Spring Web MVC

-Spring Web MVC 는 서블릿 API를 기반으로 만들어진 웹프레임워크입니다
-공식이름은 Spring Web MVC이지만 Spring MVC라는 이름으로 더 많이 부릅니다.
-Spring MVD에는 웹애플리케이션 개발을 위한 다양한 라이브러리가 포함되어 있으며 이를 통해 반복적인 작업을 상당히 줄일 수 있어 프로젝트 수행의 생산성 및 유지보수성을 높일 수 있다.

[개발방식]
-Spring MVC개발 방식은 springframework와 xml을 이용한 방법과 자바 어노테이션을 이용하는 방법을 제공한다
-springframework에 살펴봤던 내용은 모두 자바 어노테이션을 이용하는 방법을 사용할 것이며 spring MVC에 관련된 내용만 xml, 자바 어노테이션 두가지 모두 살펴보도록 하겠다.

[선수과정]
-java
-servlet/jsp
-spring framework
-html,css,javascript,jquery
-oracle database

Spring MVC 설치프로그램

Servlet/JSP + Spring MVC Library

[필요한 프로그램]
-JDK 8 (http://oracle.com)
  환경설정:JAVA_HOME c:\program files\java\jdk1.8
-Eclipse (2020-09R)
-Apache tomcat(9.0) (http://tomcat.apache.org)
-Oracle Database 11g

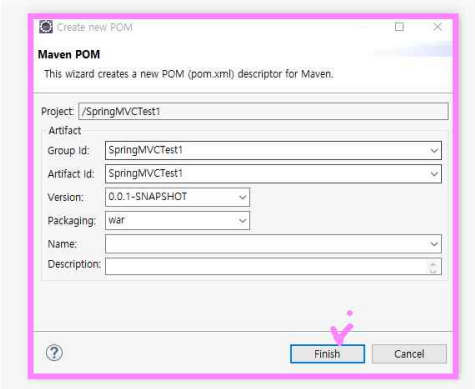
[MVC개념]
-MVC(Model-View-Controller)는 소프트웨어 엔지니어링에서 사용자 인터페이스와 애플리케이션 로직을 분리하는데 사용되는 패턴이다.
-Model은 애플리케이션의 비즈니스 계층을 정의하고, controller는 애플리케이션의 흐름을 관리하며 view는 애플리케이션의 프리젠테이션 계층을 정의
-MVC패턴은 웹프로그램에 매우 적합함
-Java컨텍스트에서 Model은 간단한 java클래스로 구성되고 controller는 서블릿으로 구성되며 View는 jsp페이지로 구성된다

DispatcherServlet
-Servlet/JSP에서 사용자 요청이 발생하면 이 요청 정보를 해석하고 개발자가 만든 코드를 동작시키는 첫 번째 서블릿이다
-Spring MVC는 DispatcherServlet을 확대하여 Srping Framework가 가지고 있는 기능을 사용할 수 있도록

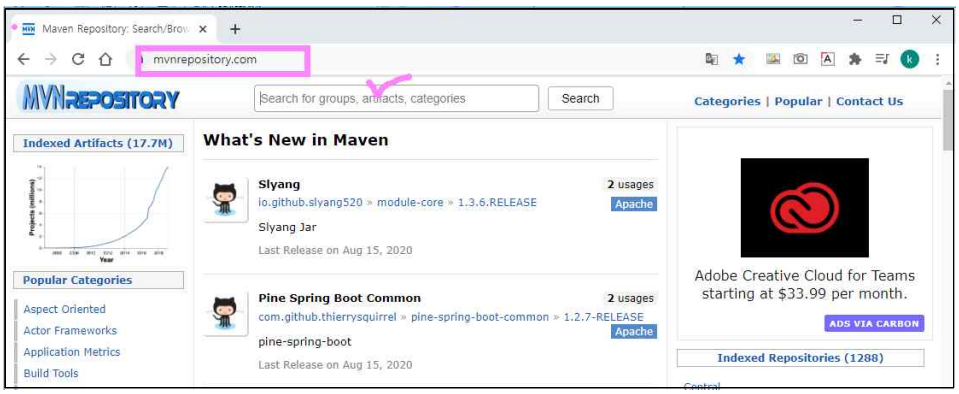
이 클래스를 재정의하고 있다
-Spring MVC프로젝트 설정에서 가장 먼저 해야하는 DispatcherServlet 클래스를 Spring MVC에서 재정의한 클래스로 설정하는 일이다
-Spring MVC프로젝트를 설정하는 방식은 XML을 이용하는 방법과 Java코드를 활용하는 방법이 있다
-두 방법 모두 설정방법이 매우 유사하다

1. XML방법으로 스프링 설정

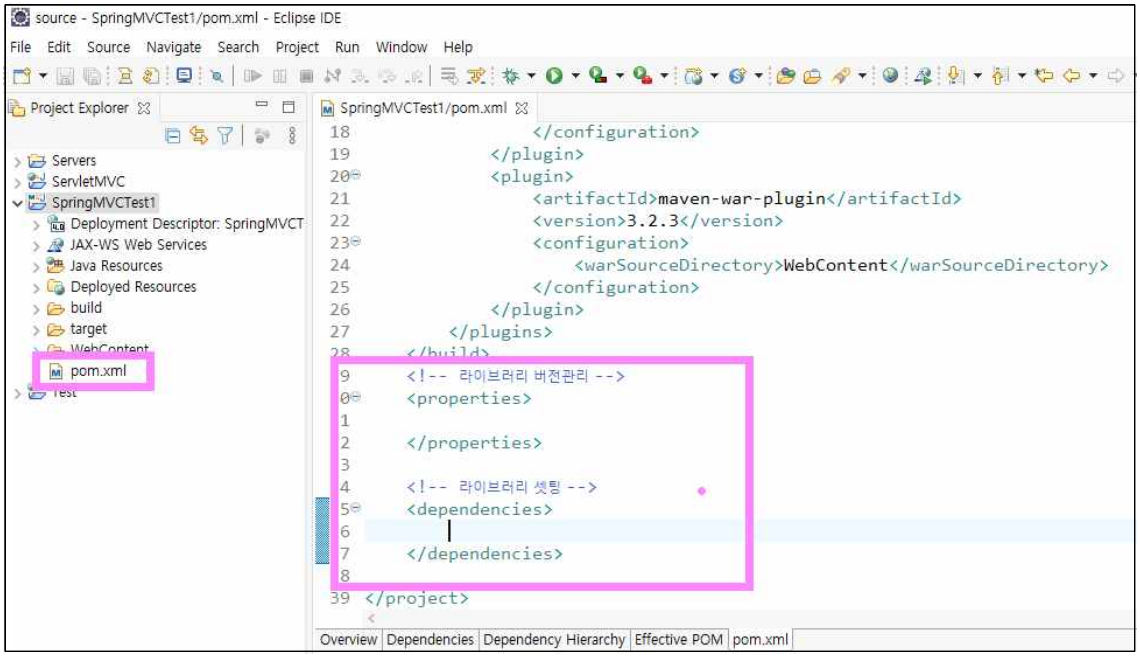
[실습1] Project Name : Hello
(1) Maven Project 로 변환
[Hello] 프로젝트 - 우클릭- Configure - Convert to Maven Project



(2) pom.xml 등록
사이트 : https://mvnrepository.com/



검색: servlet, jsp , jstl , spring web mvc




```
<properties>
		<javax.servlet-version>4.0.1</javax.servlet-version>
		<javax.servlet.jsp-version>2.3.3</javax.servlet.jsp-version>
		<javax.servlet-jstl-version>1.2</javax.servlet-jstl-version>
		<org.springframework-version>5.2.8.RELEASE</org.springframework-version>
	</properties>

	<dependencies>
		<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>javax.servlet-api</artifactId>
			<version>${javax.servlet-version}</version>
			<scope>provided</scope>
		</dependency>
		<!-- https://mvnrepository.com/artifact/javax.servlet.jsp/javax.servlet.jsp-api -->
		<dependency>
			<groupId>javax.servlet.jsp</groupId>
			<artifactId>javax.servlet.jsp-api</artifactId>
			<version>${javax.servlet.jsp-version}</version>
			<scope>provided</scope>
		</dependency>
		<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>jstl</artifactId>
			<version>${javax.servlet-jstl-version}</version>
		</dependency>

		<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
			<version>${org.springframework-version}</version>
		</dependency>

	</dependencies>
</project>
```

(3) web.xml 설정

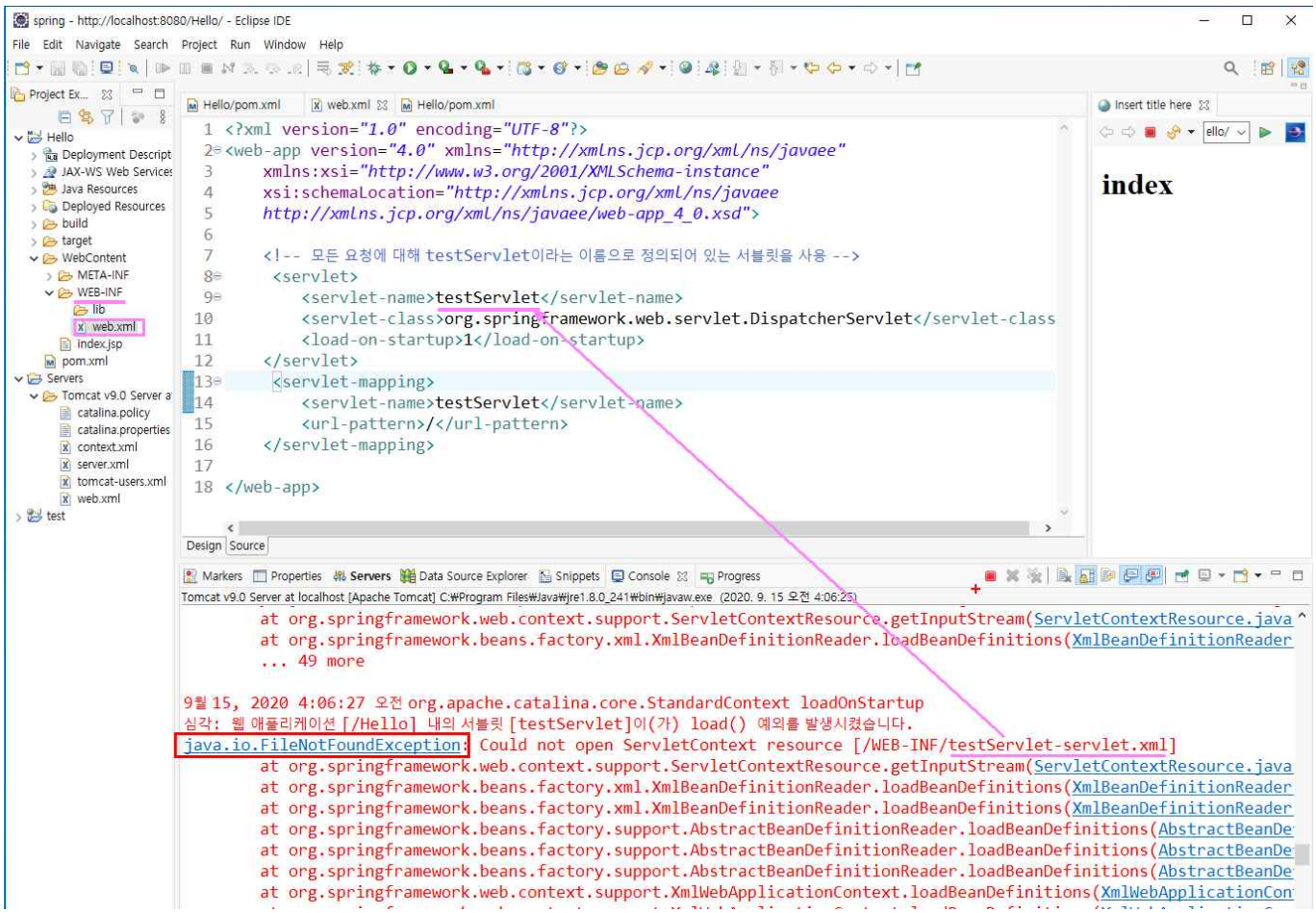
DispatcherServlet 클래스는 Springframework 에서 제공하는 클래스를 사용한다.

servlet-name으로 지정된 이름은 **웹애플리케이션 설정 파일**로 해석된다. 아래의 **testServlet** 이라는 이름으로 지정할 경우 웹애플리케이션 설정 파일로 **testServlet-servlet.xml** 이란 이름으로 인식되어 로딩된다.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

        <!-- 모든 요청에 대해 testServlet이라는 이름으로 정의되어 있는 서블릿을 사용 -->
     <servlet>
        <servlet-name>testServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
     <servlet-mapping>
        <servlet-name>testServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>
```

(4) index.jsp 생성

```jsp
<body>
    <h1>index</h1>
</body>
```



(4) web.xml 수정하시오

웹애플리케이션 설정 파일을 servlet-context.xml 로 지정하고, 프로젝트 수행시에 사용할 Bean들을 정의하는 파일은 root-context.xml 로 지정한다.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">

         <!-- 모든 요청에 대해 testServlet이라는 이름으로 정의되어 있는 서블릿을 사용 -->
    <servlet-mapping>
        <servlet-name>testServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

     <servlet>
        <servlet-name>testServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
          <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/config/servlet-context.xml</param-value>
          </init-param>

        <load-on-startup>1</load-on-startup>
    </servlet>

<context-param>
         <param-name>contextConfigLocation</param-name>
         <param-value>/WEB-INF/config/root-context.xml</param-value>
</context-param>

    <listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

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


</web-app>
```

(1) DispatcherServlet클래스는 Spring에서 제공하는 DispatcherServlet 클래스를 사용한다

(2) 웹애플리케이션 설정 파일은 servlet-context.xml로 지정

(3) 프로젝트 수행시 사용할 Bean들을 정의하는 파일은 root-context.xml로 지정

(4) listener가 동작할때 (2)(3)의 파일을 자동로딩

(5) 파라미터에 한글이 있을 경우 인코딩설정

(5) /WEB-INF/config/**root-context.xml** 내용편집

```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
         http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>
```

(6) /WEB-INF/config/**servlet-context.xml** 내용편집

```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
                                    http://www.springframework.org/schema/mvc/spring-mvc.xsd
                                    http://www.springframework.org/schema/beans
                                    http://www.springframework.org/schema/beans/spring-beans.xsd
                                    http://www.springframework.org/schema/context
                                    http://www.springframework.org/schema/context/spring-context.xsd">
```

```
</beans:beans>
```

(7) 컨트롤러를 위한 **패키지 만들어** 등록하기

src폴더- **[kr.co.korea.controller]** 패키지 생성하여 아래(8) **servlet-context.xml** 에 등록한다.

(8) servlet-context.xml 내용편집

컨트롤러 패키지 등록

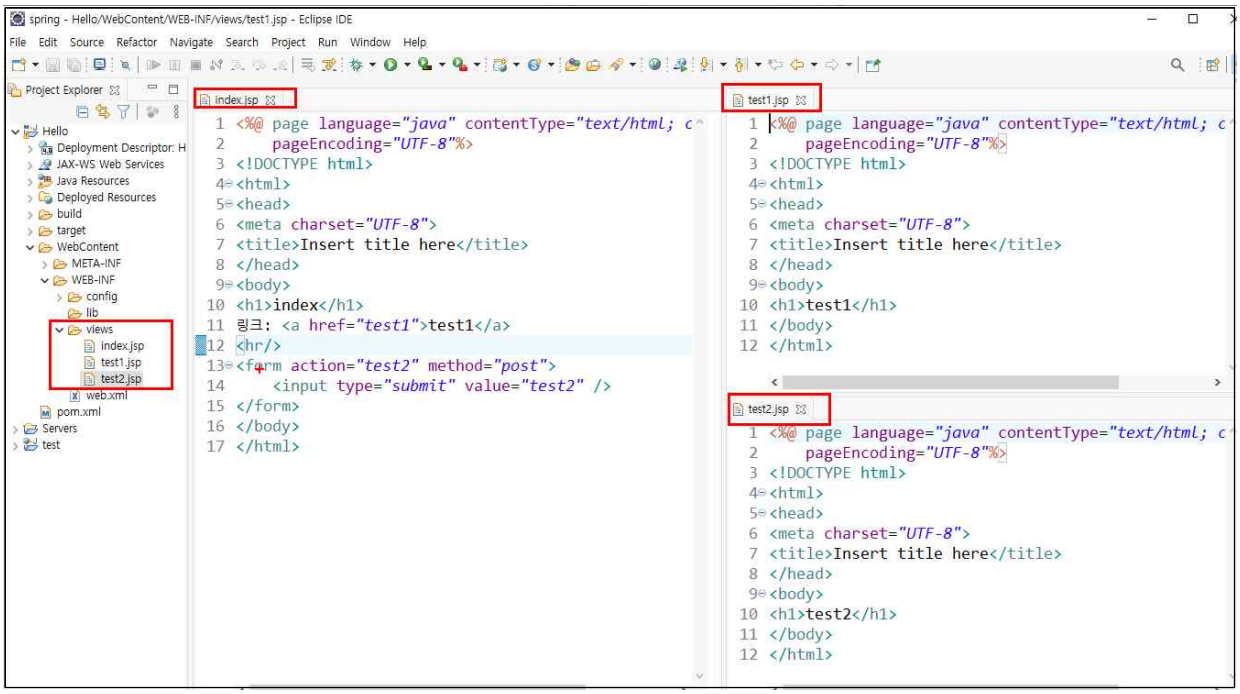
```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:beans="http://www.springframework.org/schema/beans"
                xmlns:context="http://www.springframework.org/schema/context"
                xsi:schemaLocation="http://www.springframework.org/schema/mvc
                        http://www.springframework.org/schema/mvc/spring-mvc.xsd
                        http://www.springframework.org/schema/beans
                        http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/context
                        http://www.springframework.org/schema/context/spring-context.xsd">
    <annotation-driven />
    <context:component-scan base-package="kr.co.korea.controller"></context:component-scan>

</beans:beans>
```

(9) HomeController 클래스 생성

컨트롤러 설정 또는 mapping 등록

```
package kr.co.korea.controller;

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

@Controller
public class HomeController {

        @GetMapping("/")
        public String index() {
                return "index.jsp";
        }

//      @GetMapping("/")
//      public String index() {
//              return "/WEB-INF/views/index.jsp";
//      }

        @GetMapping("/test1")
        public String test1() {
                return "/WEB-INF/views/test1.jsp";
        }

        @PostMapping("/test2")
        public String test2() {
                return "/WEB-INF/views/test2.jsp";
        }
}
```

(10) index.jsp, test1.jsp, test2.jsp 생성



(11) servlet-context.xml 내용편집

컨트롤러 작성이 수행된후 랜더링될 jsp를 설정하고, 로컬의 리소스 파일의 경로 지정

```
    </context:component-scan>

    <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>

    <resources location="/resources/" mapping="/**"></resources>

</beans:beans>
```

(12) HomeController 클래스 편집

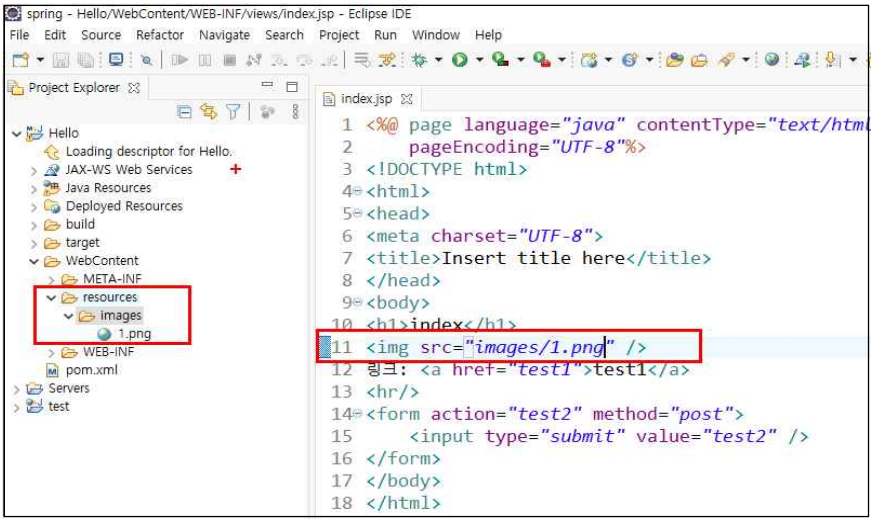
mapping 수정

```
@Controller
public class HomeController {

//      @GetMapping("/")
//      public String index() {
//              return "index.jsp";
//      }

        @GetMapping("/")
        public String index() {
                return "index";
        }

        @GetMapping("/test1")
        public String test1() {
                return "test1";
        }

        @PostMapping("/test2")
        public String test2() {
```

```
                return "test2";
        }
}
```

(13) 로컬의 리소스 파일의 경로 지정

resources 폴더와 그 내부에 images 폴더 생성



2. JAVA 방법으로 스프링 설정

[실습2] Project Name : HelloJava

(1) Maven Project 로 변환

(2) pom.xml

(3) SpringConfigClass.java 생성 (WebApplicationInitializer 인터페이스 상속)

[kr.co.korea.config]패키지-SpringConfigClass 클래스 생성

```
public class SpringConfigClass implements WebApplicationInitializer{

        @Override
        public void onStartup(ServletContext servletContext) throws ServletException {
                // TODO Auto-generated method stub
                // Spring MVC 프로젝트 설정을 위해 작성하는 클래스의 객체를 생성한다.
                AnnotationConfigWebApplicationContext servletAppContext = new
AnnotationConfigWebApplicationContext();
                servletAppContext.register(ServletAppContext.class);

                // 요청 발생 시 요청을 처리하는 서블릿을 DispatcherServlet으로 설정해준다.
                DispatcherServlet dispatcherServlet = new DispatcherServlet(servletAppContext);
                ServletRegistration.Dynamic servlet = servletContext.addServlet("dispatcher", dispatcherServlet);

                // 부가 설정
                servlet.setLoadOnStartup(1);
                servlet.addMapping("/");

                // Bean을 정의하는 클래스를 지정한다
                AnnotationConfigWebApplicationContext rootAppContext = new
AnnotationConfigWebApplicationContext();
                rootAppContext.register(RootAppContext.class);

                ContextLoaderListener listener = new ContextLoaderListener(rootAppContext);
                servletContext.addListener(listener);

                // 파라미터 인코딩 설정
                FilterRegistration.Dynamic filter = servletContext.addFilter("encodingFilter",
CharacterEncodingFilter.class);
                filter.setInitParameter("encoding", "UTF-8");
                filter.addMappingForServletNames(null, false, "dispatcher");
        }
```

```
}
```

(3) SpringConfigClass.java 생성 (AbstractAnnotationConfigDispatcherServletInitializer 인터페이스 상속)

[kr.co.korea.config]패키지-SpringConfigClass 클래스 편집

```
public class SpringConfigClass extends AbstractAnnotationConfigDispatcherServletInitializer{
	// DispatcherServlet에 매핑할 요청 주소를 셋팅한다.
	@Override
	protected String[] getServletMappings() {
		// TODO Auto-generated method stub
		return new String[] {"/"};
	}

	// Spring MVC 프로젝트 설정을 위한 클래스를 지정한다.
	@Override
	protected Class<?>[] getServletConfigClasses() {
		// TODO Auto-generated method stub
		return new Class[] {ServletAppContext.class};
	}

	// 프로젝트에서 사용할 Bean들을 정의기 위한 클래스를 지정한다.
	@Override
	protected Class<?>[] getRootConfigClasses() {
		// TODO Auto-generated method stub
		return new Class[] {RootAppContext.class};
	}

	// 파라미터 인코딩 필터 설정
	@Override
	protected Filter[] getServletFilters() {
		// TODO Auto-generated method stub
		CharacterEncodingFilter encodingFilter = new CharacterEncodingFilter();
		encodingFilter.setEncoding("UTF-8");
		return new Filter[] {encodingFilter};
	}
}
```

(4) RootAppContext 생성

[kr.co.korea.config]패키지- RootAppContext 클래스 생성

```
package kr.co.korea.config;

import org.springframework.context.annotation.Configuration;

//프로젝트 작업시 bean을 정의하는 클래스
@Configuration
public class RootAppContext {

}
```

(5) ServletAppContext 생성 (WebMvcConfigurer 인터페이스 상속)

[kr.co.korea.config]패키지-ServletAppContext 클래스 생성

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;


@Configuration
@EnableWebMvc
@ComponentScan("kr.co.korea.controller")
public class ServletAppContext implements WebMvcConfigurer{

	@Override
	public void configureViewResolvers(ViewResolverRegistry registry) {
		// TODO Auto-generated method stub
		WebMvcConfigurer.super.configureViewResolvers(registry);
		registry.jsp("/WEB-INF/views/",  ".jsp");
	}
	@Override
	public void addResourceHandlers(ResourceHandlerRegistry registry) {
```

```
                // TODO Auto-generated method stub
                WebMvcConfigurer.super.addResourceHandlers(registry);
                registry.addResourceHandler("/**").addResourceLocations("/resources/");
        }

}
```

(6) HomeController 클래스 생성

[kr.co.korea.controller]패키지-HomeController 클래스 생성

(7) views 폴더

(8) resources 폴더

3. Controller의 RequestMapping

[실습3] 컨트롤러 HomeControll, TestController, SubController 생성하여 RequestMapping등록하기

(1) [kr.co.korea.controller]폴더-컨트롤러 HomeControll, TestController, SubController 생성

```
@Controller
public class HomeController {

        @RequestMapping(value="/", method=RequestMethod.GET)
        public String home() {
                return "index";
        }

        @RequestMapping(value="/board/write", method=RequestMethod.GET )
        public String write() {
                return "/board/write";
        }

        @RequestMapping(value="/board", method=RequestMethod.GET )
        public String list() {
                return "/board/list";
        }

}
```

```
@Controller
public class TestController {

        @RequestMapping(value="/test1", method=RequestMethod.GET)
        public String test1() {
                return "/test1";
        }

        @RequestMapping(value="/test2", method=RequestMethod.GET)
        public String test2() {
                return "/test2";
        }
}
```

```
@Controller
public class SubController {

        @RequestMapping(value="/test3", method=RequestMethod.POST)
        public String test3_post() {
                return "test3_post";
        }

        @GetMapping("test4")
        public String test4() {
                return "test4";
        }

        @PostMapping("test5")
        public String test5() {
                return "test5";
        }

        @RequestMapping(value="/test6", method={RequestMethod.GET, RequestMethod.POST})
        public String test7_post() {
                return "test6";
        }
}
```

(2) view폴더-jsp파일 만들기

4. Controller의 RequestMapping에 파라메타 전달

[실습4] 컨트롤러 TestController 생성하여 RequestMapping등록하기

(1) [kr.co.korea.controller]폴더-컨트롤러 TestController 생성

```
@Controller
public class TestController {

	@GetMapping("/test1")
	public String test1(HttpServletRequest request) {  //HttpServletRequest 클래스 통하여 주입
		String id=request.getParameter("id");
		String name=request.getParameter("name");
		String hobby[]=request.getParameterValues("hobby");
		System.out.println("id="+id);
		System.out.println("name="+name);
		for(String str:hobby) {
			System.out.println("hobby:"+str);
		}
		return "result";
	}

	@PostMapping("/test2")
	public String test2(HttpServletRequest request) {
		String id=request.getParameter("id");
		String name=request.getParameter("name");
		String hobby[]=request.getParameterValues("hobby");
		System.out.println("id:"+id);
		System.out.println("name:"+name);
		if(hobby!=null ) {
			for(String str:hobby) {
				System.out.println("hobby:"+str);
			}
		}
		return "result";
	}

	@GetMapping("/test3")
	public String test3(WebRequest request) {  //WebRequest 클래스 통하여 주입
		String id=request.getParameter("id");
		String name=request.getParameter("name");
		String hobby[]=request.getParameterValues("hobby");

		System.out.println("id:"+id);
		System.out.println("name:"+name);
		for(String str:hobby) {
			System.out.println("hobby:"+str);
		}
		return "result";
	}

	@GetMapping("/test4/{id}/{name}/{hobby}")
	public String test4(@PathVariable int id,
			@PathVariable int name,
			@PathVariable int hobby) {
		System.out.println("id:"+id);
		System.out.println("name:"+name);
		System.out.println("hobby:"+hobby);
		int add=id+name+hobby;
		System.out.println("add:"+add);
		return "result";
	}

	@GetMapping("/test5")
	public String test5(@RequestParam int id,
			@RequestParam int name,
			@RequestParam int []hobby) {
		System.out.println("id:"+id);
		System.out.println("name:"+name);
		for(int str:hobby) {
			System.out.println("hobby:"+str);
		}
		return "result";
	}

	@GetMapping("/test6")
	public String test6(@RequestParam(value="id") int value1,
			@RequestParam(value="name") int value2,
			@RequestParam(value="hobby") int [] value3) {
		System.out.println("id:"+value1);
		System.out.println("name:"+value2);
		for(int str:value3) {
			System.out.println("hobby:"+str);
		}
```

```
                return "result";
        }

        @GetMapping("/test7")
        public String test7(@RequestParam int id,
                            @RequestParam(required=true) String name,
                            @RequestParam(defaultValue="0") int hobby) {
                System.out.println("id:"+id);
                System.out.println("name:"+name);
                System.out.println("hobby:"+hobby);
                return "result";
        }

        @GetMapping("/test8")
        public String test1(@RequestParam Map<String, String> map) {
                String id=map.get("id");
                String name=map.get("name");

                System.out.printf("id :%s\n", id);
                System.out.printf("name :%s\n", name);
                return "result";
        }
}
```

(2) view폴더-jsp파일 만들기

index.jsp

```
 6 <meta charset="UTF-8">
 7 <title>Insert title here</title>
 8 </head>
 9 <body>
10     <a href="test1?id=100&name=lee&hobby=reading&hobby=music">test1-get</a><br/>
11     <hr/>
12
13     <form action="test2" method="post">
14         id : <input type="text" name="id"/><br/>
15         name : <input type="text" name="name"/><br/>
16         hobby : <input type="checkbox" name="hobby" value="study"/>공부<br/>
17                 <input type="checkbox" name="hobby" value="reading"/>독서<br/>
18                 <input type="checkbox" name="hobby" value="music"/>음악<br/>
19         <button type="submit">test2-post</button>
20     </form>
21     <hr/>
22
23     <a href="test3?id=100&name=hong&hobby=study&hobby=music">test3</a>
24     <hr/>
25
26     <a href="test4/100/200/300">test4</a>
27     <hr/>
28
29     <a href="test5?id=100&name=200&hobby=300&hobby=400">test5</a>
30     <hr/>
31
32     <a href="test6?id=100&name=200&hobby=300&hobby=400">test6</a>
33     <hr/>
34
35     <a href="test7?id=100&name=lee&hobby=300">test7</a>
36     <hr/>
37
38     <a href="test8?id=100&name=lee">test8</a>
39     <hr/>
40
41 </body>
```

5. Controller의 리턴값

[실습5]

(1) 컨트롤러 TestController 에 매핑

```
package kr.co.korea.controller;


@Controller
public class TestController {

	@GetMapping("/test1")
	public String test1() {  //test1.jsp로 이동
		return "test1";
	}

	@GetMapping("/test2")
	public String test2() {   //test2.jsp로 이동
		return "test2";
	}

	@GetMapping("/test3")
	public String test3(HttpServletRequest request) {  //(1) request객체 주입받아서 데이터 담기
		request.setAttribute("data1", 100);
		request.setAttribute("data2", 200);
		return "test3";
	}

	@GetMapping("/test4")
	public String test4(Model model) {          //(2) model객체에 주입받아서 데이터 담기
		model.addAttribute("data1", 100);
		model.addAttribute("data2", 200);
		return "test4";
	}

	@GetMapping("/test5")
	public ModelAndView test5(ModelAndView mv) {  //(3) ModelAndView객체에 데이터 담기
		mv.addObject("data1",100);
		mv.addObject("data2",200);
		mv.setViewName("test5");
		return mv;
	}

}
```

(2) view 작성

```
//index.jsp
<body>
	<a href="test1">test1 --이동</a><br/>
	<a href="test2?data1=100&data2=200">test2--파라메타전달</a><br/>
	<hr />
	<a href="test3">test3-- request객체활용</a><br/>

	<hr />
	<a href="test4">test4-- model객체활용 </a><br/>
	<hr />
	<a href="test5">test5-- ModelAndView객체활용 </a><br/>
</body>


//--------------------------------
//test1.jsp
<body>
<h1>test1</h1>
</body>
//--------------------------------
```

```
//test2.jsp
<body>
<h1>test2</h1>
<h3>data1 : ${param.data1 }</h3>
<h3>data2 : ${param.data2 }</h3>
</body>
//test3.jsp
<body>
        <h1>test3</h1>
        <h3>data1:${requestScope.data1 }</h3>
        <h3>data2:${data2 }</h3>
</body>
//--------------------------------
//test4.jsp
<body>
<h1>test4</h1>
<h3>data1: ${requestScope.data1 }</h3>
<h3>data2: ${data2 }</h3>
</body>
//--------------------------------
//test5.jsp
<body>
<h1>test5</h1>
<h3>data1: ${data1}</h3>
<h3>data2: ${data2} </h3>
</body>
```

requestScope은 생략가능

문제:index.jsp에서 <form>태크를 통하여 이름,국어,영어,수학을 입력받아 test6.jsp에 이름과 총점을 출력되도록 TestController에서 작성하시오

6. DataBean객체 주입

[실습6]

(1) 주입받을 객체 생성

[kr.co.korea.beans] 패키지에 DataBean, dataBean1 2개 생성

```
1  package kr.co.korea.beans;
2
3  public class DataBean {
4      private String data1;
5      private String data2;
6      public String getData1() {
7          return data1;
8      }
9      public void setData1(String data1)
10         this.data1 = data1;
11     }
12     public String getData2() {
13         return data2;
14     }
15     public void setData2(String data2)
16         this.data2 = data2;
17     }
18
19 }
```

```
1  package kr.co.korea.beans;
2
3  public class dataBean1 {
4      private String id;
5      private String name;
6      public String getId() {
7          return id;
8      }
9      public void setId(String id) {
10         this.id = id;
11     }
12     public String getName() {
13         return name;
14     }
15     public void setName(String name) {
16         this.name = name;
17     }
18
19 }
```

(2) 컨트롤러 TestController 에 매핑

```java
@Controller
public class TestController {
        @PostMapping("/test1")
         public String test1(@ModelAttribute DataBean bean) {  //@ModelAttribute 생략가능
                  return "test1";
         }

        @PostMapping("/test2")
        public String test2(@ModelAttribute("testData") DataBean bean) {
                 return "test2";
        }

        @GetMapping("/test3")
        public String test3(dataBean1 bean2) {
                 System.out.println(bean2.getId());
                 System.out.println(bean2.getName());
                 return "test3";
        }
}
```

(3) view 작성

```jsp
//--------------------------------
//index.jsp
<body>
<form action="test1" method="post">
   data1:<input type="text" name="data1" /><br/>
   data2:<input type="text" name="data2" />
   <button type="submit">확인</button>
</form>
<hr/>

<form action="test2" method="post">
   data1:<input type="text" name="data1" /><br/>
   data2:<input type="text" name="data2" />
   <button type="submit">확인</button>
</form>
<hr/>

<a href="test3?id=hong&name=gildong">test3</a><br/>
</body>

//--------------------------------
//test1.jsp
<body>
<h1>test1</h1>
data1: ${dataBean.data1}
data2: ${dataBean.data2}
</body>

//--------------------------------
//test2.jsp
<body>
<h1>test2</h1>
data1: ${testData.data1}
data2: ${testData.data2}
</body>

//--------------------------------
//test3.jsp
<body>
<h1>test3</h1>
id:${dataBean1.id }<br/>
name:${dataBean1.name }<br/>
</body>
```

주입된 클래스는 DataBean이지만 실제로 리턴되는 객체는 클래스명의 **첫글자가 소문자로 변경**되어 dataBean이란 이름으로 사용된다.

주입된 클래스가 dataBean1처럼 소문자로 시작할 경우, 리턴되는 객체는 **첫글자가 소문자로 변경**된 dataBean1로 사용되므로 동일한 클래스명으로 사용된다.

7. Form 태그

[실습7]

(1) 주입받을 객체 생성

[kr.co.korea.beans] 패키지에 UserDataBean 생성

```java
package kr.co.korea.beans;

public class UserDataBean {
	private String user_name;
	private String user_id;
	private String user_pw;
	private String user_postcode;
	private String user_address;

	public String getUser_name() {
		return user_name;
	}
	public void setUser_name(String user_name) {
		this.user_name = user_name;
	}
	public String getUser_id() {
		return user_id;
	}
```

(2) 컨트롤러 TestController 에 매핑

```java
@Controller
public class TestController {

	@GetMapping("/test1")
	public String test1(UserDataBean bean) {
		bean.setUser_name("테스트1");
		bean.setUser_id("abcd");
		bean.setUser_pw("1234");
		bean.setUser_postcode("12345");
		bean.setUser_address("대구");
		return "test1";
	}

	@GetMapping("/test2")
	public String test2(UserDataBean bean) {
		bean.setUser_name("테스트2");
		bean.setUser_id("abcd");
		bean.setUser_pw("1234");
		bean.setUser_postcode("12345");
		bean.setUser_address("대구");
		return "test2";
	}

	@GetMapping("/test3")
	public String test3(@ModelAttribute("testBean")  UserDataBean bean) {
		bean.setUser_name("테스트3");
		bean.setUser_id("abcd");
		bean.setUser_pw("1234");
		bean.setUser_postcode("12345");
		bean.setUser_address("대구");
		return "test3";
	}
	@GetMapping("/test4")
	public String test4(Model model) {
		UserDataBean bean=new UserDataBean();
		bean.setUser_name("테스트4");
```

```
		bean.setUser_id("abcd");
		bean.setUser_pw("1234");
		bean.setUser_postcode("12345");
		bean.setUser_address("대구");

		model.addAttribute("bean4", bean);
		return "test4";
	}

	@GetMapping("/test5")
	public ModelAndView test5(ModelAndView mv) {
		UserDataBean bean=new UserDataBean();
		bean.setUser_name("테스트5");
		bean.setUser_id("abcd");
		bean.setUser_pw("1234");
		bean.setUser_postcode("12345");
		bean.setUser_address("대구");

		mv.addObject("bean5", bean);
		mv.setViewName("test5");
		return mv;
	}

	//test6 작성하시오
}
```

(3) view 작성

```
//--------------------------------
//index.jsp
<body>
<h1>index</h1>
	<a href="test1">test1</a>
	<hr/>

    <a href="test2">test2</a>
	<hr/>

	<a href="test3">test3</a>
	<hr/>

	<a href="test4">test4</a>
	<hr/>

	<a href="test5">test5</a>
	<hr/>
	test6 <br/>
	<form action="test6" method="post">
	   이름:<input type="text" name="user_name" /><br/>
	   아이디:<input type="text" name="user_id" /><br/>
	   비번:<input type="password" name="user_pw" /><br/>
	   우편번호:<input type="text" name="user_postcode" /><br/>
	   주소:<input type="text" name="user_address" /><br/>
	   <input type="submit" value="테스트6" />
	</form>
</body>
//--------------------------------
//test1.jsp
<body>
	<h1>test1</h1>
	<form action="result" method="post">
	   이름:<input type="text" name="user_name" value="${userDataBean.user_name }"/><br/>
	   아이디:<input type="text" name="user_id" value="${userDataBean.user_id }"/><br/>
	   비번:<input type="password" name="user_pw" value="${userDataBean.user_pw }"/><br/>
	   우편번호:<input type="text" name="user_postcode" value="${userDataBean.user_postcode }"/><br/>
	   주소:<input type="text" name="user_address" value="${userDataBean.user_address }"/><br/>
	</form>
</body>
```

```
//--------------------------------
//test2.jsp
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<body>
<h1>test2</h1>
<form:form modelAttribute="userDataBean" action="result" >
이름:<form:input path="user_name"/><br/>
아이디:<form:input path="user_id"/><br/>
비밀번호:<form:password path="user_pw" showPassword="true"/><br/>
우편번호:<form:input path="user_postcode"/><br/>
주소:<form:input path="user_address"/><br/>
</form:form>
</body>
//--------------------------------
//test3.jsp
<body>
<h1>test3</h1>
<form:form modelAttribute="testBean" action="result" >
이름:<form:input path="user_name"/><br/>
아이디:<form:input path="user_id"/><br/>
비밀번호:<form:password path="user_pw" showPassword="true"/><br/>
우편번호:<form:input path="user_postcode"/><br/>
주소:<form:input path="user_address"/><br/>
</form:form>
</body>
//--------------------------------
//test4.jsp
<body>
<h1>test4</h1>
<form:form modelAttribute="bean4" action="result" >
이름:<form:input path="user_name"/><br/>
아이디:<form:input path="user_id"/><br/>
비밀번호:<form:password path="user_pw" showPassword="true"/><br/>
우편번호:<form:input path="user_postcode"/><br/>
주소:<form:input path="user_address"/><br/>
</form:form>
</body>
//--------------------------------
//test5.jsp
<body>
<h1>test5</h1>
<form:form modelAttribute="bean5" action="result" >
이름:<form:input path="user_name"/><br/>
아이디:<form:input path="user_id"/><br/>
비밀번호:<form:password path="user_pw" showPassword="true"/><br/>
우편번호:<form:input path="user_postcode"/><br/>
주소:<form:input path="user_address"/><br/>
</form:form>
</body>
//--------------------------------
//test6.jsp
//작성하시오
```

8. Form태그의 요소(element)

[실습8]

(1) 주입받을 객체 생성

[kr.co.korea.beans] 패키지에 DataBean 클래스와 KeyValueBean 클래스 생성

```
public class DataBean {
        private String uid;
        private String name;
        private String pwd;
        private String intro;
        private String job;
        private String job1;
        private String job2;
        private String job3;
        private String hobby1[];
        private String hobby2[];
        private String hobby3[];
```

```
public class KeyValueBean {
        private String key;
        private String value;
        public String getKey() {
                return key;
        }

//이하생략
```

```
        private String hobby4[];
        private String gender1;
        private String gender2;
        private String gender3;
        private String gender4;

        public String getUid() {
                return uid;
        }

//이하생략
```

(2) 컨트롤러 TestController 에 매핑

```
@Controller
public class TestController {

        @GetMapping("/test1")
        public String test1(DataBean bean, Model model) {
                bean.setUid("hong");
                bean.setName("홍길동");
                bean.setPwd("hong");
                bean.setIntro("자신을 소개해주세요");
                bean.setJob("it");
                bean.setJob1("학생");
                bean.setJob2("시스템관리직");
                bean.setJob3("dba");
                //setHobby1 ~setHobby3 아래쪽에서
                bean.setGender1("male");
                bean.setGender2("male");
                bean.setGender3("male");

                //job_list1
                String [] job_list1= {"기술자","교사","학생"};
                model.addAttribute("job_list1", job_list1);


                //job_list2
                ArrayList<String> job_list2=new ArrayList<String>();
                job_list2.add("청소부");
                job_list2.add("서비스직");
                job_list2.add("시스템관리직");
                job_list2.add("DB관리직");
                model.addAttribute("job_list2", job_list2);


                //job_list3
                KeyValueBean key_bean1=new KeyValueBean();
                KeyValueBean key_bean2=new KeyValueBean();
                KeyValueBean key_bean3=new KeyValueBean();
                key_bean1.setKey("프로그래머");
                key_bean1.setValue("programmer");
                key_bean2.setKey("관리자");
                key_bean2.setValue("manager");
                key_bean3.setKey("데이터베이스관리자");
                key_bean3.setValue("dba");

                ArrayList<KeyValueBean> job_list3=new ArrayList<KeyValueBean>();
                job_list3.add(key_bean1);
                job_list3.add(key_bean2);
                job_list3.add(key_bean3);
                model.addAttribute("job_list3", job_list3);


                //hobby_list1
                String [] hobby_list1= {"오락","골프","음식만들기"};
                model.addAttribute("hobby_list1", hobby_list1);


                //hobby_list2
                ArrayList<String> hobby_list2=new ArrayList<String>();
                hobby_list2.add("만화책보기");
                hobby_list2.add("TV시청");
```

```
                hobby_list2.add("골프");
                hobby_list2.add("음식만들기");
                model.addAttribute("hobby_list2", hobby_list2);


                //hobby_list3
                key_bean1.setKey("기술책보기");
                key_bean1.setValue("itbook");
                key_bean2.setKey("골프");
                key_bean2.setValue("golf");
                key_bean3.setKey("음식만들기");
                key_bean3.setValue("food");

                ArrayList<KeyValueBean> hobby_list3=new ArrayList<KeyValueBean>();
                hobby_list3.add(key_bean1);
                hobby_list3.add(key_bean2);
                hobby_list3.add(key_bean3);
                model.addAttribute("hobby_list3", hobby_list3);


                String hobby_list[]= {"game", "골프","음식만들기"};
                bean.setHobby1(hobby_list);
                bean.setHobby2(hobby_list);
                bean.setHobby3(hobby_list);

                //test1.jsp의 gender1~gender3까지
                //radiobutton태그에 남자와 여자로 실습할것
                return "test1";
        }
}
```

(3) view 작성

```
//-----------------------------------------
//index.jsp
<body>
<a href="test1">test1</a>
</body>

//-----------------------------------------
//test1.jsp
<body>
<h1>test1</h1>
<form:form modelAttribute="dataBean" action="result">
        <form:hidden path="uid"/>
        name: <form:input path="name"/><br/>
        password: <form:password path="pwd" showPassword="true"/><br/>
        intro: <form:textarea path="intro"/><br/>
        job: <form:select path="job">
                <form:option value="student">학생</form:option>
                <form:option value="it">기술자</form:option>
                <form:option value="teacher">교사</form:option>
        </form:select> <br/>
        <hr/>

        job1: <form:select path="job1">
                <form:options items="${job_list1 }" />
        </form:select><br/>
        <hr/>

        job2: <form:select path="job2">
                <form:options items="${job_list2 }" />
        </form:select>
        <hr/>

        job3: <form:select path="job3">
                <form:options items="${job_list3 }" itemLabel="key" itemValue="value"/>
        </form:select> <br/>

        <hr/>
        hobby: <form:checkbox path="hobby1" value="game"/>게임
        <form:checkbox path="hobby1" value="study"/>스터디
        <form:checkbox path="hobby1" value="sleep"/>낮잠 <br/>
```

```
		<hr/>
		hobby1: <form:checkboxes items="${job_list1 }" path="job1"/><br/>

		<hr/>
		hobby2: <form:checkboxes items="${job_list2 }" path="job2"/><br/>

		<hr/>
		hobby3: <form:checkboxes items="${job_list3 }" path="job3" itemLabel="key" itemValue="value"/><br/>

		<hr/>
		gender: <form:radiobutton path="gender1" value="male"/>남자
		<form:radiobutton path="gender1" value="female"/>여자<br/>

		<hr/>
		gender1: <form:radiobuttons path="job1" items="${job_list1 }"/><br/>

		<hr/>
		gender2: <form:radiobuttons path="job2" items="${job_list2 }"/><br/>

		<hr/>
		gender3: <form:radiobuttons path="job3" items="${job_list3 }" itemLabel="key"
itemValue="value"/><br/>

		<form:button disabled="true">확인</form:button>
</form:form>
</body>
```

9. redirect 와 forward

- redirect : 서버가 클라이언트에게 요청주소를 응답결과로 전달하는 것. 클라이언트는 응답결과로 받은 요청주소를 직접 요청하게 되고, 브라우저가 요청하는 것이므로 주소창의 주소는 변경되고 기존의 HttpServletRequest 객체는 소멸후 새롭게 생성됨. 즉 데이터를 받아가지 못함
- forward : 서버상에서만 이동하므로 브라우저는 다른곳으로 흐름이 이동되었다는 것을 알수 없기 때문에 주소창의 주소는 변경되지 않음. 기존의 HttpServletRequest객체는 그대로 유지됨. 즉 데이터를 받아갈 수 있음

[실습9]

(1) 컨트롤러 TestController

```
@Controller
public class TestController1 {
		@GetMapping("/test1")
		public String test1(HttpServletRequest request) {
			request.setAttribute("name1", "홍길동");
			return "redirect:/sub1";
		}

		@GetMapping("/sub1")
		public String sub1(HttpServletRequest request) {
			System.out.println(request.getAttribute("name1"));
			return "sub1";
		}

		@GetMapping("/test2")
		public String test2(HttpServletRequest request) {
			request.setAttribute("name2", "이순자");
			return "forward:/sub2";
		}

		@GetMapping("/sub2")
		public String sub2(HttpServletRequest request) {
			System.out.println(request.getAttribute("name2"));
			return "sub2";
		}
}
```

redirect:/sub1 로 이동시 값을 받지못함

forward:/sub2 로 이동시 값을 받아옴

(2) view 작성

```
//index.jsp
<body>
<a href="test1">test1-redirect</a><br/>
<a href="test2">test2-forward</a>
</body>
//sub1.jsp
<body>
<h1>sub1</h1>
이름:${name1 }
</body>
//sub2.jsp
<body>
<h1>sub2</h1>
이름:${name2 }
</body>
```

10. Mybatis

pom.xml 파일에 { spring-jdbc, commons-dbcp2, mybatis-spring, mybatis } 다운로드할 파일 등록

ojdbc6.jar파일을 lib에 저장할것

[실습10]-1 Java방식 설정

(1) 데이터베이스 구성 (memtable.sql)

```
drop table memtable;
create table memtable (
   id varchar2(100),
   name varchar2(100),
   pw  varchar2(100)
);
```

(2) 주입받을 객체 생성

[kr.co.korea.beans] 패키지에 DataBean 클래스 생성

```
1  package kr.co.korea.beans;
2
3  public class DataBean {
4
5      private String id;
6      private String name;
7      private String pw;
8      public String getId() {
9          return id;
10     }
11     public void setId(String id) {
12         this.id = id;
13     }
14     public String getName() {
15         return name;
16     }
```

(3) properties 파일생성

[WEB-INF]-[properties]폴더- db.properties 파일

```
db.classname = oracle.jdbc.OracleDriver
db.url = jdbc:oracle:thin:@127.0.0.1:1521:xe
db.username = system
db.password = 123456
```

(4) db.properties 파일을 ServletAppContext 클래스에 등록

```
@Configuration
@EnableWebMvc
@ComponentScan("kr.co.korea.controller")
@PropertySource("/WEB-INF/properties/db.properties")
public class ServletAppContext implements WebMvcConfigurer{

        @Value("${db.classname}")
        private String db_classname;

        @Value("${db.url}")
        private String db_url;

        @Value("${db.username}")
        private String db_username;

        @Value("${db.password}")
        private String db_password;
```

```
        // Controller의 메서드가 반환하는 jsp의 이름 앞뒤에 경로와 확장자를 붙혀주도록 설정한다.
        @Override
        public void configureViewResolvers(ViewResolverRegistry registry) {
        // TODO Auto-generated method stub
                WebMvcConfigurer.super.configureViewResolvers(registry);
                registry.jsp("/WEB-INF/views/", ".jsp");
        }

        // 정적 파일의 경로를 매핑한다.
        @Override
        public void addResourceHandlers(ResourceHandlerRegistry registry) {
                // TODO Auto-generated method stub
                WebMvcConfigurer.super.addResourceHandlers(registry);
                registry.addResourceHandler("/**").addResourceLocations("/resources/");
        }

        // 데이터베이스 접속 정보 관리(org.apache.commons.dbcp2.BasicDataSource)
        @Bean
        public BasicDataSource dataSource() {
                BasicDataSource source = new BasicDataSource();
                source.setDriverClassName(db_classname);
                source.setUrl(db_url);
                source.setUsername(db_username);
                source.setPassword(db_password);
                //System.out.println("db connect");
                return source;
        }

}
```

(5) Mapper 인터페이스 생성

[kr.co.korea.database]패키지- MapperSql 인터페이스

```
public interface MapperSql {

  @Insert("insert into memtable (id, name, pw) values (#{id}, #{name}, #{pw})")
  void insert_data(DataBean dataBean);          // 컨트롤러에서 insert_data(dataBean) 이름으로 호출되는 메소드

  @Select("select id, name, pw from memtable") // 컨트롤러에서 select_data() 이름으로 호출되는 메소드
  List<DataBean> select_data();

 }
```

(6) MapperSql 인터페이스를 ServletAppContext 클래스에 등록

```
@Configuration
@EnableWebMvc
@ComponentScan("kr.co.korea.controller")
@PropertySource("/WEB-INF/properties/db.properties")
public class ServletAppContext implements WebMvcConfigurer{

        @Value("${db.classname}")
        private String db_classname;

        @Value("${db.url}")
        private String db_url;

        @Value("${db.username}")
        private String db_username;

        @Value("${db.password}")
        private String db_password;


        // Controller의 메서드가 반환하는 jsp의 이름 앞뒤에 경로와 확장자를 붙혀주도록 설정한다.
        @Override
        public void configureViewResolvers(ViewResolverRegistry registry) {
                // TODO Auto-generated method stub
                WebMvcConfigurer.super.configureViewResolvers(registry);
                registry.jsp("/WEB-INF/views/", ".jsp");
```

```
        }

        // 정적 파일의 경로를 매핑한다.
        @Override
        public void addResourceHandlers(ResourceHandlerRegistry registry) {
                // TODO Auto-generated method stub
                WebMvcConfigurer.super.addResourceHandlers(registry);
                registry.addResourceHandler("/**").addResourceLocations("/resources/");
        }

        // 데이터베이스 접속 정보 관리
        @Bean
        public BasicDataSource dataSource() {
                BasicDataSource source = new BasicDataSource();
                source.setDriverClassName(db_classname);
                source.setUrl(db_url);
                source.setUsername(db_username);
                source.setPassword(db_password);

                //System.out.println("db connect");
                return source;
        }

        // 쿼리문과 접속 관리하는 객체
        @Bean
        public SqlSessionFactory factory(BasicDataSource source) throws Exception{
                SqlSessionFactoryBean factoryBean = new SqlSessionFactoryBean();
                factoryBean.setDataSource(source);
                SqlSessionFactory factory = factoryBean.getObject();
                //System.out.println("sql");
                return factory;
        }

        // 쿼리문 실행을 위한 객체
        @Bean
        public MapperFactoryBean<MapperSql> test_mapper(SqlSessionFactory factory) throws Exception{
                MapperFactoryBean<MapperSql> factoryBean = new
MapperFactoryBean<MapperSql>(MapperSql.class);
                factoryBean.setSqlSessionFactory(factory);
                return factoryBean;
        }

}
```

(7) 컨트롤러 TestController 에서 MapperSql 인터페이스 로딩하여 호출하기

```
@Controller
public class TestController {

        @Autowired
        MapperSql mapper1;


        @GetMapping("/write")
        public String write() {
                return "write";
        }

        @GetMapping("/result")
        public String result(DataBean dataBean) {

                mapper1.insert_data(dataBean);
                return "result";
        }

        @GetMapping("/show")
        public String show(Model model) {
                List<DataBean> list = mapper1.select_data();
                model.addAttribute("list", list);

                return "show";
        }
}
```

(8) view 작성

```
//----------------------------
//index.jsp
<body>
        <a href="write">가입</a><br/>
        <a href="show">목록보기</a><br/>
</body>
//----------------------------
//write.jsp
<body>
<h1>write/h1>
<form action="result" method="get">
아이디 : <input type="text" name="id" /><br/>
이름 : <input type="text" name="name" /><br/>
비번 : <input type="password" name="pw" /><br/>
<button type="submit">확인</button>
</form>
</body>
//----------------------------
//show.jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<body>
        <c:forEach var="obj" items="${list}">
                ${obj.id }, ${obj.name }, ${obj.pw }<br/>
        </c:forEach>
</body>
//----------------------------
//result.jsp
<body>
<h1>저장완료</h1>
</body>
```

과제1: 아래 index.jsp 파일을 작성하여 MapperSql.java 수정하고, TestController.java 작성하시오.
edit.jsp 의 결과는 result2.jsp, del.jsp의 결과는 result3.jsp

```
//index.jsp
<body>
        <a href="write">가입</a><br/>
        <a href="show">목록보기</a><br/>
        <a href="edit">수정</a><br/>
        <a href="del">삭제</a><br/>
</body>
```

[실습10]-2 XML방식 설정

(1) 주입받을 객체 생성

[kr.co.korea.beans] 패키지에 DataBean 클래스 생성

```
1  package kr.co.korea.beans;
2
3  public class DataBean {
4
5      private String id;
6      private String name;
7      private String pw;
8      public String getId() {
9          return id;
10     }
11     public void setId(String id) {
12         this.id = id;
13     }
14     public String getName() {
15         return name;
16     }
```

(2) properties 파일생성

[WEB-INF]-[properties]폴더- db.properties 파일

```
db.classname = oracle.jdbc.OracleDriver
db.url = jdbc:oracle:thin:@127.0.0.1:1521:xe
db.username = system
db.password = 123456
```

(3) db.properties 파일을 servlet-context.xml 에 등록

```
<?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

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">


        <annotation-driven />
        <context:component-scan base-package="kr.co.korea.controller" />


        <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>

        <resources location="/resources/" mapping="/**"></resources>


<!-- properties파일의 내용을 사용할 수 있도록 Bean를 정의 -->
        <beans:bean class='org.springframework.beans.factory.config.PropertyPlaceholderConfigurer'>
                <beans:property name="location">
                        <beans:value>/WEB-INF/properties/db.properties</beans:value>
                </beans:property>
        </beans:bean>

        <beans:bean class='org.apache.commons.dbcp2.BasicDataSource' id='basic_data_source'>
                <beans:property name='driverClassName' value='${db.classname}'/>
                <beans:property name="url" value="${db.url}"/>
                <beans:property name="username" value='${db.username}'/>
                <beans:property name="password" value='${db.password}'/>
        </beans:bean>

</beans:beans>
```

(4) Mapper XML 생성

[WEB-INF]-[mapper]폴더- **mapperSql.xml**

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
                        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="test_db">
        <insert id="insert_data" parameterType="kr.co.korea.beans.DataBean">
                <![CDATA[
                        insert into memtable (id, name, pw)
                        values (#{id}, #{name}, #{pw})
                ]]>
        </insert>
        <select id='select_data' resultType="kr.co.korea.beans.DataBean">
                <![CDATA[
                        select id, name, pw from memtable
                ]]>
        </select>
```

```
</mapper>
```

(5) Mapper XML을 servlet-context.xml 에 등록

```
<?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

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">


        <annotation-driven />
        <context:component-scan base-package="kr.co.korea.controller" />


        <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>

        <resources location="/resources/" mapping="/**"></resources>

        <!-- properties파일의 내용을 사용할 수 있도록 Bean를 정의 -->

        <beans:bean class='org.springframework.beans.factory.config.PropertyPlaceholderConfigurer'>
                <beans:property name="location">
                        <beans:value>/WEB-INF/properties/db.properties</beans:value>
                </beans:property>
        </beans:bean>

        <beans:bean class='org.apache.commons.dbcp2.BasicDataSource' id='basic_data_source'>
                <beans:property name='driverClassName' value='${db.classname}'/>
                <beans:property name="url" value="${db.url}"/>
                <beans:property name="username" value='${db.username}'/>
                <beans:property name="password" value='${db.password}'/>
        </beans:bean>

        <beans:bean class='org.mybatis.spring.SqlSessionFactoryBean' id='sqlSession'>
                <beans:property name="dataSource" ref='basic_data_source'/>
                <beans:property name="mapperLocations" value='/WEB-INF/mapper/*.xml'/>
        </beans:bean>

        <beans:bean class='org.mybatis.spring.SqlSessionTemplate' >
                <beans:constructor-arg index='0' ref='sqlSession'/>
        </beans:bean>

</beans:beans>
```

(6) 컨트롤러 TestController 에서 SqlSessionTemplate 로딩하여 호출하기

```
@Controller
public class TestController {

        @Autowired
        SqlSessionTemplate sqltemp;

        @GetMapping("/write")
        public String write() {
                return "write";
        }

        @GetMapping("/result")
        public String result(DataBean dataBean) {
```

XML방식에서
SqlSessionTemplate클래스에서
지원되는 메소드

1) int insert("맵id", [parameter])
2) List<Object> selectList("맵id")
3) int selectOne("맵id")
4) int delete("맵id", [parameter])
5) int update(("맵id", [parameter])

```
        sqltemp.insert("test_db.insert_data", dataBean);
        return "result";
    }

    @GetMapping("/show")
    public String show(Model model) {
        List<DataBean> list = sqltemp.selectList("test_db.select_data");
        model.addAttribute("list", list);

        return "show";
    }
```

(7) view (Java방식 설정의 view와 동일하므로 복사하여 사용)

```
//----------------------------
//index.jsp
<body>
        <a href="write">가입</a><br/>
        <a href="show">목록보기</a><br/>
</body>
//----------------------------
//write.jsp
<body>
<h1>write/h1>
<form action="result" method="get">
아이디 : <input type="text" name="id" /><br/>
이름 : <input type="text" name="name" /><br/>
비번 : <input type="password" name="pw" /><br/>
<button type="submit">확인</button>
</form>
</body>
//----------------------------
//show.jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<body>
        <c:forEach var="obj" items="${list}">
                ${obj.id }, ${obj.name }, ${obj.pw }<br/>
        </c:forEach>
</body>
//----------------------------
//result.jsp
<body>
<h1>저장완료</h1>
</body>
```

과제2: 아래 index.jsp 파일을 작성하여 mapperSql.xml 수정하고, TestController.java 작성하시오.
edit.jsp 의 결과는 result2.jsp, del.jsp의 결과는 result3.jsp

```
//index.jsp
<body>
        <a href="write">가입</a><br/>
        <a href="show">목록보기</a><br/>
        <a href="edit">수정</a><br/>
        <a href="del">삭제</a><br/>
</body>
```

Scope

(1) Request : 클라이언트가 서버에 요청에 관련된 정보를 전송하면, 서버는 요청정보를 보관하기 위해 HttpServletRequest객체를 생성해서 요청 정보를 담아두고, HttpServletRequest객체는 응답결과가 클라이언트로 전송될 때까지 유지되며 사용이 가능. 이러한 사용 범위를 RequestScope라고 함

@RequestScope : 요청이 발생할 때마다 bean 객체가 생성되어 자동으로 주입. 주입된 bean은 요청발생시 주입만 이루어지는 것이므로, request영역에 저장되지 않음

1) 자바방식 : @RequestScope
2) xml방식 : scope="request"

(2) Session : 브라우저가 최초로 서버에 요청을 하게 되면 브라우저당 하나씩 메모리 공간을 서버에서 할당하고, 브라우저를 종료할 때까지 서버에서 사용. 메모리 영역은 브라우저당 하나씩 지정되며, 요청이 새롭게 발생하더라도 같은 메모리 공간을 사용. 이러한 사용 범위를 SessionScope라고 함

@SessionAttributes : @ModelAttribute를 통해 주입받은 Bean을 @SessionAttributes 로 지정해 놓으면 session영역에 저장되고 session영역에서 주입받을 수 있음

@SessionScope : 브라우저가 서버에 최초의 요청을 보낼때 Bean객체가 주입. 주입된 Bean은 주입만 이루어지는 것이므로 session영역에 저장되지 않음

1) 자바방식 : @SessionScope
2) xml방식 : scope="session"

(3) Application : 서버가 가동될 때부터 서버가 종료되는 시점까지의 범위를 Application Scope이라 함

ServletContext라는 클래스 타입의 객체로 관리되고, ServletContext에 저장된 데이터나 객체는 서버가 종료되기 전까지 동일한 메모리 공간을 사용.

HttpServletRequest 객체로 부터 추출이 가능. Controller에서 주입받을 수 있음

1) 자바방식 : @ApplicationScope
2) xml방식 : scope="application"

11. Request

[실습11] Java방식과 Xml방식

(1) DataBean1, DataBean2, DataBean3, DataBean4

```java
package kr.co.korea.beans;

public class DataBean1 {
  private String id;
  private String pw;

        }
        ;
}
```

```java
package kr.co.korea.beans;

public class DataBean2 {
  private String name;
  private String juso;

;
```

```java
public class DataBean3 {

  private String kor;
  private String eng;
  private String mat;
;
}
```

```java
public class DataBean4 {

  private String no;
  private String jumsu;
```

(2) TestController.java

```java
@Controller
public class TestController {

// (1) HttpRequest
        @GetMapping("/test1")
        public String test1(HttpServletRequest request)
        {
                request.setAttribute("name1", "공공");
                return "test1";
```

```
	}

	@GetMapping("/result")
	public String result(HttpServletRequest request)
	{
		String data1=(String) request.getAttribute("name1");
		//System.out.println(data1);
		return "result";
	}
//-----------------------------------------------------
// (2) 리다이렉트
	@GetMapping("/test2")
	public String test2(HttpServletRequest request) {
		request.setAttribute("name2", "최고야2");
		return "redirect:/result";
	}
//(3) 포워드
	@GetMapping("/test3")
	public String test3(HttpServletRequest request)    //request로 전달받음
	{
		request.setAttribute("name3", "김삼묘3");
		return "forward:/result";
	}
//-----------------------------------------------------
//(4) 데이터빈 객체 생성
	@GetMapping("/test4")
	public String test4(HttpServletRequest request) {
		DataBean1 bean1 = new DataBean1();
		bean1.setId("400");
		bean1.setPw("444");

		request.setAttribute("bean4", bean1);

		return "result";
	}
//-----------------------------------------------------
//(5) DataBean1 객체 bean1로 설정후 --> 모델객체 bean5 로 리턴
	@GetMapping("/test5")
	public String test5(@ModelAttribute("bean5") DataBean1 bean1)    //request로 전달받음
	{
		bean1.setId("500");
		bean1.setPw("555");
		return "result";
	}
//-----------------------------------------------------
// (6) bean6을 DataBean1 bean1로 주입하여 사용하여 모델객체는 bean6
	@GetMapping("/test6")
	public String test6(Model model, DataBean1 bean1) {
		bean1.setId("600");
		bean1.setPw("666");
		model.addAttribute("bean6", bean1);

		return "result";
	}

//-----------------------------------------------------
// (7) bean7을 DataBean1 bean1로 주입하여 사용하여 모델객체는 bean6
	@GetMapping("/test7")
	public String test7(@ModelAttribute("bean7") DataBean1 bean1,
				@ModelAttribute("bean77") DataBean2 bean2) {
		bean1.setId("700");
		bean1.setPw("777");
		bean2.setName("칠칠이");
		bean2.setJuso("칠곡");
		return "result";
	}
//-----------------------------------------------------
// (8) RootAppContext.java에 @Bean객체로 @RequestScope으로 지정할 경우
	@Autowired                                                    // ☞ ※ XML방식은 (8) 참고
	DataBean1 bean8;

	@Resource(name="testBean2")
	DataBean2 bean88;

	@GetMapping("/test8")
```

```
	public String test8(Model model) {
		bean8.setId("chichi");
		bean8.setPw("12345");

		bean88.setName("choi");
		bean88.setJuso("대구");
		model.addAttribute("bean8", bean8);
		model.addAttribute("bean88", bean88);

		return "result";
	}
//-------------------------------------------------------
// (9) DataBean3, DataBean4 클래스 상단에 @Component로 @RequestScope로 지정할 경우
	@Autowired                                          // ☞ ※ XML방식은 (8) 참고
	DataBean3 bean9;

	@Resource(name="testBean4")
	DataBean4 bean99;

	@GetMapping("/test9")
	public String test9(Model model) {
		bean9.setEng("100");
		bean9.setKor("100");
		bean9.setMat("100");

		bean99.setNo("100");
		bean99.setJumsu("대구");
		model.addAttribute("bean9", bean9);
		model.addAttribute("bean99", bean99);
		return "result";
	}
}
```

(3) RootAppContext.java 파일 ☞ ※ XML방식은 (6) 참고

- DataBean1 클래스를 @Bean객체로 @RequestScope으로 지정할 경우
- DataBean2 클래스를 @Bean객체로 @RequestScope으로 지정할 경우 (이름 지정 방법)

```
@Bean
@RequestScope
public DataBean1 dataBean1() {
	return new DataBean1();
}

@Bean("testBean2")
@RequestScope
public DataBean2 dataBean2() {
	return new DataBean2();
}
```

(4) DataBean3, DataBean4 파일 ☞ ※ XML방식도 동일하게 @Component 객체 정의

- DataBean1 클래스를 @Component객체로 @RequestScope으로 지정할 경우
- DataBean2 클래스를 @Component객체로 @RequestScope으로 지정할 경우 (이름 지정 방법)

단, ServletAppContext.java 파일에 @ComponentScan("kr.co.korea.beans") 등록할 것

☞ ※ XML방식 (7) 참고

```
@Component
@RequestScope
public class DataBean3 {

  private String kor;
  private String eng;
  private String mat;
:
}
```

```
@Component(value="testBean4")
@RequestScope
public class DataBean4 {

  private String no;
  private String jumsu;
```

(5) views (index.jsp, test0.jsp. result0.jsp, test1.jsp, result.jsp)

```
//----------------------------index.jsp-----
<body>
<table border="1" cellpadding="0" cellspacing="0">
<tr bgcolor="#ffff00"><th>RequestScope</th><th>SessionScope</th><th>ApplicationScope</th></tr>
<tr><td></td>
    <td><a href="test0">test0</a>, <a href="result0">result0</a></td>
    <td><a href="test0">test0</a>, <a href="result0">result0</a></td>
</tr>
<tr><td><a href="test1">test1</a>, <a href="result">result</a></td>
    <td><a href="test1">test1</a>, <a href="result">result</a></td>
    <td><a href="test1">test1</a>, <a href="result">result</a></td>
</tr>
<tr><td><a href="test2">test2</a></td>
<td><a href="test2">test2</a></td>
<td></td>
</tr>
<tr><td><a href="test3">test3</a></td>
<td><a href="test3">test3</a></td>
<td></td>
</tr>
<tr><td><a href="test4">test4</a></td>
<td><a href="test4">test4</a></td>
<td><a href="test4">test4</a></td>
</tr>
<tr><td><a href="test5">test5</a></td>
<td><a href="test5">test5</a></td>
<td></td>
</tr>
<tr><td><a href="test6">test6</a></td>
<td><a href="test6">test6</a></td>
<td></td>
</tr>
<tr><td><a href="test7">test7</a></td>
<td><a href="test7">test7</a></td>
<td></td>
</tr>
<tr><td><a href="test8">test8</a></td>
<td><a href="test8">test8</a></td>
<td><a href="test8">test8</a></td>
</tr>
<tr><td><a href="test9">test9</a></td>
<td><a href="test9">test9</a></td>
<td><a href="test9">test9</a></td>
</tr>

</table>
</body>
//----------------------------test0.jps-----
<body>
<h1>test0</h1>
</body>
//----------------------------test1.jps-----
<body>
<h1>test1</h1>
</body>
//----------------------------result0.jsp-----
<body>
<h1>result0</h1>
${name0 }
</body>
//----------------------------result.jsp----
<body>
<h1>result</h1>
name1: ${name1} <br/>
name2: ${name2} <br/>
name3: ${name3} <br/>
<hr/>
test4 <br/>
${bean4.id } , ${bean4.pw } <br/>
<hr/>
```

```
test5<br/>
${bean5.id } , ${bean5.pw } <br/>
<hr/>
test6<br/>
${bean6.id } , ${bean6.pw } <br/>
<hr/>
test7<br/>
${bean7.id } , ${bean7.pw },${bean77.name },${bean77.juso }    <br/>
<hr/>
test8<br/>
${bean8.id } , ${bean8.pw },${bean88.name },${bean77.juso }    <br/>
<hr/>
test9<br/>
${bean9.eng } , ${bean9.kor }, ${bean9.mat }, ${bean99.no } , ${bean99.jumsu }    <br/>
<hr/>
</body>
```

※ (6) XML방식 root-context.xml 파일

- DataBean1 클래스를 bean객체로 scope=request으로 지정할 경우
- DataBean2 클래스를 bean객체로 scope=request으로 지정할 경우 (이름 지정 방법)

```
<?xml version="1.0" encoding="UTF-8"?>
생략;

    <bean class="kr.co.korea.beans.DataBean1" scope="request" />
    <bean class="kr.co.korea.beans.DataBean2" scope="request"  id="testbean2" />

</beans>
```

※ (7) XML방식 servlet-context.xml 파일

```
<context:component-scan base-package="kr.co.korea.beans" />
```

※ (8) XML방식 TestController.java 파일

```
@Autowired
@Lazy
DataBean1 bean2;

@Resource(name="testbean2")
@Lazy
DataBean2 bean22;


@Autowired
@Lazy
DataBean3 bean3;

@Resource(name="testBean4")
@Lazy
DataBean4 bean4;
```

12. SessionScope

[실습12] Java방식

(1) TestController3

```
@Controller
//@SessionAttributes("bean6")
@SessionAttributes({"bean6", "bean7", "bean77"})
public class TestController3 {


// (0) HttpRequest에서 세션정보 가져와서 HttpSeesion에 저장하고 설정
```

```
	@GetMapping("/test0")
	public String test0(HttpServletRequest request) {
		HttpSession session = request.getSession();
		session.setAttribute("name0", "처음사용자0");
		return "test0";				//test0으로 분기한 경우, result0에서 request로 받아오지 못함
	}


	@GetMapping("/result0")
	public String result0(HttpServletRequest request) {
		HttpSession session=request.getSession();
		System.out.println(session.getAttribute("name0"));
		return "result0";
	}

//(1) HttpSession에서 설정
	@GetMapping("/test1")
	public String test1(HttpSession session) {
		session.setAttribute("name1", "홍길동1");
		return "test1";
	}

	@GetMapping("/result")
	public String result(HttpSession session) {
		return "result";
	}
//-------------------------------------------------------
//(2) 리다이렉트
	@GetMapping("/test2")
	public String test2(HttpSession session) {
		session.setAttribute("name2", "최고야2");
		return "redirect:/result";
	}
//(3) 포워드
	@GetMapping("/test3")
	public String test3(HttpSession session) {
		session.setAttribute("name3", "김삼묘3");
		return "forward:/result";
	}
//-------------------------------------------------------
//(4) 데이터빈 객체 생성
	@GetMapping("test4")
	public String test4(HttpSession session) {
		DataBean1 bean1 = new DataBean1();
		bean1.setId("400");
		bean1.setPw("444");

		session.setAttribute("bean4", bean1);

		return "result";
	}
//-------------------------------------------------------
//(5) 이미 Session에 bean4객체로 저장된것을 bean1로 주입받아서 값변경하고 session객체로 bean4사용
	@GetMapping("/test5")
	public String test5(@SessionAttribute("bean4") DataBean1 bean1) {
		bean1.setId("500");
		bean1.setPw("555");
		return "result";
	}
//-------------------------------------------------------
// (6) 클래스위에 @SessionAttributes("bean6") 사용을 선언해야 함
// bean6을 모델객체로 사용할 수 있도록 정의해두어야 함
	@ModelAttribute("bean6")
	public DataBean1 dataBean1() {
		return new DataBean1();
	}

// (6) bean6을 DataBean1 bean1로 주입하여 사용하여 모델객체는 bean6리턴
	@GetMapping("/test6")
	public String test6(@ModelAttribute("bean6") DataBean1 bean1) {
		bean1.setId("600");
		bean1.setPw("666");

		return "result";
	}
//-------------------------------------------------------
// (7) 클래스위에@SessionAttributes({"bean6","bean7","bean77"})사용을 선언
// bean7, bean77 을 모델객체로 사용할 수 있도록 정의해두어야 함
	@ModelAttribute("bean7")
	public DataBean1 dataBean7() {
		return new DataBean1();
	}

	@ModelAttribute("bean77")
```

```
	public DataBean2 dataBean77() {
		return new DataBean2();
	}
// (7) bean7을 DataBean1 bean1로 주입하여 사용하여 모델객체는 bean7
//	bean77을 DataBean2 bean2로 주입하여 사용하여 모델객체는 bean77
	@GetMapping("/test7")
	public String test7(@ModelAttribute("bean7") DataBean1 bean1,
				@ModelAttribute("bean77") DataBean2 bean2) {

		bean1.setId("700");
		bean1.setPw("777");
		bean2.setName("칠칠이");
		bean2.setJuso("칠곡");
		return "result";
	}
//-------------------------------------------------------------
// (8) RootAppContext.java에 Bean객체로 SessionScope으로 지정할 경우
	@Autowired
	DataBean1 bean8;

	@Resource(name="testBean2")
	DataBean2 bean88;

	@GetMapping("/test8")
	public String test8(Model model) {
		bean8.setId("chichi");
		bean8.setPw("12345");

		bean88.setName("choi");
		bean88.setJuso("대구");
		model.addAttribute("bean8", bean8);
		model.addAttribute("bean88", bean88);

		return "result";
	}
//-------------------------------------------------------------
// (9) DataBean3, DataBean4 클래스 상단에 @Component로 @SessionScope로 지정할 경우

	@Autowired
	DataBean3 bean9;

	@Resource(name="testBean4")
	DataBean4 bean99;

	@GetMapping("/test9")
	public String test9(Model model) {
		bean9.setEng("100");
		bean9.setKor("100");
		bean9.setMat("100");

		bean99.setNo("100");
		bean99.setJumsu("대구");
		model.addAttribute("bean9", bean9);
		model.addAttribute("bean99", bean99);
		return "result";
	}

}
```

(2) RootAppContext.java

```
@Configuration
public class RootAppContext {

	@Bean
	@SessionScope
	public DataBean1 dataBean1() {
		return new DataBean1();
	}

	@Bean("testBean2")
	@SessionScope
	public DataBean2 dataBean2() {
		return new DataBean2();
	}

}
```

(3) views 동일

과제: XML방식으로 SessionScope를 설정하시오

13. ApplicationScope

[실습13] Java방식

(1) TestController4

```
@Controller
public class TestController4 {


// (0) HttpRequest에서 세션정보 가져와서 ServletContext에 저장하고 설정
	@GetMapping("/test0")
	public String test1(HttpServletRequest request) {
		ServletContext application = request.getServletContext();
		application.setAttribute("name0", "처음사용자0");
		return "test0";
	}


	@GetMapping("/result0")
	public String result1(HttpServletRequest request) {
		ServletContext application=request.getServletContext();
		System.out.println(application.getAttribute("name0"));
		return "result0";
	}

//(1) Autowired로 ServletContext객체 주입


	@Autowired
	ServletContext app1;

	@GetMapping("/test1")
	public String test1() {
		app1.setAttribute("name1", "홍길동1");
		return "test1";
	}

	@GetMapping("/result")
	public String result1() {
		return "result";
	}

//-------------------------------------------------------------
//(4) 데이터빈 객체 생성
	@GetMapping("test4")
	public String test4() {
		DataBean1 bean1 = new DataBean1();
		bean1.setId("400");
		bean1.setPw("444");

		app1.setAttribute("bean4", bean1);

		return "result";
	}


//-------------------------------------------------------------
// (8) RootAppContext.java에 Bean객체로 ApplicationScope으로 지정할 경우
	@Autowired
	DataBean1 bean8;

	@Resource(name="testBean2")
	DataBean2 bean88;

	@GetMapping("/test8")
	public String test8(Model model) {
		bean8.setId("chichi");
```

```
                bean8.setPw("12345");

                bean88.setName("choi");
                bean88.setJuso("대구");
                model.addAttribute("bean8", bean8);
                model.addAttribute("bean88", bean88);

                return "result";
        }
//-----------------------------------------------------------
// (9) DataBean3, DataBean4 클래스 상단에 @Component로 @ApplicationScope로 지정할 경우로 지정할 경우


        @Autowired
        DataBean3 bean9;

        @Resource(name="testBean4")
        DataBean4 bean99;

        @GetMapping("/test9")
        public String test9(Model model) {
                bean9.setEng("100");
                bean9.setKor("100");
                bean9.setMat("100");

                bean99.setNo("100");
                bean99.setJumsu("대구");
                model.addAttribute("bean9", bean9);
                model.addAttribute("bean99", bean99);
                return "result";
        }

}
```

(2) RootAppContext.java

```
@Configuration
public class RootAppContext {

        @Bean
        @ApplicationScope
        public DataBean1 dataBean1() {
                return new DataBean1();
        }

        @Bean("testBean2")
        @ApplicationScope
        public DataBean2 dataBean2() {
                return new DataBean2();
        }

}
```

(3) views 동일

과제: XML방식으로 ApplicationScope를 설정하시오

14. Cookie

- 사용자 웹브라우저에 저장되는 데이터
- 웹 브라우저는 쿠키에 저장된 정보를 서버에 전달하고, 응답결과로 쿠키 정보가 전달되면 웹브라우저가 쿠키에 저장

[실습14]

(1) TestController

```
@Controller
public class TestController {

	@GetMapping("/test")
	public String test(HttpServletRequest request, Model model) {
		try {
			Cookie[] cookies=request.getCookies();
			for(Cookie cookie : cookies) {
				if(cookie.getName().equals("id"))
				model.addAttribute("id", cookie.getValue());
				if(cookie.getName().equals("name")) {
					String strname= URLDecoder.decode(cookie.getValue());
					model.addAttribute("name", strname);
				}
			}
		}catch(Exception e) {
			e.printStackTrace();
		}
		return "test";
	}

	@GetMapping("/login")
	public String save_cookie(HttpServletRequest request,HttpServletResponse response) {
		try {
			String id=URLEncoder.encode(request.getParameter("id"),"utf-8");
			String name=URLEncoder.encode("홍길동","utf-8");

			Cookie cookid=new Cookie("id", id);
			Cookie cookname=new Cookie("name", name);

			cookid.setMaxAge(365*24*60*60);
			cookname.setMaxAge(365*24*60*60);

			response.addCookie(cookid);
			response.addCookie(cookname);
		} catch (Exception e) {
			// TODO: handle exception
		}
		return "redirect:/test";
	}

	@GetMapping("/load_cookie")
	public String load_cookie(HttpServletRequest request, Model model) {
		try {
			Cookie[] cookies=request.getCookies();
			for(Cookie cookie : cookies) {
				String strname=cookie.getName();
				String strvalue=URLDecoder.decode(cookie.getValue(), "utf-8");
				System.out.println(strname+":"+strvalue);
			}
		}catch(Exception e) {
			e.printStackTrace();
		}
		return "load_cookie";
	}


	@GetMapping("/logout")
	public String logout(HttpServletRequest request, HttpServletResponse response) {

		Cookie[] cookies=request.getCookies();
		for(Cookie cookie : cookies) {
```

```
                    Cookie cook=new Cookie(cookie.getName(), "");
                    cook.setMaxAge(0);
                    response.addCookie(cook);
                }
                System.out.println("모두삭제되었습니다.");
                return "redirect:/test";
        }
}
```

(2) views

```
//index.jsp--------------------------------
<h1>index</h1>
<a href="test">test</a>
</body>
//test.jsp---------------------------------
<body>
<form action="login">
${name } 아이디<input type="text" name="id" value="${id }"/>
비번<input type="text" name="pw"/>
<input type="submit" value="Login"/><br/>
</form>
        <a href="logout">Logout</a><br/>
        <a href="load_cookie">load_cookie</a><br/>
</body>
//load_cookie.jsp--------------------------
<body>
        <h1>load_cookie</h1>
</body>
```

15. Properties 와 Message

[Properties 파일]

프로그램 실행 중 변하지 않는 값들은 별도의 properties 파일에 작성하여 사용할 수 있다.

@ProPertyScource, @PropertySources

@Value : properties파일에 작성한 값을 주입

[Message]

Properties파일을 Message로 등록하면 JSP에서도 사용가능

MessageSource객체를 이용해서 properties파일을 등록해주면 Message로 등록할 수 있다

이때 ReloadableResourceBundleMessageSource를 사용하여 일정시간마다 한번씩 갱신되도록 한다

Locale을 지정하면 다국어 처리가 가능

[실습15]

(1) db.properties, data.properties, data_ko.properties, data_en.properties

[WEB-INF-properties폴더]

```
db.classname = oracle.jdbc.OracleDriver
db.url = jdbc:oracle:thin:@127.0.0.1:1521:xe
db.username = system
db.password = 123456
```

```
dataBean1.id=hong
dataBean1.name=홍길동
dataBean1.temp=나이는 {0}이고 성별은 {1}입니다.
jumsu.kor=100
jumsu.eng=100
```

```
dataBean1.lo=한국어~~
```

```
dataBean1.lo=english~~
```

(2) TestController.java

```
@Controller
//(1)Properties파일을 @PropertySources을 이용하여 주입받기(jsp에서는 사용안됨)
@PropertySources(@PropertySource("/WEB-INF/properties/db.properties"))
```

```java
public class TestController {


	@Value("${db.classname}")
	private String db_classname;

	@Value("${db.url}")
	private String db_url;

	@Value("${db.username}")
	private String db_username;

	@Value("${db.password}")
	private String db_password;


	@GetMapping("/test1")
	public String test1() {
		System.out.println(db_classname);
		System.out.println(db_url);
		System.out.println(db_username);
		System.out.println(db_password);
		return "test1";
	}

//(2) ServletAppContext.java에 Properties파일을 Message로 등록하고 test2.jsp에서 바로 사용
	@GetMapping("/test2")
	public String test2() {
		return "test2";
	}

//(3) ServletAppContext.java에 Properties파일을 Message로 등록하고, MesssageSource를 주입받아 컨트롤러
내에서 사용
	@Autowired
	ReloadableResourceBundleMessageSource res;

	@GetMapping("/test3")
	public String test3(Model model, Locale locale) {

		System.out.printf("locale :%s\n", locale);

		String a1=res.getMessage("dataBean1.id", null, null);
		String a2=res.getMessage("dataBean1.name", null, locale);
		System.out.println(a1);
		System.out.println(a2);

		Object [] args= {30,"홍길동"};
		String a3=res.getMessage("dataBean1.temp", args,null);
		System.out.println(a3);

		model.addAttribute("args", args);
		return "test3";
	}

}
```

(3) ServletAppContext.java

```java
/* properties를 여러개 읽어오는 과정에서 인식 안되는 문제를 해결하기 위함 */
@Bean
public static PropertySourcesPlaceholderConfigurer PropertySourcesPlaceholderConfigurer() {
	return new PropertySourcesPlaceholderConfigurer();
}

//Properties 파일을 Message 로 등록한다
@Bean
public ReloadableResourceBundleMessageSource messageSource() {
	ReloadableResourceBundleMessageSource res = new ReloadableResourceBundleMessageSource();
	res.setBasename("/WEB-INF/properties/data");		//확장자 생략
	//res.setBasenames("/WEB-INF/properties/db", "/WEB-INF/properties/data");
	return res;
}
```

(4) views

```
//index.jsp
<body>
<h1>index</h1>
<h2>properties와 Message</h2>
          <a href="test1">test1</a><br/>
          <a href="test2">test2</a><br>
          <a href="test3">test3</a><br>
</body>
//test1.jsp
<body>
<h1>test1</h1>
</body>
//test2.jsp
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<h1>test2</h1>
아이디:<spring:message code="dataBean1.id" /><br/>
이름:<spring:message code="dataBean1.name" /><br/>
국어:<spring:message code="jumsu.kor" />,
영어:<spring:message code="jumsu.eng" /><br/>
</body>
//test3.jsp
<body>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<h1>test3</h1>
아이디:<spring:message code="dataBean1.id" /><br/>
이름:<spring:message code="dataBean1.name" /><br/>
기타:<spring:message code="dataBean1.temp" arguments="${args }"/><br/>
기타:<spring:message code="dataBean1.lo" /><br/>
국어:<spring:message code="jumsu.kor" />,
영어:<spring:message code="jumsu.eng" /><br/>
</body>
```

16. 유효성검사

- 유효성 검사를 위해 hibernate-validator를 이용

JSR-303 https://beanvalidation.org/1.0/spec/

@Max(값)	값보다 큰 값이 들어오면 오류
@Min(값)	값보다 작은 값이 들어오면 오류
@Size(min=글자수,max=글자수)	지정된 글자수 보다 짧거나 길면 오류가 발생
@NotNull	값이 들어오지 않으면 오류가 발생
@Null	값이 들어오면 오류가 발생
@AssertTrue	true가 아닌 값이 들어오면 오류
@AssertFalse	false가 아닌 값이 들어오면 오류
@DecimalMax(value=값, inclusive=true/false)	값보다 작거나 같은 값이 들어와야 함 Inclusive가 false면 값은 포함하지 않기 때문에 작은 값이 들어와야 함. 생략하면 true
@DecimalMin(value=값, inclusive=true/false)	값보다 크거나 같은 값이 들어와야 함 Inclusive가 false면 값은 포함하지 않기 때문에 큰 값이 들어와야 함. 생략하면 true

@Digits(integer=자릿수,fraction=자릿수)	지정된 자릿수의 숫자가 아닐 경우 오류가 발생. Integer – 정수 자릿수, fraction – 실수 자릿수
@Pattern(regexp=정규식)	주어진 정규식에 위배되면 오류 발생

JSR-380 추가규격 https://beanvalidation.org/2.0/spec/

@NotEmpty	주입된 값의 길이가 0이면 오류 발생. 공백도 글자로 인식
@NotBlank	주입된 값이 공백을 제거하고 길이가 0이면 오류 발생.
@Positive	양수가 아니라면 오류 발생
@PositiveOrZero	0 또는 양수가 아니라면 오류 발생
@Negative	음수가 아니라면 오류 발생.
@NegativeOrZero	0 또는 음수가 아니라면 오류 발생.
@Email	이메일 형식이 아니라면 오류 발생. 중간에 @가 있는지 정도만 확인

@Valid : 컨트롤러의 메서드에서 주입받는 Bean객체에 @Valid 설정하면 유효성 검사 실시함

[실습 16] spring 태그 라이브러리 이용하거나 form 태그 라이브러리 이용

(1) pom.xml 라이브러리 추가

```
<!-- https://mvnrepository.com/artifact/javax.validation/validation-api -->
<dependency>
	<groupId>javax.validation</groupId>
	<artifactId>validation-api</artifactId>
	<version>2.0.1.Final</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
<dependency>
	<groupId>org.hibernate.validator</groupId>
	<artifactId>hibernate-validator</artifactId>
	<version>6.1.2.Final</version>
</dependency>
```

(2) DataBean1.java

[kr.co.korea.beans] 폴더

```
public class DataBean1 {
	@Size(min=2, max=10)
	@Pattern(regexp = "[a-zA-Z0-9]*")
	private String id;

	@Size(min=2, max=10)
	@Pattern(regexp = "[가-힣]*")
	private String name;

	@Max(100)
	private String kor;
```

```
	@Max(100)
	private String eng;

	@NotNull
	private String level;

	@Email
	private String email;

	public String getId() {
		return id;
	}
 ;이하생략
```

(3) TestController2

```
@Controller
public class TestController2 {
//(1) spring 태그 라이브러리 이용
	@GetMapping("/input_data1")
	public String input_data1() {
		return "input_data1";
	}

	@PostMapping("/input_pro1")
	public String input_pro1(@Valid DataBean1 dataBean1, BindingResult result) {

		// 유효성 검사에서 위반된 부분이 있다면..
		if(result.hasErrors()) {
//			for(ObjectError obj : result.getAllErrors()) {
//			    System.out.println("DefaultMessage():"+obj.getDefaultMessage());
//			    String [] codes = obj.getCodes();
//			    for(String c1 : codes) {
//				System.out.println("codes:"+c1);
//			    }
			return "input_data1";	//에러가 있으면
		}
		else return "input_success";  //에러가 없으면
	}

//(2) Form 태그 라이브러리 이용
	@GetMapping("/input_data2")
	public String input_data2(DataBean1 dataBean1) {
		return "input_data2";
	}

	@PostMapping("/input_pro2")
	public String input_pro2(@Valid DataBean1 dataBean1, BindingResult result) {

		if(result.hasErrors()) {
			return "input_data2";	//에러가 있으면

		}
		else return "input_success";  //에러가 없으면
	}

}
```

(4) views

```
//index.jsp ------------------------------------------------------------
</head>
<hr/>
<h2>유효성검사</h2>
	<a href="input_data1">input_data1</a><br/>
	<a href="input_data2">input_data2</a><br/>
</body>
//input_data1.jsp ----------------------------------------spring 태그 라이브러리 이용----------
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

```
<%@ taglib prefix='c' uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix='spring' uri="http://www.springframework.org/tags" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
	<h1>input_data1</h1>
		<form action='input_pro1' method='post'>
		id <input type='text' name='id'/> <br/>
		<spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('id') }">
				${errors.getFieldError('id').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>
		<br/>
		name : <input type='text' name='name'/><br/>
		<spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('name') }">
				${errors.getFieldError('name').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>
		<br/>
		kor : <input type='text' name='kor'/><br/>
		<spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('kor') }">
				${errors.getFieldError('kor').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>
		<br/>
		eng : <input type='text' name='eng'/><br/>
		<spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('eng') }">
				${errors.getFieldError('eng').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>
		<br/>
		level : <input type="checkbox" name="level" value="middle"/>중학생
		   <input type="checkbox" name="level" value="high"/>고등학생
		   <input type="checkbox" name="level" value="big"/>대학생 <br/>
	    <spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('level') }">
				${errors.getFieldError('level').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>

		email : <input type='text' name='email'/><br/>
		<spring:hasBindErrors name="dataBean1">
			<c:if test="${errors.hasFieldErrors('email') }">
				${errors.getFieldError('email').defaultMessage }<br/>
			</c:if>
		</spring:hasBindErrors>
		<br/>
		<button type='submit'>확인</button>
	</form>
</body>
</html>


//input_success.jsp ------------------------------------------------------------
<body>
<h1>input_success</h1>
<h3>id: ${dataBean1.id }</h3>
<h3>name: ${dataBean1.name }</h3>
<h3>kor: ${dataBean1.kor }</h3>
<h3>eng: ${dataBean1.eng }</h3>
<h3>level: ${dataBean1.level }</h3>
<h3>email: ${dataBean1.email }</h3>
</body>


//input_data2.jsp ---------------------------------------form 태그 라이브러리 이용----------
<%@ page language="java" contentType="text/html; charset=UTF-8"
```

```
    pageEncoding="UTF-8"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
        <h1>input_data2</h1>
        <form:form action="input_pro2" modelAttribute="dataBean1" method="post">
        id <form:input path="id" type="text" /> <br/>
        <form:errors path="id"/><br/>
        name : <form:input path="name" type="text" /><br/>
        <form:errors path="name"/><br/>
        kor : <form:input path="kor" type="text" /><br/>
        <form:errors path="kor"/><br/>
        eng : <form:input path="eng" type="text" /><br/>
        <form:errors path="eng"/><br/>
        level : <form:checkbox path="level" value="middle"/>중학생
        <form:checkbox path="level" value="high"/>고등학생
        <form:checkbox path="level" value="big"/>대학생 <br/>
        <form:errors path="level"/><br/>
        mail : <form:input path="email" type="text" /><br/>
        <form:errors path="email"/><br/>
        <button type='submit'>확인</button>
        </form:form>
</body>
</html>
```

(5) properties 파일 활용 (form:form 태그일 때 유용함)

1) error_message.properties 파일 [WEB-INF-properties]폴더

```
Size.dataBean1.id=아이디는 2~10글자를 입력해야 합니다.
Pattern.dataBean1.id=아이디는 영문자(대소문자)또는 숫자를 입력해야 합니다.
Size.dataBean1.name=이름은 2~10글자를 입력해야 합니다.
Pattern.dataBean1.name=이름은 한글이어야 합니다.
NotEquals.dataBean1.pw= 비밀번호가 일치하지 않습니다.
NotEquals.dataBean1.pw2= 비밀번호가 일치하지 않습니다.
Max.dataBean1.kor=국어는 100이하의 정수를 넣어야 합니다.
Max.dataBean1.eng=영어는 100이하의 정수를 넣어야 합니다.
NotNull.dataBean1.level=학력은 반드시 선택해야 합니다.
Email.dataBean1.email=이메일 형식으로 입력해야 합니다.
```

2) ServletAppContext.java 에 등록

```
//Properties 파일을 Message 로 등록한다
@Bean
public ReloadableResourceBundleMessageSource messageSource() {
        ReloadableResourceBundleMessageSource res = new ReloadableResourceBundleMessageSource();
        res.setBasenames("/WEB-INF/properties/data","/WEB-INF/properties/error_message");
        //res.setBasename("/WEB-INF/properties/data");
        return res;
}
```

17. Interceptor

Controller가 메소드를 호출하기 전이나 후에 다른 메소드를 호출하도록 가로채는 기법
요청 발생시 호출되는 메서드의 코드가 중복되는 부분이 있을 때 Interceptor를 통해 처리하게 된다.
로그인 여부 확인, 등급별 서비스 사용 권한 확인 등의 작업을 처리할 때 사용
- Interceptor는 HandlerInterceptor 인터페이스를 구현하거나 HandlerInterceptorAdapter를 상속받은 클래스를 만들고 다음 메서드를 구현한다.
① **preHandle** : Controller의 메서드가 호출되기 전 호출되고, 이 메서드가 false를 반환하면 코드의 흐름이 중단된다. ② Controller 메서드 수행
③ **postHandle** : Controller의 메서드의 수행이 완료되고 view 처리를 수행하기 전에 호출된다.
④ view 처리
⑤ **afterCompletion** : view 처리까지 완료되고 응답결과가 브라우저로 전달되기 전에 호출된다.

patern 등록	Java 방법	Xml 방법
pattern 추가	addPathPatterns	<mapping>
pattern 배제	excludePathPatterns	<exclude-mapping>

(1) 인터셉터 파일 작성
1) TestLoginInterceptor.java (HandlerInterceptor 인터페이스로부터 상속)
[kr.co.korea.intercepter]패키지

```
public class TestLoginInterceptor implements HandlerInterceptor {

	@Override
	public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
			throws Exception {
		System.out.println("-----------------------------------------------");
		System.out.println("(1)preHandle Login test");
		return true;
	}

	@Override
	public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
			ModelAndView modelAndView) throws Exception {
		System.out.println("(3)postHandle Login test");
	}

	@Override
	public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception
ex)
			throws Exception {
		System.out.println("(5)afterCompletion Login test");
		System.out.println("-----------------------------------");
	}


}
```

2) TestWriterInterceptor.java (HandlerInterceptor 인터페이스로부터 상속)
[kr.co.korea.intercepter]패키지

```
public class TestWriterInterceptor implements HandlerInterceptor {
	@Override
	public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
			throws Exception {
		System.out.println("(1)preHandle Writer test");
		return true;
	}

	@Override
	public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
			ModelAndView modelAndView) throws Exception {
		System.out.println("(3)postHandle Writer test");
```

```
	}

	@Override
	public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception
ex)
				throws Exception {
		System.out.println("(5)afterCompletion Writer test");
	}
}
```

(2) ServletAppContext.java

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
	// TODO Auto-generated method stub
	WebMvcConfigurer.super.addInterceptors(registry);

	TestLoginInterceptor testLoginInterceptor=new TestLoginInterceptor();
	InterceptorRegistration reg1=registry.addInterceptor(testLoginInterceptor);
	reg1.addPathPatterns("/user/modify", "/user/logout", "/board/*");
	reg1.excludePathPatterns("/board/main");
	//reg1.addPathPatterns("/**");

	TestWriterInterceptor testWriterInterceptor=new TestWriterInterceptor();
	InterceptorRegistration reg2=registry.addInterceptor(testWriterInterceptor);
	reg2.addPathPatterns("/board/modify", "/board/delete");
	// board/modify 수행시 2회 수행됨. reg1에  "board/*" 등록되어 있음
}
```

(3) TestController.java

```
@Controller
public class TestController {

	@GetMapping("/user/modify")
	public String test1() {
		System.out.println("(2)/user/modify 컨트롤러");
		return "/user/modify";
	}

	@GetMapping("/user/logout")
	public String test2() {
		System.out.println("(2)/user/logout 컨트롤러");
		return "/user/logout";
	}

	@GetMapping("/board/write")
	public String test3() {
		System.out.println("(2)/board/write 컨트롤러");
		return "/board/write";
	}

	@GetMapping("/board/main")
	public String test4() {
		System.out.println("(2)/board/main 컨트롤러");
		return "/board/main";
	}

	@GetMapping("/board/modify")
	public String test5() {
		System.out.println("(2)/board/modify 컨트롤러");
		return "/board/modify";
	}
	@GetMapping("/board/delete")
	public String test6() {
		System.out.println("(2)/board/delete 컨트롤러");
		return "/board/delete";
	}
}
```

(4) views ([board]폴더-delete,main,modify,write), ([user]폴더-login,logout,modify)

```
//index.jsp  --------------------------------
<body>
        <a href='user/modify'>user/modify</a><br/>
        <a href='user/logout'>user/logout</a><br/>
        <a href='board/write'>board/write</a><br/>
        <a href='board/main'>board/main</a><br/>
        <a href='board/modify'>board/modify</a><br/>
        <a href='board/delete'>board/delete</a><br/>
</body>
//board/delete.jsp  ---------------------------------
<body>
<h1>(4)board/delete</h1>
</body>
//board/main.jsp  ---------------------------------
<body>
<h1>(4)board/main</h1>
</body>
//board/modify.jsp  ---------------------------------
<body>
<h1>(4)board/modify</h1>
</body>
//board/write.jsp  ---------------------------------
<body>
<h1>(4)board/write</h1>
</body>
//user/login.jsp  ---------------------------------
<body>
<h1>(4)user/login </h1>
</body>
//user/logout.jsp  ---------------------------------
<body>
<h1>(4)user/logout</h1>
</body>
//user/modify  ---------------------------------
<body>
<h1>(4)user/modify</h1>
</body>
```

18. 예외처리

오류발생할 경우 보여줄 오류페이지를 작성해두고, 오류발생시 오류페이지를 브라우저로 전달

Controller에서 @ExceptionHandler를 통해 메서드 정의해 주면 오류 발생시 이 메서드를 자동으로 호출

- @ExceptionHandler 는 Controller마다 작성
- Global Exception Handler 구현 : ExceptionHandler중 해당 오류에 대한 것이 없다면 Global Exception Handler로 이동하여 예외에 관련된 처리해줌

(1) TestController 에 @ExceptionHandler 지정하는 방법

```
@Controller
public class TestController {
//(1) ArrayIndexOutOfBoundsException 에러있음
        @GetMapping("/test1")
        public String test1(Model model) {

                int [] array1 = {10, 20, 30};
                model.addAttribute("array1", array1[10]);  //에러
                return "test1";
        }


//(2) NullPointerException 에러있음
        @GetMapping("/test2")
        public String test2() {
```

```
                ArrayList<String> list = null;
                list.add("문자열");                          //에러

                return "test2";
        }


//      @ExceptionHandler(ArrayIndexOutOfBoundsException.class)    /* Exception.class */
//      public String exception1() {
//              return "error1";
//      }
//
//      @ExceptionHandler(NullPointerException.class)     /* Exception.class */
//      public String exception2() {
//              return "error2";
//      }

}
```

(2) 별도의 Exception 클래스 만들어서 exception 파일로 분기

- GlobalException.java (RuntimeException 클래스로부터 상속받기)

  [kr.co.korea.exception]

```
@ControllerAdvice
public class GlobalException extends RuntimeException{

        @ExceptionHandler(Exception.class)        //java.lang.NullPointerException.class
        public String   handleException() {
                return "error";
        }

}
```

(3) views

```
//index.jsp
<body>
<h1>index</h1>
<h3>예외처리</h3>
<hr/>
<a href='test1'>test1 배열크기 에러 </a><br/>
<a href='test2'>test2 널값 에러</a><br/>
</body>
//error1.jsp
<body>
        <h1>test1 의 ArrayIndexOutOfBoundsException 에러</h1>
</body>
//error2.jsp
<body>
        <h1>test2의 NullPointerException에러</h1>
</body>
//error.jsp
<body>
        <h1> Exception 에러</h1>
</body>
```

19. @RestController

Spring MVC에서 Controller를 구성할 때 @Controller를 사용하면 return 하는 값은 사용할 JSP를 지정하게 된다.

@RestController는 Restful API 구성을 위해 사용하는 Controller 이다.

Restful API 서버는 웹, 모바일 PC 등 다양한 플랫폼으로 데이터를 전달할 때 사용합니다.

(1) @RetController를 통해 return하게 되면 그 값 자체를 브라우저로 전달

```java
@RestController
public class RestTestController {
//(1)
	@GetMapping("/test3")
	public String test3() {
		return "test3";
	}

//(2) 자바의 객체나 리스트 등을 JSON 문서로 구성하여 값 자체를 브라우저로 전달
//(2) pom.xml 에서 jackson-databind 라이브러리 이용하여 데이터 전달가능
	@GetMapping("/test4")
	public ResponseEntity<ArrayList<DataBean>> test4() {
		DataBean bean1=new DataBean("111","111","홍길동");
		DataBean bean2=new DataBean("222","222","이기자");
		DataBean bean3=new DataBean("333","333","최고야");

		ArrayList<DataBean> list=new ArrayList<DataBean>();
		list.add(bean1);
		list.add(bean2);
		list.add(bean3);

		ResponseEntity<ArrayList<DataBean>> entry=new ResponseEntity<ArrayList<DataBean>>(list,
HttpStatus.OK);
		return entry;
	}

}
```

(2) 자바의 객체나 리스트 등을 JSON 문서로 구성하여 값 자체를 브라우저로 전달

JSON 문서로 구성하기 위한 라이브러리

```xml
		<dependency>
			<groupId>com.fasterxml.jackson.core</groupId>
			<artifactId>jackson-databind</artifactId>
			<version>2.11.2</version>
		</dependency>
```

(3) views

```html
<body>
<h3>RestController</h3>
<a href='test3'>test3파일로 이동하는 것이 아님</a><br/>
<a href='test4'>test4파일로 이동하는 것이 아님</a><br/>
</body>
```