使用Golang设计数据和交易模型

在Hyperledger Fabric中，链码是由开发人员编写的智能合约的一种形式。链码实现了由区块链网络的利益相关者商定的业务逻辑。该功能暴露给客户端应用程序供其调用，只要它们具有正确的权限。

Chaincode在其自己的容器中作为独立进程运行，与Fabric网络的其他组件隔离。一个背书节点(endorsing peer)管理链码和事务调用的生命周期。通过响应客户调用，链码查询和更新账本并生成交易提议。

在本章中，我们将学习如何使用Go语言开发链码，并实施该场景下的智能合约业务逻辑。最后，我们将探讨开发全功能链码所需的关键概念和库。

在接下来的部分中，我们将探讨与概念相关的代码片段，您可以在以下地址完整实现链式代码：<https://github.com/HyperledgerHandsOn/trade-finance-logistics/tree/master/chaincode/src/github.com/trade_workflow_v1>

提示: 请注意，这也可以在我们上一章创建的本地git克隆中获得。我们有两个版本的链码，一个在trade\_workflow文件夹中，另一个在trade\_workflow\_v1文件夹中。 我们需要两个版本来演示第9章“区块链网络中的生活”中的升级。 在本章中，我们使用v1版本来演示如何在Go中编写链接代码。

在本章中，我们将涵盖以下内容：

* 创建chaincode
* 访问控制
* 实施chaincode功能
* 测试chaincode
* Chaincode设计主题
* 输出记录

开始链码开发

在我们开始编写链码之前，我们需要首先启动我们的开发环境。

在第3章“用业务场景设置平台”中描述了建立开发环境的步骤。但是，我们现在继续以开发模式启动Fabric网络。这种模式允许我们控制如何构建和运行链码。我们将使用这个网络在开发环境中运行我们的

下面是我们如何用开发模式开启Fabric网络：

$ cd $GOPATH/src/trade-finance-logistics/network

$ ./trade.sh up -d true

提示：如果在网络启动时遇到任何错误，可能是由一些遗留下来的Docker容器引起的。

您可以通过使用./trade.sh down -d true停止网络并运行以下命令来解决此问题：./trade.sh clean -d true。

该-d true选项告诉我们的脚本采取行动在dev网络上。

我们的开发网络现在是四个Docker容器运行。该网络由单个order，在devmode中运行的单个peer，链码容器和CLI容器组成。CLI容器在启动时创建一个名为tradechannel的区块链通道。我们将使用CLI与链码进行交互。

我们可以随意在日志目录中检查日志消息。它列出了网络启动期间执行的组件和功能。我们将保持终端打开，因为一旦chaincode被安装并调用，我们将在这里收到更多的日志消息。

编译和运行链码

克隆的源代码已经包括使用Go vendoring的所有依赖。考虑到这一点，我们现在可以开始构建代码并通过以下步骤运行链式代码：

1. 编译链码：在一个新的终端中，连接到链码容器并使用以下命令构建链码：

$ docker exec –it chaincode bash

$ cd trade\_workflow\_v1

$ go build

2.运行chaincode时执行以下命令：

$ CORE\_PEER\_ADDRESS=peer:7052 CORE\_CHAINCODE\_ID\_NAME=tw:0 ./trade\_workflow\_v1

我们现在有一个连接到peer的正在运行的链码。这里的日志消息表明链代码已启动并正在运行。您还可以检查网络终端中的日志消息，该消息列出与peer上到该链码的所有连接。

安装和实例链码

我们现在需要在启动链码之前在通道上安装它，它将调用方法Init：

1. 安装链码：在一个新的终端中，连接到CLI容器并按照以下名称tw安装链码：

$ docker exec -it cli bash

$ peer chaincode install -p chaincodedev/chaincode/trade\_workflow\_v1 -n tw -v 0

2.现在，实例以下链码：

$ peer chaincode instantiate -n tw -v 0 -c '{"Args": ["init","LumberInc","LumberBank","100000","WoodenToys","ToyBank","200000","UniversalFreight"," -C tradechannel

CLI连接的终端现在包含与链代码交互的日志消息列表。链码终端显示来自链码方法调用的消息，网络终端显示来自peer和order之间通信的消息。

调用链码

现在我们有一个运行的链码，我们就可以开始调用一些功能。我们的链码有几种创建和检索资产的方法。现在，我们只会调用其中的两个; 第一个创建一个新的贸易协议，第二个协议从账本中检索到它。要做到这一点，请完成以下步骤：

1. 使用以下命令将新的贸易协定放到账本中：

$ peer chaincode invoke -n tw -c '{"Args":["requestTrade", "50000", "Wood for Toys"]}' -C tradechannel

2. 使用以下命令检索账本中的该贸易协定：

$ peer chaincode invoke -n tw -c '{"Args":["getTradeStatus", "50000"]}' -C tradechannel

我们现在在devmode上有一个运行网络，我们已经成功测试了我们的链码。在下一节中，我们将学习如何从头开始创建和测试链码。

提示：dev mode

在生产环境中，chaincode的寿命是由peer管理。当我们需要在开发环境中反复修改和测试链码时，我们可以使用devmode，它允许开发人员控制链码的生命周期。此外，devmode将stdout和stderr标准文件引导到终端; 这些在生产环境中是被禁用的。

要使用devmode，peer必须连接到其他网络组件（如生产环境中），并以参数peer-chaincodedev = true开始。链码然后单独启动并配置为连接到peer节点。在开发过程中，链码可以根据需要从终端反复编译，启动，调用和停止。

我们将在下面的章节中使用DEVMODE启用网络。

创建链码

我们现在准备开始实施我们的链码，我们将使用Go语言进行编程。有几个IDE可用于为Go提供支持。一些更好的IDE包括Atom，Visual Studio Code等等。无论你选择任何环境都可以用我们的例子。（到该章第7页）