

26장 스프링 애너테이션 기능

26.1 스프링 애너테이션이란?

26.2 스프링 애너테이션 이용해 URL 요청 실습하기

26.3 스프링 애너테이션 이용해 로그인 기능 구현하기

26.4 @Autowired 이용해 빈 주입하기

26.1 스프링 애너테이션이란?

스프링 애너테이션(Annotation)

- 기존에 XML에서 빈 설정을 애너테이션을 이용해서 자바 코드에서 설정하는 방법
- 기능이 복잡해짐에 따라 XML에서 설정하는 것보다 유지 보수에 유리함
- 현재 애플리케이션 개발 시 XML 설정 방법과 애너테이션 방법을 혼합해서 사용함

• 26.1.1 스프링 애너테이션 제공 클래스

브라우저 URL 요청 처리 애너테이션 관련 클래스

클래스	기능
DefaultAnnotationHandlerMapping	클래스 레벨에서 @RequestMapping을 처리합니다.
AnnotationMethodHandlerAdapter	메서드 레벨에서 @RequestMapping을 처리합니다.

26.1 스프링 애너테이션이란?

• 26.1.2 <context:component-scan> 태그 기능

- <context:component-scan> 태그를 사용해 패키지 이름을 지정하면 애플리케이션 실행 시 해당 패키지에서 애너테이션으로 지정된 클래스를 빈으로 만들어 줌

• 사용 형식

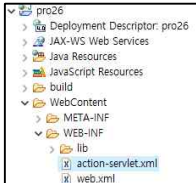
<context:component-scan base-package="패키지이름" />

여러 가지 스테레오 타입 애너테이션

애너테이션	기능
@Controller	스프링 컨테이너가 component-scan에 의해 지정한 클래스를 컨트롤러 빈으로 자동 변환합니다.
@Service	스프링 컨테이너가 component-scan에 의해 지정한 클래스를 서비스 빈으로 자동 변환합니다.
@Repository	스프링 컨테이너가 component-scan에 의해 지정한 클래스를 DAO 빈으로 자동 변환합니다.
@Component	스프링 컨테이너가 component-scan에 의해 지정한 클래스를 빈으로 자동 변환합니다.

26.2 스프링 애너테이션 이용해 URL 요청 실습

1. 새 프로젝트 pro26을 만들고 스프링 애너테이션 기능을 실습하기 위한 XML 설정 파일인 action-servlet.xml을 다음과 같이 추가합니다.



26.2 스프링 애너테이션 이용해 URL 요청 실습

2. action-servlet.xml을 다음과 같이 작성합니다.

코드 26-1 pro26/WebContent/WEB-INF/action-servlet.xml

```
...
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/test/" />
    <property name="suffix" value=".jsp"/>
</bean>
<bean class="org.springframework.web.servlet.mvc.annotation.
                                     DefaultAnnotationHandlerMapping"/>
<bean class="org.springframework.web.servlet.mvc.annotation.
                                     AnnotationMethodHandlerAdapter"/>
<context:component-scan base-package="com.spring"/>
</beans>
```

메서드 레벨에 @RequestMapping을 처리합니다.

클래스 레벨에 @RequestMapping을 처리합니다.

com.spring 패키지에 존재하는 클래스에 애너테이션이 적용되도록 설정합니다.

26.2 스프링 애너테이션 이용해 URL 요청 실습

3. 애너테이션 기능을 수행하는 자바 클래스와 JSP 파일을 추가합니다.



26.2 스프링 애너테이션 이용해 URL 요청 실습

4. MainController 클래스가 하는 일은 다음과 같습니다.

코드 26-2 pro26/src/com/spring/ex01/MainController.java

```
package com.spring.ex01;
...
@Controller("mainController")
@RequestMapping("/test")
public class MainController {
    @RequestMapping(value="/main1.do",method=RequestMethod.GET)
    public ModelAndView main1(HttpServletRequest request,
                               HttpServletResponse response) throws Exception {
        ModelAndView mav=new ModelAndView();
        mav.addObject("msg","main1");
        mav.setViewName("main");
        return mav;
    }
}
```

애너테이션이 적용되도록 하려면 해당 클래스가 반드시 <component-scan>에서 설정한 패키지나 하위 패키지에 존재해야 합니다.

@Controller 애너테이션을 이용해 MainController 클래스를 빈으로 자동 변환합니다. 빈 id는 mainController입니다.

@RequestMapping을 이용해 첫 번째 단계의 URL 요청이 /test이면 mainController 빈을 요청합니다.

@RequestMapping을 이용해 두 번째 단계의 URL 요청이 /main1.do이면 mainController 빈의 main1() 메서드에게 요청합니다. method=RequestMethod.GET으로 지정하면 GET 방식으로 요청 시 해당 메서드가 호출됩니다.

26.2 스프링 애너테이션 이용해 URL 요청 실습

```
@RequestMapping(value="/main2.do", method = RequestMethod.GET)
public ModelAndView main2(HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
    ModelAndView mav=new ModelAndView();
    mav.addObject("msg", "main2");
    mav.setViewName("main");
    return mav;
}
```

— @RequestMapping을 이용해 두 번째 단계의 URL 요청이 /main2.do이면 mainController 빈의 main2() 메서드에게 요청합니다.
method=RequestMethod.GET으로 지정하면 GET 방식으로 요청 시 해당 메서드가 호출됩니다.

26.2 스프링 애너테이션 이용해 URL 요청 실습

5. 다음은 컨트롤러에서 ModelAndView에 뷰이름으로 설정한 main.jsp입니다.

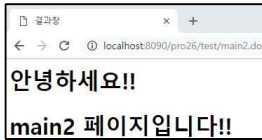
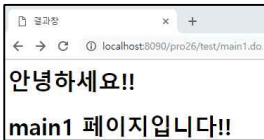
코드 26-3 pro26/WebContent/WEB-INF/views/test/main.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
    request.setCharacterEncoding("UTF-8");
%>
<html>
<head>
    <meta charset="UTF-8">
    <title>결과창</title>
</head>
<body>
    <h1>안녕하세요!!</h1>
    <h1>${msg} 페이지입니다!!</h1>
</body>
</html>
```

↑ 컨트롤러에서 넘긴 메시지를 출력합니다.

26.2 스프링 애너테이션 이용해 URL 요청 실습

6. <http://localhost:8090/pro26/test/main1.do>, <http://localhost:8090/pro26/test/main2.do>로 각각 요청하여 결과를 확인합니다.



26.3 스프링 애너테이션 이용해 로그인 기능

1. 다음은 로그인 기능과 관련된 자바 파일과 JSP 위치입니다.



26.3 스프링 애너테이션 이용해 로그인 기능

2. 실습 시 한글 깨짐 현상을 방지하기 위해 web.xml에 한글 필터 기능을 설정합니다.

코드 26-4 pro26/WebContent/WEB-INF/web.xml

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

26.3 스프링 애너테이션 이용해 로그인 기능

3. 스프링 애너테이션 기능을 이용해 로그인 시 전송된 ID와 이름을 JSP에 출력하도록 LoginController 클래스를 작성합니다.

코드 26-5 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;
...
@Controller("loginController")
public class LoginController {
    @RequestMapping(value = { "/test/loginForm.do", "/test/loginForm2.do" },
                    method={RequestMethod.GET})
    public ModelAndView loginForm(HttpServletRequest request,
                                HttpServletResponse response) throws Exception {
        ModelAndView mav=new ModelAndView();
        mav.setViewName("loginForm");
        return mav;
    }
    @RequestMapping(value = "/test/login.do", method={RequestMethod.GET,RequestMethod.POST})
    public ModelAndView login(HttpServletRequest request,
                              HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        ModelAndView mav=new ModelAndView();
        mav.setViewName("result");
    }
}
```

com.spring 하위 패키지에 클래스가 위치해야 애너테이션이 적용됩니다.

컨트롤러 빈을 자동으로 생성합니다.

/test/loginForm.do와 /test/loginForm2.do로 요청 시 loginForm()을 호출합니다.

GET 방식과 POST 방식 요청 시 모두 처리합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

```
String userID=request.getParameter("userID");
String userName =request.getParameter("userName");
mav.addObject("userID", userID);
mav.addObject("userName", userName);
return mav;
}
}
```

26.3 스프링 애너테이션 이용해 로그인 기능

4. 로그인창에서 ID와 이름을 전송하도록 loginForm.jsp를 다음과 같이 작성합니다.

코드 26-6 pro26/WebContent/WEB-INF/test/loginForm.jsp

```
...  
<form method="post" action="${contextPath}/test/login.do">  
  <table width="400">  
    <tr>  
      <td>아이디 <input type="text" name="userID" size="10"></td>  
    </tr>  
    <tr>  
      <td>이름 <input type="text" name="userName" size="10"></td>  
    </tr>  
  </table>  
  <input type="submit" value="로그인" />  
</form>  
...
```

로그인 클릭 시 /test/login.do로 요청합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

5. 로그인창에서 전송된 ID와 이름이 결과창에 나타나도록 result.jsp를 다음과 같이 작성합니다.

코드 26-7 pro26/WebContent/WEB-INF/test/result.jsp

```
...  
<h1>아이디 : ${userID}</h1>  
<h1>이름 : ${userName}</h1>  
...
```

6. <http://localhost:8090/pro26/test/loginForm.do>로 요청하여 로그인창에 ID와 이름을 입력한 후 로그인을 클릭합니다.

로그인창

← → ↻ ⓘ localhost:8090/pro26/test/loginForm.do

아이디

이름

26.3 스프링 애너테이션 이용해 로그인 기능

7. <http://localhost:8090/test/login.do>로 요청을 보냅니다.



26.3 스프링 애너테이션 이용해 로그인 기능

- **26.3.1 메서드에 @RequestParam 적용하기**

@RequestParam

- 매개변수의 수가 많아지면 일일이 `getParameter()` 메서드를 이용하는 방법은 불편함
- @RequestParam을 메서드에 적용해 쉽게 값을 얻을 수 있음

26.3 스프링 애너테이션 이용해 로그인 기능

1. spring.ex02 패키지를 만들고 LoginController 클래스를 다음과 같이 작성합니다.

코드 26-8 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;

...
@Controller("loginController")
public class LoginController {

...
    @RequestMapping(value = "/test/login2.do",
                    method = { RequestMethod.GET, RequestMethod.POST })
    public ModelAndView login(@RequestParam("userID") String userID ,
                             @RequestParam("userName") String userName,
                             HttpServletRequest request, HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        ModelAndView mav = new ModelAndView();
        mav.setViewName("result");

        /*
        String id = request.getParameter("userID");
        String name = request.getParameter("userName");
        */
    }
}
```

@RequestParam을 이용해 매개변수가 userID이면 그 값을 변수 userID에 자동으로 설정합니다.

@RequestParam을 이용해 매개변수가 userName이면 그 값을 변수 userName에 자동으로 설정합니다.

getParameter() 메서드를 이용할 필요가 없습니다.

26.3 스프링 애너테이션 이용해 로그인 기능

```

        System.out.println("userID: "+userID);
        System.out.println("userName: "+userName);
        mav.addObject("userID", userID);
        mav.addObject("userName", userName);
        return mav;
    }
}

```

2. 로그인창에서 ID와 이름을 컨트롤러로 전송하도록 loginForm.jsp를 작성합니다.

코드 26-9 pro26/WebContent/WEB-INF/test/loginForm.jsp

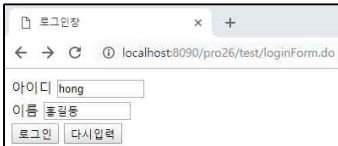
```

...
<form method="post" action="${contextPath}/test/login2.do">
    <table width="400">
        <tr>
            <td>아이디 <input type="text" name="userID" size="10"></td>
            @RequestParam에서 설정한 userID와 같습니다.
        </tr>
        <tr>
            <td>이름 <input type="text" name="userName" size="10"></td>
            @RequestParam에서 설정한 userName과 같습니다.
        </tr>
    </table>
...

```

26.3 스프링 애너테이션 이용해 로그인 기능

3. <http://localhost:8090/pro26/test/loginForm.do>로 요청하여 ID와 이름을 입력하고 로그인을 클릭합니다.



로그인창

localhost:8090/pro26/test/loginForm.do

아이디

이름

4. 그러면 /test/login2.do로 요청을 보내어 다음과 같은 결과 화면을 출력합니다



결과창

localhost:8090/pro26/test/login2.do

아이디 : hong

이름 : 홍길동

26.3 스프링 애너테이션 이용해 로그인 기능

- **26.3.2 @RequestParam의 required 속성 사용하기**

@RequestParam의 required 속성

- @RequestParam 적용 시 required 속성을 생략하면 기본값은 true임
- required 속성을 true로 설정하면 메서드 호출 시 반드시 지정한 이름의 매개변수를 전달해야함 (매개변수가 없으면 예외가 발생)
- required 속성을 false로 설정하면 메서드 호출 시 지정한 이름의 매개변수가 전달되면 값을 저장하고 없으면 null을 할당함

26.3 스프링 애너테이션 이용해 로그인 기능

1. LoginController 클래스를 다음과 같이 작성합니다.

코드 26-10 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;
...
@Controller("loginController")
public class LoginController {
    ...
    @RequestMapping(value = "/test/login2.do",
        method = { RequestMethod.GET, RequestMethod.POST })
    public ModelAndView login2(@RequestParam("userID") String userID,
        @RequestParam(value="userName", required=true) String userName,
        @RequestParam(value="email", required=false) String email,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        ModelAndView mav = new ModelAndView();
        mav.setViewName("result");
        System.out.println("userID: "+userID);
        System.out.println("userName: "+userName);
        System.out.println("email: "+ email);
        mav.addObject("userID", userID);
        mav.addObject("userName", userName);
        return mav;
    }
}
```

required 속성을 생략하면 required
의 기본값은 true입니다.

required 속성을 명시적으로 true로
설정합니다.

required 속성을 명시적으로 false로
설정합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

2. 로그인창에서 <hidden> 태그를 이용해 이메일 정보를 컨트롤러로 전송합니다.

코드 26-11 pro26/WebContent/WEB-INF/test/loginForm.jsp

...

```
<form method="post" action="${contextPath}/test/login2.do">
```

```
<input type="hidden" name="email" value="hong@test.com" />
```

```
<table width="400">
```

```
<tr>
```

```
<td>아이디 <input type="text" name="userID" size="10"></td>
```

```
</tr>
```

```
<tr>
```

```
<td>이름 <input type="text" name="userName" size="10"></td>
```

```
</tr>
```

...

_____ <hidden> 태그를 이용해 이메일 정보를 전송합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

3. 브라우저에 요청하여 로그인창이 나타나면 ID와 이름을 입력하고 로그인을 클릭하면 콘솔에 이메일 정보를 출력합니다.

```
userID: hong  
userName: 1234  
email: hong@test.com
```

4. 로그인창에서 이메일 관련 <hidden> 태그를 주석 처리한 후 요청하면 이메일 정보를 null로 출력합니다.

```
userID: hong  
userName: 1234  
email: null
```

26.3 스프링 애너테이션 이용해 로그인 기능

- **26. 3.3 @RequestParam 이용해 Map에 매개변수 값 설정하기**

- 전송되는 매개변수의 수가 많을 경우 Map에 바로 저장해서 사용하면 편리

26.3 스프링 애너테이션 이용해 로그인 기능

1. LoginController 클래스를 다음과 같이 작성합니다.

코드 26-12 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;

...
@Controller("loginController")
public class LoginController {

    ...
    @RequestMapping(value = "/test/login3.do",
                    method = { RequestMethod.GET, RequestMethod.POST })
    public ModelAndView login3(@RequestParam Map<String, String> info,
                              HttpServletRequest request,
                              HttpServletResponse response) throws Exception {

        request.setCharacterEncoding("utf-8");
        ModelAndView mav = new ModelAndView();
        String userID = info.get("userID");
        String userName = info.get("userName");
        System.out.println("userID: " + userID);
        System.out.println("userName: " + userName);
        mav.addObject("info", info);
        mav.setViewName("result");
        return mav;
    }
}
```

@RequestParam을 이용해 Map에 전송된 매개변수 이름을 key, 값을 value로 저장합니다.

Map에 저장된 매개변수의 이름으로 전달된 값을 가져옵니다.

@RequestParam에서 설정한 Map 이름으로 바인딩합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

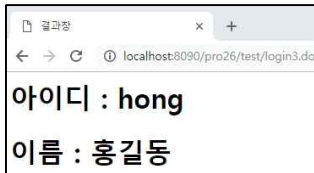
2. 결과창에서 바인딩한 속성 이름으로 출력합니다.

코드 26-13 pro26/WebContent/WEB-INF/views/test/result.jsp

```
...  
<h1>아이디 : ${info.userID}</h1>  
<h1>이름 : ${info.userName}</h1>  
...
```

Map에 key로 접근하여 값을 출력합니다.

3. 로그인창에서 아이디와 이름을 입력한 후 로그인을 클릭하면 /test/login3.do로 요청하여 결과를 출력합니다.



26.3 스프링 애너테이션 이용해 로그인 기능

• 26.3.4 @ModelAttribute 이용해 VO에 매개변수 값 설정하기

1. LoginController 클래스를 다음과 같이 작성합니다.

코드 26-14 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;

...
@Controller("loginController")
public class LoginController {
    ...
    @RequestMapping(value = "/test/login4.do",
                    method = { RequestMethod.GET, RequestMethod.POST })
    public ModelAndView login4(@ModelAttribute("info") LoginVO loginVO,
                               HttpServletRequest request,
                               HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        ModelAndView mav = new ModelAndView();
        System.out.println("userID: "+loginVO.getUserID());
        System.out.println("userName: "+loginVO.getUserName());
        mav.setViewName("result");
        return mav;
    }
}
```

@ModelAttribute를 이용해 전달되는 매개변수 값을 LoginVO 클래스와 이름이 같은 속성에 자동으로 설정합니다.
addObject()를 이용할 필요 없이 info를 이용해 바로 JSP에서 LoginVO 속성에 접근할 수 있습니다.

26.3 스프링 애너테이션 이용해 로그인 기능

2. LoginVO 클래스를 다음과 같이 작성합니다. 로그인창에서 입력한 ID와 비밀번호를 속성에 저장합니다.

코드 26-15 pro26/src/com/spring/ex02/LoginVO.java

```
package com.spring.ex02;


public class LoginVO {
    private String userID;
    private String userName;
    //각 값에 대한 getter/setter 메서드
    ...
}
```

3. 로그인창에 입력한 ID와 비밀번호가 출력되도록 result.jsp를 작성합니다.

코드 26-16 pro26/WebContent/WEB-INF/views/test/result.jsp

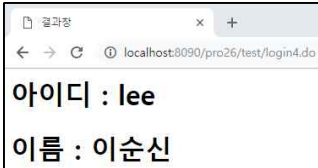
```
...
<h1>아이디 : ${info.userID} </h1>
<h1>이름 : ${info.userName} </h1>
...

```

 @ModelAttribute("info")에서 지정한 이름으로 속성에 접근합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

4. 로그인창에서 로그인을 하면 ID와 이름을 출력합니다.



26.3 스프링 애너테이션 이용해 로그인 기능

- **26.3.5 Model 클래스 이용해 값 전달하기**

Model 클래스

- Model 클래스를 이용하면 메서드 호출 시 JSP로 값을 바로 바인딩하여 전달할 수 있음
- Model 클래스의 addAttribute() 메서드는 ModelAndView의 addObject() 메서드와 같은 기능 수행
- Model 클래스는 따로 뷰 정보를 전달할 필요가 없을 때 사용하면 편리함

26.3 스프링 애너테이션 이용해 로그인 기능

1. LoginController 클래스를 다음과 같이 작성합니다.

코드 26-17 pro26/src/com/spring/ex02/LoginController.java

```
package com.spring.ex02;
import org.springframework.ui.Model;
...
@Controller("loginController")
public class LoginController {
    ...
    @RequestMapping(value = "/test/login5.do",
                    method = { RequestMethod.GET, RequestMethod.POST })
    public String login5(Model model,
                        HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        model.addAttribute("userID", "hong");
        model.addAttribute("userName", "홍길동");
        return "result";
    }
}
```

메서드 호출 시 Model 클래스 객체를 생성합니다.

JSP에 전달할 데이터를 model에 addAttribute() 메서드를 이용해서 바인딩합니다.

26.3 스프링 애너테이션 이용해 로그인 기능

2. 로그인창에서 입력한 ID와 비밀번호가 출력되도록 result.jsp를 작성합니다.

코드 26-18 pro26/WebContent/WEB-INF/views/test/result.jsp

```
...  
<h1>아이디 : ${userID}</h1>  
<h1>이름 : ${userName}</h1>  
...
```

3. 다음은 로그인창에서 요청한 실행 결과입니다.



26.4 @Autowired 이용해 빈 주입하기

- XML에서 빈을 설정한 후 애플리케이션이 실행될 때 빈을 주입해서 사용하면 XML 파일이 복잡해지면서 사용 및 관리가 불편함

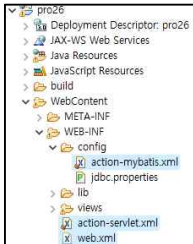
```
<bean id="memberController"
      class="com.spring.member.controller.MemberControllerImpl">
  <property name="methodNameResolver">
    <ref local="memberMethodResolver"/>
  </property>
  <property name="memberService" ref="memberService"/>
</bean>
<bean id="memberService" class="com.spring.member.service.MemberServiceImpl">
  <property name="memberDAO" ref="memberDAO"/>
</bean>
```

@Autowired의 특징

- 기존 XML 파일에서 각각의 빈을 DI로 주입했던 기능을 코드에서 애너테이션으로 자동으로 수행함
- @Autowired를 사용하면 별도의 setter나 생성자 없이 속성에 빈을 주입할 수 있음

26.4 @Autowired 이용해 빈 주입하기

1. 먼저 회원 관리 기능과 관련된 XML 파일을 설정하겠습니다. action-mybatis.xml과 jdbc.properties 파일은 24장의 파일을 복사해 config 폴더에 붙여 넣습니다.



26.4 @Autowired 이용해 빈 주입하기

2. web.xml을 다음과 같이 작성합니다. ContextLoaderListener를 이용해 애플리케이션이 실행될 때 action-mybatis.xml을 읽어 들이도록 설정합니다.

코드 26-19 pro26/WebContent/WEB-INF/web.xml

```
...
<listener>
  <listener-class> org.springframework.web.context.ContextLoaderListener</
listener-class>
</listener>
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/config/action-mybatis.xml
  </param-value>
</context-param>
...
```

26.4 @Autowired 이용해 빈 주입하기

3. action-servlet.xml의 JSP 경로를 /WEB-INF/views/member/로 변경합니다.

코드 26-20 pro26/WebContent/WEB-INF/action-servlet.xml

```
<!-- JSP 경로 변경 -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/member/" />
    <property name="suffix" value=".jsp"/>
</bean>
<bean class="org.springframework.web.servlet.mvc.annotation.
    DefaultAnnotationHandlerMapping"/>
<bean class="org.springframework.web.servlet.mvc.annotation.
    AnnotationMethodHandlerAdapter"/>
<context:component-scan base-package="com.spring"/>
</beans>
```

JSP 경로를 변경합니다.

26.4 @Autowired 이용해 빈 주입하기

4. **스프링**에서 제공하는 클래스의 빈을 사용하려면 여전히 XML로 설정해야 합니다.

코드 26-21 pro26/WebContent/WEB-INF/config/action-mybatis.xml

```
...
<bean id="dataSource" class="org.apache.ibatis.datasource.pooled.PooledDataSource">
  <property name="driver" value="${jdbc.driverClassName}" />
  <property name="url" value="${jdbc.url}" />
  <property name="username" value="${jdbc.username}" />
  <property name="password" value="${jdbc.password}" />
</bean>

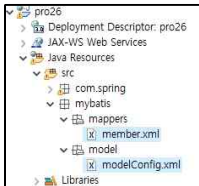
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="configLocation" value="classpath:mybatis/models/modelConfig.xml" />
  <property name="mapperLocations" value="classpath:mybatis/mappers/*.xml" />
</bean>
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg index="0" ref="sqlSessionFactory"></constructor-arg>
</bean>
<!--
<bean id="memberDAO" class="com.spring.member.dao.MemberDAOImpl">
  <property name="sqlSession" ref="sqlSession"></property>
</bean>
-->
```

MemberDAO는 개발자가 만든 클래스이므로 XML에서
설정하지 않고 자바 코드에서 애너테이션으로 설정합니다.



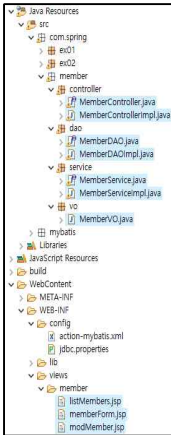
26.4 @Autowired 이용해 빈 주입하기

5. 이번에는 회원 관리 기능 매퍼 파일을 설정해 보겠습니다. 매퍼 파일들은 24장에서 만든 파일과 같은 패키지 구조로 만든 후 실습한 member.xml과 modelConfig.xml을 복사해 붙여넣습니다.



26.4 @Autowired 이용해 빈 주입하기

6. 회원 관리 기능 관련 자바 파일과 JSP를 다음과 같이 준비합니다. 일단 JSP는 24장에서 실습할 때 사용한 파일과 동일하므로 views 폴더 하위에 member 폴더를 만든 후 파일을 복사해 붙여 넣습니다.



26.4 @Autowired 이용해 빈 주입하기

7. MemberControllerImpl 클래스를 작성합니다.

코드 26-22 pro26/src/com/spring/member/controller/MemberControllerImpl.java

```
package com.spring.member.controller;

...
@Controller("memberController")
public class MemberControllerImpl implements MemberController {

    @Autowired
    private MemberService memberService;

    @Autowired
    MemberVO memberVO;

    @Override
    @RequestMapping(value="/member/listMembers.do", method = RequestMethod.GET)
    public ModelAndView listMembers(HttpServletRequest request,
                                   HttpServletResponse response) throws Exception {
        String viewName = getViewName(request);
        List membersList = memberService.listMembers();
        ModelAndView mav = new ModelAndView(viewName);
        mav.addObject("membersList", membersList);
        return mav;
    }
}
```

@Controller를 이용해 MemberControllerImpl 클래스에 대해 id가 memberController인 빈을 자동 생성합니다.

@Autowired를 이용해 id가 memberService인 빈을 자동 주입합니다.

@Autowired를 이용해 id가 memberVO인 빈을 자동 주입합니다.

두 단계로 요청 시 바로 해당 메서드를 호출하도록 매핑합니다.

26.4 @Autowired 이용해 빈 주입하기

@Override

@RequestMapping(value="/member/addMember.do" ,method = RequestMethod.POST)

```
public ModelAndView addMember(@ModelAttribute("member") MemberVO member,  
    HttpServletRequest request, HttpServletResponse response) throws Exception {  
    request.setCharacterEncoding("utf-8");  
    int result = 0;  
    result = memberService.addMember(member);  
    ModelAndView mav = new ModelAndView("redirect:/member/listMembers.do");  
    return mav;  
}
```

회원 가입창에서 전송된 회원 정보를 바로
MemberVO 객체에 설정합니다.

설정된 memberVO 객체를 SQL문으로 전달해
회원 등록을 합니다.

26.4 @Autowired 이용해 빈 주입하기

@Override

@RequestMapping(value="/member/removeMember.do", method = RequestMethod.GET)

public ModelAndView removeMember(@RequestParam("id") String id,

HttpServletRequest request,  전송된 ID를 변수 id에 설정합니다.

HttpServletRequest response) throws Exception{

request.setCharacterEncoding("utf-8");

memberService.removeMember(id);

ModelAndView mav = new ModelAndView("redirect:/member/listMembers.do");

return mav;

}

정규식을 이용해 요청명이 Form.do로 끝나면
form() 메서드를 호출합니다.

@RequestMapping(value = "/member/*Form.do", method = RequestMethod.GET)

public ModelAndView form(HttpServletRequest request,

HttpServletRequest response) throws Exception {

String viewName = getViewName(request);

ModelAndView mav = new ModelAndView();

mav.setViewName(viewName);

return mav;

}

...

26.4 @Autowired 이용해 빈 주입하기

8. MemberServiceImpl 클래스를 다음과 같이 작성합니다.

코드 26-23 pro26/src/com/spring/member/service/MemberServiceImpl.java

```
package com.spring.member.service;

...
@Service("memberService")
@Transactional(propagation = Propagation.REQUIRED)
public class MemberServiceImpl implements MemberService {
    @Autowired
    private MemberDAO memberDAO;

    @Override
    public List listMembers() throws DataAccessException {
        List membersList = null;
        membersList = memberDAO.selectAllMemberList();
        return membersList;
    }

    @Override
    public int addMember(MemberVO member) throws DataAccessException {
        return memberDAO.insertMember(member);
    }

    @Override
    public int removeMember(String id) throws DataAccessException {
        return memberDAO.deleteMember(id);
    }
}
```

MemberServiceImpl 클래스를 이용해 id가 memberService인 빈을 자동 생성합니다.

id가 memberDAO인 빈을 자동 주입합니다.

26.4 @Autowired 이용해 빈 주입하기

9. 계속해서 MemberDAOImpl 클래스를 다음과 같이 작성합니다.

코드 26-24 pro26/src/com/spring/member/dao/MemberDAOImpl.java

```
package com.spring.member.dao;
...
@Repository("memberDAO")
public class MemberDAOImpl implements MemberDAO {
    @Autowired
    private SqlSession sqlSession;

    @Override
    public List selectAllMemberList() throws DataAccessException {
        List<MemberVO> membersList = null;
        membersList = sqlSession.selectList("mapper.member.selectAllMemberList");
        return membersList;
    }

    @Override
    public int insertMember(MemberVO memberVO) throws DataAccessException {
        int result = sqlSession.insert("mapper.member.insertMember", memberVO);
        return result;
    }

    @Override
    public int deleteMember(String id) throws DataAccessException {
        int result = sqlSession.delete("mapper.member.deleteMember", id);
        return result;
    }
}
```

id가 memberDAO인 빈을 자동 주입합니다.

XML 설정 파일에서 생성한 id가 sqlSession인 빈을 자동 주입합니다.

26.4 @Autowired 이용해 빈 주입하기

10. MemberVO 클래스의 경우 @Component("memberVO")를 이용해 id가 memberVO인 빈을 자동 생성하도록 설정합니다.

코드 26-25 pro26/src/com/spring/member/vo/MemberVO.java

```
package com.spring.member.vo;
```

```
import org.springframework.stereotype.Component;
```

```
@Component("memberVO")
```

```
public class MemberVO {
```

```
    private String id;
```

```
    private String pwd;
```

```
    private String name;
```

```
    private String email;
```

```
    private Date joinDate;
```

```
    public MemberVO() {
```

```
    }
```

```
    public MemberVO(String id, String pwd, String name, String email) {
```

```
        this.id = id;
```

```
        this.pwd = pwd;
```

```
        this.name = name;
```

```
        this.email = email;
```

```
    }
```

```
    //각 속성에 대한 getter와 setter
```

```
    ...
```

26.4 @Autowired 이용해 빈 주입하기

@Autowired 와 스테레오 타입 애너테이션으로 주입한 빈 상태

bean	bean	bean	bean
memberController	memberService	memberDAO	sqlSession

11. <http://localhost:8090/pro26/member/listMembers.do>로 요청하면 다음과 같이 회원 목록을 표시합니다.
회원이입을 클릭하여 <http://localhost:8090/pro26/member/memberForm.do>로 요청하면 회원 가입창이 표시됩니다.

아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
park	1234	박찬호	park@test.com	2018-09-04	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입

26.4 @Autowired 이용해 빈 주입하기

12. 새 회원 '류현진'에 대한 정보를 입력한 후 가입하기를 클릭하여
<http://localhost:8090/pro26/member/addMember.do>로 요청하면 회원 등록 결과가 표시됩니다.

회원 가입창

아이디

비밀번호

이름

이메일

아이디	비밀번호	이름	이메일	가입일	삭제
ryu	1234	류현진	ryu@test.com	2018-10-02	삭제하기
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입

26.4 @Autowired 이용해 빈 주입하기

13. 이번에는 '류현진' 회원을 삭제해 보겠습니다. 삭제하기를 클릭하여
<http://localhost:8090/pro26/member/deleteMember.do>로 요청하면 회원이 삭제되어 표시됩니다.

아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04	삭제하기

회원가입