Restful API in Spring

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Spring5

Open API

- 개방형 API
- Programming에서 사용할 수 있는 개방되어 있는 상태의 interface를 말한다.
- Portal이나 통계청, 기상청 등과 같은 관공서에서 가지고 있는 Data를 외부 응용 program에서 사용할 수 있도록 Open API를 제공하고 있다.
- Open API와 함께 사용하는 기술 중 REST가 있고, 대부분의 Open API는 REST 방식으로 지원되고 있다.

RESTful Web Service 개요

- REST(REpresentational Safe Transfer)
  - HTTP URI + HTTP Method
  - HTTP URI를 통해 제어할 자원(Resource)을 명시하고, HTTP Method(GET, POST, PUT, DELETE)를 통해 해당 Resource를 제어하는 명령을 내리는 방식의 architecture.
  - HTTP protocol에 정의된 4개의 method들이 Resource에 대한 CRUD Operation을 정의
    - POST : Create(Insert)
    - GET : Read(Select)
    - PUT : Update or Create
    - DELETE : Delete

RESTful API?

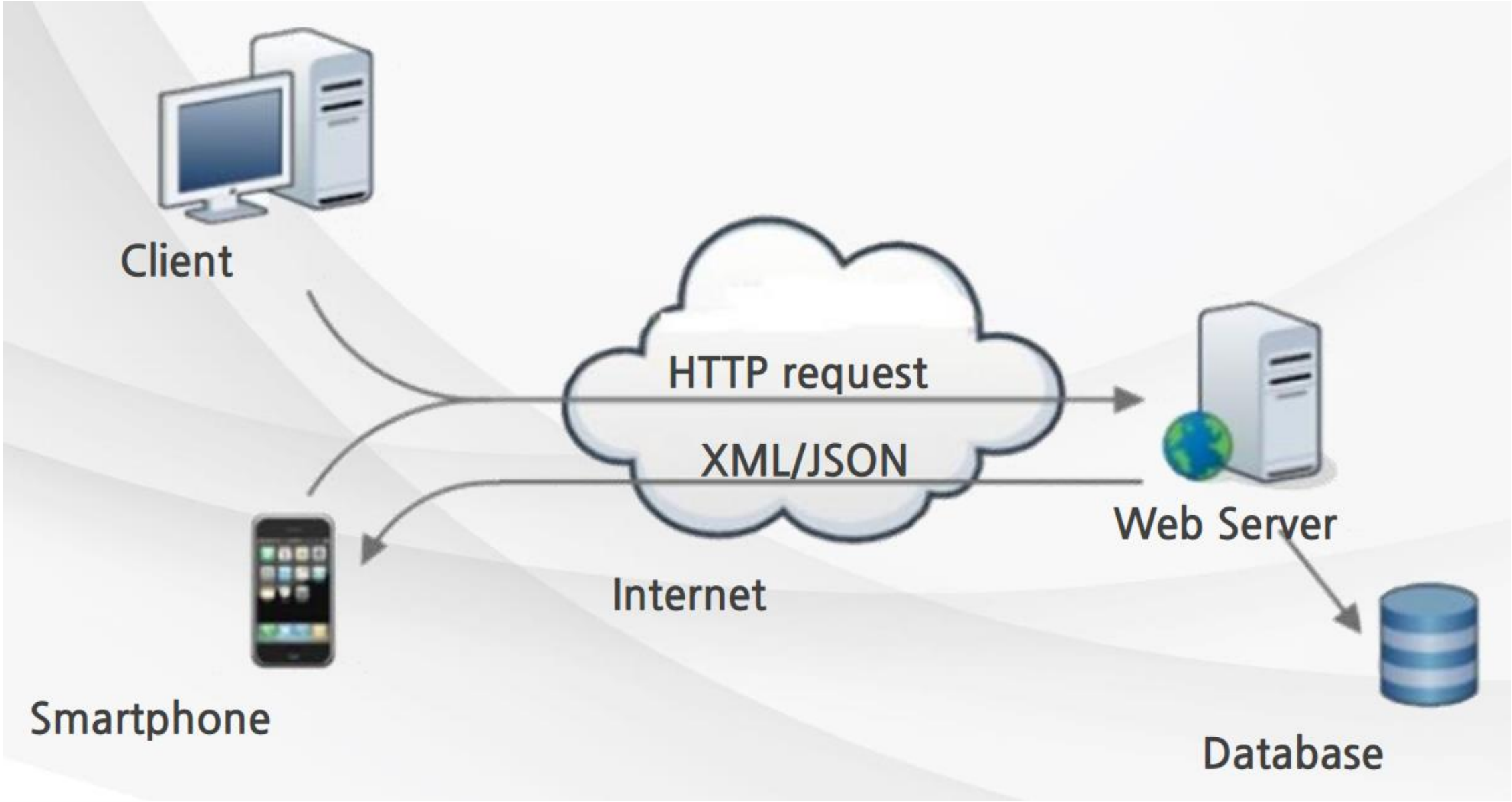
- HTTP와 URI 기반으로 자원에 접근할 수 있도록 제공하는 Application 개발 Interface.
- 즉, REST의 원리를 따르는 System을 가리키는 용어로 사용
- 기존의 Web 접근 방식과 RESTful API 방식과의 차이점(예:게시판)

	기존 게시판	RESTful API를 지원하는 게시판
글읽기	***GET*** /list.do?no=4&name=Spring	***GET*** /board/Spring/4
글등록	***POST*** /insert.do	***POST*** /board/Spring/4
글삭제	***GET*** /delete.do?no=4&name=Spring	***DELETE*** /board/Spring/4
글수정	***POST*** /update.do	***PUT*** /board/Spring/4

RESTful API? (Cont.)

- 기존의 게시판은 GET과 POST만으로 자원에 대한 CRUD를 처리하며, URI는 Action을 나타낸다.
- RESTful 게시판은 4가지 Method를 모두 사용하여 CRUD를 처리하며, URI는 제어하려는 자원을 나타낸다.

JSON과 XML


Client
HTTP request
XML/JSON
Web Server
Internet
Smartphone
Database

JSON과 XML (Cont.)

- JSON(JavaScript Object Notation)?
  - http://www.json.org
  - 경량의 Data 교환 Format
  - JavaScript에서 객체를 만들 때 사용하는 표현식을 의미
  - JSON 표현식은 사람과 기계 모두 이해하기 쉬우며 용량이 작아서, 최근에는 XML을 대체해서 Data 전송 등에 많이 사용된다.
  - 특정 언어에 종속되지 않으며, 대부분의 programming 언어에서 JSON format의 data를 handling할 수 있는 library를 제공하고 있다.
  - `name : value` 형식의 pair

JSON과 XML (Cont.)

- JSON(JavaScript Object Notation)?

```
{
    "name" : "조용필",
    "gender" : "남성",
    "age" : 50,
    "city" : "Seoul",
    "hobby" : ["등산", "낚시", "게임"]
}
```

JSON과 XML (Cont.)

- JSON(JavaScript Object Notation)?

  JSON library - Jackson

  -http://jackson.codehous.org

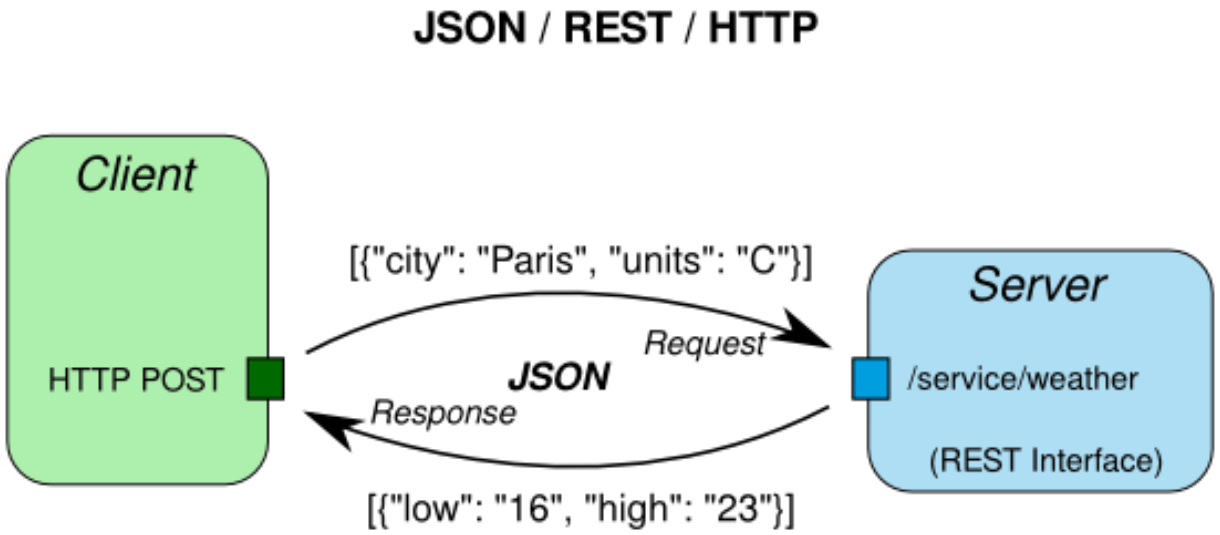
  -High-Performance JSON Processor!

  -Jackson은 JSON 형태를 Java 객체로, Java 객체를 JSON 형태로 변환해 주는 Java용 JSON library이다.

  -가장 많이 사용하는 JSON library이다.

JSON과 XML (Cont.)

- JSON library
  - http://jackson.codehous.org
  - High-Performance JSON Processor!
  - Jackson은 JSON 형태를 Java 객체로, Java 객체를 JSON 형태로 변환해 주는 Java용 JSON library이다.
  - 가장 많이 사용하는 JSON library이다.



JSON과 XML (Cont.)

- XML?
  - eXtensible Markup Language
  - Data를 저장하고 전달/교환하기 위한 언어
  - 인간/기계 모두에게 읽기 편한 언어
  - Data의 구조와 의미를 설명
  - HTML이 Data의 표현에 중점을 두었다면 XML은 Data를 전달하는 것에 중점을 맞춘 언어
  - HTML은 미리 정의된 Tag만 사용 가능하지만, XML은 사용자가 Tag를 정의할 수 있다.

JSON과 XML (Cont.)

- XML?

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
    <product>
        <name>Ballpen</name>
        <price 단위="원">150</price>
        <maker>모나미</maker>
        <color>black</color>
    </product>
</products>
```

JSON과 XML (Cont.)

- Jackson2 API 설치
  - http://mvnrepository.com에서 '***jackson databind***'로 검색
  - '***Jackson Databind***' 2.10.0 버전을 pom.xml에 추가
  - '***Jackson Core***' 2.10.0 버전을 pom.xml에 추가
  - '***Jackson Annotations***' 2.10.0 버전을 pom.xml에 추가

web.xml의 DispatcherServlet url-pattern 변경

- 기존

```
<servlet-mapping>
        <servlet-name>springDispatcherServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

- 변경

```
<servlet-mapping>
        <servlet-name>springDispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

Spring Bean Configuration File(beans.xml) 설정

- Spring MVC에 필요한 Bean들을 자동으로 등록해주는 Tag

```
<mvc:annotation-driven />
```

Spring MVC기반 RESTful Web Service 구현 절차

- RESTful Web Service를 처리할 **RestfulController** Class 작성 및 Spring Bean으로 등록
- 요청을 처리할 Method에 **@RequestMapping** **@RequestBody**와 **@ResponseBody** Annotation 선언
- REST Client Tool(***Postman***)을 사용하여 각각의 Method Test
- Ajax 통신을 하여 RESTful Web Service를 호출하는 HTML Page 작성

사용자 관리 RESTful Web Service URI와 Method – 例

- Action Resource URI HTTP Method
  - 사용자 목록 `/users` ***GET***
  - 사용자 보기 `/users/{id}` ***GET***
  - 사용자 등록 `/users` ***POST***
  - 사용자 수정 `/users` ***PUT***
  - 사용자 삭제 `/users/{id}` ***DELETE***

RESTful Controller를 위한 핵심 Annotation

- Spring MVC에서는 Client에서 전송한 XML이나 JSON Data를 Controller에서 Java 객체로 변환해서 받을 수 있는 기능(수신)을 제공하고 있다.
- Java객체를 XML이나 JSON으로 변환해서 전송할 수 있는 기능(송신)을 제공하고 있다.
- Annotation 설명
  - **`@RequestBody`** : HTTP Request Body(요청 몸체)를 Java객체로 전달받을 수 있다.
  - **`@ResponseBody`** : Java객체를 HTTP Response Body(응답 몸체)로 전송할 수 있다.

Google Postman 설치

- https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en
- [Launch app] button click
- Log in - Sign with Google

POSTMAN

Data 변환 - JSON으로 변환

- System이 복잡해지면서 다른 System과 정보를 주고받을 일이 발생하는데, 이 때 Data 교환 Format으로 JSON을 사용할 수 있다.
- 검색결과를 JSON Data로 변환하려면 가장 먼저 jackson2 library를 Download 받아야 한다.
- Jackson2는 Java 객체를 JSON으로 변환하거나 JSON을 Java 객체로 변환해주는 Library다.
- https://www.concretepage.com/spring-4/spring-4-rest-xml-response-example-with-jackson-2 참조
- https://www.mkyong.com/java/jackson-2-convert-java-object-to-from-json/ 참조

Data 변환 - JSON으로 변환 (Cont.)

- 보통 User가 Servlet이나 JSP를 요청하면 Server는 요청한 File을 찾아서 실행한다.
- 그 실행결과는 HTTP Response Package의 Body에 저장하여 Browser에 전송한다.
- 그런데, 이 응답결과를 HTML이 아니라 JSON이나 XML로 변환하여 Body에 저장하려면 Spring에서 제공하는 변환기(Converter)를 사용해야 한다.
- Spring은 **`HttpMessageConverter`**를 구현한 다양한 변환기를 제공한다.

Data 변환 - JSON으로 변환 (Cont.)

- 이 변환기를 이용하면 Java 객체를 다양한 타입으로 변환하여 HTTP Response body에 설정할 수 있다.
- **HttpMessageConverter**를 구현한 Class는 여러가지가 있으며, 이 중에서 Java 객체를 JSON responsebody로 변환할 때는 **MappingJackson2HttpMessageConverter**를 사용한다.
- 따라서 **MappingJackson2HttpMessageConverter**를 Spring 설정 File에 등록하면 되는데, 혹시 이후에 XML 변환도 처리할 예정이라면 다음처럼 설정한다.

```
<mvc:annotation-driven />
```

- Spring Bean Configuration File에 위와 같이 설정하면 **HttpMessageConverter**를 구현한 모든 변환기가 생성된다.

Data 변환 - JSON으로 변환 (Cont.)

- HomeController.java

```
@Controller
public class UserController {
        @Autowired
        private UserService userService;

        @RequestMapping("/userinfo.do")
        @ResponseBody
        public UserVO userinfo(@RequestParam("userId") String userId) {
                return userService.getUser(userId);
        }
}
```

Data 변환 - JSON으로 변환 (Cont.)

- 이전 Method와 달리 **@ResponseBody**라는 Annotation을 추가했는데, Java 객체를 Http Response Protocol의 Body로 변환하기 위해 사용된다.
- 이미 Spring Configuration File에 **<mvc:annotation-driven>**을 추가했기 때문에 **@ResponseBody**가 적용된 Method의 실행 결과는 JSON으로 변환되어 HTTP Response Body에 다음과 같이 설정된다.

```
{"userId":"jimin","name":"한지민","gender":"여","city":"서울"}
```

Data 변환 - JSON으로 변환 (Cont.)

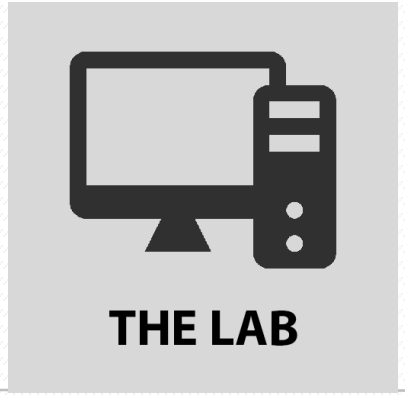
- 만일 이때, Java 객체를 JSON으로 변환할 때, 특정 변수를 제외시키려면 @JsonIgnore Annotation을 해당 변수의 Getter에 설정하면 된다.

```
package com.example.vo;
import com.fasterxml.jackson.annotation.JsonIgnore;
public class UserVO {
        ...
        @JsonIgnore
        public String getGender() {
                return gender;
        }
```

Data 변환 - JSON으로 변환 (Cont.)

- 이렇게 하면 아래와 같이 성별이 포함되지 않는다는 것을 알 수 있다.

```
{"userId":"jimin","name":"한지민","city":"서울"}
```



Task 1. Using Restful API with JSON

<mvc:default-servlet-handler/>

- Tomcat은 Client의 요청 URL을 보고 Servlet Mapping에 따라 URL에 Mapping된 Servlet이 처리를 하는 구조이다.
- 그리고 URL에 Mapping되는 Servlet이 없다면, 예를 들어 CSS, Image File 같은 정적자원들은 **defaultSevlet**이 처리하도록 되어 있다.
- 즉 CSS , Image File들은 Server 외부에서 직접 접근 할 수 없는 /WEB-INF/assets Folder 아래에 위치하는 것이 일반적인데, CSS , Image File에 접근하기 위한 Servlet Mapping을 하지 않았으면 Tomcat이 **defaultServlet**으로 처리하여 정적 File에 접근한다.
- 일반적으로 정적 File에 대해 Servlet Mapping을 하지 않는다는 것이다.

<mvc:default-servlet-handler/> (Cont.)

- Spring에서는 **DispatcherServlet**이 모든 요청을 받아 들인 후 Handler Mapping Table에 따라 Controller로 분기 한다.
- 그렇기 때문에 **DispatcherServlet**은 정적 File에 대해 Tomcat이 **defaultServlet**으로 실행할 수 있는 기회를 뺏어간다.
- 모든 요청은 일단 **DispatcherServlet**에서 처리해버리기 때문이다.
- 정적 자원에 접근하기 위한 경로 설정을 JSP/Servlet과 똑같이 해도 Spring에서는 경로를 읽지 못한다.
- 그런데 Spring Project가 아닌 JSP/Servlet Project를 만들어서 똑같은 Directory 구조로 같은 Code로 **<img>** Tag를 추가하면 정상적으로 응답되는 것을 알 수 있다.

<mvc:default-servlet-handler/> (Cont.)

- JSP/Servlet에서는 Tomcat이 **defaultServlet**이 있기 때문에 처리가 가능하지만, Spring에서는 **DispatcherServlet**이 모든 요청을 받아들이기 때문에 Tomcat의 **defaultServlet**이 정적 File을 처리할 수 있는 기회를 잃게 되는 것이다.
- 이제 Spring에서도 CSS, Image File 등이 정상적으로 응답 될 수 있도록 환경 설정을 하도록 하는 방법을 소개한다.
- ***spring-servlet.xml*** 에 아래의 code를 추가하면 된다.

```
<!-- Servlet Container의 default servlet 위임 handler -->
<mvc:default-servlet-handler />
```

<mvc:default-servlet-handler/> (Cont.)

- 이 Code는 만약 Handler Mapping에 해당하는 URL이 없으면 **default-servlet**으로 처리하겠다는 의미이다.
- 즉 Mapping이 되지 않은 URL은 ***webapp*** Folder를 시작으로 경로를 찾아가게 되고, 여기에서도 해당 경로의 자원이 존재하지 않으면 **404 Not Found**가 발생한다.
- 정적 File의 경로를 작성할 때 자신의 Application 경로를 Project Folder 이름으로 작성하는 것 말고, JSTL 표기법으로 작성할 수도 있다.
- 두 번째와 같이 JSTL로 Context 경로를 설정하는 방법의 이점은 Project Folder의 이름을 URL 주소로 사용하고 싶지 않을 때이다.

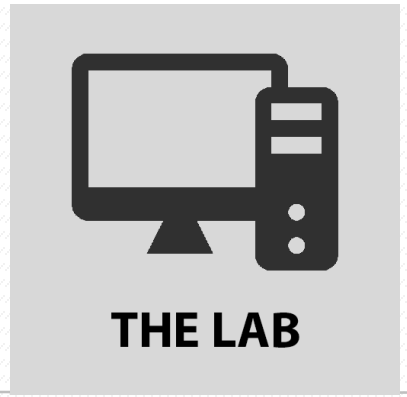
```
<img src="/guestbook/assets/images/1.jpg">
<img src="${pageContext.servletContext.contextPath }/assets/images/1.jpg">
```

<mvc:default-servlet-handler/> (Cont.)

- JSTL 표기법으로 Context 경로를 설정하면, Context 경로를 변경했을 때 일일이 /guestbook 을 새로운 경로로 바꿔주는 수고를 덜 수 있다.
- 정리하면,
- **<mvc:default-servlet-handler />**(context 설정에 따라 **<default-servlet-handler />**) 설정을 추가하면, default-servlet Handler가 bean으로 등록되며, Spring MVC는 다음과 같이 동작한다.
  - 요청 URL에 Mapping되는 Controller를 검색한다.
    - 존재할 경우, Controller를 이용해서 Client 요청을 처리한다.
  - default-servlet handler가 등록되어 있지 않다면("**<mvc:default-servlet-handler />**"를 써주지 않았다면)
    - 404응답 Error를 전송한다.
  - default-servlet handler가 등록되어 있으면, default-servlet Handler에 요청을 전달한다.
    - default-servlet handler는 WAS의 default-servlet 에 요청을 전달한다.

<mvc:default-servlet-handler/> (Cont.)

- 첨부.
  - 각 WAS는 Servlet Mapping에 존재하지 않는 요청을 처리하기 위한 default-servlet을 제공하고 있다.
  - 예를 들어, Controller `@RequestMapping`에 등록되지 않는 요청 또는 JSP에 대한 요청을 처리하는 것이 바로 default-servlet이다.
  - `DispatcherServlet`의 Mapping URL Pattern(web.xml에서 설정)을 "/"로 지정하면 JSP를 제외한 모든 요청이 `DispathcerServlet`으로 가기 때문에, WAS가 제공하는 default servlet이 *.html이나 *.css와 같은 요청을 처리할 수 없게 된다.
  - default-servlet Handler는 바로 이 default-servlet에 요청을 전달해주는 Handler로서, 요청 URL에 Mapping되는 Controller가 존재하지 않을 때 404 응답대신, default-servlet이 해당 요청 URL을 처리하도록 한다.
  - 따라서, *.css와 같은 Controller에 Mapping되어 있지 않은 URL 요청은 최종적으로 앞에서 설명한 과정을 통해 default-servlet에 전달되어 처리된다.



Task 2. Using Restful API with XML