1、环境的搭建

笔记本:	ssm		
创建时间:	2021/7/25 22:06	**更新时间:**	2021/7/27 14:19
作者:	清风渡		
URL:	about:blank		

1、创建一个maven工程

2、引入项目依赖的jar包

- spring
- springmvc

```
<!--引入项目依赖的jar包 SpringMVC,Spring-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.1.14.RELEASE</version>
</dependency>


<!--Spring-Jdbc-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.1.14.RELEASE</version>
</dependency>


<!--Spring面向切面编程-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aspects</artifactId>
  <version>5.1.14.RELEASE</version>
</dependency>
```

- mybatis

```
<!--Mybatis-->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.1</version>
</dependency>
<!--Mybatis整合Spring 的适配包-->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.2</version>
</dependency>
```

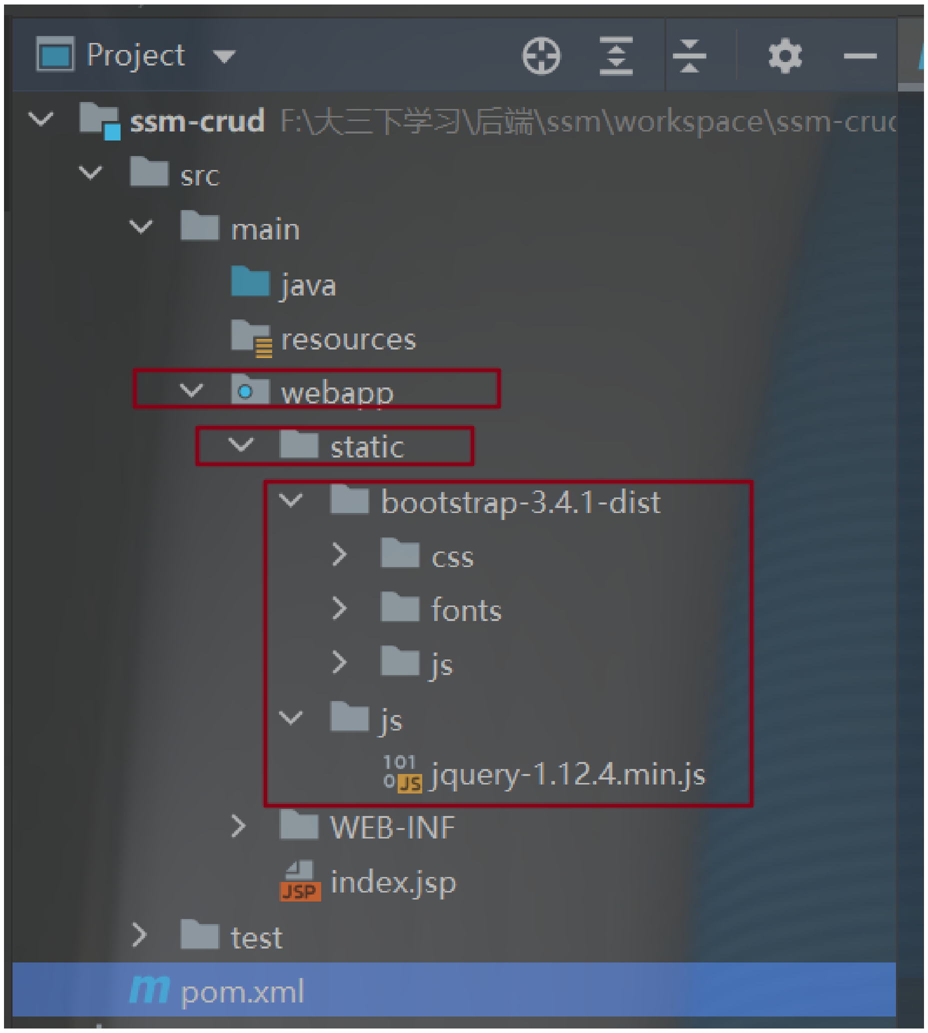
- 数据库连接池，驱动包

```
<!--数据库驱动 以及  数据库连接池-->
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>1.1.18</version>
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.13</version>
</dependency>
```

- 其他（jstl，servlet-api，junit）

```
<!--Jstl servlet-api,junit-->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

3、引入bootstrap前端框架



测试前端框架的引入

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>首页</title>
    <%--引入jQuery--%>
    <script type="text/javascript" src="static/js/jquery-1.12.4.min.js">
</script>
    <%--引入样式--%>
    <link rel="stylesheet" href="static/bootstrap-3.4.1-
dist/css/bootstrap.min.css">
    <script src="static/bootstrap-3.4.1-dist/js/bootstrap.min.js">
</script>
</head>
<body>
    <button class="btn bg-success">按钮</button>
</body>
</html>
```

4、编写ssm整合的关键配置文件

- web.xml
  - 启动Spring容器，记得在类路径下resources目录下新建spring配置文件
  - SpringMVC前端控制器，记得在web-inf目录下创建SpringMVC配置文件，文件名为servlet名字+ “-servlet”
  - 字符集编码过滤器

- 支持rest风格的URL

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" 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
   http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <display-name>Archetype Created Web Application</display-name>


  <!--1、启动Spring的容器-->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>

  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>


  <!--2、SpringMVC前端控制器,拦截所有请求-->
  <servlet>
    <!--如果这里没有配置多的信息，要在web-inf下面配置SpringMVC的配置文件，名字
必须为《下面的servlet-name+"-servlet"》-->
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
  <!--3、解决字符集编码问题，字符编码过滤器(这个过滤器一定要在其他过滤器之前)-->
  <filter>
    <filter-name>CharacterEncodingFilter</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>
    <init-param>
      <param-name>forceRequestEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>forceResponseEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>


  <filter-mapping>
```

```
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <!--rest风格URL，将页面普通的post请求转为指定的delete或者put请求-->
  <filter>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <filter-
class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>


</web-app>
```

- spring配置文件（applicationContext.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"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/aop
https://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">


        <!--扫描组件，除了控制层-->
        <context:component-scan base-package="com.study">
                <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
        </context:component-scan>


        <!--Spring配置文件，这里主要配置和业务逻辑相关的-->
        <!--数据源、事务控制-->


        <!--引入外部配置文件-->
        <context:property-placeholder
location="classpath:dbConfig.properties"/>
        <bean id="dataSource"
class="com.alibaba.druid.pool.DruidDataSource">
                <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>
        <!--配置和mybatis的整合-->
        <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
                <!--指定mybatis全局配置文件的路径-->
                <property name="configLocation" value="classpath:mybatis-
config.xml"/>
                <property name="dataSource" ref="dataSource"/>
                <!--指定mybatis mapper文件的位置-->
                <property name="mapperLocations"
value="classpath:mapper/*.xml"/>
        </bean>


        <!--配置扫描器，将mybatis接口的实现加入到IOC容器中-->
        <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
                <!--扫描所有的dao接口的实现。加入到IOC容器中-->
                <property name="basePackage" value="com.study.dao"/>
        </bean>


        <!--事务控制的配置-->
        <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
                <!--控制住数据源-->
                <property name="dataSource" ref="dataSource"/>
        </bean>


        <!--开启基于注解的事务，使用xml配置形式的事务(必要主要都是使用配置式)--
>
        <aop:config>
                <!--切入点表达式：哪些方法要切入事务-->
                <aop:pointcut id="txPoint" expression="execution(*
com.study.service..*.*(..))"/>
                <!--配置事务增强-->
                <aop:advisor advice-ref="txAdvice" pointcut-
ref="txPoint"/>
        </aop:config>


        <!--配置事务增强。事务如何切入-->
        <tx:advice id="txAdvice" transaction-manager="transactionManager">
                <tx:attributes>
                        <!--所有方法都是事务方法-->
                        <tx:method name="*"/>
                        <!--get开头的所有方法 -->
                        <tx:method name="get*" read-only="true"/>
                </tx:attributes>
        </tx:advice>
</beans>
```

- 要在resource目录下新建数据库配置文件

```
jdbc.driverClassName=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/ssm_crud?
serverTimezone=Asia/Shanghai&&rewriteBatchedStatements=true
jdbc.username=root
jdbc.password=123456
```

- springmvc（dispatcherServlet-servlet.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"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">


    <!--SpringMVC配置文件，包含网站跳转逻辑，配置-->


    <!--默认扫描所有的业务逻辑组件，记得将默认扫描关掉-->
    <context:component-scan base-package="com.study" use-default-
filters="false">
        <!--只扫描控制器-->
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>


    <!--配置视图解析器，方便页面返回-->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/"/>
        <property name="suffix" value=".jsp"/>
    </bean>


    <!--两个标准配置-->
    <!--将SpringMVC不能处理的请求交给Tomcat-->
    <mvc:default-servlet-handler/>
    <!--能支持SpringMVC更高级的一些功能，JSR303的校验，快捷的ajax。。映射动态
请求-->
    <mvc:annotation-driven/>
</beans>
```

- mybatis
  - XML 配置文件 https://mybatis.org/mybatis-3/zh/getting-started.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
        PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration
1.0//EN"
        "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">


<generatorConfiguration>
    <!--删除下面这一行-->
```

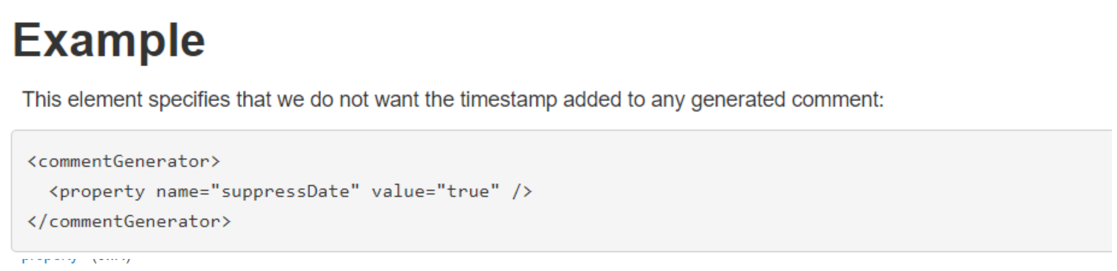
```
<!--    <classPathEntry location="/Program
Files/IBM/SQLLIB/java/db2java.zip" />-->


    <context id="DB2Tables" targetRuntime="MyBatis3">


        <!--将注释删除-->
        <commentGenerator>
            <property name="suppressAllComments" value="true" />
        </commentGenerator>


        <!--配置数据库连接-->
        <jdbcConnection driverClass="com.mysql.cj.jdbc.Driver"
                        connectionURL="jdbc:mysql://localhost:3306/ssm_crud?
serverTimezone=Asia/Shanghai"
                        userId="root"
                        password="123456">
        </jdbcConnection>


        <javaTypeResolver >
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>


        <!--指定Javabean生成的位置-->
        <javaModelGenerator targetPackage="com.study.bean"
targetProject="src/main/java">
            <property name="enableSubPackages" value="true" />
            <property name="trimStrings" value="true" />
        </javaModelGenerator>


        <!--指定sql映射文件生成的位置-->
        <sqlMapGenerator
targetPackage="mapper"  targetProject="src/main/resources">
            <property name="enableSubPackages" value="true" />
        </sqlMapGenerator>


        <!--指定dao接口生成的位置，mapper接口-->
        <javaClientGenerator type="XMLMAPPER"
targetPackage="com.study.dao"  targetProject="src/main/java">
            <property name="enableSubPackages" value="true" />
        </javaClientGenerator>


        <!--table 指定每个表的生产策略-->
        <table tableName="tb_emp" domainObjectName="Employee"></table>
        <table tableName="tb_dept" domainObjectName="Department"></table>
    </context>
</generatorConfiguration>
```

- 创建数据表
- 使用mybatis的逆向工程生成对应的bean以 及mapper

- 导入mybatis generator依赖 https://mybatis.org/generator/quickstart.html
- 工程目录下新建一个mbg.xml配置文件 文件可参考示例：https://mybatis.org/generator/configreference/xmlconfig.html

```
```

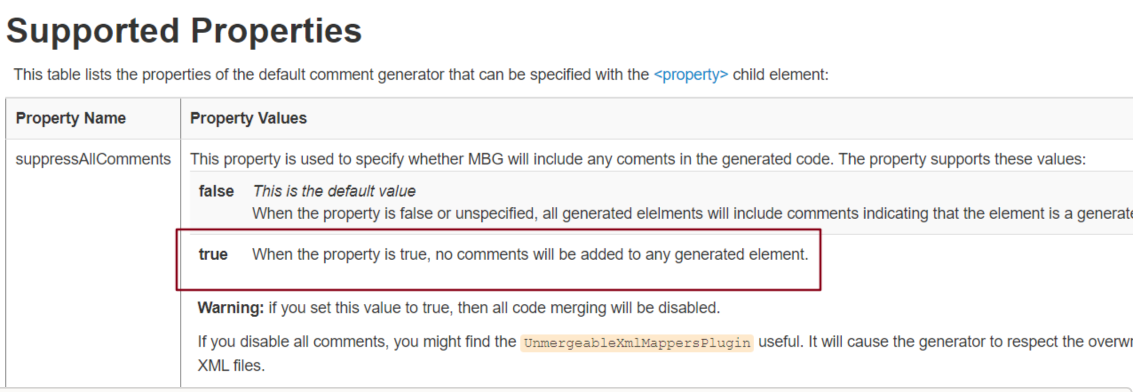
- 在测试类生成 https://mybatis.org/generator/running/runningWithJava.html
- 生成注释太多，可以在配置文件中添加配置 https://mybatis.org/generator/configreference/commentGenerator.html



Example

This element specifies that we do not want the timestamp added to any generated comment:

```
<commentGenerator>
  <property name="suppressDate" value="true" />
</commentGenerator>
```



Supported Properties

This table lists the properties of the default comment generator that can be specified with the <property> child element:

Property Name	Property Values
suppressAllComments	This property is used to specify whether MBG will include any coments in the generated code. The property supports these values: **false** *This is the default value* When the property is false or unspecified, all generated elelments will include comments indicating that the element is a generat **true** When the property is true, no comments will be added to any generated element. **Warning:** if you set this value to true, then all code merging will be disabled. If you disable all comments, you might find the `UnmergeableXmlMappersPlugin` useful. It will cause the generator to respect the overwr XML files.

```
   <!--将注释删除-->
         <commentGenerator>
             <property name="suppressAllComments" value="true" />
         </commentGenerator>
```

逆向工程以后自己新增的方法：

```
<!--要查询部门字段的sql-->
<sql id="withDept_Column_List">
    e.emp_id, e.emp_name, e.gender, e.email, e.d_id,d.dept_id,d.dept_name
</sql>
<!--
  /*查询员工对象，并且将对应的部门对象查出来*/
  List<Employee> selectByExampleWithDept(EmployeeExample example);


  /*根据主键查询员工对象，同时将部门对象查询出来*/
  Employee selectByPrimaryKeyWithDept(Integer empId);
-->
<!--查询员工带部门信息-->
<select id="selectByExampleWithDept" resultMap="withDeptResultMap">
  select
  <if test="distinct">
    distinct
  </if>
  <include refid="withDept_Column_List" />
  from tb_emp e
  LEFT JOIN tb_dept d
  on e.d_id = d.dept_id
  <if test="_parameter != null">
    <include refid="Example_Where_Clause" />
  </if>
  <if test="orderByClause != null">
    order by ${orderByClause}
  </if>
</select>
<select id="selectByPrimaryKeyWithDept" resultMap="withDeptResultMap">
  select
  <include refid="withDept_Column_List"/>
  from tb_emp e
  LEFT JOIN tb_dept d
  on e.d_id = d.dept_id
  where emp_id = #{empId,jdbcType=INTEGER}
</select>
```

5、测试mapper

在测试类中，要实现批量插入，所以要在spring配置文件中加入配置

```
applicationContext.xml
<!--配置一个可以实现批量插入的sqlSession-->
<bean class="org.mybatis.spring.SqlSessionTemplate" id="sqlSession">
        <constructor-arg name="sqlSessionFactory"
ref="sqlSessionFactory"/>
        <!--批量插入-->
        <constructor-arg name="executorType" value="BATCH"/>
</bean>


// 测试类MapperTest
import com.study.bean.Employee;
import com.study.dao.DepartmentMapper;
import com.study.dao.EmployeeMapper;
import org.apache.ibatis.session.SqlSession;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;


import java.util.UUID;


@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath:applicationContext.xml"})
public class MapperTest {


    @Autowired
    DepartmentMapper departmentMapper;


    @Autowired
    EmployeeMapper employeeMapper;

    /*注入批量插入的sqlSession*/
    @Autowired
    SqlSession sqlSession;


    @Test
    public void testCRUD(){
//        System.out.println("departmentMapper = " + departmentMapper);


        /*插入几个部门*/
//        Department department1 = new Department(null, "开发部");
//        Department department2 = new Department(null, "测试部");
//        departmentMapper.insertSelective(department1);
//        departmentMapper.insertSelective(department2);


        /*测试插入员工*/
//        Employee employee = new Employee(null, "胡涛", "男",
"2223@qq.com", 1);
//        employeeMapper.insertSelective(employee);


        /*批量插入员工,如果不导入这个批量插入的sqlSession,使用上面的插入，会要
很长时*/
        EmployeeMapper mapper =
sqlSession.getMapper(EmployeeMapper.class);
        for (int i = 0; i < 1000; i++) {
            String uid = UUID.randomUUID().toString().substring(0, 5) + i;
            mapper.insertSelective(new Employee(null, uid, "男",
uid+"@qq.com", 1));
        }
        System.out.println("插入完成");
    }


}
```