

第四章:建立和测试网络

学习 Bluemix 和区块链

Bob Dill, IBM 杰出工程师, 全球售前技术支持CTO

David Smits, 资深认证架构师, IBM 区块链



计划: 30 分钟的章节内容以及1到2小时的实践

第一章	什么是区块链? 概念和架构全览
第二章	我们要构建的故事是什么?
第二章第一节	故事的架构
第三章	建立本地Hyperledger Fabric V1 开发环境
第四章	建立和测试网络
第五章	管理员用户体验
第六章	买家支持和用户体验
第七章	卖家支持和用户体验
第八章	提供商支持和用户体验
第九章	发货人支持和用户体验
第十章	金融公司支持和用户体验
第十一章	综合演示
第十二章	事件和自动化演示

谁是参与者和他们能做什么？

定义网络会员，交易和谁应该批准交易

▪ 买家

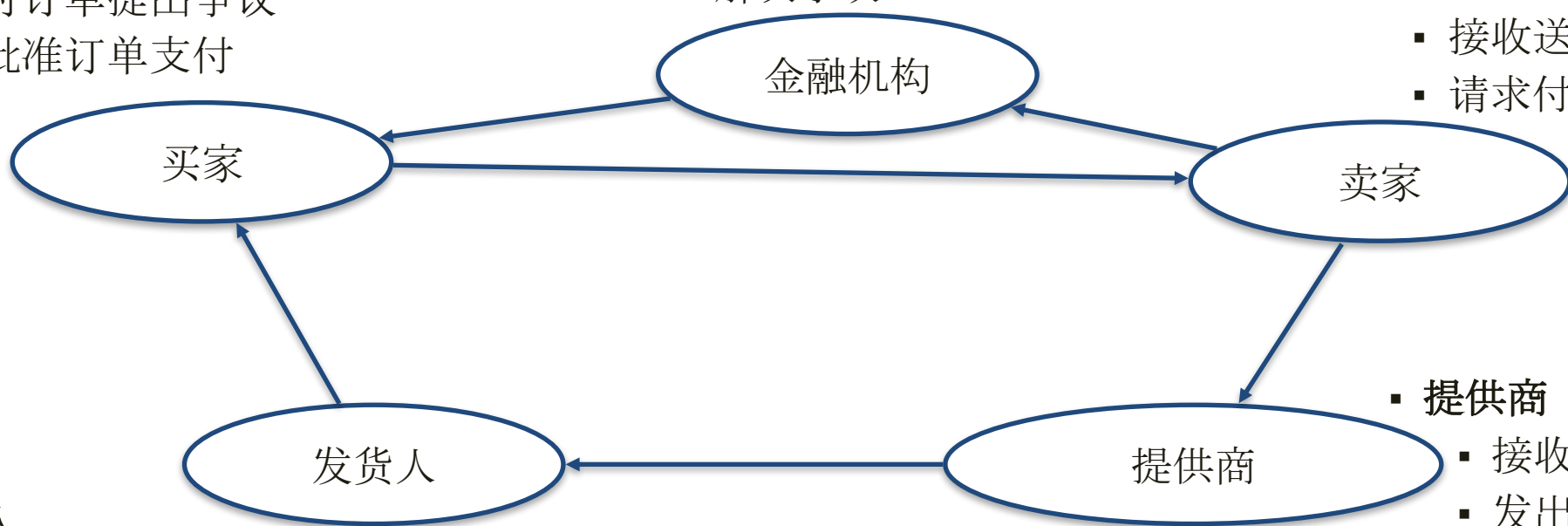
- 创建订单
- 提交订单
- 接收发货
- 对订单提出争议
- 批准订单支付

▪ 金融公司

- 接收支付请求
- 批准和支付
- 接收争议通知
- 解决争议

▪ 卖家

- 接受订单
- 提交订单给提供商
- 接收送货通知
- 请求付款



▪ 发货人

- 接收送货请求
- 发出送货通知

▪ 提供商

- 接收订单项目的请求
- 发出，解决延迟交货
- 请求送货
- 接收送货通知

定义会员

- 在我们简单的网络, 会员具有:
 - 一个公司名字
 - 一个识别码, 我们用电子邮箱地址表示
- 我们利用一个字段定义会员的抽象类型

```
18 namespace composer.base
19
20 abstract participant Member {
21     o String companyName
22 }
```

- 然后对每个类型的会员, 我们扩展抽象类型的定义.

```
participant Buyer identified by buyerID extends Member{
    o String buyerID
}
```

- 这样做的目的是介绍抽象类型并且让我们能够把不同类型的定义分成不同的文件, 这是为了在我们完成后, 易于维护和把所有东西组合在一起。

定义资产

- 在我们这个教程里定义的单一资产就是一个“订单”，如右边所示
- 很多字段用来存放日期，让我们知道什么时间发生什么。日期和理由通过交易更新，这允许参与者进行。
- 大括号，它说明数组
- 箭头符号 (- ->), 它说明参考其它网络类, 之前的定义.

```
asset Order identified by orderNumber {  
  o String orderNumber  
  o String[] items  
  o String status  
  o Integer amount  
  o String created  
  o String bought  
  o String ordered  
  o String dateBackordered  
  o String requestShipment  
  o String delivered  
  o String disputeOpened  
  o String disputeResolved  
  o String paymentRequested  
  o String orderRefunded  
  o String paid  
  o String[] vendors  
  o String dispute  
  o String resolve  
  o String backorder  
  o String refund  
  --> Buyer buyer  
  --> Seller seller
```

定义交易

- 交易采用定义资产和会员类似的模型语言.
- 这里我们命名一个交易和指定要处理这个交易时必须同时具备什么内容
- 这是一个交易类
- 它的名字是CreateOrder
- 它有一个字段(整数 amount)
- 它指向3个其它的实例
 - 订单
 - 买家
 - 卖家

```
transaction CreateOrder {  
  o Integer amount  
  --> Order order  
  --> Buyer buyer  
  --> Seller seller  
}
```

利用前面的信息， 定义下面内容

- 会员:
 - Buyer, Seller, Provider, Shipper, FinanceCo
- 资产:
 - Order
- 交易:
 - CreateOrder, Buy, OrderFromSupplier, RequestShipping, Deliver, BackOrder, Dispute, Resolve, Request Payment, Pay, Refund

让我们来检查网络

- 第1步, 更新模型文件
 - 第2步, 创建、归档和部署它
 - 第3步, 装入composer, 测试它
-
- 第1步, 答案在Documents/answers 文件夹
 - 第2步, 从Chapter04文件夹中执行下面命令
 - **buildAndDeploy**
 - 第3步, 去到:
 - 从下面导入模型文件
Chapter04/network/dist/zerotoblockchain-network.bna
 - 测试模型
 - 你会注意到Order对象没太多东西发生, 那是下一步

编写代码来实现交易

- 每个交易需要实现逻辑。例如，CreateOrder交易是用来允许买家建立订单，并且在发送给卖家前存起来。代码显示在右边。
- 你可以看见类定义 (右下角) 包括一个链接到Order, 订单的Amount, 和Buyer. 在这个交易代码中，卖家信息没有用到 – 因为订单还没有发送到卖家。

```
/**
 * create an order to purchase
 * @param {org.acme.Z2BTestNetwork.CreateOrder} purchase – the order to be processed
 * @transaction
 */
function CreateOrder(purchase) {
    purchase.order.buyer = purchase.buyer;
    purchase.order.amount = purchase.amount;
    purchase.order.created = new Date().toISOString();
    purchase.order.status = "Order Created";
    return getAssetRegistry('org.acme.Z2BTestNetwork.Order')
        .then(function (assetRegistry) {
            return assetRegistry.update(purchase.order);
        });
}
```

```
transaction CreateOrder {
    o Integer amount
    --> Order order
    --> Buyer buyer
    --> Seller seller
}
```

编写代码来测试交易

- 我们用mocha 服务来测试这个应用, 代码如下:
- 我们呆会儿将一起看看这一代码
- 当我们完成的时候, 我们可以告诉npm去测试我们创建的代码, 结果应该如下所示:

```
Finance Network
#createOrder
  ✓ should be able to create an order (82ms)
#issueBuyRequest
  ✓ should be able to issue a buy request (40ms)
#issueOrderFromSupplier
  ✓ should be able to issue a supplier order (50ms)
#issueRequestShipment
  ✓ should be able to issue a request to ship product (47ms)
#issueDelivery
  ✓ should be able to record a product delivery (39ms)
#issueRequestPayment
  ✓ should be able to issue a request to request payment for a product (58ms)
#issuePayment
  ✓ should be able to record a product payment (48ms)
#issueDispute
  ✓ should be able to record a product dispute (63ms)
#issueResolution
  ✓ should be able to record a dispute resolution (48ms)
#issueBackorder
  ✓ should be able to record a product backorder (53ms)

10 passing (1s)
```

```
describe('#createOrder', () => {

  it('should be able to create an order', () => {
    const factory = businessNetworkConnection.getBusinessNetwork().getFactory();

    // create the buyer
    const buyer = factory.newResource(NS, 'Buyer', buyerID);
    buyer.companyName = 'billybob computing';

    // create the seller
    const seller = factory.newResource(NS, 'Seller', sellerID);
    seller.companyName = 'Simon PC Hardware, Inc';

    // create the order
    let order = factory.newResource(NS, 'Order', orderNo);
    order = createOrderTemplate(order);
    order = addItem(order);
    order.orderNumber = orderNo;

    // create the buy transaction
    const createNew = factory.newTransaction(NS, 'CreateOrder');

    order.buyer = factory.newRelationship(NS, 'Buyer', buyer.$identifier);
    order.seller = factory.newRelationship(NS, 'Seller', seller.$identifier);
    createNew.order = factory.newRelationship(NS, 'Order', order.$identifier);
    createNew.buyer = factory.newRelationship(NS, 'Buyer', buyer.$identifier);
    createNew.seller = factory.newRelationship(NS, 'Seller', seller.$identifier);
    createNew.amount = order.amount;
    // the buyer should of the commodity should be buyer
    //order.buyer.$identifier.should.equal(buyer.$identifier);
    order.status.should.equal('Order Created');
    order.amount.should.equal(orderAmount);
    createNew.amount.should.equal(orderAmount);
    createNew.order.$identifier.should.equal(orderNo);
```

调用composer-rest-server

- 从Chapter04开始
 - 执行下面命令:
 - **buildAndDeploy**
 - 它将把你的整个网络装入docker
 - 执行下面命令
 - **./start_rest_server.sh**
 - 它会显示如下:

```
[?] Enter your Fabric Connection Profile Name: hlfv1
[?] Enter your Business Network Identifier : zerotoblockchain-network
[?] Enter your Fabric username : admin
[?] Enter your secret: adminpw
[?] Specify if you want namespaces in the generated REST API: always use namespaces
[?] Specify if you want the generated REST API to be secured: No
[?] Specify if you want to enable event publication over WebSockets: Yes
[?] Specify if you want to enable TLS security for the REST API: No
```

- 进入 **localhost:3000/explorer** , 检查和测试你新的RESTful APIs

计划: 30 分钟的章节内容以及1到2小时的实践

第一章	什么是区块链? 概念和架构全览
第二章	我们要构建的故事是什么?
第二章第一节	故事的架构
第三章	建立本地Hyperledger Fabric V1 开发环境
第四章	建立和测试网络
第五章	管理员用户体验
第六章	买家支持和用户体验
第七章	卖家支持和用户体验
第八章	提供商支持和用户体验
第九章	发货人支持和用户体验
第十章	金融公司支持和用户体验
第十一章	综合演示
第十二章	事件和自动化演示