出钱的公司
        address lend_company;
        // 应收金额
        uint256 money;
        // 借款日期
        uint256 start_date;
        // 还款日期
        uint256 end_date;
        // 是否通过审核
        bool is_pass;
        // 是否还款
        bool is_finish;
        // 是否是核心企业的账单
        bool is_from_upstream;
        // 收款公司的信用评级变动
        uint256 credit_growth;
    }
```

```
// 一笔向银行的融资
struct Loan {
    // 融资账单id
    uint256 loan_id;
    // 申请融资的公司
    address request_company;
    // 银行地址
    address bank_address;
    // 发起日期
    uint256 start_date;
    // 还款日期
    uint256 end_date;
    // 是否通过审核
    bool is_pass;
    // 是否还款
    bool is_finish;
    // 融资金额
    uint256 money;
}

// 交易
struct Deal {
    // 交易id
    uint256 deal_id;
    // 出钱的一方，买家
    address buyer_company;
    // 出货的一方，卖家
    address seller_company;
    // 交易发起开始日期
    uint256 start_date;
    // 交易是否完成
    bool is_finish;
    // 购买商品描述
    string merchant;
    // 购买金额
    uint256 money;
}

// // 由公司地址到公司实体的映射
// mapping(address => Company) addr2company;

// 由公司地址到公司资产的映射
mapping(address => uint256) assets;

// 由公司地址到公司信用评级的映射
mapping(address => uint256) credit_level;

// 由公司地址到公司级别的映射，表明该公司是否是核心企业
mapping(address => bool) is_upstream;
```

```
/** ----------------- **/

/** global variables defination **/

// 全部银行的集合
Bank[] banks;

// 全部公司的集合
Company[] companies;

// 所有账单的集合
Bill[] bills;
uint256 bill_id;

// 所有融资的合集
Loan[] loans;
uint256 loan_id;

// 所有交易的合集
Deal[] deals;
uint256 deal_id;

/** ----------------- **/

constructor() public {
    bill_id = 1;
    loan_id = 1;
    deal_id = 1;
}

/** methods defination **/
// 根据银行地址获取银行的信息
function getBankByAddress(address bankAddress)
    public
    view
    returns (Bank memory)
{
    Bank memory bank;

    for (uint256 i = 0; i < banks.length; i++) {
        if (bankAddress == banks[i]._addr) {
            Bank storage temp = banks[i];
            bank = temp;
            break;
        }
    }
    return bank;
}
```

```
// 根据公司地址获取公司信息
function getCompanyByAddress(address companyAddress)
    public
    view
    returns (Company memory)
{
    Company memory comp;
    for (uint256 i = 0; i < companies.length; i++) {
        if (companyAddress == companies[i]._addr) {
            Company storage temp = companies[i];
            comp = temp;
            break;
        }
    }
    return comp;
}

// 返回该公司是否是上游企业
function isCompanyUpstream(address companyAddress)
    public
    view
    returns (bool)
{
    return is_upstream[companyAddress];
}

function setCompanyUpstream(address companyAddress)
    public
    view
    returns (bool)
{
    is_upstream[companyAddress] = true;
    return true;
}

// 根据loan_id获取一笔贷款
function getLoanById(uint256 lid) public view returns (Loan memory) {
    Loan memory loan;
    for (uint256 i = 0; i < loans.length; i++) {
        if (lid == loans[i].loan_id) {
            Loan storage l = loans[i];
            loan = l;
            break;
        }
    }

    return loan;
}
```

```
// 根据bill_id获取一笔贷款
function getBillById(uint256 bid) public view returns (Bill memory) {
    Bill memory bill;
    for (uint256 i = 0; i < bills.length; i++) {
        if (bid == bills[i].bill_id) {
            Bill storage b = bills[i];
            bill = b;
            break;
        }
    }
    return bill;
}

// 根据 deal_id 获取一笔交易的内容
function getDealById(uint256 did) public view returns (Deal memory) {
    Deal memory deal;
    for (uint256 i = 0; i < deals.length; i++) {
        if (did == deals[i].deal_id) {
            Deal storage d = deals[i];
            deal = d;
            break;
        }
    }
    return deal;
}

// 注册一家银行
function registerBank(address bankAddress, string name)
    public
    returns (bool success)
{
    banks.push(Bank(name, bankAddress));
    return true;
}

// 注册一家公司，默认信用等级为0，资产为0，为非核心企业。需要后续的额外添加
function registerCompany(address companyAddress, string name)
    public
    returns (bool success)
{
    companies.push(Company(name, companyAddress, 0, 0));
    is_upstream[companyAddress] = false;
    return true;
}

// 注册公司资产
function registerCompanyAsset(address companyAddress, uint256 money)
    public
```

```
        returns (bool success)
    {
        assets[companyAddress] = money;
        for (uint256 i = 0; i < companies.length; i++) {
            if (companies[i]._addr == companyAddress) {
                companies[i].asset = money;
            }
        }
        return true;
    }

    // 注册公司的信用等级
    function registerCompanyCreditLevel(address companyAddress, uint256 cl)
        public
        returns (bool success)
    {
        credit_level[companyAddress] = cl;
        for (uint256 i = 0; i < companies.length; i++) {
            if (companies[i]._addr == companyAddress) {
                companies[i].credit = cl;
            }
        }
        return true;
    }

    // 增加某公司的信用等级
    function addCreditLevel(address companyAddress, uint256 cl)
        public
        returns (bool success)
    {
        credit_level[companyAddress] += cl;
        for (uint256 i = 0; i < companies.length; i++) {
            if (companyAddress == companies[i]._addr) {
                companies[i].credit += cl;

                break;
            }
        }
        return true;
    }

    // 增加某公司的资产数量
    function addAsset(address companyAddress, uint256 ass)
        public
        returns (bool success)
    {
        assets[companyAddress] += ass;
        for (uint256 i = 0; i < companies.length; i++) {
            if (companyAddress == companies[i]._addr) {
```

```
            companies[i].asset += ass;

            break;
        }
    }
    return true;
}

// 减少某公司的资产数量
function decreaseAsset(address companyAddress, uint256 ass)
    public
    returns (bool success)
{
    assets[companyAddress] -= ass;
    for (uint256 i = 0; i < companies.length; i++) {
        if (companyAddress == companies[i]._addr) {
            companies[i].asset -= ass;

            break;
        }
    }
    return true;
}

// 获取所有银行
function getAllBanks() public view returns (Bank[] memory) {
    Bank[] memory ret = new Bank[](banks.length);
    for (uint256 i = 0; i < banks.length; i++) {
        Bank storage temp = banks[i];
        ret[i] = temp;
    }
    return ret;
}

// 获取所有公司
function getAllCompanies() public view returns (Company[] memory) {
    Company[] memory ret = new Company[](companies.length);
    for (uint256 i = 0; i < companies.length; i++) {
        Company storage temp = companies[i];
        ret[i] = temp;
    }
    return ret;
}

// 获取所有的融资
function getAllLoans() public view returns (Loan[] memory) {
    Loan[] memory ret = new Loan[](loans.length);
    for (uint256 i = 0; i < loans.length; i++) {
        Loan storage temp = loans[i];
```

```
            ret[i] = temp;
        }
        return ret;
    }

    // 获取全部账单
    function getAllBills() public view returns (Bill[] memory) {
        Bill[] memory ret = new Bill[](bills.length);
        for (uint256 i = 0; i < bills.length; i++) {
            Bill storage temp = bills[i];
            ret[i] = temp;
        }
        return ret;
    }

    // 根据地址获取某个公司所有的作为lend_company的账单
    function getLendBillsByCompanyAddress(address companyAddress)
        public
        view
        returns (Bill[] memory)
    {
        uint256 len = bills.length;
        Bill[] memory ret = new Bill[](len);
        uint256 k = 0;
        for (uint256 i = 0; i < len; i++) {
            if (companyAddress == bills[i].lend_company) {
                ret[k++] = bills[i];
            }
        }
        return ret;
    }

    // 根据地址获取某个公司所有的作为borrow_company的账单
    function getBorrowBillsByComanyAddress(address companyAddress)
        public
        view
        returns (Bill[] memory)
    {
        uint256 len = bills.length;
        Bill[] memory ret = new Bill[](len);
        uint256 k = 0;
        for (uint256 i = 0; i < len; i++) {
            if (companyAddress == bills[i].borrow_company) {
                ret[k++] = bills[i];
            }
        }
        return ret;
    }
```

```
// 根据地址获取某个公司的所有融资
function getLoansByCompanyAddress(address companyAddress)
    public
    view
    returns (Loan[] memory)
{
    uint256 len = loans.length;
    Loan[] memory ret = new Loan[](len);
    uint256 k = 0;
    for (uint256 i = 0; i < len; i++) {
        if (companyAddress == loans[i].request_company) {
            ret[k++] = loans[i];
        }
    }
    return ret;
}

// 计算转账后增加的信用等级
function calculateCredit(
    address borrow_company, // 发起采购/借款的公司
    uint256 money, // 金额
    bool is_from_upstream // 是否来自于核心企业
) public view returns (uint256 growth) {
    if (!is_from_upstream) {
        return 0;
    }
    Company memory bc = getCompanyByAddress(borrow_company);
    uint256 prev_credit = bc.credit;
    return prev_credit / 10 + money / 50;
}

// 调整某个账单的金额
function adjustMoneyForBill(
    uint256 bid, // 待调整的账单的id
    uint256 money // 调整后的金额
) public view returns (bool value) {
    for (uint256 i = 0; i < bills.length; i++) {
        if (bills[i].bill_id == bid) {
            if (money > bills[i].money) {
                return false;
            } else {
                bills[i].money = money;
                return true;
            }
        }
    }
    return true;
}
```

```
    // 增加bill
    function addBill(
        address borrow_company,
        address lend_company,
        uint256 money,
        uint256 start_date,
        uint256 end_date,
        bool is_pass,
        bool is_finish,
        bool is_from_upstream,
        uint256 credit_growth
    ) public returns (bool success) {
        bills.push(
            Bill(
                bill_id,
                borrow_company,
                lend_company,
                money,
                start_date,
                end_date,
                is_pass,
                is_finish,
                is_from_upstream,
                credit_growth
            )
        );
        bill_id++;
        return true;
    }

    // 增加一条loan
    function addLoan(
        address request_company,
        address bank_address,
        uint256 start_date,
        uint256 end_date,
        bool is_pass,
        bool is_finish,
        uint256 money
    ) public returns (bool success) {
        loans.push(
            Loan(
                loan_id,
                request_company,
                bank_address,
                start_date,
                end_date,
                is_pass,
                is_finish,
                money
            )
        );
        loan_id++;
        return true;
    }

    // 增加一笔交易
    function addDeal(
        address buyer_company,
        address seller_company,
        uint256 start_date,
        bool is_finish,
        string merchant,
        uint256 money
    ) public returns (bool success) {
        deals.push(
            Deal(
                deal_id,
                buyer_company,
                seller_company,
                start_date,
                is_finish,
                merchant,
                money
            )
        );
        deal_id++;
        return true;
    }

    // 转让一部分应收账单
    function splitBillToAnotherCompany(
        address from, // 从哪家公司转让
        address to, // 转让到哪家公司
        uint256 bid, // 被拆分的账单
        uint256 money, // 转让金额
        uint256 timestamp // 当前时间
    ) public returns (Message memory) {
        Bill memory bill_to_be_split;
        bool flag = false;
        for (uint256 i = 0; i < bills.length; i++) {
            if (bid == bills[i].bill_id) {
                flag = true;
                bill_to_be_split = bills[i];
                break;
            }
        }
        if (!flag) {
            return Message(false, "该账单不存在");
        }

        if (bill_to_be_split.lend_company != from) {
            return Message(false, "该账单并不属于该公司");
        }

        if (bill_to_be_split.borrow_company == to) {
            return Message(false, "该账单不可被转让，因为接收者需要还款");
        }

        if (money > bill_to_be_split.money) {
            return Message(false, "转让金额大于了账单原有金额");
        }

        if (timestamp > bill_to_be_split.end_date) {
            return Message(false, "该账单已经过期");
        }

        if (!bill_to_be_split.is_pass) {
            return Message(false, "该账单还没有通过审核");
        }

        if (bill_to_be_split.is_finish) {
            return Message(false, "该账单已完成还款，不可再被拆分");
        }
        // 鉴权完成，开始转让
        // 调整原有的账单金额
        adjustMoneyForBill(bid, bill_to_be_split.money - money);
        // 新增一份订单
        addBill(
            bill_to_be_split.borrow_company,
            to,
            money,
            timestamp,
            bill_to_be_split.end_date,
            false,
            false,
            bill_to_be_split.is_from_upstream,
            0
        );

        return Message(true, "转让完成，等待审核中");
    }

    // 通过某一笔账单的审核
    function passBill(uint256 bid) public returns (Message memory) {
        for (uint256 i = 0; i < bills.length; i++) {
            if (bid == bills[i].bill_id) {
```

```
            if (bills[i].is_pass) {
                return Message(true, "该账单已经通过审核");
            } else {
                bills[i].is_pass = true;
                addCreditLevel(
                    bills[i].lend_company,
                    bills[i].credit_growth
                );
                return Message(true, "该账单成功通过审核");
            }
        }
    }
    return Message(false, "该账单不存在");
}

// 提交一笔应收款账单
function submitBill(
    address borrow_company, // 请求借钱/采购的公司
    address lend_company, // 同意采购/批准借款的公司
    uint256 money, // 金额
    uint256 start_date, // 发起时间
    uint256 end_date, // 应还款时间
    bool is_from_upstream
) public returns (Message memory) {
    if (start_date > end_date) {
        return Message(false, "时间不合法");
    }
    uint256 credit_change = calculateCredit(
        borrow_company,
        money,
        is_from_upstream
    );
    addBill(
        borrow_company,
        lend_company,
        money,
        start_date,
        end_date,
        false,
        false,
        is_from_upstream,
        credit_change
    );
}

// 判断是否银行是否能为该公司提供贷款
function canLoanToThisCompany(
    address company,
    address bank_address,
        uint256 current_credit,
        uint256 money
    ) public returns (bool can) {
        Company memory c = getCompanyByAddress(company);

        // 若贷款公司的资产大于其要借的钱数，则直接同意
        if (c.asset >= money) {
            return true;
        }

        // 若贷款公司的资产与其信用值乘50大于要借的钱数，也直接同意
        if (c.asset + current_credit * 50 >= money) {
            return true;
        }
        return false;
    }

    // 提交一笔银行融资
    function submitLoan(
        address request_company,
        address bank_address,
        uint256 start_date,
        uint256 end_date,
        uint256 money,
        uint256 current_credit
    ) public returns (Message memory) {
        if (start_date >= end_date) {
            return Message(false, "日期不合法");
        }
        if (
            canLoanToThisCompany(
                request_company,
                bank_address,
                current_credit,
                money
            )
        ) {
            addLoan(
                request_company,
                bank_address,
                start_date,
                end_date,
                false,
                false,
                money
            );
            return Message(true, "提交贷款申请成功");
        } else {
            return Message(false, "提交贷款申请失败");
        }
    }

    // 通过贷款审核
    function passLoan(uint256 lid) public returns (Message memory) {
        for (uint256 i = 0; i < loans.length; i++) {
            if (lid == loans[i].loan_id) {
                if (loans[i].is_pass) {
                    return Message(true, "该贷款已经通过审核");
                } else {
                    loans[i].is_pass = true;
                    addAsset(loans[i].request_company, loans[i].money);
                    return Message(true, "该贷款成功通过审核");
                }
            }
        }
        return Message(false, "该贷款不存在");
    }

    // 判断买家是否可以通过账单来交易
    function canBuyerDealByBill(address buyer_company, uint256 money)
        public
        view
        returns (bool can)
    {
        Bill[] memory buyer_bills =
getLendBillsByCompanyAddress(buyer_company);
        for (uint256 i = 0; i < buyer_bills.length; i++) {
            if (
                buyer_bills[i].is_from_upstream && buyer_bills[i].money > money
            ) {
                return true;
            }
        }
        return false;
    }

    // 判断买家是否能够提交交易
    function canBuyerSubmitDeal(address buyer_company, uint256 money)
        public
        view
        returns (bool can)
    {
        // 如果买家是核心企业，则可以交易
        if (is_upstream[buyer_company]) {
            return true;
        }

        // 如果买家的资金充足，也可以交易
        if (assets[buyer_company] >= money) {
            return true;
        }

        // 如果买家持有面值较大的账单，也可以交易
        if (canBuyerDealByBill(buyer_company, money)) {
            return true;
        }

        return false;
    }

    // 提交一笔交易
    function submitDeal(
        address buyer_company,
        address seller_company,
        uint256 start_date,
        string merchant,
        uint256 money
    ) public returns (Message memory) {
        if (buyer_company == seller_company) {
            return Message(false, "买家和卖家不能同为一家公司");
        }

        addDeal(
            buyer_company,
            seller_company,
            start_date,
            false,
            merchant,
            money
        );
        return Message(true, "提出交易申请成功");
    }

    // 通过一笔交易
    function passDeal(uint256 did) public returns (Message memory) {
        for (uint256 i = 0; i < deals.length; i++) {
            if (did == deals[i].deal_id) {
                if (deals[i].is_finish) {
                    return Message(true, "该交易已经被通过");
                } else {
                    deals[i].is_finish = true;
                    addAsset(deals[i].seller_company, deals[i].money);
                    decreaseAsset(deals[i].buyer_company, deals[i].money);
                    return Message(true, "该交易通过成功");
                }
            }
        }
        return Message(false, "交易不存在");
    }

    /** --------------- **/
}
```