

Part 3 Ajax 댓글 처리

Chapter 01 RestController와 Ajax

- REST(Representational State Transfer)는 하나의 URI는 하나의 고유한 리소스(Resource)를 대표하도록 설계
- 스프링은 3버전부터 @ResponseBody 어노테이션을 지원하면서 본격적으로 REST 방식의 처리를 지원한다.
- 스프링 4에 들어와서 @RestController가 지원된다.

1.1 @RestController의 소개

- 기존의 특정한 JSP와 같은 뷰를 만들어 내는 것이 목적이 아닌 REST 방식의 데이터 처리를 위해서 사용하는 어노테이션
- 스프링 3버전까지 컨트롤러는 주로 @Controller 어노테이션만을 사용해서 처리 하였고 화면 처리를 담당하는 JSP가 아닌 데이터 자체를 서비스하려면 해당 메소드나 리턴 타입에 '@ResponseBody' 어노테이션을 추가하는 형태로 작성하였다.
- @ResponseBody 어노테이션은 메소드나 리턴 타입에 사용할 수 있는 어노테이션
- 기존 JSP 방식의 처리와 달리 @ResponseBody 어노테이션의 처리는 스프링의 MessageConverter에 의해 영향을 받는다.
- 스프링에서는 전달된 데이터의 타입에 따라 MessageConverter가 데이터를 가공해서 브라우저에 전송한다.

1.2 예제 프로젝트의 생성

- 프로젝트 ex02

1.3 테스트용 컨트롤러 생성하기

- SampleController

```
@RestController
@RequestMapping("/sample")
public class SampleController {

}
```

1.3.1 @RestController의 용도

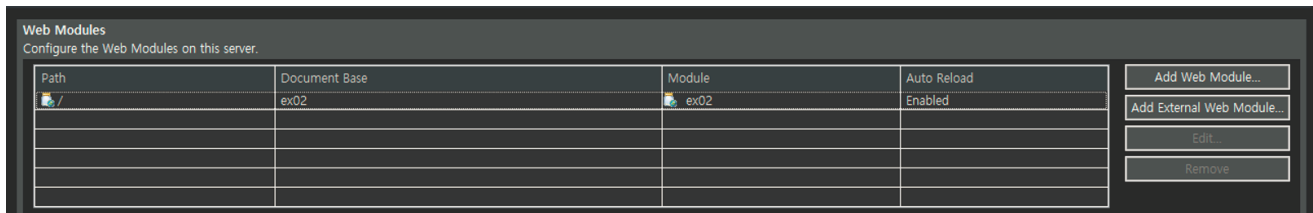
1.3.2 단순 문자열의 경우

- SampleController (sayHello)

```
@RequestMapping("/Hello")
public String sayHello() {
    return "Hello World";
}
```



1.3.3 루트 컨텍스트로 실행하기



1.3.4 객체를 JSON으로 반환하는 경우

- SampleVO

```
public class SampleVO {
    private Integer mno;
    private String firstName;
    private String lastName;
    //... getter/setter, toString()
}
```

- SampleController

```
@RequestMapping("/sendVO")
public SampleVO sendVO() {
    SampleVO vo = new SampleVO();
    vo.setFirstName("길동");
    vo.setLastName("홍");
    vo.setMno(123);
    return vo;
}
```

1.3.4.1 jackson-databind 라이브러리의 추가

- pom.xml

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.8.4</version>
</dependency>
```

- JSON 데이터 결과



```
{"mno":123,"firstName":"길동","lastName":"홍"}
```

1.3.5 컬렉션 타입의 객체를 반환하는 경우

- SampleController (sendList)

```
@RequestMapping("/sendList")
public List<SampleVO> sendList() {
    List<SampleVO> list = new ArrayList<>();

    for(int i = 0 ; i < 10; i++) {
        SampleVO vo = new SampleVO();
        vo.setFirstName("길동");
        vo.setLastName("홍");
        vo.setMno(i);
        list.add(vo);
    }
    return list;
}
```



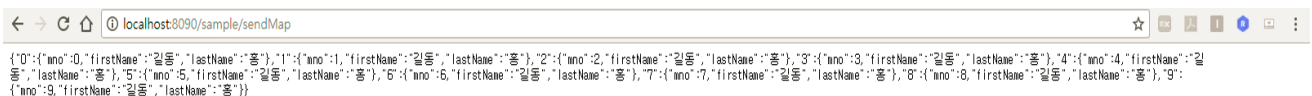
```
[{"mno":0,"firstName":"길동","lastName":"홍"}, {"mno":1,"firstName":"길동","lastName":"홍"}, {"mno":2,"firstName":"길동","lastName":"홍"}, {"mno":3,"firstName":"길동","lastName":"홍"}, {"mno":4,"firstName":"길동","lastName":"홍"}, {"mno":5,"firstName":"길동","lastName":"홍"}, {"mno":6,"firstName":"길동","lastName":"홍"}, {"mno":7,"firstName":"길동","lastName":"홍"}, {"mno":8,"firstName":"길동","lastName":"홍"}, {"mno":9,"firstName":"길동","lastName":"홍"}]
```

- SampleController (sendMap)

```
@RequestMapping("/sendMap")
public Map<Integer, SampleVO> sendMap() {
    Map<Integer, SampleVO> map = new HashMap<>();

    for(int i = 0 ; i < 10 ; i++) {
        SampleVO vo = new SampleVO();
        vo.setFirstName("길동");
        vo.setLastName("홍");
        vo.setMno(i);
        map.put(i, vo);
    }
    return map;
}
```

- sendMap 결과



```
[{"mno":0,"firstName":"길동","lastName":"홍"}, {"mno":1,"firstName":"길동","lastName":"홍"}, {"mno":2,"firstName":"길동","lastName":"홍"}, {"mno":3,"firstName":"길동","lastName":"홍"}, {"mno":4,"firstName":"길동","lastName":"홍"}, {"mno":5,"firstName":"길동","lastName":"홍"}, {"mno":6,"firstName":"길동","lastName":"홍"}, {"mno":7,"firstName":"길동","lastName":"홍"}, {"mno":8,"firstName":"길동","lastName":"홍"}, {"mno":9,"firstName":"길동","lastName":"홍"}]
```

1.3.6 ResponseEntity 타입

- ResponseEntity 타입은 개발자가 직접 결과 데이터 + HTTP의 상태 코드를 직접 제어할 수 있는 클래스
- SampleController (sendListAuth)
 - 예외 대신 ResponseEntity를 이용해서 사용자에게 정보를 전달

```
@RequestMapping("/sendErrorAuth")
public ResponseEntity<Void> sendListAuth() {
    return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
}
```

- SampleController (sendListNot)
 - 호출한 쪽으로 예외의 원인 메시지를 전송할 때 사용하는 방식
 - 결과 데이터와 HTTP의 상태 코드를 같이 사용해야하는 경우

```
@RequestMapping("/sendErrorNot")
public ResponseEntity<List<SampleVO>> sendListNot() {
    List<SampleVO> list = new ArrayList<>();
    for(int i = 0 ; i < 10 ; i++) {
        SampleVO vo = new SampleVO();
        vo.setFirstName("길동");
        vo.setLastName("홍");
        vo.setMno(i);
        list.add(vo);
    }
    return new ResponseEntity<>(list, HttpStatus.NOT_FOUND);
}
```