Golang Indianapolis Meetup

An Exploration of Data Exchange in Go

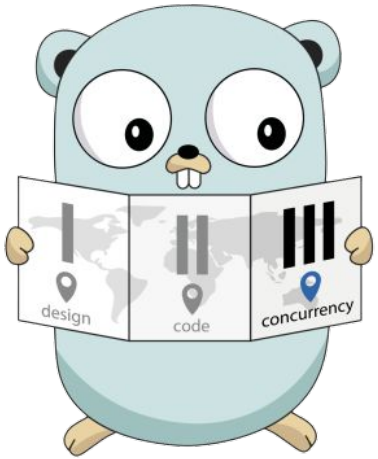
Nathan Boyd

Salesforce

github.com/nathan-boyd

Go's Values

Data Exchange - 1
Marshaling - 2
Custom Marshaling - 3



SECTION ONE

Data Exchange

What is Data Exchange

Data exchange is the transformation of data from one format into another.

Data Exchange - English Verbs

Encode: to convert from one system of communication into another

Marshal: to bring together and order in an appropriate or effective way

Serialize: to arrange or publish in serial form

Data Exchange - Computer Science Verbs

Encode: convert into a coded form

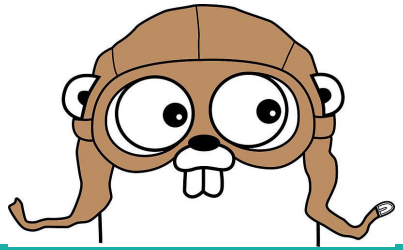
Marshal: the creation of a representation of data

Serialize: a marshaling process by which an object is converted **into bytes** for storage or transmission

Data Exchange - GO Verbs

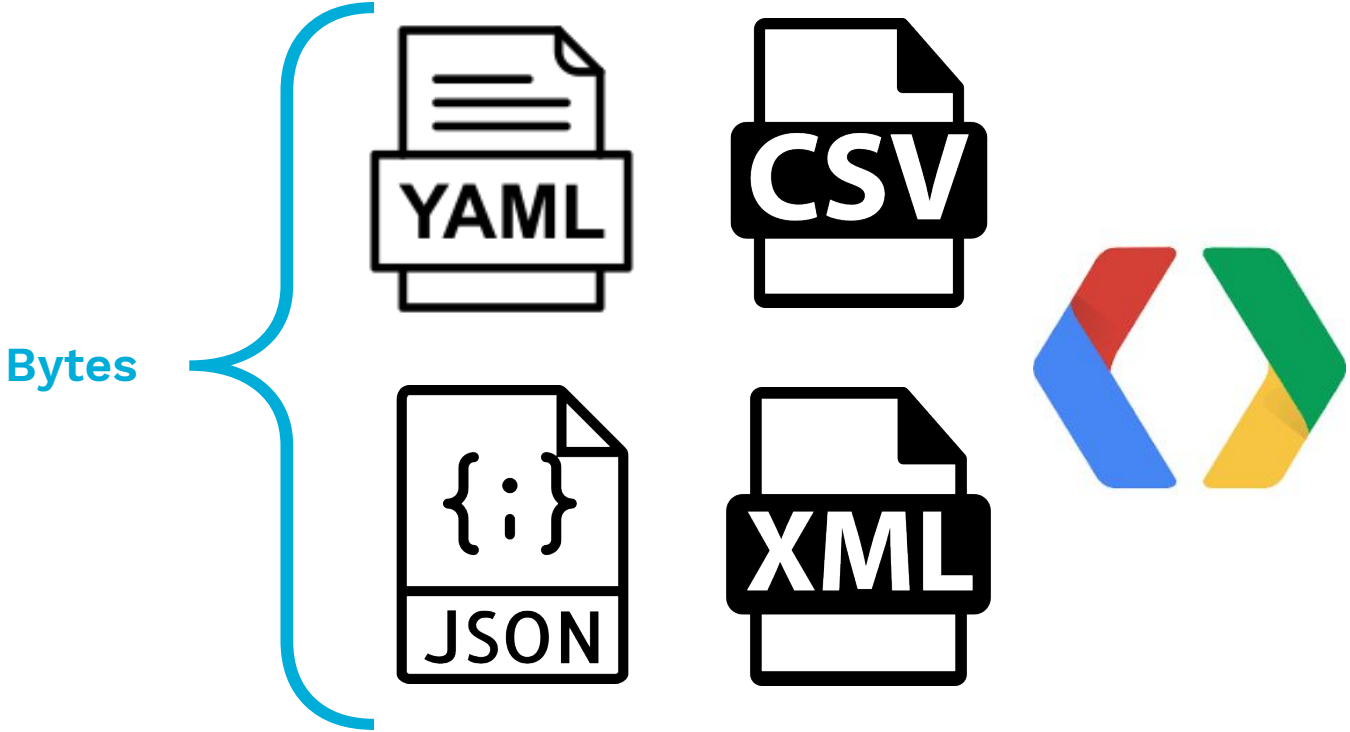
Encode: used when working with Go byte streams (unbounded)

Marshal: used when working with Go byte arrays or slices (bounded)



Data Exchange to bytes

Bytes



SECTION ONE

JSON Marshaling

JSON Marshaling and Unmarshaling

The Marshal function returns the JSON encoding of ‘v’

```
func Marshal(v interface{}) ([]byte, error)
```

The Unmarshal function parses the JSON-encoded data and stores the result in the value pointed to by v.

```
func Unmarshal(data []byte, v interface{}) error
```

SECTION TWO

Custom JSON Marshaling

Custom JSON Marshaling With Struct Tags

Struct tags allow you to attach metadata to the field which can be read using reflection.

There are many well known tags, several of them are for marshaling.

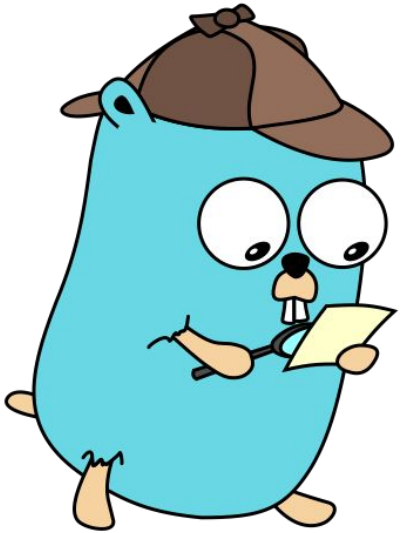
```
type Cat struct {
    Name  string `json:"pet_name"`
    Color string `json:"pet_color"`
}
```

Custom JSON Marshaling With Embedded Interfaces

Within the encoding/json package you'll find the Marshaler and Unmarshaler interfaces. These may be implemented to perform encoding operations to suit your needs.

The Marshaler Interface

```
type Marshaler interface {
    MarshalJSON() ([]byte, error)
}
```



Marshaler is the interface implemented by types that can marshal themselves into valid JSON

Custom JSON Marshaling With Embedded Interfaces

Go provides a Marshaler and Unmarshaler interface for most data exchange formats.

So if you're working with YAML, XML, or Protobufs the same pattern will likely available.

SECTION THREE

Summing up

Takeaways

Encode - used when working with Go byte streams (unbounded)

Marshal - used when working with Go byte arrays (bounded)

Takeaways

Struct Tags and Custom Marsherlers can be helpful for more complex transforms.

Takeaways

Go provides a Marshaler and Unmarshaler interface for most data exchange formats.

So if you're working with JSON, YAML, XML, or Protobufs the same patterns will likely available.

Resources

Playgrounds Based on the Demo

- Marshaling
- Unmarshaling
- Unmarshaling JSON String
- Embedding a custom Marshaler

Resources

Go Subclassing / Overriding via Embedding

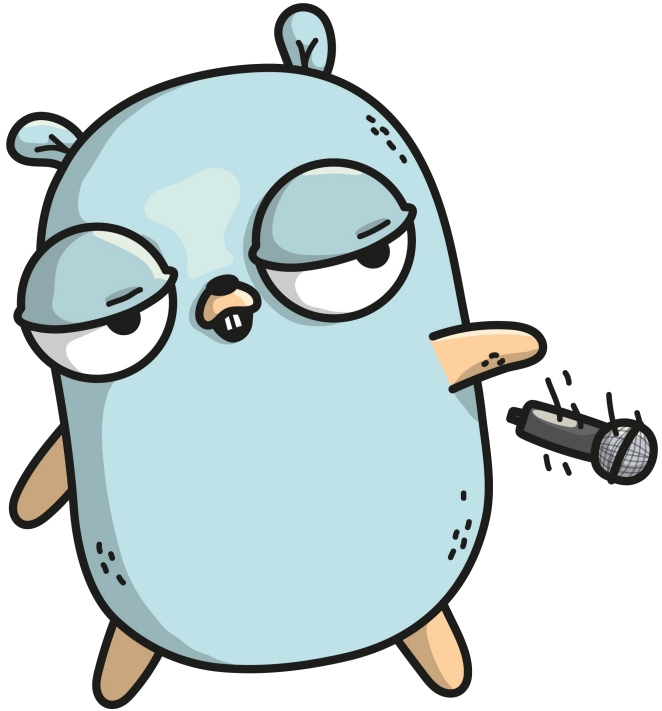
- Go Embedding

Tags in Go Struct types

- Go Struct Tag Reference
- Go well known Struct Tags
- Gopher Con Presentation Slides: The Many Faces of Go Tags
- GopherCon 2015: Sam Helman & Kyle Erf - The Many Faces of Struct Tags
- Go Struct Types

JSON IN GO

- Encoding / Json Package
- JSON and Go GoBLOG



Thanks!