

## 27장 메이븐과 스프링 STS 사용법

---

27.1 메이븐 설치하기

27.2 메이븐 환경 변수 설정하기

27.3 STS 설치하기

27.4 메이븐 프로젝트의 구조 및 구성요소

27.5 스프링 프로젝트 만들기

27.6 STS 프로젝트 실행하기

27.7 STS 환경에서 마이바티스 사용하기

27.8 log4j 란?

27.9 타일즈 알아보기

27.10 JSP 페이지에서 타일즈 사용하기

27.11 JSP 페이지에 회원 목록창 나타내기

27.12 로그인 기능 구현하기

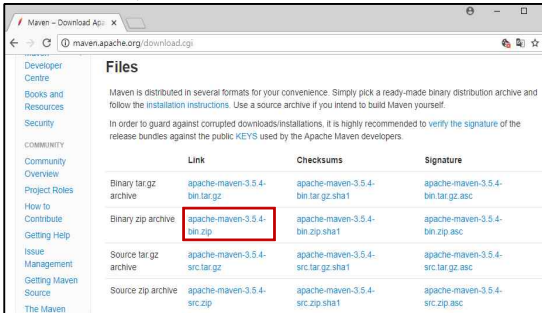
## 27.1 메이븐 설치하기

1. maven.apache.org에 접속한 후 Download를 클릭합니다.



## 27.1 메이븐 설치하기

### 2. apache-maven-3.5.4-bin.zip 파일을 다운로드합니다

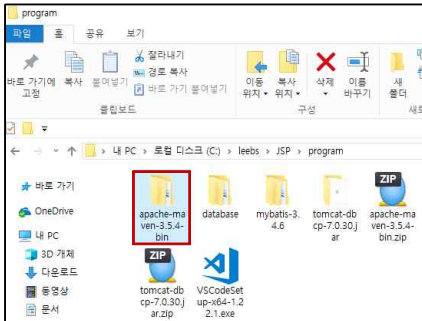


The screenshot shows the Maven download page. The browser address bar indicates the URL is `maven.apache.org/download.cgi`. The page title is "Files". The main content area contains instructions on how to download Maven and a table of available download links. The table has four columns: "Link", "Checksums", and "Signature". The "Binary zip archive" row is highlighted with a red box around the link "apache-maven-3.5.4-bin.zip".

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.5.4-bin.tar.gz</a>	<a href="#">apache-maven-3.5.4-bin.tar.gz.sha1</a>	<a href="#">apache-maven-3.5.4-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.5.4-bin.zip</a>	<a href="#">apache-maven-3.5.4-bin.zip.sha1</a>	<a href="#">apache-maven-3.5.4-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.5.4-src.tar.gz</a>	<a href="#">apache-maven-3.5.4-src.tar.gz.sha1</a>	<a href="#">apache-maven-3.5.4-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.5.4-src.zip</a>	<a href="#">apache-maven-3.5.4-src.zip.sha1</a>	<a href="#">apache-maven-3.5.4-src.zip.asc</a>

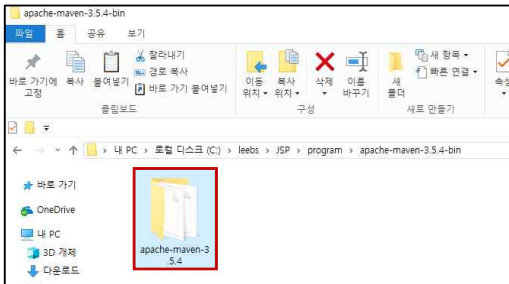
## 27.1 메이븐 설치하기

3. 원하는 폴더에 apache-maven-3.5.4-bin.zip 파일의 압축을 해제합니다.



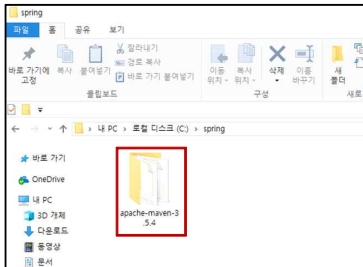
## 27.1 메이븐 설치하기

4. apache-maven-3.5.4-bin 폴더 안에 있는 apache-maven-3.5.4 폴더를 복사합니다.



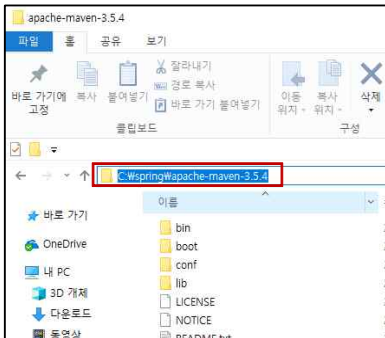
## 27.1 메이븐 설치하기

5. 복사한 폴더를 C:\spring 폴더에 붙여 넣습니다.



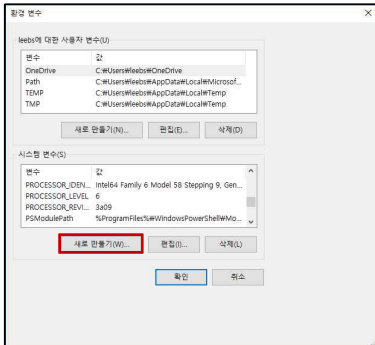
## 27.2 메이븐 환경 변수 설정하기

1. 윈도우 탐색기에서 메이븐의 홈 디렉터리 경로를 복사합니다.



## 27.2 메이븐 환경 변수 설정하기

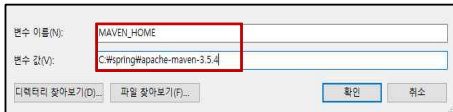
2. 환경 변수 설정창에서 시스템 변수의 새로 만들기를 클릭합니다.





## 27.2 메이븐 환경 변수 설정하기

3. 변수 이름은 MAVEN\_HOME으로 설정하고, 변수 값에는 1번 과정에서 복사한 메이븐 홈 디렉터리 경로를 붙여 넣은 후 확인을 클릭합니다.



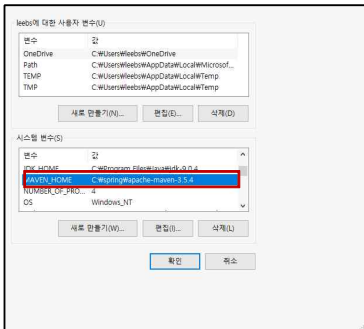
변수 이름(N): MAVEN\_HOME

변수 값(V): C:\spring\apache-maven-3.5.4

디렉터리 찾아보기(D)... 파일 찾아보기(F)... 확인 취소

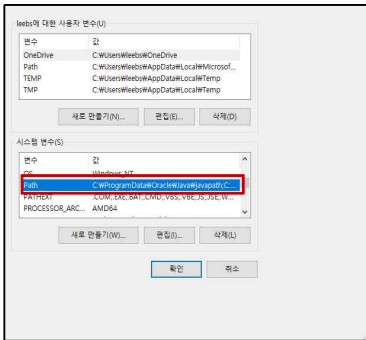
## 27.2 메이븐 환경 변수 설정하기

### 4. MAVEN\_HOME 환경 변수가 등록된 것을 확인할 수 있습니다.



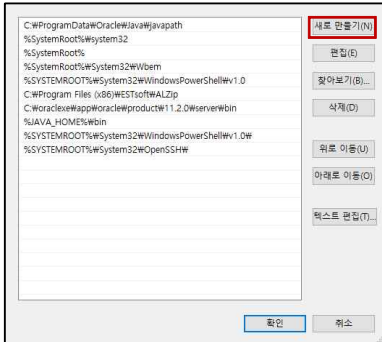
## 27.2 메이븐 환경 변수 설정하기

5. 이번에는 시스템 변수의 Path를 선택합니다.



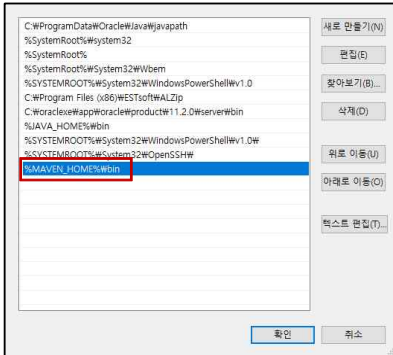
## 27.2 메이븐 환경 변수 설정하기

6. 새로 만들기를 클릭합니다.



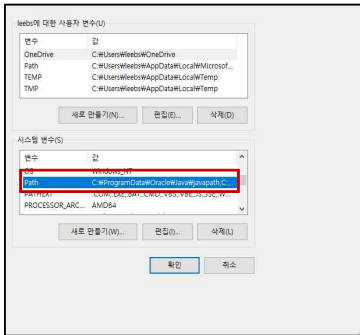
## 27.2 메이븐 환경 변수 설정하기

7. MAVEN\_HOME 환경 변수를 이용해 bin 디렉터리 경로를 설정하고 확인을 클릭합니다.



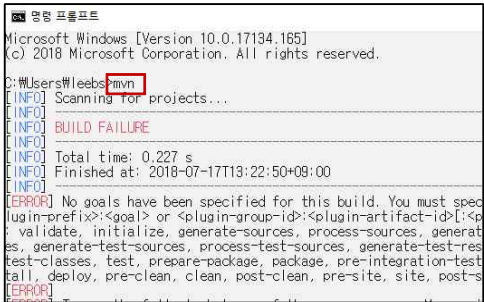
## 27.2 메이븐 환경 변수 설정하기

8. 다시 환경 변수창에서 확인을 클릭합니다.



## 27.2 메이븐 환경 변수 설정하기

9. 정상적으로 설치되었는지 확인하기 위해 명령 프롬프트에서 mvn을 입력하고 엔터키를 누릅니다. 다음과 같은 메이븐 관련 메시지가 표시되면 제대로 설치된 것입니다.



```
명령 프롬프트
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\leebs>mvn
[INFO] Scanning for projects...
[INFO] BUILD FAILURE
[INFO] Total time: 0.227 s
[INFO] Finished at: 2018-07-17T13:22:50+09:00
[ERROR] No goals have been specified for this build. You must specify at least one of the following goals: validate, initialize, generate-sources, process-sources, generate-resources, generate-test-sources, process-test-sources, generate-test-resources, test-classes, test, prepare-package, package, pre-integration-test, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, test, install, deploy.
```

## 27.3 STS 설치하기

### STS(Spring Tool Suite)

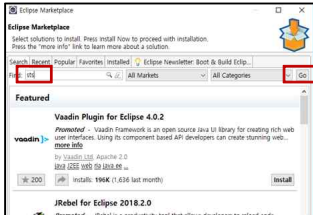
- 이클립스를 기반으로 만들어진 스프링 기반 애플리케이션 개발용 도구

### STS 설치 방법

- 이클립스에 STS 플러그인을 설치하는 방법
- 스프링 홈페이지인 <http://spring.io>에서 직접 다운로드해서 설치하는 방법

### 이클립스에 STS 플러그인을 설치하기

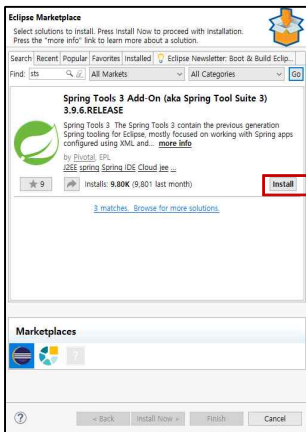
- 이클립스 상단 메뉴에서 Help > Eclipse Marketplace...를 선택하고 검색창에 sts를 입력한 후 Go를 클릭합니다.





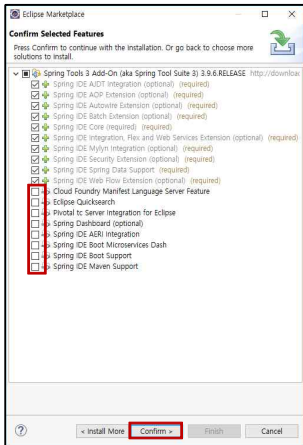
## 27.3 STS 설치하기

2. Spring Tool Suite 3 Add-On 3.9.6 RELEASE 항목의 Install을 클릭합니다.



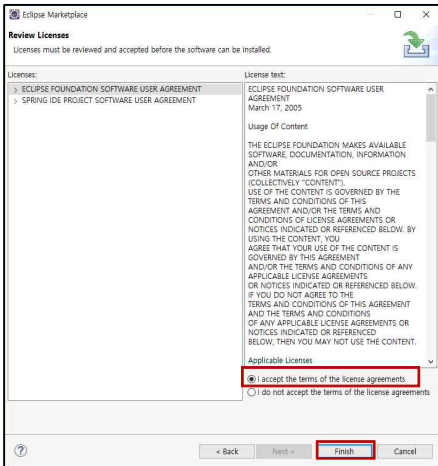
## 27.3 STS 설치하기

3. (required)라고 표시된 항목 외의 항목들은 모두 체크를 해제한 후 Confirm을 클릭합니다.



## 27.3 STS 설치하기

4. 사용 저작권에 동의한다고 체크한 후 Finish를 클릭합니다.



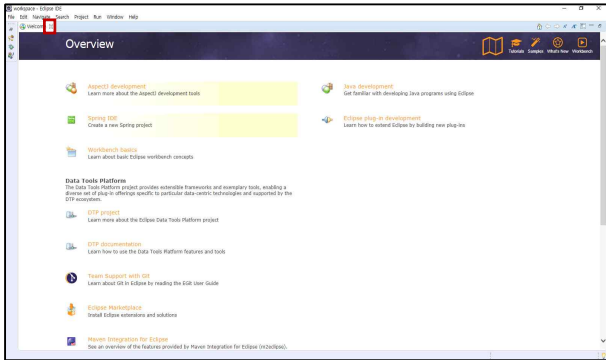
## 27.3 STS 설치하기

5. 설치 완료 후 이클립스를 재실행할 것인지 묻는 메시지가 나오면 Restart Now를 클릭합니다.



## 27.3 STS 설치하기

6. 이클립스 재실행 후 Welcome 페이지가 나타나면 정상적으로 설치된 것이므로 Welcome 페이지를 닫습니다.



## 27.4 메이븐 프로젝트의 구조와 구성 요소

### 메이븐(Maven)

- 프로젝트 구조와 내용을 기술하는 선언적 접근 방식의 오픈 소스 빌드 툴
- 프로젝트 종속 라이브러리들과 그 라이브러리에 의존하는 Dependency 자원까지 관리할 수 있음  
프로젝트 전반의 리소스 관리와 설정 파일 그리고 이와 관련된 표준디렉터리 구조를 처음부터 일관된 형태로 구성하여 관리할 수 있음

### ❖ Note

일반적인 애플리케이션은 단지 코드를 컴파일했다고 해서 동작하는 것이 아닙니다.

우리가 사용한 오픈소스 라이브러리들은 컴파일할 때 합쳐져 하나의 기능을 이룹니다. 그리고 컴파일 과정 외에 테스트, 배포 같은 과정도 거쳐야 합니다.

즉, 애플리케이션을 만들 때는 컴파일보다 더 많은 과정을 거치게 됩니다. 이런 과정을 빌드라고하고 이런 작업을 자동으로 수행해 주는 툴을 빌드 툴이라고 합니다. 이런 빌드 툴에는 **Ant, Apache Ivy, Maven, Gradle** 등이 있습니다.

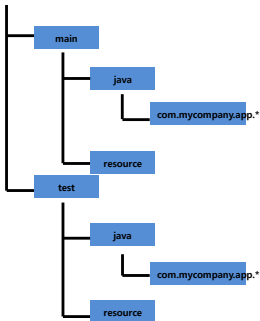
## 27.4 메이븐 프로젝트의 구조와 구성 요소

### 메이븐 기반 웹 프로젝트 기본 디렉터리 구조

프로젝트 이름

pom.xml

src



## 27.4 메이븐 프로젝트의 구조와 구성 요소

### 메이븐 프로젝트 구성 요소들

구성 요소	설명
pom.xml	프로젝트 정보가 표시되며 스프링에서 사용되는 여러 가지 라이브러리를 설정해 다운로드할 수 있습니다.
src/main/java	자바 소스 파일이 위치합니다
src/main/resources	프로퍼티 파일이나 XML 파일 등 리소스 파일이 위치합니다.
src/main/webapp	WEB_INF 등 웹 애플리케이션 리소스가 위치합니다.
src/test/java	JUnit 등 테스트 파일이 위치합니다.
src/test/resources	테스트 시에 필요한 resource 파일이 위치합니다.



## 27.4 메이븐 프로젝트의 구조와 구성 요소

### pom.xml

코드 27-1 pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.spring</groupId>
  <artifactId>mytest2</artifactId>
  <name>spring_test2</name>
  <packaging>war</packaging>
  <version>1.0.0-BUILD-SNAPSHOT</version>
  <properties>
    <java-version>1.6</java-version>
    <org.springframework-version>4.0.0.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${org.springframework-version}</version>
    </dependency>
    <dependency>
```

## 27.4 메이븐 프로젝트의 구조와 구성 요소

### pom.xml의 프로젝트 정보 설정 태그 구성 요소

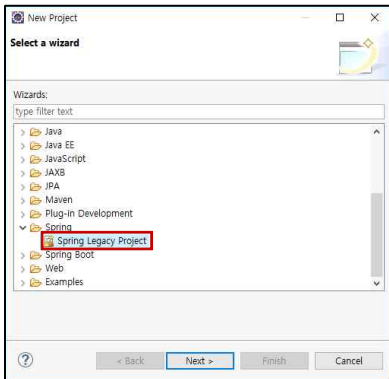
속성	설명
groupId	프로젝트 그룹 id를 나타내며 일반적으로 도메인 이름을 사용해 설정합니다.
artifactId	프로젝트 아티팩트 id를 설정합니다. 대개는 패키지 이름으로 설정합니다.
version	프로젝트의 버전을 설정합니다.
packaging	애플리케이션 배포 시 패키징 타입을 설정합니다.이 경우는 war 파일로 패키징됩니다.

### pom.xml의 dependencies 정보 설정 태그 구성 요소

속성	설명
dependency	해당 프로젝트에서 의존하는 다른 라이브러리 정보를 기술합니다.
groupId	의존하는 프로젝트의 그룹 id
artifactId	의존하는 프로젝트의 artifact id
version	의존하는 프로젝트 버전 정보

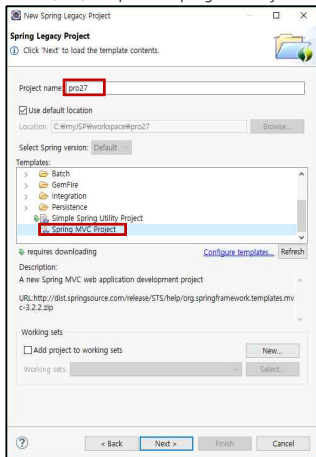
## 27.5 스프링 프로젝트 만들기

1. 메뉴에서 New > project 항목을 선택하고 Spring 항목의 Spring Legacy Project를 선택한 후 Next를 클릭합니다.



## 27.5 스프링 프로젝트 만들기

2. 프로젝트 이름으로 pro27을 입력한 후 Templates를 Spring MVC Project로 선택합니다.



## 27.5 스프링 프로젝트 만들기

3. 다운로드 메시지가 나타나면 Yes를 클릭합니다.



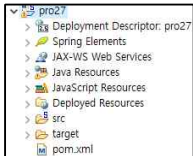
## 27.5 스프링 프로젝트 만들기

4. 패키지 이름으로 `com.myspring.pro27`(세 번째 단계의 패키지 이름, 즉 `pro27`이 브라우저에서 요청하는 컨텍스트 이름입니다)을 입력하고 Finish를 클릭합니다.



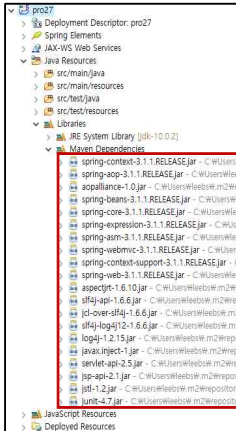
## 27.5 스프링 프로젝트 만들기

5. 이클립스에서 프로젝트가 생성된 것을 확인할 수 있습니다.



## 27.5 스프링 프로젝트 만들기

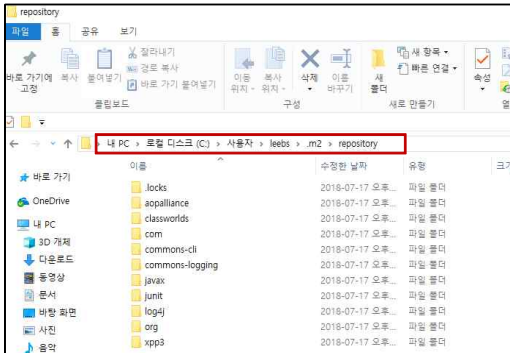
6. 프로젝트의 Maven Dependencies 폴더를 클릭하면 자동으로 다운로드된 스프링 관련 라이브러리들이 보입니다.





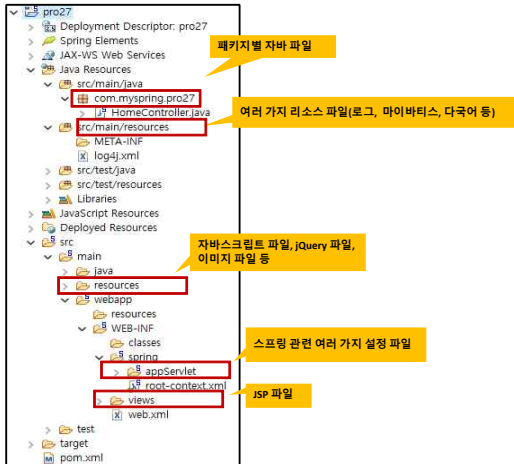
## 27.5 스프링 프로젝트 만들기

7. 라이브러리 옆에 표시된 경로를 통해 설치된 라이브러리 파일들을 볼 수 있습니다.



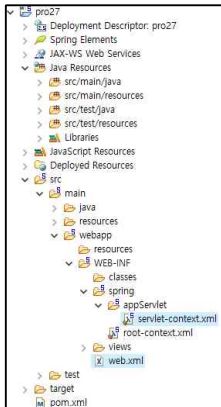
## 27.6 STS 프로젝트 실행하기

### STS 프로젝트 구조



## 27.6 STS 프로젝트 실행하기

- 27.6.1 XML 파일 설정하기



## 27.6 STS 프로젝트 실행하기

### web.xml

코드 27-2 pro27/src/main/webapp/WEB-INF/web.xml

```
...

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

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

<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet
</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```

스프링 실행 시 ~~root-context.xml~~의 설정 정보를 읽어 들입니다.  
**servlet-context.xml**

## 27.6 STS 프로젝트 실행하기

### servlet-context.xml

코드 27-3 pro27/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml

```

...
<!-- DispatcherServlet Context:
defines this servlet's request-processing infrastructure -->
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />

<!-- Handles HTTP GET requests for /resources/**
by efficiently serving up static resources in the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />
<!-- Resolves views selected for rendering by
@Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="com.spring.pro27" />
/beans:beans>

```

JSR에서 사용될 자바스크립트나 이미지 파일 경로를 지정합니다.

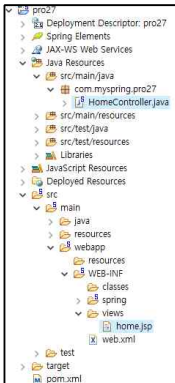
뷰리졸버 빈을 생성하면서 응답할 JSP의 경로를 지정합니다.

패키지와 애너테이션을 지정합니다.

## 27.6 STS 프로젝트 실행하기

### • 27.6.2 자바 클래스와 JSP 파일 만들기

1. 프로젝트를 만들면 다음과 같이 자동으로 자바 클래스와 JSP 파일이 생성됩니다. 일일이 추가했던 이전 실습과 비교하면 참 편리하죠?



## 27.6 STS 프로젝트 실행하기

2. HomeController 클래스를 다음과 같이 작성합니다. 모든 요청에 대해 home() 메서드를 호출하여 요청 시각을 home.jsp로 포워딩합니다.

코드 27-4 pro27/src/main/java/com/myspring/pro27/HomeController.java

```
package com.myspring.pro27;

...
/**
 * Handles requests for the application home page.
 */
@Controller ① ② @Controller를 적용합니다.
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    /**
     * Simply selects the home view to render by returning its name.
     */
    @RequestMapping(value = "/", method = RequestMethod.GET) ③ ④ 모든 요청에 대해 home() 메서드를 호출합니다
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
            DateFormat.LONG, locale);
        String formattedDate = dateFormat.format(date);
        model.addAttribute("serverTime", formattedDate ); ⑤ 브라우저에서 요청한 시각을 JSP로 전달합니다.
        return "home";
    }
}
```

⑥ 뷰리졸버로 JSP 이름을 반환합니다.

## 27.6 STS 프로젝트 실행하기

3. home.jsp는 컨트롤러에서 전달된 요청 시각을 출력하도록 작성합니다.

**코드 27-5** pro27/src/main/webapp/WEB-INF/views/home.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
  <title>Home</title>
</head>
<body>
<h1>
  Hello world!
</h1>

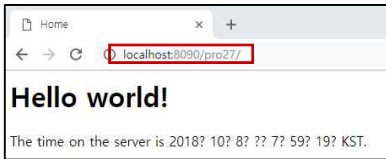
<P> The time on the server is ${serverTime}. </P>
</body>
</html>
```

브라우저에서 요청한 시각을 브라우저에 출력합니다.



## 27.6 STS 프로젝트 실행하기

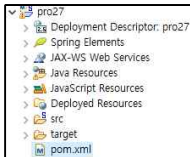
4. 톰캣에 pro27을 등록해서 실행한 후 브라우저에서 `http://localhost:8090/pro27`로 요청하면 요청 시각이 표시됩니다.



## 27.7 STS 환경에서 마이바티스 실습하기

- 27. pom.xml 이용해 마이바티스 라이브러리 설치하기

1. 다음과 같이 pom.xml을 준비합니다.



## 27.7 STS 환경에서 마이바티스 실습하기

2. MySQL과는 다르게 오라클은 오픈 소스가 아니므로 드라이버를 직접 다운로드하여 설치해야 합니다.  
따라서 다음과 같이 lib 폴더를 생성한 후 오라클 드라이버를 lib 폴더에 복사하여 붙여 넣습니다.



## 27.7 STS 환경에서 마이바티스 실습하기

3. pom.xml을 다음과 같이 작성합니다.

코드 27-6 pro27/pom.xml

```
...  
<dependency>  
  <groupId>commons-beanutils</groupId>  
  <artifactId>commons-beanutils</artifactId>  
  <version>1.8.0</version>  
</dependency>  
<dependency>  
  <groupId>commons-dbcp</groupId>  
  <artifactId>commons-dbcp</artifactId>  
  <version>1.2.2</version>  
</dependency>  
<dependency>  
  <groupId>cglib</groupId>  
  <artifactId>cglib-nodep</artifactId>  
  <version>2.2</version>  
</dependency>
```

데이터소스 관련 라이브러리를 설정합니다.

## 27.7 STS 환경에서 마이바티스 실습하기

```

<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.1.0</version>
</dependency>
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.1.0</version>
</dependency>

```

마이바티스 관련 라이브러리를 설정합니다.

로컬에 설치한 오라클 드라이버  
라이브러리를 설정합니다.

```

<dependency>
  <groupId>org.mariadb.jdbc</groupId>
  <artifactId>mariadb-java-client</artifactId>
  <version>3.1.1</version>
</dependency>

```

Path>

```

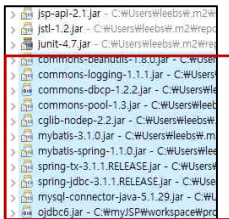
</dependencies>
...

```

<systemPath>로 로컬에 설치한 위치를 지정합니다.  
\${basedir}은 프로젝트 루트 디렉터리입니다.

## 27.7 STS 환경에서 마이바티스 실습하기

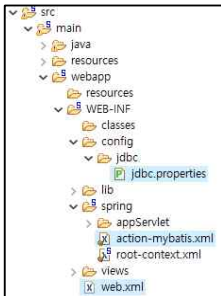
4. pom.xml에 <dependency> 태그로 설정한 후 저장할 때 다운로드한 라이브러리들을 이클립스에서 확인할 수 있습니다.



## 27.7 STS 환경에서 마이바티스 실습하기

### • 27.7.2 마이바티스 관련 XML 파일 추가하기

1. 먼저 jdbc.properties 파일은 26장의 회원 관리 기능을 구현할 때 사용한 것을 재사용합니다. WEB-INF 하위에 config 폴더를 만들고 다시 jdbc 폴더를 만든 다음 복사하여 붙여 넣습니다. action-mybatis.xml도 26장의 것을 복사하여 WEB-INF/spring 폴더에 붙여 넣습니다.



```
web.xml  pro27/pom.xml  jdbc.properties  ↗
1 jdbc.driverClassName=org.mariadb.jdbc.Driver
2 jdbc.url=jdbc:mariadb://localhost:3306/servletex
3 jdbc.username=root
4 jdbc.password=1234
5
~
```

## 27.7 STS 환경에서 마이바티스 실습하기

2. web.xml에서 action-mybatis.xml을 읽어 들이도록 수정합니다.

코드 27-7 pro27/src/main/webapp/WEB-INF/web.xml

```
...
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/action-mybatis.xml</param-value>
</context-param>
<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</listener>
...
```

web.xml에서 root-context.xml 대신  
action-mybatis.xml로 수정합니다.



## 27.7 STS 환경에서 마이바티스 실습하기

3. action-mybatis.xml을 열어 jdbc.properties 경로를 수정합니다.

**코드 27-8** pro27/src/main/webapp/WEB-INF/spring/action-mybatis.xml

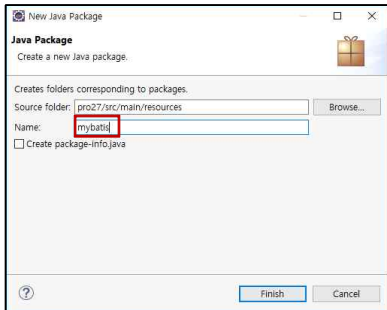
```
...
<bean id="propertyPlaceholderConfigurer"
      class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <value>/WEB-INF/config/jdbc/jdbc.properties</value>
    </property>
</bean>

<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>
```

jdbc.properties를 읽어옵니다.

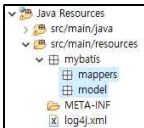
## 27.7 STS 환경에서 마이바티스 실습하기

4. 이번에는 매퍼 파일을 추가해 보겠습니다. 메이븐 프로젝트에서는 `src/main/resources` 패키지 하위에 `mybatis` 패키지를 생성합니다.

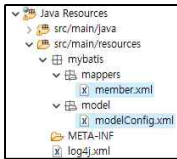


## 27.7 STS 환경에서 마이바티스 실습하기

5. 다시 mybatis 하위에 mappers 패키지와 model 패키지를 생성합니다.



6. 26장에서 사용한 member.xml과 modelConfig.xml을 각각 mappers와 model 패키지에 붙여 넣습니다.



## 27.7 STS 환경에서 마이바티스 실습하기

### 7. modelConfig.xml을 열어 패키지 이름을 수정합니다

**코드 27-9** pro27/src/main/resources/mybatis/model/modelConfig.xml

...

```
<configuration>
```

```
<typeAliases>
```

```
<typeAlias type="com.myspring.pro27.member.vo.MemberVO" alias="memberVO" />
```

```
</typeAliases>
```

```
</configuration>
```

---

MemberVO에 대한 alias를 설정합니다.

## 27.7 STS 환경에서 마이바티스 실습하기

### 27.7.3 자바 클래스와 JSP 구현하기

1. 자바 클래스 파일과 JSP 파일은 26장의 것을 재사용하면 됩니다.

따라서 자바 파일은 src/main/java 하위에 있는 com.myspring.pro27 패키지에 member 패키지를 만든 후 각각의 하위 패키지를 생성해 붙여 넣습니다(26장 참조).

JSP는 WEB-INF/views 폴더 하위에 member 폴더를 만든 후 26장의 파일을 붙여 넣습니다.



## 27.7 STS 환경에서 마이바티스 실습하기

2. 브라우저의 URL 요청명에서 뷰리졸버 설정 없이 기능별로 해당 폴더에 쉽게 접근할 수 있도록 MemberControllerImpl 클래스를 열어 getViewName() 메서드를 수정합니다.

**코드 27-10** pro27/src/main/java/com/myspring/pro27/member/controller/MemberControllerImpl.java

```
@Controller
package com.myspring.web25.member.controller;
...
@Controller("memberController")
public class MemberControllerImpl implements MemberController {
    @Autowired
    private MemberService memberService;
    @Autowired
    MemberVO memberVO ;
}
```

## 27.7 STS 환경에서 마이바티스 실습하기

```

@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;
}

...
private String getViewName(HttpServletRequest request) throws Exception {
    ...
    if(viewName.lastIndexOf("/") != -1) {
        viewName = viewName.substring(viewName.lastIndexOf("/",1), viewName.length());
    }
    return viewName;
}
}

```


 /member/listMembers.do로 요청할 경우 member/
 listMember를 파일 이름으로 가져옵니다.

---

## 27.7 STS 환경에서 마이바티스 실습하기

3. <http://localhost:8090/pro27/member/listMember.do>로 요청하면 다음과 같이 회원 목록을 표시합니다.

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

### 회원가입



## 27.8 log4j란?

❖ println() 메서드 사용 시 유지 보수 시 불편함

```

34=  @Override
35  @RequestMapping(value="/listMembers.do", method = RequestMethod.GET)
36  public ModelAndView listMembers(HttpServletRequest request) {
37      String viewName = getViewName(request);
38      System.out.println("viewName = " + viewName);
39
40      List membersList = memberService.listMembers();
41      ModelAndView mav = new ModelAndView(viewName);
42      mav.addObject("membersList", membersList);
43      return mav;
44  }

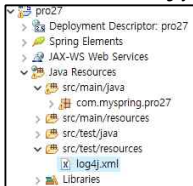
```

## log4j

- 로그 기능을 제공하는 오픈 소스 라이브러리
- 애플리케이션에서 웹 사이트에 접속한 사용자 정보나 각 클래스의 메서드 호출 시각 등 여러 가지 정보를 로그로 출력해서 관리
- 메이븐에선 프로젝트 생성 시 자동으로 log4j 라이브러리가 설치됨

## 27.8 log4j란?

STS 프로젝트 생성 시 자동으로 생성되는 log4j.xml



log4j.xml을 이루는 태그

태그	설명
<Appender>	로그의 출력 위치를 결정(파일, 콘솔, DB 등)합니다. log4j API 문서의 XXXAppender로 끝나는 클래스들의 이름을 보면 출력 위치를 알 수 있습니다.
<Layout>	Appender가 어디에 출력할 것인지 결정했다면 어떤 형식으로 출력할지 출력 레이아웃을 결정합니다.
<Logger>	로깅 메시지를 Appender에 전달합니다. 개발자가 로그 레벨을 이용해 로그 출력 여부를조정할 수 있습니다. logger는 로그 레벨을 가지고 있으며, 로그의 출력 여부는 로그문의 레벨과 로거의 레벨을 가지고 결정합니다.

## 27.8 log4j란?

### 여러 가지 Appender 클래스

Appender 클래스	설명
ConsoleAppender	org.apache.log4j.ConsoleAppender 클래스로, 콘솔에 로그 메시지를 출력합니다.
FileAppender	org.apache.log4j.FileAppender 클래스로, 파일에 로그 메시지를 출력합니다.
RollingFileAppender	org.apache.log4j.rolling.RollingFileAppender 클래스로, 파일 크기가 일정 기준을 넘으면 기존 파일을 백업 파일로 바꾸고 처음부터 다시 기록합니다.
DailyRollingAppender	org.apache.log4j.DailyRollingFileAppender 클래스로, 설정한 기간 단위로 새 파일을 만들어 로그 메시지를 기록합니다.

### PatternLayout 클래스에서 사용되는 여러 가지 출력 속성들

속성	설명
%p	debug, info, error, fatal 등 로그 레벨 이름 출력
%m	로그 메시지 출력
%d	로깅 이벤트 발생 시각 출력
%F	로깅이 발생한 프로그램 파일 이름 출력
%l	로깅이 발생한 caller의 정보 출력
%L	로깅이 발생한 caller의 라인 수 출력
%M	로깅이 발생한 method 이름 출력
%c	로깅 메시지 앞에 전체 패키지 이름이나 전체 파일 이름 출력
...	...

## 27.8 log4j란?

### log4j의 여러 가지 로그 레벨들

속성	설명
FATAL	시스템 차원에서 심각한 문제가 발생해 애플리케이션 작동이 불가능할 경우에 해당하는 레벨입니다. 일반적으로 애플리케이션에서는 사용할 일이 없습니다.
ERROR	실행 중 문제가 발생한 상태를 나타냅니다.
WARN	향후 시스템 오류의 원인이 될 수 있는 경고 메시지를 나타냅니다.
INFO	로그인, 상태 변경과 같은 실제 애플리케이션 운영과 관련된 정보 메시지를 나타냅니다.
DEBUG	개발 시 디버깅 용도로 사용한 메시지를 나타냅니다.
TRACE	DEBUG 레벨보다 상세한 로깅 정보를 출력하기 위해 도입된 레벨입니다.

### ❖ log4j 관련 사이트

- <http://logging.apache.org/log4j/1.2/manual.html>

## 27.8 log4j란?

### • 28.8.1 log4j.xml 이용해 로그 메시지 출력하기

1. 다음과 같이 log4j.xml을 작성합니다.

코드 27-11 pro27/src/test/resources/log4j.xml

```
...
<!-- Console Appenders -->
<appender name="console" class="org.apache.log4j.ConsoleAppender">
  <param name="Target" value="System.out" /> ConsoleAppender를 이용해서 로그 메시지를
  <layout class="org.apache.log4j.PatternLayout"> 콘솔로 출력합니다.
    <param name="ConversionPattern" value="%-5p: %c - %m%n" />
  </layout>
</appender> PatternLayout의 출력 형식을 지정합니다.

<!-- DailyRollingFile Appenders --> DailyRollingAppender를 이용해서 로그 메시지를 파일로 출력합니다.
<appender name="dailyFileAppender" class="org.apache.log4j.DailyRollingFileAppender">
  <param name="File" value="C:\\spring\\logs\\output.log" /> 로그 파일 생성 위치를
  <param name="Append" value="true" /> 설정합니다.
  <layout class="org.apache.log4j.PatternLayout">
    <param name="DatePattern" value="'yyyy-MM-dd'"/>
    <param name="ConversionPattern" value="[%d{HH:mm:ss}][%-5p](%F:%L) - %m%n" />
  </layout>
</appender> PatternLayout의 출력 형식을 지정합니다.
...
```

## 27.8 log4j란?

```
<!-- Application Loggers -->
```

```
<logger name="com.myspring.pro27">  
  <level value="info" />  
</logger>
```

— `<logger>` 태그로 `com.myspring.pro27` 패키지에 존재하는 클래스들의 로그 레벨을 `info`로 설정합니다.

```
...
```

```
<!-- Root Logger -->
```

```
<root>
```

```
  <priority value="info" />
```

```
  <appender-ref ref="console" />
```

— 애플리케이션 전체 로그를 콘솔로 출력합니다.

```
  <appender-ref ref="dailyFileAppender" />
```

— 애플리케이션 전체 로그를 파일로 출력합니다.

```
</root>
```

---

## 27.8 log4j란?

2. MemberControllerImpl 클래스를 다음과 같이 작성합니다.

코드 27-12 pro27/src/main/java/com/myspring/pro27/member/controller/MemberControllerImpl.java

```
...
@Controller("memberController")
@RequestMapping("/member")
public class MemberControllerImpl implements MemberController {
    private static final Logger logger =
        LoggerFactory.getLogger(MemberControllerImpl.class);
    ...
    @Override
    @RequestMapping(value="/listMembers.do", method = RequestMethod.GET)
    public ModelAndView listMembers(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        String viewName = getViewName(request);
        //System.out.println("viewName = "+ viewName);
        logger.info("info 레벨 : viewName = "+ viewName);
        logger.debug("debug 레벨 : viewName = "+ viewName);
        List membersList = memberService.listMembers();
        ModelAndView mav = new ModelAndView(viewName);
        mav.addObject("membersList", membersList);
        return mav;
    }
    ...
}
```

LoggerFactory 클래스를 이용해 Logger 클래스 객체를 가져옵니다.

Logger 클래스의 info() 메서드로 로그 메시지를 레벨을 info로 설정합니다.

Logger 클래스의 debug() 메서드로 로그 메시지를 레벨을 debug로 설정합니다.

## 27.8 log4j란?

log4j.xml에서 로그 레벨을 info로 설정했기 때문에 debug() 메서드로 설정한 메시지는 레벨이 낮아 출력되지 않습니다



```

Markers | Properties | Servers | Data Source Explorer | Snippets | Problems | Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-10.0.2\bin\java.exe (2018. 10. 9. 오후 2:17:51)
INFO : com.myspring.pro27.member.controller.MemberControllerImpl - info레벨 : viewName = /member/listMembers
  
```

log4j.xml의 로그 레벨을 debug로 변경한 후 브라우저에서 요청하면 이번에는 debug 레벨 메시지와 상위의 info 레벨 메시지가 모두 출력됩니다



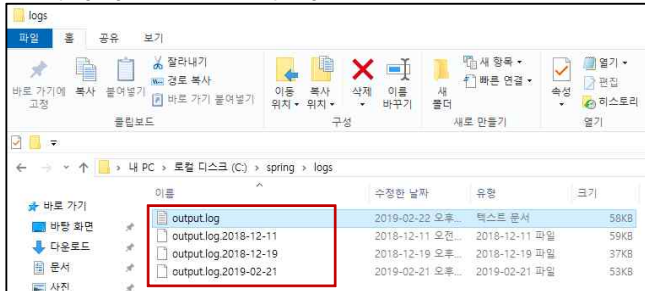
```

Markers | Properties | Servers | Data Source Explorer | Snippets | Problems | Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre-10.0.2\bin\java.exe (2018. 10. 9. 오후 2:31:50)
INFO : com.myspring.pro27.member.controller.MemberControllerImpl - info레벨 : viewName = /member/listMembers
DEBUG: com.myspring.pro27.member.controller.MemberControllerImpl - debug레벨 : viewName = /member/listMembers
  
```



## 27.8 log4j란?

❖ C:\WSspring\Wlogs 폴더에는 날짜별로 output.log 파일이 생성됨



## 27.8 log4j란?

- 27.8.2 마이바티스 SQL문을 로그로 출력하기

코드 27-13 pro27/src/main/resources/log4j.xml

```

...
<!-- Root Logger -->
<root>
  <priority value='debug' />
  <appender-ref ref="console" />
</root>
</log4j:configuration>

```

전체 애플리케이션 로그 레벨을 설정합니다.  
value 속성 값을 debug로 설정합니다.

실행하는 마이바티스 ID와 SQL문을 출력합니다.

```

DEBUG: org.mybatis.spring.transaction.SpringManagedTransaction - JDBC Connection [oracle.jdbc.driver.T4CConnection@18d22338]
DEBUG: mapper.member.selectAllMemberList - ooo Using Connection [oracle.jdbc.driver.T4CConnection@18d22338]
DEBUG: mapper.member.selectAllMemberList - ==> Preparing. select * from t_member order by joindate desc
DEBUG: mapper.member.selectAllMemberList - ==> Parameters:
DEBUG: mapper.member.selectAllMemberList - <= Columns: ID, PWD, NAME, EMAIL, JOINDATE
DEBUG: mapper.member.selectAllMemberList - <= Row: m2, 1234, 이길동, m2@test.com, 2018-09-30 16:04:39.0
DEBUG: mapper.member.selectAllMemberList - <= Row: m1, 1234, 박길동, m1@test.com, 2018-09-30 16:04:39.0
DEBUG: mapper.member.selectAllMemberList - <= Row: m3, 1234, 김길동, m3@test.com, 2018-09-30 16:04:39.0
DEBUG: mapper.member.selectAllMemberList - <= Row: ki, 1234, 기성용, ki@test.com, 2018-09-13 14:45:48.0
DEBUG: mapper.member.selectAllMemberList - <= Row: kim, 1212, 김유신, kim@jweb.com, 2018-09-04 21:35:51.0
DEBUG: mapper.member.selectAllMemberList - <= Row: lee, 1212, 이순신, lee@test.com, 2018-09-04 21:35:48.0
DEBUG: mapper.member.selectAllMemberList - <= Row: hong, 1212, 홍길동, hong@gmail.com, 2018-09-04 21:35:46.0
DEBUG: org.mybatis.spring.SqlSessionUtils - Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@18d22338]
DEBUG: org.springframework.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource

```

SQL문 실행 후 반환되는 레코드들을 출력합니다.

## 27.8 log4j란?

### • AOP 용어 설명

#### 지시자 종류

지시자	기능
execution	-가장 정교한 포인트컷을 만들 수 있음 -리턴타입, 패키지, 클래스명, 메서드명, 메서드 매개변수에 포인트컷 지정
within	-타입패턴 내에 해당하는 모든 것들을 포인트컷 지정
bean	-bean이름으로 포인트컷 지정

#### 어드바이스 동작 시점

동작시점	설명
Before	메서드 실행 전에 동작
After	메서드 실행 후에 동작
After-returning	메서드가 정상적으로 실행된 후에 동작
After-throwing	예외가 발생한 후에 동작
Around	메서드 호출 이전, 이후, 예외 발생 등 모든 시점에서 동작

## 27.8 log4j란?

### 여러가지 포인트컷 표현식

Pointcut	JointPoints
execution(public *.*(..))	public 메서드 실행
execution(* set*(..))	이름이 set으로 시작하는 모든 메서드명 실행
execution(* get*(..))	이름이 get으로 시작하는 모든 메서드명 실행
execution(* com.xyz.service.AccountService.*(..))	AccountService 인터페이스의 모든 메서드 실행
execution(* com.xyz.service.*.*(..))	service 패키지의 모든 메서드 실행
execution(* com.xyz.service..*.*(..))	service 패키지과 하위 패키지의 모든 메서드 실행
within(com.xyz.service.*)	service 패키지 내의 모든 결합점(클래스 포함)
bean(*Repository)	이름이 "Repository"로 끝나는 모든 빈
bean(*)	모든 빈
bean(board*)	이름이 "board"로 끝나는 모든 빈
bean(*"service")    bean(*Service)	이름이 "service"나 "Service"로 끝나는 모든 빈

## 27.8 log4j란?

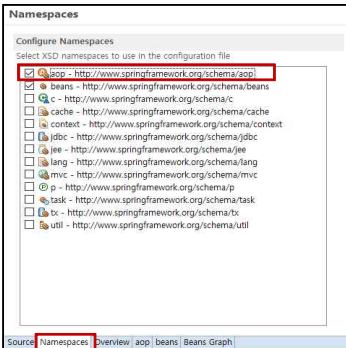
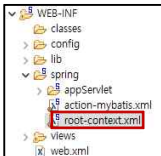
### AOP를 이용한 로그 출력 실습

1. pom.xml에 aop 관련 라이브러리를 설정합니다.

```
189 <!-- AspectJ -->
190 <dependency>
191   <groupId>org.aspectj</groupId>
192   <artifactId>aspectjrt</artifactId>
193   <version>${org.aspectj-version}</version>
194 </dependency>
195
196 <dependency>
197   <groupId>org.aspectj</groupId>
198   <artifactId>aspectjweaver</artifactId>
199   <version>${org.aspectj-version}</version>
200 </dependency>
201
202 <dependency>
203   <groupId>org.aspectj</groupId>
204   <artifactId>aspectjtools</artifactId>
205   <version>${org.aspectj-version}</version>
206 </dependency>
```


## 27.8 log4j란?

2. root-context.xml을 클릭 후 Namespaces 탭을 선택해서 aop항목을 체크해서 aop 설정을 추가합니다.



## 27.8 log4j란?

3. root-context.xml에 aop 애너테이션을 자동 인식할 수 있도록 태그를 추가합니다.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:aop="http://www.springframework.org/schema/aop"
5       xsi:schemaLocation="http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans"
6       >
7
8     <!-- Root Context: defines shared resources visible
9     <aop:aspectj-autoproxy/>
10  </beans>
```

## 27.8 log4j란?

4. LoginAdvice 클래스에 @Aspect 애너테이션을 추가합니다.



```
15 @Component
16 @Aspect
17 public class LoggingAdvice {
18     private static final Logger logger = LoggerFactory.getLogger(LoggingAdvice.class);
```



## 27.8 log4j란?

5. com.myspring.pro27.service 하위 패키지와 com.myspring.pro27.dao 패키지 하위의 모든 클래스의 메서드 호출 전 advice를 적용하도록 설정합니다.

```

20 // target 메서드의 파라미터등 정보를 출력합니다.
21 /*@Before("execution(* com.myspring.pro27.member.service.*(..)) or "
22 + "execution(* com.myspring.pro27.member.dao.*(..))")*/
23 /*@Before("execution(* com.myspring.pro27.*.service.*(..)) or "
24 + "execution(* com.myspring.pro27.*.dao.*(..))")
25 public void startLog(JoinPoint jp) {
26
27     logger.info("-----");
28     logger.info("-----");
29
30     /* 전달되는 모든 파라미터들을 Object의 배열로 가져옵니다. */
31     logger.info("1:" + Arrays.toString(jp.getArgs()));
32
33     /* 해당 Advice의 타입을 알아냅니다. */
34     logger.info("2:" + jp.getKind());
35
36     /* 실행하는 대상 객체의 메소드에 대한 정보를 알아낼 때 사용합니다. */
37     logger.info("3:" + jp.getSignature().getName());
38
39     /* target 객체를 알아낼 때 사용합니다. */
40     logger.info("4:" + jp.getTarget().toString());
41
42     /* Advice를 행하는 객체를 알아낼 때 사용합니다. */
43     logger.info("5:" + jp.getThis().toString());
44
45 }

```

## 27.8 log4j란?

6. com.myspring.pro27.service 하위 패키지와 com.myspring.pro27.dao 패키지 하위의 모든 클래스의 메서드 호출 후 advice를 적용하도록 설정합니다.

```

47=  /*@After("execution(* com.myspring.pro27.member.service.*(..)) or "
48      + "execution(* com.myspring.pro27.member.dao.*(..))")*/
49=  @After("execution(* com.myspring.pro27.*.service.*(..)) or "
50      + "execution(* com.myspring.pro27.*.dao.*(..))")
51  public void after(JoinPoint jp) {
52      logger.info("-----");
53      logger.info("-----");
54
55      // 전달되는 모든 파라미터들을 Object의 배열로 가져옵니다.
56      logger.info("1:" + Arrays.toString(jp.getArgs()));
57
58      // 해당 Advice의 타입을 알아냅니다.
59      logger.info("2:" + jp.getKind());
60
61      // 실행하는 대상 객체의 메소드에 대한 정보를 알아낼 때 사용합니다.
62      logger.info("3:" + jp.getSignature().getName());
63
64      // target 객체를 알아낼 때 사용합니다.
65      logger.info("4:" + jp.getTarget().toString());
66
67      // Advice를 행하는 객체를 알아낼 때 사용합니다
68      logger.info("5:" + jp.getThis().toString());
69
70  }

```

## 27.8 log4j란?

7. com.myspring.pro27.service 하위 패키지와 com.myspring.pro27.dao 패키지 하위의 모든 클래스의 메서드 호출 전,후 advice를 적용하도록 설정합니다.

```

73 // target 메소드의 동작 시간을 측정합니다.
74 @Around("execution(* com.myspring.pro27.member.service.*(..)) or "
75         + "execution(* com.myspring.pro27.member.dao.*(..))")
76 @Around("execution(* com.myspring.pro27.*.service.*(..)) or "
77         + "execution(* com.myspring.pro27.*.dao.*(..))")
78 public Object timeLog(ProceedingJoinPoint pjp) throws Throwable {
79     long startTime = System.currentTimeMillis();
80     logger.info(Arrays.toString(pjp.getArgs()));
81
82     // 실제 타겟을 실행하는 부분이다. 이 부분이 없으면 advice가 적용된 메소드가 동작하지 않습니다.
83     Object result = pjp.proceed(); // proceed는 Exception 보다 상위 Throwable을 처리해야 합니다.
84
85     long endTime = System.currentTimeMillis();
86     // target 메소드의 동작 시간을 출력한다.
87     logger.info(pjp.getSignature().getName() + " : " + (endTime - startTime));
88     logger.info("=====");
89
90     // Around를 사용할 경우 반드시 Object를 리턴해야 합니다.
91     return result;
92 }

```

## 27.8 log4j란?

8. MemberController 클래스에 @EnableAspectJAutoProxy 애너테이션을 적용해서 aop기능을 활성화합니다.

MemberControllerImpl.java

```
26 @Controller("memberController")
27 @EnableAspectJAutoProxy
28 public class MemberControllerImpl implements MemberController {
29     // private static final Logger logger = LoggerFactory.getLogger(M
30     @Autowired
31     private MemberService memberService;
32     @Autowired
33     MemberVO memberVO ;
```

## 27.8 log4j란?

9. 회원 조회 시 호출된 메서드에 대한 정보를 출력합니다.

```
INFO : com.myspring.pro27.common.aop.LoggingAdvice - -----
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 1:[]
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 2:method-execution
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 3:listMembers
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 4:com.myspring.pro27.member.service.MemberServiceImpl@32cec678
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 5:com.myspring.pro27.member.service.MemberServiceImpl@32cec678
INFO : com.myspring.pro27.common.aop.LoggingAdvice - {}
INFO : com.myspring.pro27.common.aop.LoggingAdvice - -----
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 1:[]
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 2:method-execution
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 3:selectAllMemberList
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 4:com.myspring.pro27.member.dao.MemberDAOImpl@32cec678
INFO : com.myspring.pro27.common.aop.LoggingAdvice - 5:com.myspring.pro27.member.dao.MemberDAOImpl@32cec678
INFO : com.myspring.pro27.common.aop.LoggingAdvice - {}
```

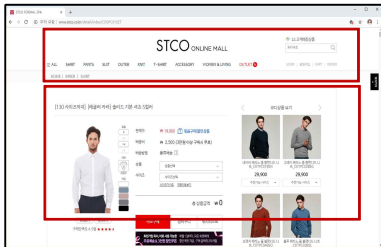
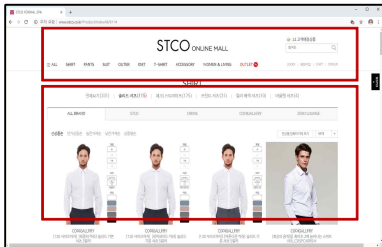
## 27.9 타일즈란?

### 타일즈(tiles)

- 화면의 레이아웃 기능을 제공하는 오픈 소스 라이브러리

### 타일즈 특징

- 페이지 레이아웃을 쉽고 단순하게 구현할 수 있음
- 공통된 레이아웃을 사용하므로 유지관리가 용이함



## 27.9 타일즈란?

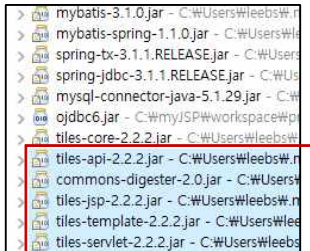
코드 27-14 pro27/pom.xml

```
...
<!-- 타일즈 관련 라이브러리 -->
<dependency>
  <groupId>org.apache.tiles</groupId>
  <artifactId>tiles-core</artifactId>
  <version>2.2.2</version>
</dependency>
<dependency>
  <groupId>org.apache.tiles</groupId>
  <artifactId>tiles-jsp</artifactId>
  <version>2.2.2</version>
</dependency>
<dependency>
  <groupId>org.apache.tiles</groupId>
  <artifactId>tiles-servlet</artifactId>
  <version>2.2.2</version>
</dependency>
...
```

타일즈 관련 라이브러리를 설치합니다.

## 27.9 타일즈란?

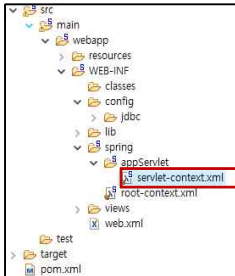
타일즈 관련 라이브러리가 설치된 상태





## 27.9 타일즈란?

### 타일즈 관련 xml 설정하기



## 27.9 타일즈란?

## servlet.xml

코드 27-15 pro27/src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml

```

...
<!--
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>
-->
<beans:bean id="tilesConfigurer"
            class="org.springframework.web.servlet.view.tiles2.TilesConfigurer">
    <beans:property name="definitions">
        <beans:list>
            <beans:value>classpath:tiles/*.xml</beans:value>
        </beans:list>
    </beans:property>
    <beans:property name="preparerFactoryClass"
        value="org.springframework.web.servlet.view.tiles2.
            SpringBeanPreparerFactory" />
</beans:bean>
<beans:bean id="viewResolver"
            class="org.springframework.web.servlet.view.UrlBasedViewResolver">
    <beans:property name="viewClass"
        value="org.springframework.web.servlet.view.tiles2.TilesView" />
</beans:bean>
<context:component-scan base-package="com.myspring.pro27" />
</beans:beans>

```

더 이상 JSP 뷰리졸버를 사용하지 않으므로 주석 처리합니다.

스프링의 TilesConfigurer 클래스를 이용해 tilesConfigurer 빈을 생성합니다.

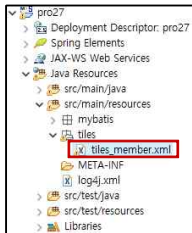
패키지 tiles의 모든 설정 XML 파일을 읽어 들입니다.

타일즈 뷰리졸버를 사용해 화면을 표시합니다.

## 27.10 JSP에 타일즈 사용하기

### • 27.10.1 tiles.xml 작성하기

1. src/main/resources 패키지에 tiles 패키지를 만든 후 tiles\_member.xml 파일을 생성합니다.



## 27.10 JSP에 타일즈 사용하기

### 2. tiles\_member.xml을 볼까요?

코드 27-16 pro27/src/main/resources/tiles/tiles\_member.xml

```

...
<tiles-definitions>
  <definition name="baseLayout" template="/WEB-INF/views/common/layout.jsp">
    <put-attribute name="title" value="" />
    <put-attribute name="header" value="/WEB-INF/views/common/header.jsp" />
    <put-attribute name="side" value="/WEB-INF/views/common/side.jsp" />
    <put-attribute name="body" value="" />
    <put-attribute name="footer" value="/WEB-INF/views/common/footer.jsp" />
  </definition>
  <definition name="main" extends="baseLayout">
    <put-attribute name="title" value="메인 페이지" />
    <put-attribute name="body" value="/WEB-INF/views/main.jsp" />
  </definition>
</tiles-definitions>

```

공통 레이아웃의 뷰 이름을 지정합니다.

전체 레이아웃을 정하는 JSP의 위치를 지정합니다.

레이아웃에서 상단(헤더)을 구성하는 JSP의 위치를 지정합니다.

레이아웃에서 사이드 메뉴를 구성하는 JSP의 위치를 지정합니다.

메인 화면의 뷰 이름을 지정합니다.

레이아웃에서 하단을 구성하는 JSP의 위치를 지정합니다.

기본적인 레이아웃은 baseLayout을 상속받습니다.

레이아웃의 제목에 표시할 문구를 지정합니다.

레이아웃의 본문에 표시할 JSP를 지정합니다.

## 27.9 타일즈란?

- 27.10.2 레이아웃에 사용되는 JSP 작성하기

1. 반드시 tiles\_member.xml에서 지정한 경로에 레이아웃 관련 JSP들이 위치해야 합니다.



resources 폴더 하위에 image 폴더를 만든 후 이미지를 위치시킵니다.

tiles\_member.xml에서 지정한 위치에 JSP를 위치시킵니다.

## 27.10 JSP에 타일즈 사용하기

2. 먼저 화면 전체의 구조를 정의하는 layout.jsp부터 작성해 보겠습니다. tiles\_member.xml 설정에 따라서 각각의 위치에 JSP를 표시합니다.

**코드 27-17** pro27/src/main/webapp/WEB-INF/views/common/layout.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isELIgnored="false"
%>
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
...
<title><tiles:insertAttribute name="title" />:</title>
</head>
<body>
    <div id="container">
        <div id="header">
            <tiles:insertAttribute name="header"/>
        </div>
        <div id="sidebar-left">
```

자바의 import문처럼 타일즈를 사용하기 위해 반드시 추가해야 합니다.

tiles\_member.xml의 <definition>의 하위 태그인 <put-attribute> 태그의 name이 title인 값(value)을 표시합니다.

tiles\_member.xml의 <definition>의 하위 태그인 <put-attribute> 태그의 name이 header인 JSP를 표시합니다.

## 27.10 JSP에 타일즈 사용하기

```
<tiles:insertAttribute name="side"/>
</div>
<div id="content">
  <tiles:insertAttribute name="body"/>
</div>
<div id="footer">
  <tiles:insertAttribute name="footer"/>
</div>
</body>
</html>
```

tiles\_member.xml의 <definition>의 하위 태그인 <put-attribute>  
태그의 name이 side인 JSP를 표시합니다.

tiles\_member.xml의 <definition>의 하위 태그인 <put-attribute>  
태그의 name이 body인 JSP를 표시합니다.

tiles\_member.xml의 <definition>의 하위 태그인 <put-attribute>  
태그의 name이 footer인 JSP를 표시합니다.

---

## 27.10 JSP에 타일즈 사용하기

### 3. header.jsp를 다음과 같이 작성합니다

3 코드 27-18 pro27/src/main/webapp/WEB-INF/views/common/header.jsp

```
...
<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>상단부</title>
</head>
<body>
<table border=0 width="100%">
  <tr>
    <td>
      <a href="${contextPath}/main.do">
        
      </a>
    </td>
    <td>
      <h1><font size=30>스프링실습 홈페이지!!</font></h1>
    </td>
    <td>
      <a href="#"><h3>로그인</h3></a>
    </td>
  </tr>
</table>
...
```

src/main/webapp/resources/image 폴더의  
이미지를 표시합니다.



## 27.10 JSP에 타일즈 사용하기

4. 페이지 왼쪽 메뉴와 하단을 담당하는 side.jsp, footer.jsp를 각각 다음과 같이 작성합니다.

**코드 27-19** pro27/src/main/webapp/WEB-INF/views/common/side.jsp

```
...
<title>사이드 메뉴</title>
</head>
<body>
  <h1>사이드 메뉴</h1>
  <h1>
    <a href="#" class="no-underline">회원관리</a><br>
    <a href="#" class="no-underline">게시판관리</a><br>
    <a href="#" class="no-underline">상품관리</a><br>
  </h1>
</body>
</html>
```

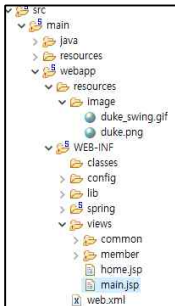
**코드 27-20** pro27/src/main/webapp/WEB-INF/views/common/footer.jsp

```
...
<body>
  <p> e-mail:admin@test.com</p>
  <p> 회사주소:서울시 강동구</p>
  <p>찾아오는 길:<a href="#">약도</a></p>
</body>
</html>
```

## 27.10 JSP에 타일즈 사용하기

### • 27.10.3 레이아웃에 표시되는 JSP 작성하기

1. 타일즈 설정 파일에서 설정한 위치에 레이아웃 본문에 표시할 JSP 파일인 main.jsp를 준비합니다.



## 27.10 JSP에 타일즈 사용하기

2. main.jsp를 다음과 같이 작성합니다. 브라우저에서 요청하면 이 내용이 레이아웃의 본문에 표시됩니다.

**코드 27-21** pro27/src/main/webapp/WEB-INF/views/main.jsp

```
...
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>메인 페이지</title>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
  <h1>메인 페이지입니다!!</h1>
</body>
</html>
```

---

## 27.10 JSP에 타일즈 사용하기

- 27.10.4 뷰이름 요청 컨트롤러 만들기

1. 자바 컨트롤러를 담당하는 HomeController 클래스를 다음과 같이 준비합니다.



## 27.10 JSP에 타일즈 사용하기

2. return문 다음의 문자열이 <definition> 태그의 뷰이름과 동일합니다.

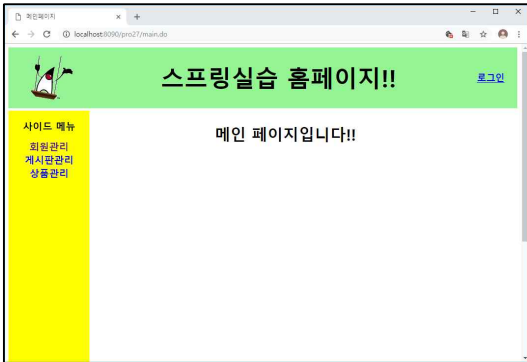
**코드 27-22** pro27/src/java/com/myspring/pro27/HomeController.java

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

/main.do로 요청 시 컨트롤러에서는 <definition> 태그에서  
설정한 뷰이름 main을 타일즈 뷰리출버로 반환합니다.

## 27.10 JSP에 타일즈 사용하기

3. `http://localhost:8090/pro27/main.do`로 요청하여 결과를 확인합니다.



## 27.11 JSP에 회원 목록 표시하기

1. 먼저 tiles\_member.xml에 /member/listMember.do로 요청했을 때 표시할 <definition> 태그를 추가합니다. name의 값은 URL 요청명에서 .do를 제외한 요청명과 일치해야 합니다.

코드 27-23 pro27/src/main/resources/tiles/tiles\_member.xml

```

...
<definition name="/member/listMembers" extends="baseLayout">
  <put-attribute name="title" value="회원목록창" />
  <put-attribute name="body" value="/WEB-INF/views/member/listMembers.jsp" />
</definition>
</tiles-definitions>

```

컨트롤러에서 반환되는 뷰이름을 지정합니다.

기본 레이아웃을 상속받습니다.

JSP 페이지의 제목을 지정합니다.

레이아웃 페이지의 본문에 표시할 JSP를 지정합니다.

## 27.11 JSP에 회원 목록 표시하기

2. 브라우저에서 컨트롤러 요청 시 요청명에 대해 뷰이름을 가져옵니다. 그리고 다시 ModelAndView 객체에 설정한 후 뷰리졸버로 반환합니다.

코드 27-24 pro27/src/main/java/com/myspring/pro27/member/controller/MemberControllerImpl.java

```
...
@Controller("memberController")
public class MemberControllerImpl implements MemberController {
    private static final Logger logger =
        LoggerFactory.getLogger(MemberControllerImpl.class);

    @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);
        logger.debug("debug레벨 : viewName = "+ viewName);
        List membersList = memberService.listMembers();
        ModelAndView mav = new ModelAndView(viewName ;
        mav.addObject("membersList", membersList);
        return mav;
    }
    ...
```

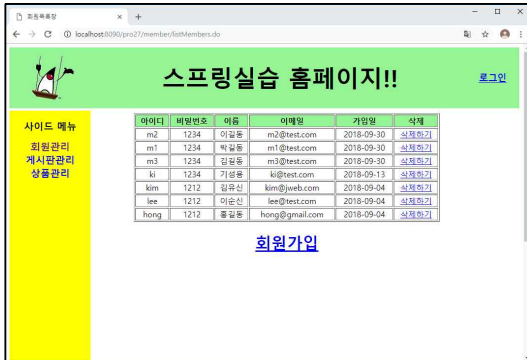
viewName이 <definition> 태그에 설정한 뷰이름과 일치합니다.

ModelAndView 객체에 설정한 뷰이름을 타입즈 뷰리졸버로 반환합니다.



## 27.11 JSP에 회원 목록 표시하기

3. 톰캣을 재실행한 후 <http://localhost:8090/pro27/member/listMembers.do>로 요청합니다



스프링실습 홈페이지!! [로그인](#)

사이드 메뉴  
회원관리  
게시판관리  
상품관리

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

[회원가입](#)

## 27.12 로그인 기능 구현하기

1. 먼저 타일즈를 설정합니다. tiles\_member.xml에 요청했을 때 로그인창을 나타내주는 <definition> 태그를 추가합니다.

**코드 27-25** pro27/src/main/resources/tiles/tiles\_member.xml

```
...  
    <definition name="/member/loginForm" extends="baseLayout">  
        <put-attribute name="title" value="로그인창" />  
        <put-attribute name="body" value="/WEB-INF/views/member/loginForm.jsp" />  
    </definition>  
...
```

---

## 27.12 로그인 기능 구현하기

2. 로그인창에서 입력한 ID와 비밀번호로 회원 정보를 조회하는 SQL문을 매퍼 파일에 추가합니다.

**코드 27-26** pro27/src/main/resources/mybatis/mappers/member.xml

```
...  
<select id="loginById" resultType="memberVO" parameterType="memberVO" >  
  <![CDATA[  
    select * from t_member  
    where id = #{id} and pwd = #{pwd}  
  ]]>  
</select>  
...
```

---

## 27.12 로그인 기능 구현하기

3. MemberControllerImpl 클래스를 다음과 같이 수정합니다.

코드 27-27 pro27/src/main/java/com/myspring/pro27/member/controller/MemberControllerImpl.java

```

...
@Override
@RequestMapping(value = "/member/login.do", method = RequestMethod.POST)
public ModelAndView login(@ModelAttribute("member") MemberVO member,
    RedirectAttributes rAttr,
    HttpServletRequest request,
    HttpServletResponse response) throws Exception {
    ModelAndView mav = new ModelAndView();
    memberVO = memberService.login(member)
    if(memberVO != null) {
        HttpSession session = request.getSession();
        session.setAttribute("member", memberVO);
        session.setAttribute("isLogOn", true);
        mav.setViewName("redirect:/member/listMembers.do");
    } else {
        rAttr.addAttribute("result", "loginFailed");
        mav.setViewName("redirect:/member/loginForm.do");
    }
    return mav;
}

```

로그인창에서 전송된 ID와 비밀번호를 MemberVO 객체인 member에 저장합니다.

RedirectAttributes 클래스를 이용해 로그인 실패 시 다시 로그인창으로 리다이렉트하여 실패 메시지를 전달합니다.

login() 메서드를 호출하면서 로그인 정보를 전달합니다.

세션에 회원 정보를 저장합니다.

세션에 로그인 상태를 true로 설정합니다.

memberVO로 반환된 값이 있으면 세션을 이용해 로그인 상태를 true로 합니다.

로그인 실패 시 실패 메시지를 로그인창으로 전달합니다.

로그인 실패 시 다시 로그인창으로 리다이렉트합니다.

## 27.12 로그인 기능 구현하기

@Override

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

public ModelAndView logout(HttpServletRequest request,

HttpServletResponse response) throws Exception {

HttpSession session = request.getSession();

session.removeAttribute("member");

로그아웃 요청 시 세션에 저장된 로그인 정보와  
회원 정보를 삭제합니다

session.removeAttribute("isLogOn");

ModelAndView mav = new ModelAndView();

mav.setViewName("redirect:/member/listMembers.do");

return mav;

}

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

private ModelAndView form(@RequestParam(value = "result", required=false) String result,

HttpServletRequest request,

HttpServletResponse response) throws Exception {

String viewName = getViewName(request);

로그인창 요청 시 매개변수 result가 전송되면  
변수 result에 값을 저장합니다. 최초로 로그인창  
을 요청할 때는 매개변수 result가 전송되지 않  
으므로 무시합니다.

ModelAndView mav = new ModelAndView();

mav.addObject("result", result);

mav.setViewName(viewName);

return mav;

}

...

## 27.12 로그인 기능 구현하기

4. Service 클래스에서는 다시 MemberDAO의 loginById() 메서드를 호출하면서 전달된 ID와 비밀번호를 전달하도록 구현합니다.

**코드 27-28** pro27/src/main/java/com/myspring/pro27/member/service/MemberServiceImpl.java

```
...  
@Override  
public MemberVO login(MemberVO memberVO) throws Exception{  
    return memberDAO.loginById(memberVO);  
}  
...
```

---

## 27.12 로그인 기능 구현하기

5. Service 클래스에서 전달된 memberVO 객체를 다시 SQL문으로 전달하여 ID와 비밀번호를 이용해 회원 정보를 조회하도록 DAO 클래스를 수정합니다.

**코드 27-29** pro27/src/main/java/com/myspring/pro27/memberdao/MemberDAOImpl.java

```
...
public MemberVO loginById(MemberVO memberVO) throws DataAccessException{
    MemberVO vo = sqlSession.selectOne("mapper.member.loginById",memberVO);
    return vo;
}
...
```

메서드 호출 시 전달된 memberVO를 SQL문으로 전달해 ID와 비밀번호에 대한 회원 정보를 MemberVO 객체로 반환합니다.

---

## 27.12 로그인 기능 구현하기

6. 마지막으로 JSP 파일을 구현할 차례입니다. 또한 컨트롤러에서 설정한 세션의 속성인 isLogOn의 값이 true이면 회원 이름과 로그아웃이 표시되도록 header.jsp를 작성합니다.

**코드 27-30** pro27/src/main/webapp/WEB-INF/views/common/header.jsp

```
...
<td>
  <!-- <a href="#"><h3>로그인</h3></a> -->
  <c:choose>
    <c:when test="${isLogOn == true && member != null}">
      <h3>환영합니다. ${member.name }님!</h3>
      <a href="${contextPath}/member/logout.do"><h3>로그아웃</h3></a>
    </c:when>
    <c:otherwise>
      <a href="${contextPath}/member/loginForm.do"><h3>로그인</h3></a>
    </c:otherwise>
  </c:choose>
</td>
</tr>
</table>
...
```

로그아웃 링크를 클릭하면 로그인을 표시합니다.

로그인 링크를 클릭하면 로그인창을 요청합니다.

isLogOn 속성 값을 체크하여 로그인 상태 시 로그아웃이 표시되게 합니다.



## 27.12 로그인 기능 구현하기

7. loginForm.jsp에서는 로그인창이 브라우저에 표시될 때 컨트롤러에서 로그인 실패 메시지가 전달되면 로그인 실패 경고 문구를 먼저 표시해 줍니다.

**코드 27-31** pro27/src/main/webapp/WEB-INF/views/member/loginForm.jsp

```
...
<head>
  <meta charset="UTF-8">
  <title>로그인창</title>
  <c:choose>
    <c:when test="${result=='loginFailed'}">
      <script>
        window.onload=function(){
          alert("아이디나 비밀번호가 틀립니다. 다시 로그인 하세요!");
        }
      </script>
    </c:when>
  </c:choose>
</head>
...
```

로그인 실패 시 리다이렉트되면서 로그인 실패 메시지를 표시합니다.

## 27.12 로그인 기능 구현하기

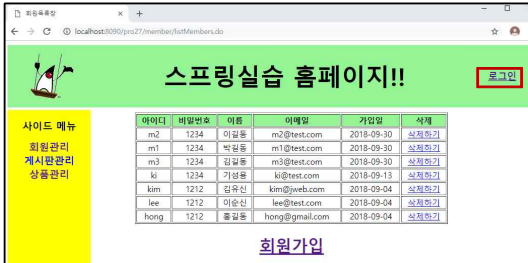
8. 실행 결과를 볼까요? 다음과 같이 상단의 로그인을 클릭한 후 로그인창에서 ID와 비밀번호를 입력하여 로그인합니다.

9. 로그인에 성공하면 사용자 이름과 함께 로그아웃 링크를 표시합니다.

아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	<a href="#">삭제하기</a>
m1	1234	박길동	m1@test.com	2018-09-30	<a href="#">삭제하기</a>


## 27.12 로그인 기능 구현하기

10. 반대로 로그아웃을 클릭하면 다시 로그인 링크를 표시합니다.



회원목록

← → ↻ local:8090/pro27/member/list/members.do ☆

 스프링실습 홈페이지!! **로그인**

사이드 메뉴

- 회원관리
- 게시판관리
- 상품관리

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

[회원가입](#)

## 27.12 로그인 기능 구현하기

11. ID와 비밀번호가 잘못된 경우에는 로그인 실패 메시지를 출력하고 다시 로그인창을 표시합니다.

