Routing

TDK uses the github.com/gorilla/mux package under the covers, to handle within TDK applications. With that said, TDK wraps the `mux` API with its own. This guide walks you through all you'll need to know how TDK handles routing.

Creating a new HTTP Server

TDK has integrated config which has default location in `/config/tkp-app.<env>.yaml` . As you can read here, mandatory config for http server looks like this:

```
server:
  http_address: "<port>"
```

The app will fail if TDK can't find `http_address` or the address already bound.

In order to create a new HTTP Server, first we have to implement this interface.

```
type HttpServerHandler interface {
    RegisterHandler(r *Router)
}
```

`RegisterHandler(*Router)` is called when TDK registering all the http routes. This is where you map all the path and the handler for your app.

The handler (`http.HandlerFunc`) have form like this:

```
import "github.com/tokopedia/tdk/app/http"

func (c http.TdkContext) error {
    // do some work
}
```

See the Context to understand the `http.TdkContext` interface.

Mapping the handler into HTTP Server takes the form of:

```
func (HttpServer) RegisterHandler(r *http.Router) {
    r.HandleFunc("/", func(ctx http.TdkContext) error {
        ctx.Write([]byte("Hello World!"))
        return nil
    }, "GET")

    r.HandleFunc("/some_path", handleSomePath, "POST")
}
```

As you can see, `struct HttpServer` should implement `RegisterHandler()` to be registerd by TDK app as a HTTP Server. In this example we map the `path` to a `handler` , if you follow the proper structure you should define the handler inside the Server layer of the app.

After `HttpServer` implement `HttpServerHandler` then we can register it into the TDK using `App` Object

```
myapp.RegisterHTTPServer(HttpServer{})
```

you can find it in generated `/cmd/<appname>/main.go`

Parameters

Query string and other parameters are available from the `http.TdkContext` that is passed into the `http.HandlerFunc` .

```
r.HandleFunc("/users", func (c http.TdkContext) error {
    name := c.Vars()["name"]
    ctx.Write([]byte("Hello " + name))
    return nil
})
```

Given the above code sample, if we make a request with `GET /users?name=tokopedia` , the response should should be `200: tokopedia` .

patterns in the mapped path that will get converted into parameters that can be accessed from a `http.TdkContext` .

```
r.HandleFunc("/users/{name}", func (c http.TdkContext) error {
    name := c.Vars()["name"]
    ctx.Write([]byte("Hello " + name))
    return nil
})
```

Given the above code sample, if we make a request with `GET /users/tokopedia` , the response should should be `200: tokopedia` .

← NSQ CONFIGURATION

CONTEXT →

Tokopedia Open Source