SOPT 17th – Connect & Express

03. Connect + Express (Module – Cookie / Session)

Review

Connect

basic

```
// node_connect.js

var Connect = require('connect');
var server = new Connect();

server.use (function (request, response) {

    response.writeHead(200, { 'Content-Type' : 'text/plain' });
    response.end('Hello, Node.js');
});

server.listen(8080, function () {

    console.log('Server running on port 8080 :)');
});
```

Connect

basic

```
// node_connect.js

var Connect = require('connect');
var server = new Connect();

server.use (function (request, response) {
    response.writeHead(200, { 'Content-Type': 'text/plain' });
    response.end('Hello, Node.js');
});

server.listen(8080, function () {
    console.log('Server running on port 8080 :)');
});
```

HTTP 모듈과 뭔 차이점이 있는지 ?

Connect

middleware

Various middleware exist to enhance the function of module ‘connect’

- Logging : morgan
- Request Routing : router
- Parser : cookie-parser, body-parser
- Session : session

…

Express

- **Web app, REST API**에 효율적인 프레임워크
- 미들웨어 사용법 숙지 및 관련 예제 작성

Express

기존 모듈과의 관계 – HTTP, Connect

- **HTTP**모듈은 **Node.js**의 내장모듈이며, 캐싱, 라우팅 등의 로직을 개발자가 직접 구현해야 한다.
- **Connect** 모듈은 개발자가 필요에 따라 미들웨어를 추가하여 개발할 수 있다.
- **Express 3.X**까지는 내부적으로 **Connect**를 사용하여 랩핑한 것이나, **4.X** 부터는 사용하지 않는다.
- 현재까지는 **Connect / Express** 에서 사용하던 미들웨어가 호환된다.

특징

- **HTTP, Connect**보다 더 많은 요청 / 응답 메서드를 제공한다.
- **Connect**에서 사용하는 다수의 미들웨어와 호환되어 기능을 확장할 수 있다.
- 구조적으로 **MVC**가 분리되어 **Webapp** 개발에 용이하며, **REST API** 작성 및 프로토타이핑에 편리하다.

Express

Install

- **npm install express-generator -g**
- **mkdir 프로젝트 폴더 && cd 프로젝트 폴더**
- **express 프로젝트 이름 --ejs**
- **npm install**
- **Windows : set DEBUG = 프로젝트 이름 & node.\bin\www**
  Mac OS X : DEBUG = 프로젝트 이름 ./bin/www

프로젝트 구조

- **Require –** 모듈 참조
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

프로젝트 구조

- **Require – 모듈 참조**
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

```
var path = require('path');

var body_parser = require('body-parser');
var express = require('express');
var express_session = require('express-session');

var routes = require('./routes/index');

var app = express();


// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```

Express

Structure

프로젝트 구조

- **Require –** 모듈 참조
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

```
var routes = require('./routes/index');

var app = express();


// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');


// body-parser setup
app.use(body_parser.urlencoded ({

     extended : false
}));

// session setup
app.use(express_session({

     name : 'session_id',
     saveUninitialized  : false,
     secret : 'keyboard cat',
     resave : false,
}));

// router setup
app.use('/', routes);
```

Express

Structure

프로젝트 구조

- **Require –** 모듈 참조
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

```
var path = require('path');

var body_parser = require('body-parser');
var express = require('express');
var express_session = require('express-session');

var routes = require('./routes/index');

var app = express();


// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');


// body-parser setup
app.use(body_parser.urlencoded ({

    extended : false
}));

// session setup
app.use(express_session({
```

Express

Structure

프로젝트 구조

- **Require –** 모듈 참조
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

```
// body-parser setup
app.use(body_parser.urlencoded ({

    extended : false
}));

// session setup
app.use(express_session({

    name : 'session_id',
    saveUninitialized  : false,
    secret : 'keyboard cat',
    resave : false,
}));

// router setup
app.use('/', routes);


// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers
```

Express

Structure

프로젝트 구조

- **Require –** 모듈 참조
- **Express** 모듈 초기화
- 미들웨어 초기화 및 적용
- 라우팅
- 할당되지 않은 요구에 대한 에러 핸들링

```
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handlers

// a. development error handler - will print stacktrace
if (app.get('env') === 'development') {
  app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
      message: err.message,
      error: err
    });
  });
}

// b. production error handler - no stacktraces leaked to user
app.use(function(err, req, res, next) {
  res.status(err.status || 500);
  res.render('error', {
    message: err.message,
    error: {}
  });
});
```

Express

Comparison

파라미터 추출

응답 메시지 작성

```
var server = http.createServer(function (req, res) {

   var parsed = url.parse(req.url);

   if (req.method == "GET") {

      if (parsed.pathname.startsWith("/") {

         res.writeHead(200, { Content-Type : 'text/html' });
         res.write('<h1>Hello with HTTP / Connect</h1>');
      }
   }

   if (req.method == "POST") {

      if (parsed.pathname.startsWith("/articles") {

         res.writeHead(200, { Content-Type : 'application/json' });
         res.end(JSON.stringify({ result: true, id: 'a10295612' }));
      }
   }
});
```

HTTP

```
router.get("/", function (req, res) {

   res.send('<h1>Hello with Express</h1>');

});

router.post("/articles", function (req, res) {

   res.json({ result: true, id: 'a10295612' });
});
```

Express

Express

Comparison

헤더 및 응답 코드 전송

파라미터 추출

응답 메시지 작성

```
var server = http.createServer(function (req, res) {

   var req_url = url.parse(req.url);
   var parsed = qs.parse(req.url.querystring);

   var query = parsed.query;

   …

});
```

HTTP

```
router.get("/", function (req, res) {

   var query = res.param('query');

   …

});
```

Express

Express

Comparison

헤더 및 응답 코드 전송

```
var server = http.createServer(function (req, res) {

   res.writeHead(200, {

      Content-Type : 'text/html',
      Content-Length : 120861392
      ETag : '9d62b696fd054f59b393221592bfaf2f'

   });
});
```

HTTP

```
router.post("/photos/:id", function (req, res) {

   res.status(200);
   res.set({

      Content-Type : 'text/html',
      Content-Length : 120861392
      ETag : '9d62b696fd054f59b393221592bfaf2f'

   });
});
```

Express

Express

Comparison

렌더링 처리

리디렉션 처리

헤더 및 응답 코드 전송

```
var server = http.createServer(function (req, res) {
   res.writeHead(302, { Location : 'http://sopt.org' });
});
```

HTTP

```
router.get("/", function (req, res) {
   res.redirect('http://sopt.org');
});
```

Express

Express

Comparison

쿠키 처리

렌더링 처리

리다렉션 처리

```
var server = http.createServer(function (req, res) {

   var context = {};

   …

   fs.readFile('page.ejs', 'utf8', function (err, data) {

      if (!err) {

         res.writeHead(200, { Content-Type : 'text/html' });
         res.end(ejs.render(data, context));
      }

   });

});
```

HTTP

```
router.get("/", function (req, res) {

   var context = {};

   …

   res.render('page', context);

});
```

Express

Express

Comparison

쿠키 처리

렌더링 처리

```
var server = http.createServer(function (req, res) {

   var cookies = request.headers.cookie;
   cookies = cookies.split(';').map(function (e) {

      e = e.split('=');
      return { e[0] : e[1] };
   });

   var last_visit = cookies.last_visit;

   …

   res.writeHead(200, {

      'Set-Cookie' : [ 'last_visit=' + new Date() ]
   }

});
```

HTTP

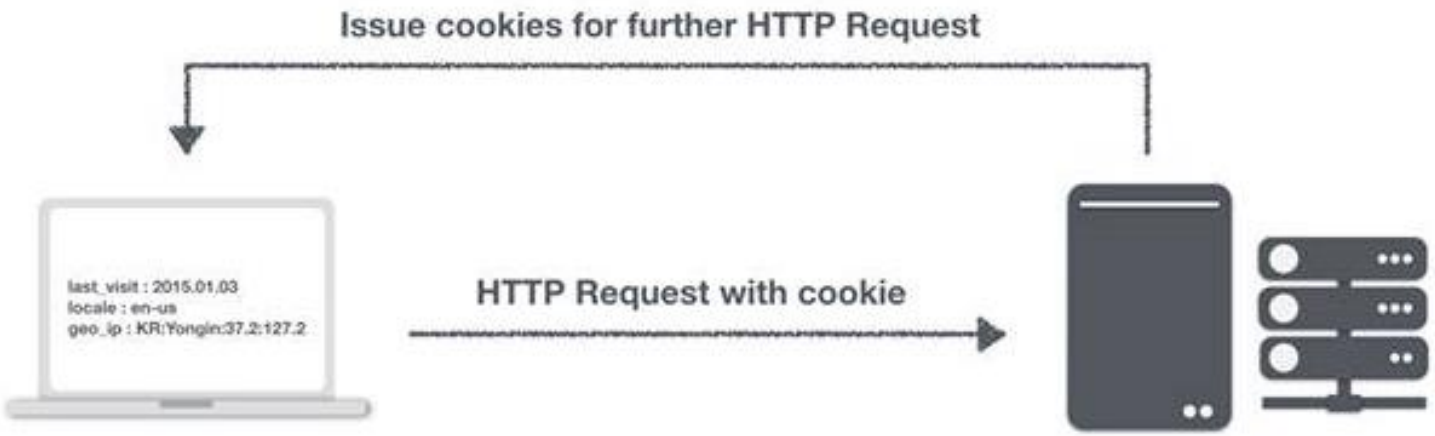
```
router.post("/sign_in", function (req, res) {

   var last_visit = req.cookies.last_visit;

   …

   res.cookie('last_visit', new Date());
});
```

Express

Cookie

› 개요, 시나리오, cookie-parser를 사용한 유저 추적

Cookie


Issue cookies for further HTTP Request
last_visit : 2015.01.03
locale : en-us
geo_ip : KR:Yongin:37.2:127.2
HTTP Request with cookie

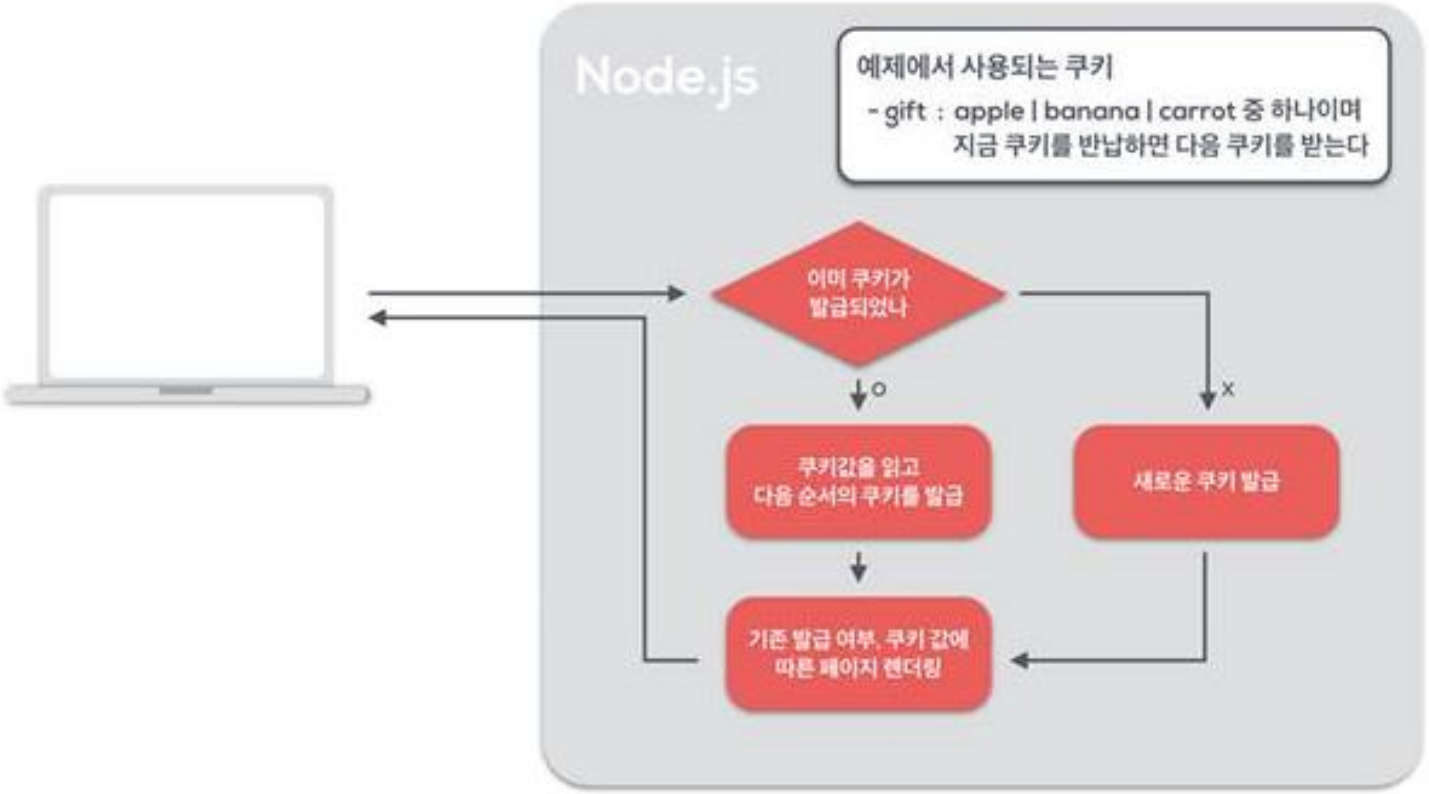
Cookie

Practice

실습 – **cookie-parser**를 사용한 유저 추적 시나리오

Cookie

Practice

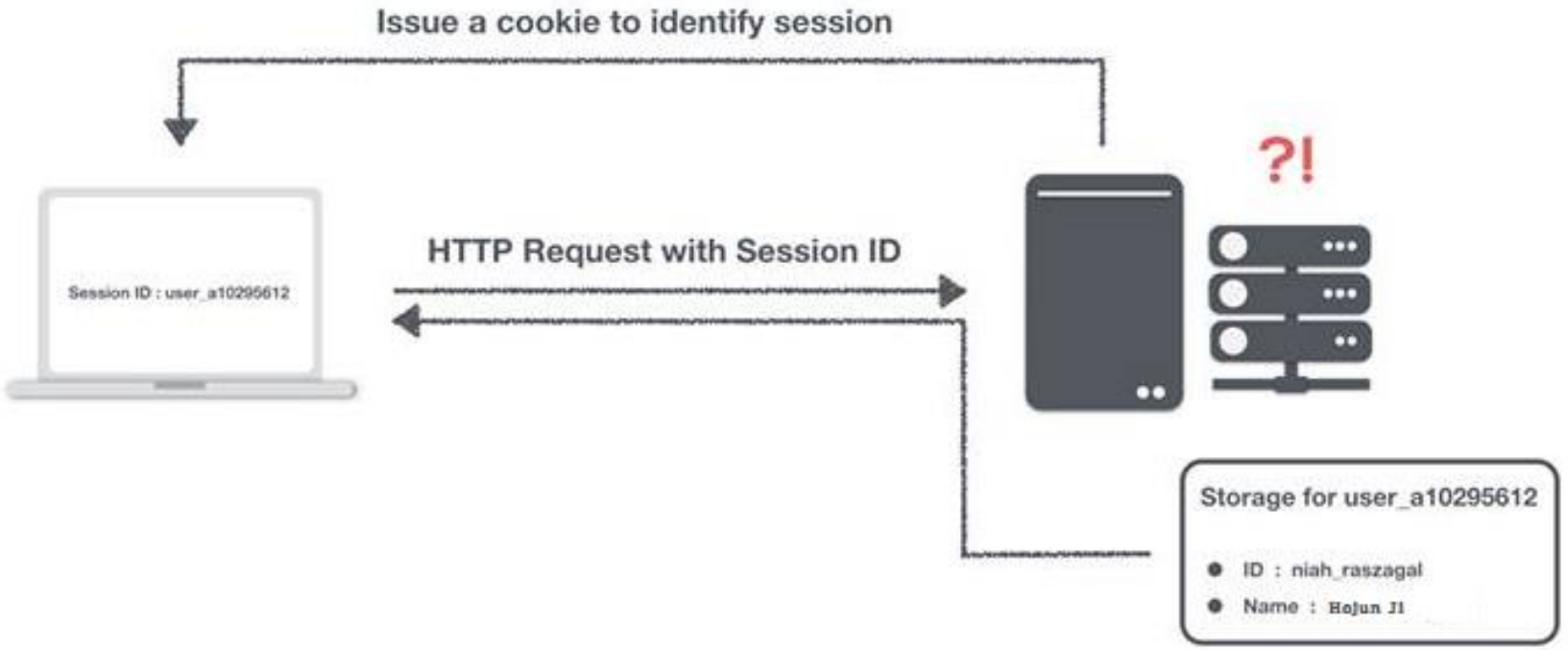


Session

> 개요 , 시나리오, express-session를 사용한 로그인 시나리오

Session

Structure



Session

Practice

실습 – **express-session**를 사용한 로그인 시나리오

Session

Practice