**https://www.mindbowser.com/golang-data-structures-2/**

**What is JSON marshalling and Unmarshalling?**

**Marshalling and unmarshalling** is used both on the client and server side. On the server side it is used to map an incoming request to a Scala or Java object and to map a Scala or Java object to an outgoing response. Scala - spray-**json** Support. Java - Jackson Support.

Beside this, what is JSON marshalling?

**JSON** stands for JavaScript Object Notation, and it's a very handy way of exchanging structured data. And it's very popular, especially when interacting with APIs. Go's terminology calls **marshal** the process of generating a **JSON** string from a data structure, and unmarshal the act of parsing **JSON** to a data structure.

Additionally, what does Marshal mean in programming? In computer science, marshalling or marshaling **is** the process of transforming the memory representation of an object to a data format suitable for storage or transmission, and it **is** typically used when data must be moved between different parts of a computer **program** or from one **program** to another.

what is the difference between marshalling and Unmarshalling?

In few words, "**marshalling**" refers to the process of converting the data or the objects inbto a byte-stream, and "**unmarshalling**" is the reverse process of converting the byte-stream beack to their original data or object.

What is JSON encoding?

**JSON** (JavaScript Object Notation) is a lightweight data-interchange format. **JSON** is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

### **Encoding –**

* Marshal → to encode GO values to JSON in string format
* Unmarshal → to decode JSON data to GO values
* Encoding → same as Marshal but in streams
* Decoding → same as unmarshal but from streams
* All fields should start with a capital letter so that they will be exported
* JSON objects only support strings as keys; if you want to encode map type to JSON then it must be in map[string] T
* There is no way for encoding of channels, function types into JSON
* Cyclic data structures are not supported
* Pointers are dereferenced before encoded into JSON
* json → used by json.encoding package, detailed at json.Marshal()
* xml → used by encoding/xml, detailed at xml.Marshal()
* yaml → used by gopkg.in/yaml.v2 package, at yaml.Marshal()
* bson→ used by gobson package, detailed at bson.Marshal()
* Other – orm, db, gorm, datastore, schema, asn

The Marshaling use to convert **Go object into JSON**  and Unmarshal is **JSON to Go Struct**. We can encode and decode struct data using golang marshal and unmarshal. We can encode and decode struct data using golang marshal and unmarshal.

We will convert struct type data into JSON by marshaling and JSON to string using unmarshalling process. The marshal and Unmarshal method returned response in Bytes types, but we can change these data to strings/JSON in Go.

JSON is very commonly data format to exchange information between client and server.This is a language-independent data format and can use in any programming interface API.

The golang is having 'encoding/json' package for json related operation into the go application. This package is providing many powerful and useful method. The JSON parsing and generating JSON data is easily available in many programming languages.

**Golang Marshal Struct to json**

The json package provides a json.Marshal() method.This method help to convert Struct into json data, This method takes object as a param and returned Bytes code. We will create simple Post struct, that have title, post and updated\_by property.

type Post struct {

Title string

Post string

Updated\_by string

}

Let’s convert Posts struct data into json using json.Marshal() method.The below code use to convert Struct data into Byte code using golang marshaling.

post\_obj := Posts**{**Title:"test", Post:"test desc", Updated\_by :"rozi"**}**

post, \_ := json.Marshal**(**post\_obj**)**

fmt.Println**(**string**(**post**))**

The above code will output below response –

**{**"Title":"test", "Post":"test desc", "Updated\_by" :"rozi"**}**

**Un-marshalling in GOLANG**

The json.Unmarshal() method is used to convert json into Struct Object, This method takes json byte data as a parameter and returned struct object as a response.

I am taking below json string and convert into byte data, Finally convert json byte data into Posts struct.

**{**"title":"test", "content":"test desc", "updated\_by" :"rozi"**}**

The Posts struct have **Title**, **Post** and **Updated\_by** property.The string keys in the JSON are matched to the field names in the structs.

Let’s create Response Struct tha will use to match byte code using unmarshal.

type Response struct **{**

Title string json:"title"

Post string json:"content"

Updated\_by string json:"updated\_by"

**}**

Now, We will convert below json string byte data and convert into Post struct using json.Unmarshal() method.

bytes := **[]**byte**(**str\_post**)**

var res Response

json.Unmarshal**(**bytes, &res**)**

fmt.Println**(**res.Title**)**