express-maria 연동 구축

1) express 설치 및 웹서버 구동 테스트

1. node init

```
$ node init
```

2. express 설치

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

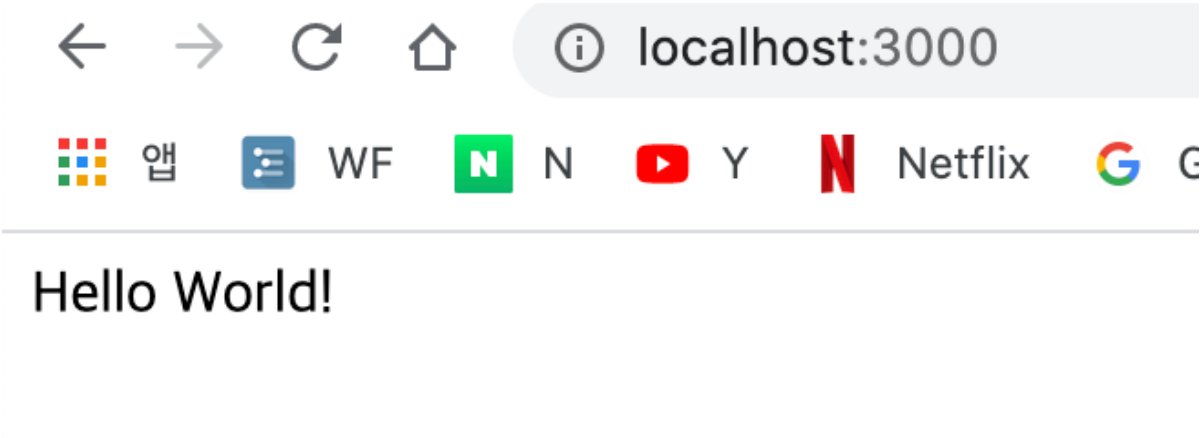
3. app.js 작성
   - 서버 열기, 라우팅 등의 역할을 한다.

   [/app.js]

```
const express = require('express')
const app = express()
const port = 3000 // 포트 지정

// GET /
/// 인덱스 페이지
app.get('/', (req, res) => {
  res.send('Hello World!')
})

// 서버 열기 (포트: 3000)
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```

4. 구동 테스트

```
$ node ./app.js
```



2) maria db 연동

1. 로컬에 maria db 설치, DB 생성, 테이블 생성 등 DB 구축 작업
2. maria db 커넥터 설치

```
$ npm install --save mariadb
```

3. DBconfig.json 작성
   - 데이터베이스 접속 정보를 소스코드에 직접 하드코딩 한다면, github 등에 디비 비번 등이 같이 올라가서 보안상 매우 좋지 못하다.
   - 따라서 따로 설정파일로 빼내고, 이 설정 파일은 github에 올리지 아니하는 것이 보안상 좋다.

   [DBconfig.json]

```
// 자신의 접속정보에 맞게 수정할 것
{
    "host": "127.0.0.1",
    "port": "3306",
    "user": "root",
    "password": "mariadb",
    "database": "nodejs_test"
}
```

4. mariaDBConn.js 작성
   - 마리아DB에 연결하는 역할을 한다.

   [mariaDBConn.js]

```
const mariadb = require('mariadb');
const vals = require('./DBconfig.json');

// DB에 접속하여 ConnectionPool 생성
const pool = mariadb.createPool({
```

```
    host: vals.host,
    port:vals.port,
    user: vals.user,
    password: vals.password,
    connectionLimit: 5
});

// DB에서 쿼리문을 통해 데이터를 가져옴
// 자신의 DB에 맞게 수정할 것
async function GetUserList(){
    let conn, rows;
    try{
        conn = await pool.getConnection();
        conn.query('USE ' + vals.database);
        rows = await conn.query('SELECT * FROM users');
    }
    catch(err){
        throw err;
    }
    finally{
        if (conn) conn.end();
        return rows;
    }
}

// 다른 js 파일에서 사용할 수 있도록 export
module.exports = {
    getUserList: GetUserList
}
```

5. app.js 수정
    - DB 접근하는 부분 추가

[app.js]

```
const mdbConn = require('./mariaDBConn.js') // DB 연결부분
const express = require('express')
const app = express()
const port = 3000 // 포트 지정

// GET /
/// 인덱스 페이지
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

```
// GET /users
/// 유저 정보들을 가져오는 페이지
app.get('/users', (req, res) => {
  mdbConn.getUserList() // DB에서 값 가져오기
    .then((rows) => { // 성공하면 실행
      res.send(rows); // 가져온 값 보여주기
    })
    .catch((err) => { // 실패하면 실행
      res.send(err); // 에러메시지 보여주기
    })
});

// 서버 열기 (포트: 3000)
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
});
```

6. 확인해보기

```
$ node ./app.js
```

3) 특정 사용자 정보만 가져오기

특정 user_id 를 갖고 있는 사용자를 검색하는 API를 만들어보자.

GET 방식이며, 파라미터로 검색할 user_id가 필요하다.

=> `GET /user/:user_id`

1. app.js 수정
    - `GET /user/:user_id` 에 대한 부분 추가

[app.js]

```
const mdbConn = require('./mariaDBConn.js') // DB 연결부분
const express = require('express')
const app = express()
const port = 3000 // 포트 지정
```

```
// GET /
/// 인덱스 페이지
app.get('/', (req, res) => {
  res.send('Hello World!')
})

// GET /users
/// 유저 정보들을 가져오는 페이지
app.get('/users', (req, res) => {
  mdbConn.getUserList() // DB에서 값 가져오기
    .then((rows) => { // 성공하면 실행
      res.send(rows); // 가져온 값 보여주기
    })
    .catch((err) => { // 실패하면 실행
      res.send(err); // 에러메시지 보여주기
    })
});

// GET /user/:user_id
/// 특정 유저 정보를 가져오는 페이지
app.get('/user/:user_id', (req, res) => {
  const user_id = req.params.user_id; // 요청 파라미터의 user_id 가져옴
  mdbConn.getUserById(user_id)
    .then((user) => {
      res.send(user);
    })
    .catch((err) => {
      res.send(err);
    })
});

// 서버 열기 (포트: 3000)
app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
});
```

2. mariaDBConn.js 수정
   - DB에서 유저 아이디 검색하는 쿼리 추가
   - export에 만든 함수 추가

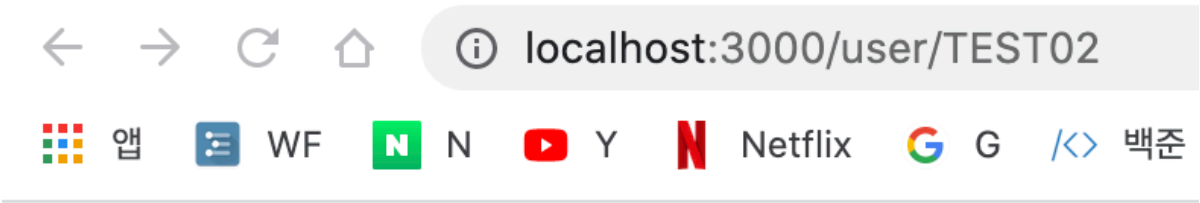
[mariaDBConn.js]

```
const mariadb = require('mariadb');
const vals = require('./DBconfig.json');

// DB에 접속하여 ConnectionPool 생성
const pool = mariadb.createPool({
    host: vals.host,
    port:vals.port,
```

```
    user: vals.user,
    password: vals.password,
    connectionLimit: 5
});

// DB에서 쿼리문을 통해 데이터를 가져옴
// 자신의 DB에 맞게 수정할 것
async function GetUserList(){
    let conn, rows;
    try{
        conn = await pool.getConnection();
        conn.query('USE ' + vals.database);
        rows = await conn.query('SELECT * FROM users');
    }
    catch(err){
        throw err;
    }
    finally{
        if (conn) conn.end();
        return rows;
    }
}

// 특정 User 검색
// 자신의 DB에 맞게 수정할 것
async function GetUserById(user_id){
    let conn, rows;
    try{
        conn = await pool.getConnection();
        conn.query('USE ' + vals.database);
        rows = await conn.query('SELECT * FROM users WHERE user_id = "' +
user_id + '"');
    }
    catch(err){
        throw err;
    }
    finally{
        if (conn) conn.end();
        return rows[0];
    }
}

// 다른 js 파일에서 사용할 수 있도록 export
module.exports = {
    getUserList: GetUserList,
    getUserById: GetUserById
}
```

3. 확인해보기

```
$ node ./app.js
```




```
{
    "user_key": 2,
    "user_id": "TEST02"
}
```