# SSM 框架整合教程：四、SSM 框架整合——尚硅谷学习笔记 2022 年



# 四、SSM 框架整合

## 4.1、ContextLoaderListener

Spring 提供了监听器 ContextLoaderListener，实现 ServletContextListener 接口，可监听 ServletContext 的状态，在 web 服务器的启动，读取 Spring 的配置文件，创建 Spring 的 IOC 容器。web 应用中必须在 web.xml 中配置。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakarta
         version="6.0">
    <!-- 配置 Spring 的监听器 -->
    <listener>
        <!-- 配置 Spring 的监听器，在服务器启动时加载 Spring 的配置文件
        Spring 配置文件默认位置和名称：/WEB-INF/applicationContext.xml
        可通过上下文参数自定义 Spring 配置文件的位置和名称
         -->
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- 自定义 Spring 配置文件的位置和名称 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring.xml</param-value>
    </context-param>

    <!-- 配置 Spring 的编码过滤器 -->
    <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>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>characterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 设置 SpringMVC 的前端控制器 -->
    <servlet>
        <servlet-name>springMVC</servlet-name>
```

```
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:springmvc.xml</param-value>
        </init-param>

        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>springMVC</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

## 4.2、准备工作

### 4.2.1 ① 创建 Maven Module

### 4.2.2 ② 导入依赖

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.myxh.ssm</groupId>
    <artifactId>spring_springmvc_mybatis</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <spring.version>6.0.9</spring.version>
    </properties>

    <dependencies>
        <!-- 基于 Maven 依赖传递性，导入 spring-context 依赖即可导入当前所需所有 jar 包 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- spring-aspects 会帮我们传递过来 aspectjweaver -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring 持久化层支持 jar 包 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
```

```
        <version>6.0.6</version>
    </dependency>

    <!-- Spring 测试相关 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>6.0.6</version>
    </dependency>

    <!-- SpringMVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!-- SpringMVC -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!-- ServletAPI -->
    <dependency>
        <groupId>jakarta.servlet</groupId>
        <artifactId>jakarta.servlet-api</artifactId>
        <version>6.0.0</version>
        <scope>provided</scope>
    </dependency>

    <!-- Spring6 和 Thymeleaf 整合包 -->
    <dependency>
        <groupId>org.thymeleaf</groupId>
        <artifactId>thymeleaf-spring6</artifactId>
        <version>3.1.1.RELEASE</version>
    </dependency>

    <!-- jackson -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.15.1</version>
```

```
        </dependency>

        <!-- Mybatis核心 -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.13</version>
        </dependency>

        <!-- Mybatis 和 Spring的整合包-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>3.0.2</version>
        </dependency>

        <!-- MySQL 驱动 -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>

        <!-- 数据源连接池 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.2.16</version>
        </dependency>

        <!-- 分页插件 -->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.3.2</version>
        </dependency>

        <!-- junit 测试 -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
```

```
        <!-- log4j 日志 -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>

        <!-- slf4j 日志门面的一个具体实现 -->
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.4.7</version>
        </dependency>
    </dependencies>
</project>
```

### 4.2.3 ③ 创建表

```sql
# 创建数据库 ssm
CREATE DATABASE IF NOT EXISTS ssm;

# 选择数据库 ssm
USE ssm;

# 创建员工表 t_employee
CREATE TABLE IF NOT EXISTS t_employee
(
    employee_id   INT AUTO_INCREMENT,
    employee_name VARCHAR(20) NULL,
    age           INT         NULL,
    gender        CHAR        NULL,
    email         VARCHAR(50) NULL,
    department_id INT         NULL,
    CONSTRAINT t_employee_pk
        PRIMARY KEY (employee_id)
);

# 添加员工表 t_employee 的数据
INSERT INTO t_employee (employee_id, employee_name, age, gender, email, department_id)
VALUES (1, 'MYXH', 21, '男', '1735350920@qq.com', 1),
       (2, '张三', 20, '男', 'zhangsan@qq.com', 1),
       (3, '李四', 22, '男', 'lisi@qq.com', 2),
       (4, '王五', 23, '男', 'wangwu@qq.com', 2),
       (5, '赵六', 24, '男', 'zhaoliu@qq.com', 3),
       (6, 'test1', 18, '男', 'test1@qq.com', null),
       (7, 'test2', 18, '男', 'test2@qq.com', null),
       (8, 'test3', 18, '男', 'test3@qq.com', null),
       (9, 'test4', 18, '男', 'test4@qq.com', null),
       (10, 'test5', 18, '男', 'test5@qq.com', null),
       (11, 'test6', 18, '男', 'test6@qq.com', null),
       (12, 'test7', 18, '男', 'test7@qq.com', null),
       (13, 'test8', 18, '男', 'test8@qq.com', null),
       (14, 'test9', 18, '男', 'test9@qq.com', null),
       (15, 'test10', 18, '男', 'test10@qq.com', null),
       (16, 'test11', 18, '男', 'test11@qq.com', null),
       (17, 'test12', 18, '男', 'test12@qq.com', null),
       (18, 'test13', 18, '男', 'test13@qq.com', null),
       (19, 'test14', 18, '男', 'test14@qq.com', null),
       (20, 'test15', 18, '男', 'test15@qq.com', null),
       (21, 'test16', 18, '男', 'test16@qq.com', null),
       (22, 'test17', 18, '男', 'test17@qq.com', null),
       (23, 'test18', 18, '男', 'test18@qq.com', null),
```

```
    (24, 'test19', 18, '男', 'test19@qq.com', null),
    (25, 'test20', 18, '男', 'test20@qq.com', null),
    (26, 'test21', 18, '男', 'test21@qq.com', null),
    (27, 'test22', 18, '男', 'test22@qq.com', null),
    (28, 'test23', 18, '男', 'test23@qq.com', null),
    (29, 'test24', 18, '男', 'test24@qq.com', null),
    (30, 'test25', 18, '男', 'test25@qq.com', null);
```

## 4.3、配置 web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakarta
         version="6.0">
    <!-- 配置 Spring 的监听器 -->
    <listener>
        <!-- 配置 Spring 的监听器，在服务器启动时加载 Spring 的配置文件
        Spring 配置文件默认位置和名称：/WEB-INF/applicationContext.xml
        可通过上下文参数自定义 Spring 配置文件的位置和名称
         -->
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- 设置 Spring 配置文件自定义的位置和名称 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring.xml</param-value>
    </context-param>

    <!-- 配置 Spring 的编码过滤器 -->
    <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>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>characterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!-- 配置处理请求方式的过滤器 -->
    <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>

    <!-- 配置 SpringMVC 的前端控制器 DispatcherServlet -->
    <servlet>
        <servlet-name>springMVC</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

        <!-- 设置 SpringMVC 配置文件自定义的位置和名称 -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:springmvc.xml</param-value>
        </init-param>

        <!-- 将 DispatcherServLet 的初始化时间提前到服务器启动时 -->
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>springMVC</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

## 4.4、创建 SpringMVC 的配置文件并配置

```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:mvc="http://www.springframework
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.
    <!-- 扫描控制层组件 -->
    <context:component-scan base-package="com.myxh.ssm.controller"/>

    <!-- 配置 Thymeleaf 视图解析器 -->
    <bean id="viewResolver" class="org.thymeleaf.spring6.view.ThymeleafViewResolver">
        <property name="order" value="1"/>
        <property name="characterEncoding" value="UTF-8"/>
        <property name="templateEngine">
            <bean class="org.thymeleaf.spring6.SpringTemplateEngine">
                <property name="templateResolver">
                    <bean class="org.thymeleaf.spring6.templateresolver.SpringResourceTemplateReso
                        <!-- 视图前缀 -->
                        <property name="prefix" value="/WEB-INF/templates/"/>

                        <!-- 视图后缀 -->
                        <property name="suffix" value=".html"/>

                        <property name="templateMode" value="HTML"/>
                        <property name="characterEncoding" value="UTF-8"/>
                    </bean>
                </property>
            </bean>
        </property>
    </bean>

    <!-- 配置默认的 servlet 处理静态资源 -->
    <mvc:default-servlet-handler/>

    <!-- 开启 mvc 的注解驱动 -->
    <mvc:annotation-driven/>

    <!-- 配置视图控制器 -->
    <mvc:view-controller path="/" view-name="index"/>

    <!-- 配置文件上传解析器 -->
    <!-- 配置 StandardServletMultipartResolver 为 MultipartResolver -->
    <!-- 该类实现了 Servlet3.0 规范的文件上传支持 -->
    <bean id="multipartResolver" class="org.springframework.web.multipart.support.StandardServletM
</beans>
```

# 4.5、搭建 MyBatis 环境

## 4.5.1 ① 创建属性文件 jdbc.properties

```
jdbc.driver=com.mysql.cj.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/ssm?serverTimezone=UTC
jdbc.username=MYXH
jdbc.password=520.ILY!
```

## 4.5.2 ② 创建 MyBatis 的核心配置文件 mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "https://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!--
        MyBatis 核心配置文件中的标签必须要按照指定的顺序配置:
        properties?,settings?,typeAliases?,typeHandlers?,
        objectFactory?,objectWrapperFactory?,reflectorFactory?,plugins?,
        environments?,databaseIdProvider?,mappers?
     -->
    <settings>
        <!-- 将下划线命名风格映射为驼峰命名风格 -->
        <setting name="mapUnderscoreToCamelCase" value="true"/>

        <!-- 开启延迟加载 -->
        <setting name="lazyLoadingEnabled" value="true"/>

        <!-- 按需加载 -->
        <setting name="aggressiveLazyLoading" value="false"/>
    </settings>

    <plugins>
        <!-- 配置分页插件 -->
        <plugin interceptor="com.github.pagehelper.PageInterceptor"/>
    </plugins>
</configuration>
```

### 4.5.3 ③ 创建 Mapper 接口和映射文件

```java
package com.myxh.ssm.mapper;

import com.myxh.ssm.pojo.Employee;

import java.util.List;

/**
 * @author MYXH
 * @date 2023/9/9
 */
public interface EmployeeMapper
{
    /**
     * 查询所有的员工信息
     *
     * @return 所有员工信息
     */
    List<Employee> getAllEmployee();
}
```

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.myxh.ssm.mapper.EmployeeMapper">
    <!-- List<Employee> getAllEmployee(); -->
    <select id="getAllEmployee" resultType="Employee">
        select * from t_employee
    </select>
</mapper>
```

## 4.5.4 ④ 创建日志文件 log4j.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
    <appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
        <param name="Encoding" value="UTF-8"/>
        <layout class="org.apache.log4j.PatternLayout">
            <param name="ConversionPattern" value="%-5p %d{MM-dd HH:mm:ss,SSS} %m (%F:%L) \n"/>
        </layout>
    </appender>

    <logger name="java.sql">
        <level value="debug"/>
    </logger>

    <logger name="org.apache.ibatis">
        <level value="info"/>
    </logger>

    <root>
        <level value="debug"/>
        <appender-ref ref="STDOUT"/>
    </root>
</log4j:configuration>
```

## 4.6、创建 Spring 的配置文件并配置

```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:tx="http://www.springfr
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.
    <!-- 扫描组件（排除控制层） -->
    <context:component-scan base-package="com.myxh.ssm">
        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Contr
    </context:component-scan>

    <!-- 引入外部属性文件 jdbc.properties -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 配置数据源 -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <!-- 配置事务管理器 -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransaction
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!--
        开启事务的注解驱动
        将使用注解 @Transactional 标识的方法或类中所有的方法进行事务管理
     -->
    <tx:annotation-driven transaction-manager="transactionManager"/>

    <!-- 配置 sqlSessionFactoryBean，可以直接在 Spring 的 IOC 中获取 sqlSessionFactory -->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 设置 MyBatis 的核心配置文件的路径 -->
        <property name="configLocation" value="classpath:mybatis-config.xml"/>

        <!-- 设置数据源 -->
        <property name="dataSource" ref="dataSource"/>

        <!-- 设置类型别名所对应的包 -->
        <property name="typeAliasesPackage" value="com.myxh.ssm.pojo"/>

        <!-- 设置 MyBatis 映射文件的路径，只有映射文件的包和 mapper 接口的包不一致时需要设置-->
```

```
        <!-- <property name="mapperLocations" value="classpath:com/myxh/ssm/mapper/*.xml"/> -->
    </bean>

    <!--
      配置 mapper 接口的扫描，可以将指定包下所有的 mapper 接口
      通过 SqlSession 创建代理实现类对象，并将这些对象交给 IOC 容器管理
     -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.myxh.ssm.mapper"/>
    </bean>
</beans>
```

# 4.7、测试功能

## 4.7.1 ① 创建组件

实体类 Employee。

```java
package com.myxh.ssm.pojo;

/**
 * @author MYXH
 * @date 2023/9/9
 */
public class Employee
{
    private Integer employeeId;
    private String employeeName;
    private Integer age;
    private String gender;
    private String email;

    public Employee()
    {

    }

    public Employee(Integer employeeId, String employeeName, Integer age, String gender, String em
    {
        this.employeeId = employeeId;
        this.employeeName = employeeName;
        this.age = age;
        this.gender = gender;
        this.email = email;
    }

    public Integer getEmployeeId()
    {
        return employeeId;
    }

    public void setEmployeeId(Integer employeeId)
    {
        this.employeeId = employeeId;
    }

    public String getEmployeeName()
    {
        return employeeName;
    }

    public void setEmployeeName(String employeeName)
```

```
    {
        this.employeeName = employeeName;
    }

    public Integer getAge()
    {
        return age;
    }

    public void setAge(Integer age)
    {
        this.age = age;
    }

    public String getGender()
    {
        return gender;
    }

    public void setGender(String gender)
    {
        this.gender = gender;
    }

    public String getEmail()
    {
        return email;
    }

    public void setEmail(String email)
    {
        this.email = email;
    }

    @Override
    public String toString()
    {
        return "Employee{" +
                "employeeId=" + employeeId +
                ", employeeName='" + employeeName + '\'' +
                ", age=" + age +
                ", gender='" + gender + '\'' +
                ", email='" + email + '\'' +
                '}';
```

```
    }
}
```

创建控制层组件 EmployeeController。

```java
package com.myxh.ssm.controller;

import com.github.pagehelper.PageInfo;
import com.myxh.ssm.pojo.Employee;
import com.myxh.ssm.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import java.util.List;

/**
 * @author MYXH
 * @date 2023/9/9
 * @description
 * 查询所有的员工信息：/employee -> GET
 * 查询员工的分页信息：/employee/page/1 -> GET
 * 根据 id 查询员工信息：/employee/1 -> GET
 * 跳转到添加页面：/to/add -> GET
 * 添加员工信息：/employee -> POST
 * 跳转到修改页面：/employee/1 -> GET
 * 修改员工信息：/employee -> PUT
 * 根据 id 删除员工信息：/employee/1 -> DELETE
 */
@Controller
public class EmployeeController
{
    @Autowired
    private EmployeeService employeeService;

    @RequestMapping(value = "/employee", method = RequestMethod.GET)
    public String getAllEmployee(Model model)
    {
        // 查询所有的员工信息
        List<Employee> employeeList = employeeService.getAllEmployee();

        // 将所有的员工信息在请求域中共享
        model.addAttribute("employeeList", employeeList);

        // 跳转到列表页面
        return "employee_list";
```

```
    }

    @RequestMapping(value = "/employee/page/{pageNum}", method = RequestMethod.GET)
    public String getEmployeePage(@PathVariable("pageNum") Integer pageNum, Model model)
    {
        // 获取员工的分页信息
        PageInfo<Employee> employeePage = employeeService.getEmployeePage(pageNum);

        // 将分页数据共享到请求域中
        model.addAttribute("employeePage", employeePage);

        // 跳转到列表页面
        return "employee_list";
    }
}
```

创建接口 EmployeeService。

```java
package com.myxh.ssm.service;

import com.github.pagehelper.PageInfo;
import com.myxh.ssm.pojo.Employee;

import java.util.List;

/**
 * @author MYXH
 * @date 2023/9/9
 */
public interface EmployeeService
{



    /**
     * 查询所有的员工信息
     *
     * @return 所有员工信息
     */
    List<Employee> getAllEmployee();

    /**
     * 获取员工的分页信息
     * @param pageNum 当前页的页码
     * @return 员工的分页信息
     */
    PageInfo<Employee> getEmployeePage(Integer pageNum);
}
```

创建实现类 EmployeeServiceImpl。

```java
package com.myxh.ssm.service.impl;

import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import com.myxh.ssm.mapper.EmployeeMapper;
import com.myxh.ssm.pojo.Employee;
import com.myxh.ssm.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

/**
 * @author MYXH
 * @date 2023/9/9
 */
@Service
@Transactional
public class EmployeeServiceImpl implements EmployeeService
{
    @Autowired
    private EmployeeMapper employeeMapper;

    /**
     * 查询所有的员工信息
     *
     * @return 所有员工信息
     */
    @Override
    public List<Employee> getAllEmployee()
    {
        return employeeMapper.getAllEmployee();
    }

    /**
     * 获取员工的分页信息
     * @param pageNum 当前页的页码
     * @return 员工的分页信息
     */
    @Override
    public PageInfo<Employee> getEmployeePage(Integer pageNum)
    {
        // 开启分页功能
```

```
        PageHelper.startPage(pageNum, 5);

        // 查询所有的员工信息
        List<Employee> employeeList = employeeMapper.getAllEmployee();

        // 获取分页相关数据
        PageInfo<Employee> employeePage = new PageInfo<>(employeeList, 5);

        return employeePage;
    }
}
```

## 4.7.2 ② 创建页面

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <title>首页</title>
  </head>

  <body>
    <h1>index.html</h1>

    <h2>1、SSM 框架整合</h2>
    <h3>1.1、查询所有的员工信息</h3>
    <a th:href="@{/employee}">查询所有的员工信息</a>

    <h3>1.2、查询员工的分页信息</h3>
    <a th:href="@{/employee/page/1}">查询员工的分页信息</a>
  </body>
</html>
```

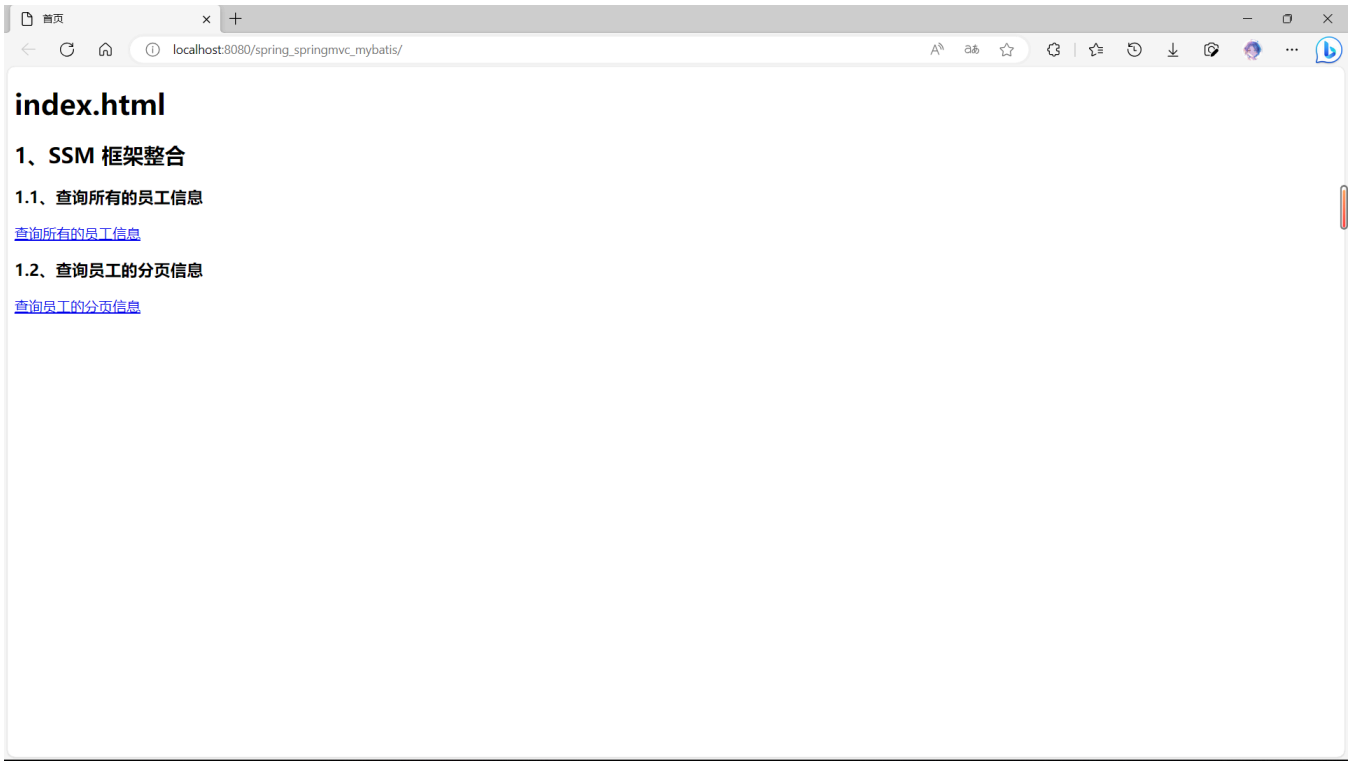
```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <title>员工列表</title>
    <link rel="stylesheet" th:href="@{/static/css/index_work.css}" />
  </head>

  <body>
    <div id="app">
      <table>
        <tr>
          <th colspan="7">员工列表</th>
        </tr>

        <tr>
          <th>序号</th>
          <th>员工 id</th>
          <th>员工姓名</th>
          <th>年龄</th>
          <th>性别</th>
          <th>电子邮件</th>
          <th>选项 (<a th:href="@{/to/add}">添加</a>) </th>
        </tr>

        <!-- <tr th:each="employee, status : ${employeeList}"> -->
        <tr th:each="employee, status : ${employeePage.list}">
          <td th:text="${status.count}"></td>
          <td th:text="${employee.employeeId}"></td>
          <td th:text="${employee.employeeName}"></td>
          <td th:text="${employee.age}"></td>
          <td th:text="${employee.gender}"></td>
          <td th:text="${employee.email}"></td>

          <td>
            <a th:href="@{|/employee/${employee.employeeId}|}">修改</a>
            <a
              @click="deleteEmployee"
              th:href="@{|/employee/${employee.employeeId}|}"
              >删除</a
            >
          </td>
        </tr>
      </table>
```

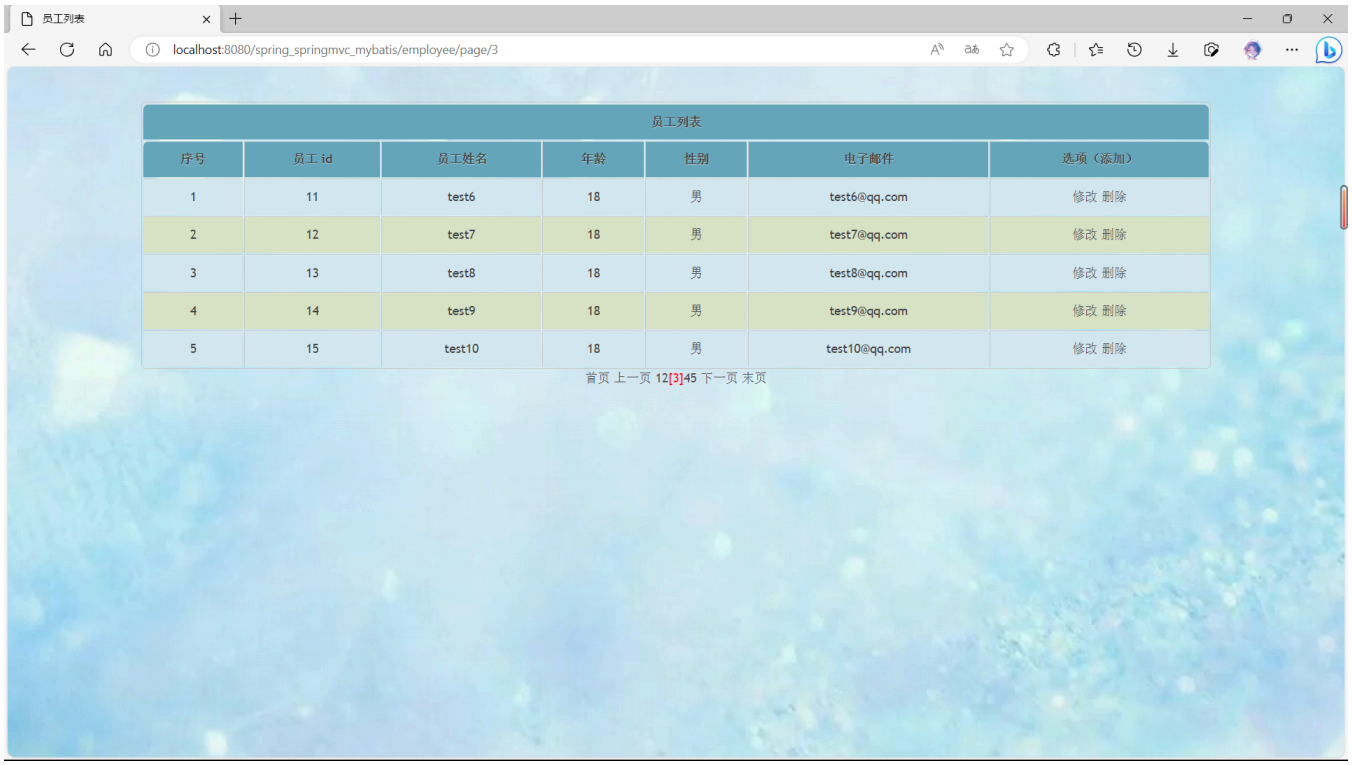
```
  <div style="text-align: center">
    <a th:if="${employeePage.hasPreviousPage}" th:href="@{/employee/page/1}"
      >首页</a
    >
    <a
      th:if="${employeePage.hasPreviousPage}"
      th:href="@{|/employee/page/${employeePage.prePage}|}"
      >上一页</a
    >
    <span th:each="navigatepageNum : ${employeePage.navigatepageNums}">
      <a
        th:if="${employeePage.pageNum == navigatepageNum}"
        style="color: red"
        th:href="@{|/employee/page/${navigatepageNum}|}"
        th:text="|[${navigatepageNum}]|"
      ></a>
      <a
        th:if="${employeePage.pageNum != navigatepageNum}"
        th:href="@{|/employee/page/${navigatepageNum}|}"
        th:text="${navigatepageNum}"
      ></a>
    </span>
    <a
      th:if="${employeePage.hasNextPage}"
      th:href="@{|/employee/page/${employeePage.nextPage}|}"
      >下一页</a
    >
    <a
      th:if="${employeePage.hasNextPage}"
      th:href="@{|/employee/page/${employeePage.pages}|}"
      >末页</a
    >
  </div>

  <form method="post">
    <input type="hidden" name="_method" value="delete" />
  </form>
</div>

<script type="text/javascript" th:src="@{/static/js/vue.js}"></script>

<script type="text/javascript">
  let vue = new Vue({
```

```
        el: "#app",

        methods: {
          deleteEmployee() {
            // 获取 form 表单
            let form = document.getElementsByTagName("form")[0];

            // 将超链接的 href 属性值赋值给 form 表单的 action 属性
            // event.target 表示当前触发事件的标签
            form.action = event.target.href;

            // 表单提交
            form.submit();

            // 阻止超链接的默认行为
            event.preventDefault();
          },
        },
      });
    </script>
  </body>
</html>
```

### 4.7.3 ③ 访问测试分页功能

http://localhost:8080/spring_springmvc_mybatis/employee/page/1


首页
localhost:8080/spring_springmvc_mybatis/
index.html
1、SSM 框架整合
1.1、查询所有的员工信息
查询所有的员工信息
1.2、查询员工的分页信息
查询员工的分页信息


员工列表
localhost:8080/spring_springmvc_mybatis/employee/page/3
员工列表
序号 员工 id 员工姓名 年龄 性别 电子邮件 选项（添加）
1 11 test6 18 男 test6@qq.com 修改 删除
2 12 test7 18 男 test7@qq.com 修改 删除
3 13 test8 18 男 test8@qq.com 修改 删除
4 14 test9 18 男 test9@qq.com 修改 删除
5 15 test10 18 男 test10@qq.com 修改 删除
首页 上一页 12[3]45 下一页 末页