Package impl

```
import "go.timothygu.me/downtomeet/server/impl"
```


Overview ▾

Package impl contains the business logic of the DownToMeet server. It is separated from the restapi package as a way to better delineate between handwritten files and automatically generated ones: this package contains all handwritten code, while the majority of restapi is generated from the Swagger API definitions.

Index ▾

Package files

auth.go context.go impl.go log.go meetup.go sessionkey_string.go user.go user_facebook.go user_google.go user_oauth.go

func **RequestFromContext**

```
func RequestFromContext(ctx context.Context) *http.Request
```

RequestFromContext retrieves the request previously stored in ctx with WithRequest. If no such request exists, RequestFromContext returns nil.

func **RequestMiddleware**

```
func RequestMiddleware(h http.Handler) http.Handler
```

RequestMiddleware returns a handler that ensures that the request's context has the request attached, so that it can later be retrieved using RequestFromContext.

func **SessionFromContext**

```
func SessionFromContext(ctx context.Context) *sessions.Session
```

SessionFromContext retrieves the session previously stored in ctx with WithSession. If no such session exists, SessionFromContext returns nil.

func **WithRequest**

```
func WithRequest(ctx context.Context, r *http.Request) context.Context
```

WithRequest returns a new context with the given request attached, that can later be retrieved using RequestFromContext. The following is guaranteed to hold:

```
RequestFromContext(WithRequest(ctx, r)) == r
```

func **WithSession**

```
func WithSession(ctx context.Context, session *sessions.Session) context.Context
```

WithSession returns a new context with the session attached, that can later be retrieved using SessionFromContext. The following is guaranteed to hold:

```
SessionFromContext(WithSession(ctx, s)) == s
```

type **Implementation**

An Implementation provides all server endpoints for the app.

```
type Implementation struct {
    Options struct {
        Production bool   `long:"production" description:"Run in production
mode"`
        Database   string `long:"database" description:"URL of Postgres DB"
default:"postgresql://localhost:5432/downtomeet"`
        Frontend   string `long:"frontend" description:"Base URL of frontend"
default:"http://localhost:3000/"`
    }
    // contains filtered or unexported fields
}
```

func NewImplementation

```
func NewImplementation() *Implementation
```

NewImplementation returns a new Implementation intended for production, with a sessions.CookieStore as the internal session store.

func NewMockImplementation

```
func NewMockImplementation(store sessions.Store, randSrc rand.Source)
*Implementation
```

NewMockImplementation returns a new Implementation with the provided parameters.

func (*Implementation) DB

```
func (i *Implementation) DB() *gorm.DB
```

DB returns the database associated with the Implementation. If an error occurred when connecting to the database, DB panics.

func (*Implementation) DeleteMeetupID

```
func (i *Implementation) DeleteMeetupID(params operations.DeleteMeetupIDParams, _
interface{}) middleware.Responder
```

DeleteMeetupID implements the DELETE /meetup/:id endpoint.

func (*Implementation) GetMeetup

```
func (i *Implementation) GetMeetup(params operations.GetMeetupParams)
middleware.Responder
```

GetMeetup implements the GET /meetup endpoint.

func (*Implementation) GetMeetupID

```
func (i *Implementation) GetMeetupID(params operations.GetMeetupIDParams)
middleware.Responder
```

GetMeetupID implements the GET /meetup/:id endpoint.

func (*Implementation) GetMeetupIdAttendee

```
func (i *Implementation) GetMeetupIdAttendee(params
operations.GetMeetupIDAttendeeParams, _ interface{}) middleware.Responder
```

GetMeetupIdAttendee implements the GET /meetup/:id/attendee endpoint.

func (*Implementation) GetUserFacebookAuth

```
func (i *Implementation) GetUserFacebookAuth(param
operations.GetUserFacebookAuthParams) middleware.Responder
```

GetUserFacebookAuth implements the GET /user/facebook/auth endpoint

func (*Implementation) GetUserFacebookRedirect

```
func (i *Implementation) GetUserFacebookRedirect(param
operations.GetUserFacebookRedirectParams) middleware.Responder
```

GetUserFacebookRedirect implements the GET /user/facebook/redirect endpoint

func (*Implementation) GetUserGoogleAuth

```
func (i *Implementation) GetUserGoogleAuth(param
operations.GetUserGoogleAuthParams) middleware.Responder
```

GetUserGoogleAuth implements the GET /user/google/auth endpoint

func (*Implementation) GetUserGoogleRedirect

```
func (i *Implementation) GetUserGoogleRedirect(param
operations.GetUserGoogleRedirectParams) middleware.Responder
```

GetUserGoogleRedirect implements the GET /user/google/redirect endpoint

func (*Implementation) GetUserID

```
func (i *Implementation) GetUserID(params operations.GetUserIDParams)
middleware.Responder
```

GetUserID implements the GET /user/:id endpoint.

func (*Implementation) GetUserLogout

```
func (i *Implementation) GetUserLogout(params operations.GetUserLogoutParams)
middleware.Responder
```

GetUserLogout implements the GET /user/logout endpoint.

func (*Implementation) GetUserMe

```
func (i *Implementation) GetUserMe(params operations.GetUserMeParams, _
interface{}) middleware.Responder
```

GetUserMe implements the GET /user/me endpoint.

func (*Implementation) NewOAuthState

```
func (i *Implementation) NewOAuthState() OAuthState
```

NewOAuthState returns a fresh OAuthState. The State field is set to a randomly generated 30-character ASCII string. The returned state is set to expire after 30 minutes.

func (*Implementation) PatchMeetupID

```
func (i *Implementation) PatchMeetupID(params operations.PatchMeetupIDParams, _
interface{}) middleware.Responder
```

PatchMeetupID implements the PATCH /meetup/:id endpoint.

func (*Implementation) PatchMeetupIdAttendee

```
func (i *Implementation) PatchMeetupIdAttendee(params
operations.PatchMeetupIDAttendeeParams, _ interface{}) middleware.Responder
```

PatchMeetupIdAttendee implements the PATCH /meetup/:id/attendee endpoint.

func (*Implementation) PatchUserID

```
func (i *Implementation) PatchUserID(params operations.PatchUserIDParams, _
interface{}) middleware.Responder
```

PatchUserID implements the PATCH /user/:id endpoint.

func (*Implementation) PostMeetup

```
func (i *Implementation) PostMeetup(params operations.PostMeetupParams, _
interface{}) middleware.Responder
```

PostMeetup implements the POST /meetup endpoint.

func (*Implementation) PostMeetupIdAttendee

```
func (i *Implementation) PostMeetupIdAttendee(params
operations.PostMeetupIDAttendeeParams, _ interface{}) middleware.Responder
```

PostMeetupIdAttendee implements the POST /meetup/:id/attendee endpoint.

func (*Implementation) SessionMiddleware

```
func (i *Implementation) SessionMiddleware(handler http.Handler) http.Handler
```

SessionMiddleware augments the request's context with a session that can be fetched using SessionFromContext(r.Context()). Additionally, if the handler wrote values into the session or if there was a preexisting session, the updated cookie added to the response.

func (*Implementation) SessionStore

```
func (i *Implementation) SessionStore() sessions.Store
```

SessionStore returns the internal session store.

type OAuthState

OAuthState represents the current OAuth 2.0 login session. In an OAuth login flow, this structure would be created upon user commencing a login, with the State field used as the "state" URL parameter and the entire structure saved to the user's cookie. Later, when the user redirects back to the server, the server would validate the validity of the state by comparing the received "state" parameter against the State field.

```
type OAuthState struct {
    State     string // a nonce used as the OAuth "state" parameter
    ExpiresAt time.Time
}
```

func (OAuthState) Validate

```
func (s OAuthState) Validate(state string) bool
```

Validate returns true if s is not expired, and the given string matches s.State.

type RequestLogHook

RequestLogHook is a log.Hook implementation that adds to the log.Entry information gleaned from the entry context, such as the HTTP request (using RequestFromContext) and the user session (using SessionFromContext).

```
type RequestLogHook struct{}
```

func (RequestLogHook) Fire

```
func (r RequestLogHook) Fire(entry *log.Entry) error
```

Fire implements log.Hook.

func (RequestLogHook) Levels

```
func (r RequestLogHook) Levels() []log.Level
```

Levels returns log.AllLevels and implements log.Hook.

type SessionKey

SessionKey is the type used to key session.Values.

```
type SessionKey int
```

Pre-defined session value keys:

```
const (
    UserID        SessionKey = 0 // session.Values[UserID] is a string
    FacebookState SessionKey = 1 // session.Values[FacebookState] is an
OAuthState
    GoogleState   SessionKey = 2 // session.Values[GoogleState] is an OAuthState
)
```

func (SessionKey) String

```
func (i SessionKey) String() string
```

Subdirectories

Name
..
nonce
responders