ExpressJS

Express Overview

- Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.
- Following are some of the core features of Express framework:
  - Allows to set up middlewares to respond to HTTP Requests.
  - Defines a routing table which is used to perform different actions based on HTTP Method and URL.
  - Allows to dynamically render HTML Pages based on passing arguments to templates.

Installing Express

- Install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

```
npm install express --save
```

Installing Express

- You should install the following important modules along with express:
  - **body-parser**: This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
  - **cookie-parser**: Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
  - **multer:** This is a node.js middleware for handling multipart/form-data.

```
npm install body-parser –save
npm install cookie-parser –save
npm install multer --save
```

RESTful API

- Web services based on REST Architecture are known as RESTful web services. These web services uses HTTP methods to implement the concept of REST architecture.
- **HTTP methods**

  Following four HTTP methods are commonly used in REST based architecture.

  GET - This is used to provide a read only access to a resource.
  PUT - This is used to create a new resource.
  DELETE - This is used to remove a resource.
  POST - This is used to update a existing resource or create a new resource.

RESTful API

- Example:
  POST - ‘/’: Login
  GET - ‘/user/:id’: Get user information by id
  POST - ‘/user’: Create new an user
  PUT - ‘/user/:id’: Update user information
  DELETE - ‘/user/:id’: Delete user by id
  GET - ‘/user’: Get all list users

Building a Simple CRUD Application

- we have a JSON based database of users having the following users in a file **users.json**:

```
{
   "user1" : {
      "name" : "mahesh",
          "password" : "password1",
          "profession" : "teacher",
          "id": 1
   },
   "user2" : {
      "name" : "suresh",
          "password" : "password2",
          "profession" : "librarian",
          "id": 2
   },
   "user3" : {
      "name" : "ramesh",
          "password" : "password3",
          "profession" : "clerk",
          "id": 3
   }
}
```

Building a Simple CRUD Application

- Based on this information we are going to provide following RESTful APIs.

S. N.	URI	HTTP Method	POST body	Result
1	listUsers	GET	empty	Show list of all the users
2	addUser	POST	JSON string	Add details of new user
3	deleteUser	DELETE	JSON string	Delete an existing user
4	:id	GET	empty	Show details of a user

List Users

- Let's implement our first RESTful API **listUsers** using the following code in a server.js file:

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/listUsers', function (req, res) {
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       console.log( data );
       res.end( data );
   });
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

List Users

- Now try to access defined API using URL: *http://127.0.0.1:8081/listUsers* and *HTTP Method : GET* on local machine using any REST client. This should produce following result:

```
{
   "user1" : {
      "name" : "mahesh",
      "password" : "password1",
      "profession" : "teacher",
      "id": 1
   },
   "user2" : {
      "name" : "suresh",
      "password" : "password2",
      "profession" : "librarian",
      "id": 2
   },
   "user3" : {
      "name" : "ramesh",
      "password" : "password3",
      "profession" : "clerk",
      "id": 3
   }
}
```

Add User

- Following API will show you how to add new user in the list. Following is the detail of the new user:

```
user = {
   "user4" : {
      "name" : "mohit",
      "password" : "password4",
      "profession" : "teacher",
      "id": 4
   }
}
```

Add User

- Following is the addUser API to a new user in the database:

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

var user = {
   "user4" : {
      "name" : "mohit",
      "password" : "password4",
      "profession" : "teacher",
      "id": 4
   }
}
…
```

Add User

- Following is the addUser API to a new user in the database:

server.js

```
...
app.post('/addUser', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       data["user4"] = user["user4"];
       console.log( data );
       res.end( JSON.stringify(data));
   });
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

Add User

- Now try to access defined API using URL: http://127.0.0.1:8081/addUser and HTTP Method : POST on local machine using any REST client. This should produce following result:

```
{
"user1":{"name":"mahesh","password":"password1","profession":"teacher","id":1},
"user2":{"name":"suresh","password":"password2","profession":"librarian","id":2},
"user3":{"name":"ramesh","password":"password3","profession":"clerk","id":3},
"user4":{"name":"mohit","password":"password4","profession":"teacher","id":4}
}
```

Show Detail

- Now we will implement an API which will be called using user ID and it will display the detail of the corresponding user.

server.js

```
...
var server = app.listen(8081, function () {

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

})
```

Show Detail

- Now we will implement an API which will be called using user ID and it will display the detail of the corresponding user.

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/:id', function (req, res) {
  // First read existing users.
  fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
     users = JSON.parse( data );
     var user = users["user" + req.params.id]
     console.log( user );
     res.end( JSON.stringify(user));
  });
})
...
```

Show Detail

- Now we will implement an API which will be called using user ID and it will display the detail of the corresponding user.

server.js

```
...
var server = app.listen(8081, function () {

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

})
```

Show Detail

- Now try to access defined API using URL: *http://127.0.0.1:8081/2* and *HTTP Method : GET* on local machine using any REST client. This should produce following result:

```
{"name":"suresh","password":"password2","profession":"librarian","id":2}
```

Delete User

- This API is very similar to addUser API where we receive input data through req.body and then based on user ID we delete that user from the database. To keep our program simple we assume we are going to delete user with ID 2.

server.js

```
var express = require('express');
var app = express();
var fs = require("fs");

var id = 2;

app.delete('/deleteUser', function (req, res) {
   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       delete data["user" + 2];

       console.log( data );
       res.end( JSON.stringify(data));
   });
})
…
```

Delete User

- This API is very similar to addUser API where we receive input data through req.body and then based on user ID we delete that user from the database. To keep our program simple we assume we are going to delete user with ID 2.

server.js

```
...
var server = app.listen(8081, function () {

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

})
```

Delete User

- Now try to access defined API using URL: *http://127.0.0.1:8081/deleteUser* and *HTTP Method : DELETE* on local machine using any REST client. This should produce following result:

```
{"user1":{"name":"mahesh","password":"password1","profession":"teacher","id":1},
"user3":{"name":"ramesh","password":"password3","profession":"clerk","id":3}}
```

Exercises

Define all APIs and build a system based on RESTful APIs with some of the core features:

- **Register**:
  User registers a new account with email, password, password-confirm … Password must be encrypted. Server can store user’s account to file or database.
- **Login**:
  User logs in by email and password. If user logs in successfully, a token will be generated and returned to the user.
- **Forgot password**:
  User clicks on “forgot password” link and input email from front-end. Server generates a token and sets that token to expire after 3 days. Server sends a confirmation email to user. That email consists of a link which has the token.
- **Update password**:
  User clicks on the link in the email, he/she will be redirected to “update password” page. User inputs a new password. System checks availability of the token. If the token is valid, the new password will be updated for the user.

<u>Note</u>:

- It’s not necessary to code front-end for this exercises.
- Use Postman to test all the APIs.
- Submit your source code and Postman collection in order to be graded.

Thank you