[npm]

npm은 Node Package Manager의 약어로, 이름 그대로 노드 패키지 매니저입니다. 1.1.2절에서는 노드가 자바스크립트 프로그램을 컴퓨터에서도 실행할 수 있게 해준다고 이야기했습니다. 대

부분의 자바스크립트 프로그램은 패키지라는 이름으로 npm에 등록되어 있으므로 특정 기능을 하는 패키지가 필요하다면 npm에서 찾아 설치하면 됩니다.

서비스에 필요한 패키지를 하나씩 추가하다 보면 어느새 패키지 수가 100개를 훌쩍 넘어버리게 됩니다. 그리고 사용할 패키지는 저마다 고유한 버전이 있으므로 어딘가에 기록해두어야 합니다. 같은 패키지라도 버전별로 기능이 다를 수 있으므로 프로젝트를 설치할 때 패키지도 동일한 버전을 설치하지 않으면 문제가 생길 수 있습니다. 이때 설치한 패키지의 버전을 관리하는 파일이 바로 package.json입니다.

따라서 노드 프로젝트를 시작하기 전에는 폴더 내부에 무조건 package.json부터 만들고 시작해야 합니다. npm은 package.json을 만드는 명령어를 제공합니다.

먼저 콘솔로 프로젝트를 시작할 폴더로 이동한 후, 다음 명령어를 입력합니다.

```
source₩ex_08_package> npm init
```

```
package name: (폴더명) [프로젝트 이름 입력]

version: (1.0.0) [프로젝트 버전 입력]
description: [프로젝트 설명 입력]
entry point: index.js
test command: [엔터 키 클릭]
git repository: [엔터 키 클릭]
keywords: [엔터 키 클릭]
author: [여러분의 이름 입력]
license: (ISC) [엔터 키 클릭]
```

npm init 실행이 완료되면 폴더에 다음과 같은 파일이 생성됩니다.

package.json

```
{
  "name": "npmtest",
  "version": "0.0.1",
  "description": "hello package.json",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "ZeroCho",
  "license": "ISC"
}
```

[팩키지 설치]

express라는 팩키지를 설치해봅니다.

source₩ex_08_package> **npm i express**

드디어 첫 패키지를 설치했습니다! 설치한 패키지가 package.json에 기록됩니다.

package.json

```
{
  "name": "npmtest",
  ...
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
  }
}
```

dependencies라는 속성이 새로 생겼고, express라는 이름과 함께 설치된 버전이 저장되었습니다. 설치된 버전은 이 책과 다를 수 있습니다. 버전 앞에 ^ 표시가 붙어있는데, 여기에는 특별한 의미가 있습니다. 다음 절에서 자세히 알아보겠습니다.

추가로 node_modules라는 폴더도 생성되었습니다. 그 안에 설치한 패키지들이 들어 있습니다. 분명히 Express 하나만 설치했는데 패키지가 여러 개 들어 있습니다. 이는 Express가 의존하는 패키지들입니다. 패키지 하나가 다른 여러 패키지에 의존하고, 그 패키지들은 또 다른 패키지들에 의존합니다. 이렇게 의존 관계가 복잡하게 얽혀 있어 package.json이 필요한 것입니다.

[express 웹서버 만들기]

익스프레스는 http 모듈의 요청과 응답 객체에 추가 기능들을 부여했습니다. 기존 메서드들도 계속 사용할 수 있지만, 편리한 메서드들을 추가하여 기능을 보완했습니다. 또한, 코드를 분리하기 쉽게 만들어 관리하기도 용이합니다. 그리고 더 이상 if문으로 요청 메서드와 주소를 구별하지 않아도 됩니다.

express라는 팩키지를 설치해봅니다. (해당 디렉토리에 이미 설치하였으면 다시 설치안해도 됨)

source₩ex_08_package> **npm i express**

source₩ex_08_package> index.html

```html
<html>
<head>
  <meta charset="UTF-8" />
  <title>익스프레스 서버</title>
</head>
<body>
  <h1>익스프레스</h1>
  <p>배워봅시다.</p>
</body>
</html>
```

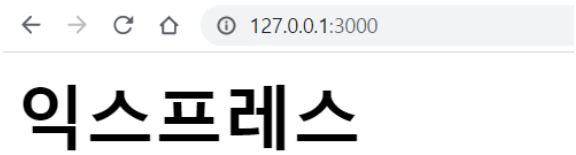
source₩ex_08_package>app.js

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

const app = express();
app.set('port', process.env.PORT || 3000);

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, '/index.html'));
});
```

```
app.listen(app.get('port'), () => {
  console.log(app.get('port'), '번 포트에서 대기 중');
});
```

실행

```
PS C:\leeji\node.js\source\ex08_package> node app
3000 번 포트에서 대기 중
```



[session]

express, express-session 패키지를 설치

항상

package.json을 제일 먼저 생성해야 합니다. package.json을 생성해주는 npm init 명령어를 콘솔에서 호출해도 되고 직접 파일을 만들어도 됩니다. version이나 description, author, license는 원하는 대로 자유롭게 수정해도 괜찮습니다.

```
source₩ex_09_package> npm init
source₩ex_09_package> npm i express
source₩ex_09_package> npm i express-session, session-file-store
```

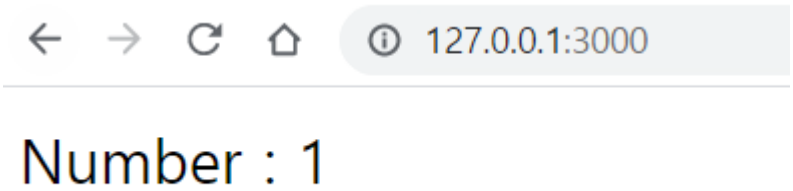
미들웨어는 app.use와 함께 사용됩니다. app.use(미들웨어) 꼴입니다. 익스프레스 서버에 미들웨어를 연결해봅시다.

source₩ex_09_package>session_app.js

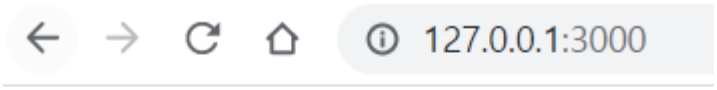
```
const express = require('express');
const session = require('express-session');
const FileStore = require('session-file-store')(session); // 1
const app = express();
app.set('port', process.env.PORT || 3000);

app.use(session({  // 2
  secret: 'keyboard cat',  // 암호화
  resave: false,
  saveUninitialized: true,
  store: new FileStore()
}));

app.get('/', (req, res, next) => {  // 3
  console.log(req.session);
  if(!req.session.num){
    req.session.num = 1;
  } else {
    req.session.num = req.session.num + 1;
  }
  res.send(`Number : ${req.session.num}`);
});

app.listen(app.get('port'), () => {
  console.log(app.get('port'), '번 포트에서 대기 중');
});
```

실행

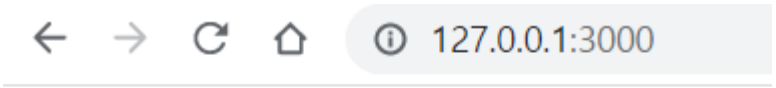
```
PS C:\leeji\node.js\source\ex09_session> node session_app
3000 번 포트에서 대기 중
```



다시, 요청하면 number 값 증가함.



Number : 2



Number : 3

세션

```
req.session.name = 'zerocho'; // 세션 등록
req.sessionID; // 세션 아이디 확인
req.session.destroy(); // 세션 모두 제거
```

[라우터로 간단히 경로 지정]

express, express-session 팩키지를 설치

source₩ ex_express_router > **npm init**

source₩ ex_express_router > **npm i express , express-session, session-file-store**

source₩ ex_express_router > **npm i morgan , cookie-parser , dotenv**

(참고 : module not found 에러를 만나면 해당 모듈을 설치하여 해결. npm i 설치할모듈이름)

ex_express_router/app.js

```
const express = require('express');
const morgan = require('morgan');
const cookieParser = require('cookie-parser');
const session = require('express-session');
const dotenv = require('dotenv');
const path = require('path');

dotenv.config();
const indexRouter = require('./routes');
const userRouter = require('./routes/user');

const app = express();
app.set('port', process.env.PORT || 3000);

app.use(morgan('dev'));
app.use('/', express.static(path.join(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser(process.env.COOKIE_SECRET));
app.use(session({
  resave: false,
  saveUninitialized: false,
  secret: process.env.COOKIE_SECRET,
  cookie: {
    httpOnly: true,
    secure: false,
  },
  name: 'session-cookie',
}));

app.use('/', indexRouter);
app.use('/user', userRouter);

app.use((req, res, next) => {
```

```
  res.status(404).send('Not Found');
});

app.use((err, req, res, next) => {
  console.error(err);
  res.status(500).send(err.message);
});

app.listen(app.get('port'), () => {
  console.log(app.get('port'), '번 포트에서 대기 중');
});
```

ex_express_router/router/index.js

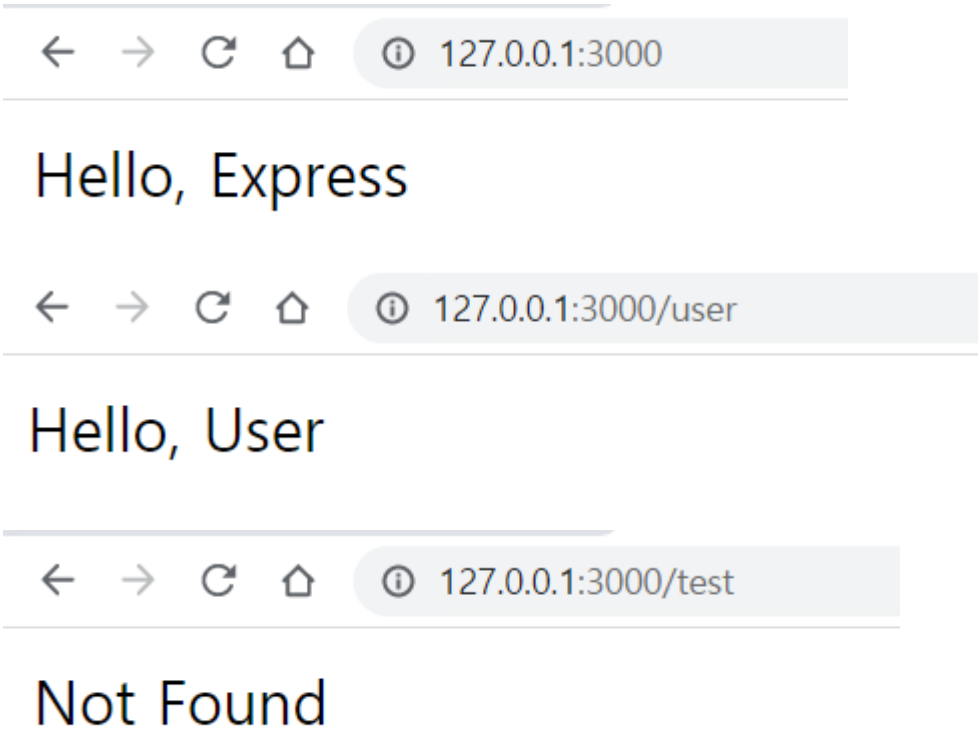
```
const express = require('express');

const router = express.Router();

// GET / 라우터
router.get('/', (req, res) => {
  res.send('Hello, Express');
});

module.exports = router;
```

ex_express_router/router/user.js

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

const router = express.Router();

// GET /user 라우터
router.get('/', (req, res) => {
  res.send('Hello, User');
});

module.exports = router;
```

실행

```
PS C:\leeji\node.js\source\ex_Express_router> node app
3000 번 포트에서 대기 중
```



[사용자 입력 값 서버에서 읽기]

express, express-session 팩키지를 설치

source₩ ex_express_router_param > **npm init**
source₩ ex_express_router_param > **npm i express , express-session, session-file-store**
source₩ ex_express_router_param > **npm i morgan , cookie-parser , dotenv**

ex_express_router_param /routes/login.html

```
<!DOCTYPE html>
<html>
 <body>
   <form action="http://127.0.0.1:3000/loginprocess" method="post">
        <input type="text" id="id" name="id" />
        <input type="password" id="pw" name="pw" />
        <input type="submit"   name="login" />
   </form>
 </body>
</html>
```

ex_express_router_param /routes/main.html

```
<!DOCTYPE html>
<html>
 <body>
    main 입니다.
 </body>
</html>
```

ex_express_router_param / login.js

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

const router = express.Router();

// GET /login 라우터
router.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'login.html'));
});

module.exports = router;
```

ex_express_router_param / loginprocess.js

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

const router = express.Router();



// POST 방식 요청이면  /loginProcess 라우터
```

```
router.post('/', (req, res) => {
  let  id  = req.body['id'] // POST 방식 요청이면  req.body[".."] 로
  let  pw  = req.body['pw'] // POST 방식 요청이면  req.body[".."] 로

  console.log("req.query=", req.query, "req.body=", req.body, id)

  if (id=="kim" & pw=="1234"){ //로그인 성공
    // res.sendFile(path.join(__dirname, 'main.html'));

    if(!req.session.loginuser){
      req.session.loginuser = id;
    }
    res.send(`로그인 유저 : ${req.session.loginuser} 님 안녕하세요` );


  }else {
    res.send("로그인 실패. 다시 로그인하세요.")

  }
});


// GET 방식 요청이면  req.query[".."] 로  사용자가 입력한 값 읽어야 함.
router.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'login.html'));
  let  id  = req.query['id']// GET 방식 요청이면  req.query[".."] 로
  let  pw  = req.query['pw']// GET 방식 요청이면  req.query[".."] 로
  console.log("req.query=", req.query, "req.body=", req.body, id)
});
// 중략

module.exports = router;
```

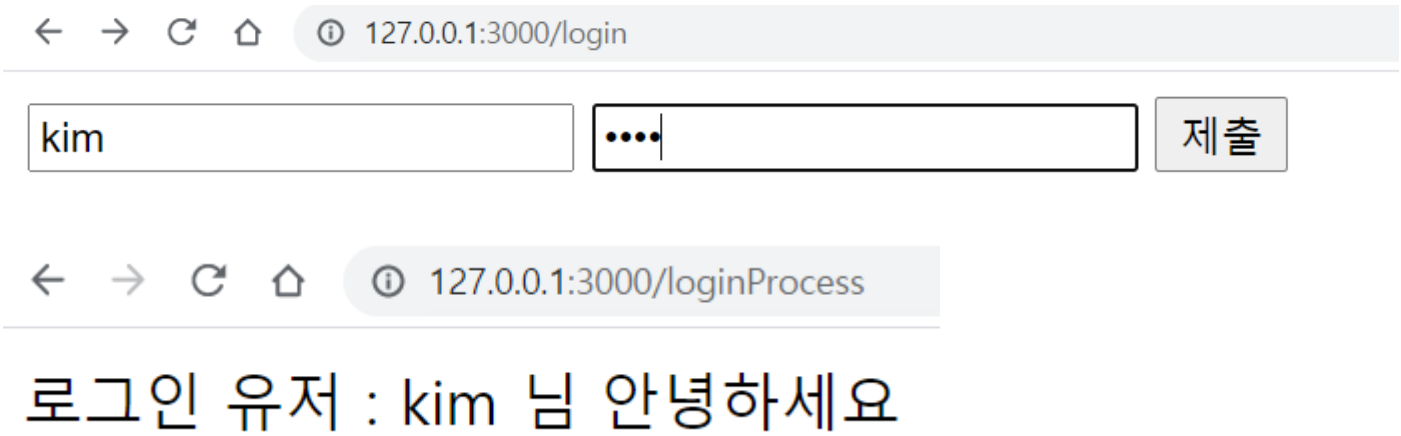
ex_express_router_param / app.js

```
const express = require('express');
const morgan = require('morgan');
const cookieParser = require('cookie-parser');
const session = require('express-session');
const dotenv = require('dotenv');
const path = require('path');

dotenv.config();
const loginRouter = require('./routes/login');
const loginProcessRouter = require('./routes/loginProcess');

const app = express();
app.set('port', process.env.PORT || 3000);
```

```
app.use(morgan('dev'));
app.use('/', express.static(path.join(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser(process.env.COOKIE_SECRET));
app.use(session({
  resave: false,
  saveUninitialized: false,
  secret: process.env.COOKIE_SECRET,
  cookie: {
    httpOnly: true,
    secure: false,
  },
  name: 'session-cookie',
}));

app.use('/login', loginRouter);
app.use('/loginprocess', loginProcessRouter);

app.use((req, res, next) => {
  res.status(404).send('Not Found');
});

app.use((err, req, res, next) => {
  console.error(err);
  res.status(500).send(err.message);
});

app.listen(app.get('port'), () => {
  console.log(app.get('port'), '번 포트에서 대기 중');
});
```

실행

```
PS C:\leeji\node.js\source\ex_express_router_param> node app
3000 번 포트에서 대기 중
```



[세션과 HTML Rendering]

express, express-session 팩키지를 설치

source₩ > cd "ex_express_router_param 2"
source₩ ex_express_router_param 2> **npm init**

source₩ ex_express_router_param 2> **npm i express , express-session, session-file-store**
source₩ ex_express_router_param 2> **npm i morgan , cookie-parser , dotenv**
source₩ ex_express_router_param 2> **npm i ejs**

ex_express_router_param_2/loginprocess.js

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

const router = express.Router();



// POST 방식 요청이면  /loginProcess 라우터
router.post('/', (req, res) => {
  let  id  = req.body['id'] // POST 방식 요청이면  req.body[".."] 로
  let  pw  = req.body['pw'] // POST 방식 요청이면  req.body[".."] 로

  console.log("req.query=", req.query, "req.body=", req.body, id)

  if (id=="kim" & pw=="1234"){ //로그인 성공
    // res.sendFile(path.join(__dirname, 'main.html'));

    if(!req.session.loginuser){
      req.session.loginuser = id;
    }
    // res.send(`로그인 유저 : ${req.session.loginuser} 님 안녕하세요` );
    res.render('main',{userid:req.session.loginuser});

  }else {
    res.send("로그인 실패. 다시 로그인하세요.")

  }
});


// GET 방식 요청이면  req.query[".."] 로  사용자가 입력한 값 읽어야 함.
```

```
router.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'login.html'));
  let  id  = req.query['id']// GET 방식 요청이면  req.query[".."] 로
  let  pw  = req.query['pw']// GET 방식 요청이면  req.query[".."] 로
  console.log("req.query=", req.query, "req.body=", req.body, id)
});


module.exports = router;
```

views_ejs/main.ejs

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    example ejs
    <h1><%= userid %></h1>
    <footer>
    </footer>
  </body>
</html>
```

App.js

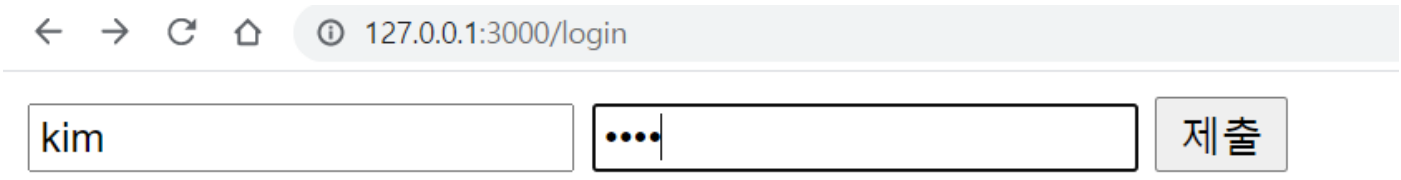
Html rendering을 위해 아래 두 코드 추가.

app.set('view engine','ejs');

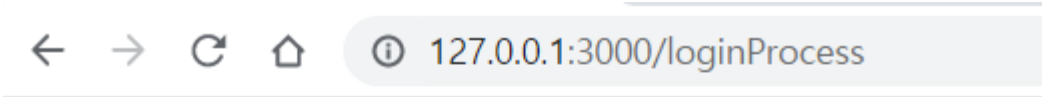
app.set('views','./views_ejs');

```
const express = require('express');
const morgan = require('morgan');
const cookieParser = require('cookie-parser');
const session = require('express-session');
const dotenv = require('dotenv');
const path = require('path');

dotenv.config();
const loginRouter = require('./routes/login');
const loginProcessRouter = require('./routes/loginProcess');

const app = express();
app.set('port', process.env.PORT || 3000);
```

```
app.use(morgan('dev'));
app.use('/', express.static(path.join(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser(process.env.COOKIE_SECRET));
app.use(session({
  resave: false,
  saveUninitialized: false,
  secret: process.env.COOKIE_SECRET,
  cookie: {
    httpOnly: true,
    secure: false,
  },
  name: 'session-cookie',
}));

app.use('/login', loginRouter);
app.use('/loginprocess', loginProcessRouter);

app.use((req, res, next) => {
  res.status(404).send('Not Found');
});

app.use((err, req, res, next) => {
  console.error(err);
  res.status(500).send(err.message);
});

app.set('view engine','ejs');
app.set('views','./views_ejs');

app.listen(app.get('port'), () => {
  console.log(app.get('port'), '번 포트에서 대기 중');
});
```



]



example ejs

kim