**III. 스프링 연동**

프로젝트명 : step01

Spring에 MyBatis를 활용하기 위한 pom.xml문서에 dependency를 지정하자!

|  |
| --- |
| <dependencies>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.26.RELEASE</version>  </dependency>  <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.3.26.RELEASE</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.3.26.RELEASE</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.3.26.RELEASE</version>  </dependency>  <!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>1.4</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.5.4</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>2.0.4</version>  </dependency>  </dependencies> |

프로젝트 설정파일 web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>  <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/spring/root-context.xml</param-value>  </context-param>    <!-- Creates the Spring Container shared by all Servlets and Filters -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- Processes application requests -->  <servlet>  <servlet-name>appServlet</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>    <servlet-mapping>  <servlet-name>appServlet</servlet-name>  <url-pattern>**\*.do**</url-pattern>  </servlet-mapping>  **<!-- 한글처리 -->**  <filter>  <filter-name>charEncoding</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>charEncoding</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |

스프링 설정파일

|  |
| --- |
| root-context.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd"*>    <!-- Root Context: defines shared resources visible to all other web components -->  <!-- Root Context: defines shared resources visible to all other web components -->  <bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <property name=*"driverClassName"* value=*"oracle.jdbc.driver.OracleDriver"*></property>  <property name=*"url"* value=*"jdbc:oracle:thin:@localhost:1521:orcl"*></property>  <property name=*"username"* value=*"scott"*></property>  <property name=*"password"* value=*"tiger"*></property>  </bean>    <bean id=*"sqlSessionFactory"* class=*"org.mybatis.spring.SqlSessionFactoryBean"*>  <property name=*"dataSource"* ref=*"dataSource"* />  <property name=*"configLocation"* value=*"classpath:/mybatis-config.xml"*></property>  <property name=*"mapperLocations"* value=*"classpath:mappers/\*\*/\*Mapper.xml"*></property>  </bean>    <bean id=*"sqlSession"* class=*"org.mybatis.spring.SqlSessionTemplate"* destroy-method=*"clearCache"*>  <constructor-arg name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"*></constructor-arg>  </bean>    <!-- <context:component-scan base-package="org.javassem"></context:component-scan> -->      </beans> |
| servlet-context.xml |
| <!-- Enables the Spring MVC @Controller programming model -->  <annotation-driven />  <!-- Handles HTTP GET requests for /resources/\*\* by efficiently serving up static resources in the ${webappRoot}/resources directory -->  <resources mapping=*"/resources/\*\*"* location=*"/resources/"* />  <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->  <beans:bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <beans:property name=*"prefix"* value=*"/WEB-INF/views/"* />  <beans:property name=*"suffix"* value=*".jsp"* />  </beans:bean>    <context:component-scan base-package=*"mvc"* /> |

다음은 vo를 정의하자

|  |
| --- |
| mvc.vo.DeptVO.java |
| **package** vo;  **public class** DeptVO {  **private int** deptno;  **private** String dname,loc;  **//getter와 setter들을 만든다.**  } |

이제는 SQL문을 지정하는 mapper를 만들자!

|  |
| --- |
| mappers/deptMapper.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=*"dept"*>  <select id=*"list"* resultType=*"deptvo"*>  select \* from dept order by 1 desc  </select>  </mapper> |

그리고 config.xml 만들어서 Alias를 지정하자.

|  |
| --- |
| 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>  <typeAliases>  <typeAlias type=*"mvc.vo.DeptVO"* alias=*"deptvo"*/>  </typeAliases>  </configuration> |

다음은 위의 mapper를 호출하는 DAO를 작성하자!

|  |
| --- |
| mvc.dao.DeptDao.java |
| **package** mvc.dao;  **import** java.util.List;  **import** org.mybatis.spring.SqlSessionTemplate;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Repository;  **import** mvc.vo.DeptVO;  @Repository  **public class** DeptDao {  @Autowired  **private** SqlSessionTemplate ss;    **public** List<DeptVO> getDeptList(){  **return** ss.selectList("dept.list");  }  } |

|  |
| --- |
| DeptController.java |
| **package** mvc.controller;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** com.nexacro17.xapi.data.DataSet;  **import** com.nexacro17.xapi.data.DataTypes;  **import** mvc.dao.DeptDao;  **import** mvc.vo.DeptVO;  @Controller  **public class** DefaultController {  @Autowired  **private** DeptDao deptDao;    @RequestMapping(value="/selectDept.do")  **public** String defaultMain(Model model) {  List<DeptVO>list= deptDao.getDeptList();  // 나중에 넥사크로의 데이터셋으로 바인딩 될 이름  DataSet ds = **new** DataSet("ar");  // 데이터 셋에 들어갈 이름과 자료형이 동일해야 한다.  ds.addColumn("deptno", DataTypes.***INT***,10);  ds.addColumn("dname", DataTypes.***STRING***,20);  ds.addColumn("loc", DataTypes.***STRING***,20);  **for**(DeptVO e : list){  **int** row = ds.newRow();  ds.set(row, "deptno", e.getDeptno());  ds.set(row, "dname", e.getDname());  ds.set(row, "loc", e.getLoc());  }  model.addAttribute("ds", ds);  **return**"all";  }  } |

다음은 지정된 View Resolver에 all.jsp를 다음과 같이 작성해 보자!

|  |
| --- |
| /WEB-INF/views/all.jsp |
| <%@ page import=*"com.nexacro17.xapi.data.DataTypes"*%>  <%@ page import=*"com.nexacro17.xapi.tx.PlatformType"*%>  <%@ page import=*"com.nexacro17.xapi.tx.HttpPlatformResponse"*%>  <%@ page import=*"com.nexacro17.xapi.data.PlatformData"*%>  <%@ page import=*"com.nexacro17.xapi.data.DataSet"*%>  <%@ page import=*"java.util.ArrayList"*%>  <%@ page import=*"mvc.vo.DeptVO"*%>  <%@ page import=*"java.util.List"*%>  <%@ page contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <!DOCTYPEhtml>  <html>  <head>  <metacharset="UTF-8">  <title>Insert title here</title>  </head>  <body>  <%  DataSet ds1 = (DataSet)request.getAttribute("ds");  PlatformData pData = **new** PlatformData();  pData.addDataSet(ds1);  HttpPlatformResponse res =  **new** HttpPlatformResponse(response,PlatformType.CONTENT\_TYPE\_XML,"utf-8");  out.clear();  out = pageContext.pushBody();  res.setData(pData);  res.sendData();  %>  </body>  </html> |

이렇게 해서 Spring환경에 준비할 것은 끝이 났다.

[ 참고 ]

다른 프로젝트의 파일을 복사하다 보면 갑자기 index.jsp 도 404 에러가 발생하거나

콘솔에서 확인하면 에러 발생

java.lang.ClassNotFoundException: org.springframework.web.filter.CharacterEncodingFilter

Project > properties > Deployment Assembly > add button > java Build Path Entries  선택 >

Maven dependencies 선택

그러면 Maven Dependencis -> WEB-INF/lib가 새로 생성

다시 실행가면 된다