

29장 스프링 REST API 사용하기

29.1 REST란?

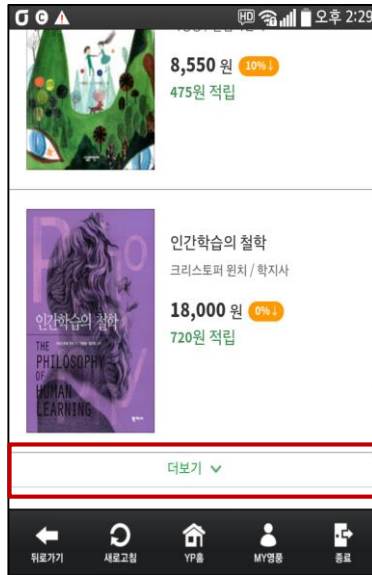
29.2 @RestController 사용하기

29.3 @PathVariable 사용하기

29.4 @RequestBody와 @ResponseBody 사용하기

29.5 REST 방식으로 URI 표현하기

29.1 REST란?



- 모바일 쇼핑몰에서 '더보기'클릭 시 데이터만 전송받아서 기존 화면에 추가해서 표시함
- 일반적으로 자원이 열악한 모바일 환경에선 화면은 그대로 유지하면서 데이터만 전송해서 서비스함

REST

- Representational State Transfer의 약자로, 하나의 URI가 고유한 리소스를 처리하는 공통 방식
- REST 방식으로 제공되는 API를 REST API(또는 RESTful API)라고 하며, 이는 트위터와 같은 Open API에서 많이 사용됨

29.2 @RestController 사용하기

- 스프링 3버전에서는 @ResponseBody 애너테이션을 지원하면서 REST 방식의 데이터 처리를 지원함
- 스프링4 버전에서는 @RestController 애너테이션을 이용해 REST 방식의 데이터 처리를 지원함

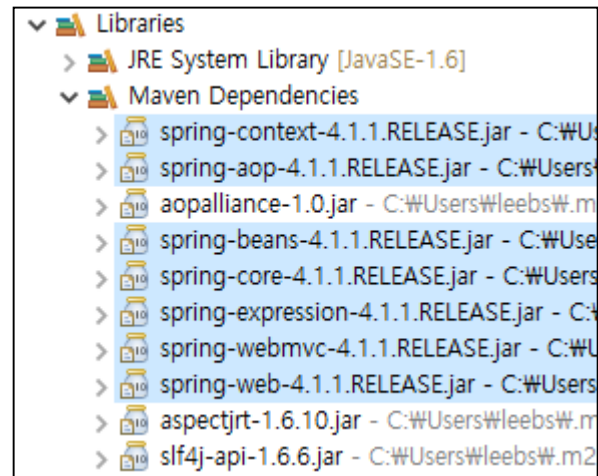
• 29. 2.1 @RestController 이용해 REST 기능 구현하기

1. pom.xml을 열어 스프링 버전을 4.1.1로 변경한 후 저장합니다.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>com.myspring</groupId>
6   <artifactId>pro29</artifactId>
7   <name>pro29</name>
8   <packaging>war</packaging>
9   <version>1.0.0-BUILD-SNAPSHOT</version>
10  <properties>
11    <java-version>1.6</java-version>
12    <org.springframework-version>4.1.1.RELEASE</org.springframework-version>
13    <org.aspectj-version>1.6.10</org.aspectj-version>
14    <org.slf4j-version>1.6.6</org.slf4j-version>
15  </properties>
```

29.2 @RestController 사용하기

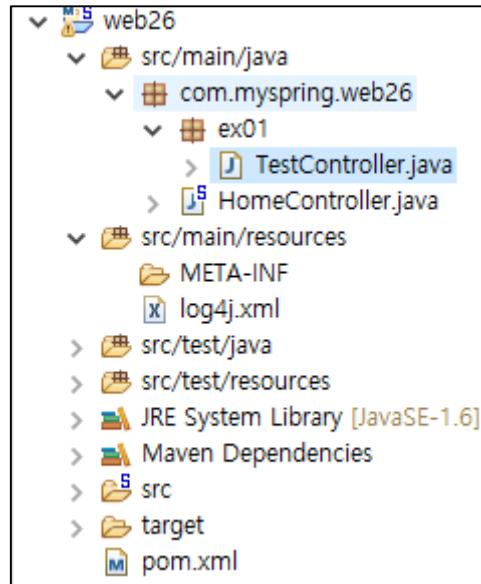
2. Maven Dependencies에서 스프링 4 버전으로 업그레이드되어 있는 것을 확인할 수 있습니다.



29.2 @RestController 사용하기

- 29.2.2 @RestController 이용해 문자열 전달하기

1. TestController 클래스 파일을 준비합니다



29.2 @RestController 사용하기

2. JSP 같은 뷰를 반환하는 것이 아니고 JSON, XML 같은 데이터를 브라우저로 전송하는 컨트롤러인 @RestController를 설정합니다.

코드 29-1 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;

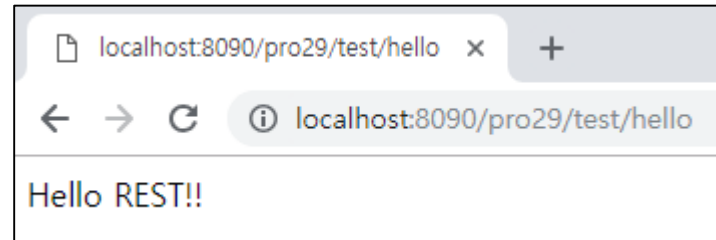
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/test/*")
public class TestController {
    @RequestMapping("/hello")
    public String hello() {
        return "Hello REST!!";
    }
}
```

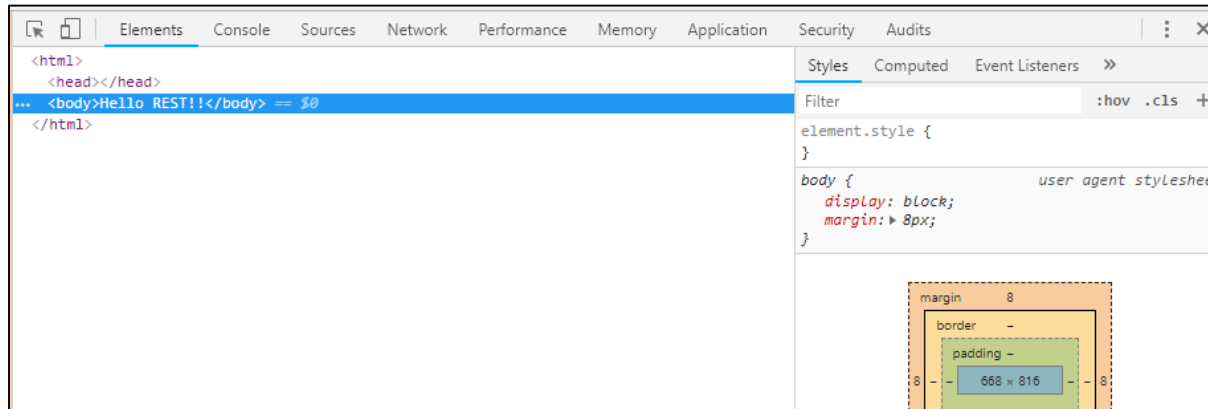
/hello로 요청 시 브라우저로 문자열을 전송합니다.

29.2 @RestController 사용하기

3. `http://localhost:8090/pro29/test/hello`로 요청하여 전송된 문자열을 표시합니다

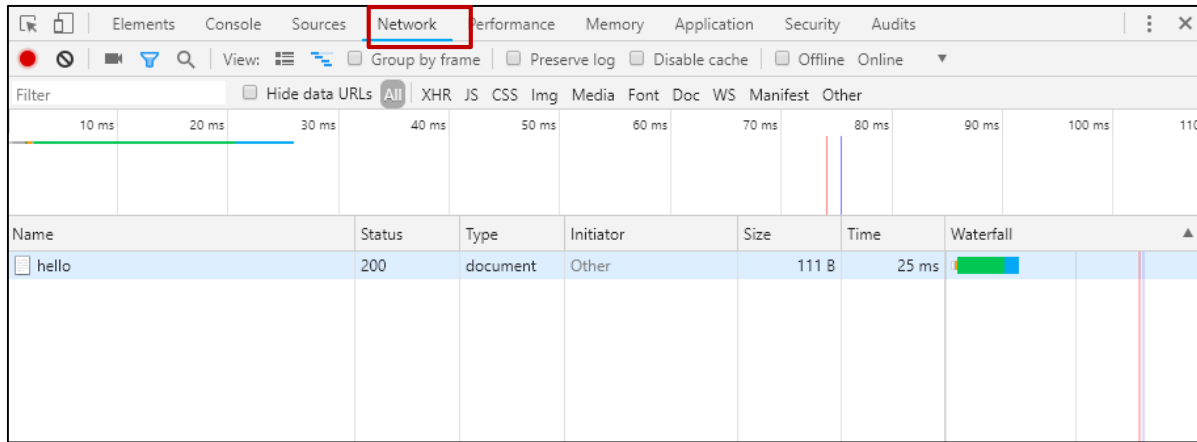


4. 그럼 전송된 데이터의 종류는 어떤 것인지 알아보까요? 크롬 브라우저를 기준으로 하겠습니다. F12를 눌러 콘솔창을 엽니다.

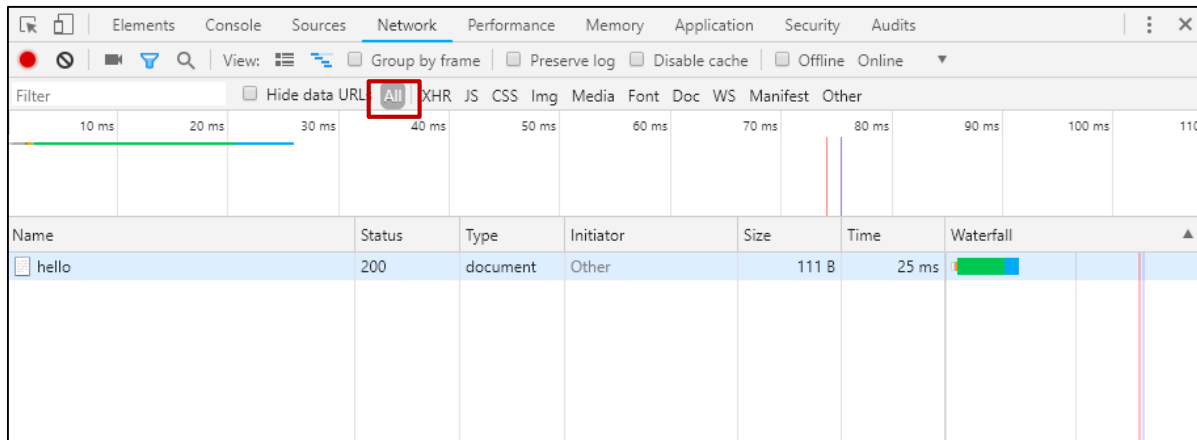


29.2 @RestController 사용하기

5. Network 탭을 클릭합니다.



6. All을 클릭한 후 다시 브라우저에 재요청합니다



29.2 @RestController 사용하기

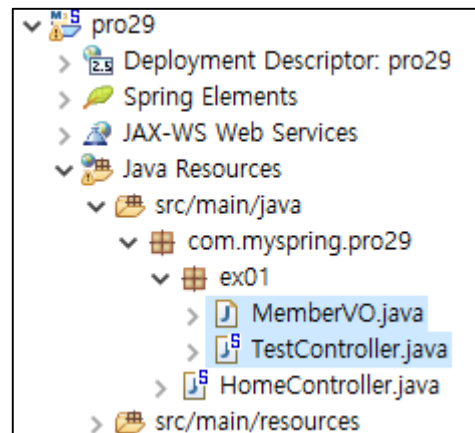
7. Name의 hello를 클릭해 'Response Headers'의 Content-Type 속성을 확인합니다.

The screenshot shows the 'Headers' tab in a web browser's developer tools. The 'Name' column on the left has a row for 'hello' which is selected and highlighted with a red box. The right pane shows the details for this request. The 'General' section displays the Request URL as `http://localhost:8090/pro29/test/hello`, Request Method as `GET`, Status Code as `200`, Remote Address as `[::1]:8090`, and Referrer Policy as `no-referrer-when-downgrade`. The 'Response Headers' section is expanded, showing `Content-Length: 12`, `Content-Type: text/html` (highlighted with a red box), and `Date: Sat, 13 Oct 2018 04:56:02 GMT`. The 'Request Headers' section is also expanded, showing `Accept: text/html,application/xhtml+xml,application/xml;q=0.8`, `Accept-Encoding: gzip, deflate, br`, `Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7`, and `Cache-Control: max-age=0`.

29.1 REST란?

- 29.2.3 @RestController 이용해 VO 객체 전달하기

1. 다음과 같이 클래스 파일을 준비합니다



29.2 @RestController 사용하기

2. JSON 기능을 이용하기 위해 pom.xml에 JSON 관련 라이브러리를 추가합니다.

코드 29-2 pro29/pom.xml

```
...
<!-- JSON -->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.5.4</version>
</dependency>
...
```

29.2 @RestController 사용하기

3. 컨트롤러에서 MemberVO의 속성들을 JSON으로 변환하여 전송합니다.

코드 29-3 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;

...

@RestController
@RequestMapping("/test/*")
public class TestController {

    ...

    @RequestMapping("/member")
    public MemberVO member() {
        MemberVO vo = new MemberVO();
        vo.setId("hong");
        vo.setPwd("1234");
        vo.setName("홍길동");
        vo.setEmail("hong@test.com");
        return vo;
    }
}
```

MemberVO 객체의 속성 값을 저장한 후
JSON으로 전송합니다.

29.2 @RestController 사용하기

4. 회원 정보를 저장할 MemberVO 클래스를 구현합니다.

코드 29-4 pro29/src/main/java/com/myspring/pro29/ex01/MemberVO.java

```
package com.myspring.pro29.ex01;
```

```
public class MemberVO {  
    private String id;  
    private String pwd;  
    private String name;  
    private String email;
```

```
//각 속성에 대한 getter/setter
```

```
...
```

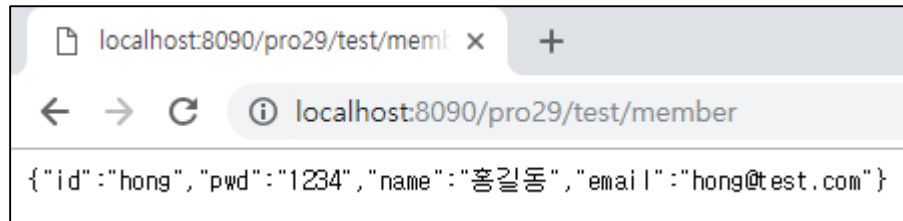
```
@Override  
public String toString() {  
    String info = id+", "+ pwd+", "+ name+", "+ email;  
    return info;  
}
```

```
}
```

회원 속성 정보를
출력합니다.

29.2 @RestController 사용하기

5. `http://localhost:8090/pro29/test/member`로 요청하여 객체의 속성 값들이 JSON 형태로 전달되는 것을 확인할 수 있습니다



29.2 @RestController 사용하기

- 29.2.4 @RestController 이용해 컬렉션 객체 전달하기

코드 29-5 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;
```

```
...
```

```
@RestController
```

```
@RequestMapping("/test/*")
```

```
public class TestController {
```

```
...
```

```
@RequestMapping("/membersList")
```

```
public List<MemberVO> listMembers () {
```

```
    List<MemberVO> list = new ArrayList<MemberVO>();
```

MemberVO 객체를 저장할
ArrayList 객체를 생성합니다.

```
    for (int i = 0; i < 10; i++) {
```

```
        MemberVO vo = new MemberVO();
```

```
        vo.setId("hong"+i);
```

```
        vo.setPwd("123"+i);
```

```
        vo.setName("홍길동"+i);
```

```
        vo.setEmail("hong"+i+"@test.com");
```

```
        list.add(vo);
```

MemberVO 객체를 10개 생성해
ArrayList에 저장합니다.

```
    }
```

```
    return list;
```

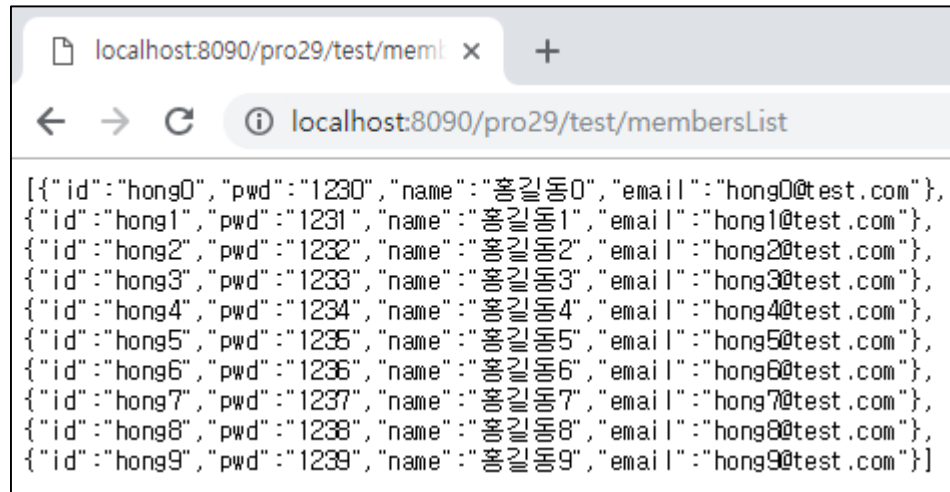
ArrayList를 JSON으로 브라우저에 전송합니다.

```
}
```

```
}
```

29.2 @RestController 사용하기

브라우저에서 요청 시 List의 객체들을 JSON으로 전송합니다.



29.2 @RestController 사용하기

- 29.2.5 @RestController 이용해 Map 전달하기

코드 29-6 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;
...
@RestController
@RequestMapping("/test/*")
public class TestController {
    ...
    @RequestMapping("/membersMap")
    public Map<Integer, MemberVO> membersMap() {
        Map<Integer, MemberVO> map = new HashMap<Integer, MemberVO>();
        for (int i = 0; i < 10; i++) {
            MemberVO vo = new MemberVO();
            vo.setId("hong" + i);
            vo.setPwd("123"+i);
            vo.setName("홍길동" + i);
            vo.setEmail("hong"+i+"@test.com");
            map.put(i, vo);
        }
        return map;
    }
}
```

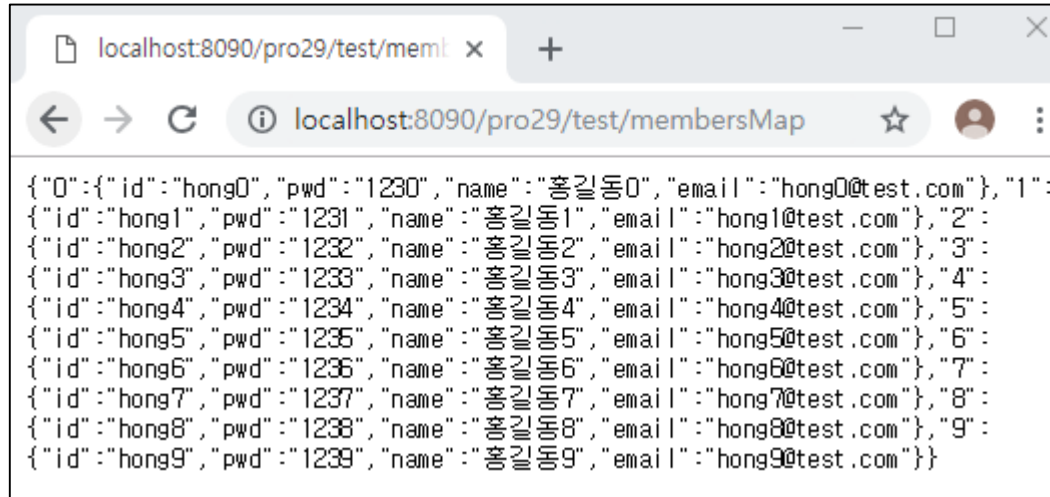
MemberVO 객체를 저장할 HashMap 객체를 생성합니다.

MemberVO 객체를 HashMap에 저장합니다.

HashMap 객체를 브라우저로 전송합니다.

29.2 @RestController 사용하기

브라우저에서 요청 시 map 데이터를 JSON으로 전송합니다.



```
{
  "0": {
    "id": "hong0",
    "pwd": "1230",
    "name": "홍길동0",
    "email": "hong0@test.com"
  },
  "1": {
    "id": "hong1",
    "pwd": "1231",
    "name": "홍길동1",
    "email": "hong1@test.com"
  },
  "2": {
    "id": "hong2",
    "pwd": "1232",
    "name": "홍길동2",
    "email": "hong2@test.com"
  },
  "3": {
    "id": "hong3",
    "pwd": "1233",
    "name": "홍길동3",
    "email": "hong3@test.com"
  },
  "4": {
    "id": "hong4",
    "pwd": "1234",
    "name": "홍길동4",
    "email": "hong4@test.com"
  },
  "5": {
    "id": "hong5",
    "pwd": "1235",
    "name": "홍길동5",
    "email": "hong5@test.com"
  },
  "6": {
    "id": "hong6",
    "pwd": "1236",
    "name": "홍길동6",
    "email": "hong6@test.com"
  },
  "7": {
    "id": "hong7",
    "pwd": "1237",
    "name": "홍길동7",
    "email": "hong7@test.com"
  },
  "8": {
    "id": "hong8",
    "pwd": "1238",
    "name": "홍길동8",
    "email": "hong8@test.com"
  },
  "9": {
    "id": "hong9",
    "pwd": "1239",
    "name": "홍길동9",
    "email": "hong9@test.com"
  }
}
```

29.3 @PathVariable 사용하기

- @PathVariable을 사용하면 브라우저에서 요청 URL로 전달된 매개변수를 가져올 수 있음

코드 29-7 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;
```

```
...
```

```
@RestController
```

```
@RequestMapping("/test/*")
```

```
public class TestController {
```

```
...
```

```
@RequestMapping(value= "/notice/{num}" , method = RequestMethod.GET)
```

```
public int notice(@PathVariable("num") int num) throws Exception {
```

```
    return num;
```

```
}
```

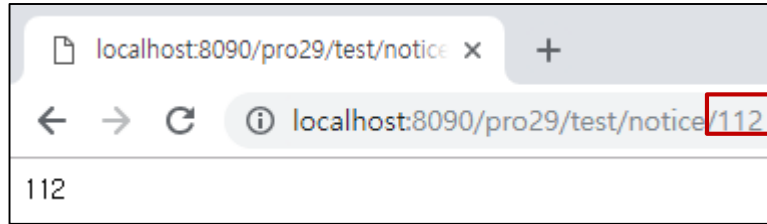
```
}
```

브라우저에서 요청 시 {num} 부분의 값이
@PathVariable로 지정됩니다.

요청 URL에서 지정된 값이 num에
자동으로 할당됩니다.

29.3 @PathVariable 사용하기

브라우저에 요청하여 notice/112로 전송할 경우 112가 num에 할당됩니다.

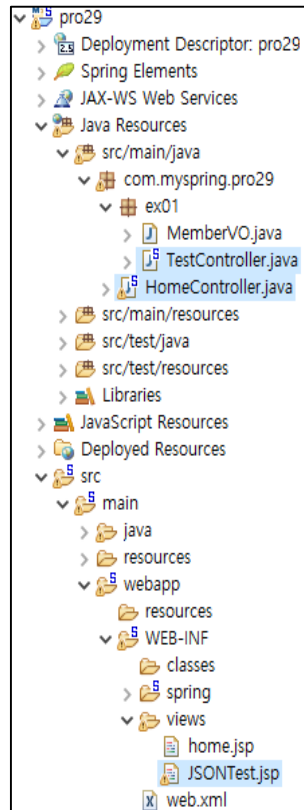


29.4 @RequestBody와 @ResponseBody 사용하기

• 29.4.1 @RequestBody 사용하기

- @RequestBody를 사용하면 브라우저에서 전달되는 JSON 데이터를 객체로 자동 변환해줌

1. 관련된 자바 클래스와 JSP 파일을 다음과 같이 추가합니다



29.4 @RequestBody와 @ResponseBody 사용하기

2. TestController를 다음과 같이 작성합니다. @RequestBody를 이용해 JSON 데이터를 MemberVO 객체로 자동 변환합니다.

코드 29-8 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;

...

@RestController
@RequestMapping("/test/*")
public class TestController {
    static Logger logger = Logger.getLogger(TestController.class);
    ...
    @RequestMapping(value= "/info", method = RequestMethod.POST)
    public void modify(@RequestBody MemberVO vo) {
        logger.info(vo.toString());
    }
}
```

JSON으로 전송된 데이터를 MemberVO 객체의 속성에 자동으로 설정해 줍니다.

29.4 @RequestBody와 @ResponseBody 사용하기

3. /pro29로 요청 시 JSONTest.jsp를 표시하도록 지정합니다

코드 29-9 pro29/src/main/java/com/myspring/pro29/HomeController.java

```
package com.myspring.pro29;

...

@Controller
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "JSONTest";
    }
}
```

29.4 @RequestBody와 @ResponseBody 사용하기

4. 회원 정보 보내기를 클릭하면 Ajax를 이용해 회원 정보를 JSON으로 만들어서 컨트롤러로 전송합니다.

코드 29-10 pro29/src/main/webapp/WEB-INF/views/JSONTest.jsp

```
...
<title>Home</title>
<script src="http://code.jquery.com/jquery-latest.js"></script>
<script>
$(function() {
    $("#checkJson").click(function() {
        var member = { id:"park",
            name:"박지성",
            pwd:"1234",
            email:"park@test.com" };
        $.ajax({
            type:"post",
            url: '${contextPath}/test/info',
            contentType: "application/json",
            data :JSON.stringify(member),
            success:function (data,textStatus){
            },
            error:function(data,textStatus){
                alert("에러가 발생했습니다.");
            },
            complete:function(data,textStatus){
            }
        });
    });
});
</script>
</head>
```

회원 정보를 JSON으로 생성합니다.

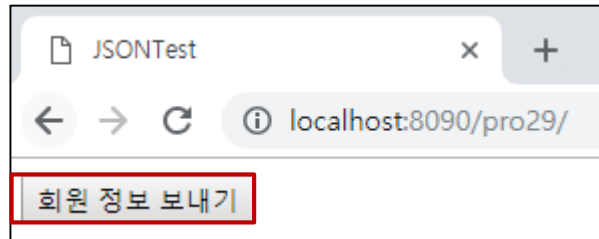
/test/info로 요청합니다.

회원 정보를 JSON 문자열로 변환합니다

29.4 @RequestBody와 @ResponseBody 사용하기

```
<body>
  <input type="button" id="checkJson" value="회원 정보 보내기"/><br><br>
  <div id="output"></div>
</body>
</html>
```

5. <http://localhost:8090/pro29>로 JSONTest.jsp로 요청한 후 회원 정보 보내기를 클릭합니다.



6. 이클립스 콘솔에 JSON으로 전송된 회원 정보가 출력된 것을 볼 수 있습니다.

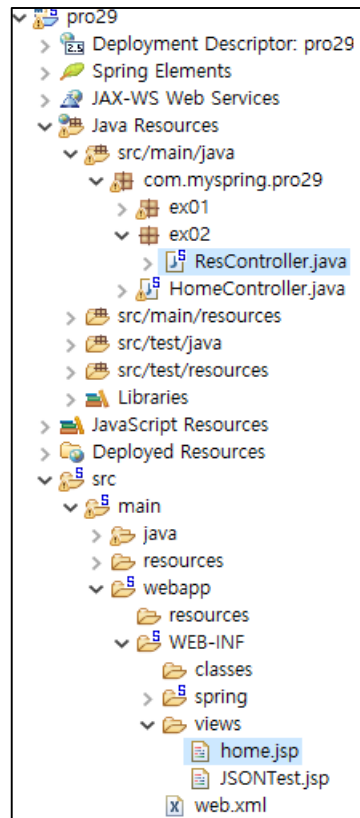
```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-10.0.2\bin\javaw.exe (2018. 10. 13. 오후 5:00:56)
INFO : com.myspring.pro29.ex01.TestController - park, 1234, 박지성, park@test.com
```

29.4 @RequestBody와 @ResponseBody 사용하기

• 29.4.2 @ResponseBody 사용하기

- 컨트롤러의 특정 메서드에 @ResponseBody를 적용하면 JSP가 아닌 텍스트나 JSON으로 결과를 전송할 수 있음

1. 다음과 같이 실습에 필요한 파일들을 준비합니다



29.4 @RequestBody와 @ResponseBody 사용하기

2. ResController 클래스를 다음과 같이 작성합니다.

코드 29-11 pro29/src/main/java/com/myspring/pro29/ex02/ResController.java

```
package com.myspring.pro29.ex02;
```

```
...
```

```
@Controller
```

```
public class ResController
```

• @RestController로 지정되지 않았습니다.

```
    @RequestMapping(value = "/res1")
```

```
    @ResponseBody
```

• 메서드 호출 시 데이터를 전송하도록 설정합니다.

```
    public Map<String, Object> res1() {
```

```
        Map<String, Object> map = new HashMap<String, Object>();
```

```
        map.put("id", "hong");
```

```
        map.put("name", "홍길동");
```

```
        return map;
```

```
    }
```

• Map 데이터를 브라우저로 전송합니다.

```
    @RequestMapping(value = "/res2")
```

```
    public ModelAndView res2() {
```

```
        return new ModelAndView("home");
```

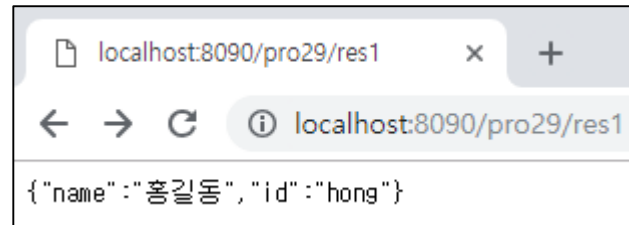
```
    }
```

• 메서드 호출 시 home.jsp를 브라우저로 전송합니다.

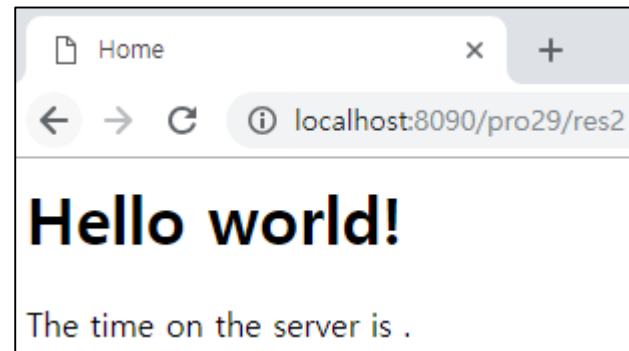
```
}
```

29.4 @RequestBody와 @ResponseBody 사용하기

3. `http://localhost:8090/pro29/res1`로 요청하여 JSON 데이터를 표시합니다.



4. `http://localhost:8090/pro29/res2`로 요청하면 `home.jsp`를 표시합니다.



29.4 @RequestBody와 @ResponseBody 사용하기

• 29.4.3 @ResponseBody 사용해서 응답하기

- @RestController는 별도의 View를 제공하지 않은 채 데이터를 전달하므로 전달 과정에서 예외가 발생할 수 있음
- 예외에 대해 좀 더 세밀한 제어가 필요한 경우 ResponseEntity 클래스를 사용함

HTTP 상태 코드 표

그룹	코드	상수	설명
정보 응답	100	CONTINUE	상태가 괜찮으며 클라이언트가 계속해서 요청하거나 요청이 완료된 경우에는 무시해도 된다는 정보를 알려줍니다.
	101	SWITCHING_PROTOCOL	클라이언트가 보낸 upgrade 요청 헤더에 대한 응답으로, 서버에서 프로토콜을 변경할 것임을 알려줍니다.
성공 응답	200	OK	요청이 성공적으로 완료되었다는 의미입니다
	201	CREATED	요청이 성공적이었으며 그 결과로 새로운 리소스가 생성되었다는 의미입니다.
	202	ACCEPTED	요청을 수신했지만 그에 응하여 행동할 수 없다는 의미입니다
...

❖ 출처

- <https://developer.mozilla.org/ko/docs/Web/HTTP/Status>

29.4 @RequestBody와 @ResponseBody 사용하기

TestController 컨트롤러에 추가된 메서드는 ResponseEntity에 오류 코드를 설정하여 응답합니다.

코드 29-12 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
package com.myspring.pro29.ex01;

...
@RestController
@RequestMapping("/test/*")
public class TestController {

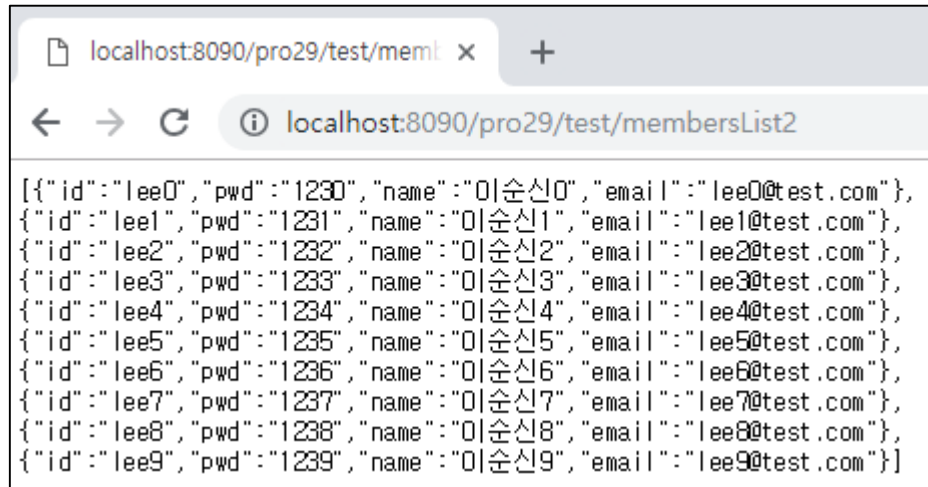
    ...
    @RequestMapping("/membersList2")
    public ResponseEntity<List<MemberVO>> listMembers2() {
        List<MemberVO> list = new ArrayList<MemberVO>();
        for (int i = 0; i < 10; i++) {
            MemberVO vo = new MemberVO();
            vo.setId("lee" + i);
            vo.setPwd("123"+i);
            vo.setName("이순신" + i);
            vo.setEmail("lee"+i+"@test.com");
            list.add(vo);
        }
        return new ResponseEntity(list, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

ResponseEntity로 응답합니다.

오류 코드 500으로 응답합니다.

29.4 @RequestBody와 @ResponseBody 사용하기

실행결과



A screenshot of a web browser window. The address bar shows the URL `localhost:8090/pro29/test/membersList2`. The page content displays a JSON array of 10 member objects. Each object contains fields for `id`, `pwd`, `name`, and `email`. The names are "이순신0" through "이순신9".

```
[{"id": "lee0", "pwd": "1230", "name": "이순신0", "email": "lee0@test.com"}, {"id": "lee1", "pwd": "1231", "name": "이순신1", "email": "lee1@test.com"}, {"id": "lee2", "pwd": "1232", "name": "이순신2", "email": "lee2@test.com"}, {"id": "lee3", "pwd": "1233", "name": "이순신3", "email": "lee3@test.com"}, {"id": "lee4", "pwd": "1234", "name": "이순신4", "email": "lee4@test.com"}, {"id": "lee5", "pwd": "1235", "name": "이순신5", "email": "lee5@test.com"}, {"id": "lee6", "pwd": "1236", "name": "이순신6", "email": "lee6@test.com"}, {"id": "lee7", "pwd": "1237", "name": "이순신7", "email": "lee7@test.com"}, {"id": "lee8", "pwd": "1238", "name": "이순신8", "email": "lee8@test.com"}, {"id": "lee9", "pwd": "1239", "name": "이순신9", "email": "lee9@test.com"}]
```

개발자 도구로 상태 코드를 조회한 결과

Name	× Headers Preview Response Cookies Timing
membersList2	<div>▼ General</div> <div>Request URL: http://localhost:8090/pro29/test/membersList2</div> <div>Request Method: GET</div> <div>Status Code: 500</div> <div>Remote Address: [::1]:8090</div> <div>Referrer Policy: no-referrer-when-downgrade</div>

29.4 @RequestBody와 @ResponseBody 사용하기

- HttpHeaders 클래스를 이용해 ResponseEntity로 전송할 데이터의 종류와 한글 인코딩을 설정할 수 있음

코드 29-13 pro29/src/main/java/com/myspring/pro29/ex01/TestController.java

```
...
@RequestMapping(value = "/res3")
public ResponseEntity res3() {
    HttpHeaders responseHeaders = new HttpHeaders();
    responseHeaders.add("Content-Type", "text/html; charset=utf-8");
    String message = "<script>";
    message += " alert('새 회원을 등록합니다.');"
    message += " location.href='/pro29/test/membersList2'; ";
    message += " </script>";
    return new ResponseEntity(message, responseHeaders, HttpStatus.CREATED);
}
...
```

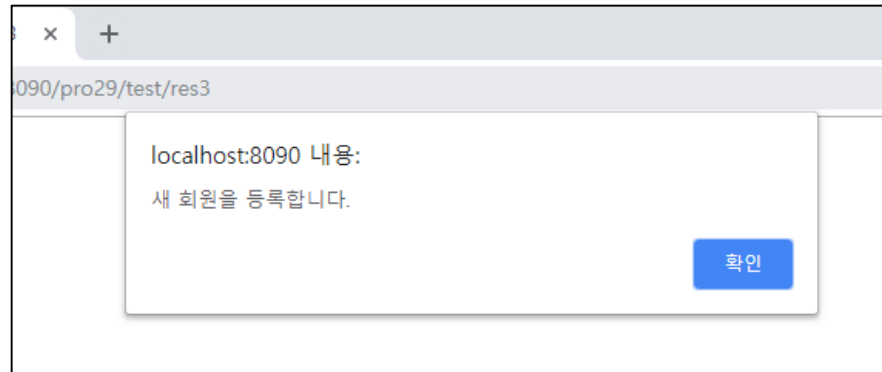
전송할 데이터의 종류와 인코딩을 설정합니다.

전송할 자바스크립트 코드를 문자열로 작성합니다.

ResponseEntity를 이용해 HTML 형식으로 전송합니다.

29.4 @RequestBody와 @ResponseBody 사용하기

브라우저에서 /test/res3으로 요청하면 다음과 같이 전송된 자바스크립트 경고 메시지를 표시합니다.



29.5 REST 방식으로 URI 표현하기

- REST 방식은 서버에 데이터 조회뿐만 아니라 추가, 수정, 삭제 작업 요청 시 HTTP 메서드를 이용함

HTTP 메서드의 기능

메서드	설명
POST	추가(Create)
GET	조회(Select)
PUT	수정(Update)
DELETE	삭제>Delete)

- REST 방식으로 요청 하는 URI 형식

/작업명/기본키 + 메서드 + 데이터

- 작업명: 요청하는 작업 종류
- 기본키: 요청하는 작업에 해당하는 대상의 기본키
- 메서드: 요청하는 기능
- 데이터: 기능 수행에 필요한 JSON 데이터

29.5 REST 방식으로 URI 표현하기

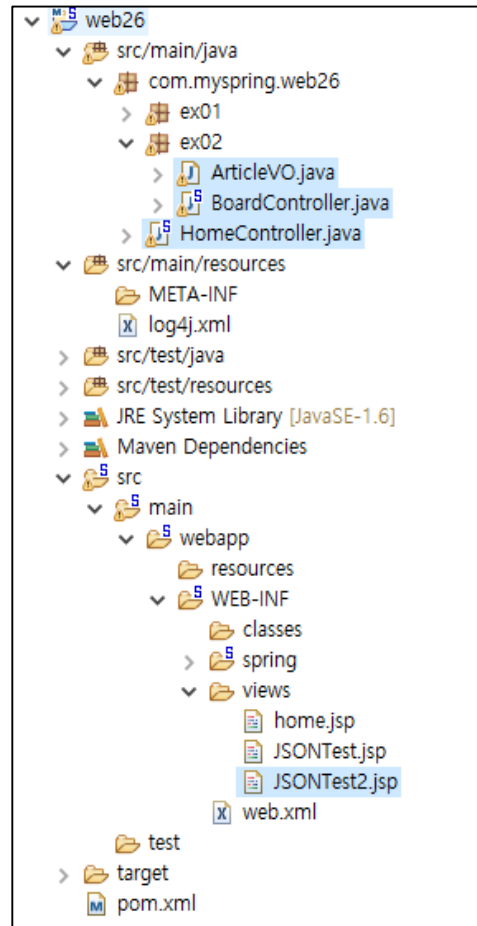
REST로 게시판 기능 관련 URI 만들기

메서드	URI	설명
POST	/boards + 데이터	새 글 등록하기
GET	/boards/133	133번 글 조회하기
PUT	/boards/133 + 데이터	133번 글 수정하기
DELETE	/boards/133	133번 글 삭제하기

29.5 REST 방식으로 URI 표현하기

- 29.5.1 게시판 기능 REST API 만들기

1. 게시판 글에 대한 CRUD 기능을 하는 BoardController와 관련된 파일들을 준비합니다



29.5 REST 방식으로 URI 표현하기

2. 앞에서 설명한 REST 방식으로 컨트롤러의 메서드들을 구현합니다. method의 속성에 GET, POST, PUT, DELETE를 지정하여 각 메서드들의 기능을 정의합니다.

코드 29-14 pro29/src/main/java/com/myspring/pro29/ex03/BoardController.java

```
package com.myspring.pro29.ex03;
...
@RestController
@RequestMapping("/boards")
public class BoardController {
    static Logger logger = Logger.getLogger(BoardController.class);

    @RequestMapping(value = "/all", method = RequestMethod.GET)
    public ResponseEntity<List<ArticleVO>> listArticles() {
        logger.info("listArticles 메서드 호출");
        List<ArticleVO> list = new ArrayList<ArticleVO>();
        for (int i = 0; i < 10; i++) {
            ArticleVO vo = new ArticleVO();
            vo.setArticleNO(i);
            vo.setWriter("이순신"+i);
            vo.setTitle("안녕하세요"+i);
            vo.setContent("새 상품을 소개합니다."+i);
            list.add(vo);
        }
        return new ResponseEntity(list, HttpStatus.OK);
    }
}
```

첫 번째 단계의 요청명을 매핑합니다.

GET 방식으로 요청하므로 모든 글의 정보를 조회합니다.

29.5 REST 방식으로 URI 표현하기

GET 방식으로 요청하면서 글 번호를 전달하므로
글 번호에 대한 글 정보를 조회합니다.

```
@RequestMapping(value = "/{articleNO}", method = RequestMethod.GET)
public ResponseEntity<ArticleVO> findArticle (
    @PathVariable("articleNO") Integer articleNO) {
    logger.info("findArticle 메서드 호출");
    ArticleVO vo = new ArticleVO();
    vo.setArticleNO(articleNO);
    vo.setWriter("홍길동");
    vo.setTitle("안녕하세요");
    vo.setContent("홍길동 글입니다");
    return new ResponseEntity(vo, HttpStatus.OK);
}
```

POST 방식으로 요청하므로 요청 시 JSON으로
전달되는 객체를 새 글로 추가합니다.

```
@RequestMapping(value = "", method = RequestMethod.POST)
public ResponseEntity<String> addArticle (@RequestBody ArticleVO articleVO) {
    ResponseEntity<String> resEntity = null;
    try {
        logger.info("addArticle 메서드 호출");
        logger.info(articleVO.toString());
        resEntity = new ResponseEntity("ADD_SUCCEEDED", HttpStatus.OK);
    } catch (Exception e) {
        resEntity = new ResponseEntity(e.getMessage(), HttpStatus.BAD_REQUEST);
    }
    return resEntity;
}
```

PUT 방식으로 요청하므로 articleNO에 대한 글을
전달되는 JSON 정보로 수정합니다.

```
@RequestMapping(value = "/{articleNO}", method = RequestMethod.PUT)
```

29.5 REST 방식으로 URI 표현하기

```

public ResponseEntity<String> modArticle(@PathVariable("articleNO") Integer
                                         articleNO, @RequestBody ArticleVO articleVO) {
    ResponseEntity<String> resEntity = null;
    try {
        logger.info("modArticle 메서드 호출");
        logger.info(articleNO);
        logger.info(articleVO.toString());
        resEntity = new ResponseEntity("MOD_SUCCEEDED", HttpStatus.OK);
    } catch (Exception e) {
        resEntity = new ResponseEntity(e.getMessage(), HttpStatus.BAD_REQUEST);
    }

    return resEntity;
}

@RequestMapping(value =("/{articleNO}", method = RequestMethod.DELETE)
public ResponseEntity<String> removeArticle (@PathVariable("articleNO") Integer articleNO) {
    ResponseEntity<String> resEntity = null;
    try {
        logger.info("removeArticle 메서드 호출");
        logger.info(articleNO);
        resEntity = new ResponseEntity("REMOVE_SUCCEEDED", HttpStatus.OK);
    } catch (Exception e) {
        resEntity = new ResponseEntity(e.getMessage(), HttpStatus.BAD_REQUEST);
    }

    return resEntity;
}
}

```

전송된 JSON 회원 정보를 바로 ArticleVO 객체의 속성에 설정합니다.

DELETE 방식으로 요청하므로 전달되는 articleNO에 대한 글을 삭제합니다.

29.5 REST 방식으로 URI 표현하기

3. 앞에서 실습한 HomeController 클래스의 반환 값을 JSONTest2로 변경합니다(코드 29-9 참조).
4. 새 글 등록, 수정, 삭제에 사용할 JSONTest2.jsp를 다음과 같이 작성합니다.

코드 29-15 pro29/src/main/webapp/WEB-INF/views/JSONTest2.jsp

```
...  
<script src="http://code.jquery.com/jquery-latest.js"></script>  
<script>  
    $(function() {  
        $("#checkJson").click(function() {  
            var article = {articleNO:"114",  
                           writer:"박지성",  
                           title:"안녕하세요",  
                           content:"상품 소개 글입니다."  
            };  
        };
```

새 글 정보를 JSON으로 생성합니다.

29.5 REST 방식으로 URI 표현하기

```
$.ajax({
  type: "POST",
  url: "${contextPath}/boards",
  /*
  type: "PUT",
  url: "${contextPath}/boards/114",
  */
  contentType: "application/json",
  data: JSON.stringify(article),
  success: function (data, textStatus){
    alert(data);
  },
  error: function(data, textStatus){
    alert("에러가 발생했습니다.");
  },
  complete: function(data, textStatus){
  }
});
});
});
</script>
</head>
<body>
  <input type="button" id="checkJson" value="새글 쓰기"/><br><br>
  <div id="output"></div>
</body>
```

새 글 등록은 POST 방식으로 요청합니다.

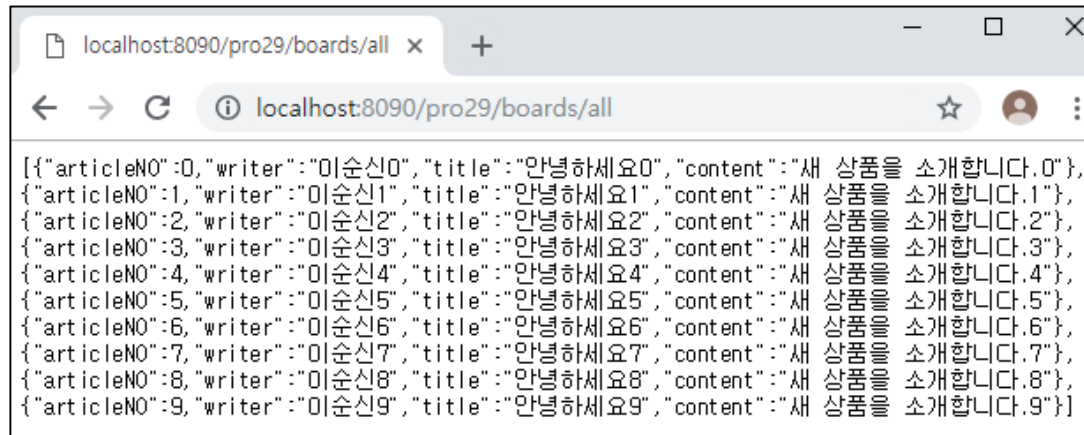
새 글을 등록하는 메서드를 호출합니다.

글 번호 114번에 대해 수정을 요청합니다.

글 정보를 JSON 형식으로 전송합니다.

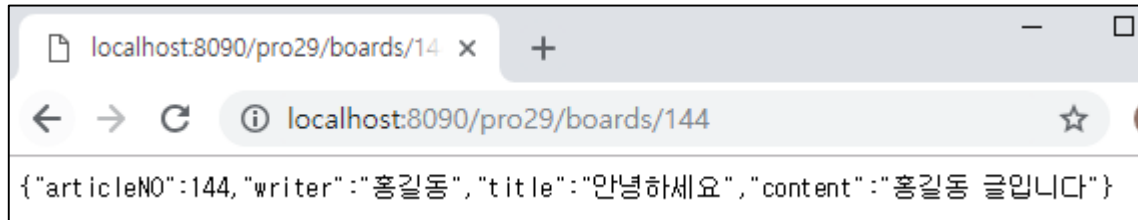
29.5 REST 방식으로 URI 표현하기

5. 브라우저 주소창에서 `http://localhost:8090/pro29/boards/all`로 요청할 경우 다음과 같이 전체 글 정보를 전송합니다

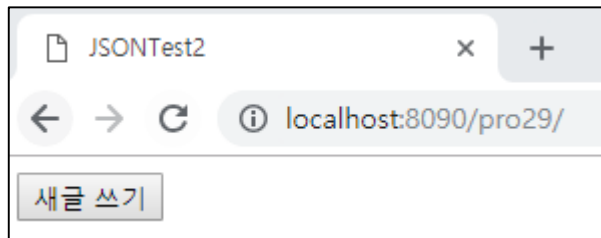


29.5 REST 방식으로 URI 표현하기

6. 브라우저 주소창에서 `http://localhost:8090/pro29/boards/144`로 요청하면 144번 글에 대한 정보만 조회합니다.



7. 브라우저 주소창에서 `http://localhost:8090/pro29/`로 요청하여 JSONTest2.jsp를 표시한후 새글 쓰기를 클릭합니다



29.5 REST 방식으로 URI 표현하기

8. JSONTest2.jsp에서 Ajax로 전송한 새 글을 이클립스 콘솔로 출력합니다.

```
INFO : com.myspring.pro29.ex03.BoardController - addArticle 메서드 호출  
INFO : com.myspring.pro29.ex03.BoardController -  
114  
박지성  
안녕하세요  
상품 소개 글입니다.
```

9. JSONTest2.jsp의 Ajax 구문에서 type을 PUT으로, url 속성을 /boards/114로 수정하여 다시 요청합니다. 그러면 다음과 같이 글 수정 메서드를 호출합니다.

```
INFO : com.myspring.pro29.ex03.BoardController - modArticle 메서드 호출  
INFO : com.myspring.pro29.ex03.BoardController - 114  
INFO : com.myspring.pro29.ex03.BoardController -  
114  
박지성  
안녕하세요  
상품 소개 글입니다.
```