SSM补充及整合

SSM整合

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SpringIOC容器，web应用中必须在web.xml中配置

```
<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
    -->

 <listenerclass>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>
```

2、准备工作

①创建Maven Module

②引入依赖

```
<packaging>war</packaging>
<properties>
    <spring.version>5.3.1</spring.version>
</properties>

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

    <!--springmvc-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <!-- Mybatis核心 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.7</version>
```

```
    </dependency>

    <!--mybatis和spring的整合包-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.6</version>
    </dependency>

    <!-- 连接池 -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
        <version>1.0.9</version>
    </dependency>

    <!-- junit测试 -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>

    <!-- MySQL驱动 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.16</version>
    </dependency>

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

    <!-- https://mvnrepository.com/artifact/com.github.pagehelper/pagehelper -->
    <dependency>
        <groupId>com.github.pagehelper</groupId>
        <artifactId>pagehelper</artifactId>
        <version>5.2.0</version>
    </dependency>

    <!-- 日志 -->
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.2.3</version>
    </dependency>

    <!-- ServletAPI -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.1.0</version>
        <scope>provided</scope>
```

```
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.12.1</version>
    </dependency>
    <dependency>
        <groupId>commons-fileupload</groupId>
        <artifactId>commons-fileupload</artifactId>
        <version>1.3.1</version>
    </dependency>

    <!-- Spring5和Thymeleaf整合包 -->
    <dependency>
        <groupId>org.thymeleaf</groupId>
        <artifactId>thymeleaf-spring5</artifactId>
        <version>3.0.12.RELEASE</version>
    </dependency>
</dependencies>
```

③创建表

```
CREATE TABLE `t_emp` (
    `emp_id` int(11) NOT NULL AUTO_INCREMENT,
    `emp_name` varchar(20) DEFAULT NULL,
    `age` int(11) DEFAULT NULL,
    `sex` char(1) DEFAULT NULL,
    `email` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`emp_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

3、配置web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

<!--    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PUT和DELETE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-->
    <servlet>
        <servlet-name>DispatcherServlet</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<!--        设置SpringMVC的配置文件的位置和名称-->
        <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>DispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

<!--    设置Spring的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的监听器-->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
</web-app>
```

4、创建SpringMVC的配置文件并配置

```
<!--扫描组件-->
<context:component-scan base-package="com.atguigu.ssm.controller">
</context:component-scan>

<!--配置视图解析器-->
```

```
<bean id="viewResolver"
class="org.thymeleaf.spring5.view.ThymeleafViewResolver">
    <property name="order" value="1"/>
    <property name="characterEncoding" value="UTF-8"/>
    <property name="templateEngine">
        <bean class="org.thymeleaf.spring5.SpringTemplateEngine">
            <property name="templateResolver">
                <bean
class="org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver">
                    <!-- 视图前缀 -->
                    <property name="prefix" value="/WEB-INF/templates/"/>
                    <!-- 视图后缀 -->
                    <property name="suffix" value=".html"/>
                    <property name="templateMode" value="HTML5"/>
                    <property name="characterEncoding" value="UTF-8" />
                </bean>
            </property>
        </bean>
    </property>
</bean>

<!-- 配置访问首页的视图控制 -->
<mvc:view-controller path="/" view-name="index"></mvc:view-controller>

<!-- 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 />
```

5、搭建MyBatis环境

①创建属性文件jdbc.properties

```
jdbc.user=root
jdbc.password=atguigu
jdbc.url=jdbc:mysql://localhost:3306/ssm?serverTimezone=UTC
jdbc.driver=com.mysql.cj.jdbc.Driver
```

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"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <settings>
        <!--将下划线映射为驼峰-->
        <setting name="mapUnderscoreToCamelCase" value="true"/>
    </settings>

    <plugins>
        <!--配置分页插件-->
```

```
        <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>
    </plugins>
</configuration>
```

③创建Mapper接口和映射文件

```
public interface EmployeeMapper{
    List<Employee> getEmployeeList();
}
```

```
<?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="com.atguigu.ssm.mapper.EmployeeMapper">
   <select id="getEmployeeList" resultType="Employee">
       select * from t_emp
   </select>
</mapper>
```

④创建日志文件log4j.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>
```

6、创建Spring的配置文件并配置

```
<?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"
       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">
    <!--扫描组件-->
    <context:component-scan base-package="com.atguigu.ssm">
        <context:exclude-filter type="annotation"

expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 引入jdbc.properties -->
    <context:property-placeholder location="classpath:jdbc.properties">
</context:property-placeholder>

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

    <!-- 配置用于创建SqlSessionFactory的工厂bean -->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">

        <!-- 设置MyBatis配置文件的路径（当MyBatis配置文件没有配置时可以不设置） -->
        <property name="configLocation" value="classpath:mybatis-config.xml">
</property>

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

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

        <!--
             设置映射文件的路径
             若映射文件所在路径和mapper接口所在路径一致，则不需要设置
         -->

        <!--
         <property name="mapperLocations" value="classpath:mapper/*.xml">
</property>
         -->

    </bean>
```

```
    <!--
        配置mapper接口的扫描配置
        由mybatis-spring提供，可以将指定包下所有的mapper接口创建动态代理
        并将这些动态代理作为IOC容器的bean管理
     -->

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.atguigu.ssm.mapper"></property>
    </bean>
</beans>
```

7、测试功能

①创建组件

1）实体类Employee

```
public class Employee{
    private Integer empId;
    private String empName;
    private Integer age;
    private String sex;
    private String email;
    //构造器
    //get
    //set
}
```

2）创建控制层组件EmployeeController

```
@Controller
public class EmployeeController{
    @Autowired
    private EmployeeService employeeService;

    @RequestMapping(value = "/employee/page/{pageNum}", method =
                    RequestMethod.GET)

    public String getEmployeeList(Model model, @PathVariable("pageNum") Integer
pageNum){
        PageInfo<Employee> page = employeeService.getEmployeeList(pageNum);
        model.addAttribute("page", page);
        return "employee_list";
    }
}
```

3）创建接口EmployeeService

可以继承父类 IService

```
public interface EmployeeService{
    PageInfo<Employee> getEmployeeList(Integer pageNum);
}
```

4）创建实现类EmployeeServiceImpl

可以继承实现父类 ServiceImpl<Mappeer,pojo>

```
@Service
public class EmployeeServiceImpl implements EmployeeService {
   @Autowired
   private EmployeeMapper employeeMapper;

   @Override
   public PageInfo<Employee> getEmployeeList(Integer pageNum) {
       PageHelper.startPage(pageNum, 4);
       List<Employee> list = employeeMapper.getEmployeeList();
       PageInfo<Employee> page = new PageInfo<>(list, 5);
       return page;
   }
}
```

②创建页面

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
   <head>
       <meta charset="UTF-8">
       <title>Employee Info</title>
       <link rel="stylesheet" th:href="@{/static/css/index_work.css}">
   </head>
   <body>
       <table>
           <tr>
               <th colspan="6">Employee Info</th>
           </tr>
           <tr>
               <th>emp_id</th>
               <th>emp_name</th>
               <th>age</th>
               <th>sex</th>
               <th>email</th>
               <th>options</th>
           </tr>
           <tr th:each="employee : ${page.list}">
               <td th:text="${employee.empId}"></td>
               <td th:text="${employee.empName}"></td>
               <td th:text="${employee.age}"></td>
               <td th:text="${employee.sex}"></td>
```

```
                <td th:text="${employee.email}"></td>
                <td>
                    <a href="">delete</a>
                    <a href="">update</a>
                </td>
            </tr>
            <tr>
                <td colspan="6">
                    <span th:if="${page.hasPreviousPage}">
                        <a th:href="@{/employee/page/1}">首页</a>
                        <a th:href="@{'/employee/page/'+${page.prePage}}">上一页
</a>
                    </span>
                    <span th:each="num : ${page.navigatepageNums}">
                        <a th:if="${page.pageNum==num}"
                           th:href="@{'/employee/page/'+${num}}"
th:text="'['+${num}+']'"
                           style="color:red;"></a>
                        <a th:if="${page.pageNum!=num}"
                           th:href="@{'/employee/page/'+${num}}" th:text="${num}
"></a>
                    </span>
                    <span th:if="${page.hasNextPage}">
                        <a th:href="@{'/employee/page/'+${page.nextPage}}">下一页
</a>
                        <a th:href="@{'/employee/page/'+${page.pages}}">末页</a>
                    </span>
                </td>
            </tr>
        </table>
    </body>
</html>
```

③访问测试分页功能

localhost:8080/employee/page/1