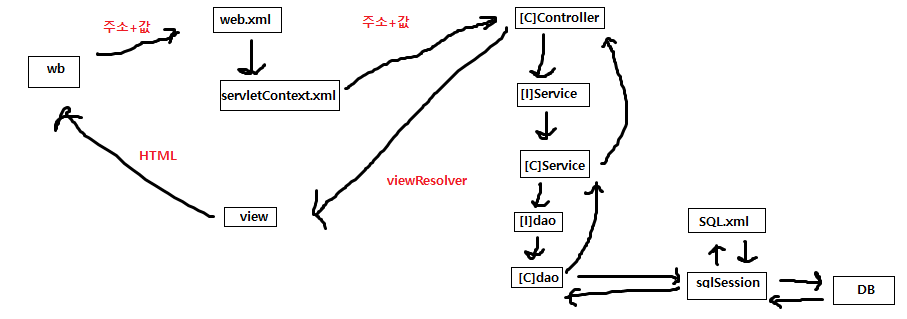


-root-context.xml 추가된 내용

|  |
| --- |
| **<!-- 접속 정보가 있는 파일의 위치 지정 -->**  <bean id="**propertyConfigurer**" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"> <property name="locations">  <list>  <value>**/WEB-INF/spring/jdbc.properties**</value>  </list>  </property> </bean>  **<!-- 접속 정보 객체 -->**  <bean id="**dataSource**" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close"> <property name="**driverClassName**" value="**${jdbc.driverClassName}**" /> <property name="**url**" value="**${jdbc.url}**" /> <property name="**username**" value="**${jdbc.username}**" /> <property name="**password**" value="**${jdbc.password}**" /> </bean>  **<!-- 접속 객체 -->**  <bean id="**sqlSessionFactory**" class="org.mybatis.spring.SqlSessionFactoryBean">  **<!-- java.util.HashMap을 hashmap으로 쓸 수 있도록 alias 설정한 곳 -->**  <property name="**configLocation**" value="classpath:**mybatis-config.xml**" />  <property name="dataSource" ref="dataSourceLog" />  **<!-- mapper 디렉토리 아래에 \_SQL.xml로 끝나는 파일 매핑 -->**  <property name="**mapperLocations**" value="classpath\*:**mapper/\*\*/\*\_SQL.xml**" /> </bean>  **<!-- 접속 활용 객체 -->**  <bean id="**sqlSession**" class="org.mybatis.spring.SqlSessionTemplate">  <constructor-arg ref="sqlSessionFactory" />  </bean> |

-jdbc.properties 파일 내용

|  |  |
| --- | --- |
| **키** | **값** |
| jdbc.driverClassName | oracle.jdbc.driver.OracleDriver |
| jdbc.url | jdbc:oracle:thin:@127.0.0.1:1521:XE |
| jdbc.username | TEST |
| jdbc.password | 1234 |



-Controller, Service, Repository 분리 이유  
1) 용도에 따라 관리포인트를 달리하기 위해  
2) 재사용성을 위해

-컨트롤러 영역(Controller / Service / Dao)

|  |
| --- |
| package com.spring.sample.web.test.controller;  **[TestController.java]** import java.util.HashMap; import java.util.List;  import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.servlet.ModelAndView; import com.spring.sample.web.test.service.ITestService;  **@Controller // 컨트롤러 선언** public class TestController {  **@Autowired // 서비스 인터페이스와 연결  public ITestService iTestService;**    @RequestMapping(value="/test1")  public ModelAndView test1(@RequestParam HashMap<String, String> params,  ModelAndView mav) throws Throwable {  List<HashMap<String,String>> list = iTestService.getList(params);  mav.addObject("list", list);  mav.setViewName("test/test1");  return mav;  } }  **[iTestService.java]** package com.spring.sample.web.test.service; import java.util.HashMap; import java.util.List; public interface ITestService {  List<HashMap<String, String>> getList(HashMap<String, String> params) throws Throwable; }  **[TestService.java]**  package com.spring.sample.web.test.service; import java.util.HashMap; import java.util.List; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import com.spring.sample.web.test.dao.ITestDao;  **@Service // 서비스 선언** public class TestService implements ITestService {  **@Autowired // Dao 인터페이스 연결  public ITestDao iTestDao;**   @Override  public List<HashMap<String, String>> getList(HashMap<String, String> params)   throws Throwable {  return iTestDao.getList(params);  } }  **[ITestDao.java]**  package com.spring.sample.web.test.dao; import java.util.HashMap; import java.util.List; public interface ITestDao {  public List<HashMap<String, String>> getList(HashMap<String, String> params)  throws Throwable;  }  **[TestDao.java]**  package com.spring.sample.web.test.dao; import java.util.HashMap; import java.util.List; import org.apache.ibatis.session.SqlSession; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Repository;  **@Repository // 저장소 선언**  public class TestDao implements ITestDao {  **@Autowired**  **public SqlSession sqlSession;**  @Override  public List<HashMap<String, String>> getList(HashMap<String, String> params) throws Throwable {  return sqlSession.selectList("test.getList", params);  }      }  **[Test\_SQL.xml]**  <?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="test">  <select id="getList" parameterType="hashmap" resultType="hashmap">  SELECT SELL\_ID, CUSTOMER, PRODUCT, PRICE, QUANTITY, SELL  FROM SELL  **<if test="searchTxt != null and searchTxt != ''">**  WHERE 1 = 1  **<choose>**  **<when test="searchGbn == 0">**  AND SELL\_ID LIKE '%' || #{searchTxt} || '%'  **</when>**  **<when test="searchGbn == 1">**  AND CUSTOMER LIKE '%' || #{searchTxt} || '%'  **</when>**  **<when test="searchGbn == 2">**  AND PRODUCT LIKE '%' || #{searchTxt} || '%'  **</when>**  **</choose>**  **</if>**  </select>  </mapper>  **[test1.html]**  <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Insert title here</title>  <script type="text/javascript" src="resources/script/jquery/jquery-1.12.4.min.js"></script>  <script type="text/javascript">  $(document).ready(function() {  $("#searchBtn").on("click", function() {  $("#actionForm").submit();  });  });  </script>  </head>  <body>  <!-- 검색의 경우 상태를 유지할 필요가 있음 -->  <form action="test1" id="actionForm" method="post">  <select name="searchGbn">  <c:choose>  <c:when test="${param.searchGbn eq 0}">  <option value="0" selected="selected">거래ID</option>  </c:when>  <c:otherwise>  <option value="0">거래ID</option>  </c:otherwise>  </c:choose>  <c:choose>  <c:when test="${param.searchGbn eq 1}">  <option value="1" selected="selected">고객</option>  </c:when>  <c:otherwise>  <option value="1">고객</option>  </c:otherwise>  </c:choose>  <c:choose>  <c:when test="${param.searchGbn eq 2}">  <option value="2" selected="selected">제품</option>  </c:when>  <c:otherwise>  <option value="2">제품</option>  </c:otherwise>  </c:choose>  </select>  <input type="text" name="searchTxt" value="${param.searchTxt}" />  <input type="button" value="검색" name="searchBtn" />  </form>  <table border="1">  <thead>  <tr>  <th>거래ID</th>  <th>고객</th>  <th>제품</th>  <th>단가</th>  <th>수량</th>  <th>판매</th>  </tr>  </thead>  <tbody>  <c:forEach var="data" items="${list}">  <tr>  <td>${data.SELL\_ID}</td>  <td>${data.CUSTOMER}</td>  <td>${data.PRODUCT}</td>  <td>${data.PRICE}</td>  <td>${data.QUANTITY}</td>  <td>${data.SELL}</td>  </tr>  </c:forEach>  </tbody>  </table>  </body>  </html> |

-DB 작업 플로우  
1) Controller에서 DB에 접근할 메소드에 throws Throwable 추가  
2) service 호출 생성  
3) service 호출 메소드 빨간줄에 커서이동 후 create method

4) 생성된 인터페이스 메소드 앞에 public 뒤에 throws Throwable 추가  
5) 인터페이스명에 ctrl 누른 상태로 커서 이동 후 Open Implimentation

6) service 클래스명 빨간줄에 커서 이동 후 add unimpli~~~~~  
7) dao 호출 생성  
8) dao 호출 메소드 빨간줄에 커서이동 후 create method  
9) 생성된 인터페이스 메소드 앞에 public 뒤에 throws Throwable 추가  
10) 인터페이스명에 ctrl 누른 상태로 커서 이동 후 Open Implimentation  
11) dao 클래스명 빨간줄에 커서이동 후 add unimplie~~~