第4天: SpringMVC

一、目标

1. 掌握ssm整合(SpringMVC+Spring+Mybatis)

二、SSM整合

> 所谓SSM，即spring MVC + spring +mybatis，标准的MVC设计模式，将整个系统划分为表现层、controller层、service层、DAO层四层，因此，在搭建框架时项目的目录结构最好要包含这些，方便管理和查看。当然了，spring MVC、spring、mybatis各司其职，在整个框架中有着不同的作用

- Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架，主要实现 **业务对象管理**;
- Spring MVC框架，通过实现MVC模式将数据、业务与展现进行分离，主要负责 **请求的转发和视图管理**;
- MyBatis 是一个基于Java的持久层框架，作为 **数据对象的持久化** 引擎;

2.1 项目分层

- 控制层 (controller): SpringMVC
- 业务层 (service): Spring
- 持久层 (dao): Mybatis

2.2 环境准备

2.2.1 数据准备

```
1  CREATE TABLE `account` (
2    `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键',
3    `uid` int(11) DEFAULT '1' COMMENT '用户编号',
4    `money` decimal(10,2) DEFAULT '0.00' COMMENT '余额',
5    PRIMARY KEY (`id`)
6  ) ENGINE=InnoDB AUTO_INCREMENT=106 DEFAULT CHARSET=utf8;
7
8  -- ---------------------------
9  -- Records of account
10 -- ---------------------------
11 INSERT INTO `account` VALUES ('1', '1', '10.00');
12 INSERT INTO `account` VALUES ('2', '10', '0.00');
13 INSERT INTO `account` VALUES ('3', '24', '99.00');
```

2.2.2 项目环境

- 工程名称: mvc-day03-ssm
- 添加依赖: pom.xml

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

```xml
  5      <parent>
  6          <artifactId>mvc3</artifactId>
  7          <groupId>com.itheima</groupId>
  8          <version>1.0-SNAPSHOT</version>
  9      </parent>
 10      <modelVersion>4.0.0</modelVersion>
 11
 12      <artifactId>mvc-day03-ssm</artifactId>
 13
 14      <dependencies>
 15          <!-- 1. Spring相关 依赖 -->
 16          <dependency>
 17              <groupId>org.springframework</groupId>
 18              <artifactId>spring-context</artifactId>
 19              <version>5.1.7.RELEASE</version>
 20          </dependency>
 21          <dependency>
 22              <groupId>org.springframework</groupId>
 23              <artifactId>spring-tx</artifactId>
 24              <version>5.1.7.RELEASE</version>
 25          </dependency>
 26          <dependency>
 27              <groupId>org.aspectj</groupId>
 28              <artifactId>aspectjweaver</artifactId>
 29              <version>1.8.13</version>
 30          </dependency>
 31          <dependency>
 32              <groupId>org.springframework</groupId>
 33              <artifactId>spring-jdbc</artifactId>
 34              <version>5.1.7.RELEASE</version>
 35          </dependency>
 36          <dependency>
 37              <groupId>mysql</groupId>
 38              <artifactId>mysql-connector-java</artifactId>
 39              <version>8.0.17</version>
 40          </dependency>
 41          <dependency>
 42              <groupId>com.alibaba</groupId>
 43              <artifactId>druid</artifactId>
 44              <version>1.1.12</version>
 45          </dependency>
 46          <!-- 2. SpringMVC相关 依赖 -->
 47          <dependency>
 48              <groupId>org.springframework</groupId>
 49              <artifactId>spring-web</artifactId>
 50              <version>5.1.7.RELEASE</version>
 51          </dependency>
 52          <dependency>
 53              <groupId>org.springframework</groupId>
 54              <artifactId>spring-webmvc</artifactId>
 55              <version>5.1.7.RELEASE</version>
 56          </dependency>
 57          <!-- 3. Mybatis相关 依赖 -->
 58          <dependency>
 59              <groupId>org.mybatis</groupId>
```

```xml
63          <dependency>
64              <groupId>org.mybatis</groupId>
65              <artifactId>mybatis</artifactId>
66              <version>3.5.2</version>
67          </dependency>
68          <!-- 4. Jsp相关 依赖 -->
69          <dependency>
70              <groupId>javax.servlet</groupId>
71              <artifactId>jsp-api</artifactId>
72              <version>2.0</version>
73              <scope>provided</scope>
74          </dependency>
75          <dependency>
76              <groupId>javax.servlet</groupId>
77              <artifactId>servlet-api</artifactId>
78              <version>2.5</version>
79              <scope>provided</scope>
80          </dependency>
81          <dependency>
82              <groupId>jstl</groupId>
83              <artifactId>jstl</artifactId>
84              <version>1.2</version>
85          </dependency>
86          <!-- 5. SpringTest 依赖 -->
87          <dependency>
88              <groupId>org.springframework</groupId>
89              <artifactId>spring-test</artifactId>
90              <version>5.1.7.RELEASE</version>
91          </dependency>
92          <dependency>
93              <groupId>junit</groupId>
94              <artifactId>junit</artifactId>
95              <version>4.12</version>
96          </dependency>
97          <!-- 6. Log4j 依赖 -->
98          <dependency>
99              <groupId>log4j</groupId>
100             <artifactId>log4j</artifactId>
101             <version>1.2.17</version>
102         </dependency>
103         <dependency>
104             <groupId>org.slf4j</groupId>
105             <artifactId>slf4j-api</artifactId>
106             <version>1.7.26</version>
107         </dependency>
108         <dependency>
109             <groupId>org.slf4j</groupId>
110             <artifactId>slf4j-log4j12</artifactId>
111             <version>1.7.7</version>
112         </dependency>
113     </dependencies>
114 </project>
```

- 创建实体: com.itheima.ssm.domain.Account

```
3  /**
4   *  账户实体类.
5   *
6   * @author : Jason.lee
7   * @version : 1.0
8   */
9  public class Account {
10     private Integer id;
11     private Integer uid;
12     private Double money;
13
14     public Integer getId() {
15         return id;
16     }
17
18     public void setId(Integer id) {
19         this.id = id;
20     }
21
22     public Integer getUid() {
23         return uid;
24     }
25
26     public void setUid(Integer uid) {
27         this.uid = uid;
28     }
29
30     public Double getMoney() {
31         return money;
32     }
33
34     public void setMoney(Double money) {
35         this.money = money;
36     }
37
38     @Override
39     public String toString() {
40         return "Account{" +
41                 "id=" + id +
42                 ", uid=" + uid +
43                 ", money=" + money +
44                 '}';
45     }
46  }
```

2.3 单独Spring环境

2.3.1 业务层

- com.itheima.ssm.service.AccountService

```
1  package com.itheima.ssm.service;
2
3  import com.itheima.ssm.domain.Account;
4  import org.springframework.stereotype.Service;
```

```java
7    import java.util.List;
8
9    /**
10    * 账户业务.
11    *
12    * @author : Jason.lee
13    * @version : 1.0
14    */
15   @Service
16   public class AccountService {
17
18
19       /**
20        * 查询所有账户
21        */
22       public List<Account> findAll(){
23           System.out.println("查询所有账户业务..");
24           return null;
25       }
26
27       /**
28        * 保存账户
29        */
30       @Transactional
31       public void save(Account account){
32           System.out.println("保存账户业务..");
33       }
34   }
```

2.3.2 添加配置

- applicationContext.xml

```xml
1    <?xml version="1.0" encoding="UTF-8"?>
2    <beans xmlns="http://www.springframework.org/schema/beans"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xmlns:context="http://www.springframework.org/schema/context"
5           xmlns:tx="http://www.springframework.org/schema/tx"
6           xsi:schemaLocation="http://www.springframework.org/schema/beans
7           http://www.springframework.org/schema/beans/spring-beans.xsd
8           http://www.springframework.org/schema/context
9           https://www.springframework.org/schema/context/spring-
     context.xsd
10          http://www.springframework.org/schema/tx
11          http://www.springframework.org/schema/tx/spring-tx.xsd">
12
13       <!-- Spring单独环境 -->
14       <!-- 1. 扫描Spring组件注解 -->
15       <context:component-scan base-package="com.itheima.ssm.service"/>
16       <!-- 2. 开启Spring事务注解 -->
17       <tx:annotation-driven/>
18   </beans>
```

2.3.3 单元测试

```xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <beans xmlns="http://www.springframework.org/schema/beans"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xmlns:context="http://www.springframework.org/schema/context"
5          xmlns:tx="http://www.springframework.org/schema/tx"
6          xsi:schemaLocation="http://www.springframework.org/schema/beans
7          http://www.springframework.org/schema/beans/spring-beans.xsd
8          http://www.springframework.org/schema/context
9          https://www.springframework.org/schema/context/spring-
    context.xsd
10         http://www.springframework.org/schema/tx
11         http://www.springframework.org/schema/tx/spring-tx.xsd">
12
13      <!-- 数据库基础配置 -->
14      <context:property-placeholder location="classpath:db.properties"/>
15      <bean id="dataSource"
    class="com.alibaba.druid.pool.DruidDataSource">
16          <property name="driverClassName" value="${db.driver}"/>
17          <property name="url" value="${db.url}"/>
18          <property name="username" value="${db.username}"/>
19          <property name="password" value="${db.password}"/>
20      </bean>
21
22
23      <!-- Spring单独环境 -->
24      <!-- 1. 扫描Spring组件注解 -->
25      <context:component-scan base-package="com.itheima.ssm.service"/>
26      <!-- 2. 开启Spring事务注解 -->
27      <tx:annotation-driven/>
28      <!-- 3. 配置事务管理器 -->
29      <bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager
    ">
30          <property name="dataSource" ref="dataSource"/>
31      </bean>
32  </beans>
```

2.4 单独SpringMVC环境

2.4.1 控制层

- com.itheima.ssm.controller.AccountController2.4.2 添加配置

- web.xml

```xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xmlns="http://java.sun.com/xml/ns/javaee"
4       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5       version="2.5">
6
7
8       <!-- 1. SpringMVC前端控制器 -->
9       <!-- 1.1 处理的请求路径 -->
```

```
12          <url-pattern>/</url-pattern>
13      </servlet-mapping>
14      <!-- 1.2 处理请求的控制器 -->
15      <servlet>
16          <!-- 1.2.1 定义控制器 -->
17          <servlet-name>dispatcherServlet</servlet-name>
18          <servlet-
    class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
19          <!-- 1.2.2 设置控制器的配置 -->
20          <init-param>
21              <param-name>contextConfigLocation</param-name>
22              <param-value>classpath:springMVC.xml</param-value>
23          </init-param>
24      </servlet>
25  </web-app>
```

2.4.3 单元测试

- WEB-INF/jsp/account/list.jsp

```
1   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2   <%--
3     Created by IntelliJ IDEA.
4     User: Jason
5     Date: 2019/8/30
6     Time: 13:03
7     To change this template use File | Settings | File Templates.
8   --%>
9   <%@ page contentType="text/html;charset=UTF-8" language="java" %>
10  <html>
11  <head>
12      <title>账户列表</title>
13  </head>
14  <body>
15  <h3>账户列表</h3>
16  <table>
17      <tr>
18          <th>账户编号</th>
19          <th>用户编号</th>
20          <th>账户余额</th>
21      </tr>
22      <c:forEach items="${list}" var="account">
23          <tr>
24              <td>${account.id}</td>
25              <td>${account.uid}</td>
26              <td>${account.money}</td>
27          </tr>
28      </c:forEach>
29  </table>
30  </body>
31  </html>
```

2.5 SpringMVC与Spring整合

- 希望整合Spring获取到IOC中的业务对象注入到控制层中

- com.itheima.ssm.controller.AccountController

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

import com.itheima.ssm.domain.Account;
import com.itheima.ssm.service.AccountService;
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 java.util.List;

/**
 * 账户请求处理器(控制器).
 *
 * @author : Jason.lee
 * @version : 1.0
 */
@Controller
public class AccountController {


    /**
     * 希望整合Spring获取到IOC中的业务对象
     */
    @Autowired
    AccountService accountService;

    /**
     * 查询账户列表
     * @return 返回视图
     */
    @RequestMapping("list")
    public String list(Model model){
        List<Account> list = accountService.findAll();
        model.addAttribute("list", list);
        return "account/list";
    }
}
```

2.5.2 配置

- 配置Spring提供的监听器: 在容器启动时, 加载Spring配置文件启动IOC容器

- web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    version="2.5">
```

```
9       <!-- 1.1 处理的请求路径 -->
10      <servlet-mapping>
11          <servlet-name>dispatcherServlet</servlet-name>
12          <url-pattern>/</url-pattern>
13      </servlet-mapping>
14      <!-- 1.2 处理请求的控制器 -->
15      <servlet>
16          <!-- 1.2.1 定义控制器 -->
17          <servlet-name>dispatcherServlet</servlet-name>
18          <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
19          <!-- 1.2.2 设置控制器的配置 -->
20          <init-param>
21              <param-name>contextConfigLocation</param-name>
22              <param-value>classpath:springMVC.xml</param-value>
23          </init-param>
24      </servlet>
25
26      <!-- 2. Spring配置 -->
27      <!-- 2.1 配置Spring提供的ServletContext监听器：Servlet容器启动时触发 -->
28      <listener>
29          <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
30      </listener>
31      <!-- 2.2 设置监听器执行参数：Spring配置文件 -->
32      <context-param>
33          <param-name>contextConfigLocation</param-name>
34          <param-value>classpath:applicationContext.xml</param-value>
35      </context-param>
36  </web-app>
```

2.6 单独Mybatis环境

2.6.1 持久层

- com.itheima.ssm.dao.AccountDao

```java
1   package com.itheima.ssm.dao;
2
3   import com.itheima.ssm.domain.Account;
4   import org.apache.ibatis.annotations.Insert;
5   import org.apache.ibatis.annotations.Select;
6
7   import java.util.List;
8
9   /**
10   * 账户持久层.
11   *
12   * @author : Jason.lee
13   * @version : 1.0
14   */
15  public interface AccountDao {
```

```
18        *  查询所有账户
19        */
20       @Select("select * from account")
21       List<Account> findAll();
22
23       /**
24        *  保存账户
25        */
26       @Insert("insert into account(id, uid, money) values(#{id}, #{uid},
    #{money})")
27       void save(Account account);
28   }
```

2.6.2 添加配置

- sqlMapConfig.xml

```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
   "http://mybatis.org/dtd/mybatis-3-config.dtd">
3  <configuration>
4
5      <!-- 1. 加载数据库配置文件 -->
6      <properties resource="db.properties"/>
7
8      <settings>
9          <!-- 开启二级缓存 -->
10         <setting name="cacheEnabled" value="false"/>
11         <!-- 设置超时时间 -->
12         <setting name="defaultStatementTimeout" value="5"/>
13         <!-- 使用驼峰命名法 -->
14         <setting name="mapUnderscoreToCamelCase" value="true"/>
15         <!-- 允许JDBC自动生成主键 -->
16         <setting name="useGeneratedKeys" value="true"/>
17     </settings>
18
19     <!-- 2. 设置Mybatis环境 -->
20     <environments default="dev">
21         <environment id="dev">
22             <transactionManager type="JDBC"></transactionManager>
23             <dataSource type="pooled">
24                 <property name="driver" value="${db.driver}"/>
25                 <property name="url" value="${db.url}"/>
26                 <property name="username" value="${db.username}"/>
27                 <property name="password" value="${db.password}"/>
28             </dataSource>
29         </environment>
30     </environments>
31
32     <!-- 3. 给指定包下的接口动态生成持久层代理对象 -->
33     <mappers>
34         <package name="com.itheima.ssm.dao"/>
35     </mappers>
36 </configuration>
```

- SsmTests

```java
import com.itheima.ssm.dao.AccountDao;
import com.itheima.ssm.domain.Account;
import com.itheima.ssm.service.AccountService;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.apache.ibatis.session.SqlSessionManager;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mybatis.spring.SqlSessionFactoryBean;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import java.io.InputStream;
import java.util.List;

/**
 * SSM整合单元测试代码.
 *
 * @author : Jason.lee
 * @version : 1.0
 */
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath:applicationContext.xml")
public class SsmTests {

    @Autowired
    AccountService accountService;

    @Test
    public void testSpring (){
        accountService.findAll();
        Account account = new Account();
        account.setId(10);
        account.setUid(10);
        account.setMoney(10D);
        accountService.save(account);
    }

    @Test
    public void testMybatis (){
        // 1. 获取配置文件输入流
        InputStream in =
this.getClass().getClassLoader().getResourceAsStream("sqlMapConfig.xml"
);
        // 2. 创建SqlSessionFactoryBuilder
        SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(in);
        // 3. 获取SqlSession
        SqlSession sqlSession = factory.openSession();
        // 4. 获取持久层动态代理对象
```

```
53            List<Account> list = accountDao.findAll();
54            Account account = new Account();
55            account.setId(10);
56            account.setUid(10);
57            account.setMoney(10D);
58            accountDao.save(account);
59            // 6. 事务提交
60            sqlSession.commit();
61            sqlSession.close();
62        }
63    }
```

2.7 Spring与Mybatis整合

2.7.1 需求

- 希望整合Spring获取到IOC中的业务对象注入到业务层中

- com.itheima.ssm.service.AccountService

```java
1    package com.itheima.ssm.service;
2
3    import com.itheima.ssm.dao.AccountDao;
4    import com.itheima.ssm.domain.Account;
5    import org.springframework.beans.factory.annotation.Autowired;
6    import org.springframework.stereotype.Service;
7    import org.springframework.transaction.annotation.Transactional;
8
9    import java.util.List;
10
11   /**
12    * 账户业务.
13    *
14    * @author : Jason.lee
15    * @version : 1.0
16    */
17   @Service
18   public class AccountService {
19
20       /**
21        * 希望整合Spring获取到IOC中的持久层对象
22        */
23       @Autowired
24       AccountDao accountDao;
25
26
27       /**
28        * 查询所有账户
29        */
30       public List<Account> findAll(){
31           return accountDao.findAll();
32       }
33
34       /**
35        * 保存账户
```

```
39          accountDao.save(account);
40      }
41  }
```

2.7.2 配置

- applicationContext.xml

```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.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/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd">

    <!-- 数据库基础配置 -->
    <context:property-placeholder location="classpath:db.properties"/>
    <bean id="dataSource"
class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${db.driver}"/>
        <property name="url" value="${db.url}"/>
        <property name="username" value="${db.username}"/>
        <property name="password" value="${db.password}"/>
    </bean>


    <!-- Spring单独环境 -->
    <!-- 1. 扫描Spring组件注解 -->
    <context:component-scan base-package="com.itheima.ssm.service"/>
    <!-- 2. 开启Spring事务注解 -->
    <tx:annotation-driven/>
    <!-- 3. 配置事务管理器 -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager
">
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!-- Spring整合Mybatis -->
    <!-- 1. 创建SqlSessionFactory -->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <!-- 2. 创建持久层动态代理对象 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.itheima.ssm.dao"/>
    </bean>
</beans>
```

- 浏览器操作