

# Network with REST Server

The slide features two solid horizontal bars at the bottom. The top bar is red and the bottom bar is orange, both spanning the full width of the slide.

- RESTful 서비스
- 리소스 다루는 행위(HTTP 메소드)
  - 리소스 추가 : Post
  - 리소스 보기 : Get
  - 리소스 수정 : Put
  - 리소스 삭제 : Delete

- 리소스 접근
- 리소스 간 관계/계층 구조
  - 영화 목록 : /movies
  - 영화 상세 : /movies/avata

- 리소스 접근 표현 + 리소스 다루는 행위
  - 영화 목록보기 : Get /movies
  - 영화 상세 정보 보기 : Get /movies/avata
  - 영화 정보 추가 : Post /movies
  - 영화 상세 정보 삭제 : Delete /movies/titanic

# 서버 동작시키기

---

- Node.js 설치
- npm install
- node server.js
- 서버 주소!

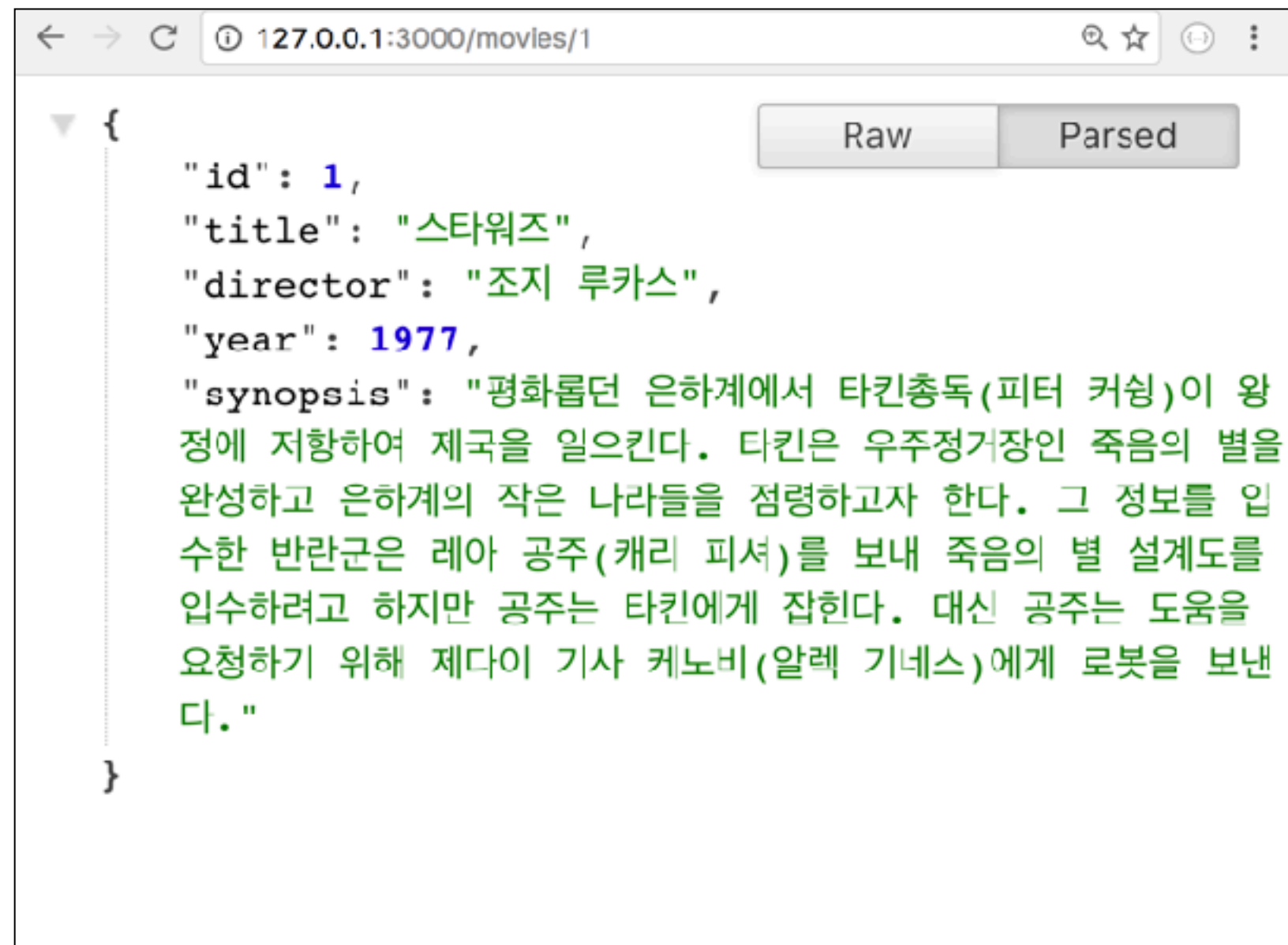
# 영화 목록보기

- 영화 목록보기
  - Get SERVER/movies



```
127.0.0.1:3000/movies
{
  "count": 3,
  "data": [
    {
      "title": "아바타",
      "id": 0
    },
    {
      "title": "스타워즈",
      "id": 1
    },
    {
      "title": "인터스텔라",
      "id": 2
    }
  ]
}
```

- 영화 상세보기
  - Get /movies/1



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:3000/movies/1`. The browser is displaying a JSON response from a REST client. The response is a single object with the following fields:

- `"id": 1`
- `"title": "스타워즈"`
- `"director": "조지 루카스"`
- `"year": 1977`
- `"synopsis": "평화롭던 은하계에서 타킨총독(피터 커싱)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."`

The JSON is displayed in a 'Parsed' view, with a 'Raw' button visible. The text is color-coded: numbers are blue, strings are green, and the synopsis is a longer green string.

- POST /movies
- POST를 이용한 정보 전달
- HTTP 메시지 바디 사용
- 메시지 바디 인코딩 : urlencoded, json



# HTTP 메시지

요청 라인	GET / HTTP/1.1
헤더 필드	Host: 127.0.0.1:3000 Connection: keep-alive Cache-Control: max-age=0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 Accept-Encoding: gzip, deflate, sdch Accept-Language: ko-KR,ko;q=0.8,en-US;q=0.6,en;q=0.4
CRLF	
메세지 바디	

- 요청 메시지 구조 : URLEncoded

- 헤더

- Content-Type : application/x-www-form-urlencoded

- 바디

- title=TITLE&director=DIRECTOR

- 요청 메시지 구조 : JSON

- 헤더

- Content-Type : application/json

- 바디

- { "title": "영화 제목", "director": "영화 감독" }

- POST 요청 메시지

```
var params : [String : String] = [  
    "title": "MOVIE-TITLE",  
    "director" : "MOVIE-DIRECTOR",  
    "year" : MOVIE-YEAR,  
    "synopsis" = "MOVIE-SYNOPSIS"  
]
```

```
Alamofire.request(ServerAddress, method: .post, parameters: params, encoding:  
URLEncoding.httpBody, headers: nil)
```

- 인코딩

URLEncoding.queryString : URL의 쿼리 문자열

URLEncoding.httpBody : URLEncoded

JSONEncoding.default : JSON

- 응답 분석

- 요청 성공 여부 : 응답 객체 여부
- 응답 성공 여부 : 상태 코드로 체크

```
Alamofire.request(ServerAddress, method: .post, parameters: params,
    encoding: URLEncoding.httpBody, headers: nil)
    .responseJSON { (result : DataResponse<Any>) in

    guard let response = result.response else {
        // "서버가 응답하지 않음"
        return
    }
    // 응답 코드 체크
    let code = response.statusCode
    guard code >= 200 && code < 300 else {
        // 정상 처리 안됨
        return
    }
    print("Success")
    self.dismiss(animated: true, completion: nil)
}
```