su-si-ism-354-2018 / **bos_api**

Branch: master ▾ **bos_api** / **server_generic.js** Find file Copy path

barnsza Added Generic API e743dcb 11 days ago

1 contributor

105 lines (84 sloc) 3.45 KB

```
const restify = require('restify');
const MongoClient = require('mongodb').MongoClient;
const jsSHA = require('jssha');
const Passport = require('passport');
const PassportHTTPBearer = require('passport-http-bearer');
const uuidv3 = require('uuid/v3');
const errs = require('restify-errors');
const restify_cors = require('restify-cors-middleware')

const UUID_NAMESPACE = "353ac1c0-cab3-11e8-9211-4f01514d4e40";

(async () => {
  const db = (await MongoClient.connect('mongodb://localhost:27017',
    { useNewUrlParser: true })).db('generic');

  // Simple HMAC based bearer token strategy, not explicitly secure.
  Passport.use(new PassportHTTPBearer.Strategy(async (token, next) => {
    const [hash, username] = token.split('%');

    const user = await db.collection('users').findOne({ username: username });
    if (!user) return next(false);

    const hmac = new jsSHA('SHA-256', 'TEXT');
    hmac.setHMACKey(user['password'], 'TEXT');
    hmac.update(user['username']);
    hmac.update(Date.now().toString(36).substring(0, 4));

    if (hash === hmac.getHMAC('HEX')) return next(null, username);
    return next(false);
  }));

  const cors = restify_cors({
    origins: ['*'],
    allowHeaders: ['Authorization']
  })

  const server = restify.createServer();
  server.pre(restify.pre.sanitizePath());
  server.use(restify.plugins.bodyParser());
  server.pre(cors.preflight)
  server.use(cors.actual)

  server.post('/user', // Create New User
    async (req, res, next) => {
      if (!(req.body.username && req.body.password)) return next(
        new errs.MissingParameterError("Must supply username and password."));

      const id = uuidv3(req.body.username, UUID_NAMESPACE);
      const user = {
        _id: id,
        username: req.body.username,
        password: req.body.password
      };

      const collection = db.collection('users');
      try {
        const new_user = await collection.insertOne(user);
        res.send(new_user);
        next();
      }
      catch(e) {
        return next(new errs.PreconditionFailedError("Could not create user."));
      }

  });

  server.get('/item/:id', // Get Generic Item by _id, owned by a single user.
    Passport.authenticate('bearer', { session: false }),
    async (req, res, next) => {
      const collection = db.collection('generic');
      const item = await collection.findOne({ username: req.user, _id: req.params.id })
      if (!item) return next(new errs.NotAuthorizedError('Not Authorized.'));
      res.send(item);
      next();
  });

  server.post('/item', // New generic item
    Passport.authenticate('bearer', { session: false }),
    async (req, res, next) => {
      req.body.username = req.user;
      req.body._id = uuidv3(`${Date.now()}:${req.user}`, UUID_NAMESPACE);

      const collection = db.collection('generic');
      const item = await collection.replaceOne({ _id: req.body._id }, req.body, { upsert: true });

      res.send(item);
      next();
  });

  server.del('/item/:id', // Delete Generic Item by _id, owned by a single user.
    Passport.authenticate('bearer', { session: false }),
    async (req, res, next) => {
      const collection = db.collection('generic');
      const item = await collection.findOne({ username: req.user, _id: req.params.id })
      if (!item) return next(new errs.NotAuthorizedError('Not Authorized.'));

      await collection.deleteOne({ _id: req.params.id })

      res.send(item);
      next();
  });

  server.listen(8081, () => { console.log(`Listening: ${server.url}`) });
})();
```