

Mục lục

I – Server	2
1. Tạo một server mới	2
2. Server trả về dữ liệu của một file	3
3. Nhận dữ liệu post lên và phản hồi lại	3
4. Tạo server theo cách khác	4
5. Kết nối DB sqlite3	5
a. Tạo DB trong sqlite3	5
b. Tạo table trong DB sử dụng thư viện viết sẵn	5
c. Insert dữ liệu vào DB	6
d. Update dữ liệu vào DB	7
e. Select dữ liệu từ DB	7
f. Delete dữ liệu trong DB	8
g. Server insert thông tin vào DB	9
h. Mở rộng	10
II - Đưa ứng dụng lên heroku	12

Download và cài đặt nodeJS, git-scm

Cài git: Vào đường dẫn <https://git-scm.com/download/win> để download và tiến hành cài đặt

Cài node: Vào đường dẫn <https://nodejs.org/en/> để download và tiến hành cài đặt

Cài VS code: Vào đường dẫn <https://code.visualstudio.com/docs/?dv=win> để download và tiến hành cài đặt

I – Server

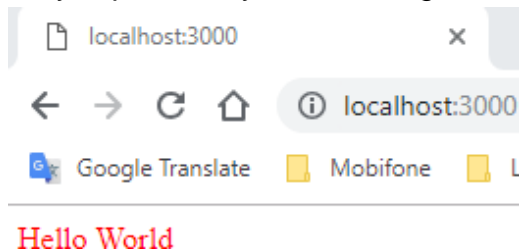
1. Tạo một server mới

- chạy lệnh **npm init** để tạo file **package.json**
- Download thư viện express về máy với lệnh **npm i express**: xuất hiện thư mục **node_modules** và file **package-lock.json**
- Tạo file **server.js** để khai báo một server đơn giản

```
var express = require("express");
var app = express();
var server = require("http").createServer(app);
server.listen(3000, () => console.log("Server is running"));

app.get("/", (req, res) => {
  res.send("<font color=red>Hello World</font>");
});
```

- Chạy lệnh **node server.js** để khởi động server
- Truy cập trình duyệt với đường dẫn <http://localhost:3000/> sẽ được như sau:



2. Server trả về dữ liệu của một file

- Khai báo thêm biến `var fs = require("fs");` để cho phép đọc file
- Đoạn lệnh sau trả về dữ liệu của 1 file

```
app.get('/*', (req, res) => {
  fs.readFile(__dirname + req.url, { encoding: 'utf-8', flag: 'r' },
    (error, data) => {
      if (!error) {
        res.writeHead(200, { 'Access-Control-Allow-Origin': '*' });
        res.end(data);
      } else {
        res.writeHead(404, { 'Access-Control-Allow-Origin': '*' });
        res.end(JSON.stringify(error));
      }
    }
  );
});
```

3. Nhận dữ liệu post lên và phản hồi lại

- Cài thư viện **body-parser** để nhận các thông tin gửi lên
- Thêm đoạn lệnh để cho phép post lên server từ một địa chỉ khác

```
app.use((req, res, next) => {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', 'Content-Type,X-Requested-With');
  next();
});
```

- Đoạn code sau nhận thông tin từ client gửi lên và phản hồi lại

```
app.post('/signin', bodyParser, (req, res) => {
  if (req.body.username == "dinh" && req.body.password == "1234")
    res.send({ status: "ok", message: "login thanh cong!" });
  else
    res.send({ status: "nok", message: "login that bai!" });
});
```

4. Tạo server theo cách khác

- Khai báo một server theo cách sau:

```
const express = require('express');
const app = express();

var main = (isHttp) => {
  const resource = require('./routes/resource');
  app.use('/excel', resource);

  if (isHttp) {
    const httpServer = require('http').createServer(app);
    const portHttp = process.env.PORT || isHttp;
    httpServer.listen(portHttp, () => console.log("Server is running!"));
  }
}
const isHttp = 8080;
main(isHttp);
```

- Khai báo file./routes/resource

```
const router = require('express').Router();
const handlers = require('./handlers/handler');

router.get('/users', handlers.getUsers);
router.post('/users', handlers.postUsers);

module.exports = router;
```

- Khai báo filehandlers/handler

```
const getUsers = (req, res) => {
  res.send({ status: "ok1", message: "login thanh cong!" });
}

const postUsers = (req, res) => {
  res.send({ status: "ok2", message: "login thanh cong!" });
}

module.exports = {
  getUsers: getUsers,
  postUsers: postUsers
};
```

5. Kết nối DB sqlite3

a. Tạo DB trong sqlite3

- Import thư viện sqlite3:

```
const sqlite3 = require('sqlite3').verbose();
```

- Đoạn lệnh để tạo DB

```
const dirDB = 'db';
const dbFile = './' + dirDB + '/mydb.db';

var db = new sqlite3.Database(dbFile, (err) => {
  if (err) {
    console.error('Could not connect to database', err);
  } else {
    console.log('Connected to database ' + dbFile);
  }
})
```

b. Tạo table trong DB sử dụng thư viện viết sẵn

- Import thư viện vào:

```
const SQLiteDAO = require('./db/sqlite3/sqlite-dao');
```

- Tạo db bằng thư viện viết sẵn:

```
var db = new SQLiteDAO(dbFile);
```

- Khai báo thông tin bảng cần tạo

```
var table = {
  name: 'sinhvien',
  cols: [
    {
      name: 'id',
      type: "INTEGER",
      option_key: 'PRIMARY KEY NOT NULL',
      description: 'day la field duy nhat'
    },
    {
      name: 'name',
      type: "TEXT",
      option_key: 'NOT NULL',
      description: 'Mo ta truong name'
    }
  ]
}
```

- Lệnh tạo bảng (sử dụng thư viện viết sẵn)

```
db.createTable(table)
  .then(data => {
    console.log("Tao bang thanh cong!", data);
  })
  .catch(err => {
    console.log("Loi", err);
  });
```

c. Insert dữ liệu vào DB

- Khai báo thông tin cần insert

```
var insertTable = {  
  name: 'sinhvien',  
  cols: [  
    {  
      name: 'name',  
      value: "nguyen van dinh"  
    }  
  ]  
};
```

- Lệnh insert dữ liệu vào bảng (sử dụng thư viện viết sẵn)

```
db.insert(insertTable)  
  .then(data => {  
    console.log("Da them thanh cong!", data);  
  })  
  .catch(err => {  
    console.log("Loi", err);  
  });
```

d. Update dữ liệu vào DB

- Khai báo thông tin cần update

```
var updateTable = {  
  name: 'sinhvien',  
  cols: [{  
    name: 'name',  
    value: 'Nguyen Van Dinh'  
  }],  
  wheres: [{  
    name: 'id',  
    value: 1  
  }]  
}
```

- Lệnh update dữ liệu vào bảng (sử dụng thư viện viết sẵn)

```
db.update(updateTable)  
  .then(data => {  
    console.log("Đã update thành công!", data);  
  })  
  .catch(err => {  
    console.log("Lỗi", err);  
  });
```

e. Select dữ liệu từ DB

- Sử dụng thư viện để kết nối DB:

```
const SQLiteDAO = require('./db/sqlite3/sqlite-dao');  
const dirDB = 'db';  
const dbFile = './' + dirDB + '/users.db';  
var db = new SQLiteDAO(dbFile);
```

- Select dữ liệu từ DB:

```
db.getRsts("SELECT * FROM sinhvien")  
  .then(data => {  
    console.log(data);  
  })  
  .catch(err => {  
    console.log("Lỗi", err);  
  });
```

f. Delete dữ liệu trong DB

- Khai báo thông tin cần delete

```
var deleteTable = {  
  name: 'sinhvien',  
  cols: [{  
    name: 'name',  
    value: 'Nguyen Van Dinh'  
  }],  
  wheres: [{  
    name: 'id',  
    value: 1  
  }]  
}
```

- Lệnh delete dữ liệu trong bảng (sử dụng thư viện viết sẵn)

```
db.delete(deleteTable)  
  .then(data => {  
    console.log("Đã delete thành công!", data);  
  })  
  .catch(err => {  
    console.log("Lỗi", err);  
  });
```


g. Server insert thông tin vào DB

- Lắng nghe dữ liệu từ client post lên và thêm vào DB

```
app.post('/add', bodyParser, (req, res) => {
  let insertTable = {
    name: 'sinhvien',
    cols: [
      {
        name: 'name',
        value: req.body.name
      }
    ]
  };
  db.insert(insertTable)
    .then(data => {
      res.send({ status: "ok", message: "insert thanh cong: " + data });
    })
    .catch(err => {
      res.send({ status: "nok", message: "insert that bai: " + err });
    });
});
```

- Lắng nghe thông tin client post lên, select dữ liệu từ DB, kiểm tra login

```
app.post('/login', bodyParser, (req, res) => {
  var promise = new Promise((resolve, reject) => {
    db.getRsts("SELECT * FROM sinhvien")
      .then(data => {
        resolve(data);
      })
      .catch(err => {
        reject(err);
      });
  })
  promise.then(data => {
    let yes = 0;
    data.forEach(el=>{
      if (req.body.name == el.name) {
        yes = 1;
        res.send({ status: "ok", message: "login thanh cong!" });
        return true;
      }
    });
    if (yes == 0) res.send({ status: "nok", message: "login that bai!" });
  })
  .catch(err => {
    console.log(err);
  })
});
```

h. Mở rộng:

- Tạo Server:

```
const express = require('express');
const app = express();

var main = (isHttp) => {

  const resource = require('./routes/resource');
  app.use('/user', resource);

  if (isHttp) {
    const httpServer = require('http').createServer(app);
    const portHttp = process.env.PORT || isHttp;
    httpServer.listen(portHttp, () => console.log("Server is running!"));
  }
}

const isHttp = 8080;

main(isHttp);
```

- File ./routes/resource

```
const router = require('express').Router();

const handlers = require('../handlers/handler');

router.get('/get-users', handlers.getUsers);
router.post('/add', handlers.postAddUser);

module.exports = router;
```

- File ../handlers/handler

```
const SQLiteDAO = require('../db/sqlite3/sqlite-dao');

const dirDB = 'db';
const dbFile = './' + dirDB + '/users.db';

var db = new SQLiteDAO(dbFile);

const getUsers = (req, res) => {
  res.send({ status: "ok", message: "login thanh cong!" });
}

module.exports = {
  getUsers: getUsers,
  postAddUser: postAddUser
};
```

- Insert dữ liệu vào DB

- Lắng nghe dữ liệu từ client post lên và thêm vào DB

```
const postAddUser = (req, res) => {
  let postDataString = '';
  req.on('data', (chunk) => {
    postDataString = chunk;
  });
  req.on('end', () => {
    req.json_data = JSON.parse(postDataString);
    try {
      let insertTable = {
        name: 'sinhvien',
        cols: [
          {
            name: 'name',
            value: req.json_data.name
          }
        ]
      };
      db.insert(insertTable)
        .then(data => {
          res.send({ status: "ok", message: "insert thanh cong: " + data });
        })
        .catch(err => {
          res.send({ status: "nok1", message: "insert that bai: " + err });
        });
    } catch (err) {
      res.send({ status: "nok2", message: "insert that bai: " + err });
    }
  })
}
```

II - Đưa ứng dụng lên heroku

- Khai báo file Procfile: **web: npm start**
- Trong file **package.json**, thêm đoạn này vào:

```
"scripts": {  
  "start": "node server.js"  
},
```

- Thêm đoạn này vào file **package.json** (nếu chưa có)

```
"cordova": {  
  "plugins": {  
    "cordova-plugin-whitelist": {},  
    "cordova-plugin-statusbar": {},  
    "cordova-plugin-device": {},  
    "cordova-plugin-splashscreen": {},  
    "cordova-plugin-ionic-webview": {},  
    "cordova-plugin-ionic-keyboard": {}  
  },  
  "platforms": [  
    "browser"  
  ]  
}
```

- Trong file **server.js**: Thêm đoạn này vào

```
app.use(express.static(__dirname + '/www'));
```

- Thay đổi đường dẫn server thật chứ không dùng server localhost nữa
- Chạy lệnh **ionic cordova build browser --prod --release** để tạo ra thư mục **www** từ **src** rồi copy thư mục **www** vào để push lên heroku
- Copy toàn bộ những thư mục liên qua đến server vào
- Tiến hành push lên heroku
 - Khi push lên heroku thì trong thư mục dự án chỉ cần những thư mục hoặc file:
 - .git
 - www
 - gitignore
 - package.json
 - Procfile
 - server.js