package main

import (

"encoding/json"

"fmt"

"log"

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

)

// AssetTransfer provides functions for managing assets

type AssetTransfer struct {

contractapi.Contract

}

// Asset describes basic details of what makes up a simple asset

type Asset struct {

ID string `json:"ID"`

Owner string `json:"Owner"`

Color string `json:"Color"`

Size int `json:"Size"`

AppraisedValue int `json:"AppraisedValue"`

}

// CreateAsset issues a new asset to the world state with given details.

func (t \*AssetTransfer) CreateAsset(ctx contractapi.TransactionContextInterface, id string, owner string, color string, size int, appraisedValue int) error {

asset := Asset{

ID: id,

Owner: owner,

Color: color,

Size: size,

AppraisedValue: appraisedValue,

}

assetJSON, err := json.Marshal(asset)

if err != nil {

return err

}

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

}

// ReadAsset returns the asset stored in the world state with given id.

func (t \*AssetTransfer) ReadAsset(ctx contractapi.TransactionContextInterface, id string) (\*Asset, error) {

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

if err != nil {

return nil, fmt.Errorf("failed to read from world state: %v", err)

}

if assetJSON == nil {

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

}

var asset Asset

err = json.Unmarshal(assetJSON, &asset)

if err != nil {

return nil, err

}

return &asset, nil

}

// UpdateAsset updates an existing asset in the world state with provided parameters.

func (t \*AssetTransfer) UpdateAsset(ctx contractapi.TransactionContextInterface, id string, owner string, color string, size int, appraisedValue int) error {

asset, err := t.ReadAsset(ctx, id)

if err != nil {

return err

}

asset.Owner = owner

asset.Color = color

asset.Size = size

asset.AppraisedValue = appraisedValue

assetJSON, err := json.Marshal(asset)

if err != nil {

return err

}

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

}

// DeleteAsset deletes an given asset from the world state.

func (t \*AssetTransfer) DeleteAsset(ctx contractapi.TransactionContextInterface, id string) error {

return ctx.GetStub().DelState(id)

}

// GetAllAssets returns all assets found in world state

func (t \*AssetTransfer) GetAllAssets(ctx contractapi.TransactionContextInterface) ([]\*Asset, error) {

queryString := `{"selector": {}}`

resultsIterator, err := ctx.GetStub().GetQueryResult(queryString)

if err != nil {

return nil, err

}

defer resultsIterator.Close()

var assets []\*Asset

for resultsIterator.HasNext() {

queryResponse, err := resultsIterator.Next()

if err != nil {

return nil, err

}

var asset Asset

err = json.Unmarshal(queryResponse.Value, &asset)

if err != nil {

return nil, err

}

assets = append(assets, &asset)

}

return assets, nil

}

func main() {

chaincode, err := contractapi.NewChaincode(new(AssetTransfer))

if err != nil {

log.Panicf("Error creating asset-transfer chaincode: %v", err)

}

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

log.Panicf("Error starting asset-transfer chaincode: %v", err)

}

}