**Chain-code Development**

Contents

[Introduction 2](#_Toc46657410)

[Create Property.go file 2](#_Toc46657411)

[Add Import Section 2](#_Toc46657412)

[Create Property Transfer contract 2](#_Toc46657413)

[Define Property Object 2](#_Toc46657414)

[Define Add Property Function 3](#_Toc46657415)

[Define QueryAll Properties Function 3](#_Toc46657416)

[Define Query Property by Id Function 4](#_Toc46657417)

[Define Transfer Property Function 5](#_Toc46657418)

[Define Main Function 5](#_Toc46657419)

[Complete File 5](#_Toc46657420)

# Introduction

This guide helps you to develop the property transfer application that you will deploy and run on Hyperledger Fabric network in Go Language.

## Create Property.go file

Create the file with the name Property.go

## Add Import Section

Open the newly created file and add below content where you add the contract API package to your chaincode.

*package main*

*import (*

*"encoding/json"*

*"fmt"*

*"github.com/hyperledger/fabric-contract-api-go/contractapi"*

*)*

## Create Property Transfer contract

Append below content in your file where you define the property transfer contract.

*// PropertyTransfer smart contract to show the property transfer transactions*

*type PropertyTransferSmartContract struct {*

*contractapi.Contract*

*}*

## Define Property Object

Next step is to define the Property object. Append below content in your file.

*// Property Obejct*

type Property struct {

ID string `json:"id"`

Name string `json:"name"`

Area int `json:"area"`

OwnerName string `json:"ownerName"`

Value int `json:"value"`

}

## Define Add Property Function

Next step is to define the Add Property function which helps to add new property to the ledger.

// This function helps to Add new property

func (pc \*PropertyTransferSmartContract) AddProperty(ctx contractapi.TransactionContextInterface, id string, name string, area int, ownerName string, value int) error {

propertyJSON, err := ctx.GetStub().GetState(id)

if err != nil {

return fmt.Errorf("Failed to read the data from world state", err)

}

if propertyJSON != nil {

return fmt.Errorf("the property %s already exists", id)

}

prop := Property{

ID: id,

Name: name,

Area: area,

OwnerName: ownerName,

Value: value,

}

propertyBytes, err := json.Marshal(prop)

if err != nil {

return err

}

return ctx.GetStub().PutState(id, propertyBytes)

}

## Define QueryAll Properties Function

Next step is to define the QueryAllProperties function which helps to query all the existing properties.

// This function returns all the existing properties

func (pc \*PropertyTransferSmartContract) QueryAllProperties(ctx contractapi.TransactionContextInterface) ([]\*Property, error) {

propertyIterator, err := ctx.GetStub().GetStateByRange("", "")

if err != nil {

return nil, err

}

defer propertyIterator.Close()

var properties []\*Property

for propertyIterator.HasNext() {

propertyResponse, err := propertyIterator.Next()

if err != nil {

return nil, err

}

var property \*Property

err = json.Unmarshal(propertyResponse.Value, &property)

if err != nil {

return nil, err

}

properties = append(properties, property)

}

return properties, nil

}

## Define Query Property by Id Function

Next step is to define the QueryPropertyById function which helps to query the property by id.

// This function helps to query the property by Id

func (pc \*PropertyTransferSmartContract) QueryPropertyById(ctx contractapi.TransactionContextInterface, id string) (\*Property, error) {

propertyJSON, err := ctx.GetStub().GetState(id)

if err != nil {

return nil, fmt.Errorf("Failed to read the data from world state", err)

}

if propertyJSON == nil {

return nil, fmt.Errorf("the property %s does not exist", id)

}

var property \*Property

err = json.Unmarshal(propertyJSON, &property)

if err != nil {

return nil, err

}

return property, nil

}

## Define Transfer Property Function

Next step is to define the TransferPropertyfunction which helps to transfer the ownership of the property to new owner.

// This functions helps to transfer the ownership of the property

func (pc \*PropertyTransferSmartContract) TransferProperty(ctx contractapi.TransactionContextInterface, id string, newOwner string) error {

property, err := pc.QueryPropertyById(ctx, id)

if err != nil {

return err

}

property.OwnerName = newOwner

propertyJSON, err := json.Marshal(property)

if err != nil {

return err

}

return ctx.GetStub().PutState(id, propertyJSON)

}

## Define Main Function

Next step is to define the main function.

func main() {

propTransferSmartContract := new(PropertyTransferSmartContract)

cc, err := contractapi.NewChaincode(propTransferSmartContract)

if err != nil {

panic(err.Error())

}

if err := cc.Start(); err != nil {

panic(err.Error())

}

}

## Complete File

