用 Golang 實作 Clean Architecture

- Tung

https://github.com/**Lockingdong/url-shortener**

Profile

Tung 東東

- Backend Engineer at XRSPACE
- 4.5 years' experience in Web Development
- Backend: Golang、PHP
- Frontend: Vue
- Hobbies: coffee、fitness、locking

階層式架構的問題點

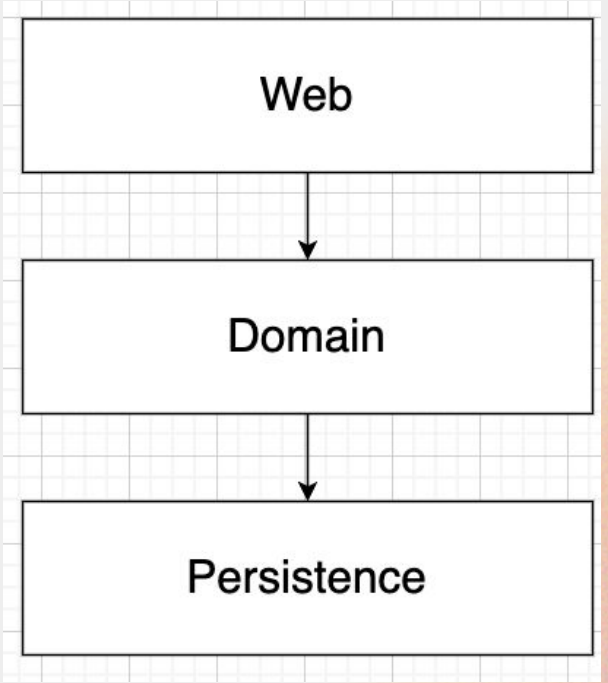
Troubles to Layered Architecture

階層式架構

網頁層

業務邏輯層

儲存層

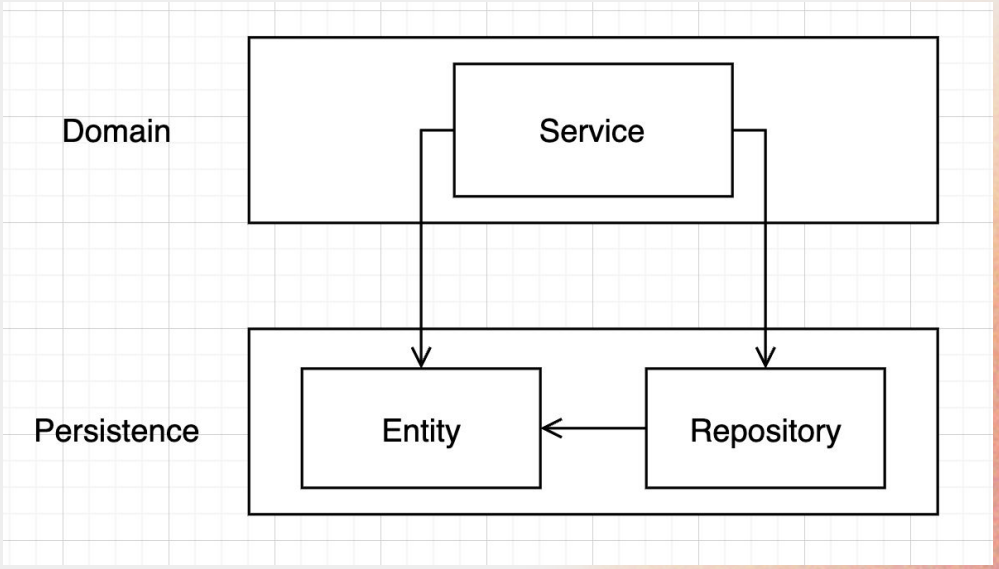


V C

M

M

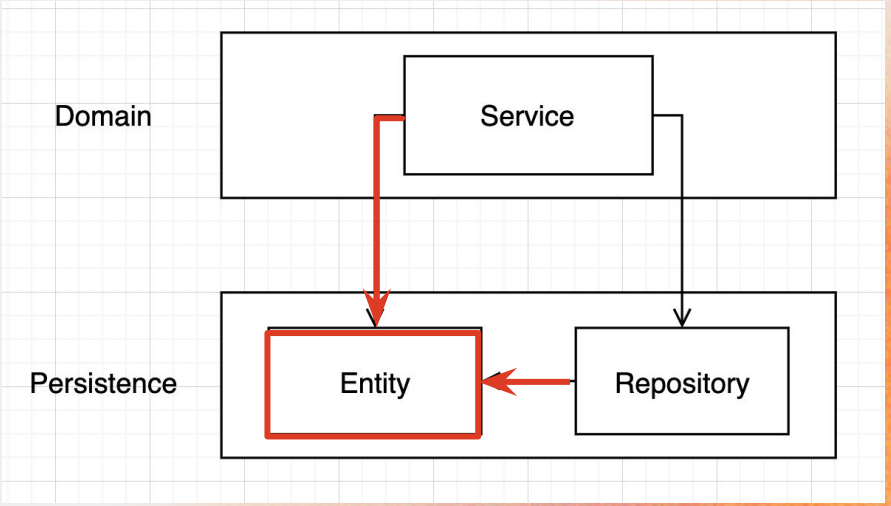
階層式架構



階層式架構

Q: 所以問題點是？

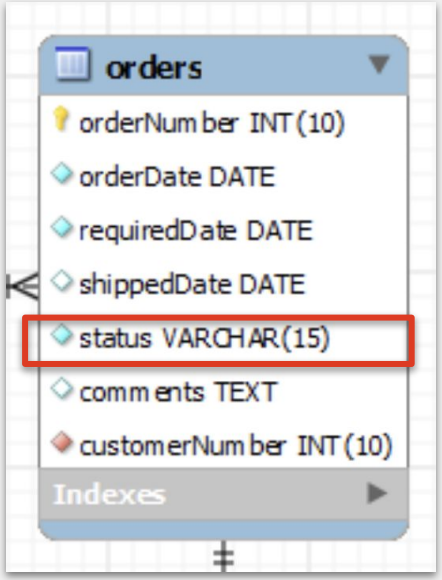
A: 資料庫驅動



階層式架構

舉例：想獲取訂單的**狀態**

Order Table



Order Model / Entity

```php
class Order extends Model
{
    protected $fillable = [
        'order_number',
        'order_date',
        'required_date',
        'shipped_date',
        'status',
        'comments',
        'customer_number'
    ];
}
```

階層式架構

修改實體就要考慮到資料庫似乎不太合理

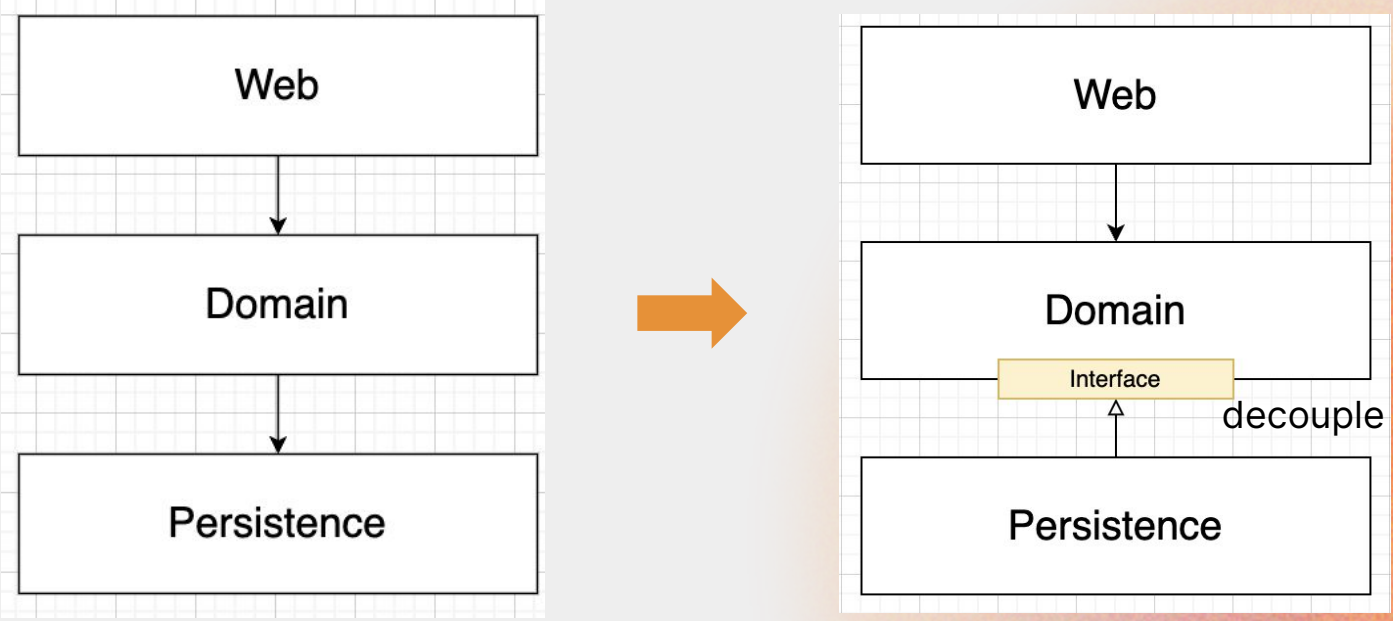
能不能只關注在**實體或業務邏輯？**

Clean Architecture

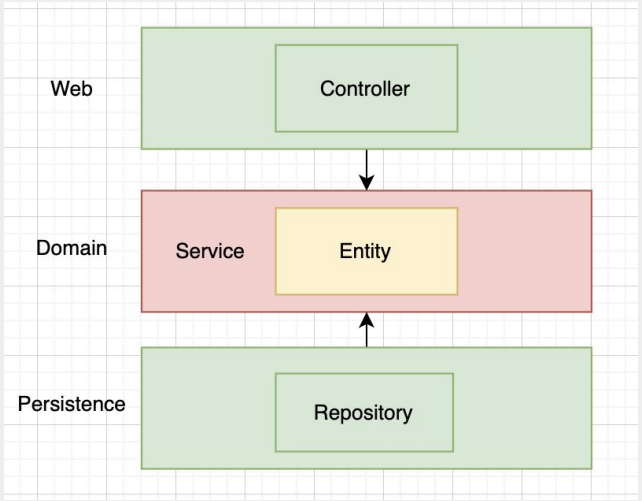
Introduction to Clean Architecture

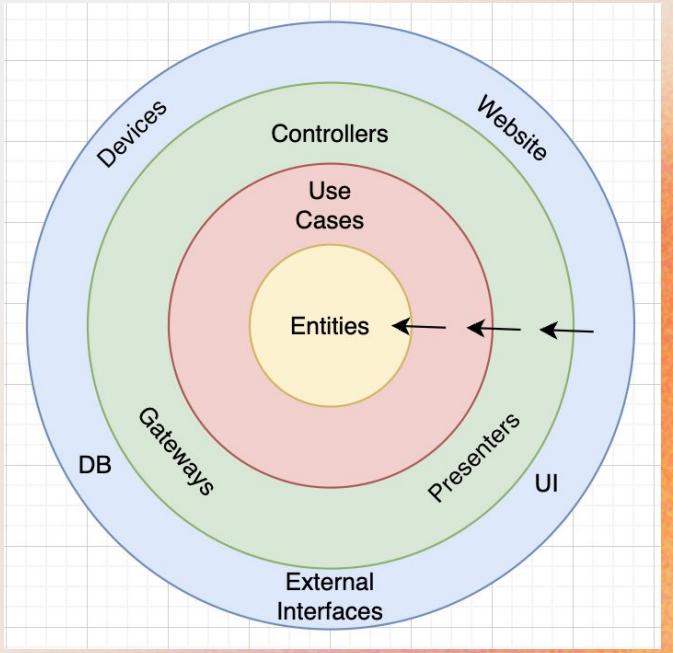
Clean Architecture

依賴反轉 (DIP)



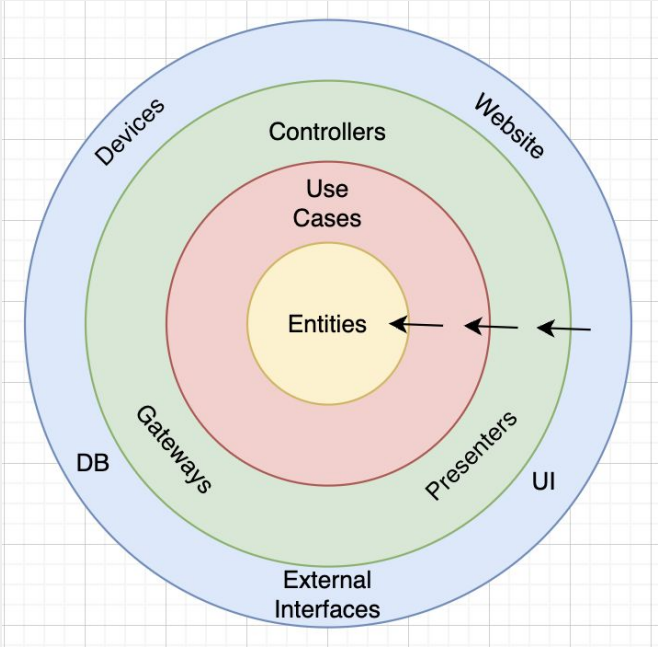
Clean Architecture





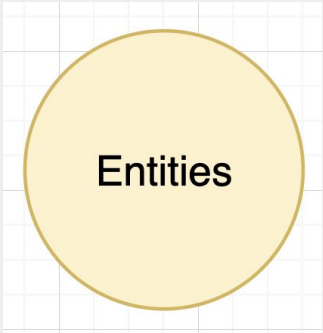
Clean Architecture

原則:只能向內依賴



Clean Architecture

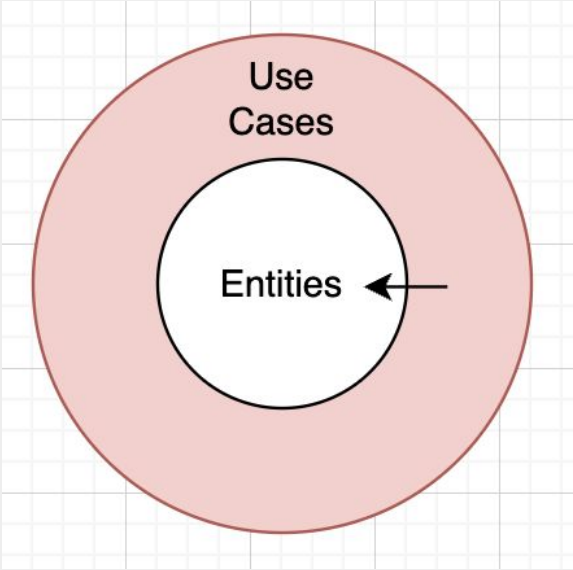
Entites: Business Rules. A data structure with methods.




```go
type Post struct {
    ID          string
    Title       string
    Content     string
    DisabledAt time.Time
}

func (p *Post) Disable() {
    p.DisabledAt = time.Now()
}
```

Clean Architecture

Application

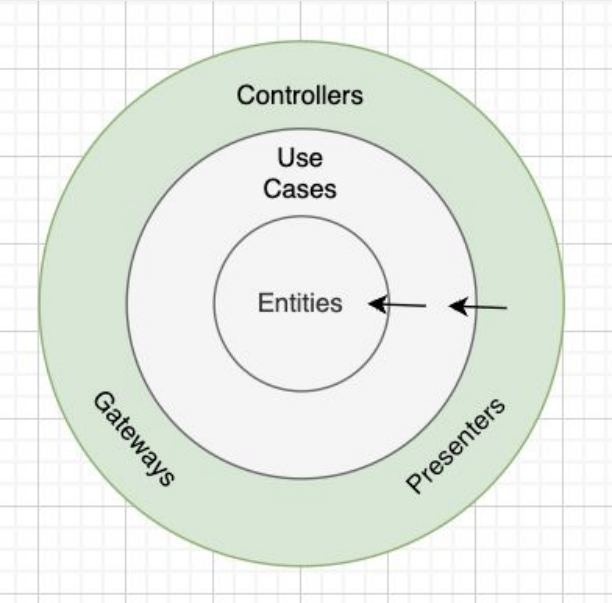


Application: Business Rules (fine-grained service). Interact with entities.

```go
type CreatePostUseCase struct {
    repository ICreatePostRepository
}

func (u *CreatePostUseCase) Execute(cmd *CreatePostCommand) (*CreatePostResp, error) {

    post := entity.NewPost(
        "test_title",
        "test_content",
    )

    if err := u.repository.SavePost(post); err != nil {
        return nil, err
    }

    return &CreatePostResp{}, nil
}
```

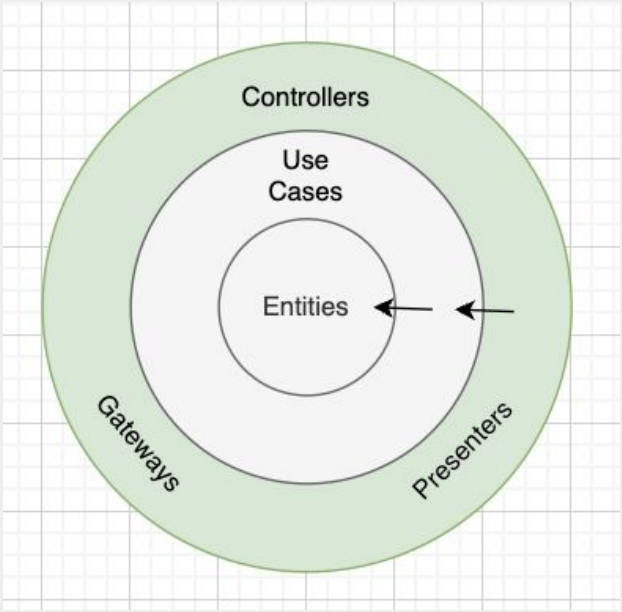
Clean Architecture

Adapters



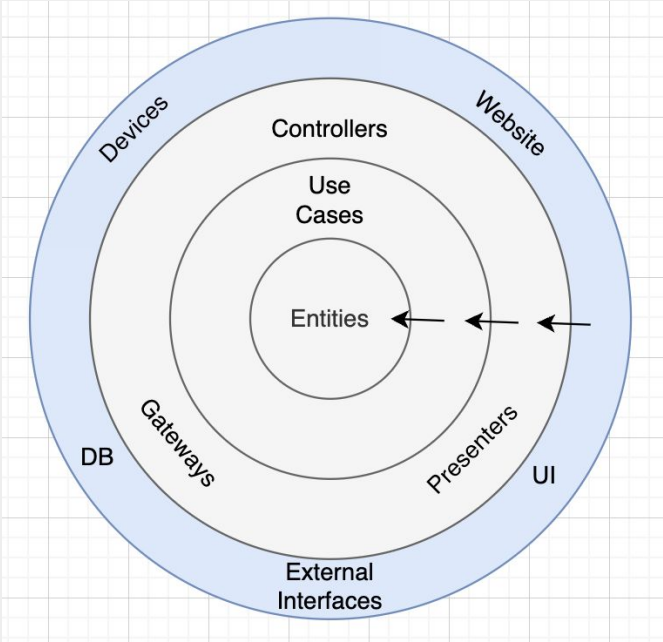
Adapters: Communicate with the inner and the outer layer. Not Business Rules.

Clean Architecture




```go
type Controller struct {
    CreatePostUseCase application.ICreatePostUseCase
}

func (c *Controller) CreatePost(ctx *gin.Context) {

    cmd := &application.CreatePostCommand{}
    if err := ctx.ShouldBind(cmd); err != nil {
        ctx.JSON(
            http.StatusBadRequest,
            err.Error(),
        )
        return
    }

    result, err := c.CreatePostUseCase.Execute(cmd)
    if err != nil {
        ctx.JSON(
            http.StatusInternalServerError,
            err.Error(),
        )
        return
    }

    ctx.JSON(
        http.StatusOK,
        result,
    )
}
```

Clean Architecture



Frameworks and Drivers

gRPC

REST API

MySQL

mongoDB

RabbitMQ

redis

Clean Architecture

Layered Architecture	Clean Architecture
prone to **Database Driven** 資料庫驅動	**Domain Driven** 領域(業務)驅動

Event Storming

Introduction to Event Storming

Event Storming

Q: 為什麼無法完成需求？

認知不同

Event Storming

未對業務場景取得共識的情況下：

媽媽：小明你去買麥當勞回來

結果，小明買了一整間麥當勞

Event Storming

What is the purpose of **Event Storming** ?

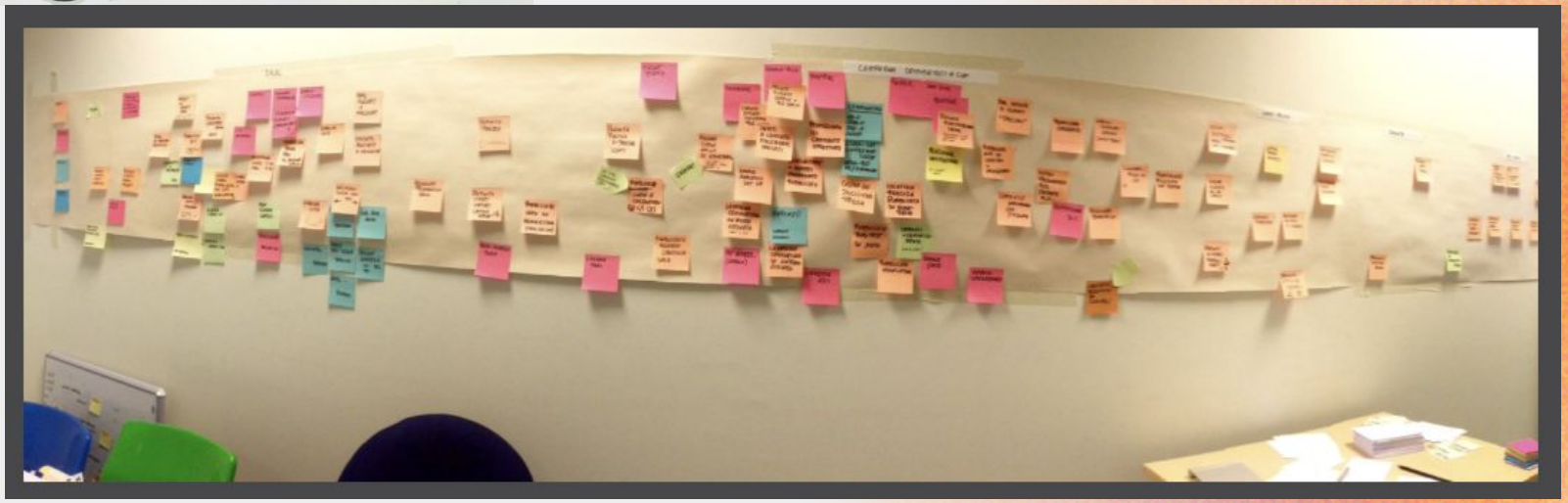
取得對業務場景的共識並對業務需求有一致性的認知

Event Storming

麥當勞是**店還是餐？**

定義通用語言

Event Storming



Event Storming

Q: 如何進行？

Step 1 - 準備好很長的白板、各種顏色的便利貼

Step 2 - 召集相關人員，必須包含 **Domain Expert** （PO、PM、RD）

Step 3 - 將系統內**已知的事件(Domain Event)**寫在便條紙上，並照時間先後順序由左至右排列

Step 4 - Google ...

Event Storming

便利貼顏色介紹

Domain Event. N + V-ed
Significant in system

Command or action.
present tense.

Read Model. Input or
Output Model.

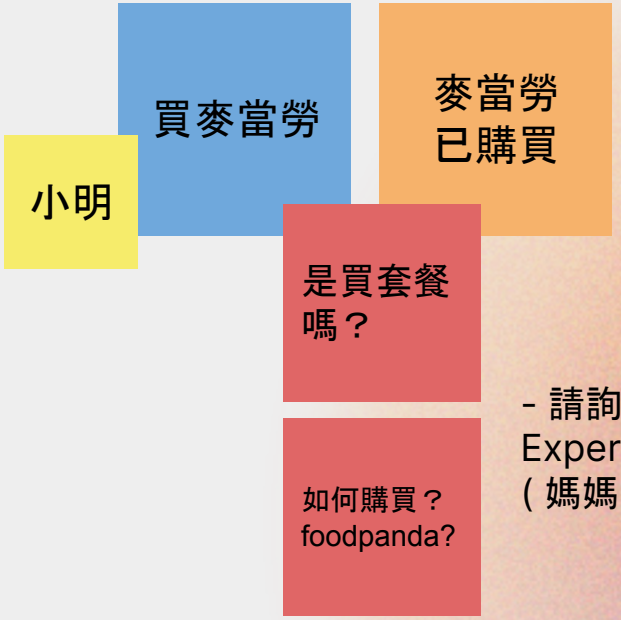
Actor

System

Hotspot

Event Storming

以小明買麥當勞為例子



- 請詢問 Domain Expert
(媽媽)

Event Storming

小明請用foodpanda買麥當勞2號餐

Clean Architecture

PM: 我需要短網址功能

蛤？！

Event Storming

短網址功能需求介紹

1. 輸入**網址** → 取得**短網址代碼**
2. 輸入**短網址代碼** → 取得**網址**

Event Storming

定義通用語言:

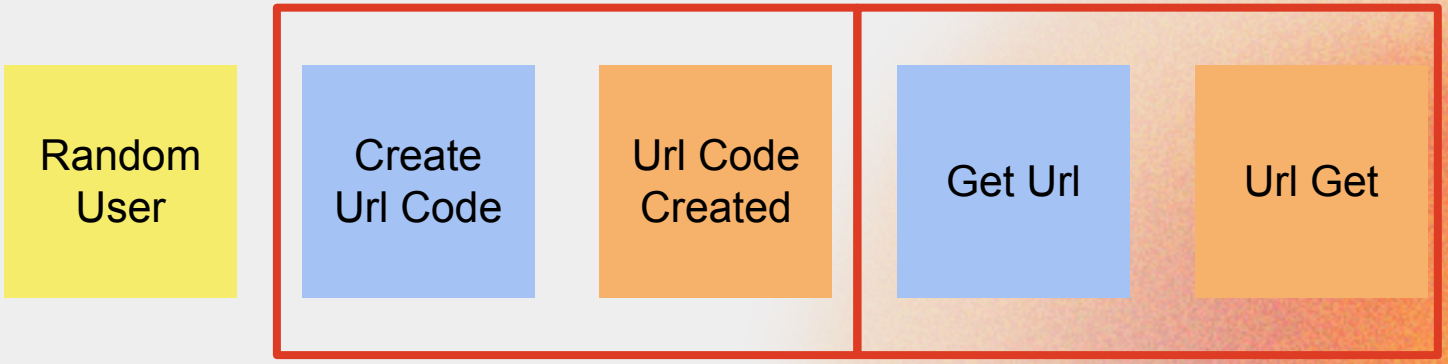
專案名稱 → **Url Shortener**

短網址代碼 → **Url Code**

網址 → **Url**

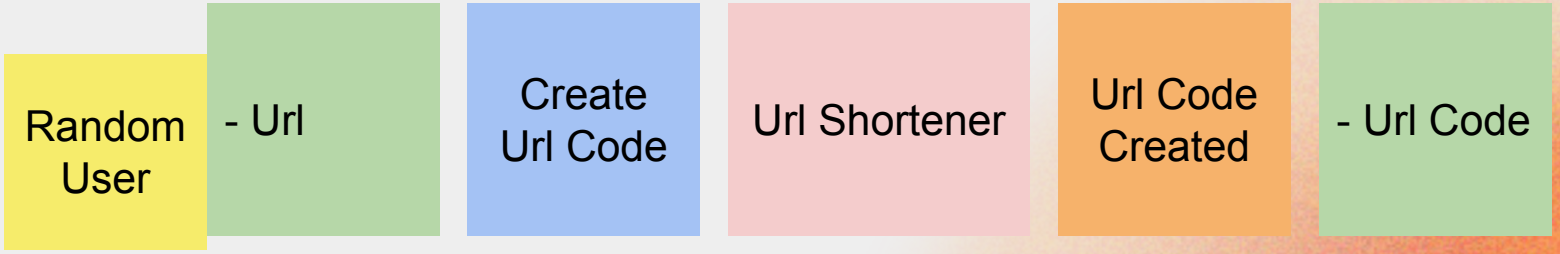
Event Storming

Url Shortener



Event Storming

Create Url Code

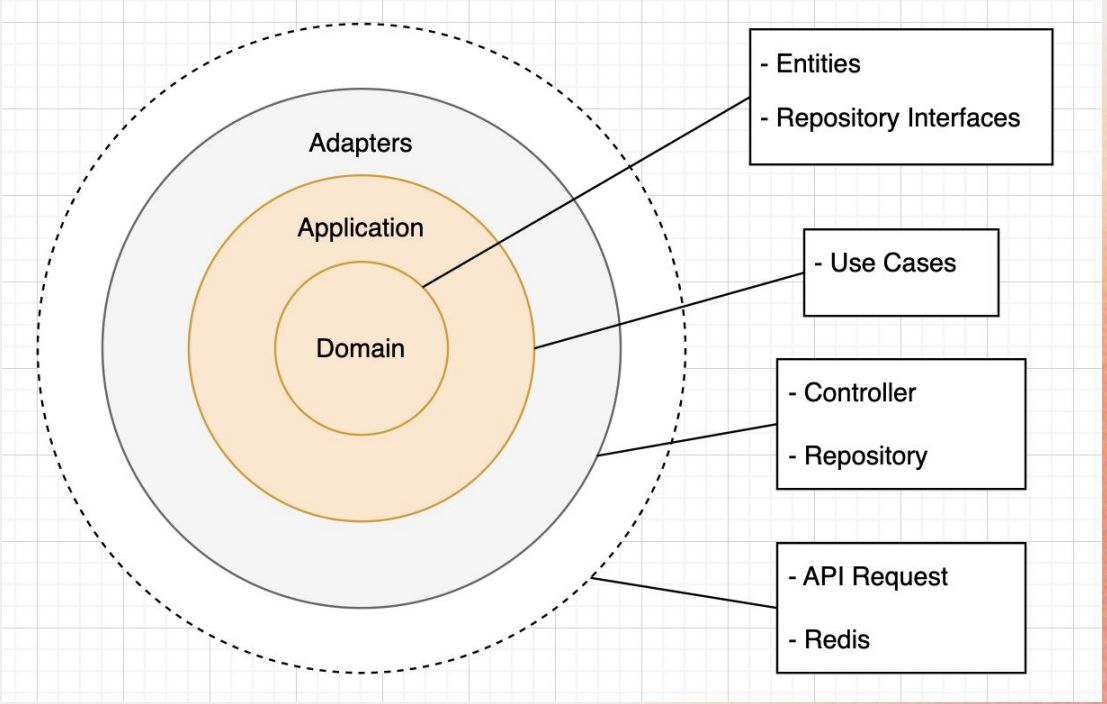


Get Url

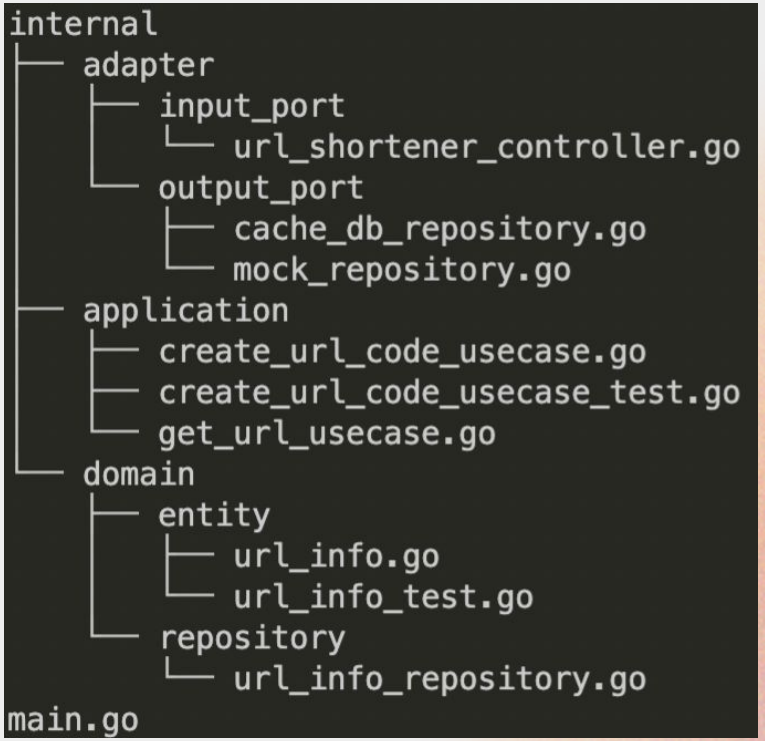
Implementation

Implemention of Clean Architecture

Implementation



Implementation



Implementation

建模 (Modeling)

src/domain/entity/url_info.go

UrlInfo

-ID (*required)

-Url

-UrlCode

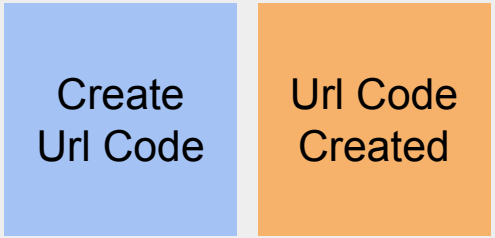
-CreatedAt

```
type UrlInfo struct {
    CreatedAt time.Time
    ID        string
    Url       string
    UrlCode   string
}
```

Implementation

Use Case: Create Url Code

src/application/create_url_code_usecase.go

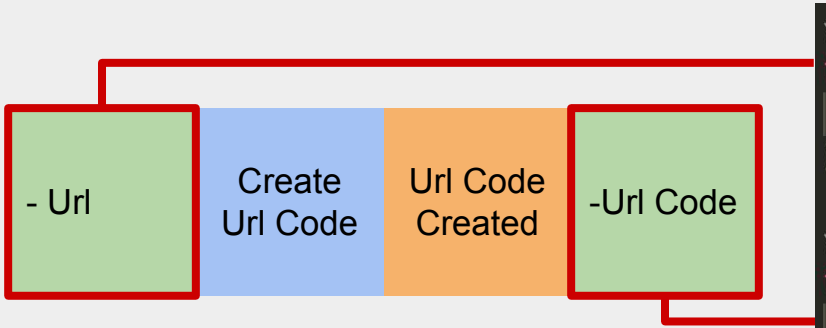



```go
func (u *CreateUrlCodeUseCase) Execute(
    ctx context.Context,
    cmd *CreateUrlCodeCommand,
) (*CreateUrlCodeResponse, error) {

    urlCode := shortid.MustGenerate()

    urlInfo := entity.NewUrlInfo(&entity.UrlInfoParams{
        Url:     cmd.Url,
        UrlCode: urlCode,
    })

    if err := u.repository.SaveUrlInfo(ctx, urlInfo); err != nil {
        return nil, err
    }

    return &CreateUrlCodeResponse{
        UrlCode: urlCode,
    }, nil
}
```

Implementation

Use Case: Create Url Code

src/application/create_url_code_usecase.go




```
You, 14 分鐘前 | 1 author (You)
type CreateUrlCodeCommand struct {
    Url string `json:"url" binding:"required,uri"`
}

You, 36 分鐘前 | 1 author (You)
type CreateUrlCodeResponse struct {
    UrlCode string `json:"url_code"`
}
```

Implementation

src/application/create_url_code_usecase.go

```go
type CreateUrlCodeUseCase struct {
    repository repository.ICreateUrlCodeRepository
}
```

src/domain/repository/url_info_repository.go

```go
type ICreateUrlCodeRepository interface {
    SaveUrlInfo(ctx context.Context, ent *entity.UrlInfo) error
}
```

Implementation

src/adapter/output_port/mock_repository.go

```go
func (r *UrlInfoMockRepository) SaveUrlInfo(
    _ context.Context,
    ent *entity.UrlInfo,
) error {

    dto := &UrlInfoMockDTO{
        ID:        ent.ID,
        Url:       ent.Url,
        UrlCode:   ent.UrlCode,
        CreatedAt: ent.CreatedAt,
    }

    r.urlInfos[dto.UrlCode] = dto

    return nil
}
```

Domain Entity to Persistence Entity

src/adapter/input_port/url_shortener_controller.go

```go
func (c *UrlShortenerController) CreateUrlCode(ctx *gin.Context) {

    cmd := &application.CreateUrlCodeCommand{}
    if err := ctx.ShouldBind(cmd); err != nil {
        ctx.JSON(
            http.StatusBadRequest,
            err.Error(),
        )
        return
    }

    result, err := c.createUrlCodeUseCase.Execute(ctx, cmd)
    if err != nil {
        ctx.JSON(
            http.StatusInternalServerError,
            err.Error(),
        )
        return
    }

    ctx.JSON(
        http.StatusOK,
        result,
    )
}
```

Implementation

main.go

```
repo := output_port.NewUrlInfoMockRepository(nil)
controller := input_port.NewUrlShortenerController(repo)

r := gin.Default()
r.GET("/", func(ctx *gin.Context) {
    ctx.JSON(http.StatusOK, "Hello, Url Shortener!")
})

r.POST("/api/short_url", controller.CreateUrlCode)
r.GET("/api/short_url", controller.GetUrl)
r.Run(":8000")
```

Implementation

怎麼沒看到 **Get Url** 的實作？

請開始你的表演

Thank you for your listening

@medium

@nomadortw

參考資料

- **Clean Architecture -** https://www.tenlong.com.tw/products/9789864342945
- **Clean Architecture 實作篇 -** https://www.tenlong.com.tw/products/9786263331815
- **Introducing EventStorming -** https://leanpub.com/introducing_eventstorming
- **Event Storming - what it is and why you should use it with Domain-Driven Design** - https://youtu.be/7LFxWgfJEeI