Building an **enterprise service** in Go

by example

Marcus Olsson
@marcusolsson

Marcus Olsson

@marcusolsson

Klarna™

previously

CITERUS

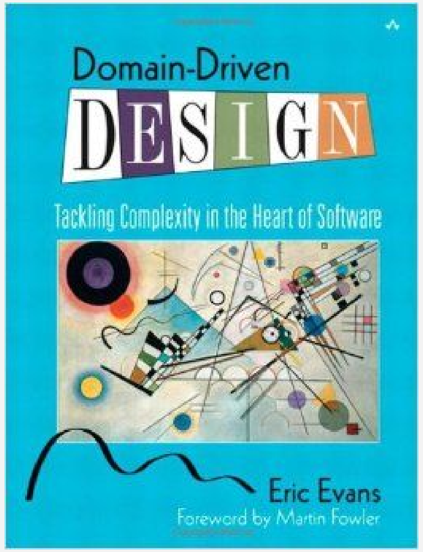
Real-world processes

↓

Code

Best practices for writing **business applications?**

Domain-Driven Design

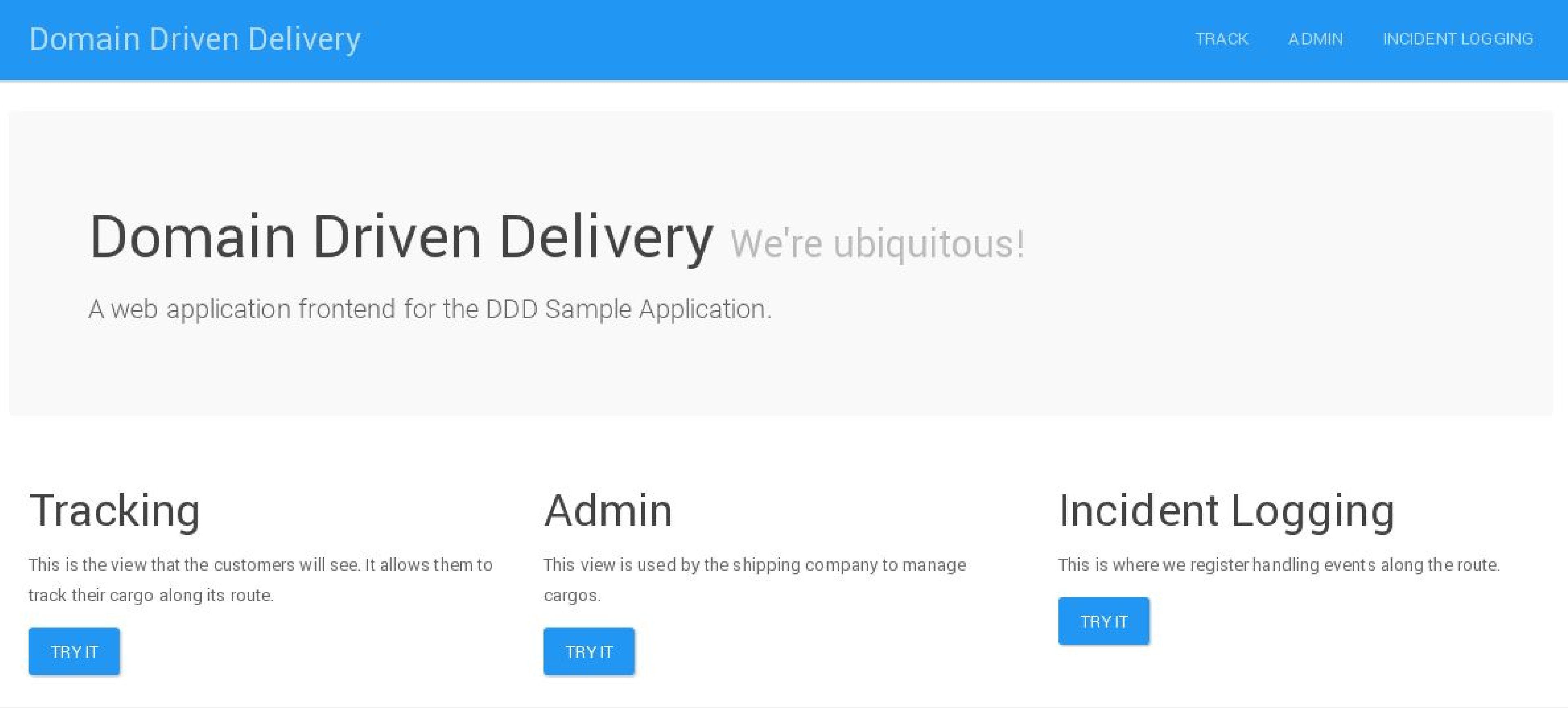

Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

DDD Sample App

http://dddsample.sourceforge.net

goddd

An **idiomatic** Go port
of the DDD Sample App


Domain Driven Delivery

Domain Driven Delivery We're ubiquitous!

A web application frontend for the DDD Sample Application.

Tracking

This is the view that the customers will see. It allows them to track their cargo along its route.

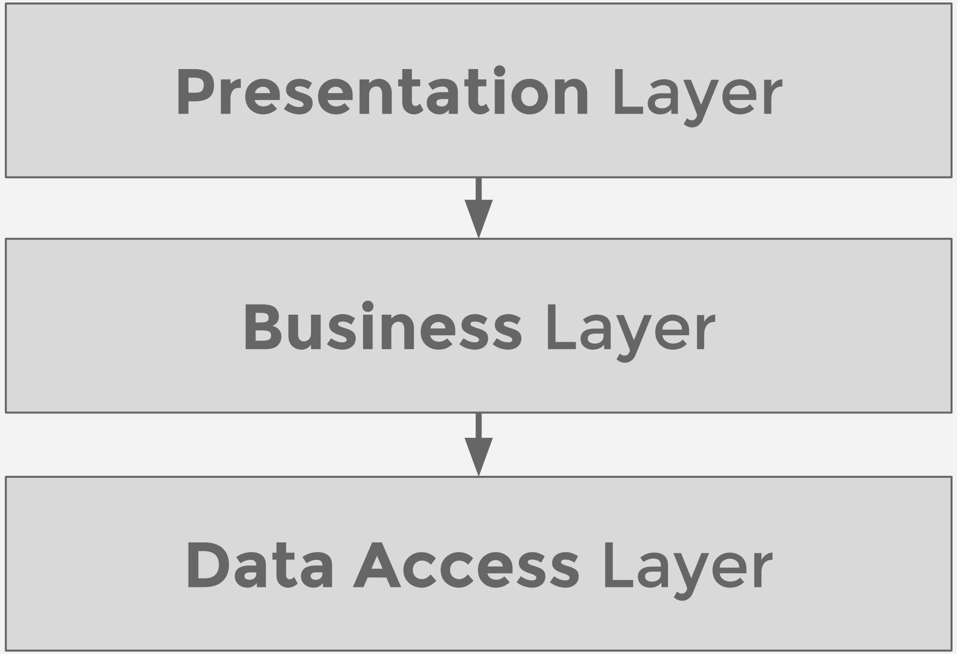
Admin

This view is used by the shipping company to manage cargos.

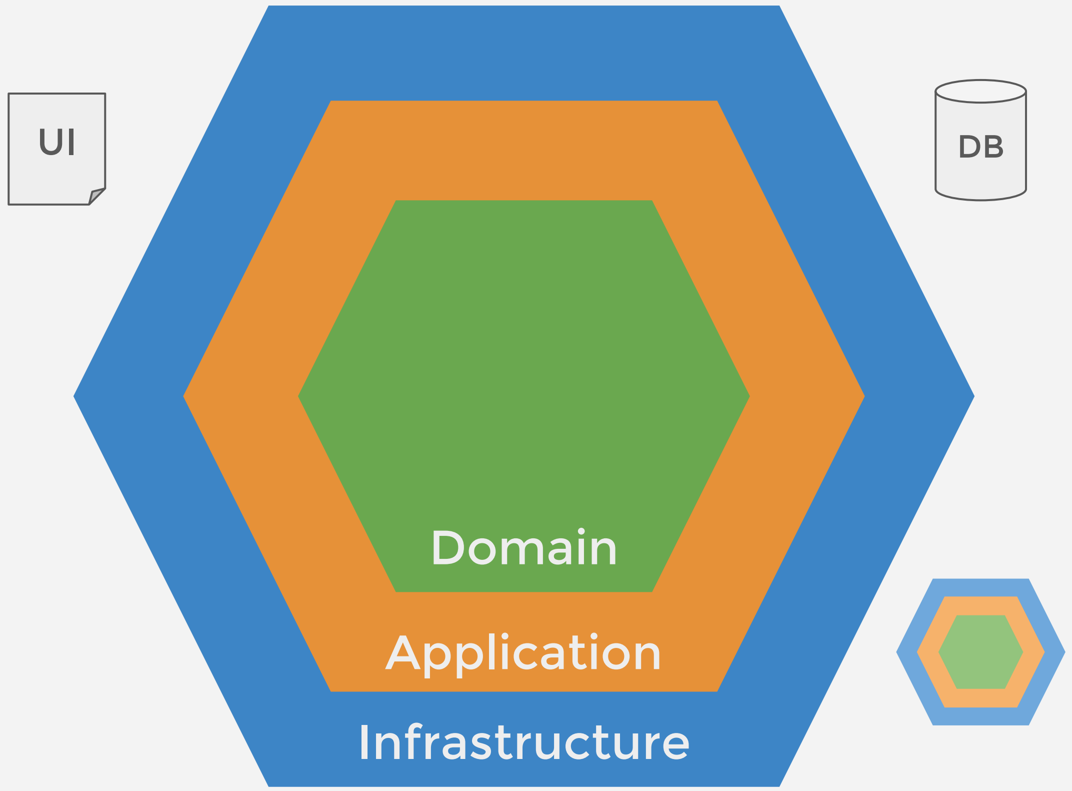
Incident Logging

This is where we register handling events along the route.

http://marcusolsson.github.io/dddelivery-angularjs


Presentation Layer
Business Layer
Data Access Layer

Clean Architecture



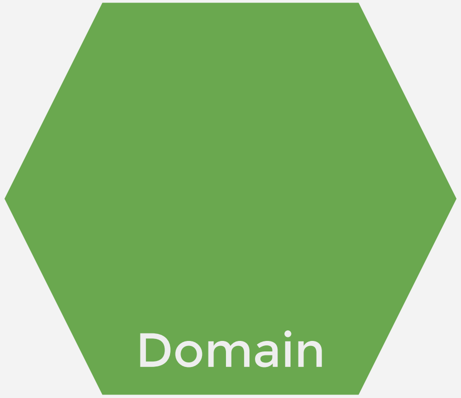
Inversion of control

Domain

```
type Repository interface {
	Store(cargo *Cargo) error
	Find(trackingID TrackingID) (*Cargo, error)
	FindAll() []*Cargo
}
```

Infrastructure

```
type cargoRepository struct {
	session *mgo.Session
}
func (r *cargoRepository) Store(cargo *Cargo) error { ... }
func (r *cargoRepository) Find(trackingID TrackingID) (*Cargo, error) { ... }
func (r *cargoRepository) FindAll() []*Cargo { ... }
```


Domain

Domain Objects

"An object defined primarily by its identity is called an ***ENTITY****."*

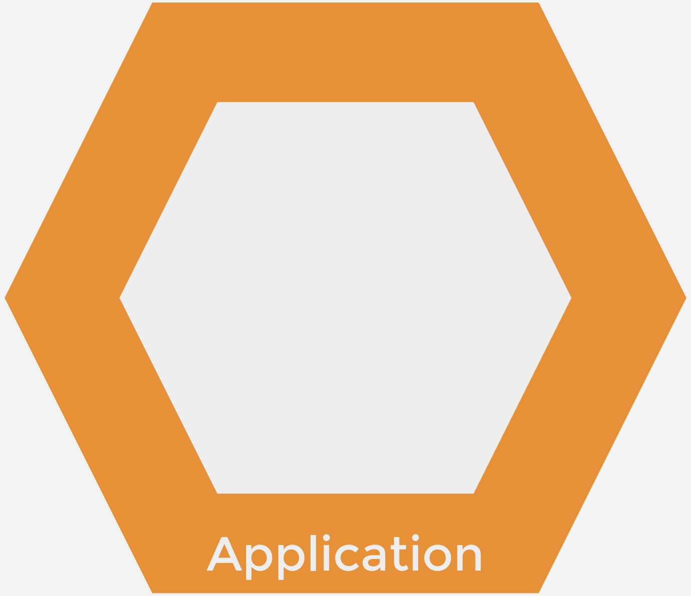
"An object that represents a descriptive aspect of the domain with no conceptual identity is called a ***VALUE OBJECT****."*

- Eric Evans
Domain-Driven Design

Domain objects as method receivers

```
type Cargo struct {
     ID        TrackingID // identity
     Itinerary Itinerary
}
func (c *Cargo) AssignToRoute(i Itinerary) { ... }

// vs.

type Itinerary struct {
    Legs []Leg
}
func (i Itinerary) IsEmpty() bool { ... }
```

```go
type RouteSpecification struct {
    Origin          location.UNLocode
    Destination     location.UNLocode
    ArrivalDeadline time.Time
}
```

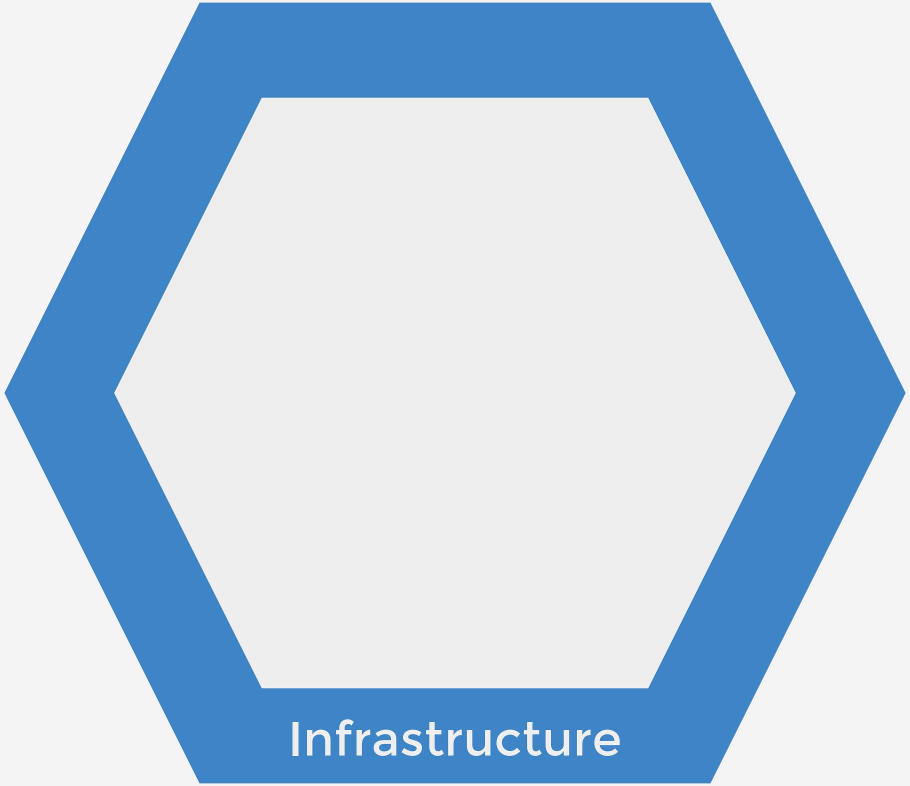

Application

Application Service: Booking

```go
type Service interface {
	// BookNewCargo registers a new cargo in the tracking system, not yet
	// routed.
	BookNewCargo(origin location.UNLocode,
		destination location.UNLocode,
		arrivalDeadline time.Time) (cargo.TrackingID, error)

	// AssignCargoToRoute assigns a cargo to the route specified by the
	// itinerary.
	AssignCargoToRoute(id cargo.TrackingID, itinerary cargo.Itinerary) error

	// ...
}
```

```go
func (s *service) BookNewCargo(origin, destination location.UNLocode, arrivalDeadline time.Time)
(cargo.TrackingID, error) {

	if origin == "" || destination == "" || arrivalDeadline.IsZero() {
		return "", ErrInvalidArgument
	}

	id := cargo.NextTrackingID()
	rs := cargo.RouteSpecification{
		Origin:          origin,
		Destination:     destination,
		ArrivalDeadline: arrivalDeadline,
	}

	c := cargo.New(id, rs)

	if err := s.cargos.Store(c); err != nil {
		return "", err
	}

	return c.TrackingID, nil
}
```

```go
func (s *service) AssignCargoToRoute(id cargo.TrackingID, itinerary cargo.Itinerary) error {
	// 1. Validate inputs
	if id == "" || len(itinerary.Legs) == 0 {
		return ErrInvalidArgument
	}

	// 2. Load the cargo
	c, err := s.cargos.Find(id)
	if err != nil {
		return err
	}

	// 3. Do your thing
	c.AssignToRoute(itinerary)

	// 4. Save the updated cargo
	return s.cargos.Store(c)
}
```


Infrastructure

Go kit

```go
type loggingService struct {
	logger log.Logger
	Service
}

func (s *loggingService) BookNewCargo(origin location.UNLocode, destination location.UNLocode,
	arrivalDeadline time.Time) (id cargo.TrackingID, err error) {

	defer func(begin time.Time) {
		s.logger.Log(
			"method", "book",
			"origin", origin,
			"destination", destination,
			"arrival_deadline", arrivalDeadline,
			"took", time.Since(begin),
			"err", err,
		)
	}(time.Now())

	return s.Service.BookNewCargo(origin, destination, arrivalDeadline)
}
```

```
        domain.NewCargo()
   application.BookingService()
infrastructure.NewPostgresCargoRepository()
```

😱

Domain modules

*“If your [domain] model is telling a **story**, the **MODULES** are **chapters**.”*

- Eric Evans

Modules (a.k.a Packages), *Domain-Driven Design*

Domain modules as subpackages

```
cargo/                                   // examples
  cargo.go                               cargo.TrackingID
  delivery.go
  handling.go
  itinerary.go
location/
  location.go                            location.UNLocode
voyage/
  voyage.go                              voyage.Number
```

Application services as subpackages

```
booking/
  service.go                                  booking.NewService()
  logging.go                                  booking.NewLoggingService()
  intrumenting.go                             booking.NewInstrumentingService()
  ...
tracking/
  service.go                                  tracking.NewService()
  logging.go                                  tracking.NewLoggingService()
  intrumenting.go                             tracking.NewInstrumentingService()
  ...
```

Dependencies as subpackages

```
// domain interface
cargo.Repository

// implementations
mongo.CargoRepository
mock.CargoRepository
```

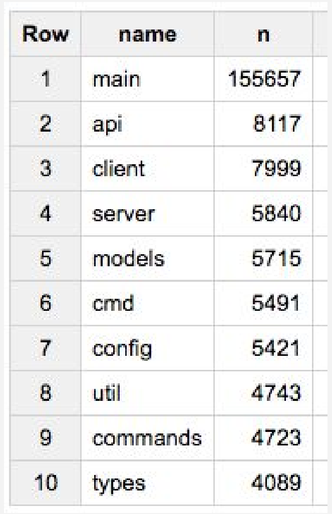
```
// application interface
inspection.EventHandler

// TODO: implementation
amqp.EventHandler
```

Wiring it up in `main`

```
var (
	cargos         = inmem.NewCargoRepository()
	locations      = inmem.NewLocationRepository()
	handlingEvents = inmem.NewHandlingEventRepository()

	logger         = log.NewLogfmtLogger(os.Stderr)
	requestCounter = kitprometheus.NewCounter(...)
)

// configure domain service
var rs routing.Service
rs = routing.NewProxyingMiddleware(*routingServiceURL, ctx)(rs)

// configure application services
var bs booking.Service
bs = booking.NewService(cargos, locations, handlingEvents, rs)
bs = booking.NewLoggingService(logger, bs)
bs = booking.NewInstrumentingService(requestCounter, bs)
```

Bonus: The **most** generic application

Most popular package names



Row	name	n
1	main	155657
2	api	8117
3	client	7999
4	server	5840
5	models	5715
6	cmd	5491
7	config	5421
8	util	4743
9	commands	4723
10	types	4089

Analyzing Go code with BigQuery
by Francesc Campoy

https://medium.com/google-cloud/analyzing-go-code-with-bigquery-485c70c3b451#.b70ku0721

```
api/
client/
config/
models/
server/
util/
main.go
```

A word of **caution**

Not **all** applications are alike.

The **Curse** of Sample Applications

Links

Demo
https://marcusolsson.github.io/dddelivery-angularjs

Frontend
https://github.com/marcusolsson/dddelivery-angularjs

Backend (also available as the Go kit *shipping* example)
https://github.com/marcusolsson/goddd

Mock routing service
https://github.com/marcusolsson/pathfinder

Blogged: Domain Driven Design in Go, part 1-3
http://marcusoncode.se

Go kit
https://github.com/go-kit/kit

Standard Package Layout, by Ben Johnson
https://medium.com/@benbjohnson/standard-package-layout-7cdbc8391fc1