**# CORS : Cross Origin Resource Sharing**

CORS는 에러가 아니었다.

CORS란?

: **보안상**의 이유로 *하나의 web program은 하나의 resource domain만 소유*하게 되어 있다. BUT, 대규모의 프로젝트가 많아지면서 하나의 도메인이 아닌 여러 도메인에서 데이터를 요청하는 작업이 늘어나게 됨. 그래서 생겨난 것이 **CORS Mechanism**이다!

예를 들면,

<http://test-a:8080/> 이 main domain이라고 하자.

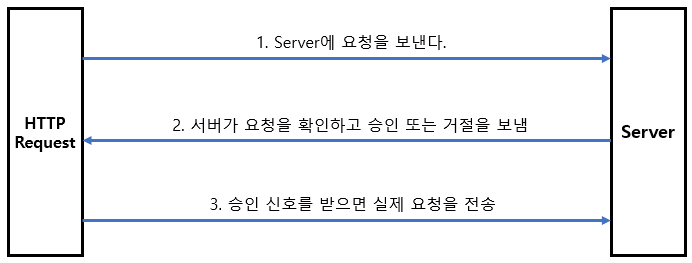
여기서 <http://test-b:8080/> 에 있는 resource를 요청할 때 CORS 경고가 발생

해결 방법은?

**Http Request Header**에 있는 **Access-Control-Allow-Origin**에 접속을 허용할 도메인을 적어주던가, 또는 모두 허용하고 싶다면 **\*** 를 넣어준다.

추가 공부:

Http Request와 서버의 구조에 대해서 간단하게 알아보자



여기서 승인 신호가 status 신호 **200** 이다.

그런데 더 편하게 해결하는 방법을 Spring에서 제공한다.

@CrossOrigin annotation을 사용하면 된다!

**# Rest Controller**

Spring project에서 return 하는 값이 path가 아닐 때! @responseBody annotation을 붙여주지만 이러한 작업을 한번에 끝내주는게 @RestController다.

**# JSON Data 처리 (json-simple library)**

JS에서 Java로 데이터를 보낼 때 **encodeURI()** method를 사용한다.

Ex) encodeURI(JSON.stringify(“DATA”));

안 그러면 에러가… JSON을 URL에 포함시켜서 보내면서 특수문자 때문에 에러가 나는 것 같다..

Java에서 JSON 데이터를 받을 때

1. JSONParser를 이용해서 Object를 만들고
2. 그 Object를 JSONObject에 대입하면서 캐스팅

Ex) JSONObject j\_obj = new JSONParser.parse(“DATA”);