스프링 CH16
JSON

정승혜

JSON(javaScript Object Notation)

- (Key-value) 쌍
- 웹에서 Ajax를 이용해서 서버 API를 호출할 때
  이 API는 요청에 대한 응답으로 HTML 대신 JSON/XML을 사용함

Jackson 라이브러리

- com.fasterxml.jackson.core
- com.fasterxml.jackson.datatype

Ajax(Asynchronous Javascript And Xml)

- JavaScript를 사용하여 비동기 통신, 클라이언트/서버 간 JSON/ XML 데이터를 주고받는 기술
- 브라우저가 가지고 있는 XMLHttpRequest 객체를 이용해서
  페이지를 새로 고침 하지 않고도 페이지의 일부만을 위해 데이터를 로드

장점	단점
1. 웹페이지 속도 향상 2. 서버의 처리가 완료 될때까지 기다리지 않고 처리 가능 3. 서버에서 Data만 전송하면 되므로 전체적인 코딩의 양이 줄어든다 4. 기존웹에서는 불가능했던 다양한 UI를 제공	1. 히스토리 관리가 안됨 (보안에 신경써야됨) 2. 연속으로 데이터를 요청하면 서버 부하 증가 3. XMLHttpRequest를 통해 통신을 하는 경우 사용자에게 아무런 진행 정보가 보이지 않음 => 요청이 완료되지 않았는데 사용자가 페이지를 떠나거나 / 오작동 발생 가능

JSON in Spring

@RestController 어노테이션

- 스프링 5가 나오면서 생긴 기능 (이전 : @Controller, @ResponseBody)
- GetMapping가 붙은 메서드가 리턴한 객체를
  클래스 path에 Jackson이 존재하면 JSON 형태로 응답 데이터로 전송

@JsonIgnore

- 응답 결과로 제공되어야 할 JSON 목록에서
  제외되어야 할 요소에 대한 설정

ex) password

```
@JsonIgnore
Private String password;
```

JSON in Spring – 객체 to JSON

@JsonFormat

- 전송 데이터(LocalDataTime) 응답 데이터(JSON)일 때
- 자바 객체의 요소 중 시간을 나타내는 객체는 LocalDateTIme인데,
  Json값은 배열이나 숫자일 때, Jackson을 통해 변환 가능

```
// ISO-8601 형식으로 변환
@JsonFormat(shape= Shape.String)
private LocalDateTime registerDateTime //원하는 형식
@JsonFormat(pattern = "yyyyMMddHHmmss")
private LocalDateTime registerDateTime;
```

JSON in Spring - 객체 to JSON

```
@JsonFormat(shape= Shape.String)
private LocalDateTime registerDateTime
```

- 그러나 변환할 모든 대상에 어노테이션을 붙이는 것은 힘듦

⇒날짜 타입에 해당하는 모든 대상에 동일한 변환 규칙을 적용하자

⇒ 스프링 MVC 설정 변경

```
@Override
public void extendMessageConverters(
                                        List<HttpMessageConverter<?>> converters) {
        ObjectMapper objectMapper = Jackson2ObjectMapperBuilder
        .json()
        .featuresToDisable( SerializationFeature.WRITE_DATES_AS_TIMESTAMPS)
        .build();
        converters.add(0, new
        MappingJackson2HttpMessageConverter(objectMapper)); } }
```

JSON in Spring – JSON to 객체

@RequestBody

- POST나 PUT 방식을 사용할때 쿼리 문자열 형식이 아니라 JSON 형식의 데이터를 요청 데이터로 전송할 수 있다
- JSON 형식으로 전송된 요청 데이터를 커맨드 객체로 전달 받으려면 스프링 MVC에 extendMessageConverters() 메서드를 통해 등록

-> Configuration 파일에 extendMessageConverters()는 여러 개 등록해도 상관없으나, 원래 있는 Converter를 대체하므로 사용시 유의

-> 메서드에 등록한 포맷보다 해당 객체에 @JsonFormat으로 등록한 포맷의 형태가 적용시 우선함

JSON in Spring – JSON to 객체

```
@Override
public void extendMessageConverters(
                                        List<HttpMessageConverter<?>> converters) {
        DateTimeFormatter formatter =
                DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        ObjectMapper objectMapper = Jackson2ObjectMapperBuilder
        .json()
        .featuresToEnable(SerializationFeature.INDENT_OUTPUT)
        .deserializerByType(LocalDateTime.class,
                new LocalDateTimeDeserializer(formatter))
        .simpleDateFormat("yyyyMMddHHmmss")
        .build();
converters.add(0, MappingJackson2HttpMessageConverter(objectMapper));
}
```

JSON in Spring – 커맨드 객체 검증

@Valid

```
@PostMapping("/api/members")
public void newMember(
        @RequestBody @Valid RegisterRequest regReq,
        HttpServletResponse response ) throws IOException {
 ... }
```

JSON in Spring – 커맨드 객체 검증

따로 Validator를 두는 경우 – 직접 상태 코드 처리 필요함

```
@PostMapping("/api/members")
public void newMember(
        @RequestBody RegisterRequest regReq, Errors errors,
        HttpServletResponse response) throws IOException {
        try {
                new RegisterReqeustValidator().validate(regReq, errors);
        if ( errors.hasErrors()){
                response.sendError(HttpServletResponse.SC_BAD_REQUEST);
                return;
        }
        ... } catch ( DuplicateMemberException dupEx ) {
                response.sendError(HttpServletResponse.SC_CONFLICT);
        }
}
```

JSON in Spring – HTTP 상태 코드와 JSON

원래 Http 상태 코드 지정방법 :

HttpServletResponse의 setStatus(), sendError() 메서드를 사용

문제점 :

ex) 404 응답을 하면 JSON 형식이 아니라 서버가 기본으로 제공하는 HTML을 응답 결과로 제공함

⇒ 이때, 프로그램 입장에서 JSON/HTML 응답을 모두 처리하는 것은 부담

Error에 대해 HTML 대신에 JSON 형식의 응답 데이터를 전송해야 API 호출 프로그램이 일관된 방법으로 응답을 처리 할 수 있음

=> ResponseEntity

JSON in Spring – HTTP 상태 코드와 JSON

```
@GetMapping("/api/members/{id}")
public ResponseEntity<Object> member(@PathVariable Long id) {
        Member member = memberDao.selectById(id);
        if (member == null) {
                return ResponseEntity
                .status(HttpStatus.NOT_FOUND)
                .body(new ErrorResponse("no member"));
        }
        return ResponseEntity.ok(member);
        // return ResponseEntity.status(HttpStatus.OK).body(member);
}
```

- 리턴 타입이 ResponseEntity이면

  ResponseEntity의 body로 지정한 객체를 사용해서 변환처리를 함

- 객체를 JSON으로 변환

JSON in Spring – ResponseEntity와 예외처리

- 한 메서드에서 정상 응답/에러 응답을 ResponseBody로 생성하면 코드가 중복될 수 있음

⇒@ExceptionHandler 사용

방법 1 – 한 클래스 내부에서 @ExceptionHandler 메서드 선언

```
@ExceptionHandler(MemberNotFoundException.class)
public ResponseEntity<ErrorResponse> handleNoData() {
        return ResponseEntity
        .status(HttpStatus.NOT_FOUND)
        .body(new ErrorResponse("no member"));
}
```

JSON in Spring – ResponseEntity와 예외처리

방법 2 - @RestControllerAdvice(“controller”)

API Exception을 위한 클래스 생성

```
@RestControllerAdvice("controller")
public class ApiExceptionAdvice {

        @ExceptionHandler(MemberNotFoundException.class)
        public ResponseEntity<ErrorResponse> handleNoData() {
                return ResponseEntity
                .status(HttpStatus.NOT_FOUND)
                .body(new ErrorResponse("no member"));
        }

        …. 추가 핸들러 삽입
}
```

JSON in Spring – 커맨드 객체 검증

@Valid

```
@PostMapping("/api/members")
public void newMember(
        @RequestBody @Valid RegisterRequest regReq,
        HttpServletResponse response ) throws IOException {
... }
```

- Http 상태코드에 대한 Http 응답을 처리했지만,
  @Valid가 값 검증에 실패해도 400 상태 코드를 응답한다.

- 그런데 이때도 json이 아닌 HTML로 응답하기 때문에,
  JSON으로 받고 싶다면 Errors 타입 파라미터를 추가해서
  직접 에러 응답을 생성해야함