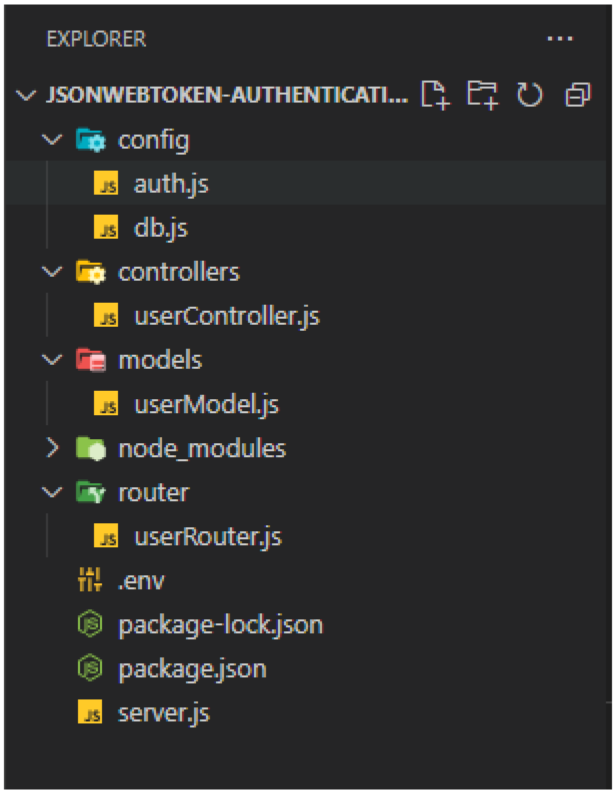
Building an Authentication API with JWT Token in Node.js Part 2

Welcome back guys! I hope you guys are also excited for this amazing tutorial of authentication using node js and json web token. If you don't know that is JSON web token, and how it works, please go back and check out first part of it. So, Let's get started. Before that, let's have a quick review what we did in previous part, we have created directories for routes, models and controllers and we setup our server. Now we are going to implement our logic. Directory structure till now.



```
EXPLORER
JSONWEBTOKEN-AUTHENTICATI...
  config
    auth.js
    db.js
  controllers
    userController.js
  models
    userModel.js
  node_modules
  router
    userRouter.js
  .env
  package-lock.json
  package.json
  server.js
```

1. First, go to ***userRouter.js*** file inside router directory and put following code in it.

```
const express = require("express");
const router = express.Router();
const userController = require("../controllers/userController");

router.post("/register", userController.register);
router.post("/login", userController.login);
module.exports = router;
```

2. Then go to ***userModel.js*** file inside models directory and put following code in it.

```
const mongoose = require("mongoose");
const userSchema = mongoose.Schema({
  userName: { type: String, required: true, trim: true },
  userEmail: { type: String, required: true, unique: true, trim: true },
  userPassword: { type: String, required: true },
  token: { type: String, default: "" },
});
module.exports = new mongoose.model("User", userSchema);
```

3. Go to ***userController.js*** inside controller's directory and put following code inside it.

```
const userModel = require("../models/userModel");
const jsonwebtoken = require("jsonwebtoken");
//function for registering user
const register = async (req, res) => {
  const { userName, userEmail, userPassword } = req.body;
  if (!userName && !userEmail && !userPassword) {
    res.status(400).send("Please provide all details");
  } else {
    //checking if account already exists
    const isExists = await userModel.findOne({ userEmail: userEmail });
    if (isExists) {
      res.status(400).send(`User with email ${userEmail} already exists...`);
    } else {
      //creating user now
      const newUser = await userModel.create({
        userName,
        userEmail,
        userPassword,
      });

      //now creating json web token
      const token = jsonwebtoken.sign(
        { userId: newUser._id, userEmail },
        process.env.JSON_WEB_TOKEN_SECRET,
        {
          expiresIn: "2h",
        }
      );

      //setting it to newUser instance
      const updated = await userModel.findByIdAndUpdate(
        newUser._id,
        { token: token },
        { new: true }
      );

      res.status(200).send(updated);
    }
  }
};

//this function will check user credentials and authenticate user
const login = async (req, res) => {
  const { userEmail, userPassword } = req.body;
  if (!userEmail && !userPassword) {
    res.status(400).send("Please enter email and password");
  } else {
    //lets now first check whether user exists or now
    const user = await userModel.findOne({ userEmail: userEmail });
    if (user && user.userPassword === userPassword) {
      //user logged in

      //now creating json web token
      const token = jsonwebtoken.sign(
        { userId: user._id, userEmail },
        process.env.JSON_WEB_TOKEN_SECRET,
        {
          expiresIn: "2h",
        }
      );

      res.status(200).json(user);
    } else {
      res.status(400).send("Invalid Credentials");
    }
  }
};

//exporting functions
module.exports = {
  register,
  login,
};
```

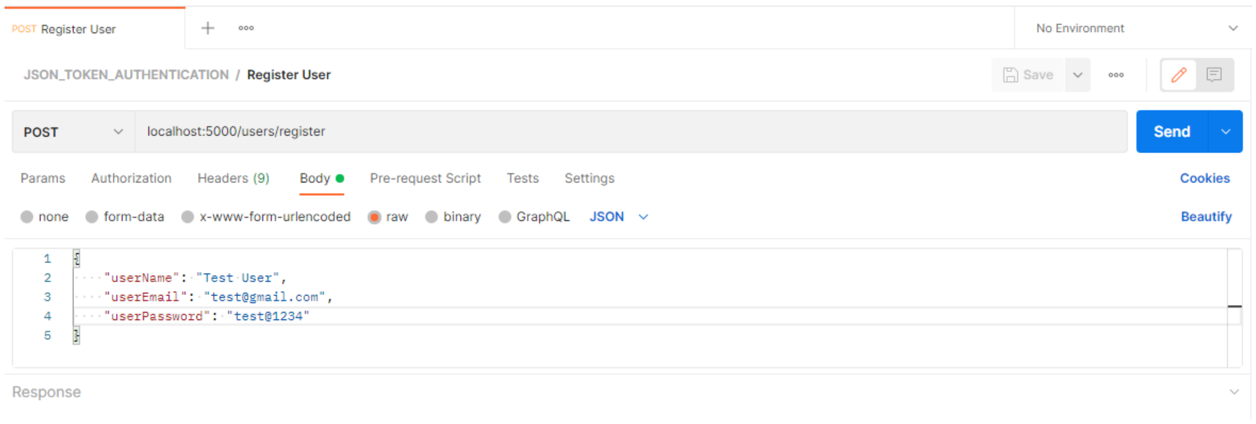
4. Update server.js file by

```
const express = require("express");
const cors = require("cors");
const dotEnv = require("dotEnv");
const app = express();
const getConnection = require("./config/db");
const userRouter = require("./router/userRouter");

//registering middlewares
dotEnv.config();
app.use(express.json());
app.use(cors());
app.use("/users", userRouter);
//connecting to db
getConnection();

//listening to server
const port = process.env.PORT || 5000;
app.listen(port, () => {
  console.log(`server is running on port ${port}`);
});
```

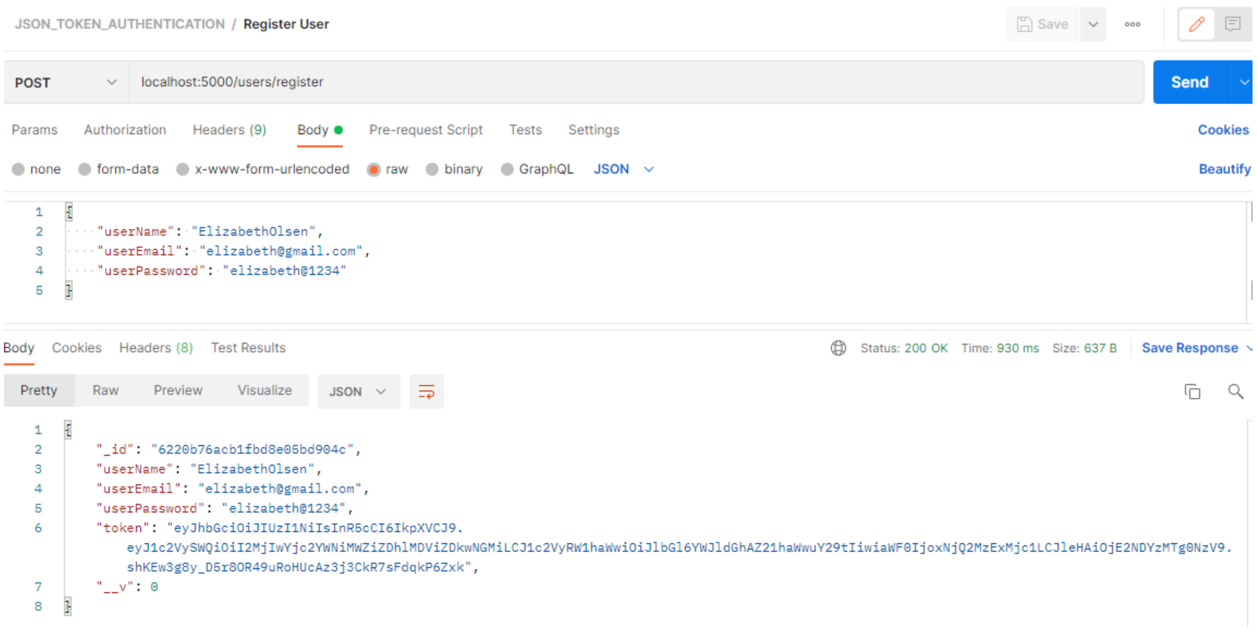
5. Now we will run server by command ***npm start***
6. Now open postman.
7. And create post request for registering user



POST localhost:5000/users/register

```
{
    "userName": "Test User",
    "userEmail": "test@gmail.com",
    "userPassword": "test@1234"
}
```

8. Click on send, and see the response



JSON_TOKEN_AUTHENTICATION / Register User

POST localhost:5000/users/register

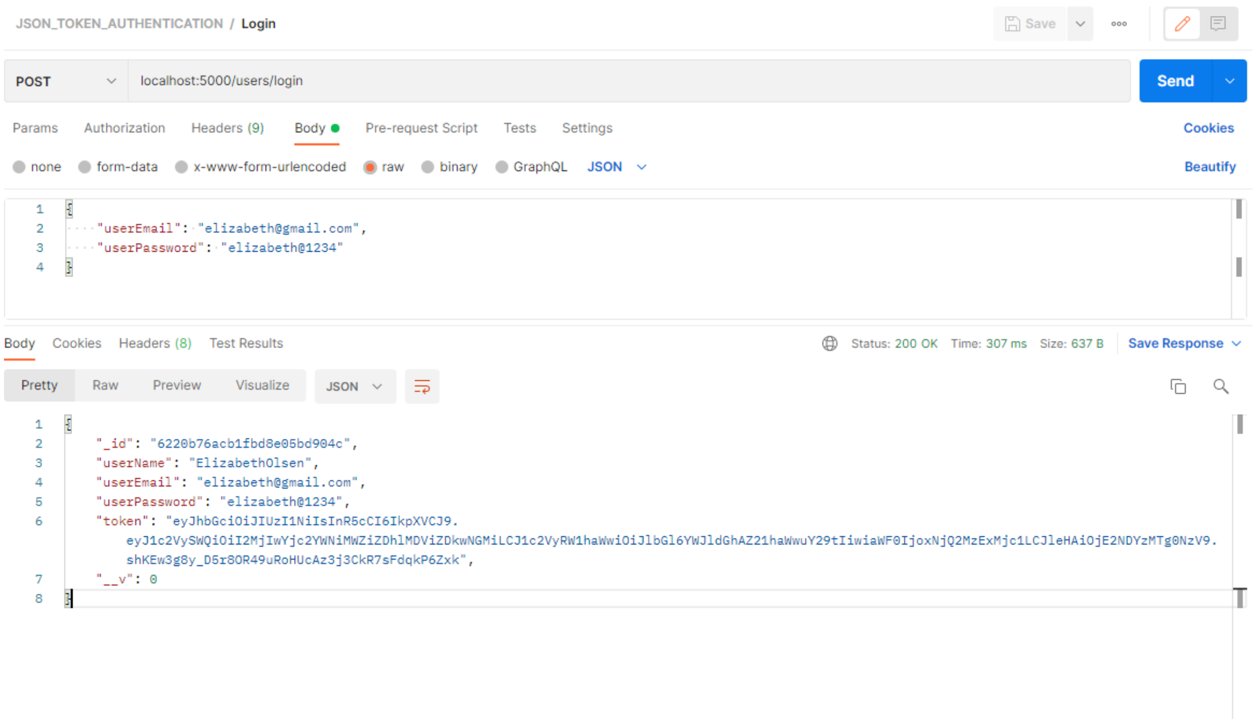
Body: raw JSON

```
{
    "userName": "ElizabethOlsen",
    "userEmail": "elizabeth@gmail.com",
    "userPassword": "elizabeth@1234"
}
```

Status: 200 OK Time: 930 ms Size: 637 B

```
{
    "_id": "6220b76acb1fbd8e05bd904c",
    "userName": "ElizabethOlsen",
    "userEmail": "elizabeth@gmail.com",
    "userPassword": "elizabeth@1234",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2MjIwYjc2YWNiMWZiZDhlMDViZDkwNGMiLCJ1c2VyRW1haWwiOiJlbGl6YWJldGhAZ21haWwuY29tIiwiaWF0IjoxNjQ2MzExMjc1LCJleHAiOjE2NDYzMTg0NzV9.
        shKEw3g8y_D5r8OR49uRoHUcAz3j3CkR7sFdqkP6Zxk",
    "__v": 0
}
```

9. Now let’s check login request, again create login post



JSON_TOKEN_AUTHENTICATION / Login

POST localhost:5000/users/login

Body: raw JSON

```
{
    "userEmail": "elizabeth@gmail.com",
    "userPassword": "elizabeth@1234"
}
```

Status: 200 OK Time: 307 ms Size: 637 B

```
{
    "_id": "6220b76acb1fbd8e05bd904c",
    "userName": "ElizabethOlsen",
    "userEmail": "elizabeth@gmail.com",
    "userPassword": "elizabeth@1234",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2MjIwYjc2YWNiMWZiZDhlMDViZDkwNGMiLCJ1c2VyRW1haWwiOiJlbGl6YWJldGhAZ21haWwuY29tIiwiaWF0IjoxNjQ2MzExMjc1LCJleHAiOjE2NDYzMTg0NzV9.
        shKEw3g8y_D5r8OR49uRoHUcAz3j3CkR7sFdqkP6Zxk",
    "__v": 0
}
```

Great, our users api for registering and logging in working fine. Now let’s write middle ware for authenticating user via token.

10. Now put following code inside auth.js which is under config directory

```
const jwt = require("jsonwebtoken");
const userModel = require("../models/userModel");
```

```
//we will pass token as body, or in request
//then using this token we will authenticate user, if authenticated we will
add user to this request
const authentication = async (req, res, next) => {
  const token = req.body.token || req.query.token;

  if (!token) {
    return res.status(403).send("A token is required for authentication");
  }
  try {
    const decoded = jwt.verify(token, process.env.JSON_WEB_TOKEN_SECRET);
    const _user = await userModel.findOne({ email: decoded.userEmail });
    req.user = _user;
  } catch (err) {
    return res.status(401).send(err.message);
  }
  return next();
};

module.exports = authentication;
```

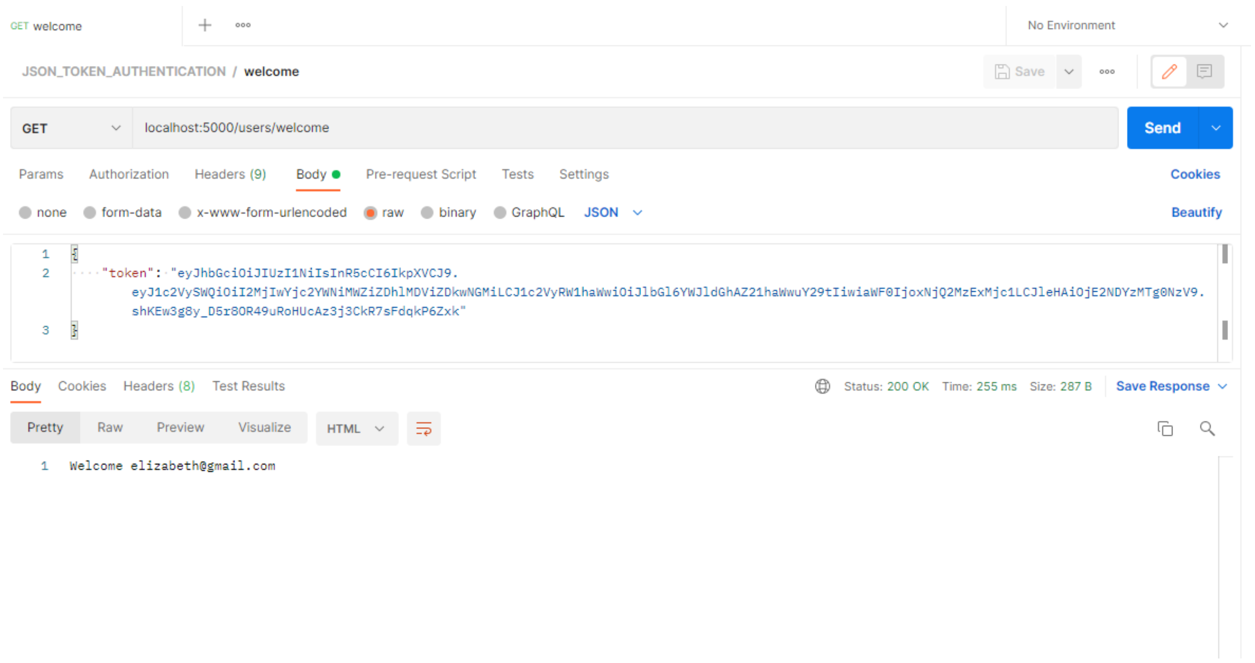
Now Let's test authentication middleware, for this let's add following snippet in ***userController.js***

```
//lets create controller for welcoming user
const welcome = async (req, res) => {
  try {
    res.status(200).send(`Welcome ${user.userEmail}`);
  } catch (error) {
    res.status(404).json(error);
  }
};
```

And don't forget to add this function ***welcome*** in module.exports. Now add following snippet in ***userRouter.js*** file.

```
//lets create route, which will only be accessible if user is authenticated
router.get("/welcome", authentication, userController.welcome);
```

Let’s test it on postman.



GET welcome

No Environment

JSON_TOKEN_AUTHENTICATION / welcome

GET localhost:5000/users/welcome

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VySWQiOiI2MjIwYjc2YWNiMWZiZDhlMDViZDkwNGMiLCJ1c2VyRW1haWwiOiJlbGl6YWJldGhAZ21haWwuY29tIiwiaWF0IjoxNjQ2MzExMjc1LCJleHAiOjE2NDYzMTg0NzV9.
        shKEw3g8y_D5r8OR49uRoHUcAz3j3CkR7sFdqkP6Zxk"
}
```

Status: 200 OK Time: 255 ms Size: 287 B

```
Welcome elizabeth@gmail.com
```

Congratulations 😊, we implemented authentication using node js and json web token. I really hope you enjoyed this tutorial. You can read more about json web token on https://jwt.io/.