```
1. 개요
 1
 2
     1)서버가 제공하는 컨텐츠가 데스크 탑용 웹 브라우저가 아니라 스마트 폰에서 동작하는 어플리케이션도
     서버에서 컨텐츠를 얻어서 사용자에게 제공한다.
 3
 4
     2)서버 구분
 5
        -웹 기반 서비스 서버
 6
        -모바일 웹을 위한 서버 : 모바일 서버
 7
 8
     3)뭐가 다른가
 9
        -웹 기반 서비스
10
          --HTML을 기반으로 동작
11
          --태그를 이용해서 컨텐츠를 렌더링한다.
12
          --웹 브라우저
13
          --모바일 웹 브라우저
14
        -모바일 서버
15
          --자체적인 UI 작성하고 그 UI 요소를 이용해서 컨텐츠를 렌더링한다.
16
            --문서 구조와 렌더링 정보를 포함하는 HTML 불필요
17
          --화면에 출력되는 방식에 대한 데이터는 불필요
18
          --모바일 앱을 작성한 코드에서 데이터를 분석할 수 있는 정보 필요
19
          --이러한 문서의 형식은 XML과 JSON이다.
20
          --화면 이동(네비게이션)은 모바일 앱에서 담당하기 때문에 PRG 패턴 불필요
21
          --네이티브 웹
22
23
     4)웹 + 모바일 지원 서비스
24
        -API 서비스
25
          --모바일 웹
26
          --api.moviest.com/path
27
        -HTML 기반의 서비스
28
          --웹 브라우저를 위한 HTML 기반의 서비스
29
          --www.moviest.com/path
30
31
   2. JSON
32
33
     1) Java Script Object Notification
34
     2)초기 JavaScript에서 사용하던 포맷
35
        -이름 : 값 쌍으로 구성
36
        -'greeting': 'Hello, World'
37
38
     3)이후 다양한 분야에서 사용
39
        -문서크기가 작아서 네트워크를 이용해서 주고 받기가 편리
40
        -기계(프로그램) 해석 가능
41
     4)JSON 객체'
42
43
        {
44
            "greeting": "Good Morning",
45
            "when": "morning"
46
        -JSON 객체 내 객체
47
48
          "who" : {
49
              "name": "Jobs"
50
51
52
        }
53
        -이때 객체 내의 프라퍼티는 유일해야 함.
54
55
     5)JSON 데이타 타입
        -숫자: 숫자만
56
57
        -문자열: 따옴표
58
        -부울: true/false
59
       -null: null
```

```
60
           -배열 : 대괄호([])
 61
           -객체 : 중괄호({})
 62
 63
        6)JSON 의 예
 64
           {
              "feed" : {
 65
                 "entry" : [
 66
 67
                   {
                      "title": "Bang Bang - Jessie J, Ariana Grande & Nicki Minaj",
 68
 69
                      "price" : {"currency" : "USD"}
 70
                   },
 71
                      "title": "It was Always You - Maroon 5",
 72
                      "price": {"currency": "USD"}
 73
 74
                   }
 75
                ]
              }
 76
 77
           }
 78
 79
        7)Lab
 80
           <jsonServer.js>
 81
              var http = require('http');
 82
 83
              var movieList = [{title:'아바타', director:'제임스 카메론'}];
 84
 85
              http.createServer(function (req, res) {
 86
                 if ( req.method.toLowerCase() == 'post' ) {
 87
                      var buffer = ";
                      reg.on('data', function (chunk) {
 88
 89
                          buffer += chunk;
 90
                      });
 91
                      req.on('end', function () {
 92
 93
                         var parsed = JSON.parse(buffer);
 94
                         var titleData = parsed.title;
 95
                         var directorData = parsed.director;
 96
 97
                         movieList.push({title:titleData, director:directorData});
 98
 99
                         res.writeHead(200, {'Content-Type':'application/json'});
                          res.end(JSON.stringify({result:'success'}));
100
101
                      });
102
                 }
                 else {
103
104
                      var result = {
105
                            count: movieList.length,
106
                            data: movieList
107
                      };
108
                      res.writeHead(200, {'Content-Type':'application/json'});
                      res.write(JSON.stringify(result), 'utf-8');
109
                      res.end();
110
111
112
              }).listen(3000);
113
114
              결과 확인은 Postman으로
              http://127.0.0.1:3000 GET방식
115
116
              {
                 "count": 1,
117
                 "data": [
118
119
                   {
```

```
120
                     "title": "아바타",
121
                     "director": "제임스 카메론"
122
                   }
123
                ]
             }
124
125
126
             http://127.0.0.1:3000 POST방식 > Body > raw > JSON
127
                "title": "스타워즈",
128
129
                "director": "조지 루카스"
130
             }
             결과
131
132
                "result": "success"
133
134
135
             http://127.0.0.1:3000 GET방식
136
137
             {
                "count": 2,
138
139
                "data": [
140
                   {
                     "title": "아바타",
141
142
                      "director": "제임스 카메론"
143
                   },
144
                      "title": "스타워즈",
145
                     "director": "조지 루카스"
146
147
148
                ]
149
             }
150
151
        8)JSON 다루기
152
           -JSON: V8 내장 클래스, 모듈 로딩 불필요
153
           -JSON 생성
154
             JSON.stringify()
155
           -JSON Parsing
156
             JSON.parse()
157
158
        9)JSON 생성 예
159
          [Lab: jsondemo.js]
160
          var entry = {
161
                profile : {
162
                   name: '나훈아',
163
                  job: 'Singer'
164
                }
165
          };
166
167
          var jsonObj = JSON.stringify(entry);
168
          console.log('stringify', jsonObj);
169
170
          stringify {"profile":{"name":"나훈아","job":"Singer"}}
171
172
        10)JSON 파싱 --> JavaScript Object
173
          [Lab: isondemo1.js]
          var json = '{ "profile" : { "name" : "나훈아", "job" : "Singer" }}';
174
175
176
          var parsed = JSON.parse(json);
177
          var profile = parsed.profile;
178
          console.log('name', profile.name);
179
          console.log('job', profile.job);
```

```
180
          console.log('age', profile.age);
181
          name 나훈아
182
183
          job Singer
184
          age undefined
185
186
       11)Node.js에서 JSON을 이용한 요청과 응답
187
          -JSON 요청
188
             --요청 메시지의 바디에 기록된 JSON
189
             --요청 객체에서 바디 메시지 분석
190
            --JSON.parse()
191
192
            function (req, res) {
               var buffer = ";
193
194
               req.on('data', function (chunk) {
195
                  buffer += chunk;
196
               });
197
198
               req.on('end', function () {
                  var parsed = JSON.parse(buffer);
199
200
                   res.end('JSON Request Example');
201
               });
202
             }
203
204
            In Postman,
205
            http://127.0.0.1:3000 POST방식 > Body > raw > JSON
206
               "title": "스타워즈",
207
               "director": "조지 루카스"
208
209
             }
210
          -JSON 응답
211
212
             --응답 메시지 바디에 JSON 기록하기
213
            --응답 데이터에서 JSON 생성
             --JSON.stringify()
214
215
            --Content-Type : application/json
216
            function(req, res){
217
218
               var data = {
                  name: '나훈아',
219
220
                  job: 'Singer'
221
222
               res.setHeader('Content-Type':'application/json');
223
               res.end();
224
225
226
       12)Lab: formidable 모듈을 이용한 JSON 파일 파싱하기
          -input.html
227
228
             <!DOCTYPE html>
229
             <html lang="en">
             <head>
230
231
                <meta charset="UTF-8">
232
                <title>성적 프로그램</title>
233
             </head>
234
             <body>
235
                <form method='post' action='.' enctype="multipart/form-data">
                   File: <input type='file' name='myfile' /><br />
236
                   <input type='submit' value='Upload' />
237
238
                </form>
239
             </body>
```

```
240
             </html>
241
242
          -jsonserver1.js
243
             var http = require('http');
244
             var fs = require('fs');
245
             var formidable = require('formidable');
246
247
             var students = [];
248
249
             var server = http.createServer(function(request, response){
250
                if(request.url == '/' && request.method.toLocaleLowerCase() == 'get'){
                    response.writeHead(200, {'Content-Type':'text/html'});
251
252
                   fs.createReadStream('./input.html').pipe(response);
253
                }else if(request.url == '/result' && request.method.toLocaleLowerCase() ==
                'get'){
254
                    response.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
255
                    response.write('');
256
                   students.forEach(function(item, index){
257
                       response.write('' + JSON.parse(item).hakbun + '');
258
                    });
259
                   response.write('');
260
                    response.end();
261
                }else if(request.method.toLocaleLowerCase() == 'post'){
262
                   var form = formidable.IncomingForm();
263
                   form.parse(request, function(err, fields, files) {
264
                       fs.renameSync(files.myfile.path, './files/' + files.myfile.name);
265
                       var contents = fs.readFileSync('./files/' + files.myfile.name, 'utf8');
266
                       var array = JSON.parse(contents).students;
267
268
                       array.forEach(function(item, index){
269
                           students.push(JSON.stringify(item));
270
                       });
271
                       response.statusCode = 302;
272
                       response.setHeader('Location', '/result');
273
                       response.end();
274
                   });
275
276
             });
277
278
             server.listen(80, function(){
279
                console.log("Running");
280
             })
281
     3. XML
282
283
        1)eXtensible Markup Language
284
        2)Markup language: 메타데이터로 문서 구조 표현
285
          -기계 해석 가능
286
          -HTML, XML
287
        3)사용분야
          -오피스 어플리케이션
288
289
          -안드로이드, iOS 같은 모바일의 레이아웃 정의 파일
290
291
        4)HTML vs XML
292
          -HTML
293
             --웹 브라우저를 위한 언어
294
             --사람을 위한 렌더링 정보
295
             --<h1>인터스텔라</h1>
296
          -XML
297
             --기계 해석을 위한 언어
298
             --코드로 분석 가능
```

```
299
            --<title>아바타</title>
300
301
       5)XML 구성요소
302
          -Tag
303
          -Element
304
          -Attribute
305
          -Markup, Contents
306
307
       6)XML 정의
308
          -XML 선언
309
             <?xml version="1.0" encoding="utf-8" ?>
310
          -Element : 논리 단위
311
             <greeting>Hello, World</greeting>
312
          -Tag
313
            --시작태그 <section>
314
            --종료태그 </second>
315
            --빈 태그 ebreak />
316
          -Attribute
317
            --태그 내의 이름=값 형식
318
            <step number="3">Connect A to B</step>
319
320
       7)XML 예
321
          <feed>
322
             <entry>
323
               <title>Bang Bang - Jessie J, Ariana Grande & Nicki Minaj</title>
324
               <price currency="USD">$1.29</price>
325
             </entry>
326
            <entry>
327
               <title>It was Always You - Maroon 5</title>
328
               <price currency="USD">$1.29</price>
329
             </entry>
330
          </feed>
331
332
       8)Node.js에서 XML 다루기
          -XML 요청
333
334
            --XML에서 데이터 분석
335
            --XML 파싱
336
          -XML 응답
337
            --데이터에서 XML 만들기
338
       9)XML 분석
339
          -XML parser module
340
            --libxmljs
341
342
               ---https://github.com/libxmljs/libxmljs
               ---$ npm install libxmljs
343
344
345
                            |----libxmljs@0.18.4
346
347
            --xml-stream
348
            --xmldoc
349
350
          -XML Parser 방식
351
            -- DOM parser
352
            --SAX parser
353
            --PULL parser
354
355
       10)DOM Parsing
356
          -각 요소의 요서를 객체화
            --Parent Node
357
358
            --Child Node
```

```
359
             --Sibling Node
360
          -DOM Parser
361
             --Node
362
                ---getChildNode
363
               ---getParentNode
364
               ---getSiblingNode
365
               ---getNodeName
366
               ---getNodeValue
367
        11)DOM Parser 예제
368
369
          var libxmljs = require('libxmljs');
370
371
          //DOM Parsing
372
          var xmlDoc = libxmljs.parseXml(xml);
373
          //루트의 자식 노드 중 foo 어트리뷰트 값 출력
374
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
          -https://github.com/davidcalhoun/jstoxml
381
382
          -Install
383
             $ npm install jstoxml
384
385
                  |----jstoxml@0.2.4
          -Contents Type'
386
387
             application/xml
388
389
        13)객체 --> XML 생성
390
          var entry = {
             profile : {
391
               name: "나훈아",
392
393
               job: "Singer"
             }
394
395
          };
396
          var jstoxml = require('jstoxml');
397
          var xmlContent = jstoxml.toXML(entry, {header:true});
398
        14)XML 요청과 응답
399
          -XML 요청
400
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
          if(req.methdo.toLowerCase() == "post"){
416
             req.on('end', function(){
417
418
               var xmlDoc = libxml.parseXmlString(buffer);
```

```
var title = xmlDoc.get('/movie/title').text();
419
420
                var director = xmlDoc.get('/movie/director').text();
                movieList.push({title:title, director:director});
421
                res.writeHead(200, {'Content-Type': 'application/xml'});
422
423
                res.end(jstoxml.toXML({result:'success'}));
424
             });
425
           }else{
426
             res.writeHead(200, {'Content-Type': 'application/xml'});
427
             var data = {count:movieList.length, data:movieList};
428
             var result = jstoxml.toXML({result:data});
429
              res.end(result);
430
           }
431
        15)Lab
432
433
           <xmlServer.js>
             var http = require('http');
434
435
             var jstoxml = require('jstoxml');
436
             var libxmljs = require("libxmljs");
437
438
             var movieList = [{title:'아바타', director:'제임스 카메론'}];
439
440
             http.createServer(function (req, res) {
441
                   if ( req.method.toLowerCase() == 'post' ){
442
                        var buffer = ";
                        req.on('data', function (chunk) {
443
444
                              buffer += chunk;
445
                        });
446
447
                        reg.on('end', function(){
                              var xmlDoc = libxmljs.parseXml(buffer);
448
449
                              var title = xmlDoc.get('/movie/title').text();
450
                              var director = xmlDoc.get('/movie/director').text();
451
                              movieList.push({title:title, director:director});
                              res.writeHead(200, {'Content-Type' : 'application/xml'});
452
453
                              res.end(jstoxml.toXML({result:'success'}));
454
                        });
                 }else {
455
456
                        res.writeHead(200, {'Content-Type' : 'application/xml'});
457
                        var data = {count:movieList.length, data:movieList};
458
                        var result = jstoxml.toXML({result:data});
459
                        res.end(result);
460
461
              }).listen(3000);
462
463
             In Postman,
464
             http:127.0.0.1:3000 GET
465
              <result>
466
                   <count>1</count>
467
                   <data>
468
                         <title>아바타</title>
469
                         <director>제임스 카메론</director>
470
                   </data>
471
              </result>
472
473
             http://127.0.0.1:3000 POST방식 > Body > raw > XML(application/xml)
474
              <movie>
475
                <title>스타워즈</title>
476
                <director>조지 루카스</director>
477
              </movie>
478
```

```
479
            결과: <result>success</result>
480
            http:127.0.0.1:3000 GET
481
            <result>
482
483
                 <count>2</count>
484
                 <data>
485
                      <title>아바타</title>
                      <director>제임스 카메론</director>
486
487
                      <title>스타워즈</title>
                      <director>조지 루카스</director>
488
489
                 </data>
490
            </result>
491
```