**Các khái niệm và thành phần trong Cosmos SDK**

**1. Các cài đặt yêu cầu**

Yêu cầu cài golang version 1.13 trở lên

**2. Cấu trúc thư mục và các khái niệm**

Cấu trúc thư mục của một project được dụng trên Cosmos SDK

./project\_name

├── Makefile

├── Makefile.ledger

├── app.go

├── cmd

│ ├── nscli

│ │ └── main.go

│ └── nsd

│ └── main.go

├── go.mod

├── go.sum

└── x

└── nameservice

├── alias.go

├── client

│ ├── cli

│ │ ├── query.go

│ │ └── tx.go

│ └── rest

│ ├── query.go

│ ├── rest.go

│ └── tx.go

├── genesis.go

├── handler.go

├── keeper

│ ├── keeper.go

│ └── querier.go

├── types

│ ├── codec.go

│ ├── errors.go

│ ├── expected\_keepers.go

│ ├── key.go

│ ├── msgs.go

│ ├── querier.go

│ └── types.go

└── module.go

Cosmos SDK được thiết kế để giúp bạn xây dựng các **state machine**. SDK là một **modular framework**, có nghĩa là các ứng dụng được xây dựng bằng lắp ghép các mdule lại để chúng tương tác với nhau. Mỗi module sẽcchứa bộ xử lý meessage / transaction riêng của nó, và SDK chịu trách nhiệm định tuyến từng message đến module tương ứng với message đó.

**Một số module có sẵn trong Cosmos SDK:**

* auth : Module này quản lý các account, phí giao dịch và cung cấp quyền truy cập vào các chức năng này cho phần các thành phần còn lại của ứng dụng.
* bank : Module này cho phép ứng dụng tạo và quản lý một loại token.
* distribution : Module này cho phép tạo ra một **validator** cho ứng dụng.
* slashing :
* supply
* product: Module này chưa tồn tại, chúng ta sẽ code module này

**State**

Ở mỗi thời điểm ứng dụng sẽ luôn có 1 state xác đinh. State cho biết mỗi account đang sở hữu sở hữu bao nhiêu token và các asset khác.

**Message**

Message được chứa trong các transaction. Message sẽ trigger các hành động cập nhật state. Mỗi module sẽ xác định riêng sách các message mà chúng sẽ bắt và cách xử lý.

**3.Các thành phần chính**

**3.1 Types**

Types giúp định nghĩa data struct của các object từ thực tế vào trong ứng dụng của chúng ta, ví dụ như sản phẩm, xe, nhà, con người.

type Product struct {

ProductID string `json:"productID"`

Description string `json:"description"`

Owner sdk.AccAddress `json:"owner"`

}

**3.2 Msg**

Msg định nghĩa các msg được tạo ra và broadcast bởi người dùng

type MsgSetProduct struct {

ProductID string `json:"productID"`

Description string `json:"description"`

Signer sdk.AccAddress `json:"owner"`

}

**3.2 Keepers**

Lõi chính của Cosmos SDK là một thành phần được gọi là **Keeper**. Nó là thành phần trực tiếp cập nhật state trong store, và nó được gọi bởi một thành phần khác được gọi là **Handler**.

type Keeper struct {

CoinKeeper types.BankKeeper

storeKey sdk.StoreKey // Unexposed key to access store from sdk.Context

cdc \*codec.Codec // The wire codec for binary encoding/decoding.

}

Trong project mình sẽ làm ở bài sau, Keeper của mình chỉ cần tích hợp 3 module ở trên là đủ.

* Module types.BankKeeper giúp quản lý việc tạo và gửi token giữa các accounts.
* Module sdk.StoreKey giúp keeper tác động trực tiếp vào store để cập nhật state
* Module codec.Codec phục vụ encode và decode data.

**Getters và Setters**

Store của Cosmos SDK sẽ lưu trữ data dưới dạng key-value . Các phương thức Get, Set là chức năng của module sdk.StoreKey được gọi thông qua Keeper.

func (k Keeper) SetProduct(ctx sdk.Context, productKey string, product types.Product) {

store := ctx.KVStore(k.storeKey)

store.Set([]byte(productKey), k.cdc.MustMarshalBinaryBare(product))

}

func (k Keeper) GetProduct(ctx sdk.Context, productKey string) types.Product {

store := ctx.KVStore(k.storeKey)

bz := store.Get([]byte(productKey))

var product types.Product

k.cdc.MustUnmarshalBinaryBare(bz, &product)

return product

}

**Delete**

Ngoài ra module sdk.Storekey còn có phương thức delete giúp xóa hoàn toàn data của một key nào đó.

func (k Keeper) DeleteProduct(ctx sdk.Context, productKey string) {

store := ctx.KVStore(k.storeKey)

store.Delete([]byte(productKey))

}

**3.3 Handler**

Handler xác định hành động mà nó sẽ gọi keeper thực hiện khi nó nhận bắt một Msg nhất định.

// NewHandler creates the msg handler of this module, as required by Cosmos-SDK standard.

func NewHandler(keeper Keeper) sdk.Handler {

return func(ctx sdk.Context, msg sdk.Msg) (\*sdk.Result, error) {

ctx = ctx.WithEventManager(sdk.NewEventManager())

switch msg := msg.(type) {

case MsgSetProduct:

return handleMsgSetProduct(ctx, keeper, msg)

default:

return nil, sdkerrors.Wrapf(sdkerrors.ErrUnknownRequest, "unrecognized %s message type: %T", ModuleName, msg)

}

}

}

func handleMsgSetProduct(ctx sdk.Context, keeper Keeper, msg MsgSetProduct) (\*sdk.Result, error) {

key := "Product-" + msg.ProductID

var product = Product{

ProductID: msg.ProductID,

Description: msg.Description,

Owner: msg.Signer,

}

keeper.SetProduct(ctx, key, product)

return &sdk.Result{}, nil // return

}

**3.4 Client**

Là thành phần giúp người dùng có thể tương tác với ứng dụng. Có thể là command-line hoặc API server.