使用 Wire 簡化 Dependency Injection

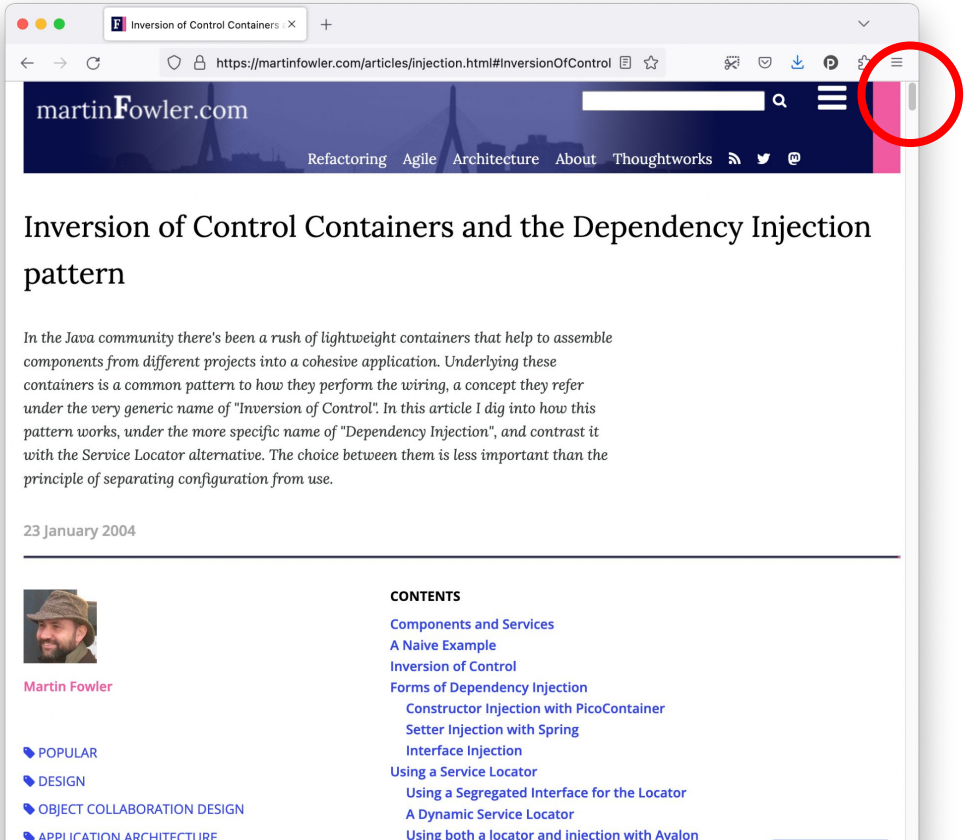
by Kenji Pa, Oursky

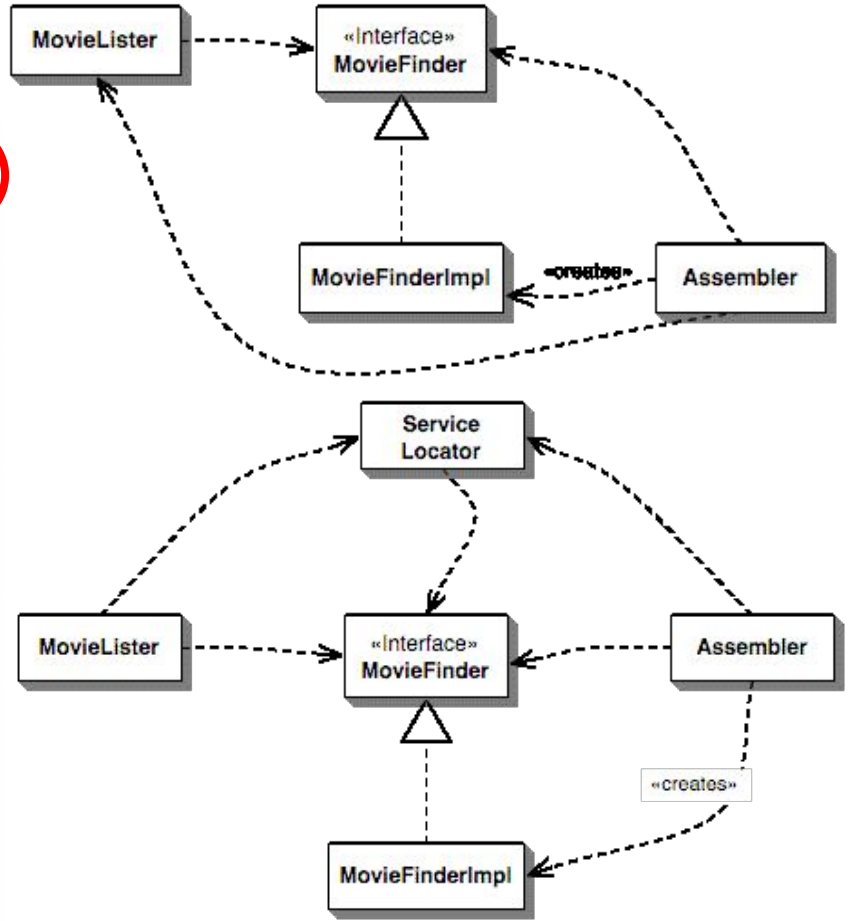
今天講什麼

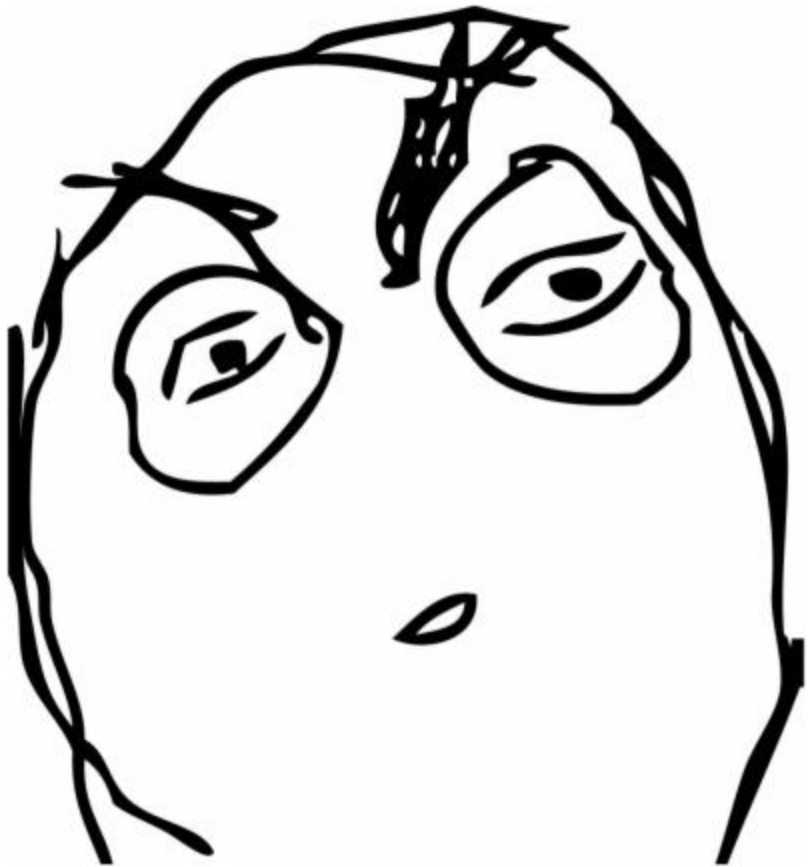
1. Dependency Injection (DI、依賴注入) 是什麼
2. 如何使用 Wire 自動化 DI 相關的 code
3. Wire 的實際應用
4. Q&A

DI 是什麼?

A Google search…







沒有 DI 的世界

```go
type BlogPost string

// 假設是個只有一個 BlogPost 的 Database
type Database struct{}

func NewDatabase() Database {
    return Database{}
}

func (db *Database) GetBlogPost() BlogPost {
    return BlogPost("It's a blog post.")
}
```

```go
type BlogPostService struct{}

func NewBlogPostService() BlogPostService {
    return BlogPostService{}
}

func (p *BlogPostService) PrintBlogPost() {
    database := NewDatabase()
    post := database.GetBlogPost()
    fmt.Println(post)
}

func main() {
    service := NewBlogPostService()
    service.PrintBlogPost()
}
```

沒有 DI 的世界

```go
type BlogPost string

// 假設是個只有一個 BlogPost 的 Database
type Database struct{}

func NewDatabase() Database {
    return Database{}
}

func (db *Database) GetBlogPost() BlogPost {
    return BlogPost("It's a blog post.")
}
```

```go
type BlogPostService struct{}

func NewBlogPostService() BlogPostService {
    return BlogPostService{}
}

func (p *BlogPostService) PrintBlogPost() {
    database := NewDatabase()
    post := database.GetBlogPost()
    fmt.Println(post)
}

func main() {
    service := NewBlogPostService()
    service.PrintBlogPost()
}
```

沒有 DI 的世界

- NewDatabase signature 改變的話，PrintBlogPost 的 implementation 也會被動到
- 相同的 initialization 會出現在所有要用到 Database 的地方
- Database 沒有辦法被 mock，測試變得困難

手動 DI 的世界

```go
type BlogPostService struct {
    database Database // 增加 database
}

// 從外部接收 database
func NewBlogPostService(database Database)
BlogPostService {
    return BlogPostService{
        database: database,
    }
}

func (p *BlogPostService) PrintBlogPost() {
    // 直接使用已有的 database
    post := p.database.GetBlogPost()
    fmt.Println(post)
}
```

```go
func main() {
    database := NewDatabase()

    service := NewBlogPostService(database)
    service.PrintBlogPost()
}
```

Wire 的世界

```
// wire.go
//go:build wireinject
// +build wireinject

package main

import "github.com/google/wire"

func InitializeBlogPostService()
BlogPostService {
   panic(wire.Build(NewBlogPostService,
NewDatabase))
}
```

```
func main() {
   service := InitializeBlogPostService()
   service.PrintBlogPost()
}
```

Wire 的世界

```
// Injectors from wire.go:

func InitializeBlogPostService() BlogPostService {
   database := NewDatabase()
   blogPostService := NewBlogPostService(database)
   return blogPostService
}
```

¯_(ツ)_/¯

才幾句，怎麼不自己寫

現實中的手動 DI

https://github.com/SkygearIO/skygear-server/blob/a8704c2697533bcacfae8b246563bdec32d4834b/pkg/auth/inject.go

¯_(ツ)_/¯

怎麼不用其他

為什麼用 Wire

- 唯一的 Compile-time DI 工具
- 生成的 code 可以直接 debug

- Google 出品應該會 maintain…吧？

在 Authgear 的使用例

```
// https://github.com/authgear/authgear-server/blob/master/cmd/authgear/elasticsearch/wire.go
func NewAppLister(
   ctx context.Context,
   pool *db.Pool,
   databaseCredentials *config.DatabaseCredentials,
) *AppLister {
   panic(wire.Build(DependencySet))
}

func NewReindexer(
   ctx context.Context,
   pool *db.Pool,
   databaseCredentials *config.DatabaseCredentials,
   appID config.AppID,
) *Reindexer {
   panic(wire.Build(DependencySet))
}
```

在 Authgear 的使用例

```
// https://github.com/authgear/authgear-server/blob/master/cmd/authgear/elasticsearch/wire.go
func NewAppLister(
    ctx context.Context,
    pool *db.Pool,
    databaseCredentials *config.DatabaseCredentials,
) *AppLister {
    panic(wire.Build(DependencySet))
}

func NewReindexer(
    ctx context.Context,
    pool *db.Pool,
    databaseCredentials *config.DatabaseCredentials,
    appID config.AppID,
) *Reindexer {
    panic(wire.Build(DependencySet))
}
```

Wire: Injector 可以收 Argument

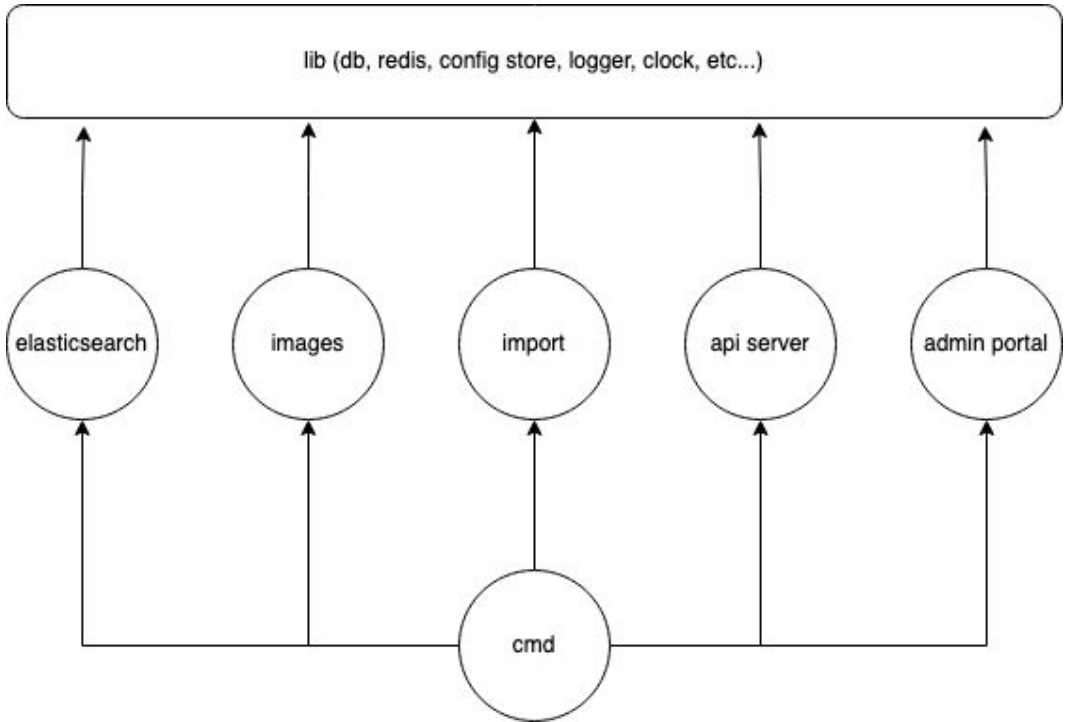
```
// Injectors from wire.go:

func NewAppLister(ctx context.Context, pool *db.Pool, databaseCredentials *config.DatabaseCredentials) *AppLister {
   globalDatabaseCredentialsEnvironmentConfig := NewGlobalDatabaseCredentials(databaseCredentials)
   databaseEnvironmentConfig := config.NewDefaultDatabaseEnvironmentConfig()
   factory := NewLoggerFactory()
   handle := globaldb.NewHandle(ctx, pool, globalDatabaseCredentialsEnvironmentConfig, databaseEnvironmentConfig, factory)
   sqlBuilder := globaldb.NewSQLBuilder(globalDatabaseCredentialsEnvironmentConfig)
   sqlExecutor := globaldb.NewSQLExecutor(ctx, handle)
   store := &configsource.Store{
       SQLBuilder:  sqlBuilder,
       SQLExecutor: sqlExecutor,
   }
   appLister := &AppLister{
       Handle: handle,
       Store:  store,
   }
   return appLister
}
```

相關 Dependency 可以重用

```
// https://github.com/authgear/authgear-server/blob/master/cmd/authgear/elasticsearch/deps.go
var DependencySet = wire.NewSet(
   NewLoggerFactory,
   config.NewDefaultDatabaseEnvironmentConfig,
   NewGlobalDatabaseCredentials,
   NewEmptyIdentityConfig,
   globaldb.DependencySet,
   appdb.NewHandle,
   appdb.DependencySet,
   clock.DependencySet,
   wire.Struct(new(user.Store), "*"),
   wire.Struct(new(identityoauth.Store), "*"),
   wire.Struct(new(identityloginid.Store), "*"),
   wire.Struct(new(configsource.Store), "*"),
   wire.Struct(new(AppLister), "*"),
   wire.Struct(new(Reindexer), "*"),
)
```

Wire: Dependency Set

- Authgear: 每個 package 在 `deps.go` 裡面提供 DependencySet 給上一層



可以的話我也不想用，但 code 實在是太多了

```
> wc -l $(find . -name 'wire_gen.go')
    719 ./cmd/authgear/background/wire_gen.go
     32 ./cmd/authgear/importer/wire_gen.go
     79 ./cmd/authgear/images/server/wire_gen.go
     76 ./cmd/authgear/server/wire_gen.go
     76 ./cmd/authgear/elasticsearch/wire_gen.go
     51 ./cmd/portal/cmd/cmdpricing/wire_gen.go
     49 ./cmd/portal/plan/wire_gen.go
     62 ./cmd/portal/server/wire_gen.go
     57 ./cmd/portal/usage/wire_gen.go
    190 ./cmd/portal/analytic/wire_gen.go
    557 ./pkg/portal/wire_gen.go
     37 ./pkg/portal/smtp/wire_gen.go
     30 ./pkg/auth/handler/webapp/wire_gen.go
  49103 ./pkg/auth/wire_gen.go
    149 ./pkg/images/wire_gen.go
   1007 ./pkg/admin/wire_gen.go
     84 ./pkg/worker/wire_gen.go
    955 ./pkg/resolver/wire_gen.go
  53313 total
```

References

- https://martinfowler.com/articles/injection.html
- https://github.com/google/wire/blob/main/_tutorial/README.md
- https://github.com/google/wire/blob/main/docs/best-practices.md
- https://github.com/authgear/authgear-server
- https://github.com/SkygearIO/skygear-server


Q&A

Thank you for your time!

工商時間

工商時間

- 我們的產品 (˚∀˚)
  - authgear.com

  - formx.ai

- 我們在招人 (˚∀˚)

  go.oursky.com/careers