Spring Boot 시작하기
(version 2.0.6)

Jade

Spring Boot

2014년부터 개발된 스프링 하위 프로젝트

단독 실행 가능한 수준의 Spring Application 제작 가능

내장된 가벼운 WAS (Tomcat, Jetty, UnderTow등)로 설치없이 가볍게 개발 가능

쉽고 간단하고 자동화된 설정 방식

http://spring.io/projects/spring-boot

Create Spring Boot Project

1. File > New Spring Starter Project SBP (version 2.0+)
2. add dependent libs
   - Web, Lombok, MySQL, MyBatis, Thymeleaf, Devtools
3. https://start.spring.io/
   https://start.spring.io/starter.zip?name=sbp&groupId=com.jade&artifactId=sbp&version=0.0.1-SNAPSHOT&description=Demo+project+for+Spring+Boot&packageName=com.jade.sbp&type=maven-project&packaging=war&javaVersion=1.8&language=java&bootVersion=2.0.5.RELEASE&dependencies=lombok&dependencies=mysql&dependencies=mybatis&dependencies=thymeleaf&dependencies=web

Auto Reload (pom.xml :: devtools)

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
</dependency>
```

```
spring.devtools.livereload.enabled=true
spring.thymeleaf.cache=false
```

Spring Unit Test

Default Includes

- **JUnit**, Spring Test & Spring Boot Test
- **AssertJ**(assertThat), Hamcrest, **Mockito**
- JSONAssert, JsonPath

@SpringBootTest under **@RunWith(SpringRunner.class)**

@WebMvcTest(HelloController.class**)** under **@RunWith(SpringRunner.class)**

```
assertThat(conn).isInstanceOf(Connection.class);
```

@To Do

1. Set Server Port 9090 and DataSource & MyBatis
2. Make HelloController (REST) & HelloControllerTest (MockMVC)
3. Make UserMapper interface
4. use Interceptor
5. use JSP
6. use Thymeleaf Template Engine
7. 한글 Test

1. Set Server Port 9090 and DataSource & MyBatis

src/main/resources/application.properties

server.port=9090

spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://115.71.233.22:3306/testdb?characterEncoding=UTF-8
spring.datasource.username=<dbuserid>
spring.datasource.password=<dbuserpw>

1. DataSource & MyBatis (Cont'd) : Make Test

// SbpApplicationTests.java (default Test Unit)

```
@Autowired
private DataSource ds;

@Test
public void testDataSource() throws Exception {
	System.out.println("DS=" + ds);

	try (Connection conn = ds.getConnection()) {
		System.out.println("Coooooooooonn=" + conn);
		assertThat(conn).isInstanceOf(Connection.class);

		assertEquals(100, getLong(conn, "select 100"));

	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

2. Make HelloController & HelloControllerTest

```
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;

@RunWith(SpringRunner.class)
@WebMvcTest(HelloController.class)
public class HelloControllerTest {
   @Autowired
   MockMvc mock;

   @Test
   public void testHello() throws Exception {
         mock.perform(get("/hello"))
            .andExpect(status().isOk())
            .andExpect(content().string("Hello 스프링부트!!"));

         MvcResult result = mock.perform(get("/hello"))
                     .andExpect(status().isOk())
                     .andReturn();

         assertEquals("Hello 스프링부트!!", result.getResponse().getContentAsString());
         assertThat(result.getResponse().getContentAsString()).isEqualTo("Hello 스프링부트!!");

         System.out.println("RRR>>" + result.getResponse().getContentAsString());
   }
}
```

```
@RestController
public class HelloController {
      @RequestMapping("/hello")
      public String hello() {
            return "Hello 스프링부트!!";
      }
}
```

3. Make UserMapper interface

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jade.sbp.mapper.UserMapper">
    <select id="getLoginInfo"
resultType="com.jade.sbp.domain.User">
        select * from User where uid = #{uid}
    </select>
</mapper>
```

1. Make com.jade.sbp.mapper package
2. Make UserMapper interface
3. Set MapperScan in SbpApplication.java
   @MapperScan(value={"com.jade.sbp.mapper"})
4. Add UserMapperTest to SbpApplicationTests.java
   ```
   @Autowired private UserMapper mapper;
   assertEquals("김일수", mapper.getUname("user1"));
   ```
5. Make com.jade.sbp.domain.**User** Value Object
   ```
   @Data & @ToString(exclude={"upw","naverid","googleid"})
   ```
6. Add UserMapper.getLoginInfo() by using Mapper XML
7. Make /resources/mappers/userMapper.xml & regist application.properties
   mybatis.mapper-locations=classpath:/mappers/*Mapper.xml

4. Use Interceptor

1. Make HelloInterceptor extends HandlerInterceptorAdaptor

2. Make SbpWebMvcConfig extends **WebMvcConfigurer**
   (warning: **WebMvcConfigurerAdapter is deprecated by 2.0+**)

```
@Configuration
public class SbpWebMvcConfig implements WebMvcConfigurer {
  @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new HelloInterceptor())
            .addPathPatterns("/hello");
    }
}
```

5. use JSP

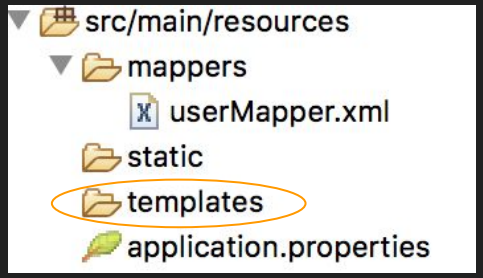
pom.xml

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

src/main/resources/application.properties

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

6. use **Thymeleaf** Template Engine



pom.xml (with Web)

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

src/main/resources/application.properties (if not auto-reload)

```
spring.thymeleaf.cache=false
```

6. use **Thymeleaf** Template Engine (Cont'd)

```
<h1>Thymeleaf Test : <span th:text="${uname}"/></h1>

<ul>
    <li th:each="data: ${users}" th:text="${data}"></li>
    <li th:each="data: ${users}" th:text="${data.uid} + '. ' + ${data.uname}"></li>
</ul>

<div>
    <a th:href="@{http://localhost:9090/hello}">HELLO</a> |
    <a th:href="@{http://localhost:9090/hello2}">HELLO22</a>
</div>

<p th:text="'Name: ' + ${name}"></p>
<form method="POST" action="korean">
    <input type="text" name="name" value="세종대왕" />
    <button type="submit">KOREAN</button>
</form>
```

6. use **Thymeleaf** Template Engine (Cont'd)

```
<!-- IF -->
<div th:if="${user}">
   <h1 th:object="${user}">Thymeleaf Test : <span th:text="|*{uname}|"/></h1>
   <h3>Thymeleaf Test2 : <span th:text="${user.uname}"/></h3>

   <div th:switch="${user.uname}">
        <p th:case="'admin'">User is an administrator</p>
        <p th:case="${uname}">User is a manager!!!!!!!!!!!!</p>
        <p th:case="*">User is some other thing</p>
    </div>
</div>

<!-- ELSE -->
<div th:unless="${user}">
   <h1>Thymeleaf Test : Guest</h1>
</div>
```

6. use Thymeleaf Template Engine (Cont'd)

```java
@Controller
public class UserController {

   @Autowired
   private UserMapper mapper;

   @RequestMapping("/hello2")
   public void hello2(Model model) throws Exception {
         System.out.println("HELLO22222222222222222222222222");
         model.addAttribute("uname", "전성호2");

         List<User> users = mapper.getUsers();
         System.out.println("UUUUUsers>>" + users.size());
         model.addAttribute("users", users);

         if (users != null && users.size() > 0)
               model.addAttribute("user", users.get(0));
   }

   @RequestMapping(value="/korean", method=RequestMethod.POST)
   public String korean(@RequestParam String name, Model model) {
         System.out.println("name=" + name);
         model.addAttribute("name", name);

         return "/hello2";
   }
}
```

그동안 Spring MVC 수업 받느라 정말 고생들 많으셨고,

성원해 주신 모든 분들께 가슴깊이 감사드립니다.

상아, 찬, 정수, 경민, 성원, ...