

32장 스프링 부트 사용하기

32.1 스프링 부트란?

32.9 인터셉터 사용하기

32.2 스프링 부트 전용 STS 설치하기

32.3 스프링 부트 프로젝트 생성하기

32.4 스프링 부트 프로젝트 실행하기

32.5 스프링 부트 웹 페이지 만들기

32.6 그레이들 이용해 스프링 부트 실행하기

32.7 마이바티스 사용하기

32.8 타일즈 사용하기

32.1 스프링 부트란?

스프링 부트(Spring Boot)

- 메이븐의 라이브러리 자동 업데이트 기능을 이어받으면서 기존 스프링 프레임워크의 복잡한 설정 과정은 최대한 줄이면서 개발할 수 있음
- 배포 또는 테스트 역시 스프링 프레임워크보다 쉽고 빠르게 할 수 있음

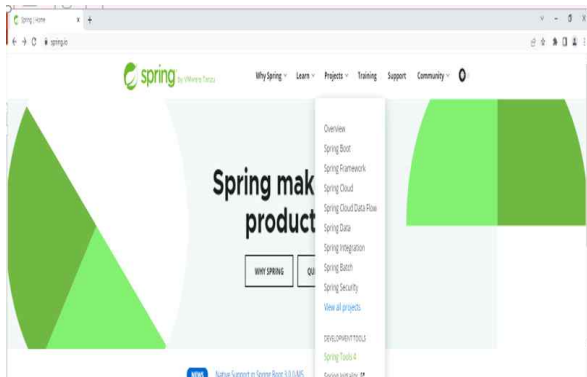
스프링 부트 특징

- 일반적인 응용 프로그램을 단독으로 실행하는 수준으로 스프링 애플리케이션을 구현할 수 있음
- 프로젝트 환경을 구축할 때 필요한 톰캣, Jetty, UnderFlow 같은 서버 외적인 툴이 내장되어 있어 따로 설치할 필요가 없음
- XML 기반 설정이나 코드 없이 환경 설정을 자동화할 수 있음
- 의존성 관리를 쉽게 자동으로 할 수 있음

32.1 스프링 부트 전용 STS 설치하기

- 32.2 스프링 부트 전용 STS 설치하기

1. <https://spring.io>로 접속한 후 맨 하단의 TOOLS를 클릭합니다.



32.1 스프링 부트 전용 STS 설치하기

2. 자신의 운영체제에 맞는 STS4를 선택하여 다운로드합니다.

Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.
Free. Open source.

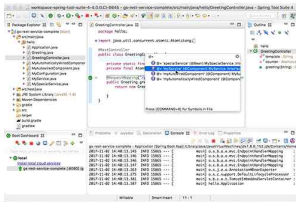
4.17.1 - LINUX X86_64

4.17.1 - LINUX ARM_64

4.17.1 - MACOS X86_64

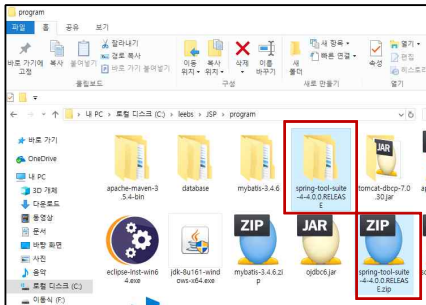
4.17.1 - MACOS ARM_64

4.17.1 - WINDOWS X86_64



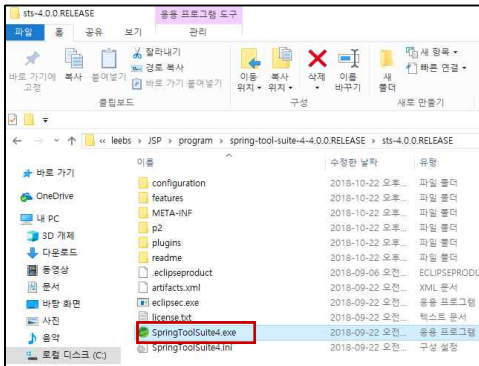
32.1 스프링 부트 전용 STS 설치하기

3. 다운로드를 마치면 로컬 PC의 원하는 폴더에 압축을 풉니다.



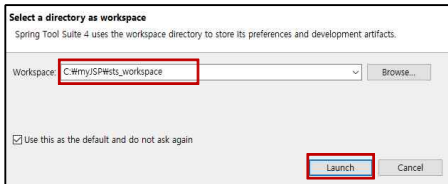
32.1 스프링 부트 전용 STS 설치하기

4. 압축을 푼 폴더로 이동한 후 **SpringToolSuite4.exe**를 더블클릭해서 실행합니다.



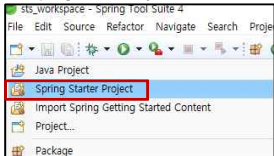
32.1 스프링 부트 전용 STS 설치하기

5. Browse...를 클릭해 워크스페이스를 지정하고 Use this as the default and do not ask again 옵션 체크박스에 체크 한 후 Launch를 클릭합니다.



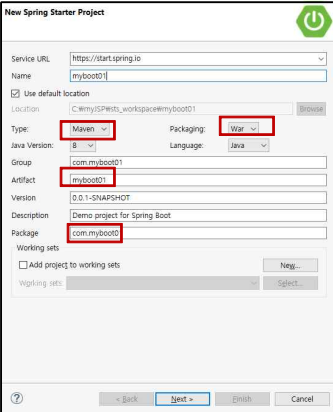
32.3 스프링 부트 프로젝트 생성하기

1. 상단 아이콘(▼)을 클릭하고 **Spring Starter Project**를 선택합니다.



32.3 스프링 부트 프로젝트 생성하기

2. Name 항목에 **myboot01**로 수정하고 Type을 **Maven**, Packaging을 **War**로 선택한 후 Group과 Package 항목을 **com.myboo01**로 변경합니다.



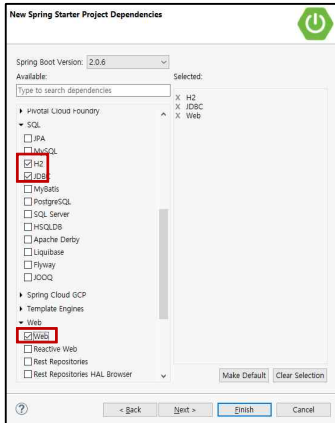
The image shows the 'New Spring Starter Project' dialog box in an IDE. The dialog has a title bar with a green power icon. The fields are as follows:

- Service URL: `https://start.spring.io`
- Name: `myboot01`
- ☒ Use default location
- Location: `C:\my\SP\sts_workspace\myboot01` (with a 'Browse' button)
- Type: `Maven` (selected in a dropdown, highlighted with a red box)
- Packaging: `War` (selected in a dropdown, highlighted with a red box)
- Java Version: `8`
- Language: `Java`
- Group: `com.myboot01`
- Artifact: `myboot01` (highlighted with a red box)
- Version: `0.0.1-SNAPSHOT`
- Description: `Demo project for Spring Boot`
- Package: `com.myboot0` (highlighted with a red box)
- Working sets section:
 - ☐ Add project to working sets (with a 'New...' button)
 - Working sets: (dropdown menu) (with a 'Select...' button)

At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', and 'Cancel'.

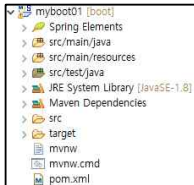
32.3 스프링 부트 프로젝트 생성하기

3. SQL 항목에서 **H2**, **JDBC** 를 선택하고, WEB 항목에서 **web**을 선택합니다.



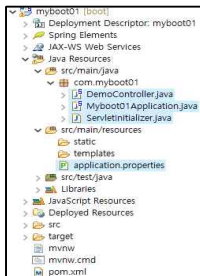
32.3 스프링 부트 프로젝트 생성하기

4. myboo01 프로젝트가 생성된 것을 확인합니다.



32.4 스프링 부트 프로젝트 실행하기

실습 파일 위치



1. application.properties에서는 프로젝트 전체와 관련된 기능을 설정합니다. 먼저 톰캣 요청 포트 번호와 세션 유효 시간을 설정합니다.

코드 32-1 myboot01/src/main/resources/application.properties

```
#Server
```

```
server.port=8090
```

톰캣 포트 번호를 설정합니다.

```
server.session.timeout=360000
```

32.4 스프링 부트 프로젝트 실행하기

2. 다음은 스프링 프로젝트 생성 시 자동으로 만들어지는 main() 메서드입니다.

코드 32-2 myboot01/src/main/java/com/myboot01/Myboot01Application.java

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication  
public class Myboot01Application {  
    public static void main(String[] args) {  
        SpringApplication.run(Myboot01Application.class, args);  
    }  
}
```

스프링 부트 애플리케이션으로 설정합니다.

스프링 부트 프로젝트는 반드시 main() 메서드가 있어야 합니다.

❖ Note

부트 프로젝트는 main() 메서드를 시작점으로 실행하므로 Myboot01Application.java가 반드시 있어야 합니다. 이는 스프링 부트의 웹 애플리케이션을 일반 자바 애플리케이션처럼 개발하려는 의도 때문입니다.

그리고 ServletInitializer.java 파일에 생성된 ServletInitializer 클래스는 SpringBootServletInitializer 클래스를 상속받습니다. SpringBootServletInitializer의 역할은 스프링 부트 애플리케이션을 web.xml 없이 톰캣에서 실행하게 해주는 것입니다.

32.4 스프링 부트 프로젝트 실행하기

3. DemoController 컨트롤러 클래스는 모든 요청에 대해 "Hello Boot!"라는 메시지를 브라우저에 출력하는 역할을 합니다.

코드 32-3 myboot01/src/main/java/com/myboot01/DemoController.java

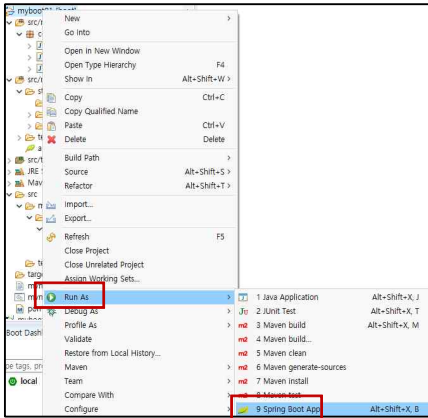
```
package com.myboot01;

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

@Controller
public class DemoController {
    @ResponseBody
    @RequestMapping("/") •————— 모든 요청을 처리합니다.
    public String home(){
        System.out.println("Hello Boot!!");
        return "Hello Boot!!"; •————— 브라우저로 출력합니다.
    }
}
```

32.4 스프링 부트 프로젝트 실행하기

4. 스프링 부트 애플리케이션은 내장된 톱켓을 통해 실행합니다. 따라서 예전처럼 톱켓을 설치해 애플리케이션을 등록할 필요 없이 일반 자바 애플리케이션처럼 실행하면 됩니다.



32.4 스프링 부트 프로젝트 실행하기

5. 스프링 부트 애플리케이션이 실행되면서 STS 콘솔창에 다음과 같은 로그가 출력됩니다.

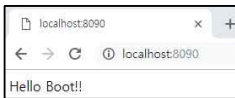
```

  ____
 /  _ \
/  / \  \
/  /  __ \
/___/  /_/
:: Spring Boot ::      (v2.0.6.RELEASE)

2018-10-22 17:49:05.836 INFO 71192 --- [main]
2018-10-22 17:49:05.841 INFO 71192 --- [main]
2018-10-22 17:49:05.963 INFO 71192 --- [main]

```

6. `http://localhost:8090`으로 요청하면 브라우저에서 "Hello Boot!!"를 출력합니다.



7. STS 콘솔에도 Hello Boot!!를 출력합니다.

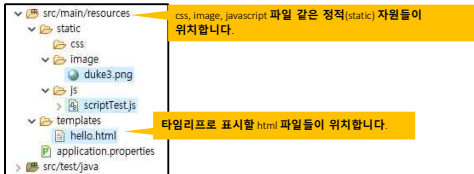
```

2018-10-22 12:46:56.990 INF
2018-10-22 12:46:56.991 INF
2018-10-22 12:46:57.029 INF
Hello Boot!!

```

32.5 스프링 부트 웹 페이지 만들기

실습 파일 위치



1. src/main/resources 아래 static 폴더에 정적 자원들을 저장할 css, image, js 폴더를 각각 만듭니다.
2. image 폴더에 실습에 사용할 이미지 파일을 복사해서 붙여 넣습니다.

32.5 스프링 부트 웹 페이지 만들기

2. application.properties 파일에 JSP 파일 위치를 설정합니다.

코드 32-9 myboot01/src/main/resources/application.properties

```
#Server
server.port=8090
server.session.timeout=360000
```

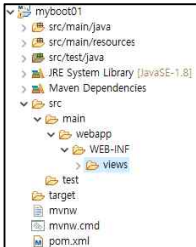
```
#Spring MVC
```

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

src/main/webapp 폴더를 기준으로 JSP의
위치를 설정합니다.

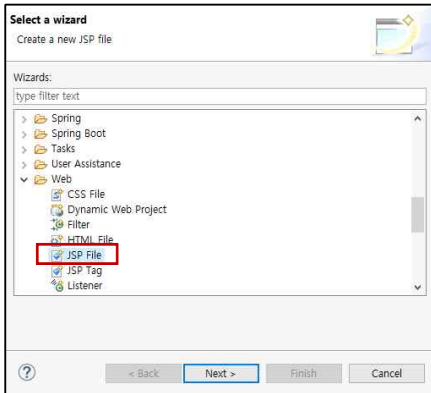
32.5 스프링 부트 웹 페이지 만들기

3. application.properties 파일에서 지정한 경로인 src/main/webapp/WEB-INF 폴더 하위에 views 폴더를 생성합니다.



32.5 스프링 부트 웹 페이지 만들기

4. views 폴더에서 마우스 오른쪽 버튼을 클릭한 후 New > Other...를 클릭하고 Web 항목의 JSP File을 선택하고 Next를 클릭합니다.



32.5 스프링 부트 웹 페이지 만들기

5. 파일 이름이 hello.jsp인 JSP 파일이 생성된 것을 확인할 수 있습니다.



32.5 스프링 부트 웹 페이지 만들기

6. hello.jsp를 다음과 같이 작성합니다

코드 32-10 myboot01/src/main/webapp/WEB-INF/views/ hello.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" %>
<c:set var="contextPath" value="${pageContext.request.contextPath}" />
<%
    request.setCharacterEncoding("UTF-8");
%>
<html>
<head>
    <script src="${contextPath}/js/scriptTest.js" type="text/javascript"></script>
    <meta charset="utf-8">
    <title>hello.JSP 페이지</title>
</head>
<body>
    안녕하세요 <br>
    <h2>${message}</h2>
     <br>
    <input type="button" name="테스트" value="테스트" onClick="test();">
</body>
</html>
```

static 폴더의 자바스크립트 파일 위치를 지정합니다.

static 폴더의 이미지 파일 위치를 지정합니다.

32.5 스프링 부트 웹 페이지 만들기

7. DemoController 클래스의 message 속성 값을 "hello.jsp입니다.!"로 변경한 후 브라우저에서 <http://localhost:8090/hello.do>로 요청하여 결과를 확인합니다



32.6 그레이들 이용해 스프링 부트 실행하기

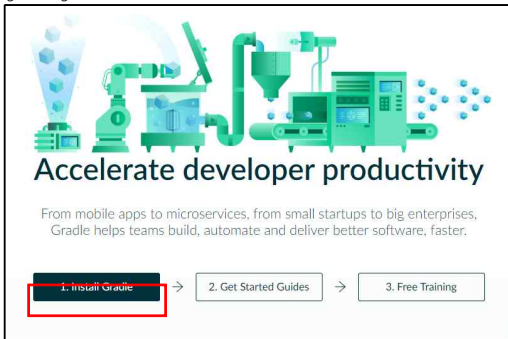
그레이들 특징

- 메이븐이 XML 기반의 정적인 빌드를 제공했다면, 그레이들은 그루비(groovy) 스크립트 기반의 동적인 빌드 기능을 제공
- 메이븐보다 빌드 작업이 간단하며 프로그래밍만으로 기능을 추가할 수 있음

32.6 그레이들 이용해 스프링 부트 실습하기

- **32.6.1 그레이들 설치하기**

1. www.gradle.org에 접속하여 Install Gradle을 클릭합니다.



32.6 그레이들 이용해 스프링 부트 실습하기

2. 화면 중간에 있는 Install manually로 이동해서 **Download** 링크를 클릭합니다.

Install manually

Step 1. **Download** the latest Gradle distribution

The current Gradle release is version 4.9, released on 16 Jul 2018. The distribution zip file comes in two flavors:

- [Binary-only](#)
- [Complete](#), with docs and sources

If in doubt, choose the binary-only version and browse [docs](#) and [sources](#) online.

Need to work with an older version? See the [releases page](#).

Step 2. Unpack the distribution

Linux & MacOS users

Unzip the distribution zip file in the directory of your choosing, e.g.:

```
$ mkdir /opt/gradle
$ unzip -d /opt/gradle gradle-4.9-bin.zip
$ ls /opt/gradle/gradle-4.9
LICENSE NOTICE bin getting-started.html init.d lib media
```

Microsoft Windows users

32.6 그레이들 이용해 스프링 부트 실습하기

3. v4.10.2의 **binary-only**를 클릭해서 다운로드합니다.

There are many [Gradle tutorials](#) available to help you get started quickly. Source distributions also include many [working samples](#) for which guides are not yet written.

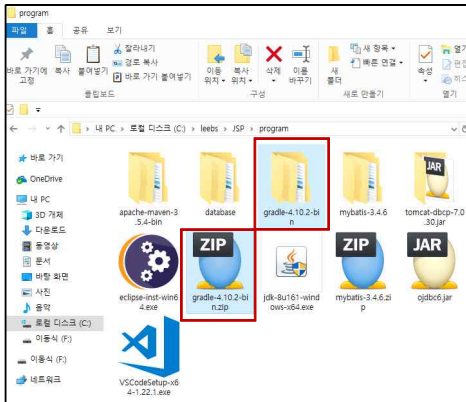
 v4.10.2

 Sep 19, 2018

- Download [binary-only](#) or [complete](#)
- [User Manual](#)
- [API Javadoc](#)
- [DSL Reference](#)
- [Release Notes](#)

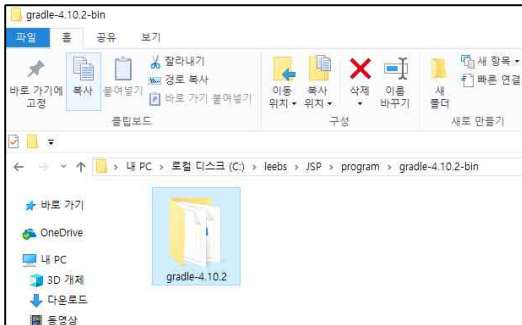
32.6 그레이들 이용해 스프링 부트 실습하기

4. 다운로드 후 파일의 압축을 풉니다.

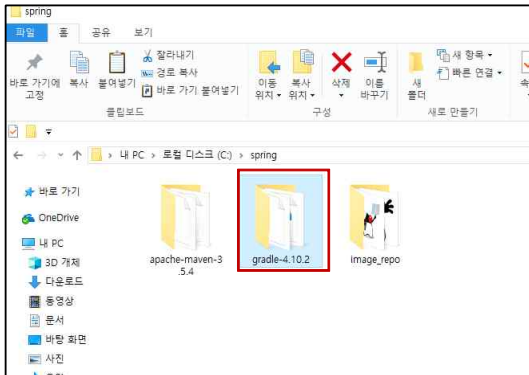


32.6 그레이들 이용해 스프링 부트 실습하기

5. 압축 푼 폴더로 이동해서 gradle-4.10.2 폴더를 복사해서 C 드라이브의 spring 폴더에 붙여 넣습니다.

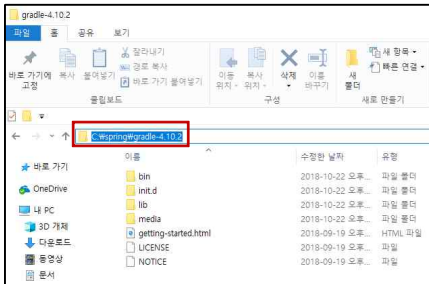


32.6 그레이들 이용해 스프링 부트 실습하기



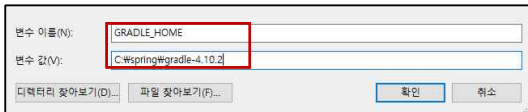
32.6 그레이들 이용해 스프링 부트 실습하기

6. gradle-4.10.2 폴더로 이동한 후 해당 경로를 복사합니다.



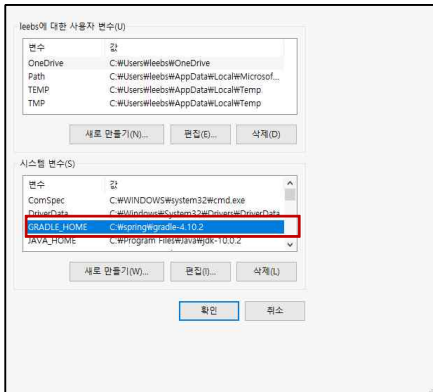
32.6 그레이들 이용해 스프링 부트 실행하기

7. 윈도 탐색기의 **내 PC**에서 마우스 오른쪽 버튼을 눌러 **속성 > 고급 시스템 설정 > 환경변수**를 클릭해서 그레이들 환경 변수를 설정합니다.



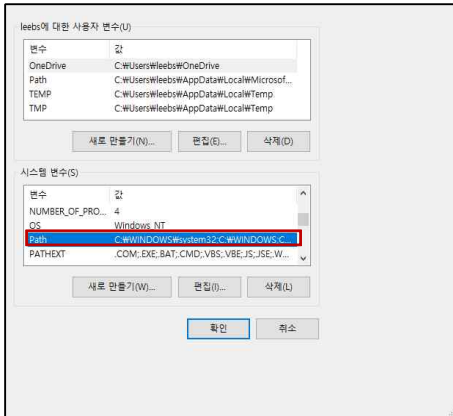
32.6 그레이들 이용해 스프링 부트 실행하기

8. 환경 변수가 생성되었음을 확인합니다



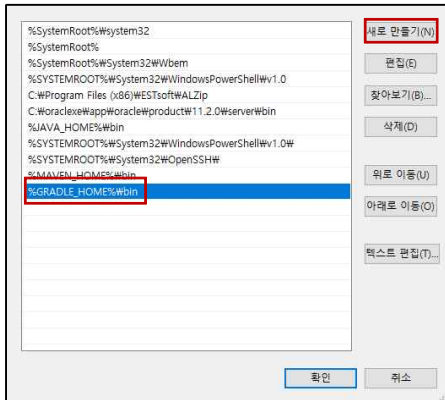
32.6 그레이들 이용해 스프링 부트 실행하기

9. 이번에는 환경변수 **Path**를 클릭합니다.



32.6 그레이들 이용해 스프링 부트 실행하기

10. 새로 만들기를 클릭해서 그레이들의 bin 폴더 경로를 설정하고 확인을 클릭합니다.



32.6 그레이들 이용해 스프링 부트 실습하기

11. 정상적으로 동작하는지 명령 프롬프트를 열고 `gradle -v` 명령어를 입력합니다. 다음과 같이 그레이들 버전이 나오면 성공입니다!



```
선택 명령 프롬프트
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\leebs>gradle -v

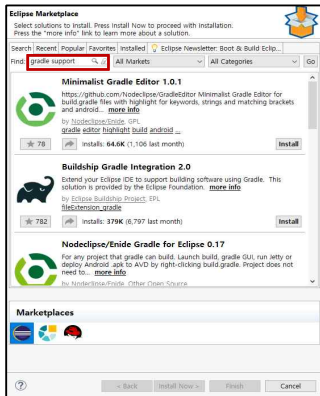
Welcome to Gradle 4.10.2!

Here are the highlights of this release:
- Incremental Java compilation by default
- Periodic Gradle caches cleanup
- Gradle Kotlin DSL 1.0-RC6
- Nested included builds
- SNAPSHOT plugin versions in the 'plugins {}' block
```

32.6 그레이들 이용해 스프링 부트 실습하기

• 32.6.2 그레이들 플러그인 설치하기

1. 이클립스에서 **Help > Eclipse MarketPlace** 선택하고 검색 창에 **gradle support**를 입력하고 **Go**를 눌러 검색합니다.



32.6 그레이들 이용해 스프링 부트 실습하기

2. BuildShip Gradle Integration 2.0의 **Install**을 클릭합니다.



Buildship Gradle Integration 2.0

Extend your Eclipse IDE to support building software using Gradle. This solution is provided by the Eclipse Foundation. [more info](#)

by [Eclipse Buildship Project](#), [EPL](#),
[fileExtension_gradle](#)

★ 782

 Installs: **379K** (6,797 last month)

Install



Buildship Gradle Integration 3.0

Extend your Eclipse IDE to support building software using Gradle. This solution is provided by the Eclipse Foundation. [more info](#)

by [Eclipse Buildship Project](#), [EPL](#),
[fileExtension_gradle](#)

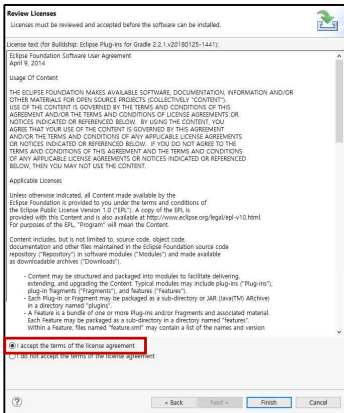
★ 1306

 Installs: **501K** (632 last month)

installed

32.6 그레이들 이용해 스프링 부트 실행하기

3. 사용권에 동의한다고 체크한 후 **Finish**를 클릭해서 설치 완료 후 재실행합니다.



32.6 그레이들 이용해 스프링 부트 실습하기

• 32.6.3 그레이들 기반 스프링 부트 프로젝트 만들기

1. 메뉴에서 File > New > Spring Starter Project를 선택한 후 name을 myboot02로 입력하고 Type을 Gradle로 설정하여 그레이들 기반 스프링 부트 프로젝트를 생성합니다.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

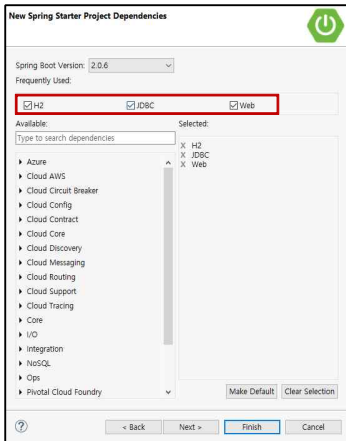
Working sets

☐ Add project to working sets

Working sets:

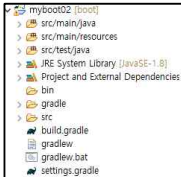
32.6 그레이들 이용해 스프링 부트 실습하기

2. 자주 사용하는 항목(H2, JDBC, Web)에 체크한 후 **Finish**를 클릭합니다.



32.6 그레이들 이용해 스프링 부트 실행하기

3. myboot02로 프로젝트가 생성된 것을 확인합니다.

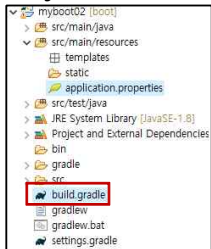


32.6 그레이들 이용해 스프링 부트 실습하기

• 32.6.4 JSP 이용해 웹 페이지 나타내기

그레이들 기반 프로젝트는 pom.xml에 설정하는 것이 아니라 build.gradle에 설정합니다.

build.gradle 위치



32.6 그레이들 이용해 스프링 부트 실습하기

1. build.gradle을 다음과 같이 작성합니다.

코드 32-11 myboot02/build.gradle

```
...  
dependencies {  
    implementation('org.springframework.boot:spring-boot-starter-jdbc')  
    implementation('org.springframework.boot:spring-boot-starter-web')  
    implementation('org.apache.tomcat.embed:tomcat-embed-jasper')  
    implementation('javax.servlet:jstl:1.2')  
    runtimeOnly('com.h2database:h2')  
    testImplementation('org.springframework.boot:spring-boot-starter-test')  
}
```

↑ 톰캣으로 실행하기 위해 설정합니다.
↑ JSP를 사용하기 위해 설정합니다.

32.6 그레이들 이용해 스프링 부트 실습하기

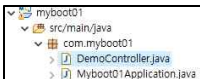
3. application.properties를 다음과 같이 설정합니다.

코드 32-12 myboot02/src/main/resources/application.properties

```
#Server
server.port=8090
server.session.timeout=360000
#Spring MVC
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

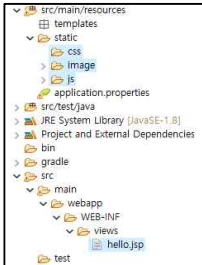
JSP 파일 경로를 지정합니다.

4. 자바 클래스는 앞 절에서 myboot01 프로젝트를 만들 때 사용한 DemoController.java 파일을 복사해 그대로 사용합니다.



32.6 그레이들 이용해 스프링 부트 실습하기

5. src/main/webapp/WEB-INF/views 폴더를 만든 후 마찬가지로 앞 절에서 사용한 hello.jsp를 복사해서 붙여 넣습니다.



32.6 그레이들 이용해 스프링 부트 실행하기

6. 프로젝트 이름 위에서 마우스 오른쪽 버튼을 눌러 **Run As > Spring Boot App**을 클릭해서 실행합니다.
그리고 브라우저에서 `http://localhost:8090/hello.do`로 요청합니다.



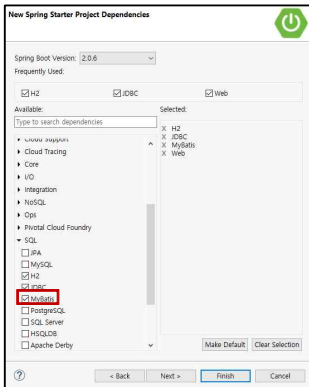
❖ Tip

안드로이드 애플리케이션 개발 도구인 안드로이드 스튜디오는 그레이들을 이용해서 빌드를 수행합니다

32.7 마이바티스 사용하기

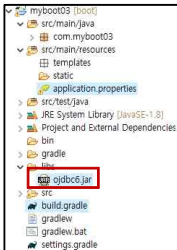
• 32.7.1 마이바티스 적용해 회원 기능 구현하기

1. 새 프로젝트 myboot03을 생성할 때 MyBatis 항목의 체크박스에 체크합니다.



32.7 마이바티스 사용하기

2. 프로젝트 루트 디렉터리에 libs 폴더를 생성하고 마이바티스에서 연동할 오라클 데이터베이스 드라이버인 **ojdbc6.jar**를 복사해서 붙여 넣습니다.



32.7 마이바티스 사용하기

3. build.gradle에서는 로컬에 위치하는 오라클 드라이버를 로컬 리포지토리에 추가하는 설정을 합니다. 그리고 반드시 프로젝트 이름 위에서 마우스 오른쪽 버튼을 클릭한 후 Gradle >Refresh Gradle Project를 선택합니다.

코드 32-13 myboot03/build.gradle

```
...
dependencies {
    implementation('org.springframework.boot:spring-boot-starter-jdbc')
    implementation('org.springframework.boot:spring-boot-starter-web')
    implementation('org.mybatis.spring.boot:mybatis-spring-boot-starter:1.3.2')
    implementation('org.apache.tomcat.embed:tomcat-embed-jasper')
    implementation('javax.servlet:jstl:1.2')
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation files('libs/ojdbc6.jar')
    runtimeOnly('com.h2database:h2')
    testImplementation('org.springframework.boot:spring-boot-starter-test')
}
...
```

↑ JSP를 사용하기 위해 반드시 추가합니다.

↑ 프로젝트 루트 디렉터리의 libs 디렉터리의 jar 파일을 읽어 들입니다.

↑ 오라클 드라이버를 로컬 리포지토리에 추가합니다.

32.7 마이바티스 사용하기

4. application.properties에서는 데이터베이스 연결 정보와 마이바티스 설정 파일 위치를 지정합니다. 그리고 마이바티스 매퍼 파일에서 사용할 alias 클래스의 패키지 이름을 지정합니다. 그러면 해당 패키지에 있는 클래스는 자동으로 alias로 변환됩니다.

코드 32-14 myboot03/src/main/resources/application.properties

```
#Server
server.port=8090
server.session.timeout=360000

#Spring MVC
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

#Database config

```
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.username=scott
spring.datasource.password=tiger
spring.datasource.driverClassName=oracle.jdbc.driver.OracleDriver
```

#mybatis config

```
mybatis.config=classpath:mybatis-config.xml
```

```
mybatis.type-aliases-package=com.myboot03.member.vo
```

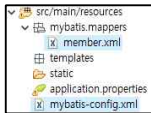
데이터베이스 연결 정보를 설정합니다.

마이바티스 설정 파일 위치를 지정합니다.

마이바티스 매퍼 파일에서 사용할 alias (memberVO)가 있는 패키지를 지정합니다.

32.7 마이바티스 사용하기

5. 다음으로 마이바티스 설정 파일을 구현하기 위해 src/main/resources 폴더에 mybatis.mappers 폴더를 만든 후 member.xml 파일을 생성합니다.



6. 마이바티스 설정 파일인 mybatis-config.xml을 src/main/resources 폴더에 추가한 후 다음과 같이 매퍼 파일의 위치를 지정합니다

코드 32-15 myboot03/src/main/resources/mybatis-config.xml

```

...
<configuration>
  <mappers>
    <mapper resource="mybatis/mappers/member.xml"/>
    <!--
    <mapper resource="mybatis/mappers/board.xml"/>
    -->
  </mappers>
</configuration>

```

src/main/resources에 위치한 매퍼 파일을 지정합니다.

다른 매퍼 파일을 추가합니다.

32.7 마이바티스 사용하기

7. 회원 관련 SQL문을 설정하는 매퍼 파일은 30장에서 실습한 member.xml을 수정하여 사용합니다.

코드 32-16 myboot03/src/main/resources/mybatis/mappers/member.xml

```
...
<mapper namespace="com.myboot03.member.dao.MemberDAO">
    <resultMap id="memResult" type="memberVO">
        <result property="id" column="id" />
        <result property="pwd" column="pwd" />
        <result property="name" column="name" />
        <result property="email" column="email" />
        <result property="joinDate" column="joinDate" />
    </resultMap>

    <select id="selectAllMemberList" resultMap="memResult">
        <![CDATA[
            select * from t_member order by joinDate desc
        ]]>
    </select>

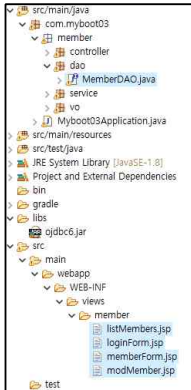
    <insert id="insertMember" parameterType="memberVO">
        <![CDATA[
            insert into t_member(id,pwd, name, email)
            values(#{id}, #{pwd}, #{name}, #{email})
        ]]>
    </insert>
...

```

namespace의 값을 SQL문 호출 시 사용하는 MemberDAO 인터페이스 이름으로 수정합니다.

32.7 마이바티스 사용하기

8. 자바 클래스들은 30장에서 실습한 파일을 사용합니다.



32.7 마이바티스 사용하기

9. 아직 인터셉터를 적용하기 전이므로 컨트롤러 클래스 ModelAndView에 뷰이름을 직접 지정합니다.

코드 32-17 myboot03/src/main/java/com/myboot03/member/controller/MemberControllerImpl.java

```
...
@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 = (String)request.getAttribute("viewName");
        List membersList = memberService.listMembers();
        //ModelAndView mav = new ModelAndView(viewName);
        ModelAndView mav = new ModelAndView("/member/listMembers");
        mav.addObject("membersList", membersList);
        return mav;
    }
    ...
}
```

뷰이름을 직접 지정합니다.

32.7 마이바티스 사용하기

10. MemberDAO 인터페이스를 다음과 같이 작성합니다. 인터페이스에 @Mapper를 적용해 실행 시 매퍼 파일을 읽어 들이고, 인터페이스에 @Repository를 적용합니다. 이때 반드시 추상 메서드 이름과 매퍼 파일에 있는 SQL문의 id는 같아야 합니다.

코드 32-18 myboot03/src/main/java/com/myboot03/member/dao/MemberDAO.java

```
...
@Mapper
@Repository("memberDAO")
public interface MemberDAO {
    public List selectAllMemberList() throws DataAccessException;
    public int insertMember(MemberVO memberVO) throws DataAccessException ;
    public int deleteMember(String id) throws DataAccessException;
    public MemberVO loginById(MemberVO memberVO) throws DataAccessException;
}
```

실행 시 인터페이스에서 매퍼 파일을 읽어 들이도록 지정합니다.

매퍼 파일의 id가 selectAllMemberList인 SQL문을 호출합니다.

매퍼 파일의 id가 loginById인 SQL문을 호출합니다.

매퍼 파일의 id가 deleteMember인 SQL문을 호출합니다.

매퍼 파일의 id가 insertMember인 SQL문을 호출합니다.

32.7 마이바티스 사용하기

11. 프로젝트를 실행한 후 브라우저에서 `http://localhost:8080/member/listMembers.do`로 요청하여 결과를 확인합니다.

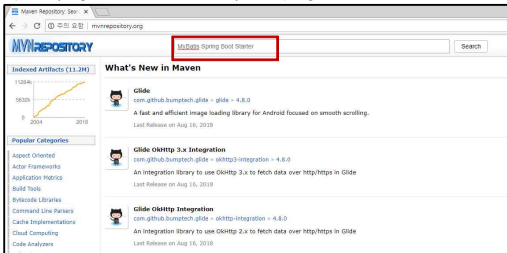
아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04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

회원가입

32.7 마이바티스 사용하기

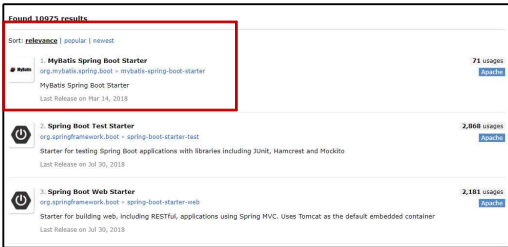
➤ build.gradle에 추가할 라이브러리를 찾는 방법

1. mvnrepository.org에 접속해 검색창에서 MyBatis Spring Boot Starter를 검색합니다.



32.7 마이바티스 사용하기

2. 조회 결과에서 **MyBatis Spring Boot Starter**를 클릭합니다.



The screenshot shows a search interface with the following elements:

- Found 10975 results**: A red box highlights the top search result.
- Sort: relevance | popular | newest**: Sorting options.
- 1. MyBatis Spring Boot Starter**: The first result, highlighted by a red box. It includes the org ID `org.mybatis.spring.boot`, the artifact ID `mybatis-spring-boot-starter`, and the text "MyBatis Spring Boot Starter".
- 71 usages**: The number of usages for the first result, with a blue "Apache" license tag.
- 2. Spring Boot Test Starter**: The second result, including the org ID `org.springframework.boot`, the artifact ID `spring-boot-starter-test`, and the description "Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito".
- 2,068 usages**: The number of usages for the second result, with a blue "Apache" license tag.
- 3. Spring Boot Web Starter**: The third result, including the org ID `org.springframework.boot`, the artifact ID `spring-boot-starter-web`, and the description "Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container".
- 2,181 usages**: The number of usages for the third result, with a blue "Apache" license tag.

32.7 마이바티스 사용하기

3. 추가할 버전을 클릭합니다. 여기서는 1.3.2 버전을 선택합니다.

Home » org.mybatis.spring.boot » mybatis-spring-boot-starter

MyBatis Spring Boot Starter
MyBatis Spring Boot Starter

License: Apache 2.0

Tags: spring, persistence, starter

Used By: 71 artifacts

Central (11)

	Version	Repository
	1.3.2	Central
1.3.x	1.3.1	Central
	1.3.0	Central
	1.2.2	Central

32.7 마이바티스 사용하기

4. Gradle 탭을 클릭하고 내용을 복사한 후 build.gradle에 붙여 넣습니다.



License: Apache 2.0

Date: (Mar 14, 2018)

Files: pom (1 KB) jar (2 KB) View All

Repositories: Central Mulesoft Public Sonatype

Used By: 71 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
// https://www.repository.com/artifact/org.mybatis.spring.boot/mybatis-spring-boot-starter
compile group: 'org.mybatis.spring.boot', name: 'mybatis-spring-boot-starter', version: '1.3.2'
```

☒ Include comment with link to declaration

반드시 프로젝트 선택 > 우클릭 > Gradle > Refresh Gradle Project로 추가한 라이브러리를 설치합니다.

32.8 타일즈 사용하기

• 32.8.1 회원 기능에 타일즈 적용하기

1. build.gradle에 타일즈 라이브러리를 설정합니다. 설정을 추가한 후 반드시 Gradle을 리프레시하는 것을 잊지 마세요.

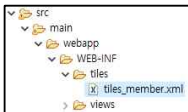
코드 32-19 myboot03/build.gradle

```
...
dependencies {
    implementation('org.springframework.boot:spring-boot-starter-jdbc')
    implementation('org.springframework.boot:spring-boot-starter-web')
    implementation('org.mybatis.spring.boot:mybatis-spring-boot-starter:1.3.2')
    implementation('org.apache.tomcat.embed:tomcat-embed-jasper')
    implementation('javax.servlet:jstl:1.2')
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation files('libs/ojdbc6.jar')
    implementation('org.apache.tiles:tiles-jsp:3.0.4')
    runtimeOnly('com.h2database:h2')
    testImplementation('org.springframework.boot:spring-boot-starter-test')
}
```

↑ 타일즈 기능을 사용하기 위해 추가합니다.

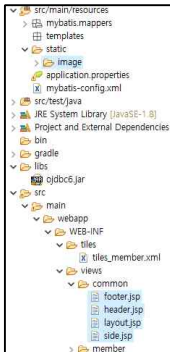
32.8 타일즈 사용하기

2. /WEB-INF 하위에 tiles 폴더를 만든 후 30장에서 사용한 tiles_member.xml을 복사해 붙여 넣습니다.



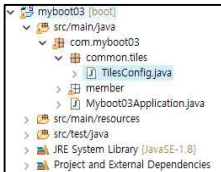
32.8 타일즈 사용하기

3. 타일즈 설정 파일에서 지정한 경로에 30장에서 사용한 공통 레이아웃에 사용하는 JSP 파일들을 복사해 붙여 넣습니다.



32.8 타일즈 사용하기

4. src/main/java/ 폴더 아래에 common.tiles 패키지를 만든 후 타일즈 설정 클래스인 TilesConfig.java를 생성합니다.



자바 클래스에 사용되는 설정 애너테이션

애너테이션	설명
@Configuration	자바 클래스에 적용해 Spring 설정 클래스로 지정합니다.
@Bean	자바 클래스의 메서드에 적용해 Bean을 반환하도록 지정합니다.

32.8 타일즈 사용하기

5. TilesConfig 클래스를 다음과 같이 작성합니다.

코드 32-20 myboot03/src/main/java/com/myboot03/common/TilesConfig.java

```
...
@Configuration  ← 설정 클래스로 지정합니다.
public class TilesConfig{
    @Bean  ← id가 tilesConfigurer인 빈을 생성합니다.
    public TilesConfigurer tilesConfigurer() {  ← 한 개의 타일즈 설정 파일을 읽어 들입니다.
        final TilesConfigurer configurator = new TilesConfigurer();
        configurator.setDefinitions(new String[] { "WEB-INF/tiles/tiles_member.xml" });
        //configurator.setDefinitions(new String[]
        //    { "WEB-INF/tiles/tiles_member.xml", "WEB-INF/tiles/tiles_board.xml" });
        configurator.setCheckRefresh(true);  ← 여러 개의 타일즈 설정 파일을 읽어 들입니다.
        return configurator;
    }
    @Bean  ← id가 tilesViewResolver인 빈을 생성합니다.
    public TilesViewResolver tilesViewResolver() {
        final TilesViewResolver resolver = new TilesViewResolver();
        resolver.setViewClass(TilesView.class);
        return resolver;
    }
}
```

32.8 타일즈 사용하기

첫번째 @Bean

```
<bean id="tilesConfigurer" class="com.myboot03.common.TilesConfigurer"/>
```

두 번째 @Bean

```
<bean id="tilesViewResolver" class="com.myboot03.common.TilesViewResolver"/>
```

32.8 타일즈 사용하기

6. 프로젝트를 실행하고 `http://localhost:8090/member/listMembers.do`로 요청하여 결과를 확인합니다.



스프링실습 홈페이지!!

[로그인](#)

사이드 메뉴

- 회원관리
- 게시판관리
- 상품관리

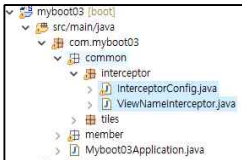
아이디	비밀번호	이름	이메일	가입일	삭제
m2	1234	이길동	m2@test.com	2018-09-30	삭제하기
m1	1234	박길동	m1@test.com	2018-09-30	삭제하기
m3	1234	김길동	m3@test.com	2018-09-30	삭제하기
ki	1234	기성용	ki@test.com	2018-09-13	삭제하기
kim	1212	김유신	kim@jweb.com	2018-09-04	삭제하기
lee	1212	이순신	lee@test.com	2018-09-04	삭제하기
hong	1212	홍길동	hong@gmail.com	2018-09-04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

[회원가입](#)

32.9 인터셉터 기능 사용하기

- 32.9.1 인터셉터 구현하기

1. common.interceptor 패키지를 만들고 30장에서 실습한 ViewNameInterceptor.java를 복사해 붙여 넣습니다.



32.9 인터셉터 기능 사용하기

2. 인터셉트 설정 클래스인 `InterceptorConfig.java`를 다음과 같이 구현합니다

코드 32-21 myboot03/src/main/java/com/myboot03/common/interceptor/InterceptorConfig.java

```
...
@Configuration
public class InterceptorConfig extends WebMvcConfigurerAdapter {
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new ViewNameInterceptor())
            .addPathPatterns("/*.do")
            .addPathPatterns("/*/*.do")
            .excludePathPatterns("/users/login");
    }
}
```

WebMvcConfigurerAdapter를 상속받습니다.

1단계, 2단계 요청에 대해 모두 인터셉터를 적용합니다.

/users/login 요청에 대해서는 인터셉터 요청을 적용하지 않습니다.

32.9 인터셉터 기능 사용하기

3. 이제는 뷰이름을 일일이 지정할 필요 없이 request에서 `getAttribute()` 메서드로 뷰이름을 가져와 바로 타일즈로 전달합니다

코드 32-22 myboot03/src/main/java/com/myboot03/member/controller/MemberControllerImpl.java

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

인터셉터에서 전달된 뷰이름을 가져옵니다.

뷰이름을 직접 지정하지 않아도 됩니다.

32.9 인터셉터 기능 사용하기

4. 프로젝트를 실행하고 <http://localhost:8090/member/listMembers.do>로 요청하여 결과를 확인합니다.



스프링실습 홈페이지!!

[로그인](#)

사이드 메뉴

- 회원관리
- 게시판관리
- 상품관리

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](#)