Trello 개발

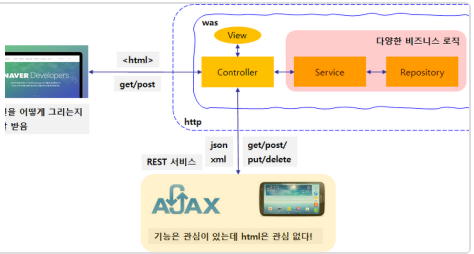
7/6 ~ 7/10 1주차

출처

01.Rest

REST란 Representational State Transfer 의 약자로 하나의 URI는 하나의 고유한 리소스와 연결되며 이 리소스를 GET/POST/PUT/DELETE 등 HTTP 메서드로 제어하자는 개념이다. Representational은 웹 상의

https://goodteacher.tistory.com/264?category=828440

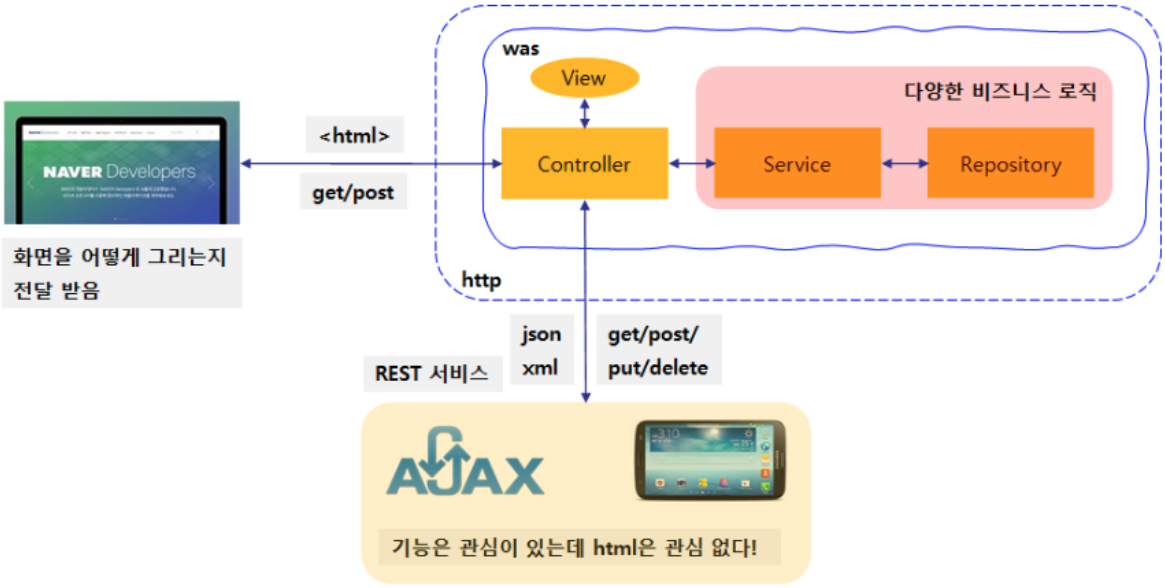


▼ Spring & Spring Boot에 대한 정리

Rest란 Representational(웹 상의 상태를 표현할 수 있는 어떤 자원) State(자원의 상태) Transfer (데이터의 전송

하나의 URI는 하나의 고유한 리소스와 연결되며 이 리소스를 GET/POST/PUT/DELETE 등 HTTP 메서드로 제어하는 개념이다.

특히 서버에 접근하는 클라이언트의 종류가 단순히 브라우저를 넘어 스마트폰, 다른 서비스 등으로 다양해지면서 화면에 대한 관심은 없고 데이터, 비즈니스 로직에만 관심있는 경우가 많고 이때 사용되는 것이 REST이다.



rest 형태로 사용자가 원하는 기능을 제공하는 것을 rest api라고 한다. 또한, rest 방식으로 서비스를 제공하는 것을 restful 하다라고 말한다.

다음 예는 http://example.com/api/orders라는 라는 URI를 여러 HTTP method로 접근했을 때의 의미이다.

Http Method	동작	예
GET	전체 리소스에 대한 정보 획득	http://example.com/api/orders : 모든 주문 내역 조회
GET	특정 조건에 맞는 모든 정보 획득	http://example.com/api/orders?from=100 : 100번 주문부터 모든 주문 내역 조회
GET	특정 리소스에 대한 정보 획득	http://example.com/api/orders/123 : 123번 주문에 대한 상세 내역 조회
POST	새로운 리소스 저장	http://example.com/api/orders : 새로운 주문 생성. 파라미터는 request body로 전달
PUT	기존 리소스 업데이트	http://example.com/api/orders/123 : 123번 주문에 대한 수정. 파라미터는 request body로 전달
DELETE	기존 리소스 삭제	http://example.com/api/orders/123 : 123번 주문 삭제

Spring에서의 REST

→ 스프링은 REST를 편리하게 사용할 수 있도록 다양한 애너테이션들을 제공한다.

@PathVariable

Rest는 URI template을 구성해서 리소스를 가르킨다. 즉 orderNp와 같이 매번 바뀌어야 다른 값을 조회할 수 있을 것이다. 이런 경우 @RequestMaipping의 path는 /api/order/{orderNo}와 같은 형태로 구성한다.

@PathVariable은 말 그대로 Path 즉 경로상에 변수(path variable)를 참조할 때 사용된다.

```
package com.eshome.rest.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
public class HtmlService {

    @GetMapping("/api/{aptNm}/{dong}")
    public String pathVariableTest(@PathVariable String aptNm, @PathVariable Integer dong) {
        return String.format("전달 받은 값은: %s %s", aptNm, dong);
    }
}
```

@GetMapping의 path에 aptNm, dong이 path variable로 선언되어있고 handler method에서 @PathVariable을 이용해서 참조하는 것을 볼 수 있다. dong의 경우에서 알 수 있듯이 @PathVariable도 @RequestParam 처럼 **자동 형변환을 지원**한다.

http://local:8080/api/개나리아파트/1104

페이지가 출력되는 것이 아닌 Rest는 <html>이 아니라 데이터에만 관심이 있다는 것을 기억해두기.

@ResponseBody

응답 즉 response를 바로 <body>에 써버리는 용도이다.

즉 사용되면 return된 문자열이 view controller를 거치지 않고 곧바로 클라이언트로 전달된다.

```java
@GetMapping("/api/{aptNm}/{dong}")
@ResponseBody
public String pathVariableTest(@PathVariable String aptNm, @PathVariable Integer dong) {
    return String.format("전달 받은 값은: %s %s", aptNm, dong);
}
```

Controller클래스에 @ResponseBody를 선언하면 해당 Controller에 있는 모든 handler method들은 @ResponseBody로 간주된다.

@ResController

Rest서비스가 필요할 때마다 메서드에 @ResponseBody를 선언하기는 번거롭다. 그래서 일반적으로 Rest서비스를 위한 Controller와 일반 서비스를 위한 Controller로 나눠서 사용하는 경우가 많다.

이 때 @Controller와 @ResponseBody를 두개 쓰기 번거롭다면 @RestController를 사용하자.

@RestControllersms는 이 둘을 포함한다.

@RestController를 사용하면 보다 편리하게 Rest 서비스 구현이 가능하다.

```
package com.eshome.rest.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
public class RestServiceController {

    @GetMapping("/always")
    public String alwaysRestService() {
        return "언제나 rest";
    }
}
```

```
    # 문자열은 물론 DTO나 Collection등도 반환 가능하다
    # 이들은 MessageConverter에 의해 JSON 또는 XML 형태의 문자열로 변환해서 리턴된다.

# DTO
@GetMapping("/city")
public City city() {
  return new City("서울", "KO");
}

# List
@GetMapping("/list")
public List<String> list() {
  List<String> fruits = Arrays.asList<"banana", "apple");
  return fruits;
}
```

@RequestBody

@ResponseBody가 응답을 처리하기 위한 애너테이션이었다면 @RequestBody는 request의 body를 통해 JSON 형태로 전달되는 데이터를 객체로 변환하기 위해서 사용된다.

> form의 파라미터 형태로 전달되지 않고 JSON 형태로 전달되는 형태를 처리한다.

```
@PutMapping("/city")
public Boolean insertCity(@RequestBody City city) {
    log.trace("도시 저장: {}", city);
    return true;
}
```

간단히 JSON 요청을 만들기는 어려우니 실행은 생략한다.

Rest 테스트를 위한 클라이언트로 Talend API Tester, Postman 같은 게 있다.

ResponseEntity

앞 선 예에서는 단순히 조회된 데이터를 클라이언트로 전송하도록만 처리되고 있다.

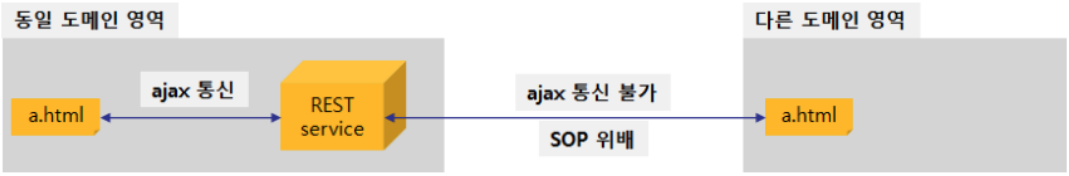
그런데 많은 경우 조회된 데이터와 함께 HTTP 상태 코드는 물론 요청의 성공 여부, 실패시 오류 내용등 다양한 내용을 전달할 필요가 있다. 이런경우에 사용하는 것이 ResponseEntity!

```java
@GetMapping("/countries")
public ResponseEntity<Map<String, Object>> allCountries(@RequestParam Integer per,
                                                        @RequestParam Integer page) {
    PageHelper.startPage(page, per);
    ResponseEntity<Map<String, Object>> entity = null;
    try {
        Page<Country> data = repo.selectAll();
        entity = handleSuccess(data);
    } catch (RuntimeException e) {
        entity = handleException(e);
    }
    return entity;
}

private ResponseEntity<Map<String, Object>> handleSuccess(Object data) {
    Map<String, Object> resultMap = new HashMap<>();
    resultMap.put("success", true);
    resultMap.put("data", data);
    return new ResponseEntity<Map<String, Object>>(resultMap, HttpStatus.OK);
}

private ResponseEntity<Map<String, Object>> handleException(Exception e) {
    Map<String, Object> resultMap = new HashMap<>();
    resultMap.put("success", false);
    resultMap.put("data", e.getMessage());
    return new ResponseEntity<Map<String, Object>>(resultMap, HttpStatus.INTERNAL_SERVER_ERROR);
}
```

이제 요청 성공 시 Map을 통해 Country 정보(data) 뿐 아니라 성공 여부(success) 가 HttpStatus.OK 즉 200 상태 코드와 함께 전달된다. 또한 실패 시에는 성공 여부와 오류 원인(data)와 500 상태 코드로 서버의 오류 상태를 전달한다.

SOP와 대책

SOP(Same Origin Policy: 동일 근원 정책)은 JavaScript에서 Ajax를 이용해 서버 자원을 호출할 때 동일한 도메인으로만 데이터 전송을 허용한다는 점이다.



이 경우 아래와 같은 오류가 발생한다.

```
XMLHttpRequest cannot load http://127.0.0.1:8080/~~~.
No 'Access-Control-Allow-Origin' header is present on the requested resource.
Origin 'http://localhost:8080' is therefore not allowed access.
```

이 문제를 처리하기 위해서 서버에서 명시적으로 다른 도메인에서의 접근을 허용해줘야 한다. 이때 @CrossOrigin 애너테이션을 사용한다.

```
@RestController
@RequestMapping("/api")
@Slf4j
@CrossOrigin("*")
public class RestServiceController {...}
```

@CrossOrigin의 value에는 접근을 허용할 origin들을 배열 형태로 적어주는데 모든 접근을 허용하려면 *를 사용한다.

Axios와 관련된 내용일 수 있다.