1. 개요
   1)REST(Representational State Transfer)
   2)Roy Fiedling의 박사학위 논문에서 소개. HTTP Protocol 주요 저자
   3)2000년 논문 'Architectural Styles and the Desing of Network-based Software Architectures'
   4)사전정의 : '분산 하이퍼미디어 시스템을 위한 소프트웨어 아키텍처'
   5)네트워크를 이용해서 자원에 접근하고 자원을 다룰 수 있는 아키텍처
   6)90년대 웹 도입과 성장 시대에 웹 기반 소프트웨어의 이상적인 아키텍처 제시

2. REST 아키텍처의 제한 조건
   1)클라이언트/서버 : 클라이언트의 요청과 서버의 응답 기반
   2)상태 없음 : 클라이언트의 상태와 관계없이 요청으로만 응답. 일회성 통신용
   3)캐시 : 클라이언트는 서버의 응답을 캐쉬. 네트워크 비용 절감
   4)계층화 시스템 : 서버는 다양한 형태의 중간 계층을 이용해서 확장할 수 있다. 클라이언트는 서버의 계층에 무관하게 통신을 할 수 있다.
   5)Code on Demand : 리소스를 다룰 수 있는 코드 전송.
   6)인터페이스 일관성 : 시스템 구조를 단순화시키고 작은 단위로 분리해서 독립적으로 개선하고 확장할 수 있다.

3. REST 인터페이스를 작성할 때 가이드
   1)자원의 식별 : 개별 리소스를 식별할 수 있어야 한다. 예.URL, URI
   2)메시지를 통한 리소스의 조작 : 메시지에 작성한 리소스를 다루는 정보를 이용해서 리소스를 얻어오거나 리소스를 조작한다.
   3)자기서술적 메시지 : 요청과 응답 메시지에는 메세지를 처리할 수있는 정보를 포함한다.
   4)어플리케이션의 상태에 대한 엔진으로서 하이퍼 미디어 : 하이퍼링크를 이용해서 유기적으로 연결된 문서로 쉽고 간단하게 정보를 접할 수 있다.

4. 서비스 설계 가이드
   1)HTTP Protocol 사용
   2)인터페이스 설계
      -리소스 접근하는 인터페이스(URL)
      -리소스를 다루기 위한 HTTP 메소드
      -간단하고 직관적인 API
      -리소스를 다루는 행위는 HTTP 메소드 사용
         -GET, POST, PUT, DELETE
      -API 버전
         --api.mystore.com/v1/items
      -명사형 단어 사용 권장
      -목록 형태의 리소스를 다루는 API는 복수형 명사
      -목록에서 특정 조건으로 필터링 : 쿼리 문자열
         --ap.mystore.com/v1/items?year=2015&category=food

   3)요청과 응답 메시지 설계
      -요청 메시지 구조
      -응답 메시지 구조(JSON)
         --프로퍼티 이름
            ---의미를 충분히 반영
            ---camelCase
            ---예약어 사용불가
         --데이터와 보조 데이터 활용
         --에러 발생시 에러 정보 제공

   4)응답 메시지 예
      {
            "total" : 100,
            "query" : "food",
            "data" : [
               {"itemID" : "0001", "name" : "item1"},
               {"itemID" : "0002", "name" : "item2"},
            ]

```
    }
    -에러 정보
      {
          "error" : { code:400, "message" : "필수 입력값 없음"}
      }

  5)메시지 구조의 예 : 페이지 번호
    api.mystore.com/v1/items?page=1
    -메시지 구조
      {
          "totalPage" : 10,
          "currentPage" : 0,
          "itemsPerPage" : 10,
          "data" : [
              //데이터
          ]
      }

  6)메시지 구조의 예 : 리소스 인덱스
    http://mystore.com/items/start=20
    -메시지 구조
      {
          "totalItems" : 100,
          "itemsPerPage" : 10,
          "startIndex" : 20,
          "paging" : {
            "prev" : "http://mystore.com/items/start=10",
            "next" : "http://mystore.com/items/start=30"
          },
          "data" : [
              //데이터
          ]
      }


4. 실습 : 영화 정보 서비스
  1)개요
    -컨텐츠 : 영화 정보
    -기본 URI : http://api.movie.com/movies
    -미디어 타입 : JSON
    -HTTP 메소드 : GET, PUT, POST, DELETE

  2)영화 목록 보기 API 설계
    -URL : http://api.movie.com/movies
    -메소드 : GET
    -파라미터 : 없음
    -Response : JSON
    -예
    {
        "count" : 2,
        "data" : [
          {
              "id" : 0,
              "title" : "아바타"
          },
          {
              "id" : 1,
              "title" : "스타워즈"
          }
        ]
```

```
}

3)영화 상세 보기 API 설계
  -URL : http://api.movie.com/movies/0
  -메소드 : GET
  -파라미터 : 없음
  -Response : JSON
  -예
  {
      "id" : 0,
      "title" : "아바타",
      "director" : "제임스 카메론",
      "year" : 2009,
      "synopsis" : "인류의 마지막 희망, 행성 판도라! 이곳을 정복하기 위한 '아바타 프로젝트'가 시작된다!"
  }

4)영화 정보 추가 API 설계
  -URL : http://api.movie.com/movies
  -메소드 : POST
  -Response / Request : JSON
  -요청 메시지 예
  {
      "title" : "스타워즈",
      "director" : "조지 루카스",
      "year" : 2016,
      "synopsis" : "캐어난 포스"
  }
  -응답 메시지 예
  { "message" : "성공", "id" : "3"}

5)영화 정보 삭제 API 설계
  -URL : http://api.movie.com/movies/0
  -메소드 : DELETE
  -파라미터 : 없음
  -Response  : JSON
  -예
  {
      "message" : "success",
      "item" : {
        "id" : 0,
        "title" : "아바타",
        "director" : "제임스 카메론",
        "year" : 2009,
        "synopsis" : "인류의 마지막 희망, 행성 판도라! 이곳을 정복하기 위한 '아바타 프로젝트'가 시작된다!"
       }
  }

6)영화 정보 인터페이스 설계
  -GET, POST만 사용 가능한 환경
    --수정, 삭제, 추가 행위를 URL로 전달
    --movie.com/movies/add
    --movie.com/movies/edit
    --movie.com/movies/?id=starwars&operation=delete

7)라우팅 구현
  -요청 메소드와 URL 분석
    var server = http.createServer(function(req, res){
      switch(req.method){
```

```
            case 'get' :
               //GET 요청 처리 : 영화 목록 보기, 영화 상세 정보 보기
               handleGetRequest(req, res);
               return;
            case 'post' :
               //POST 요청 처리 : 영화 정보 추가
               handlePostRequest(req, res);
               return;
            case 'put' :
               //삭제처리 등
         }
      });

8)영화 정보 제공
   -영화 목록 요청 - 응답
      function handleGetRequest(request, response){
         var url = request.url;
         if(url == '/movies'){
            //영화 목록 생성
            var list = [];
            //목록 정보와 메타 데이터 준비
            var result = {count:list.length, data:list}
            //JSON 형태로 응답
            res.end(JSON.stringify(result);
         }else{
            //영화 상세 정보 제공
         }
      }

   -영화 상세 정보 요청에 대한 응답
      funciton handleGetRequest(request, response){
         var url = request.url;
         if(url == '/movies'){
            //영화 목록 응답
         }else{
            //영화 상세 정보 제공
            //요청 정보 분석 : URL에서 id 찾기
            var id = url.split('/')[2];
            //상세 정보 찾기, 응답 생성
            var movie = {title : 'avata', ...};
            //JSON으로 응답 메시지 전송
            res.end(JSON.stringify(movie));
         }
      }

9)Lab
   <movieData.json>
      [
         {
               "id":1,
               "title":"아바타",
               "director":"제임스 카메론",
               "year":2009,
               "synopsis":"인류의 마지막 희망, 행성 판도라! 이 곳을 정복하기 위한 '아바타
               프로젝트'가 시작된다! 가까운 미래, 지구는 에너지 고갈 문제를 해결하기 위해 머나먼 행성
               판도라에서 대체 자원을 채굴하기 시작한다. 하지만 판도라의 독성을 지닌 대기로 인해 자원
               획득에 어려움을 겪게 된 인류는 판도라의 토착민 '나비(Na'vi)'의 외형에 인간의 의식을
               주입, 원격 조종이 가능한 새로운 생명체 '아바타'를 탄생시키는 프로그램을 개발한다."
         },
         {
```

```
            "id":2,
            "title":"스타워즈",
            "director":"조지 루카스",
            "year":1977,
            "synopsis":"평화롭던 은하계에서 타킨총독(피터 커쉥)이 왕정에 저항하여 제국을
            일으킨다. 타킨은 우주정거장인 죽음의 별을 완성하고 은하계의 작은 나라들을 점령하고자
            한다. 그 정보를 입수한 반란군은 레아 공주(캐리 피셔)를 보내 죽음의 별 설계도를
            입수하려고 하지만 공주는 타킨에게 잡힌다. 대신 공주는 도움을 요청하기 위해 제다이 기사
            케노비(알렉 기네스)에게 로봇을 보낸다."
        },
        {
            "id":3,
            "title":"인터스텔라",
            "director":"크리스토퍼 놀란",
            "year":2014,
            "synopsis":"세계 각국의 정부와 경제가 완전히 붕괴된 미래가 다가온다. 지난 20세기에
            범한 잘못이 전 세계적인 식량 부족을 불러왔고, NASA도 해체되었다. 이때 시공간에
            불가사의한 틈이 열리고, 남은 자들에게는 이 곳을 탐험해 인류를 구해야 하는 임무가
            지워진다. 사랑하는 가족들을 뒤로 한 채 인류라는 더 큰 가족을 위해, 그들은 이제 희망을
            찾아 우주로 간다. 그리고 우린 찾을 것이다. 늘 그랬듯이…"
        }
    ]

<movieserver.js>
    var http = require('http');
    var fs = require('fs');

    var data = fs.readFileSync('./movieData.json');
    var movieList = JSON.parse(data);
    var idx = movieList[movieList.length - 1].id;

    var server = http.createServer(function (req, res) {
        var method = req.method.toLowerCase();
        switch (method) {
            case 'get':
                // 영화 목록 보기 : /movies
                // 영화 상세 정보 보기 : /movies/1, /movies/2
                handleGetRequest(req, res);
                break;
            case 'post':
                // 영화 정보 추가 : /movies
                handlePostRequest(req, res);
                break;
            case 'put':
                // 영화 정보 수정 : /movies/1, /movies/2
                handlePutRequest(req, res);
                break;
            case 'delete':
                // 영화 정보 삭제 : /movies, /moives/1
                handleDeleteRequest(req, res);
                break;
            default:
                res.statusCode = 404;
                res.end('잘못된 요청입니다.');
                break;
        }
    });
    server.listen(80);

    function handleGetRequest(req, res) {
```

```
282             var url = req.url;
283             if (url == '/movies') {
284                 // 영화 목록 만들기
285                 var list = [];
286                 for (var i = 0; i < movieList.length; i++) {
287                         var movie = movieList[i];
288                         list.push({ id: movie.id, title: movie.title });
289                 }
290
291                 // 항목 갯수와 영화 목록 정보
292                 var result = {
293                         count: list.length,
294                         data: list
295                 }
296
297                 res.writeHead(200, { 'Content-Type': 'application/json;charset=utf-8' });
298                 res.end(JSON.stringify(result));
299             }else {
300                 // 영화 상세 정보를 위한 id 추출 /movies/0
301                 var id = url.split('/')[2];
302                 var movie = null;
303                 // 영화 데이터베이스에서 영화 검색
304                 for (var i = 0; i < movieList.length; i++) {
305                     var item = movieList[i];
306                     if (id == item.id) {
307                         movie = item;
308                         break;
309                     }
310                 }
311                 // 검색된 영화 정보 제공
312                 if (movie) {
313                     res.writeHead(200, { 'Content-Type': 'application/json;charset=utf-8'});
314                     res.end(JSON.stringify(movie));
315                 }else {
316                     // 영화 정보가 없는 경우
317                     res.writeHead(404, { 'Content-Type': 'application/json;charset=utf-8' });
318                     var message = {
319                         error : {
320                             code : 404,
321                             message : '영화 정보가 없습니다.'
322                         }
323                     }
324                     res.end(JSON.stringify(message));
325                 }
326             }
327         }
328
329         function handlePostRequest(req, res) {
330             var buffer = '';
331             req.on('data', function (chunk) {
332                 buffer += chunk;
333             });
334
335             req.on('end', function () {
336                 var parsed = JSON.parse(buffer);
337                 var titleData = parsed.title;
338                 var directorData = parsed.director;
339                 var yearData = parsed.year;
340                 var synopsisData = parsed.synopsis;
341                 var idData = idx + 1;
```

```
342             movieList.push({id:idData, title:titleData, director:directorData,
                year:yearData, sysnopsis:synopsisData});
343
344             fs.writeFileSync('./movieData.json', JSON.stringify(movieList));
345             res.writeHead(200, {'Content-Type':'application/json'});
346             res.end(JSON.stringify({"result" : "success"}));
347         });
348     }
349
350     function handlePutRequest(req, res) {
351         var buffer = '';
352         var id = req.url.split('/')[2];
353         if(!id){
354             res.writeHead(404, { 'Content-Type': 'application/json;charset=utf-8' });
355             var message = {
356                 error : {
357                     code : 500,
358                     message : 'Internal Server Error'
359                 }
360             }
361             res.end(JSON.stringify(message));
362         }else{
363             req.on('data', function (chunk) {
364                 buffer += chunk;
365             });
366             req.on('end', function () {
367                 var parsed = JSON.parse(buffer);
368                 var obj = {
369                     id : id,
370                     title : parsed.title,
371                     director : parsed.director,
372                     year : parsed.year,
373                     synopsis : parsed.synopsis
374                 }
375                 for(var i = 0 ; i < movieList.length ; i++){
376                     var item = movieList[i];
377                     if(item.id == id) {
378                         movieList.splice(i, 1, obj);
379                         break;
380                     }
381                 }
382                 fs.writeFileSync('./movieData.json', JSON.stringify(movieList));
383                 res.writeHead(200, {'Content-Type':'application/json'});
384                 res.end(JSON.stringify({"result" : "success"}));
385             });
386         }
387     }
388
389     function handleDeleteRequest(req, res) {
390         var id = req.url.split('/')[2];
391         for(var i = 0 ; i < movieList.length ; i++){
392             var item = movieList[i];
393             if(item.id == id) {
394                 movieList.splice(i, 1);
395                 break;
396             }
397         }
398         fs.writeFileSync('./movieData.json', JSON.stringify(movieList));
399         res.writeHead(200, {'Content-Type':'application/json'});
400         res.end(JSON.stringify({"result" : "success"}));
```

```
        }

        ------------------------------------------------
    10)GET
      -GET http://127.0.0.1/movies
      -결과 :
        {
          "count": 3,
          "data": [
            {
              "id": 1,
              "title": "아바타"
            },
            {
              "id": 2,
              "title": "스타워즈"
            },
            {
              "id": 3,
              "title": "인터스텔라"
            }
          ]
        }

    11)GET
      -GET http://127.0.0.1/movies/3
      -결과 :
        {
          "id": 3,
          "title": "인터스텔라",
          "director": "크리스토퍼 놀란",
          "year": 2014,
          "synopsis": "세계 각국의 정부와 경제가 완전히 붕괴된 미래가 다가온다. 지난 20세기에 범한 잘못이 전 세계적인 식량 부족을 불러왔고, NASA도 해체되었다. 이때 시공간에 불가사의한 틈이 열리고, 남은 자들에게는 이 곳을 탐험해 인류를 구해야 하는 임무가 지워진다. 사랑하는 가족들을 뒤로 한 채 인류라는 더 큰 가족을 위해, 그들은 이제 희망을 찾아 우주로 간다. 그리고 우린 찾을 것이다. 늘 그랬듯이…"
        }

    12)POST
      -POST  http://127.0.0.1/
        POST > Body > raw > JSON(application/json)
        {
          "title":"밀정",
          "director":"김지운",
          "year":2016,
          "synopsis":"1920년대 일제강점기. 조선인 출신 일본경찰 이정출(송강호)은 무장독립운동 단체 의열단의 뒤를 캐라는 특명으로 의열단의 리더 김우진(공유)에게 접근하고, 한 시대의 양 극단에 서 있는 두 사람은 서로의 정체와 의도를 알면서도 속내를 감춘 채 가까워진다. 출처를 알 수 없는 정보가 쌍방간에 새어나가고 누가 밀정인지 알 수 없는 가운데, 의열단은 일제의 주요 시설을 파괴할 폭탄을 경성으로 들여오기 위해, 그리고 일본 경찰은 그들을 쫓아 모두 상해에 모인다..."
        }
      -결과 :
        {
          "result": "success"
        }
      -GET http://127.0.0.1/movies
      -결과 :
        {
          "count": 4,
```

```
        "data": [
          {
            "id": 0,
            "title": "아바타"
          },
          {
            "id": 1,
            "title": "스타워즈"
          },
          {
            "id": 2,
            "title": "인터스텔라"
          },
          {
            "id": 4,
            "title": "밀정"
          }
        ]
      }
```


13)DELETE

-DELETE http://127.0.0.1/movies/2

-결과 :


```
{
  "result": "success"
}
```


14)PUT

-GET http://127.0.0.1/movies/1

-결과 :


```
{
  "id":1,
  "title":"스타워즈",
  "director":"조지 루카스",
  "year":1977,
  "synopsis":"평화롭던 은하계에서 타킨총독(피터 커쉥)이 왕정에 저항하여 제국을 일으킨다. 타킨은 우주정거장인 죽음의 별을 완성하고 은하계의 작은 나라들을 점령하고자 한다. 그 정보를 입수한 반란군은 레아 공주(캐리 피셔)를 보내 죽음의 별 설계도를 입수하려고 하지만 공주는 타킨에게 잡힌다. 대신 공주는 도움을 요청하기 위해 제다이 기사 케노비(알렉 기네스)에게 로봇을 보낸다."
}
```


-PUT http://127.0.0.1/movies/1


```
{
  "title":"스타워즈8",
  "director":"조지 루카스",
  "year":2017,
  "synopsis":"라스트 제다이"
}
```


결과 :


```
{
  "result": "success"
}
```