Refer to https://gist.github.com/ihoneymon/8a905e1dd8393b6b9298
https://www.concretepage.com/spring-boot/spring-boot-thymeleaf-maven-example
https://medium.com/@trevormydata/week-5-thymeleaf-with-spring-mvc-rapid-introduction-to-the-essentials-799f1fba8c07
쇼다 츠야노 저, 김완섭 역, '스프링부터 프로그래밍 입문', (주)도서출판 길벗, 2017:4, 서울

1. 소개
1)Spring Framework는 시간이 지나면서 하위 project가 점점 늘어나서 방대해졌기 때문에 그중에서 무엇을 사용해야 할지 알수 없다는 의견이 많았다.
2)실제로 각 project를 조합해서 사용하려면 초기에 설정할 것도 많고 조합하는 방법 자체에도 knowhow가 필요했다.
3)하지만, Spring MVC를 사용하기 위해서는 필요한 framework들과 library를 정확하게 설정하지 않으면 안된다.
4)또한, 기본적인 처리를 구축하기 위해 MVC의 각 code를 작성해야 한다.
5)즉, 실질적인 프로그래밍에 들어가기 전에 하는 작업이 매우 복잡하다.
6)그래서 좀 더 효율적으로, 최소한의 작업만으로 Spring MVC를 사용한 web application을 개발할 수 없을까 하는 고민에서 탄생한 것.
7)이때 등장한 것인 Spring Boot이다.
8)덕분에 개발자는 적은 양의 application code를 작성해서 바로 실행할 수 있다.
9)실운영 server에서 사용할 수 있는 web application을 최소한의 작업으로 개발할 수 있게 설계됐다.
10)기존의 수 많은 XML 설정 file을 이용하는 Java EE 개발 방식에서 '설정 file을 사용하지 않고 annotation을 사용'하는 방식으로 방향을 바꾸었다.
11)또한 Java EE의 세계에서는 비슷한 처리를 하는 비슷한 code를 반복해서 작성하는 경우가 많았지만 이것을 'code를 작성하지 않고 처리를 구현'하는 방식으로 변경했다.
12)그래서 실제로 Database 관련 처리는 class만 정의하면 method의 정의 없이 구현이 가능하다.
13)결론은 Spring boot는 Spring MVC를 대체하는 것이 아니라 Spring MVC를 좀 더 편하게 사용하도록 만들어 주는 도구이다.
14)Spring Boot는 Dropwizard(https://www.dropwizard.io)에서 영향을 받아 개빌되었다.
15)Spring Boot를 이용하는 application은 Groovy 계열과 Java 계열로 크게 나누어 생각할 수 있다.
-Groovy를 이용한 application
--본격적으로 개발에 들어가기에 앞서 prototype 등을 빠르게 개발해야 하는 경우에 도움
--단시간에 application의 틀을 만들어서 보여주고 그것을 바탕으로 본격적인 개발에 들어가는 것
-Java를 이용한 application
--Maven or Gradle이라는 build tool을 사용하는 project로 생성되며 Java를 사용해 code한다.

2. 기능
1)Create stand-alone Spring applications
2)Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
3)Provide opinionated 'starter' component to simplify your build configuration
4)Automatically configure Spring whenever possible
5)Provide production-ready features such as metrics, health checks and externalized configuration
6)Absolutely no code generation and no requirement for XML configuration

3. Spring Boot 시작하기
1)Maven으로 Template Project 시작하기
2)Groovy로 application 개발하기
3)http://start.spring.io에서 생성하기
4)STS로 project 만들기

4. Lab
1)Template Project 생성하기
-Maven으로 project template 생성하기

```
$ cd SpringHome

$ mvn -B archetype:generate -DgroupId=com.example -DartifactId=springbootdemo
-Dversion=1.0.0-SNAPSHOT -DarchetypeArtifactId=maven-archetype-quickstart
[INFO] Scanning for projects...
...

[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
maven-archetype-quickstart:1.0
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: C:\SpringHome
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: springbootdemo
[INFO] Parameter: packageName, Value: com.example
```

```
[INFO] Parameter: version, Value: 1.0.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\SpringHome\springbootdemo
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 08:37 min
[INFO] Finished at: 2018-06-24T22:23:10+09:00
[INFO] ------------------------------------------------------------------------
ESC[0mESC[0m
```

-springbootdemo folder로 이동

```
$ cd springbootdemo
   |
   |----src
          |
          |-----main
          |     |
          |     |------java
          |             |
          |             |------com
          |                     |
          |                     |-----example
          |                             |
          |                             |------App.java
          |
          |------test
```

2)pom.xml 설정하기

```
<?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.example</groupId>
   <artifactId>springbootdemo</artifactId>
   <packaging>jar</packaging>
   <version>1.0.0-SNAPSHOT</version>
   <name>springbootdemo</name>
   <url>http://maven.apache.org</url>

   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>1.5.14.RELEASE</version>
   </parent>

   <dependencies>
      <dependency>
         <groupId>junit</groupId>
         <artifactId>junit</artifactId>
         <version>4.12</version>
         <scope>test</scope>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-web</artifactId>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-test</artifactId>
         <scope>test</scope>
      </dependency>
   </dependencies>

```

```
    <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
    </build>

    <properties>
      <java.version>1.8</java.version>
    </properties>
    </project>
```

-<parent> 설정하기
  --Spring Boot의 설정 정보를 상속한다.
  -- 여기서 지정한 version이 spring boot의 version이 된다.
  --spring boot의 version을 올리려면 <version> tag 안에 있는 설정 값을 변경한다.

-spring-boot-starter-web
  --spring boot로 web application을 만들 때 참조할 기본 library 정보를 설정한다.
  --이렇게 쓰기만 해도 web application 제작에 필요한 spring framework 관련 library와 third-party library를 이용할 수 있게 된다.
  --version은 위 parent에서 설정한 spring-boot-starter-parent 안에 정의되어 있으므로, 여기서는 지정하지 않아도 된다.

3)사용가능한 libary 확인

```
$ mvn dependency:tree
  ...
[INFO] com.example:springbootdemo:jar:1.0.0-SNAPSHOT
[INFO] +- junit:junit:jar:3.8.1:test
[INFO] +- org.springframework.boot:spring-boot-starter-web:jar:1.5.14.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter:jar:1.5.14.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot:jar:1.5.14.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-autoconfigure:jar:1.5.14.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-starter-logging:jar:1.5.14.RELEASE:compile
[INFO] |  |  |  +- ch.qos.logback:logback-classic:jar:1.1.11:compile
[INFO] |  |  |  |  \- ch.qos.logback:logback-core:jar:1.1.11:compile
[INFO] |  |  |  +- org.slf4j:jcl-over-slf4j:jar:1.7.25:compile
[INFO] |  |  |  +- org.slf4j:jul-to-slf4j:jar:1.7.25:compile
[INFO] |  |  |  \- org.slf4j:log4j-over-slf4j:jar:1.7.25:compile
[INFO] |  |  \- org.yaml:snakeyaml:jar:1.17:runtime
[INFO] |  +- org.springframework.boot:spring-boot-starter-tomcat:jar:1.5.14.RELEASE:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-core:jar:8.5.31:compile
[INFO] |  |  |  \- org.apache.tomcat:tomcat-annotations-api:jar:8.5.31:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-el:jar:8.5.31:compile
[INFO] |  |  \- org.apache.tomcat.embed:tomcat-embed-websocket:jar:8.5.31:compile
[INFO] |  +- org.hibernate:hibernate-validator:jar:5.3.6.Final:compile
[INFO] |  |  +- javax.validation:validation-api:jar:1.1.0.Final:compile
[INFO] |  |  +- org.jboss.logging:jboss-logging:jar:3.3.2.Final:compile
[INFO] |  |  \- com.fasterxml:classmate:jar:1.3.4:compile
[INFO] |  +- com.fasterxml.jackson.core:jackson-databind:jar:2.8.11.2:compile
[INFO] |  |  +- com.fasterxml.jackson.core:jackson-annotations:jar:2.8.0:compile
[INFO] |  |  \- com.fasterxml.jackson.core:jackson-core:jar:2.8.11:compile
[INFO] |  +- org.springframework:spring-web:jar:4.3.18.RELEASE:compile
[INFO] |  |  +- org.springframework:spring-aop:jar:4.3.18.RELEASE:compile
[INFO] |  |  +- org.springframework:spring-beans:jar:4.3.18.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-context:jar:4.3.18.RELEASE:compile
[INFO] |  \- org.springframework:spring-webmvc:jar:4.3.18.RELEASE:compile
[INFO] |     \- org.springframework:spring-expression:jar:4.3.18.RELEASE:compile
[INFO] \- org.springframework.boot:spring-boot-starter-test:jar:1.5.14.RELEASE:test
[INFO]    +- org.springframework.boot:spring-boot-test:jar:1.5.14.RELEASE:test
[INFO]    +- org.springframework.boot:spring-boot-test-autoconfigure:jar:1.5.14.RELEASE:test
[INFO]    +- com.jayway.jsonpath:json-path:jar:2.2.0:test
[INFO]    |  +- net.minidev:json-smart:jar:2.2.1:test
[INFO]    |  |  \- net.minidev:accessors-smart:jar:1.1:test
```

```
[INFO]    | |     \- org.ow2.asm:asm:jar:5.0.3:test
[INFO]    | \- org.slf4j:slf4j-api:jar:1.7.25:compile
[INFO]    +- org.assertj:assertj-core:jar:2.6.0:test
[INFO]    +- org.mockito:mockito-core:jar:1.10.19:test
[INFO]    | \- org.objenesis:objenesis:jar:2.1:test
[INFO]    +- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO]    +- org.hamcrest:hamcrest-library:jar:1.3:test
[INFO]    +- org.skyscreamer:jsonassert:jar:1.4.0:test
[INFO]    | \- com.vaadin.external.google:android-json:jar:0.0.20131108.vaadin1:test
[INFO]    +- org.springframework:spring-core:jar:4.3.18.RELEASE:compile
[INFO]    \- org.springframework:spring-test:jar:4.3.18.RELEASE:test
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 10:09 min
[INFO] Finished at: 2018-06-24T23:07:52+09:00
[INFO] ------------------------------------------------------------------------
```

4)Hello World!를 출력하는 Web application 작성하기

-src/main/java/com/example/App.java

```
package com.example;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@EnableAutoConfiguration
public class App {

   @RequestMapping("/")
   String home(){
      return "Hello, World!";
   }

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

-@RestController

--이 annotation을 붙이면 web appication에서 request을 받아들이는 controller class임을 나타낸다.

-@EnableAutoConfiguration

--이 annotation은 매우 중요하다.

--이 annotation을 붙이면 다양한 설정이 자동으로 수행되고 기존의 spring application에 필요했던 설정 file들이 필요없게 된다.

-@RequestMapping("/")

--이 annotation이 붙으면 이 method가 HTTP 요청을 받아들이는 method임을 나타낸다.

-return "Hello World!";

--HTTP 응답을 반환한다.

--@RestController annotation이 붙은 class에 속한 method에서 문자열을 반환하면 해당 문자열이 그대로 HTTP 응답이 되어 출력된다.

-SpringApplication.run(App.clas, args);

--spring boot applicaton을 실행하는 데 필요한 처리를 main() 안에서 작성한다.

--@EnableAutoConfiguration annotation이 붙은 class를 SpringApplication.run()의 첫번째 인자로 지정한다.

5)Web Application 실행하기

```
$ mvn spring-boot:run
   ...
```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::       (v1.5.14.RELEASE)
   ...
   ...
2018-06-24 23:19:10.688  INFO 31008 --- [           main] com.example.App                          :
Started App in 2.191 seconds (JVM running for 70.248)
```

-출력된 log 내용을 보면 8080 port로 tomcat이 시작된다는 것을 알 수 있다.
-SpringApplication.run() method에서 내장 server를 시작했기 때문이다.
-http://localhost:8080/로 접속해보자.
-Web browser에 'Hello, World!'가 출력된다.
-application을 끝내려면 Ctrl + C를 누르고, '[일괄 작업을 끝내시겠습니까 (Y/N)?'라는 질문에 'y'를 입력하고 enter key를 누르면 된다.
-여기서 알게 된 사실
  --설정할 의존 관계의 갯수가 적다.
  --Java Class 하나만 작성하면 된다.
  --명령 prompt에서 application을 실행한다.

6)실행 가능한 jar file 작성하기

```
$ mvn package
```

-target folder 안에 springbootdemo-1.0.0-SNAPSHOT.jar file이 생성되어 있다.
-실행해 보자.

```
$ cd target
$ java -jar springbootdemo-1.0.0-SNAPSHOT.jar
```

-위와 동일하게 실행된다.
-이 jar file만 있으면 application 배포가 가능하다.
-HTTP Port도 변경할 수 있다.
  --실행할 때 아래의 option을 붙이면 된다.

    --server.port=PORT 번호

7)STS로 spring boot application 개발하기
-File > Import > Maven > Existing Maven Projects > Next
-Click [Browse...] > Select [springbootdemo] > Finish
-src/main/java/com.example.App.java > right-click > Run As > Spring Boot App

8)STS로 spring boot project 생성하기
-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : demo
-Type : Maven
-Packaging : jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : demo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.demo

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select Web > check Web > Finish

-src/main/java/com.example.demo.DemoApplication.java 수정하기

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@EnableAutoConfiguration
public class DemoApplication {

   @RequestMapping("/")
   String home() {
      return "Hello, World!";
   }

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

-DemoApplication.java > right-click > Run As > Spring Boot App
-http://localhost:8080/

9)Spring loaded로 간편하게 개발하기
-spring loaded를 사용하면 Java code를 수정했을 때 application을 실행한 상태에서 수정한 code의 내용을 application에 반영할 수 있다.
-이러한 기능을 'Hot Reloading'이라고 부른다.
-spring loaded를 사용하기 위해 pom.xml에 code를 추가한다.

```
<build>
   <plugins>
      <plugin>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-maven-plugin</artifactId>
         <!-- 여기부터 시작  -->
         <dependencies>
            <dependency>
               <groupId>org.springframework</groupId>
               <artifactId>springloaded</artifactId>
               <version>1.2.3.RELEASE</version>
            </dependency>
         </dependencies>
         <!-- 여기까지 -->
      </plugin>
   </plugins>
</build>
```

-command prompt에서

```
$ cd demo
$ mvn spring-boot:run
```

-http://localhost:8080/

```
Hello, World!
```

-DemoApplication.java 수정

```
String home() {
   return "Hello Spring Boot!";
}
```

-Save 하고 다시 http://localhost:8080으로 가면 자동으로 변경되어

```
Hello Spring Boot!";
```

5. Groovy로 application 개발하기

1)준비
-Visit https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html
-10.2.1 Manual Installation에서 spring-boot-cli-2.0.3.RELEASE-bin.zip link를 click한다.
-하지만 여기서는 1.5.3을 download하기 위해 https://repo.spring.io/release/org/springframework/boot/spring-boot-cli/1.5.3.RELEASE/를 방문한다.
-Download spring-boot-cli-1.5.3.RELEASE-bin.zip
-Unzip > Move to C:\Program Files\spring-1.5.3.RELEASE
-path 설정
--%PATH%;C:\Program Files\spring-1.5.3.RELEASE\bin

2)Groovy Script 작성하기
-Editor를 열어서 아래의 code를 적당한 위치(즉 C:\temp)에 file 이름은 app.groovy라고 저장한다.

--app.groovy

```
@RestController
class App {

    @RequestMapping("/")
    def home() {
        "Hello!!!"
    }
}
```

3)app.groovy 실행하기
-Command Prompt에서,
$ cd C:\temp
$ spring run app.groovy

Resolving dependencies.....

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v1.5.3.RELEASE)
```

2018-06-25 23:36:32.093 INFO 27432 --- [runner-0] o.s.boot.SpringApplication : Starting application on DESKTOP-DEVEXPERT with PID 27432 (started by Instructor in C:\Temp)
2018-06-25 23:36:32.101 INFO 27432 --- [runner-0] o.s.boot.SpringApplication : No active profile set, falling back to default profiles: default
2018-06-25 23:36:32.308 INFO 27432 --- [runner-0] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@1c388377: startup date [Mon Jun 25 23:36:32 KST 2018]; root of context hierarchy
2018-06-25 23:36:33.337 INFO 27432 --- [runner-0] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2018-06-25 23:36:33.348 INFO 27432 --- [runner-0] o.apache.catalina.core.StandardService : Starting service Tomcat
2018-06-25 23:36:33.349 INFO 27432 --- [runner-0] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.14
2018-06-25 23:36:33.720 INFO 27432 --- [ost-startStop-1] org.apache.catalina.loader.WebappLoader : Unknown loader org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader@3751af9 class org.springframework.boot.cli.compiler.ExtendedGroovyClassLoader$DefaultScopeParentClassLoader
2018-06-25 23:36:33.742 INFO 27432 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2018-06-25 23:36:33.742 INFO 27432 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed

```
in 1434 ms
2018-06-25 23:36:33.873  INFO 27432 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Mapping servlet: 'dispatcherServlet' to [/]
2018-06-25 23:36:33.879  INFO 27432 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2018-06-25 23:36:33.880  INFO 27432 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2018-06-25 23:36:33.880  INFO 27432 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2018-06-25 23:36:33.880  INFO 27432 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2018-06-25 23:36:34.145  INFO 27432 --- [      runner-0] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@1c388377: startup date [Mon Jun 25 23:36:32 KST 2018]; root of context hierarchy
2018-06-25 23:36:34.213  INFO 27432 --- [      runner-0] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/]}" onto public java.lang.Object App.home()
2018-06-25 23:36:34.217  INFO 27432 --- [      runner-0] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-06-25 23:36:34.218  INFO 27432 --- [      runner-0] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2018-06-25 23:36:34.256  INFO 27432 --- [      runner-0] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-25 23:36:34.256  INFO 27432 --- [      runner-0] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-25 23:36:34.305  INFO 27432 --- [      runner-0] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2018-06-25 23:36:34.683  INFO 27432 --- [      runner-0] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-06-25 23:36:34.745  INFO 27432 --- [      runner-0] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2018-06-25 23:36:34.750  INFO 27432 --- [      runner-0] o.s.boot.SpringApplication               : Started application in 3.006 seconds (JVM running for 37.809)
```

4)Web browser로 http://localhost:8080에 접속한다.

Hello!!!

5)app.groovy code 수정하기

-Command Prompt 에서 Ctrl + C를 눌러서 종료시킨다.

-일괄 작업을 끝내시겠습니까 (Y/N)? Y

-아래와 같이 code를 수정한다.

```
@RestController
class App {

    @RequestMapping("/")
    def home() {
        def header = "<html><body>"
    def footer = "</body></html>"
    def content = "<h1>Hello! Spring Boot with Groovy</h1><p>This is html content.</p>"

    header + content + footer
    }
}
```

-다시 script를 실행한다.

```
$ spring run app.groovy
```

-Browser를 refresh 한다.

```
Hello! Spring Boot with Groovy
This is html content.
```

6)Template 사용하기

-template은 HTML을 기반으로 작성된 code를 읽어 rendering해서 web page에 출력하는 기능이다.

-이런 기능의 template이 몇 가지 종류가 있지만, spring boot에서는 thymeleaf(타임리프)라고 하는 library를 자주 사용한다.

-http://www.thymeleaf.org

-template file 작성

--C:\temp\templates\home.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />  <!--반드시 종결 tag 필요 -->
    <title>Index Page</title>
    <style type="text/css">
      h1 { font-size:18pt; font-weight:bold; color:gray; }
      body { font-size:13pt; color:gray; margin:5px 25px; }
    </style>
  </head>
  <body>
    <h1>Hello! Spring Boot with Thymeleaf</h1>
    <p>This is sample web page.</p>
  </body>
</html>
```

-template file은 controller가 있는 곳의 templates folder 안에 두어야 한다.

-controller 수정하기

--app.groovy

```
@Grab("thymeleaf-spring4")

@Controller
class App {

    @RequestMapping("/")
    @ResponseBody
    def home(ModelAndView mav) {
        mav.setViewName("home")
    mav
    }
}
```

-다시 script 실행

```
$ spring run app.groovy
```

7)form 전송하기

-home.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Index Page</title>
    <style type="text/css">
```

```
                h1 { font-size:18pt; font-weight:bold; color:gray; }
                body { font-size:13pt; color:gray; margin:5px 25px; }
            </style>
        </head>
        <body>
            <h1>Hello!</h1>
            <p th:text="${msg}">${msg}</p>
            <form method="post" action="/send">
                <input type="text" name="text1" th:value="${value}" />
                <input type="submit" value="Send" />
            </form>
        </body>
    </html>
```

-app.groovy

```
    @Grab("thymeleaf-spring4")

    @Controller
    class App {

        @RequestMapping(value = "/", method=RequestMethod.GET)
        @ResponseBody
        def home(ModelAndView mav) {
            mav.setViewName("home")
            mav.addObject("msg", "Please write your name...")
            mav
        }

        @RequestMapping(value = "/send", method=RequestMethod.POST)
        @ResponseBody
        def send(@RequestParam("text1") String str, ModelAndView mav){
            mav.setViewName("home")
            mav.addObject("msg", "Hello, " + str + "!!!")
            mav.addObject("value", str)
            mav
        }
    }
```

-script 실행

6. SPRING INITIALIZR : http://start.spring.io에서 생성하기, Maven
   1)Visit http://start.spring.io/
   2)설정
   -Maven Project
   -Java
   -1.5.15 (SNAPSHOT)
   -Click [Switch to the full version]
   -Group : com.example
   -Artifact : demo
   -Name : demo
   -Description : Demo project for Spring Boot
   -Package Name : com.example.demo
   -Packaging : Jar
   -Java Version : 8
   -dependencies : Web

   3)Click [Generate Project]
   4)Downloads [demo.zip] : 48.7KB
   5)Unpack to Spring workspace.
   6)In STS, Package Explorer > right-click > Import > Maven > Existing Maven Projects > Next
   7)Click [Browse...] > demo folder > Finish
   8)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"
602     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
603     <modelVersion>4.0.0</modelVersion>
604
605     <groupId>com.example</groupId>
606     <artifactId>demo</artifactId>
607     <version>0.0.1-SNAPSHOT</version>
608     <packaging>jar</packaging>
609
610     <name>demo</name>
611     <description>Demo project for Spring Boot</description>
612
613     <parent>
614        <groupId>org.springframework.boot</groupId>
615        <artifactId>spring-boot-starter-parent</artifactId>
616        <version>2.0.3.RELEASE</version>
617        <relativePath/> <!-- lookup parent from repository -->
618     </parent>
619
620     <properties>
621        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
622        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
623        <java.version>1.8</java.version>
624     </properties>
625
626     <dependencies>
627        <dependency>
628           <groupId>org.springframework.boot</groupId>
629           <artifactId>spring-boot-starter-web</artifactId>
630        </dependency>
631
632        <dependency>
633           <groupId>org.springframework.boot</groupId>
634           <artifactId>spring-boot-starter-test</artifactId>
635           <scope>test</scope>
636        </dependency>
637     </dependencies>
638
639     <build>
640        <plugins>
641           <plugin>
642              <groupId>org.springframework.boot</groupId>
643              <artifactId>spring-boot-maven-plugin</artifactId>
644           </plugin>
645        </plugins>
646     </build>
647
648  </project>
649
650 9)src/test/java/com.example.demo.DemoApplicationTests.java > right-click > Run As > JUnit Test
    > Green bar
651 10)demp project > right-click > Run As > Spring Boot App
652 11)http://localhost:8080/
653
654    Whitelabel Error Page
655    This application has no explicit mapping for /error, so you are seeing this as a fallback.
656
657    Tue Jun 26 23:14:28 KST 2018
658    There was an unexpected error (type=Not Found, status=404).
659    No message available
660
661 12)Controller 생성
662    -src/main/java/com.example.demo > right-click > New > Class
663    -Name : HelloController
664
665       package com.example.demo;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

    @GetMapping("/")
    public String hello() {
        return "Hello, Spring Boot World";
    }
}
```

13)Relaunch demo
14)http://localhost:8080/

7. SPRING INITIALIZR : http://start.spring.io에서 생성하기, Gradle
   1)Visit http://start.spring.io/
   2)설정
      -Gradle Project
      -Java
      -1.5.15 (SNAPSHOT)
      -Click [Switch to the full version]
      -Group : com.example
      -Artifact : demoweb
      -Name : demoweb
      -Description : Demo project for Spring Boot
      -Package Name : com.example.demoweb
      -Packaging : Jar
      -Java Version : 8
      -dependencies : Core > DevTools, Web > Web

   3)Click [Generate Project]
   4)Downloads [demoweb.zip] : 55.6KB
   5)Unpack to Spring workspace.
   6)In STS, Package Explorer > right-click > Import > Gradle > Existing Gradle Project > Next > Next
   7)Click [Browse...] > demoweb folder > Next > Finish
   8)build.gradle

```
buildscript {
    ext {
        springBootVersion = '1.5.15.BUILD-SNAPSHOT'
    }
    repositories {
        mavenCentral()
        maven { url "https://repo.spring.io/snapshot" }
        maven { url "https://repo.spring.io/milestone" }
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'org.springframework.boot'

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
    mavenCentral()
    maven { url "https://repo.spring.io/snapshot" }
    maven { url "https://repo.spring.io/milestone" }
}
```

```
    dependencies {
        compile('org.springframework.boot:spring-boot-starter-web')
        runtime('org.springframework.boot:spring-boot-devtools')
        testCompile('org.springframework.boot:spring-boot-starter-test')
    }
```

9)src/test/java/com.example.demoweb.DemowebApplicationTests.java > right-click > Run As > JUnit Test > Green bar
9)demp project > right-click > Run As > Spring Boot App
10)http://localhost:8080/

```
    Whitelabel Error Page
    This application has no explicit mapping for /error, so you are seeing this as a fallback.

    Tue Jun 26 23:14:28 KST 2018
    There was an unexpected error (type=Not Found, status=404).
    No message available
```

11)Controller 생성
-src/main/java/com.example.demoweb > right-click > New > Class
-Name : HomeController

```
    package com.example.demo;

    import org.springframework.web.bind.annotation.GetMapping;
    import org.springframework.web.bind.annotation.RestController;

    @RestController
    public class HelloController {

        @GetMapping("/")
        public String home() {
            return "Hello, Spring Boot World";
        }
    }
```

12)Relaunch demo
13)http://localhost:8080/

8. STS로 Spring Boot project : 정적 page

1)Spring Boot project 생성
-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : springweb
-Type : Maven
-Packaging : jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : springweb
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select Web > check Core > DevTools, Web > Web > Next > Finish

2)Controller 생성
-src/main/java/com.example.biz > right-click > New > Class
-Name : HomeController

```
    package com.example.biz;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HomeController {

   @GetMapping("/")
   public ModelAndView home(ModelAndView mav) {
      mav.setViewName("index.html");
      return mav;
   }
}
```

3)static file 생성
-src/main/resources/static/images folder 생성
--spring-boot.png 추가할 것
-src/main/resources/static/js folder 생성
--jquery-3.3.1.slim.min.js 추가할 것
-src/main/resources/static/css folder 생성
--bootstrap.min.css 추가할 것
-src/main/resources/static/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Home page</title>
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css" />
<script src="js/jquery-3.3.1.slim.min.js"></script>
<script>
   $(document).ready(function(){
      alert("Hello, Spring Boot World!!!");
   });
</script>
</head>
<body>
   <div>
      <img src="images/spring-boot.png" />
   </div>
   <div class="jumbotron">
     <h1>Hello, Spring Boot World</h1>
     <p>...</p>
     <p><a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a></p>
   </div>
</body>
</html>
```

4)Spring Boot에서는 template을 사용하지 않을 경우 기본적으로 static file은 src/main/resources/static에서 찾는다.
5)이럴 때는 반드시 file의 확장자 .html까지 넣어야 한다.

9. STS로 Spring Boot project : JSP
1)Spring Boot에서는 JSP 사용을 권장하지 않는다.
2)'jar' 형식으로 동작하지 않고 War file로 배포해야 하는 등의 몇 가지 제약이 있어서이기도 하지만, 가장 큰 이유는 이미 JSP 자체가 Server 측 언어로 그 사용 빈도가 줄고 있기 때문이다.
3)view 부분에 code가 섞여서 logic을 분리하기 어렵고, HTML과 같은 tag를 사용하므로 HTML 편집기 등에서 JSP 삽입 부분을 분리하기 어려우며 Visual 편집기 등에서도 사용이 어렵다.
4)그래서 template을 통해 code를 분리해야 할 필요가 있는 것이다.
5)Spring Boot project 생성
-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : springwebdemo
-Type : Maven
-Packaging : jar
-Java Version : 8

-Language : Java
-Group : com.example
-Artifact : springwebdemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select Web > check Web > Finish

6)pom.xml
-jstl을 위해

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

-JSP를 위해

```
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jasper</artifactId>
    <version>8.5.32</version>
</dependency>
```

7)folder 준비
-src/main folder 안에 webapp folder 생성
-webapp folder 안에 WEB-INF folder 생성
-WEB-INF folder 안에 jsp folder 생성

8)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.
9)src/mian/resources/application.properties code 추가

spring.mvc.view.prefix : /WEB-INF/jsp/
spring.mvc.view.suffix : .jsp

10)jsp folder 안에 jsp file 생성
-index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.Date, java.text.SimpleDateFormat" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Index page</h1>
    <%=new SimpleDateFormat("yyyy년 MM월 dd일").format(new Date()) %>
</body>
</html>
```

11)com.example.biz.HomeController.java

```
package com.example.biz;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
    public String index() {
```

```
            return "index";
        }
    }
```

12)http://localhost:8080/

Index page
2018년 06월 27일

10. Lab : thymeleaf template 사용하기

1)Spring Boot project 생성

-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : MyBootApp
-Type : Maven
-Packaging : jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : MyBootApp
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select Web > check Web > Finish

2)src/main/java/com.example.biz.MyBootAppApplication.java 수정하기

```
package com.example.biz;

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

@SpringBootApplication
public class MyBootAppApplication {

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

3)실행

-MyBootAppApplication.java > right-click > Run As > Spring Boot App
-http://localhost:8080/

Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Jun 26 09:42:08 KST 2018
There was an unexpected error (type=Not Found, status=404).
No message available

4)pom.xml

-parent
--상속에 관한 설정 정보이다.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.15.BUILD-SNAPSHOT</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

-properties

```
        <properties>
            <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
            <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
            <java.version>1.8</java.version>
        </properties>


    -dependencies

        <dependencies>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>

            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
            </dependency>
        </dependencies>

5)Controller 작성하기
    -com.example.biz > right-click > New > Class
    -Name : HelloController > Finish
    -HelloController.java

        package com.example.biz;

        import org.springframework.web.bind.annotation.RequestMapping;
        import org.springframework.web.bind.annotation.RestController;

        @RestController
        public class HelloController {

            @RequestMapping("/")
            public String index() {
                return "Hello Spring Boot World!";
            }
        }

6)project > right-click > Run As > Spring Boot App
7)http://localhost:8080/

    Hello Spring Boot World!

8)매개변수 전달
    -HelloController.java 수정

        @RequestMapping("/{num}")
        public String index(@PathVariable int num) {
            int result = 0;
            for(int i = 1 ; i <= num ; i++)  result += i;
            return "total : " + result;
        }

9)project > right-click > Run As > Spring Boot App
10)http://localhost:8080/100

    total : 5050

10)객체를 JSON으로 출력하기
    -src/main/java/com.example.biz/Student.java 생성

        package com.example.biz;
```

```
public class Student {
   private int userid;
   private String name;
   private int age;
   private String address;
   public Student(int userid, String name, int age, String address) {
      this.userid = userid;
      this.name = name;
      this.age = age;
      this.address = address;
   }
   public int getUserid() {
      return userid;
   }
   public void setUserid(int userid) {
      this.userid = userid;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String getAddress() {
      return address;
   }
   public void setAddress(String address) {
      this.address = address;
   }
}
```

11)HelloController.java 수정

```
@RestController
public class HelloController {
   String [] names = {"조용필", "이미자", "설운도"};
   int [] ages = {56, 60, 70};
   String [] addresses = {"서울특별시", "부산광역시", "대전광역시"};

   @RequestMapping("/{userid}")
   public Student index(@PathVariable int userid) {
      return new Student(userid, names[userid], ages[userid], addresses[userid]);
   }
}
```

12)http://localhost:8080/1

```
{"userid":1,"name":"이미자","age":60,"address":"부산광역시"}
```

13)thymeleaf 추가하기

-pom.xml 수정하기

-Select Dependencies tab > Add

-Group Id : org.springframework.boot

-Artifact Id : spring-boot-starter-thymeleaf

-Version :

-Scope : compile

-OK

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
```

```
    </dependency>
```

14)HelloController.java 수정

```
package com.example.biz;

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

@Controller
public class HelloController {

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

15)template file 생성

-src/main/resources/templates > right-click > New > Other...> Web > HTML File > Next

-File name : index.html

```
<!doctype html>
<html lang="en">
   <head>
      <meta charset="UTF-8" />  <!--반드시 종결 tag 필요 -->
      <title>Index Page</title>
      <style type="text/css">
         h1 { font-size:18pt; font-weight:bold; color:gray; }
         body { font-size:13pt; color:gray; margin:5px 25px; }
      </style>
   </head>
   <body>
      <h1>Hello! Spring Boot with Thymeleaf</h1>
      <p>This is sample web page.</p>
   </body>
</html>
```

16)http://localhost:8080/

```
Hello! Spring Boot with Thymeleaf
This is sample web page.
```

17)template에 값 표시하기

-index.html code 수정

```
<body>
   <h1>Hello! Spring Boot with Thymeleaf</h1>
   <p class="msg" th:text="${msg}"></p>
</body>
```

-HelloController.java 수정

```
@Controller
public class HelloController {

   @RequestMapping("/{num}")
   public String index(@PathVariable int num, Model model) {
      int result = 0;
      for(int i = 1 ; i <= num ; i++) result += i;
      model.addAttribute("msg", "total : " + result);
      return "index";
   }
}
```

18)http://localhost:8080/100

```
Hello! Spring Boot with Thymeleaf
```

total : 5050

19)ModelAndView class 사용하기
-HelloController.java 수정

```
@Controller
public class HelloController {

   @RequestMapping("/{num}")
   public ModelAndView index(@PathVariable int num, ModelAndView mav) {
      int result = 0;
      for(int i = 1 ; i <= num ; i++) result += i;
      mav.addObject("msg", "total : " + result);
      mav.setViewName("index");
      return mav;
   }
}
```

20)http://localhost:8080/100

Hello! Spring Boot with Thymeleaf
total : 5050

21)form 사용하기
-index.html 수정

```
<!doctype html>
<html lang="en">
   <head>
      <meta charset="UTF-8" />
      <title>Index Page</title>
      <style type="text/css">
         h1 { font-size:18pt; font-weight:bold; color:gray; }
         body { font-size:13pt; color:gray; margin:5px 25px; }
      </style>
   </head>
   <body>
      <h1>Hello!</h1>
      <p th:text="${msg}">${msg}</p>
      <form method="post" action="/send">
         <input type="text" name="text1" th:value="${value}" />
         <input type="submit" value="Send" />
      </form>
   </body>
</html>
```

-HelloController.java 수정

```
@Controller
public class HelloController {

   @RequestMapping(value = "/", method = RequestMethod.GET)
   public ModelAndView index(ModelAndView mav) {
      mav.setViewName("index");
      mav.addObject("msg", "Please write your name...");
      return mav;
   }

   @RequestMapping(value = "/send", method = RequestMethod.POST)
   public ModelAndView send(@RequestParam("txtName") String name,
                            ModelAndView mav) {
      mav.addObject("msg", "안녕하세요! " + name + "님!");
      mav.addObject("value", name);
      mav.setViewName("index");
      return mav;
   }
```

```
        }

22)http://localhost:8080

23)기타 form controller
   -index.html 수정

        <!doctype html>
        <html lang="en">
        <head>
        <meta charset="UTF-8" />
        <title>Index Page</title>
        <style type="text/css">
        h1 {
           font-size: 18pt;
           font-weight: bold;
           color: gray;
        }

        body {
           font-size: 13pt;
           color: gray;
           margin: 5px 25px;
        }
        </style>
        </head>
        <body>
           <h1>Hello!</h1>
           <pre th:text="${msg}">Please wait...</pre>
           <form method="post" action="/">
              <div>
                 <input type="checkbox" id="ckeck1" name="check1" /> <label
                    for="ckeck1">체크</label>
              </div>
              <div>
                 <input type="radio" id="male" name="gender" value="male" /> <label
                    for="male">남성</label>
              </div>
              <div>
                 <input type="radio" id="female" name="gender" value="female" /> <label
                    for="female">여성</label>
              </div>
              <div>
              <select name="selOs" size="4">
                 <option>--선택--</option>
                 <option value="Windows">windows</option>
                 <option value="MacOS">MacOS</option>
                 <option value="Linux">Linux</option>
              </select>
              <select name="selEditors" size="4" multiple="multiple">
                 <option>--선택--</option>
                 <option value="Notepad">Notepad</option>
                 <option value="Editplus">Editplus</option>
                 <option value="Visual Studio Code">Visual Studio Code</option>
                 <option value="Sublime Text">Sublime Text</option>
                 <option value="Eclipse">Eclipse</option>
              </select>
              </div>
              <input type="submit" value="전송" />
           </form>
        </body>
        </html>

24)HelloController.java 수정

     @Controller
     public class HelloController {
```

```
1332
1333        @RequestMapping(value = "/", method = RequestMethod.GET)
1334        public ModelAndView index(ModelAndView mav) {
1335           mav.setViewName("index");
1336           mav.addObject("msg", "값을 입력후 전송버튼을 눌러주세요.");
1337           return mav;
1338        }
1339
1340        @RequestMapping(value = "/", method = RequestMethod.POST)
1341        public ModelAndView send(@RequestParam(value="check1", required=false) boolean check1,
1342                                 @RequestParam(value="gender", required=false) String gender,
1343                                 @RequestParam(value="selOs", required=false) String selOs,
1344                                 @RequestParam(value="selEditors", required=false) String []
                                 selEditors,
1345                                 ModelAndView mav) {
1346
1347           String result = "";
1348           try {
1349              result = "check : " + check1 +
1350                       ", gender : " + gender +
1351                       ", OS : " + selOs +
1352                       "\nEditors : ";
1353           }catch(NullPointerException ex) {}
1354           try {
1355              result += selEditors[0];
1356              for(int i = 1 ; i < selEditors.length ; i++) result += ", " + selEditors[i];
1357           }catch(NullPointerException ex) {
1358              result += "null";
1359           }
1360           mav.addObject("msg", result);
1361           mav.setViewName("index");
1362           return mav;
1363        }
1364     }
1365
1366  25)http://localhost:8080
1367
1368  26)Redirect
1369     -index.html 수정
1370
1371        <body>
1372           <h1>Hello! index.</h1>
1373        </body>
1374
1375     -HelloController.java 수정
1376
1377        @Controller
1378        public class HelloController {
1379
1380           @RequestMapping("/")
1381           public ModelAndView index(ModelAndView mav) {
1382              mav.setViewName("index");
1383              return mav;
1384           }
1385
1386           @RequestMapping("/other")
1387           public String other() {
1388              return "redirect:/";
1389           }
1390
1391           @RequestMapping("/home")
1392           public String home() {
1393              return "forward:/";
1394           }
1395        }
1396
1397     -redirect와 forward 차이점 구분하기
```

11. Templates

1)Thymeleaf

-th:00 형식으로 속성을 HTML tag에 추가해서 값이나 처리 등을 page에 심을 수 있다.
-자체 tag를 사용하지 않으므로 HTML visual editor와도 궁합이 좋으며 편집이 쉽다는 장점이 있다.
-Spring Boot에서는 이 template을 선택하는 사람이 가장 많다고 한다.
-기본이 되는 HTML 위에 필요한 속성을 추가만 하면 되기 때문에 간단하고 이해하기 쉬운 구성으로 되어 있다.
-따라서 template 관련 새로운 기술을 배우지 않고서도 바로 사용할 수 있어 학습에 대한 부담감이 줄어든다.

2)JSP

-Spring Boot에서는 기본적으로 사용하지 않는다.
-Spring Boot에서는 application을 Java Server와 함께 Jar file로 묶어 배포하는 경우가 많지만 이 방법에서는 JSP가 동작하지 않는다.
-옛날 방식인, 이미 Java Server가 실행되고 있고 그 위에 War file을 upload하는 경우에만 동작한다.
-최근 cloud를 사용해서 app을 배포하는 경우가 늘고 있는데, 이 경우에는 jar file을 이용하기 때문에 JSP를 사용할 수 없는 경우가 늘고 있다.

3)FreeMarker

-이것은 ${}형식으로 값을 채워나가는 template이다.
-Thymeleaf가 특수한 속성으로 값을 선정하는 것에 비해 이 template은 text를 표시할 위치에 직접 ${}를 넣어야 표시할 수 있다.
-또한 제어를 위해서 HTML tag와 같은 <#OO>형식으로 tag를 사용하므로 HTML Visual Editor에서는 HTML 구조에 영향을 끼칠 가능성이 있다.

4)Groovy

-다른 template과 달리 HTML 기반으로 하지 않고 HTML과는 전혀 다른 code 체계를 가지고 있기 때문에 HTML밖에 모르는 개발자의 입장에서는 적합하지 않다.
-code를 이용하여 화면 전체를 작성하는 것이 편한 사람에게 권장.

5)Velocity

-Apache Software Foundation이 개발 중인 template engine이다.
-$나 #같은 특수한 문자를 사용해서 변수 등을 HTML code 내에 직접 작성할 수 있다.
-FreeMarker와 비슷하지만, FreeMarker가 tag 형태로 구문을 작성하는 것에 비해 이것은 tag 없이 바로 작성할 수 있어서 Visual Editor 등에 영향이 적다.

6)Mustache

-다양한 언어에 적용할 수 있는 template이다.
-Java 뿐만 아니라 PHP, Ruby, Python, JavaScript 등에서 사용할 수 있어서 Java 이외의 언어를 사용하고 싶은 사람에게 좋은 대안이 될 수 있다.
-{{}} 기호를 사용해서 변수 등을 HTML code 내에 기술한다.

7)기본은 변수식과 OGNL

-Thymeleaf의 기본은 '값을 출력하는(표시하는) 것'이다.
-이것은 ${OO} 형식으로 작성
-이 형식을 '변수식'이라고 부른다.
-변수식 안에 작성하는 것은 'OGNL(Object-Graph Navigation Language)식'이다.
-OGNL은 Java의 값에 접근하기 위한 식이다.
-Thymeleaf뿐만 아니라 다양한 library와 framework 등에서 사용되고 있다.

8)Lab

-Spring Boot project 생성
  --Package Explorer > right-click > New > Spring Starter Project
  --Service URL :http://start.spring.io
  --Name : templatedemo
  --Type : Maven
  --Packaging : Jar
  --Java Version : 8
  --Language : Java
  --Group : com.example
  --Artifact : templatedemo
  --Version : 0.0.1-SNAPSHOT
  --Description : Demo project for Spring Boot
  --Package : com.example.biz

  --Next >
  --Spring Boot Version : 1.5.15 (SNAPSHOT)

--Select Web > Web, Template Engines > Thymeleaf > Finish

-HomeController.java 생성
--com.example.biz > right-click > New > Class
--Name : HomeController

```
package com.example.biz;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

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

-index.html 생성
--src/main/resources/templates > New > Other > Web > HTML File > Next
--File name : index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Insert title here</title>
</head>
<body>
   <h1>Hello Page</h1>
   <p th:text="${new java.util.Date().toString()}"></p>
</body>
</html>
```

-실행
--http://localhost:8080

```
Hello Page
Thu Jun 28 22:27:22 KST 2018
```

9)Utility Object
-변수식은 이름 그대로 변수를 기술해서 그대로 출력할 수 있다.
-이 변수는 이미 예제에서 확인했듯이, controller에서 값을 준비하고 그것을 template 측에 전달해서 출력하는 것이 기본적인 흐름이다.
-하지만, Java class 중에는 이런 template에서 사용 빈도가 높은 것이 있다.
-이런 class를 사용할 때도 항상 controller에서 class를 변수로 준비하고...하는 것이 매우 번거로운 일이다.
-그래서 Thymeleaf에서는 자주 사용되는 class를 '#이름' 이라는 상수로 정의해서 변수식 안에 직접 작성할 수 있게 하고 있다.
-이것을 utility object라고 한다.
-utility object의 종류
--#strings : String class의 상수
--#numbers : Number class의 상수
--#bools : Boolean class의 상수
--#dates : Date class의 상수
--#objects : Object class의 상수
--#arrays : Array class의 상수
--#lists : List class의 상수
--#sets : Set class의 상수
--#maps : Map class의 상수
-이 객체들은 class의 상수이므로 직접 class method등을 호출해서 사용할 수 있다.
-단, new #dates처럼 해서는 Date instance를 만들수 없다.
-#dates.OO과 같이 class field나 method 호출 등에 사용한다.

-Lab
--index.html 수정

```
<body>
    <h1>Hello Page</h1>
    <p th:text="${#dates.format(new java.util.Date(), 'dd/MM/yyyy HH:mm')}"></p>
    <p th:text="${#numbers.formatInteger(1234,7)}"></p>
    <p th:text="${#strings.toUpperCase('Welcome to Spring.')}"></p>
</body>
```

-실행

```
Hello Page
28/06/2018 22:38

0001234

WELCOME TO SPRING.
```

10)매개변수에 접근하기

-만일 controller를 거치지 않고 query string 같은 매개변수를 사용하고자 할때는 param 이라는 변수를 사용한다.
-이것은 변수식 안에서 직접 사용할 수 있는 변수이다.
-이 변수 안에 있는 매개변수명을 지정해서 추출할 수 있다.
-${param.id}라고 하면 id=javaexpert의 형태로 전송된 값을 받을 수 있다.
-독특하게도 이 방식은 배열 형태로 되어 있어서 배열내에서 추출해야 한다.
-<body> 부분을 아래와 같이 수정한다.

```
<body>
    <h1>Helo page</h1>
    <p th:text="'from parameter... id=' + ${param.id[0] + ',name=' +
    param.name[0]}"></p>
</body>
```

-HomeController.java를 수정한다.

```
@RequestMapping("/")
public ModelAndView index(ModelAndView mav) {
    mav.setViewName("index");
    return mav;
}

@RequestMapping("/home")
public String home() {
    return "forward:/";
}
```

-그리고 id와 name을 query string으로 지정해서 접속한다.
--http://localhost:8080/home/?id=javaexpert&name=Springboot
--결과는 아래와 같다.

```
Helo page
from parameter... id=javaexpert,name=Springboot
```

-controller를 거치지 않고 template내에서 직접 전달된 값을 사용할 수 있다.
-중요한 것은 param내의 id나 name 배열에서 첫 번째 요소를 지정해서 추출한다는 것이다.
-그 이유는 query string으로 값을 전송할 때 같은 이름의 값을 여러 개 전송하기 위해서다.
-예를 들면, http://localhost:8080/home/?id=123&id=456&name=javaexpert&name=peter
-이렇게 하면 param에서 추출하는 id와 name이 각각 {123,456}, {javaexpert,peter}가 되는 것이기 때문이다.

-여기서 사용하고 있는 th:text의 값을 보면 큰따옴표 안에 다시 작은 따옴표를 사용해서 값을 작성하고 있다.
-이것은 OGNL로 text literal을 작성할 때 사용하는 방식이다.
-이렇게 하면 다수의 literal을 연결할 때 큰 따옴표안에 작은 따옴표 literal을 사용할 수 있게 된다.

```
th:text="one two three"
th:text="'one' + ' two ' + 'three'"
```

11)message식

-${}라는 변수식 외에도 thymeleaf에서 사용할 수 있는 message식이 있다.
-이것은 project에 미리 준비해둔 properties file(설정 file)에서 값을 가져와서 표시하는 것이다.

-Java에서는 properties file에 미리 text(설정값)을 저장해두고, 필요에 따라 file 내의 text값을 꺼내서 사용하는 경우가 있다.
-이것을 template에서 사용할 수 있게 한 것이 message식이다.
-형식은 아래와 같다.
#{값 지정}
-${}가 아니라 #{}로 작성하는 것이 특징이다.
-Lab
--src/main/resources > right-click > New > File
--File name : messages.properties > Finish

```
content.title=Message sample page.
content.message=This is sample message from properties.
```

--index.html 수정

```
<body>
  <h1 th:text="#{content.title}">Hello page</h1>
  <p th:text="#{content.message}"></p>
</body>
```

--실행

```
Message sample page.
This is sample message from properties.
```

12)Link식과 href
-web page에서는 URL을 지정하는 link도 다양한 곳에 사용되며 이 link를 지정하기 위한 전용식도 존재한다.

@{주소}

-이것은 기본적으로 URL을 지정하는 속성(<a> tag의 href 등)에서 사용한다.
-예를 들어, @{/index}라고 작성하면 /index로 가는 link를 지정할 수 있다.
-Lab
--index.html

```
<body>
  <h1 th:text="#{content.title}">Helo page</h1>
  <p><a th:href="@{/home/{orderId}(orderId=${param.id[0]})}">link</a></p>
</body>
```

--접속할 때 query string에 id를 지정한다.
--예를 들어 http://localhost:8080/?id=123에 접속하면 link에는 /home/123이 설정된다.

13)선택 객체와 변수식
-변수식에서 객체를 사용하려고 하면 문제가 발생한다.
-물론 ${object.name}처럼 객체의 property나 method를 지정해서 작성하면 되지만 객체안에서 다수의 값이 존재하는 경우 일일이 작성하는 것은 매우 귀찮은 일이다.
-이런 경우 객체를 지정해서 해당 객체 안에 있는 값을 추출할 수 있는 전용 변수식을 사용하면 된다.

*{}

-이 변수식은 객체를 처리하는 변수식 내부에서 사용한다.

```
<OO th:object="${객체}">
  <XX th:text="*{property}">
</OO>
```

-tag에 'th:object'라는 속성을 사용해서 객체를 설정한다.
-이렇게 하면 해당 tag 내부에서 *{}를 이용한 변수식을 사용할 수 있게 된다.
-이 * 형태의 변수식에선 객체 내의 property 등을 이름만으로도 지정할 수 있다.

-Lab
--src/main/java/com.example.biz > right-click > New > Class
--Name : Member

```
package com.example.biz;
```

```
public class Member {
   private int id;
   private String username;
   private int age;

   public Member(int id, String username, int age) {
      this.id = id;
      this.username = username;
      this.age = age;
   }

   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public String getUsername() {
      return username;
   }
   public void setUsername(String username) {
      this.username = username;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
}
```

--HomeController.java 수정

```
   @RequestMapping("/")
   public ModelAndView index(ModelAndView mav) {
      mav.setViewName("index");
      mav.addObject("msg","Current data.");
      Member obj = new Member(123, "javaexpert",24);
      mav.addObject("object",obj);
      return mav;
   }
```

--index.html 수정

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>top page</title>
   <meta charset="UTF-8" />
   <style>
      h1 { font-size:18pt; font-weight:bold; color:gray; }
      body { font-size:13pt; color:gray; margin:5px 25px; }
      tr { margin:5px; }
      th { padding:5px; color:white; background:darkgray; }
      td { padding:5px; color:black; background:#e0e0ff; }
   </style>
</head>
<body>
   <h1 th:text="#{content.title}">Hello page</h1>
   <p th:text="${msg}">message.</p>
   <table th:object="${object}">
      <tr><th>ID</th><td th:text="*{id}"></td></tr>
      <tr><th>NAME</th><td th:text="*{username}"></td></tr>
      <tr><th>AGE</th><td th:text="*{age}"></td></tr>
   </table>
</body>
</html>
```

--실행
--controller에서 object를 저장해둔 Member 값이 표 형태로 출력된다.

12. Lab : Spring Boot와 JDBC 연동하기

1)Spring Boot project 생성

-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : BootJdbcDemo
-Type : Maven
-Packaging : Jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : BootJdbcDemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select SQL > check MySQL, JDBC
-Select Web > check Web > Finish

-위에서 type을 선택시 고려사항
--만일 Embeded된 tomcat으로 Stand-Alone형태로 구동시키기 위한 목적이라면 Packaging Type을 jar로 선택한다.
--war로 선택할 경우, 기존과 같이 외부의 tomcat으로 deploy 하는 구조로 만들어진다.
--물론, source의 최종배포의 형태가 server의 tomcat에 deploy해야 하는 구조라면 처음부터 war로 만들어서 작업해도 상관없다.
--jar로 선택하고, local에서 개발 및 test를 하다가 나중에 배포할 경우 war로 변경해서 배포를 할 수도 있다.

2)pom.xml

-jstl 사용을 위한

```
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>jstl</artifactId>
   <version>1.2</version>
</dependency>
```

-jsper 사용을 위한

```
<dependency>
   <groupId>org.apache.tomcat</groupId>
   <artifactId>tomcat-jasper</artifactId>
   <version>8.5.31</version>
</dependency>
```

3)src/main/resources/application.properties

```
spring.application.name=BootJDBCDemo
spring.datasource.url=jdbc:mysql://localhost:3306/world
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=javamysql
```

4)Table 생성

```
CREATE TABLE world.User
(
   username    VARCHAR(20) PRIMARY KEY,
   age TINYINT NOT NULL
);
```

5)VO object 생성

-src/main/java > right-click > New > Package
-Name : com.example.vo
-com.example.vo > right-click > New > Class
-Name : UserVO

```
package com.example.vo;

public class UserVO {
   private String username;
   private int age;

   public String getUsername() {
      return username;
   }
   public void setUsername(String username) {
      this.username = username;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   @Override
   public String toString() {
      return "UserVO [username=" + username + ", age=" + age + "]";
   }
}
```

6)Dao object 생성
-src/main/java > right-click > New > Package
-Name : com.example.dao
-com.example.dao > right-click > New > Inteface
-Name : UserDao

```
package com.example.dao;

import java.util.List;

import com.example.vo.UserVO;

public interface UserDao {
   int create(UserVO userVO);
   List<UserVO> readAll();
   UserVO read(String username);
   int update(UserVO userVO);
   int delete(String username);
}
```

-com.example.dao > right-click > New > Class
-Name : UserDaoImpl

```
package com.example.dao;

import static org.mockito.Mockito.RETURNS_DEEP_STUBS;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

import com.example.vo.UserVO;
```

```
@Repository
public class UserDaoImpl implements UserDao {
   @Autowired
   private JdbcTemplate jdbcTemplate;

   class UserMapper implements RowMapper<UserVO> {
      public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
         UserVO user = new UserVO();
         user.setUsername(rs.getString("username"));
         user.setAge(rs.getInt("age"));
         return user;
      }
   }

   @Override
   public int create(UserVO userVO) {
      String sql = "INSERT INTO User(username, age) VALUES(?, ?)";
      return this.jdbcTemplate.update(sql, userVO.getUsername(), userVO.getAge());
   }

   @Override
   public List<UserVO> readAll() {
      String sql = "SELECT * FROM User";
      return this.jdbcTemplate.query(sql, new UserMapper());
   }

   @Override
   public UserVO read(String username) {
      String sql = "SELECT * FROM User WHERE username = ?";
      return this.jdbcTemplate.queryForObject(sql, new Object[] {username}, new
      UserMapper());
   }

   @Override
   public int update(UserVO userVO) {
      String sql = "UPDATE User SET age = ? WHERE username = ?";
      return this.jdbcTemplate.update(sql, userVO.getAge(), userVO.getUsername());
   }

   @Override
   public int delete(String username) {
      String sql = "DELETE FROM User WHERE username = ?";
      return this.jdbcTemplate.update(sql, username);
   }

}
```

7)Controller
-com.example.biz > right-click > New > Class
-Name : MainController

```
package com.example.biz;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import com.example.dao.UserDao;
import com.example.vo.UserVO;

@Controller
public class MainController {
   @Autowired
   private UserDao userDao;

```

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

        @PostMapping("/user")
        public String insert(UserVO userVO) {
            this.userDao.create(userVO);
            return "redirect:/user";
        }

        @GetMapping("/user")
        public String list(Model model) {
            model.addAttribute("users", this.userDao.readAll());
            return "list";
        }
    }
```

8)static resources 준비
- -src/main/resources/static/images folder 생성
  - --spring-boot.png 추가할 것
- -src/main/resources/static/js folder 생성
  - --jquery-3.3.1.slim.min.js 추가할 것
- -src/main/resources/static/css folder 생성
  - --style.css

```
@charset "UTF-8";
body {
    background-color:yellow;
}
h1{
    color : blue;
}
```

9)JSP를 위한 folder 준비
- -기본적으로 Spring Boot 에서는 jsp파일을 인식이 되지 않는다.
- -그래서 만일 jar로 packaging 한다면 embedded tomcat이 인식하는 web루트를 생성한다.
- -src > main 폴더 밑에 webapp와 jsp가 위치할 폴더를 만들어준다.
- -src/main folder 안에 webapp folder 생성
- -webapp folder 안에 WEB-INF folder 생성
- -WEB-INF folder 안에 jsp folder 생성

10)만일 JSP를 template으로 사용하는 경우에는 이 WEB-INF folder 안에 template file을 준비할 필요가 있다.

11)folder를 다 만들었으면, WEB 루트(Context Root)로 인식할 있도록 환경설정을 아래와 같이 추가한다.
- -src/mian/resources/application.properties code 추가

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

12)jsp folder 안에 jsp file 생성
- -index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>New User Insertion Page</title>
<link rel="stylesheet" type="text/css" href="/css/style.css">
<c:url var="jsurl" value="/js/jquery-3.3.1.slim.min.js" />
<script src="${jsurl}"></script>
<script>
    $(document).ready(function() {
        alert("Hello, Spring Boot!");
    });
```

```
    </script>
    </head>
    <body>

        <img src="/images/spring-boot.png" />
        <h1>New User Insertion Page</h1>
        <form action="/user" method="post">
            Name : <input type="text" name="username" /><br />
            Age : <input type="number" name="age" /><br />
            <button type="submit">Submit</button>
        </form>
    </body>
    </html>

-list.jsp

    <%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>User List</title>
    <link rel="stylesheet" type="text/css" href="/css/style.css">
    </head>
    <body>
        <h1>User List</h1>
        <ul>
            <c:forEach var="user" items="${users}">
                <li>${user.username}(${user.age})</li>
            </c:forEach>
        </ul>
    </body>
    </html>

13)src/main/java/com.example.biz/BootJdbcDemoApplication.java

    package com.example.biz;

    import org.springframework.boot.SpringApplication;
    import org.springframework.boot.autoconfigure.SpringBootApplication;
    import org.springframework.context.annotation.ComponentScan;

    @SpringBootApplication
    @ComponentScan("com.example")    <-- 추가 code
    public class BootJdbcDemoApplication {

        public static void main(String[] args) {
            SpringApplication.run(BootJdbcDemoApplication.class, args);
        }
    }

    -@SpringBootConfiguration 은 다음의 annotation 3개를 모두 담고 있다.
        --@SpringBootConfiguration
        --@EnableAutoConfiguration
        --@ComponentScan
    -만약, AutoScan이 되어야 하는 component class들 - 대표적으로 @Controller, @Service, @Repository, @Component등-의 위치가 main class가 위치한 package보다 상위 package에 있거나,  하위가 아닌 다른 package에 있는 경우, scan이 되지 않는다.
    -Controller class가 main의 하위 class에 있으면 상관없지만, 예를 들어, main class가 다른 package나 하위 package가 아니면 아래와 같이 해줘야 한다.
    -명시적으로 ComponentScan을 할 Base Package를 지정해주면 된다.
        --ex) @ComponentScan(basePackages = "com.springboot.demo")

14)project > right-click > Run As    > Spring Boot App
15)http://localhost:8080
```

13. Spring Boot와 JPA 연동하기
-Refer to http://jdm.kr/blog/121
1)Spring Boot의 Database 처리는 기본적으로 JPA 기술을 기반으로 하고 있다.
2)이 JPA를 Spring Framework에서 사용할 수 있게 한 것이 'Spring Data JPA' framework이다.
3)Spring Boot에서는 JTA(Java Transaction API; Java EE에 transaction 처리를 제공), Spring ORM, Spring Aspects/Spring AOP는 'Spring Boot Starter Data JPA'라는 library를 사용해서 통합적으로 사용할 수 있다.
4)즉, 이 library는 각종 library를 조합해서 간단히 database 접속을 구현하게 한 기능이다.
5)Spring Boot project 생성
-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : JpaDemo
-Type : Maven
-Packaging : Jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : JpaDemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Check SQL > JPA, H2 Finish

6)Entity
-JPA에서 Entity라는 것은 Database에 저장하기 위해서 정의한 class이다.
-일반적으로 RDBMS에서 Table 같은 것이다.
-com.example.biz > right-click > New > Class
-Name : MemberVO

```
package com.example.biz;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity(name="Member")
public class MemberVO {
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private long id;

   @Column
   private String username;

   @Column
   private int age;

   public MemberVO() {}
   public MemberVO(String username, int age) {
      this.username = username;
      this.age = age;
   }

   public long getId() {
      return id;
   }
   public void setId(long id) {
      this.id = id;
   }
   public String getUsername() {
      return username;
```

```
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "MemberVO [id=" + id + ", username=" + username + ", age=" + age + "]";
    }
}
```

-여기서 주의할 점은 기본 생성자는 반드시 넣어야 한다.

7)Repository

-Entity class를 구성했다면 이번엔 Repository interface를 만들어야 한다.

-Spring Framework에서는 Entity의 기본적인 삽입, 조회, 수정, 삭제가 가능하도록 CrudRepository라는 inteface가 있다.

-com.example.biz > right-click > New > Interface

-Name : MemberRepository

```
package com.example.biz;

import java.util.List;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

public interface MemberRepository extends CrudRepository<MemberVO, Long> {
    List<MemberVO> findByUsernameAndAgeLessThan(String username, int age);

    @Query("select t from Member t where username= :username and age < :age")
    List<MemberVO> findByUsernameAndAgeLessThanSQL(@Param("username") String
    username, @Param("age") int age);

    List<MemberVO> findByUsernameAndAgeLessThanOrderByAgeDesc(String username, int
    age);
}
```

-위의 코드는 실제로 MemberVO Entity를 이용하기 위한 Repository class이다.

-기본적인 method 외에도 추가적인 method를 지정할 수 있다.

-method 이름을 기반(Query Method)으로 해서 만들어도 되고 @Query를 이용해 기존의 SQL처럼 만들어도 된다.

-findByUsernameAndAgeLessThan method와 findByUsernameAndAgeLessThanSQL method는 같은 결과를 출력하지만 전자의 method는 method 이름을 기반으로 한 것이고 후자의 method는 @Query annotation을 기반으로 해서 만든 것이다.

-method 이름 기반으로 해서 만들면 추후에 사용할 때 method 이름만으로도 어떤 query인지 알 수 있다는 장점이 있다.

-반대로 @Query annotation으로 만든 method는 기존의 source를 converting하는 경우 유용하게 사용할 수 있다.

-@Query

--다만 @Query annotation으로 query를 만들 때에 from 절에 들어가는 table은 Entity로 지정된 class 이름이다.

-method 이름 기반 작성법

-해당 부분은 Spring 문서(https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation)를 통해 확인할 수 있다.

8)Application 작성

-Entity 및 Repository가 준비 되었다면 실제로 @SpringBootApplication annotation이 있는 class에서 실제로 사용해 보자.

```
package com.example.biz;
```

```
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JpaDemoApplication  implements CommandLineRunner {

   @Autowired
   MemberRepository memberRepository;

   public static void main(String[] args) {
      SpringApplication.run(JpaDemoApplication.class, args);
   }

   @Override
   public void run(String... args) throws Exception {
      memberRepository.save(new MemberVO("a", 10));
      memberRepository.save(new MemberVO("b", 15));
      memberRepository.save(new MemberVO("c", 10));
      memberRepository.save(new MemberVO("a", 5));

      Iterable<MemberVO> list1 = memberRepository.findAll();

      System.out.println("findAll() Method.");
      for( MemberVO m : list1){
         System.out.println(m.toString());
      }

      System.out.println("findByUserNameAndAgeLessThan() Method.");
      List<MemberVO> list2 = memberRepository.findByUsernameAndAgeLessThan("a", 10);
      for( MemberVO m : list2){
         System.out.println(m.toString());
      }

      System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
      List<MemberVO> list3 = memberRepository.findByUsernameAndAgeLessThanSQL("a",
      10);
      for( MemberVO m : list3){
         System.out.println(m.toString());
      }

      System.out.println("findByUserNameAndAgeLessThanSQL() Method.");
      List<MemberVO> list4 =
      memberRepository.findByUsernameAndAgeLessThanOrderByAgeDesc("a", 15);
      for( MemberVO m : list4){
         System.out.println(m.toString());
      }

      memberRepository.deleteAll();
   }
}
```

9)실행

```
findAll() Method.
Member [id=1, name=a, age=10]
Member [id=2, name=b, age=15]
Member [id=3, name=c, age=10]
Member [id=4, name=a, age=5]
findByNameAndAgeLessThan() Method.
Member [id=4, name=a, age=5]
findByNameAndAgeLessThanSQL() Method.
Member [id=4, name=a, age=5]
findByNameAndAgeLessThanSQL() Method.
```

Member [id=1, name=a, age=10]
Member [id=4, name=a, age=5]

14. Lab : Spring Boot JPA

1)Spring Boot project 생성

-Package Explorer > right-click > New > Spring Starter Project
-Service URL :http://start.spring.io
-Name : BootJpaDemo
-Type : Maven
-Packaging : Jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : BootJpaDemo
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

-Next >
-Spring Boot Version : 1.5.15 (SNAPSHOT)
-Select Core > DevTools, SQL > HSQLDB, JPA
-Select Web > Web, Template Engines > Thymeleaf > Finish

-DevTools
--It provides developer tools.
--These tools are helpful in application development mode.
--One of the features of developer tool is automatic restart of the server for any change in code.

2)Entity 작성하기

-com.example.biz > right-click > New > Package
-Name : com.example.biz.vo
-com.example.biz.vo > right-click > New > Class
-Name : User

```
package com.example.biz.vo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column
    private long id;

    @Column(length = 20, nullable = false)
    private String username;

    @Column(length = 100, nullable = true)
    private String email;

    @Column(nullable = true)
    private Integer age;

    @Column(nullable = true)
    private String memo;

    public long getId() {
```

```
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    public String getMemo() {
        return memo;
    }
    public void setMemo(String memo) {
        this.memo = memo;
    }
}
```

3)Repository 생성하기

-com.example.biz > right-click > New > Package

-Name : com.example.biz.dao

-com.example.biz.dao > right-click > New > Interface

-Name : UserRepository > Finish

```
package com.example.biz.dao;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.biz.vo.User;

@Repository("repository")
public interface UserRepository extends JpaRepository<User, Long>{

}
```

-JpaRepository라는 interface는 새로운 repository를 생성하기 위한 토대가 된다.

-모든 Repository는 이 JpaRepository를 상속해서 작성한다.

4)HelloController 작성

-com.example.biz > right-click > New > Class

-Name : HelloController

```
package com.example.biz;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import com.example.biz.dao.UserRepository;
import com.example.biz.vo.User;

@Controller
public class HelloController {
```

```
    @Autowired
    UserRepository repository;

    @RequestMapping("/")
    public ModelAndView index(ModelAndView mav) {
      mav.setViewName("index");
      mav.addObject("msg","this is sample content.");
      Iterable<User> list = repository.findAll();
      mav.addObject("data",list);
      return mav;
    }
  }
```

-UserRepository에는 findAll 같은 method가 정의되어 있지 않다.
-이것은 부모 interface인 JpaRepository가 가지고 있는 method이다.
-이를 통해 모든 entity가 자동으로 추출되는 것이다.

5)JUnit Test
-src/test/java/com.example.biz.BootJpaDemoApplicationTests.java > right-click > Run As > JUnit Test
-Green bar

6)template 준비하기
-src/main/resources > right-click > New > File
-File name : messages.properties

```
content.title=Message sample page.
content.message=This is sample message from properties.
```

-src/main/resources/static > right-click > New > Web > HTML Files
-Name : index.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>top page</title>
  <meta charset="UTF-8" />
  <style>
    h1 { font-size:18pt; font-weight:bold; color:gray; }
    body { font-size:13pt; color:gray; margin:5px 25px; }
    pre { border: solid 3px #ddd; padding: 10px; }
  </style>
</head>
<body>
  <h1 th:text="#{content.title}">Hello page</h1>
  <pre th:text="${data}"></pre>
</body>
</html>
```

-src/main/resources > right-click > New > File
--File name : messages.properties > Finish

```
content.title=Message sample page.
content.message=This is sample message from properties.
```

-실행해서 접속
--저장돼있는 data가 회색 사각 틀 안에 표시된다.
-아직 아무 data가 없기 때문에 빈 배열 []라고 표시된다.

7)Entity의 CRUD 처리하기 : form으로 data 저장하기
-index.html 수정

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>top page</title>
```

```
        <meta charset="UTF-8" />
        <style>
        h1 { font-size:18pt; font-weight:bold; color:gray; }
        body { font-size:13pt; color:gray; margin:5px 25px; }
        tr { margin:5px; }
        th { padding:5px; color:white; background:darkgray; }
        td { padding:5px; color:black; background:#e0e0ff; }
        </style>
    </head>
    <body>
        <h1 th:text="#{content.title}">Hello page</h1>
        <table>
        <form method="post" action="/" th:object="${formModel}">
            <tr><td><label for="username">이름</label></td>
                <td><input type="text" name="username" th:value="*{username}" /></td></tr>
            <tr><td><label for="age">연령</label></td>
                <td><input type="text" name="age"  th:value="*{age}" /></td></tr>
            <tr><td><label for="email">메일</label></td>
                <td><input type="text" name="email"  th:value="*{email}" /></td></tr>
            <tr><td><label for="memo">메모</label></td>
            <td><textarea name="memo"  th:text="*{memo}"
                    cols="20" rows="5"></textarea></td></tr>
            <tr><td></td><td><input type="submit" /></td></tr>
        </form>
        </table>
        <hr/>
        <table>
            <tr><th>ID</th><th>이름</th></tr>
            <tr th:each="obj : ${datalist}">
                <td th:text="${obj.id}"></td>
                <td th:text="${obj.username}"></td>
            </tr>
        </table>
    </body>
    </html>
```

-HelloController.java 수정

```
    package com.example.biz;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Controller;
    import org.springframework.transaction.annotation.Transactional;
    import org.springframework.web.bind.annotation.ModelAttribute;
    import org.springframework.web.bind.annotation.RequestMapping;
    import org.springframework.web.bind.annotation.RequestMethod;
    import org.springframework.web.servlet.ModelAndView;

    import com.example.biz.dao.UserRepository;
    import com.example.biz.vo.User;

    @Controller
    public class HelloController {

        @Autowired
        UserRepository repository;

        @RequestMapping(value = "/", method = RequestMethod.GET)
        public ModelAndView index(@ModelAttribute("formModel") User mydata, ModelAndView
        mav) {
            mav.setViewName("index");
            mav.addObject("msg","this is sample content.");
            Iterable<User> list = repository.findAll();
            mav.addObject("datalist",list);
            return mav;
        }
```

```
    @RequestMapping(value = "/", method = RequestMethod.POST)
    @Transactional(readOnly=false)
    public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView
    mav) {
        repository.saveAndFlush(mydata);
        return new ModelAndView("redirect:/");
    }
}
```

-접속해서 실행
--form에 text를 입력해서 전송하면 data가 추가돼서 아래쪽 table에 표시된다.

8)@ModelAttribute와 data 저장
-@ModelAttribute
--이것은 entity class의 instance를 자동으로 적용할 때 사용
--인수에는 instance 이름을 지정한다.
--이것은 전송 form에서 th:object로 지정하는 값이 된다.
--전송된 form의 값이 자동으로 User instance로 저장된다.
--따라서 이 annotation을 이용하면 이렇게 쉽게 전송한 data를 저장할 수 있다.

9)saveAndFlush() method
-HomeController.java의 아래 code를 보자.

```
    @RequestMapping(value = "/", method = RequestMethod.POST)
    @Transactional(readOnly=false)
    public ModelAndView form(@ModelAttribute("formModel") User mydata, ModelAndView mav) {
        repository.saveAndFlush(mydata);
        return new ModelAndView("redirect:/");
    }
```

-미리 설정한 entity는 JpaRepository의 saveAndFlush라는 method를 통해 entity를 영구화한다.
-Database를 사용하고 있다면 Database에 그대로 저장된다.

10)@Transactional과 transaction
-바로 위의 code에서 @Transactional(readOnly=false)가 있다.
-이 annotation은 transaction을 위한 것이다.
-이 annotation때문에 method내에서 실행되는 database 처리가 일괄적으로 실행되게 된다.
-data 변경 처리는 도중에 외부 접속에 의해 data 구조나 내용이 바뀌면 data 일관성에 문제가 발생하게 된다.
-이런 문제를 방지하기 위해 transaction이 사용되는 것이다.
-code를 보면 readOnly=false라고 설정하고 있다.
-이 readOnly는 문자 그대로 '읽기 전용(변경 불가)임을 의미한다.
-readOnly=false라고 설정하면 변경을 허가하는 transaction이다.

11)Data 초기화 처리
-저장한 data는 application으르 종료하고 다시 실행하면 지워진다.
-HSQLDB는 기본적으로 memory내에 data를 cache하고 있으므로 종료와 함께 지워지는 것이다.
-controller에 data를 작성하는 초기화 처리를 넣기로 한다.
-HelloController.java code 추가

```
    @PostConstruct
    public void init(){
        User user1 = new User();
        user1.setUsername("한지민");
        user1.setAge(24);
        user1.setEmail("javaexpert@nate.com");
        user1.setMemo("Hello, Spring JPA");
        repository.saveAndFlush(user1);

        User user2 = new User();
        user2.setUsername("조용필");
        user2.setAge(66);
        user2.setEmail("aaa@aaa.com");
        user2.setMemo("Good Morning!");
        repository.saveAndFlush(user2);

        User user3 = new User();
        user3.setUsername("이미자");
```

```
            user3.setAge(70);
            user3.setEmail("bbb@bbb.com");
            user3.setMemo("Spring Boot is very good.");
            repository.saveAndFlush(user3);
        }
```

-@PostConstruct는 생성자를 통해 instance가 생성된 후에 호출되는 method임을 나타낸다.
-Controller는 처음에 한 번만 instance를 만들고 이후에는 해당 instance를 유지한다.
-따라서 여기에 test용 data 작성 처리를 해두면 application 실행시에 반드시 한 번 실행되어, data가 준비되는 것이다.

12)User Find 및 Update 처리하기
-src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
-File name : edit.html > Finish

```
        <!DOCTYPE HTML>
        <html xmlns:th="http://www.thymeleaf.org">
        <head>
            <title>edit page</title>
            <meta charset="UTF-8" />
            <style>
                h1 { font-size:18pt; font-weight:bold; color:gray; }
                body { font-size:13pt; color:gray; margin:5px 25px; }
                tr { margin:5px; }
                th { padding:5px; color:white; background:darkgray; }
                td { padding:5px; color:black; background:#e0e0ff; }
            </style>
        </head>
        <body>
            <h1 th:text="${title}">Edit page</h1>
            <table>
            <form method="post" action="/edit" th:object="${formModel}">
                <input type="hidden" name="id" th:value="*{id}" />
                <tr><td><label for="username">이름</label></td>
                    <td><input type="text" name="username" th:value="*{username}" /></td></tr>
                <tr><td><label for="age">연령</label></td>
                    <td><input type="text" name="age"  th:value="*{age}" /></td></tr>
                <tr><td><label for="email">메일</label></td>
                    <td><input type="text" name="email"  th:value="*{email}" /></td></tr>
                <tr><td><label for="memo">메모</label></td>
                    <td><textarea name="memo"  th:text="*{memo}"
                    cols="20" rows="5"></textarea></td></tr>
                <tr><td></td><td><input type="submit" /></td></tr>
            </form>
            </table>
        </body>
        </html>
```

-UserRepository code 추가

```
        @Repository("repository")
        public interface UserRepository extends JpaRepository<User, Long> {
            public User findById(Long id);
        }
```

-RequestHandler 작성하기
---HelloController.java code 추가

```
            @RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
            public ModelAndView edit(@ModelAttribute User user,  @PathVariable int id,
            ModelAndView mav) {
                mav.setViewName("edit");
                mav.addObject("title","edit mydata.");
                User findUser = repository.findById((long)id);
                mav.addObject("formModel", findUser);
                return mav;
            }
```

```
    @RequestMapping(value = "/edit", method = RequestMethod.POST)
    @Transactional(readOnly=false)
    public ModelAndView update(@ModelAttribute User user, ModelAndView mav) {
      repository.saveAndFlush(user);
      return new ModelAndView("redirect:/");
    }
```

-접속해서 아래와 같이 URL을 입력하면

```
http://localhost:8080/edit/1
```

-해당 ID의 data가 표시된다.
-data를 변경하고 전송해보자.
-findById는 어디서 구현되는 것일까?
--repository는 method의 이름을 기준으로 entity 검색 처리를 자동 생성한다.
--즉 repository에 method 선언만 작성하고, 구체적인 처리를 구현할 필요가 없다.

13)Entity delete 구현하기
-update를 하고 select를 했으니 이번에는 delete를 해 보자.
-delete.html template를 작성한다.
-src/main/resources/templates > right-click > New > Other > Web > HTML File > Next
-File name : delete.html > Finish

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>edit page</title>
  <meta charset="UTF-8" />
  <style>
    h1 { font-size:18pt; font-weight:bold; color:gray; }
    body { font-size:13pt; color:gray; margin:5px 25px; }
    td { padding:0px 20px; background:#eee;}
  </style>
</head>
<body>
  <h1 th:text="${title}">Delete page</h1>
  <table>
  <form method="post" action="/delete" th:object="${formModel}">
    <input type="hidden" name="id" th:value="*{id}" />
    <tr><td><p th:text="|이름 :  *{username}|"></p></td></tr>
    <tr><td><p th:text="|연령 :  *{age}|" ></p></td></tr>
    <tr><td><p th:text="*{email}"></p></td></tr>
    <tr><td><p th:text="*{memo}"></p></td></tr>
    <tr><td><input type="submit" value="delete"/></td></tr>
  </form>
  </table>
</body>
</html>
```

-RequestHandler 작성

```
@RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
public ModelAndView delete(@PathVariable int id, ModelAndView mav) {
  mav.setViewName("delete");
  mav.addObject("title","delete mydata.");
  User user = repository.findById((long)id);
  mav.addObject("formModel",user);
  return mav;
}

@RequestMapping(value = "/delete", method = RequestMethod.POST)
@Transactional(readOnly=false)
public ModelAndView remove(@RequestParam long id, ModelAndView mav) {
  repository.delete(id);
  return new ModelAndView("redirect:/");
}
```

-접속해서 실행

2703 http://localhost:8080/delete/2라고 하면 id가 2번이 출력되고 delete button을 누르면 삭제된다.