# Config

Resave : force session to be saved back to session store

saveUninitialized : Forces a session that is "uninitialized" to be saved to the store

secret(string | array)

# APIs

Regenerate : toàn bộ session reset

req.session.regenerate(function(err) {

// will have a new session here

})

Xóa session

req.session.destroy(function(err) {

// cannot access session here

})

Reload : load from store và replace session

req.session.reload(function(err) {

// session updated

})

Save : save session to store, replace session to store

req.session.save(function(err) {

// session saved

})

## Properties

req.session.id : unique ID không thể thay đổi

req.session.cookie

* expires = false 🡪 không bị hủy
* maxAge(milisec)

var hour = 3600000

req.session.cookie.expires = new Date(Date.now() + hour)

req.session.cookie.maxAge = hour

Session Store Implementation

Có thể chọn lưu ở session ở File hoặc Database gì đó ở đây

* store.all(callback(err, sessions))
* store.destroy(sid, callback(err))
* store.clear(callback(err)) : delete all session from store
* store.length(callback(err, len)) : count all session from store
* store.get(sid, callback(err, session))
* store.set(sid, callback(err))
* store.touch(sid, session, callback(err))

Implementation :

[Firebase - Firestore](https://www.npmjs.com/package/firestore-store)

[Firebase - Realtime DB](https://www.npmjs.com/package/connect-session-firebase)

[Native MySQL](https://www.npmjs.com/package/express-mysql-session)

[Mongo DB](https://www.npmjs.com/package/connect-mongo)

Firebase Implementation

$ npm install firebase-admin connect-session-firebase –save

const express = require('express');

const session = require('express-session');

const FirebaseStore = require('connect-session-firebase')(session);

const firebase = require('firebase-admin');

const ref = firebase.initializeApp({

credential: firebase.credential.cert('path/to/serviceAccountCredentials.json'),

databaseURL: 'https://databaseName.firebaseio.com'

});

express()

.use(session({

store: new FirebaseStore({

database: ref.database()

}),

secret: 'keyboard cat'

resave: true,

saveUninitialized: true

}));

Firebase security

{

  "rules": {

    ".read": "false",

    ".write": "false",

    "sessions": {

      ".read": "false",

      ".write": "false"

    },

    "some\_public\_data": {

      ".read": "true",

      ".write": "auth !== null"

    }

  }

}

# Template

const express = require('express');

const session = require('express-session');

const app = express();

app.use(session({

resave: true,

saveUninitialized: true,

secret: 'somesecret',

cookie: { maxAge: 60000 }}));

var server = app.listen(3000, "127.0.0.1", function () {

var host = server.address().address

var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)

});

*//set session*

app.get('/set\_session', (req, res) => {

*//set a object to session*

req.session.User = {

website: 'anonystick.com',

type: 'blog javascript',

like: '4550'

}

return res.status(200).json({status: 'success'})

})

*//set session*

app.get('/get\_session', (req, res) => {

*//check session*

if(req.session.User){

return res.status(200).json({status: 'success', session: req.session.User})

}

return res.status(200).json({status: 'error', session: 'No session'})

})

*//destroy session*

app.get('/destroy\_session', (req, res) => {

*//destroy session*

req.session.destroy(function(err) {

return res.status(200).json({status: 'success', session: 'cannot access session here'})

})

})