5.마이바티스를 이용한 스프링

1) 마이바티스 라이브러리

① 메이븐으로 스프링 연동 모듈관리

메이븐은 라이브러리의 groupId와 artifactId를 명시해주면 해당 라이브러리뿐 아니라 추가로 필요한 라이브러리를 모두 자동으로 가져온다.

```
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.2.1</version>
</dependency>
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>1.2.0</version>
</dependency>
```

pom.xml

```
01 <project xmlns="http://maven.apache.org/POM/4.0.0"
02         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
04                 http://maven.apache.org/xsd/maven-4.0.0.xsd">
05         <modelVersion>4.0.0</modelVersion>
06         <groupId>sp4</groupId>
07         <artifactId>mybatis-spring</artifactId>
08         <version>0.0.1-SNAPSHOT</version>
09         <packaging>war</packaging>
10         <dependencies>
11                 <dependency>
12                         <groupId>javax.servlet.jsp</groupId>
13                         <artifactId>jsp-api</artifactId>
14                         <version>2.2</version>
15                         <scope>provided</scope>
16                 </dependency>
17                 <dependency>
18                         <groupId>javax.servlet</groupId>
19                         <artifactId>javax.servlet-api</artifactId>
20                         <version>3.0.1</version>
21                         <scope>provided</scope>
22                 </dependency>
23                 <dependency>
24                         <groupId>javax.servlet</groupId>
25                         <artifactId>jstl</artifactId>
```

```
26                          <version>1.2</version>
27                  </dependency>
28                  <dependency>
29                          <groupId>org.springframework</groupId>
30                          <artifactId>spring-webmvc</artifactId>
31                          <version>4.1.0.RELEASE</version>
32                  </dependency>
33
34                  <dependency>
35                          <groupId>org.springframework</groupId>
36                          <artifactId>spring-jdbc</artifactId>
37                          <version>4.1.0.RELEASE</version>
38                  </dependency>
39                  <dependency>
40                          <groupId>com.mchange</groupId>
41                          <artifactId>c3p0</artifactId>
42                          <version>0.9.2.1</version>
43                  </dependency>
44                  <dependency>
45                          <groupId>log4j</groupId>
46                          <artifactId>log4j</artifactId>
47                          <version>1.2.17</version>
48                  </dependency>
49
50          <dependency>
51                  <groupId>org.mybatis</groupId>
52                  <artifactId>mybatis</artifactId>
53                  <version>3.2.1</version>
54          </dependency>
55          <dependency>
56                  <groupId>org.mybatis</groupId>
57                  <artifactId>mybatis-spring</artifactId>
58                  <version>1.2.0</version>
59          </dependency>
60      <dependency>
61        <groupId>junit</groupId>
62        <artifactId>junit</artifactId>
63        <version>3.8.1</version>
64        <scope>test</scope>
65      </dependency>
66  </dependencies>
```

```
67
68
69          <build>
70                  <plugins>
71                          <plugin>
72                                  <artifactId>maven-compiler-plugin</artifactId>
73                                  <version>3.1</version>
74                                  <configuration>
75                                          <source>1.7</source>
76                                          <target>1.7</target>
77                                          <encoding>utf-8</encoding>
78                                  </configuration>
79                          </plugin>
80                  </plugins>
81          </build>
82
83  </project>
```

WEB-INF\web.xml

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03          xmlns="http://java.sun.com/xml/ns/javaee"
04          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
05                  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
06          version="3.0">
07
08          <servlet>
09                  <servlet-name>dispatcher</servlet-name>
10                  <servlet-class>
11                          org.springframework.web.servlet.DispatcherServlet
12                  </servlet-class>
13                  <init-param>
14                          <param-name>contextConfigLocation</param-name>
15                          <param-value>
16                                  classpath:applicationContext.xml
17                                  classpath:applicationController.xml
18                                  classpath:applicationMember.xml
19                          </param-value>
20                  </init-param>
21                  <load-on-startup>1</load-on-startup>
22          </servlet>
```

```
23
24          <servlet-mapping>
25                  <servlet-name>dispatcher</servlet-name>
26                  <url-pattern>/</url-pattern>
27          </servlet-mapping>
28
29          <filter>
30                  <filter-name>encodingFilter</filter-name>
31                  <filter-class>
32                          org.springframework.web.filter.CharacterEncodingFilter
33                  </filter-class>
34                  <init-param>
35                          <param-name>encoding</param-name>
36                          <param-value>UTF-8</param-value>
37                  </init-param>
38          </filter>
39          <filter-mapping>
40                  <filter-name>encodingFilter</filter-name>
41                  <url-pattern>/*</url-pattern>
42          </filter-mapping>
43
44  </web-app>
```

src\applicationController.xml

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans xmlns="http://www.springframework.org/schema/beans"
03          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04          xmlns:context="http://www.springframework.org/schema/context"
05          xsi:schemaLocation="http://www.springframework.org/schema/beans
06          http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
07          http://www.springframework.org/schema/context
08          http://www.springframework.org/schema/context/spring-context-3.1.xsd">
09
10
11          <bean class="controller.CommentController" >
12          </bean>
13  </beans>
```

src\applicationMember.xml

```
01  <?xml version="1.0" encoding="UTF-8"?>
```

```
02 <beans xmlns="http://www.springframework.org/schema/beans"
03         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04         xmlns:context="http://www.springframework.org/schema/context"
05         xsi:schemaLocation="http://www.springframework.org/schema/beans
06         http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
07         http://www.springframework.org/schema/context
08         http://www.springframework.org/schema/context/spring-context-3.1.xsd">
09
10         <bean id="commentSessionRepository" class="repository.CommentSessionRepository"/>
11
12         <bean id="commentService" class="service.CommentService" />
13
14 </beans>
```

src\applicationContext.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03         xmlns:mvc="http://www.springframework.org/schema/mvc"
04         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
05         xsi:schemaLocation="http://www.springframework.org/schema/beans
06                 http://www.springframework.org/schema/beans/spring-beans.xsd
07                 http://www.springframework.org/schema/mvc
08                 http://www.springframework.org/schema/mvc/spring-mvc.xsd">
09
10         <mvc:annotation-driven />
11
12         <bean
13  class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor"
14 />
15
16         <mvc:view-resolvers>
17                 <mvc:jsp prefix="/WEB-INF/view/" />
18         </mvc:view-resolvers>
19
20         <bean id="messageSource"
21                 class="org.springframework.context.support.ResourceBundleMessageSource">
22                 <property name="basenames">
23                         <list>
24                                 <value>message.label</value>
25                         </list>
```

```
                </property>
                <property name="defaultEncoding" value="UTF-8" />
        </bean>

</beans>
```

② 스프링의 데이터베이스 관련 설정

스프링의 데이터베이스 관련 설정(applicationContext.xml)

src\mybatis.properties

```
jdbc.url=jdbc:oracle:thin:@localhost:1521:XE
```

src\mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
        <!-- 외부 프로퍼티 파일 로드및 공통 프로퍼티 정의 -->
        <properties resource="mybatis.properties">
        <!--properties url="file:d:\mybatis.properties"-->
          <property name="jdbc.driver" value="oracle.jdbc.driver.OracleDriver"/>
          <property name="jdbc.username" value="smrit"/>
          <property name="jdbc.password" value="oracle"/>
        </properties>

        <!-- 마이바티스의 작동 규칙정의 -->
        <settings>
                <setting name="cacheEnabled" value="false" />
                <setting name="useGeneratedKeys" value="true" />
                <setting name="mapUnderscoreToCamelCase" value="true" />
                <setting name="autoMappingBehavior" value="PARTIAL" />
        </settings>

        <!-- 타입별칭 -->
        <typeAliases>
                <typeAlias type="model.Comment" alias="Comment" />
        </typeAliases>

        <!-- 데이터베이스및 트랜잭션 관리자 -->
        <environments default="development">
```

```
29                  <environment id="development">
30                          <transactionManager type="JDBC" />
31                          <dataSource type="POOLED">
32                                  <property name="driver" value="${jdbc.driver}" />
33                                  <property name="url" value="${jdbc.url}" />
34                                  <property name="username" value="${jdbc.username}" />
35                                  <property name="password" value="${jdbc.password}" />
36                          </dataSource>
37                  </environment>
38          </environments>
39
40          <!-- 매퍼정의 -->
41          <mappers>
42                  <mapper resource="repository/mapper/CommentMapper.xml" />
43          </mappers>
44  </configuration>
```

Comment.java

```
01  package model;
02
03  import java.util.Date;
04
05  public class Comment {
06          private Long commentNo;
07          private String userId;
08          private Date regDate;
09          private String commentContent;
10
11          public Comment(Long commentNo, String userId, Date regDate,
12                          String commentContent) {
13                  this.commentNo = commentNo;
14                  this.userId = userId;
15                  this.regDate = regDate;
16                  this.commentContent = commentContent;
17          }
18          public Comment() {
19          }
20          public Long getCommentNo() {
21                  return commentNo;
22          }
```

```
23
24          public void setCommentNo(Long commentNo) {
25                  this.commentNo = commentNo;
26          }
27
28          public String getUserId() {
29                  return userId;
30          }
31
32          public void setUserId(String userId) {
34                  this.userId = userId;
35          }
36
37          public Date getRegDate() {
38                  return regDate;
39          }
40
41          public void setRegDate(Date regDate) {
42                  this.regDate = regDate;
43          }
44
45          public String getCommentContent() {
46                  return commentContent;
47          }
48
49          public void setCommentContent(String commentContent) {
50                  this.commentContent = commentContent;
51          }
52
53  }
```

CommentMapper.xml

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
03  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
04
05  <mapper namespace="repository.mapper.CommentMapper">
06    <cache />
07
08    <select id="selectCommentByPrimaryKey" parameterType="long" resultType="Comment">
09          SELECT
```

```
10                  comment_no,
11                  user_id,
12                  comment_content,
13                  reg_date
14          FROM comment1
15          WHERE comment_no = #{commentNo}
16    </select>
17    <resultMap id="baseResultMap" type="comment">
18          <id column="comment_no"  jdbcType="BIGINT" property="commentNo" />
19          <result  column="user_id" jdbcType="VARCHAR" property="userId" />
20          <result column="reg_date" jdbcType="TIMESTAMP" property="regDate" />
21          <result  column="comment_content"  jdbcType="VARCHAR"  property="commentContent"
22  />
23    </resultMap>
24    <select id="selectCommentByCondition" parameterType="long"  resultMap="baseResultMap">
25      SELECT
26                  comment_no,
27                  user_id,
28                  comment_content,
29                  reg_date
30          FROM comment1
31          <where>
32                  <if test="commentNo != null">
33                          comment_no = #{commentNo}
34                  </if>
35          </where>
36  <!--
37          <if test="commentNo!=null">
38                  where  comment_no = #{commentNo}
39          </if>
40  -->
41    </select>
42    <insert id="insertComment" parameterType="Comment">
43      INSERT INTO comment1(comment_no, user_id, comment_content, reg_date)
44          VALUES (#{commentNo}, #{userId}, #{commentContent}, #{regDate})
45    </insert>
46    <update id="updateComment" parameterType="Comment">
47                  UPDATE comment1
48                          <set>
49                          <if test="commentContent != null">
50                                  comment_content = #{commentContent},
```

```
51                          </if>
52                          <if test="userId != null">
53                                  user_id = #{userId},
54                          </if>
55                          <if test="regDate != null">
56                                  reg_date = #{regDate}
57                          </if>
58                          </set>
59                  WHERE comment_no = #{commentNo}
60    </update>
61    <delete id="deleteComment" parameterType="long">
62      DELETE FROM comment1
63          WHERE comment_no = #{commentNo}
64    </delete>
65  </mapper>
```

17행 ~ 23행 : 매핑규칙 정의를 설명하기 위한 간단한 매핑구문
24행 ~ 41행 : 매핑규칙 정의를 설명하기 위한 간단한 매핑구문

• 결과 매핑의 아이디는 BaseResultMap이고, 24행 ~ 41행의 매핑 구문의 resultMap속성에서 사용한다.
대상 클래스 타입은 "ldg.mybatis.model.Comment"다.
결과매핑을 상용해 반환되는 타입은 댓글 또는 댓글을 갖는 List이다.

• id 엘리먼트는 기본키의 컬럼을 지정할 때 사용한다. 댓글 테이블의 기본 키는 댓글 번호인 comment_no 컬럼이다.

• comment_no 컬럼은 property 속성을 commentNo로 했기 때문에 댓글 객체의 commentNo 필드에 리플렉션으로 값을 설정하거나 setCommentNo 메소드를 사용해서 값을 설정한다.

• comment_no 컬럼에서 jdbcType속성에 정의한 bigint값을 보고 JDBC가 내부적으로 값을 처리할 때 값의 타입을 bigint로 인식한다.

• result 엘리먼트를 사용한 user_id 컬럼은 property 속성을 userId로 설정했기 때문에 댓글 객체의 userId 필드에 리플렉션으로 값을 설정하거나 setUserId 메소드를 사용해서 값을 설정한다.

• reg_gate 컬럼은 preperty속성을 regDate로 설정했기 때문에 댓글 객체의 regDate 필드에 리플렉션으로 값을 설정하거나 setRegDate메소드를 사용해서 값을 설정한다.

• comment_content 컬럼은 preperty속성을commentContent로 설정했기 때문에 댓글 객체의 commentContent 필드에 리플렉션으로 값을 설정하거나 setCommentContent메소드를 사용해서 값을 설정한다.

이렇게 resultMap 엘리먼트에 컬럼과 대상필드 및 setter 메소드의 규칙을 정의해주면 컬럼명과 필드명이 다르더라도 값을 설정할 수 있다.

AbstractRepository.java

```
01 package repository;
02
03 import java.io.IOException;
04 import java.io.InputStream;
05
06 import org.apache.ibatis.io.Resources;
07 import org.apache.ibatis.session.SqlSessionFactory;
08 import org.apache.ibatis.session.SqlSessionFactoryBuilder;
09
10 public abstract class AbstractRepository {
11         private static SqlSessionFactory sqlSessionFactory;
12
13         static {
14                 setSqlSessionFactory();
15         }
16
17         private static void setSqlSessionFactory() {
18                 String resource = "mybatis-config.xml";
19                 InputStream inputStream;
20                 try {
21                         inputStream = Resources.getResourceAsStream(resource);
22                 } catch (IOException e) {
23                         throw new IllegalArgumentException(e);
24                 }
25                 sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
26         }
27
28         protected SqlSessionFactory getSqlSessionFactory() {
29                 return sqlSessionFactory;
30         }
31 }
```

CommentSessionRepository.java

```
01 package repository;
02
03 import java.util.Calendar;
04 import java.util.List;
05
06 import org.apache.ibatis.session.SqlSession;
07 import org.springframework.stereotype.Repository;
```

```
08
09 import model.Comment;
10
11 @Repository
12 public class CommentSessionRepository extends AbstractRepository {
13         private final String namespace = "repository.mapper.CommentMapper";
14
15         public Integer deleteComment(Long commentNo){
16                 SqlSession sqlSession = getSqlSessionFactory().openSession();
17                 try{
18                 int result =
19
20 sqlSession.delete(String.format("%s.deleteComment",namespace),commentNo);
21                                 if(result > 0) sqlSession.commit();
22                                 return result;
23                 }finally{
24                         sqlSession.close();
25                 }
26
27         }
28
29         public Integer  insertComment(Comment comment){
30                 SqlSession sqlSession = getSqlSessionFactory().openSession();
31                 try{
32                         String statement = namespace + ".insertComment";
33                         comment.setRegDate(Calendar.getInstance().getTime());
34                         System.out.println("2222     " + comment.getCommentNo());
35                         System.out.println("2222     " + comment.getUserId());
36                         System.out.println("2222     " + comment.getCommentContent());
37                         System.out.println("2222     " + comment.getRegDate());
38                         Integer result= sqlSession.insert(statement, comment);
39                         if(result > 0) sqlSession.commit();
40                         return result;
41                 }finally{
42                         sqlSession.close();
43                 }
44         }
45
46         public List<Comment> selectCommentByCondition(Comment comment) {
47                 SqlSession sqlSession = getSqlSessionFactory().openSession();
48                 try{
```

```
49                  String statement = namespace + ".selectCommentByCondition";
50                  System.out.println("2222      " + comment.getCommentNo());
51                  return sqlSession.selectList( statement , comment);
52                  }finally{
53                          sqlSession.close();
54                  }
55          }
56
57          public Integer updateComment(Comment comment){
58                  SqlSession sqlSession = getSqlSessionFactory().openSession();
59                  try{
60                          comment.setRegDate(Calendar.getInstance().getTime());
61                          String statement = namespace + ".updateComment";
62                          System.out.println("2222      " + comment.getCommentNo());
63                          System.out.println("2222      " + comment.getUserId());
64                          System.out.println("2222      " + comment.getCommentContent());
65                          System.out.println("2222      " + comment.getRegDate());
66                          Integer result =  sqlSession.insert(statement, comment);
67                          if(result > 0) sqlSession.commit();
68                          return result;
69                  }finally{
70                          sqlSession.close();
71                  }
72          }
73  }
```

1.@Repositiry

리파지토리 클래스는 데이터 접근 객체(DAO)의 역할을 담당하기 때문에 데이터베이스와 연동해서 데이터를 가져오거나 입력, 수정한다.

스프링을 사용할 때 가장 큰 장점은 실질적으로 데이터를 다루는 작업만 집중할 수 있게 해준다. 설정 파일을 통해 개발자가 매번 데이터베이스설정을 사용해서 연결을 가져오거나 트랜잭션을 관리하지 않아도 되게 내부적으로 알아서 처리해준다.

2.SqlSessionTemplate

SqlSessionTemplate 빈은 스프링과 연동한 마이바티스 기능을 사용하기 위해서는 리파지토리 클래스에서 SqlSessionTemplate 빈을 반드시 선언하고 사용해야 한다.

CommentService.java

```
01  package service;
02
03  import java.util.List;
04
```

```
05 import org.springframework.beans.factory.annotation.Autowired;
06 import org.springframework.stereotype.Service;
07
08 import model.Comment;
09 import repository.CommentSessionRepository;
10
11 @Service
12 public class CommentService {
13         @Autowired
14         private CommentSessionRepository commentRepository;
15
16         public List<Comment> selectComment(Comment comment){
17                 System.out.println("111111  " + comment.getCommentNo());
18                 return commentRepository.selectCommentByCondition(comment);
19         }
20         public Integer updateComment(Comment comment){
21                 System.out.println("service   " + comment.getCommentNo());
22                 return commentRepository.updateComment(comment);
23         }
24
25         public Integer insertComment(Comment comment) {
26                 System.out.println("service   " + comment.getCommentNo());
27                 System.out.println("service   " + comment.getUserId());
28                 System.out.println("service   " + comment.getCommentContent());
29                 System.out.println("service   " + comment.getRegDate());
30                 return commentRepository.insertComment(comment);
31         }
32         public Integer deleteComment(Long commentNo){
33                 System.out.println("service   " + commentNo);
34                 return commentRepository.deleteComment(commentNo);
35         }
36 }
```

1. @Service : 웹 애플리케이션을 작성해보면 사용자의 입력을 받는 컨트롤러 클래스와 데이터베이스를 처리하는 클래스 사이에 비즈니스 로직이나 트랜잭션을 처리하는 클래스를 둔다. 스프링은 이 역할을 담당하는 클래스를 서비스 클래스로 정의하고 @Service 애노테이션을 사용하여 선언한다.

2. @Transaction : 서비스 클래스가 주로 담당하는 역할은 비즈니스 로직을 처리하거나 트랜잭션을 처리하는 것이다. 스프링은 트랜잭션을 처리하기 위해 코드를 사용해 처리하는 방법도 제공하지만, 좀더 편리하게 하기 위해 애노테이션만 지정하면 자동으로 트랜잭션을 제어하게 기능을 제공하고 있다.
이렇게 애노테이션을 사용해서 트랜잭션을 처리할 때 사용하는 것이 @Transactional 애노테이션이다.

CommentController.java

```
01 package controller;
02
03 import java.util.List;
04
05 import org.springframework.beans.factory.annotation.Autowired;
06 import org.springframework.stereotype.Controller;
07 import org.springframework.ui.Model;
08 import org.springframework.web.bind.annotation.RequestMapping;
09 import org.springframework.web.bind.annotation.RequestMethod;
10
11 import model.Comment;
12 import service.CommentService;
13
14 @Controller
15 public class CommentController {
16         @Autowired
17         private CommentService commentService;
18
19         @RequestMapping(value="/form.do", method = RequestMethod.GET)
20         public String handleStep1(Model model) {
21                 model.addAttribute("comment", new Comment());
22                 return "jsp/comment_form";
23         }
24
25         @RequestMapping(value="/comment_insert", method = RequestMethod.POST)
26         public String handleStep1(Comment comment, Model model) {
27                 Integer result = commentService.insertComment(comment);
28                 model.addAttribute("result",result);
29                 return "jsp/comment_insert";
30         }
31
32
33         @RequestMapping(value="/comment_select", method = RequestMethod.POST)
34         public String handleStep5(Comment comment, Model model) {
35                 System.out.println(comment.getCommentNo());
36                 List<Comment> result = commentService.selectComment(comment);
37                 model.addAttribute("result", result);
38                 return "jsp/comment_select";
39         }
40
```

```
41          @RequestMapping(value="/comment_update", method = RequestMethod.POST)
42          public String handleStep2(Comment comment, Model model){
43
44                  Integer result = commentService.updateComment(comment);
45                  model.addAttribute("result" , result);
46                  return "jsp/comment_update";
47          }
48
49          @RequestMapping(value="/comment_delete", method = RequestMethod.POST)
50          public String handleStep3(Comment comment, Model model){
51                  Integer result = commentService.deleteComment(comment.getCommentNo());
52                  model.addAttribute("result" , result);
53                  return "jsp/comment_delete";
54          }
55  }
```

src\message\label.properties

```
01  commentInsert = 댓글 저장
02  commentNo= 댓글 번호
03  userId= 사용자 아이디
04  commentContent= 댓글 내용
05  Insert.btn= 저장버튼
06  commentDelete= 댓글 삭제
07  Delete.btn= 삭제버튼
08  commentUpdate= 댓글 수정
09  Update.btn= 수정버튼
10  commentSelect= 댓글 검색
11  select.btn= 검색버튼
```

WEB-INF\view\jsp\comment_form.jsp

```
01  <%@ page language="java" contentType="text/html; charset=EUC-KR"
02      pageEncoding="EUC-KR"%>
03  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
04  <%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
05  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
06  "http://www.w3.org/TR/html4/loose.dtd">
07  <html>
08  <head>
09  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
10  <title>Insert title here</title>
11  </head>
```

```
<body>
	<form:form commandName="comment" action="/mybatis-spring-web/comment_select">
		<label><spring:message code="commentSelect" />:
		<form:input path="commentNo" /><br>
		</label>
		<input type="submit" value="<spring:message code="select.btn" />"><br>
	</form:form>
	<form:form commandName="comment" action="/mybatis-spring-web/comment_update">
		<label><spring:message code="commentUpdate" />:
		<form:input path="commentNo" /><br>
		</label>
		<label><spring:message code="userId" />:
		<form:input path="userId" /><br>
		</label>
		<label><spring:message code="commentContent" />:
		<form:input path="commentContent" /><br>
		</label>
		<input type="submit" value="<spring:message code="Update.btn" />"><br>
	</form:form>

	<form:form commandName="comment" action="/mybatis-spring-web/comment_insert">
		<label><spring:message code="commentInsert" />:
		<form:input path="commentNo" /><br>
		</label>
		<label><spring:message code="userId" />:
		<form:input path="userId" /><br>
		</label>
		<label><spring:message code="commentContent" />:
		<form:input path="commentContent" /><br>
		</label>
		<input type="submit" value="<spring:message code="Insert.btn" />"><br>
	</form:form>
	<form:form commandName="comment" action="/mybatis-spring-web/comment_delete">
		<label><spring:message code="commentDelete" />:
		<form:input path="commentNo" /><br>
		</label>
		<input type="submit" value="<spring:message code="Delete.btn" />"><br>
	</form:form>
</body>
</html>
```

WEB-INF\view\jsp\comment_select.jsp

```
<%@page
import="java.io.*,javax.servlet.*,java.util.*,model.*,org.springframework.context.*,org.springfram
ework.context.support.*" contentType="text/html; charset=utf-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!doctype html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Document</title>
 </head>
 <body>
<c:forEach var="comment" items="${result}" varStatus="status">
${status.index + 1}번 <BR>
댓글번호 : ${comment.commentNo}<br>
작성자아이디 : ${comment.userId}<br>
작성일시 : ${comment.regDate}<br>
댓글내용 : ${comment.commentContent}<br><br>
</c:forEach>
 </body>
</html>
```

WEB-INF\view\jsp\comment_insert.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
${result}
```

WEB-INF\view\jsp\comment_delete.jsp

```
<%@page import="java.io.*,javax.servlet.*" contentType="text/html; charset=utf-8"%>
${result}
```

WEB-INF\view\jsp\comment_update.jsp

```
<%@page import="java.io.*,javax.servlet.*" contentType="text/html; charset=utf-8"%>
${result}
```

2) 스프링 JDBC 사용

· SQL 작성

· 파라미터를 선언하고 파라미터 값을 설정

· 반복문 내에서 별도 처리

① 스프링 JDBC를 사용해서 데이터 조회하기

```
public class CommentJdbcRepository {
	public Comment selectCommentByPrimaryKey(Long commentNo) {
		Assert.notNull(commentNo, "댓글번호가 없습니다.");

		StringBuilder sql = new StringBuilder("");
		sql.append("SELECT comment_no, user_id, comment_content, reg_date FROM COMMENT1 WHERE
comment_no = :commentNo ");

		Map<String, Object> condition = new HashMap<String, Object>();
		condition.put("commentNo", commentNo);
		SqlParameterSource namedParameters = new MapSqlParameterSource(condition);
		RowMapper<Comment>                              rowMapper                              =
ParameterizedBeanPropertyRowMapper.newInstance(Comment.class);
		List<Comment>   result   =   this.jdbcTemplate.query(sql.toString(),   namedParameters,
rowMapper);

		if (!CollectionUtils.isEmpty(result)) {
			return result.get(0);
		}
		return null;
	}
}
```

동적으로 값을 설정하기 위해 MapSqlParameterSource객체를 사용한다.
이객체는 Map을 사용하는 스프링 JDBC의 파라미터 객체이다.
SQL에서 ":Map의 키" 형태로 표현식을 사용하면 Map의 키와 일치하는 값이 자동으로 설정해주는 기능을 한다.
여기서는 :commentNo가 설정 대상이다.
MapSqlParameterSource를 사용해서 SQL에 파라미터를 설정했다면 SQL 을 실행하고 결과를 자바 객체에 설정해야 한다. 스프링JDBC는 결과를 자바객체에 설정하기 위해 RowMapper 를 제공한다.
자바빈에결과를 설정하는 ParameterizedBeanPropertyRowMapperfmf 사용한다.
ParameterizedBeanPropertyRowMapper는 컬럼명에 해당하는 setter메소드를 사용해서 자동으로 값을 설정한 후 객체에 반환하는 역할 을 한다.

컬럼명과 setter메소드의 이름이 동일한 규칙으로 지정돼 있다면 추가로 할 작업은 없다.

② 스프링 JDBC를 사용해서 데이터 입력하기

```
	public Integer insertComment(Comment comment) {
		Assert.notNull(comment, "댓글 데이터가 없습니다.");

		StringBuilder sql = new StringBuilder("");
```

```
                sql.append("INSERT INTO COMMENT1(comment_no, user_id, comment_content, reg_date) ");
                sql.append("VALUES (:commentNo, :userId, :commentContent, :regDate) ");

                SqlParameterSource namedParameters = new BeanPropertySqlParameterSource(comment);
                return this.jdbcTemplate.update(sql.toString(), namedParameters);
        }
```

스프링 JDBC를 사용해서 데이터를 입력하는 코드이다.
StringBuilder 객체를 사용해서 사용할 SQL을 만든다.
자바빈 형태로 파라미터를 설정했기 때문에 BeanPropertySqlParameterSource객체를 사용한다.

③ 마이바티스와 스프링 JDBC를 함께 사용하는 서비스 클래스

```
@Service
public class CommentService {
        @Autowired
        private CommentMapperRepository commentRepository;
        @Autowired
        private CommentJdbcRepository commentJdbcRepository;
        @Transactional
        @SuppressWarnings("unused")
        public Integer withSpringJdbc(Comment comment) {
                Long commentNo = comment.getCommentNo();

                commentJdbcRepository.selectCommentByPrimaryKey(commentNo);
                commentJdbcRepository.deleteComment(commentNo);
                if (true) {
                        throw new IllegalStateException("rollback test");
                }
                return commentRepository.insertComment(comment);
        }
}
```

마이바티스와 스프링 JDBC를 사용하는 두 개의 리파지토리 클래스를 가진다.
마이바티스 클래스는 CommentJdbcRepository 이다.
마이바티스와 스프링 JDBCfmf 함께 사용할 수 있고 구현체가 다르더라도 트랜잭션을 함께 걸어 줄 수 있다.
스프링 JDBC 클래스를 사용해서 데이터 조회, 삭제, 입력이 정상적으로 처리 되어야 하나 삭제한 후 입력할 때 예외가 발생한다면 삭제까지 롤백되어야 한다.