```
HOL : MyBatis in Spring Boot
-------------------------------------
Task1. String Jdbc API 사용하기
1. Spring Boot project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음의 각 항목의 값을 입력후 Next 클릭
    -Service URL :http://start.spring.io
    -Name : SpringBootJdbcDemo
    -Type : Maven
    -Packaging : Jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : SpringBootJdbcDemo
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz

  3)다음 각 항목을 선택후 Finish 클릭
    -Spring Boot Version : 2.2.6
    -Select
      --SQL > check Oracle Driver, JDBC API
      --Web > Spring Web
      --Developer Tools > Spring Boot DevTools, Lombok
      --Template Engines > Thymeleaf
    -Finish

2. src/main/resources/application.properties
  spring.datasource.url=jdbc:mariadb://localhost:3306/test
  spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
  spring.datasource.username=root
  spring.datasource.password=javamariadb


3. Table 생성
  CREATE TABLE Member
  (
    userid     VARCHAR2(20) PRIMARY KEY,
    username VARCHAR2(20) NOT NULL,
    age          NUMBER(2) NOT NULL
  );


4. VO object 생성
  1)src/main/java > right-click > New > Package
  2)Name : com.example.vo
  3)Finish
  4)com.example.vo > right-click > New > Class
  5)Name : MemberVO

    package com.example.vo;

    import lombok.AllArgsConstructor;
    import lombok.Getter;
    import lombok.NoArgsConstructor;
    import lombok.Setter;
    import lombok.ToString;

```

```
    @Setter
    @Getter
    @NoArgsConstructor
    @AllArgsConstructor
    @ToString
    public class MemberVO {
      private String userid;
      private String username;
      private int age;
    }


5. Dao object 생성
  1)src/main/java > right-click > New > Package
  2)Name : com.example.dao
  3)Finish
  4)com.example.dao > right-click > New > Inteface
  5)Name : MemberDao

    package com.example.dao;

    import java.util.List;
    import com.example.vo.MemberVO;

    public interface MemberDao {
      int create(MemberVO member);

      List<MemberVO> readAll();

      MemberVO read(String userid);

      int update(MemberVO member);

      int delete(String userid);
    }

  6)com.example.dao > right-click > New > Class
  7)Name : MemberDaoImpl
  8)Interfaces : com.example.MemberDao
  9)Finish

    package com.example.dao;

    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.MemberVO;

    @Repository
    public class MemberDaoImpl implements MemberDao {
      @Autowired
      private JdbcTemplate jdbcTemplate;
```

```
117
118         @Override
119         public int create(MemberVO member) {
120           String sql = "INSERT INTO Member(userid, username, age) VALUES(?,?,?)";
121           return this.jdbcTemplate.update(sql, member.getUserid(), member.getUsername(),
              member.getAge());
122         }
123
124         private class MyRowMapper implements RowMapper<MemberVO> {
125           @Override
126           public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
127             return new MemberVO(rs.getString("userid"), rs.getString("username"),
                rs.getInt("age"));
128           }
129         }
130
131         @Override
132         public List<MemberVO> readAll() {
133           return this.jdbcTemplate.query("SELECT * FROM Member ORDER BY userid DESC",
              new MyRowMapper());
134         }
135
136         @Override
137         public MemberVO read(String userid) {
138           String sql = "SELECT * FROM Member WHERE userid = ?";
139           return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new
              MyRowMapper());
140         }
141
142         @Override
143         public int update(MemberVO member) {
144           String sql = "UPDATE Member SET age = ?, username = ? WHERE userid = ?";
145           return this.jdbcTemplate.update(sql, member.getAge(), member.getUsername(),
              member.getUserid());
146         }
147
148         @Override
149         public int delete(String userid) {
150           String sql = "DELETE FROM Member WHERE userid = ?";
151           return this.jdbcTemplate.update(sql, userid);
152         }
153       }
154
155
156   6. Controller
157     1)com.example.biz > right-click > New > Class
158     2)Name : MainController
159
160       package com.example.biz;
161
162       import org.springframework.beans.factory.annotation.Autowired;
163       import org.springframework.stereotype.Controller;
164       import org.springframework.ui.Model;
165       import org.springframework.web.bind.annotation.GetMapping;
166       import org.springframework.web.bind.annotation.PostMapping;
167
168       import com.example.dao.MemberDao;
169       import com.example.vo.MemberVO;
```

```
170
171     @Controller
172     public class MainController {
173       @Autowired
174       private MemberDao memberDao;
175
176       @GetMapping("/")
177       public String index() {
178         return "index"; // templates/index.html
179       }
180
181       @PostMapping("/member")
182       public String insert(MemberVO member) {
183         this.memberDao.create(member);
184         return "redirect:/member";
185       }
186
187       @GetMapping("/member")
188       public String list(Model model) {
189         model.addAttribute("members", this.memberDao.readAll());
190         return "list"; // templates/list.html
191       }
192     }
193
194
195 7. static resources 준비
196   1)src/main/resources/static/images folder 생성
197     -spring-boot.png 추가할 것
198   2)src/main/resources/static/js folder 생성
199     -jquery-3.5.0.min.js 추가할 것
200   3)src/main/resources/static/css folder 생성
201     -style.css
202
203       @charset "UTF-8";
204       body {
205         background-color:yellow;
206       }
207       h1{
208         color : blue;
209       }
210
211
212 8. templates/index.html 생성
213
214   <!DOCTYPE html>
215   <html>
216   <head>
217   <meta charset="UTF-8" />
218   <title>New User Insertion Page</title>
219   <link rel="stylesheet" type="text/css" href="/css/style.css">
220   <script src="/js/jquery-3.5.0.min.js"></script>
221   <script>
222     $(document).ready(function() {
223       alert("Hello, Spring Boot!");
224     });
225   </script>
226   </head>
227   <body>
```

```
    <img src="/images/spring-boot.jpg" />
    <h1>New User Insertion Page</h1>
    <form action="/member" method="post">
      <ul>
        <li>ID : <input type="text" name="userid" /></li>
        <li>Name : <input type="text" name="username" /></li>
        <li>Age : <input type="number" name="age" /></li>
        <li><button>Submit</button></li>
      </ul>
    </form>
  </body>
  </html>
```

9. templates/list.html 생성

```
  <!DOCTYPE html>
  <html xmlns:th="http://www.thymeleaf.org">
  <head>
  <meta charset="UTF-8" />
  <title>회원정보</title>
  <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>회원명부</h1>
    <table border='1'>
      <thead>
        <tr>
          <th>아이디</th>
          <th>이름</th>
          <th>나이</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="member : ${members}">
          <td th:text="${member.userid}"></td>
          <td th:text="${member.username}"></td>
          <td th:text="${member.age}"></td>
```

```
        </tr>
      </tbody>
    </table>
  </body>
  </html>


10. src/main/java/com.example.biz/SpringBootJdbcDemoApplication.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")
  public class SpringBootJdbcDemoApplication {

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


11. project > right-click > Run As > Spring Boot App
12. http://localhost:8080



-----------------------------------------------------
Task2. Spring Boot에서 MyBatis 사용하기
1. Spring Boot project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음의 각 항목의 값을 입력후 Next 클릭
    -Service URL :http://start.spring.io
    -Name : SpringBootMyBatisDemo
    -Type : Maven
    -Packaging : Jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : SpringBootMyBatisDemo
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz

  3)다음 각 항목을 선택후 Finish 클릭
    -Spring Boot Version : 2.2.6
    -Select
      --SQL > check Oracle Driver, MyBatis Framework
      --Web > Spring Web
      --Developer Tools > Spring Boot DevTools, Lombok, Spring Configuration Processor
      --Template Engines > Thymeleaf
    -Finish


2. src/main/resources/application.properties
```

```
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE    <--반드시 넣을 것, 매우 주의
spring.datasource.hikari.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.hikari.jdbc-url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.hikari.username=hr
spring.datasource.hikari.password=hr
spring.datasource.hikari.connection-timeout=20000
spring.datasource.hikari.minimum-idle=5
spring.datasource.hikari.maximum-pool-size=20
spring.datasource.hikari.idle-timeout=300000
spring.datasource.hikari.max-lifetime=1200000
spring.datasource.hikari.auto-commit=true


3. DataSource 설정하기
  1)src/main/java > right-click > New > package
  2)Name : com.example.config
  3)Finish
  4)com.example.config > right-click > New > Class
  5)Name : DatabaseConfiguration
  6)Finish

    package com.example.config;

    import javax.sql.DataSource;

    import org.springframework.boot.context.properties.ConfigurationProperties;
    import org.springframework.context.annotation.Bean;
    import org.springframework.context.annotation.Configuration;
    import org.springframework.context.annotation.PropertySource;

    import com.zaxxer.hikari.HikariConfig;
    import com.zaxxer.hikari.HikariDataSource;

    import lombok.extern.slf4j.Slf4j;

    @Slf4j
    @Configuration
    @PropertySource("classpath:/application.properties")
    public class DatabaseConfiguration {

      @Bean
      @ConfigurationProperties(prefix="spring.datasource.hikari")
      public HikariConfig hikariConfig() {
        return new HikariConfig();
      }

      @Bean
      public DataSource dataSource() throws Exception{
        DataSource dataSource = new HikariDataSource(hikariConfig());
        log.info("datasource : {}", dataSource);
        return dataSource;
      }
    }


4. src/main/java/com.example.biz.SpringbootMyBatisDemoApplication

  package com.example.biz;
```

```
402
403   import org.springframework.boot.SpringApplication;
404   import org.springframework.boot.autoconfigure.SpringBootApplication;
405   import org.springframework.context.annotation.ComponentScan;
406
407   @SpringBootApplication
408   @ComponentScan(basePackages = "com.example")
409   public class SpringBootMyBatisDemoApplication {
410
411     public static void main(String[] args) {
412       SpringApplication.run(SpringBootMyBatisDemoApplication.class, args);
413     }
414   }
415
416
417 5. Connection Test
418   1)이제까지 설정한 것은 hikariConfig에서 설정한 정보를 가지고 DataSoure를 생성한 것이다.
419   2)설정이 제대로 된 것인지 Project를 실행해 보자.
420   3)logger에 의해 dataSource 정보를 확인할 수 있다.
421
422   4)Project > Run As > Spring Boot App
423
424     2020-04-21 23:50:38.033  INFO 8420 --- [  restartedMain]
      com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
425     2020-04-21 23:50:38.037  WARN 8420 --- [  restartedMain]
      com.zaxxer.hikari.util.DriverDataSource  : Registered driver with
      driverClassName=oracle.jdbc.driver.OracleDriver was not found, trying direct instantiation.
426     2020-04-21 23:50:38.252  INFO 8420 --- [  restartedMain]
      com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
427     2020-04-21 23:50:38.252  INFO 8420 --- [  restartedMain]
      c.example.config.DatabaseConfiguration   : datasource : HikariDataSource (HikariPool-1)
428
429
430 6. Mabatis 연동하기
431   1)com.example.config.DatabaseConfiguration.java 수정하기
432
433     package com.example.config;
434
435     import javax.sql.DataSource;
436
437     import org.apache.ibatis.session.SqlSessionFactory;
438     import org.mybatis.spring.SqlSessionFactoryBean;
439     import org.mybatis.spring.SqlSessionTemplate;
440     import org.springframework.beans.factory.annotation.Autowired;
441     import org.springframework.boot.context.properties.ConfigurationProperties;
442     import org.springframework.context.ApplicationContext;
443     import org.springframework.context.annotation.Bean;
444     import org.springframework.context.annotation.Configuration;
445     import org.springframework.context.annotation.PropertySource;
446     import org.springframework.core.io.Resource;
447     import org.springframework.core.io.support.PathMatchingResourcePatternResolver;
448
449     import com.zaxxer.hikari.HikariConfig;
450     import com.zaxxer.hikari.HikariDataSource;
451
452     import lombok.extern.slf4j.Slf4j;
453
454     @Slf4j
```

```
455     @Configuration
456     @PropertySource("classpath:/application.properties")
457     public class DatabaseConfiguration {
458       @Autowired
459       private ApplicationContext applicationContext;
460
461       @Bean
462       @ConfigurationProperties(prefix="spring.datasource.hikari")
463       public HikariConfig hikariConfig() {
464         return new HikariConfig();
465       }
466
467       @Bean
468       public DataSource dataSource() throws Exception{
469         DataSource dataSource = new HikariDataSource(hikariConfig());
470         //System.out.println(dataSource.toString());
471         log.info("datasource : {}", dataSource);
472         return dataSource;
473       }
474
475       @Bean
476       public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception{
477         SqlSessionFactoryBean sqlSessionFactory = new SqlSessionFactoryBean();
478         sqlSessionFactory.setDataSource(dataSource);
479         Resource configLocation = new
           PathMatchingResourcePatternResolver().getResource("classpath:/mapper/mybatis-conf
           ig.xml");
480         sqlSessionFactory.setMapperLocations(applicationContext.getResources("classpath:/m
           apper/mybatis-mapper.xml"));
481         sqlSessionFactory.setConfigLocation(configLocation);
482         return sqlSessionFactory.getObject();
483       }
484
485       @Bean
486       public SqlSessionTemplate sqlSessioinTemplate(SqlSessionFactory sqlSessionFactory) {
487         return new SqlSessionTemplate(sqlSessionFactory);
488       }
489     }
490
491   2)Mapper Folder 생성하기
492     -Mapper Folder에는 Application에서 사용할 Query를 담고 있는 XML 파일을 저장한다.
493     -src/main/resources/mapper folder 생성한다.
494     -src/main/resources > right-click > New > Folder
495     -Folder name : mapper
496     -Finish
497
498   3)src/main/resources/mapper/mybatis-config.xml 생성
499
500     <?xml version="1.0" encoding="UTF-8"?>
501     <!DOCTYPE configuration
502       PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
503         "http://mybatis.org/dtd/mybatis-3-config.dtd">
504     <configuration>
505     </configuration>
506
507   4)src/main/resources/mapper/mybatis-mapper.xml 생성
508
509     <?xml version="1.0" encoding="UTF-8"?>
```

```
510     <!DOCTYPE mapper
511       PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
512         "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
513     <mapper namespace="com.example.vo.UserVO">
514     </mapper>
515
516   5)MyBatis 연결 Test
517     -src/test/java/com.example.biz.SpringBootMyBatisDemoApplicationTests.java
518
519       package com.example.biz;
520
521       import org.junit.jupiter.api.Test;
522       import org.mybatis.spring.SqlSessionTemplate;
523       import org.springframework.beans.factory.annotation.Autowired;
524       import org.springframework.boot.test.context.SpringBootTest;
525
526       import lombok.extern.slf4j.Slf4j;
527
528       @Slf4j
529       @SpringBootTest
530       class SpringBootMyBatisDemoApplicationTests {
531         @Autowired
532         private SqlSessionTemplate sqlSession;
533
534         @Test
535         void contextLoads() {
536         }
537
538         @Test
539         public void test1() throws Exception{
540           log.info("SqlSession = " + this.sqlSession);
541         }
542       }
543
544     -Run As > JUnit Test
545       Green Bar
546       2020-04-22 00:45:48.019  INFO 17336 --- [         main]
          .b.SpringBootMyBatisDemoApplicationTests : Started
          SpringBootMyBatisDemoApplicationTests in 4.945 seconds (JVM running for 7.697)
547       2020-04-22 00:45:48.488  INFO 17336 --- [         main]
          .b.SpringBootMyBatisDemoApplicationTests : SqlSession =
          org.mybatis.spring.SqlSessionTemplate@40faff12
548
549
550 7. 모든 회원정보 가져오기
551   1)MainController.java
552
553     package com.example.biz;
554
555     import org.springframework.beans.factory.annotation.Autowired;
556     import org.springframework.stereotype.Controller;
557     import org.springframework.ui.Model;
558     import org.springframework.web.bind.annotation.GetMapping;
559     import org.springframework.web.bind.annotation.PostMapping;
560
561     import com.example.dao.MemberDao;
562     import com.example.vo.MemberVO;
563
```

```
564     @Controller
565     public class MainController {
566       @Autowired
567       private MemberDao memberDao;
568
569       @GetMapping("/")
570       public String index() {
571         return "index"; // templates/index.html
572       }
573
574       @PostMapping("/member")
575       public String insert(MemberVO member) {
576         this.memberDao.create(member);
577         return "redirect:/member";
578       }
579
580       @GetMapping("/member")
581       public String list(Model model) {
582         model.addAttribute("members", this.memberDao.readAll());
583         return "list"; // templates/list.html
584       }
585     }
586
587   2)MemberDaoImpl.java
588     package com.example.dao;
589
590     import java.util.List;
591
592     import org.apache.ibatis.session.SqlSession;
593     import org.springframework.beans.factory.annotation.Autowired;
594     import org.springframework.stereotype.Repository;
595
596     import com.example.vo.MemberVO;
597
598     @Repository
599     public class MemberDaoImpl implements MemberDao {
600       @Autowired
601       private SqlSession sqlSession;
602
603       @Override
604       public int create(MemberVO member) {
605         return 0;
606       }
607
608
609       @Override
610       public List<MemberVO> readAll() {
611         return this.sqlSession.selectList("selectAll");
612       }
613
614       @Override
615       public MemberVO read(String userid) {
616         return null;
617       }
618
619       @Override
620       public int update(MemberVO member) {
621         return 0;
```

```
622         }
623
624         @Override
625         public int delete(String userid) {
626           return 0;
627         }
628
629       }
630
631     3)mybatis-mapper.xml
632
633       <?xml version="1.0" encoding="UTF-8"?>
634       <!DOCTYPE mapper
635         PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
636           "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
637       <mapper namespace="com.example.vo.MemberVO">
638         <resultMap id="memberVOResult" type="memberVO">
639           <result property="userid" column="userid" />
640             <result property="username" column="username" />
641             <result property="age" column="age" />
642         </resultMap>
643         <select id="selectAll" resultType="memberVO" resultMap="memberVOResult">
644           SELECT * FROM Member
645         </select>
646       </mapper>
```