Gmail Bela Berde <bela.berde@gmail.com>

Client Adapters for Golang

Thomas Darimont <thomas.darimont@googlemail.com> Mon, Mar 5, 2018 at 10:07 AM
To: Bela Berde <bela.berde@gmail.com>
Cc: keycloak-dev <keycloak-dev@lists.jboss.org>

Hello Bela,

what framework or library are you using in your Go Web App? The thing is, there is no such thing as a "single" Keycloak integration,
due to the myriads of web frameworks available for Go.

Of course most of them support a Servlet Filter like middleware pattern (as shown below) that could be implemented somewhat generic but in the end you probably
need an integration that is custom to your framework.

I just started to adding support for Keycloak to https://github.com/stretchr/gomniauth but the library is missing a bit of functionality such as logout etc.
There is https://github.com/gambol99/keycloak-proxy where you could look for the OIDC protocol interaction create your own implementation.

Another option would be to just use an OIDC go client implementation like: https://github.com/coreos/go-oidc
or https://github.com/ericchiang/oidc

Cheers,
Thomas

```
package main

import (
	"flag"
	"fmt"
	"github.com/gorilla/mux"
	"log"
	"net/http"
	"net/http/httputil"
	"os"
	"time"
)

var greeting string

func main() {

	flag.StringVar(&greeting, "g", "Hello", "The greeting message, defaults to 'Hello'")
	flag.Parse()

	logger := *log.New(os.Stdout, "logger: ", log.Lshortfile)

	router := mux.NewRouter()
```

```
        router.Handle("/greet", http.HandlerFunc(greet))

        //for all routes apply the follwing list of handlers
        http.Handle("/", Adapt(router,
        WithLogger(&logger),
        WithRequestTracing(),
        WithHeaders(map[string]string{"foo": "42", "bar": "1337"}),
        ))

        http.ListenAndServe(":8080", nil)
}

// Adapter wraps an http.Handler with Additional functionality.
type Adapter func(http.Handler) http.Handler

// Adapt h with all specified adapters.
func Adapt(h http.Handler, adapters ...Adapter) http.Handler {

        // apply handler backwards so that they are executed in declared order
        for i := len(adapters) - 1; i >= 0; i-- {
        h = adapters[i](h)
        }

        return h
}

// curl -v http://localhost:8080/greet
func greet(w http.ResponseWriter, r *http.Request) {

        w.WriteHeader(http.StatusOK)
        fmt.Fprintf(w, greeting+" %v\n", time.Now().String())
}

func WithRequestTracing() Adapter {
        return func(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {

        dump, err := httputil.DumpRequest(r, true)
        if err != nil {
        http.Error(w, fmt.Sprint(err), http.StatusInternalServerError)
        return
        }

        fmt.Printf("Request Dump:\n%q\n", dump)

        h.ServeHTTP(w, r)
        })
        }
}

func WithHeaders(headers map[string]string) Adapter {
        return func(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {

        for key, value := range headers {
        w.Header().Add(key, value)
        }
        h.ServeHTTP(w, r)
        })
        }
}

func WithLogger(l *log.Logger) Adapter {
        return func(h http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {

        l.Println(r.Method, r.URL.Path)

        h.ServeHTTP(w, r)
        })
        }
}
```

[Quoted text hidden]

> [Quoted text hidden]
>
> ____
> keycloak-dev mailing list
> keycloak-dev@lists.jboss.org
> https://lists.jboss.org/mailman/listinfo/keycloak-dev