2023.01.26: 스프링 기초

작성일시	@2023년 1월 26일 오전 9:08
강의 번호	
유형	
자료	
복습	☐
태그	스프링프레임워크

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c18144b4-bc87-4712-8e6c-de5cdd657d97/Spring_%EA%B2%8C%EC%8B%9C%ED%8C%90_%EB%A7%8C%EB%93%A4%EA%B8%B0.pptx

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6711371c-353a-4ace-b411-995d463ee56b/Spring_%EA%B2%8C%EC%8B%9C%ED%8C%90_%EB%A7%8C%EB%93%A4%EA%B8%B0.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/86fdbe4a-b5cf-4923-808f-57c5de4c58b4/Spring_%EA%B5%90%EC%9E%AC.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fea33598-ac2f-43fd-86bc-6e49c05c2562/%EB%8B%B5%EB%B3%80%ED%98%95_%EA%B2%8C%EC%8B%9C%ED%8C%90.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/17d76024-dae2-45c0-85e7-4e1aa328907a/%EC%8A%A4%ED%94%84%EB%A7%81%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8%EB%A7%8C%EB%93%A4%EA%B8%B0.txt

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a7824fa2-9e9c-4887-bf70-5a660bfb3023/jstl%EC%82%AC%EC%9A%A9%EB%B2%95-bambabo(%EC%88%98%EC%A0%95%EC%98%88%EC%A0%9C%EC%B6%94%EA%B0%80).doc

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0c0349ed-523f-48e0-94f9-de043d55b260/spring_framework_%EA%B8%B0%EB%B3%B8%EA%B0%9C%EC%9A%94.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/dd029133-b483-4cad-8240-899ffe3f6494/apache-tomcat-9.0.38-windows-x64.zip

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/109a4d3a-2799-4ebe-9b87-b05f111835f1/spring-tool-suite-3.9.12.RELEASE-e4.15.0-win32-x86_64.zip

	spring	spring boot
사용	자바 기반 어플리케이션을 만드는 데 사용됨	주로 REST API 개발을 위해 사용됨
주요기능	의존성 주입	AutoConfiguration
개발 유형	느슨하게 결합된 어플리케이션을 만드는데 도움이 된다.	독립 실행형 어플리케이션을 만드는데 도움이 된다.
서버 종속성	서버를 명시적으로 설정해야 함	Tomcat이나 Jetty와 같은 임베디드 서버를 제공
구성	수동으로 구성을 빌드해야 함	부트스트랩 가능한 기본 구성이 있음

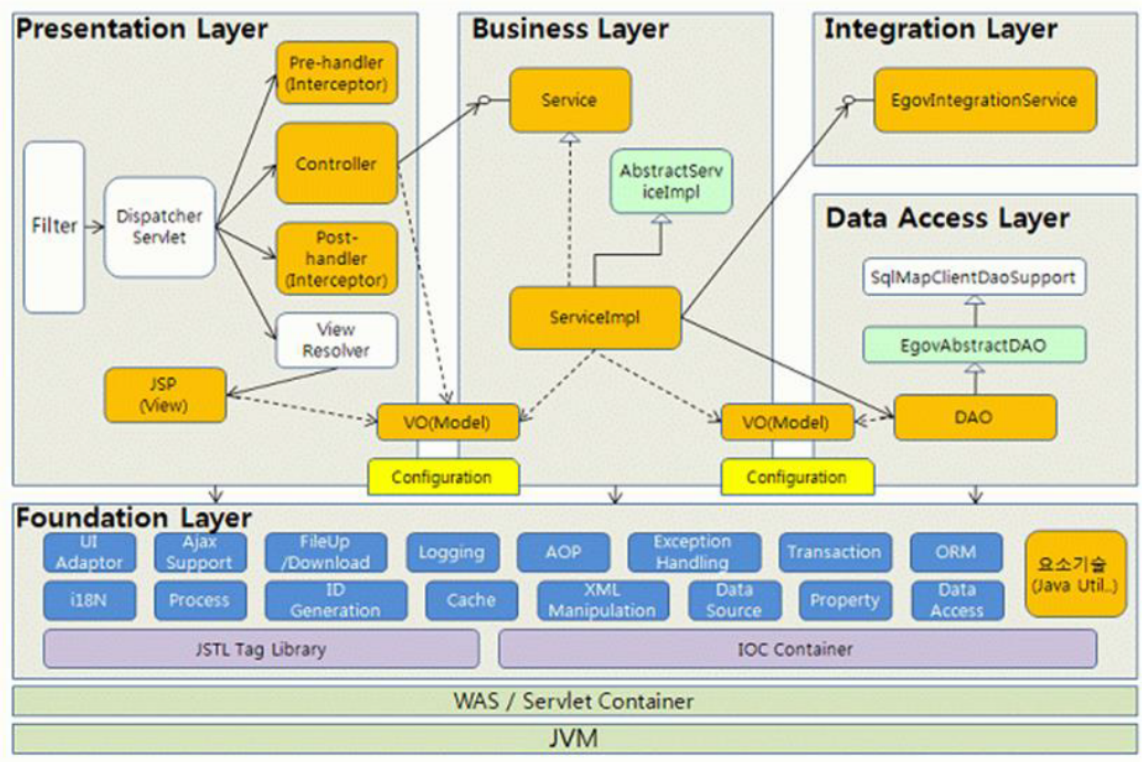
https://yamyam-spaghetti.tistory.com/56

- json 응답을 기본으로 함 → 스프링부트
- 규모가 큰 쪽 → 스프링프레임워크

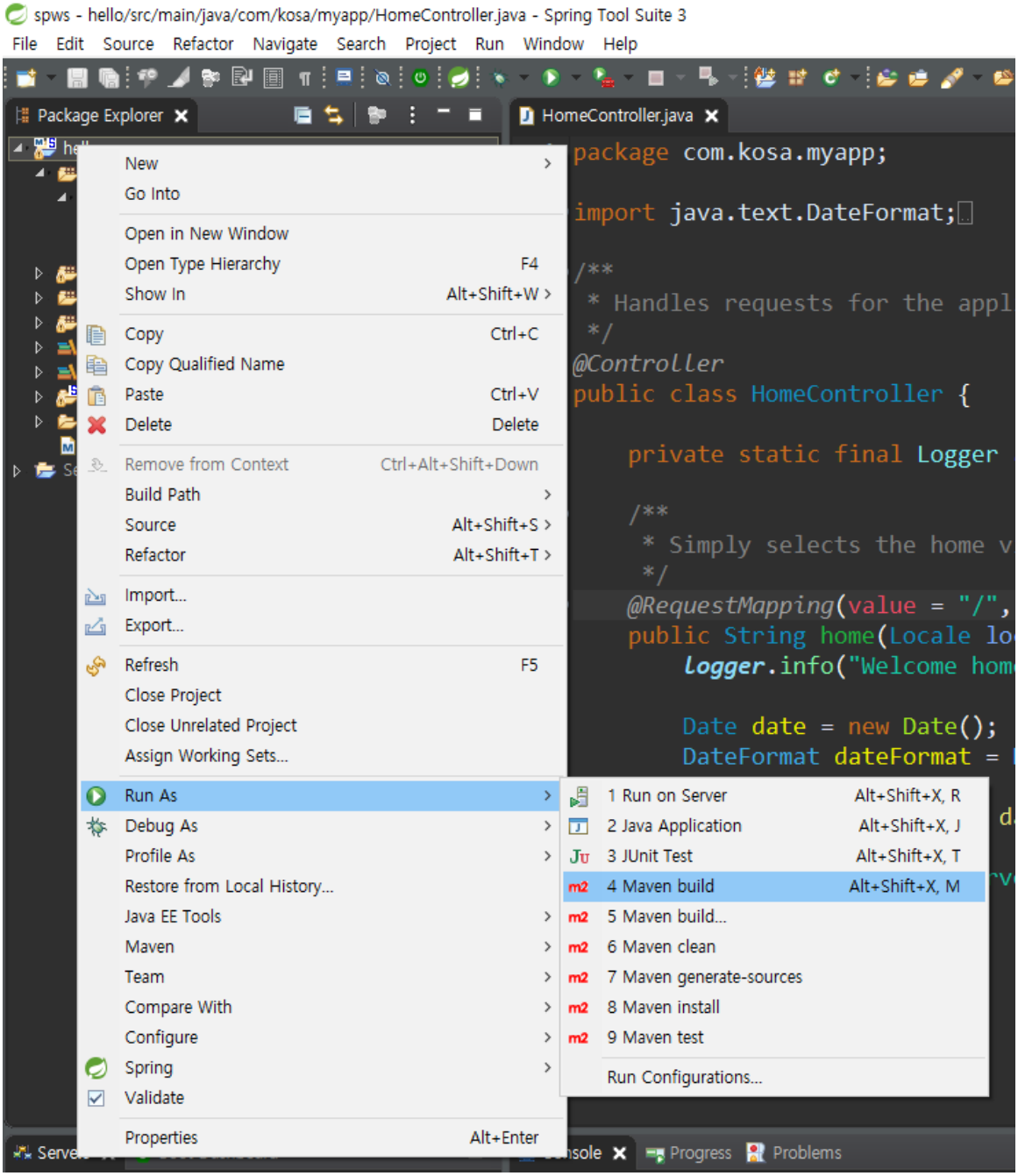
- 스프링은 전체 경로에(폴더) 한글 이름 있으면 안 됨 → 로그 인식이 안 됨

- 국가기관, 금융권, 학교, 대기업 등 대부분은 스프링 프레임워크나 스프링 부트(jsp 엔진이 사라짐, restful api 서버 전용) 등을 사용한다.
- 국가기관은 전자정부 프레임워크 ⇒ 스프링 프레임워크의 일종

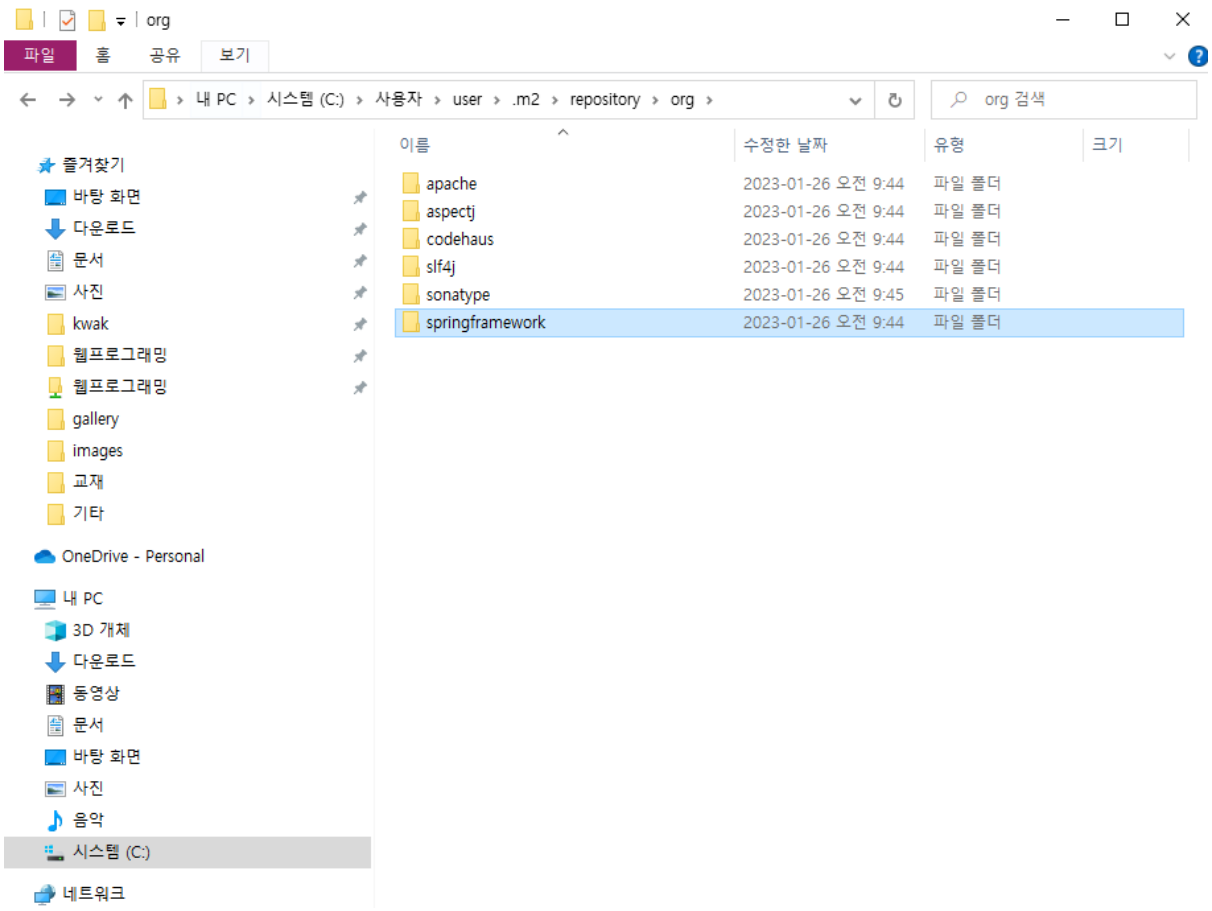
- 스프링4부터 스프링 부트 본격적으로 지원
- 스프링3은 스프링 프레임워크 위주



스프링 기본 구조



- maven: 배포 툴(프레임워크에서 주로 사용할 예정)
  - 모든 jar 파일을 묶어놓고 그 공간에 대한 정보를 xml 형태로 만들어 웹에서 한번에 복사, 붙여넣기



다운로드 받다가 오류 날 경우 → repository 지웠다 다시 깔기

```
https://mvnrepository.com/
```

→ 이 사이트에서 maven 링크 복사 해서 pom.xml에 붙여넣기: jar 파일 자동 생성

- pom.xml : 프로젝트 설정
  - 프로젝트 빌딩(컴파일과 배포)과 관련된 내용이 저장된다. maven이 사용한다.
- web.xml : 웹 설정
  - 사이트 운영정보에 관련된 부분 담당.
  - 필터, 리스너, 서블릿 설정
  - 필터 부분만 프로그래밍 해주면 됨.
- root-context.xml : db 관련(DB와 트랜잭션 전담)
  - DB 관련(디비 객체 만들고), MyBatis 연결, 트랜잭션 처리
- servlet-context.xml : 스프링 관련 설정 파일들
  - 스프링 프레임워크

- @Controller, @Service, @Repository
- 세개의 어노테이션(↑↑↑)이 있는 클래스의 객체들을 자동으로 만들도록 지시한다.
- <annotation-driven/> 태그가 지정되면 위의 3개의 어노테이션을 찾아 객체를 생성한다.
- `<context:component-scan base-package="com.kosa.myapp" />`
  - 어디서부터 클래스를 찾아볼 지를 지정한다.
  - base-package에 저장된 패키지보다 아래를 전부 순회하면서 찾아낸다.
- `<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">`

  `<beans:property name="prefix" value="/WEB-INF/views/" />`
  `<beans:property name="suffix" value=".jsp" />`
  `</beans:bean>`
  - beans: 콩깍지, bean: 콩
  - 자바 만든 사람들이 객체를 콩으로 비유. 객체 모아놓으면 콩깍지
  - InternalResourceViewResolver 클래스를 객체를 만들어라
  - property : 변수명
  - InternalResourceViewResolver 클래스 안에 prefix, suffix 두 개의 변수가 있고 변수에 값 설정
  - jsp 페이지를 WEB-INF/views 폴더 아래에 둔다.
  - 예전에는(자바 1.5 이전버전에서) xml 파일에 다 모든 객체에 대한 정보를 기술해야 한다.
- url → DispatchServlet 클래스가 다 받아서 url과 맵핑되는 함수를 갖고 있는 클래스 객체를 전달한다. (Controller를 찾는다).
  - Controller의 메소드로부터 반환하는 문자열을 jsp 파일로 본다.
  - 하나의 컨트롤러에 여러개의 함수를 만들고 각 함수마다 url을 부여할 수 있다.
  - ```
    @RequestMapping(value=”url”, method=RequestMethod.GET)
    public String 함수명(String user_id, UserDto dto){
    ….
    return “jsp 페이지명”;
    }
    ```
  - <input name=”user_id” ⇒ 파라미터로 전달이 온다.

▼ xml + 설명

hello/pom.xml ⇒ 프로젝트 설정 파일

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.kosa</groupId>
  <artifactId>myapp</artifactId>
  <name>hello</name>
  <packaging>war</packaging>
  <version>1.0.0-BUILD-SNAPSHOT</version>
  <properties>
```
```

```
    <java-version>1.6</java-version>
    <org.springframework-version>3.1.1.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
  </properties>
  <dependencies> <!-- 필요한 라이브러리 놓는 곳 -->
    <!-- Spring -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${org.springframework-version}</version>
      <exclusions>
        <!-- Exclude Commons Logging in favor of SLF4j -->
        <exclusion>
          <groupId>commons-logging</groupId>
          <artifactId>commons-logging</artifactId>
         </exclusion>
      </exclusions>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${org.springframework-version}</version>
    </dependency>

    <!-- AspectJ -->
    <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjrt</artifactId>
      <version>${org.aspectj-version}</version>
    </dependency>

    <!-- Logging -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>${org.slf4j-version}</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>jcl-over-slf4j</artifactId>
      <version>${org.slf4j-version}</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>${org.slf4j-version}</version>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.15</version>
      <exclusions>
        <exclusion>
          <groupId>javax.mail</groupId>
          <artifactId>mail</artifactId>
        </exclusion>
        <exclusion>
          <groupId>javax.jms</groupId>
          <artifactId>jms</artifactId>
        </exclusion>
        <exclusion>
          <groupId>com.sun.jdmk</groupId>
          <artifactId>jmxtools</artifactId>
        </exclusion>
        <exclusion>
          <groupId>com.sun.jmx</groupId>
          <artifactId>jmxri</artifactId>
        </exclusion>
      </exclusions>
      <scope>runtime</scope>
    </dependency>
```

```
    <!-- @Inject -->
    <dependency>
      <groupId>javax.inject</groupId>
      <artifactId>javax.inject</artifactId>
      <version>1</version>
    </dependency>

    <!-- Servlet -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>

    <!-- Test -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.7</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
    <build>
        <plugins>
            <plugin>
                <artifactId>maven-eclipse-plugin</artifactId>
                <version>2.9</version>
                <configuration>
                    <additionalProjectnatures>
                        <projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
                    </additionalProjectnatures>
                    <additionalBuildcommands>
                        <buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
                    </additionalBuildcommands>
                    <downloadSources>true</downloadSources>
                    <downloadJavadocs>true</downloadJavadocs>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.5.1</version>
                <configuration>
                    <source>1.6</source>
                    <target>1.6</target>
                    <compilerArgument>-Xlint:all</compilerArgument>
                    <showWarnings>true</showWarnings>
                    <showDeprecation>true</showDeprecation>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.2.1</version>
                <configuration>
                    <mainClass>org.test.int1.Main</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app_2_5.xs
d">

  <!-- filter: 한글 안 깨지게하는 필터. request.setCharacterEncoding("utf-8") 안해도 됨 -->
  <filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter
    </filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
      <param-name>forceEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>

  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!-- The definition of the Root Spring Container shared by all Servlets
    and Filters -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
=> db 관련 작업하는 xml 문서 루트 지정
  </context-param>

  <!-- Creates the Spring Container shared by all Servlets and Filters -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <!-- Processes application requests -->
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
=> 객체 관리 도맡는 역할 dispatcher
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml
      </param-value>
=> dispatcher 관련한 거 정리한 파일 servlet-context.xml
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans https://www.springframework.org/schema/be
ans/spring-beans.xsd">

  <!-- Root Context: defines shared resources visible to all other web components -->
    <!-- DB와 트랜잭션 전담 -->
</beans>
```

servlet-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/
spring-mvc.xsd
    http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.x
sd
    http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-con
text.xsd">

<!-- 스프링 관련 설정 파일들. DB만 root-context.xml에서 담당한다. -->

  <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->

  <!-- Enables the Spring MVC @Controller programming model

  servlet board.do?cmd=list
  controller -- view하고 model을 연결시킨다
  service -- 모델단
  dao -- 모델단

  service는 한 화면에서 여러개의 테이블에 데이터를 한 꺼번에 넣거나 단순히 디비에 데이터 읽고 쓰기 말고 복잡한 작업이 필요한 경우에
  dao는 테이블 하나 당, 그리고 단순히 데이터 읽고 쓰기 담당
  controller는 복잡한 업무는 하면 안 됨. => 서비스한테 작업을 넘기면 서비스가 dao를 통해 데이터 읽고 쓰기를 한다.

  예) 사장(컨트롤러)---노동자(dao) : 일의 규모가 작을 때는 괜찮음
  일이 점점 커지면 =>
  사장(컨트롤러)---중간관리자(서비스)---노동자1(dao 1)....n(dao n)

  스프링: DI(Dependency Injection) 의존성 주입

  예)  목욕탕: 손님이 목욕하러 오면 그 때서야 옷 보관함을 만들기 -> 그 손님 나가면 옷 보관함 파괴 (이 과정을 반복)
    => 이클립스에서 했던 mvc 구조

    보관함을 미리 많이 만들어놓고 옷 보관함별로 번호표를 만들어서 번호표를 손님에게 배부
    => 프로그램도 이렇게 만들자: 컨트롤러, 서비스, dao 미리 만들어 놓기 -> 사용 요청을 하면 참조를 전달, 그리고 반납하고 나감

  스프링이 하는 주 역할은 객체를 미리 만들어서 관리하는 역할
      우리가 만들어서 스프링한테 주는 게 아니라 스프링이 객체를 만들어서 우리한테 준다(DI, 제어의 역행)
    => xml 파일에 객체 생성해서 연결해줄 것을 지정했었음 -> 자바 1.5 이전까지 (오로지 xml로만)
    => 자바 1.5가 만들어지면서 제네릭(generic), 어노테이션(annotation)이 추가됨

      @Controller
      @Service
      @Repository (dao)
      annotation-driven : 프로젝트를 다 뒤져가면서 위의 세개의 어노테이션 써있는거 객체를 만들어서 보관하라
    context: component-scan base-pacakge-"com.kosa.myapp"
      com.kosa.myapp 보다 아래에 있는 폴더만 적용 대상
   -->
  <annotation-driven />

  <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${weba
ppRoot}/resources directory -->
  <resources mapping="/resources/**" location="/resources/" />
```

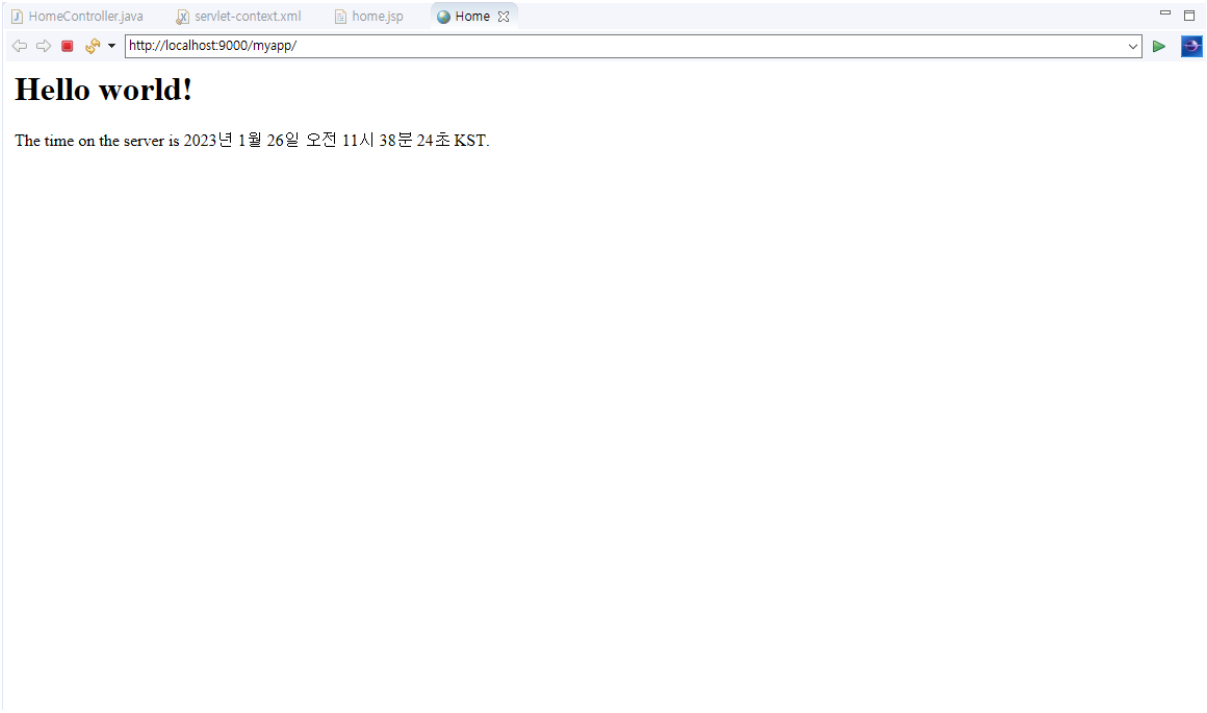
```
<!--
  CSS나 이미, 자바스크립트 url을 외부로 내보내기
 -->

  <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views direct
ory -->
  <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
  </beans:bean>

  <context:component-scan base-package="com.kosa.myapp" />



</beans:beans>
```



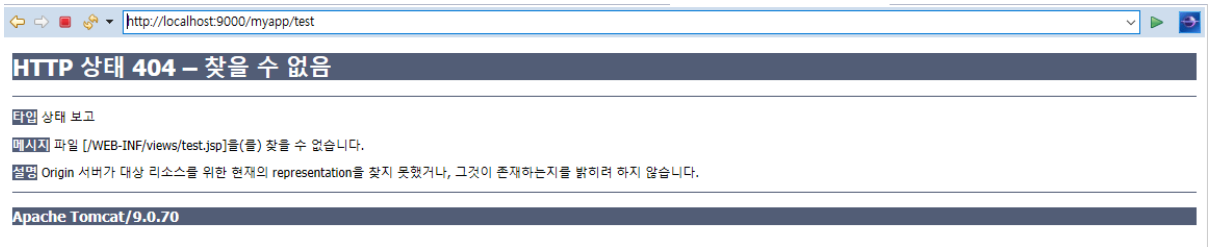
`TestController`

```
package com.kosa.myapp;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class TestController {
//  url : http://127.0.0.1:9000/test
  @RequestMapping(value = "/test", method = RequestMethod.GET)
  public String test() {
    System.out.println("test");
    return "test";
```

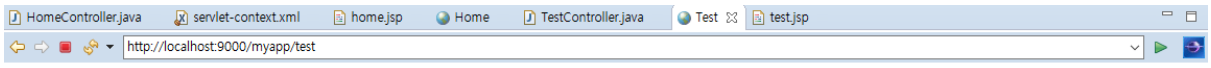
```
  }
}
```



→ test.jsp 파일을 안 만들어줘서 이동은 했지만 오류가 남

test.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Test</title>
</head>
<body>
  <h1>First Spring Framework</h1>
</body>
</html>
```



First Spring Framework

```
package com.kosa.myapp;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
```

```java
public class TestController {
//  url : http://127.0.0.1:9000/test
  @RequestMapping(value = "/test", method = RequestMethod.GET)
  public String test() {
    System.out.println("test");
    return "test";
  }

//  url: http://127.0.0.1:9000/myapp/test2?user_id=test&password=1234
  @RequestMapping(value = "/test2", method = RequestMethod.GET)
  public String test2(String user_id, String password, Model model, TestDto dto) {
//    Model : request 객체 대신에 jsp로 값 전달할 목적으로 사용함 => 이게 없으면 jsp에서 값을 전달 못 받음.
    model.addAttribute("user_id", user_id);
    model.addAttribute("password", password);

    System.out.println("user_id: "+user_id);
    System.out.println("password: "+password);

    System.out.println("user_id: "+dto.getUser_id());
    System.out.println("password: "+dto.getPassword());

    return "test2";
  }

}
```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>test2</title>
</head>
<body>
<h1>test 2</h1>
<h6>값 전달 방식</h6>
<h3><%=request.getAttribute("user_id") %></h3>
<h3>표현식: ${password}</h3>
<!-- 표현식은 for문을 못 씀 -->
</body>
</html>
```

test 2

```
package com.kosa.myapp;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class TestController {
//  url : http://127.0.0.1:9000/test
  @RequestMapping(value = "/test", method = RequestMethod.GET)
  public String test() {
    System.out.println("test");
    return "test";
  }

//  url: http://127.0.0.1:9000/myapp/test2?user_id=test&password=1234
  @RequestMapping(value = "/test2", method = RequestMethod.GET)
  public String test2(String user_id, String password, Model model, TestDto dto) {
//    Model : request 객체 대신에 jsp로 값 전달할 목적으로 사용함 => 이게 없으면 jsp에서 값을 전달 못 받음.
    model.addAttribute("user_id", user_id);
    model.addAttribute("password", password);

    System.out.println("user_id: "+user_id);
    System.out.println("password: "+password);

    System.out.println("user_id: "+dto.getUser_id());
    System.out.println("password: "+dto.getPassword());

    return "test2";
  }

  @RequestMapping(value = "/add", method = RequestMethod.GET)
  public String add(HttpServletRequest req, Model model) {
    int x = Integer.parseInt(req.getParameter("x"));
    int y = Integer.parseInt(req.getParameter("y"));

    model.addAttribute("x",x);
    model.addAttribute("y",y);
    model.addAttribute("result",x+y);

    return "add";
  }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>add</title>
</head>
<body>
<h6>${x} + ${y} = ${result}</h6>
</body>
</html>
<!-- ${}: spring에서만 쓸 수 있는 문법은 아님. servlet에서도 가능 -->
=> 주석에 ${} 넣으면 터져
```

783 + 291 = 1074

```
package com.kosa.myapp;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class TestController {
//  url : http://127.0.0.1:9000/test
  @RequestMapping(value = "/test", method = RequestMethod.GET)
  public String test() {
    System.out.println("test");
    return "test";
  }

//  url: http://127.0.0.1:9000/myapp/test2?user_id=test&password=1234
  @RequestMapping(value = "/test2", method = RequestMethod.GET)
  public String test2(String user_id, String password, Model model, TestDto dto) {
//    Model : request 객체 대신에 jsp로 값 전달할 목적으로 사용함 => 이게 없으면 jsp에서 값을 전달 못 받음.
    model.addAttribute("user_id", user_id);
    model.addAttribute("password", password);

    System.out.println("user_id: "+user_id);
    System.out.println("password: "+password);

    System.out.println("user_id: "+dto.getUser_id());
    System.out.println("password: "+dto.getPassword());

    return "test2";
  }

  @RequestMapping(value = "/add", method = RequestMethod.GET)
  public String add(HttpServletRequest req, Model model) {
    int x = Integer.parseInt(req.getParameter("x"));
    int y = Integer.parseInt(req.getParameter("y"));

    model.addAttribute("x",x);
    model.addAttribute("y",y);
    model.addAttribute("result",x+y);

    return "add";
  }


  @RequestMapping(value = "/sub/{x}/{y}", method = RequestMethod.GET) //데이터 입력&표시 방법 변경(http://127.0.0.1/m
yapp/sub/343/231)
  public String sub(@PathVariable("x")int x, @PathVariable("y")int y, Model model) {
    System.out.println("x= "+x);
    System.out.println("y= "+y);

    model.addAttribute("x",x);
    model.addAttribute("y",y);
    model.addAttribute("result",x-y);
```

```
    return "sub";
  }


}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>sub</title>
</head>
<body>
<h3>${x} - ${y} = ${result}</h3>
</body>
</html>
```

45 - 34 = 11

```
package com.kosa.myapp;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller //이거 안 쓰면 오류!!!!!!
public class ShapeController {

  @RequestMapping(value = "/circle/{x:.+}", method = RequestMethod.GET)
//{x:.+} -> double로 받을 수 있게.
  public String circle(@PathVariable("x") double x, Model model) {
    System.out.println("x= "+x);

    model.addAttribute("x", x);
    model.addAttribute("resultCircle", x*x*3.14);

    return "circle";
  }

  @RequestMapping(value = "/rectangle/{a}/{b}", method = RequestMethod.GET)
  public String rectangle(@PathVariable("a") int a, @PathVariable("b") int b, Model model) {
    System.out.println("a="+a);
    System.out.println("b="+b);

    model.addAttribute("a", a);
    model.addAttribute("b", b);
    model.addAttribute("resultRectangle", a*b);
    return "rectangle";
  }

  @RequestMapping(value = "/triangle/{c}/{d}", method = RequestMethod.GET)
```

```
  public String triangle(@PathVariable("c") int c, @PathVariable("d") int d, Model model) {
    System.out.println("c="+c);
    System.out.println("d="+d);

    model.addAttribute("c", c);
    model.addAttribute("d", d);
    model.addAttribute("resultTriangle", c*d/2);
    return "triangle";
  }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>circle</title>
</head>
<body>
<h1>반지름 ${x }의 면적: ${resultCircle }</h1>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>rectangle</title>
</head>
<body>
<h1>가로 ${a }, 세로 ${b }의 면적: ${resultRectangle }</h1>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>triangle</title>
</head>
<body>
<h1>가로 ${c }, 높이 ${d }의 면적: ${resultTriangle }</h1>
</body>
</html>
```

post/add

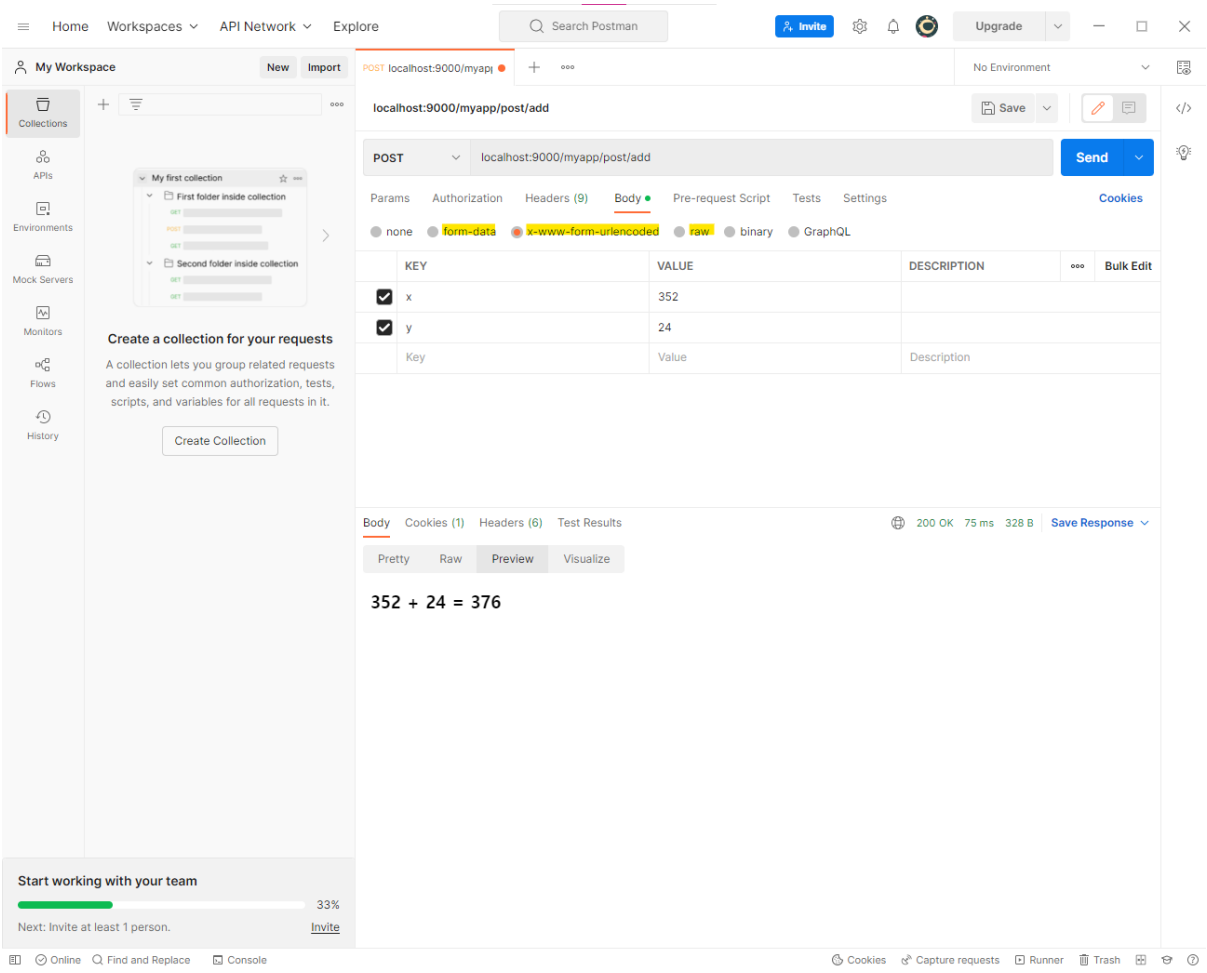
```
package com.kosa.myapp;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class BoardController {

  @RequestMapping(value = "/post/add", method = RequestMethod.POST)
  public String add(int x, int y, Model model) {
    model.addAttribute("x", x);
    model.addAttribute("y", y);
    model.addAttribute("result", x+y);

    return "post/add";
  }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>add</title>
</head>
<body>
<h3>${x} + ${y} = ${result}</h3>

</body>
</html>
```

- get 방식이나 post 방식으로 접근할 때 테스트용 도구

- curl
  - 카카오톡 서비스제공이나 구글 서비스 제공할 때 설명을 저 프로그램으로 해놓는다.
  - curl 프로그램은 사용은 좀 불편하지만 리눅스나 윈도우 둘 다 가능. 대중적으로 더 많이 사용
- postman
  - 윈도우 os만
- insomnia
  - 무료

POST Man (포스트 맨)

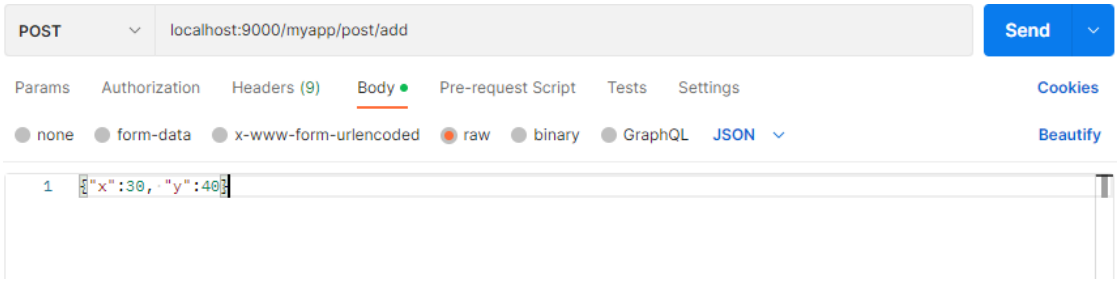


- get → params에 파라미터 추가
- ***post***
  - **form-data**: 파일 전송 시

    ```
    enctype="multipart/form-data"
    ```

    → 이 방식을 시뮬레이션 할 때
  - **x-www-form-urlencoded**: 보통의 form 전송 시
  - **raw** (- json): json 전송방식

    ```
    {
    name:"홍길동", age:23
    }
    ```

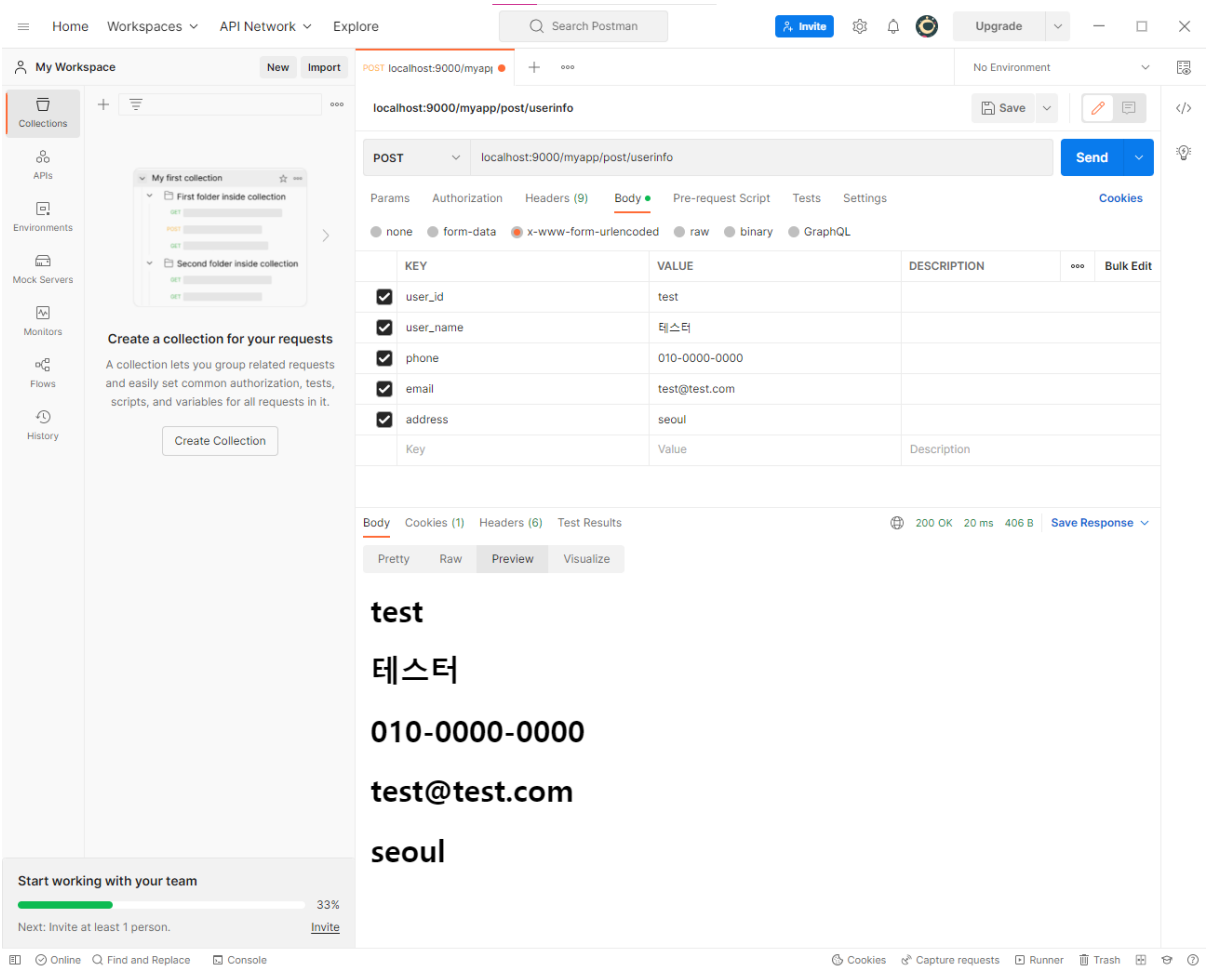


UserController

- UserDto 에 변수 넣고
  - user_id
  - user_name
  - phone
  - email
  - address
- /userinfo.jsp 만들어서 뿌리기

```
package com.kosa.myapp;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class UserController {


  @RequestMapping(value = "/post/userinfo", method = RequestMethod.POST)
  public String userinfo(UserDto dto, Model model) {
    model.addAttribute("user_id", dto.getUser_id());
    model.addAttribute("user_name", dto.getUser_name());
    model.addAttribute("phone", dto.getPhone());
    model.addAttribute("email", dto.getEmail());
    model.addAttribute("address", dto.getAddress());

    return "post/userinfo";
  }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>user info</title>
</head>
<body>
<h1>${user_id }</h1>
<h1>${user_name }</h1>
<h1>${phone }</h1>
<h1>${email }</h1>
<h1>${address }</h1>
</body>
</html>
```



- dto 통째로 넘겨주기

```
package com.kosa.myapp;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class UserController {


  @RequestMapping(value = "/post/userinfo", method = RequestMethod.POST)
  public String userinfo(UserDto dto, Model model) {
    model.addAttribute("dto", dto);


    return "post/userinfo";
  }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>user info</title>
</head>
<body>
<h1>${dto.getUser_id() }</h1>
<h1>${dto.getUser_name() }</h1>
<h1>${dto.getPhone() }</h1>
<h1>${dto.getEmail() }</h1>
<h1>${dto.getAddress() }</h1>
</body>
</html>
```