1. 개요
   1)서버가 제공하는 컨텐츠가 데스크 탑용 웹 브라우저가 아니라 스마트 폰에서 동작하는 어플리케이션도 서버에서 컨텐츠를 얻어서 사용자에게 제공한다.

   2)서버 구분
      -웹 기반 서비스 서버
      -모바일 웹을 위한 서버 : 모바일 서버

   3)뭐가 다른가
      -웹 기반 서비스
         --HTML을 기반으로 동작
         --태그를 이용해서 컨텐츠를 렌더링한다.
         --웹 브라우저
         --모바일 웹 브라우저
      -모바일 서버
         --자체적인 UI 작성하고 그 UI 요소를 이용해서 컨텐츠를 렌더링한다.
            --문서 구조와 렌더링 정보를 포함하는 HTML 불필요
         --화면에 출력되는 방식에 대한 데이터는 불필요
         --모바일 앱을 작성한 코드에서 데이터를 분석할 수 있는 정보 필요
         --이러한 문서의 형식은 XML과 JSON이다.
         --화면 이동(네비게이션)은 모바일 앱에서 담당하기 때문에 PRG 패턴 불필요
         --네이티브 웹

   4)웹 + 모바일 지원 서비스
      -API 서비스
         --모바일 웹
         --api.moviest.com/path
      -HTML 기반의 서비스
         --웹 브라우저를 위한 HTML 기반의 서비스
         --www.moviest.com/path

2. JSON
   1)JavaScript Object Notification
   2)초기 JavaScript에서 사용하던 포맷
      -이름 : 값 쌍으로 구성
      -'greeting' : 'Hello, World'

   3)이후 다양한 분야에서 사용
      -문서크기가 작아서 네트워크를 이용해서 주고 받기가 편리
      -기계(프로그램) 해석 가능

   4)JSON 객체'
      ```
      {
            "greeting" : "Good Morning",
            "when" : "morning"
      }
      ```
      -JSON 객체 내 객체
      ```
      {
         "who" : {
               "name" : "Jobs"
         }
      }
      ```
      -이때 객체 내의 프라퍼티는 유일해야 함.

   5)JSON 데이타 타입
      -숫자 : 숫자만
      -문자열 : 따옴표
      -부울 : true/false
      -null : null

-배열 : 대괄호([])
-객체 : 중괄호({})

6)JSON 의 예

```
{
   "feed" : {
      "entry" : [
         {
            "title" : "Bang Bang - Jessie J, Ariana Grande & Nicki Minaj",
            "price" : {"currency" : "USD"}
         },
         {
            "title" : "It was Always You - Maroon 5",
            "price" : {"currency" : "USD"}
         }
      ]
   }
}
```

7)Lab

<jsonServer.js>

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

var movieList = [{title:'아바타', director:'제임스 카메론'}];

http.createServer(function (req, res) {
    if ( req.method.toLowerCase() == 'post' ) {
        var buffer = '';
        req.on('data', function (chunk) {
            buffer += chunk;
        });

        req.on('end', function () {
            var parsed = JSON.parse(buffer);
            var titleData = parsed.title;
            var directorData = parsed.director;

            movieList.push({title:titleData, director:directorData});

            res.writeHead(200, {'Content-Type':'application/json'});
            res.end(JSON.stringify({result:'success'}));
        });
    }
    else {
        var result = {
              count : movieList.length,
              data : movieList
        };
        res.writeHead(200, {'Content-Type':'application/json'});
        res.write(JSON.stringify(result), 'utf-8');
        res.end();
    }
}).listen(3000);
```

결과 확인은 Postman으로

http://127.0.0.1:3000 GET방식

```
{
   "count": 1,
   "data": [
      {
```

```
                "title": "아바타",
                "director": "제임스 카메론"
            }
        ]
    }

    http://127.0.0.1:3000 POST방식 > Body > raw > JSON
    {
        "title" : "스타워즈",
        "director" : "조지 루카스"
    }
    결과
    {
        "result": "success"
    }

    http://127.0.0.1:3000 GET방식
    {
        "count": 2,
        "data": [
            {
                "title": "아바타",
                "director": "제임스 카메론"
            },
            {
                "title": "스타워즈",
                "director": "조지 루카스"
            }
        ]
    }

8)JSON 다루기
    -JSON : V8 내장 클래스, 모듈 로딩 불필요
    -JSON 생성
        JSON.stringify()
    -JSON Parsing
        JSON.parse()

9)JSON 생성 예
    [Lab : jsondemo.js]
    var entry = {
            profile : {
                name : '나훈아',
                job : 'Singer'
            }
    };

    var jsonObj = JSON.stringify(entry);
    console.log('stringify', jsonObj);
    -------------------------------------------------------
    stringify {"profile":{"name":"나훈아","job":"Singer"}}

10)JSON 파싱 --> JavaScript Object
    [Lab : jsondemo1.js]
    var json = '{ "profile" : { "name" : "나훈아", "job" : "Singer" }}';

    var parsed = JSON.parse(json);
    var profile = parsed.profile;
    console.log('name', profile.name);
    console.log('job', profile.job);
```

```
   console.log('age', profile.age);
   ----------------------------------------------------------
   name 나훈아
   job Singer
   age undefined

11)Node.js에서 JSON을 이용한 요청과 응답
   -JSON 요청
      --요청 메시지의 바디에 기록된 JSON
      --요청 객체에서 바디 메시지 분석
      --JSON.parse()

      function (req, res) {
         var buffer = '';
         req.on('data', function (chunk) {
            buffer += chunk;
         });

         req.on('end', function () {
            var parsed = JSON.parse(buffer);
            res.end('JSON Request Example');
         });
      }

      In Postman,
      http://127.0.0.1:3000 POST방식 > Body > raw > JSON
      {
         "title" : "스타워즈",
         "director" : "조지 루카스"
      }

   -JSON 응답
      --응답 메시지 바디에 JSON 기록하기
      --응답 데이터에서 JSON 생성
      --JSON.stringify()
      --Content-Type : application/json

      function(req, res){
         var data = {
            name : '나훈아',
            job : 'Singer'
         };
         res.setHeader('Content-Type':'application/json');
         res.end();
      }

12)Lab : formidable 모듈을 이용한 JSON 파일 파싱하기
   -input.html
      <!DOCTYPE html>
      <html lang="en">
      <head>
          <meta charset="UTF-8">
          <title>성적 프로그램</title>
      </head>
      <body>
          <form method='post' action='.' enctype="multipart/form-data">
              File : <input type='file' name='myfile' /><br />
              <input type='submit' value='Upload' />
          </form>
      </body>
```

```
        </html>

    -jsonserver1.js
        var http = require('http');
        var fs = require('fs');
        var formidable = require('formidable');

        var students = [];

        var server = http.createServer(function(request, response){
            if(request.url == '/' && request.method.toLocaleLowerCase() == 'get'){
                response.writeHead(200, {'Content-Type':'text/html'});
                fs.createReadStream('./input.html').pipe(response);
            }else if(request.url == '/result' && request.method.toLocaleLowerCase() ==
            'get'){
                response.writeHead(200, {'Content-Type':'text/html;charset=utf-8'});
                response.write('<ul>');
                students.forEach(function(item, index){
                    response.write('<li>' + JSON.parse(item).hakbun + '</li>');
                });
                response.write('</ul>');
                response.end();
            }else if(request.method.toLocaleLowerCase() == 'post'){
                var form = formidable.IncomingForm();
                form.parse(request, function(err, fields, files) {
                    fs.renameSync(files.myfile.path, './files/' + files.myfile.name);

                    var contents = fs.readFileSync('./files/' + files.myfile.name, 'utf8');
                    var array = JSON.parse(contents).students;
                    array.forEach(function(item, index){
                        students.push(JSON.stringify(item));
                    });
                    response.statusCode = 302;
                    response.setHeader('Location', '/result');
                    response.end();
                });
            }
        });

        server.listen(80, function(){
            console.log("Running");
        })

3. XML
    1)eXtensible Markup Language
    2)Markup language : 메타데이터로 문서 구조 표현
        -기계 해석 가능
        -HTML, XML
    3)사용분야
        -오피스 어플리케이션
        -안드로이드, iOS 같은 모바일의 레이아웃 정의 파일

    4)HTML vs XML
        -HTML
            --웹 브라우저를 위한 언어
            --사람을 위한 렌더링 정보
            --<h1>인터스텔라</h1>
        -XML
            --기계 해석을 위한 언어
            --코드로 분석 가능
```

```
      --<title>아바타</title>

   5)XML 구성요소
      -Tag
      -Element
      -Attribute
      -Markup, Contents

   6)XML 정의
      -XML 선언
         <?xml version="1.0" encoding="utf-8" ?>
      -Element : 논리 단위
         <greeting>Hello, World</greeting>
      -Tag
         --시작태그 <section>
         --종료태그 </second>
         --빈 태그 <line-break />
      -Attribute
         --태그 내의 이름=값 형식
         <step number="3">Connect A to B</step>

   7)XML 예
      <feed>
         <entry>
            <title>Bang Bang - Jessie J, Ariana Grande & Nicki Minaj</title>
            <price currency="USD">$1.29</price>
         </entry>
         <entry>
            <title>It was Always You - Maroon 5</title>
            <price currency="USD">$1.29</price>
         </entry>
      </feed>

   8)Node.js에서 XML 다루기
      -XML 요청
         --XML에서 데이터 분석
         --XML 파싱
      -XML 응답
         --데이터에서 XML 만들기

   9)XML 분석
      -XML parser module
         --libxmljs
            ---https://github.com/libxmljs/libxmljs
            ---$ npm install libxmljs
                              |
                              |-----libxmljs@0.18.4

         --xml-stream
         --xmldoc

      -XML Parser 방식
         --DOM parser
         --SAX parser
         --PULL parser

   10)DOM Parsing
      -각 요소의 요서를 객체화
         --Parent Node
         --Child Node
```

```
359         --Sibling Node
360       -DOM Parser
361         --Node
362           ---getChildNode
363           ---getParentNode
364           ---getSiblingNode
365           ---getNodeName
366           ---getNodeValue
367
368     11)DOM Parser 예제
369       var libxmljs = require('libxmljs');
370
371       //DOM Parsing
372       var xmlDoc = libxmljs.parseXml(xml);
373
374       //루트의 자식 노드 중 foo 어트리뷰트 값 출력
375       var children = xmlDoc.root().childNodes();
376       var child = children[0];
377       console.log(child.attr('foo').value());
378
379     12)XML 응답
380       -XML 생성 모듈 : jstoxml
381       -https://github.com/davidcalhoun/jstoxml
382       -Install
383         $ npm install jstoxml
384               |
385               |-----jstoxml@0.2.4
386       -Contents Type'
387         application/xml
388
389     13)객체 --> XML 생성
390       var entry = {
391         profile : {
392           name : "나훈아",
393           job : "Singer"
394         }
395       };
396       var jstoxml = require('jstoxml');
397       var xmlContent = jstoxml.toXML(entry, {header:true});
398
399     14)XML 요청과 응답
400       -XML 요청
401         <movie>
402           <title>스타워즈</title>
403           <director>조지 루카스</director>
404         </movie>
405       -XML 응답
406         <result>
407           <count>2</count>
408           <data>
409             <title>아바타</title>
410             <director>제임스 카메론</director>
411             <title>스타워즈</title>
412             <director>조지 루카스</director>
413           </data>
414         </result>
415
416       if(req.methdo.toLowerCase() == "post"){
417         req.on('end', function(){
418           var xmlDoc = libxml.parseXmlString(buffer);
```

```
            var title = xmlDoc.get('/movie/title').text();
            var director = xmlDoc.get('/movie/director').text();
            movieList.push({title:title, director:director});
            res.writeHead(200, {'Content-Type' : 'application/xml'});
            res.end(jstoxml.toXML({result:'success'}));
         });
      }else{
         res.writeHead(200, {'Content-Type' : 'application/xml'});
         var data = {count:movieList.length, data:movieList};
         var result = jstoxml.toXML({result:data});
         res.end(result);
      }

   15)Lab
      <xmlServer.js>
         var http = require('http');
         var jstoxml = require('jstoxml');
         var libxmljs = require("libxmljs");

         var movieList = [{title:'아바타', director:'제임스 카메론'}];

         http.createServer(function (req, res) {
               if ( req.method.toLowerCase() == 'post' ){
                     var buffer = '';
                     req.on('data', function (chunk) {
                           buffer += chunk;
                     });

                     req.on('end', function(){
                           var xmlDoc = libxmljs.parseXml(buffer);
                           var title = xmlDoc.get('/movie/title').text();
                           var director = xmlDoc.get('/movie/director').text();
                           movieList.push({title:title, director:director});
                           res.writeHead(200, {'Content-Type' : 'application/xml'});
                           res.end(jstoxml.toXML({result:'success'}));
                     });
               }else {
                     res.writeHead(200, {'Content-Type' : 'application/xml'});
                     var data = {count:movieList.length, data:movieList};
                     var result = jstoxml.toXML({result:data});
                     res.end(result);
               }
         }).listen(3000);
         ------------------------------------------------
         In Postman,
         http:127.0.0.1:3000 GET
         <result>
               <count>1</count>
               <data>
                     <title>아바타</title>
                     <director>제임스 카메론</director>
               </data>
         </result>

         http://127.0.0.1:3000 POST방식 > Body > raw > XML(application/xml)
         <movie>
            <title>스타워즈</title>
            <director>조지 루카스</director>
         </movie>
```

결과 : <result>success</result>

http:127.0.0.1:3000 GET

```
<result>
	<count>2</count>
	<data>
		<title>아바타</title>
		<director>제임스 카메론</director>
		<title>스타워즈</title>
		<director>조지 루카스</director>
	</data>
</result>
```