UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

Tecnologias e Programação de Sistemas de Informação

Environment Variables & JWT in NODE

Desenvolvimento Web - Back-End | David Jardim

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

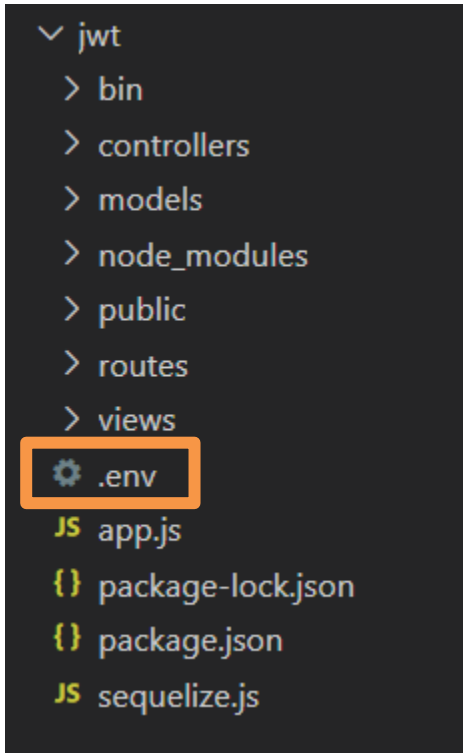
Environment Variables

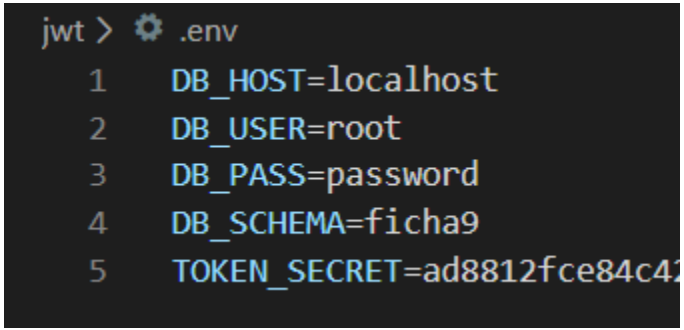
- Allow us to externalize all environment specific aspects of our app and keep the app encapsulated
  - Meaning the application can run anywhere
  - Change variables without changing the code
  - Change the variables without rebuilding the application

Environment Variables

- When should we use environment variables?
  - Define which HTTP port to listen?
  - Pointing to a development, staging, test, or production database
  - Tokens
  - What path and folder your files are located in

Environment Variables - .env file





Environment Variables - Usage

$npm install dotenv --save

```
var dotenv = require("dotenv");
// read .env file
dotenv.config();
```

This configuration should be done as soon as possible!

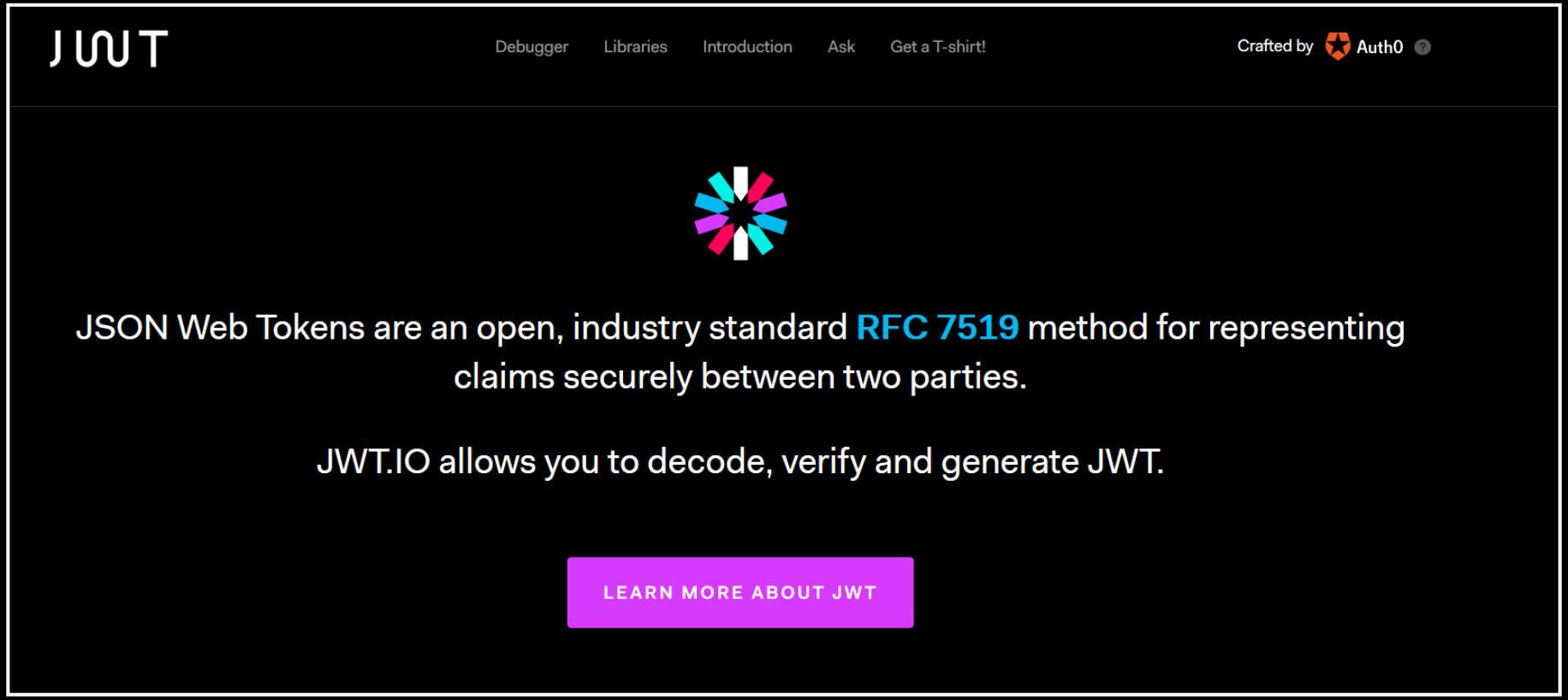
```
// Connection pool
const sequelize = new Sequelize(process.env.DB_SCHEMA,
  host: process.env.DB_HOST,
  dialect: 'mysql',
  pool: {
    max: 10,
    min: 0,
    acquire: 30000,
    idle: 10000
  }
});
```

Environment Variables – Best Practices

- Never upload the .env file to your code repository

- We could create a module to easily gather all the variables facilitating refactoring and maintenance

```
// config.js
const dotenv = require('dotenv');
dotenv.config();
module.exports = {
  dbhost: process.env.DB_HOST,
  dbpass: process.env.DB_PASS,
  token: process.env.TOKEN_SECRET
};
```


JWT
Debugger
Libraries
Introduction
Ask
Get a T-shirt!
Crafted by Auth0
JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.
JWT.IO allows you to decode, verify and generate JWT.
LEARN MORE ABOUT JWT

Install Json Web Token Dependencies

```
"dependencies": {
  "connect-flash": "^0.1.1",
  "cookie-parser": "~1.4.4",
  "crypto": "^1.0.1",
  "debug": "~2.6.9",
  "dotenv": "^8.2.0",
  "ejs": "~2.6.1",
  "express": "~4.16.1",
  "express-session": "^1.17.1",
  "http-errors": "~1.6.3",
  "jsonwebtoken": "^8.5.1",
  "morgan": "~1.9.1",
  "mysql2": "^2.1.0",
  "sequelize": "^5.21.9"
```

JSON Web Token

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

How does it works?

- Generate a JWT token in the backend that is specific to a user
- Pass this JWT token to the frontend/client
- The frontend sends this token alongside requests to access protected API routes

Generating a token

In order to generate a token we need 3 pieces of information:

- The token secret
- The data to hash in the token
- The token expire time

The token secret

```
var crypto = require('crypto');
var tokenSecret = crypto.randomBytes(64).toString('hex');
console.log(tokenSecret);
```

Save this token secret on the .env file

```
TOKEN_SECRET=ad8812fce84c42ed347435730870e43ad76130b9d673abe823c9f064
```

The data to hash in the token

The **data** to you hash in your token can be a user ID or a username, or a more complex object. In either case, it should be an *identifier* for a *specific* user.

```
function generateAccessToken(email, password) {
    // expires after half and hour (1800 seconds = 30 minutes)
    return jwt.sign({ email, password }, process.env.TOKEN_SECRET, { expiresIn: '1800s' });
}
```

data to hash

token secret

time to expire

Send the token to the frontend/client

```
app.post('/signup', (req, res) => {
    // ...
    const token = generateAccessToken(email, password);
    res.json(token);
    // ...
});
```

The token can be sent as json or added to the session

```
const token = generateAccessToken(email, password);
req.session.user = result;
req.session.token = token;
res.redirect('/profile');
```

JWT as a middleware function

When a request is made to this route, the middleware function will verify if the token is valid

```
router.get('/profile', authenticateTokenFromSession, function (req, res) {
  res.render('profile.ejs', {
    user: req.session.user // get the user out of session and pass to template
  });
});
```

Validate the JWT access token from the request header

```
function authenticateTokenFromHeaders(req, res, next) {
  // Gather the jwt access token from the request header
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (token == null) return res.sendStatus(401);
    jwt.verify(token, process.env.TOKEN_SECRET, (err, user) => {
    if (err)
      return res.sendStatus(403);
    req.user = user;
    next();
  });
}
```

Validate the JWT access token from session

```
function authenticateTokenFromSession(req, res, next) {
  const token = req.session.token;
  if (token == null) return res.sendStatus(401);
  jwt.verify(token, process.env.TOKEN_SECRET, (err, user) => {
    if (err)
      return res.sendStatus(403);
    req.user = user;
    next();
  });
}
```


UNIVERSIDADE da MADEIRA
CTeSP
CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS