任务三：SSM整合

课程任务目标

```
* 实现SSM框架整合
```

1.1 需求和步骤分析

需求

使用ssm框架完成对`account`表的增删改查操作。

步骤分析

```
1. 准备数据库和表记录

2. 创建web项目

3. 编写mybatis在ssm环境中可以单独使用

4. 编写spring在ssm环境中可以单独使用

5. spring整合mybatis

6. 编写springMVC在ssm环境中可以单独使用

7. spring整合springMVC
```

1.2 环境搭建

1）准备数据库和表记录

```sql
CREATE TABLE `account` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(32) DEFAULT NULL,
  `money` double DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;

insert  into `account`(`id`,`name`,`money`) values (1,'tom',1000),
(2,'jerry',1000);
```

2）创建web项目

1.3 编写mybatis在ssm环境中可以单独使用

需求：基于mybatis先来实现对account表的查询

1）相关坐标

```xml
<!--mybatis坐标-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.47</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid</artifactId>
    <version>1.1.15</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.1</version>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

2）Account实体

```java
public class Account {

    private Integer id;
    private String name;
    private Double money;
}
```

3）AccountDao接口

```java
public interface AccountDao {

    public List<Account> findAll();
}
```

4）AccountDao.xml映射

```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="com.lagou.dao.AccountDao">
    <select id="findAll" resultType="Account">
        select * from account
    </select>
</mapper>
```

5）mybatis核心配置文件

jdbc.properties

```properties
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/spring_db
jdbc.username=root
jdbc.password=root
```

SqlMapConfig.xml

```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>

    <!--加载properties-->
    <properties resource="jdbc.properties"/>

    <!--类型别名配置-->
    <typeAliases>
        <package name="com.lagou.domain"/>
    </typeAliases>

    <!--环境配置-->
    <environments default="mysql">
        <!--使用MySQL环境-->
        <environment id="mysql">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="${jdbc.driver}"/>
                <property name="url" value="${jdbc.url}"/>
                <property name="username" value="${jdbc.username}"/>
                <property name="password" value="${jdbc.password}"/>
            </dataSource>
        </environment>
    </environments>

    <!--加载映射-->
    <mappers>
        <package name="com.lagou.dao"/>
    </mappers>

</configuration>
```

6）测试代码

```java
public class MyBatisTest {

    @Test
    public void testMybatis() throws Exception {
        // 加载核心配置文件
        InputStream is = Resources.getResourceAsStream("SqlMapConfig.xml");
        // 获得sqlsession工厂对象
        SqlSessionFactory sqlSessionFactory = new
SqlSessionFactoryBuilder().build(is);
        // 获得sqlsession会话对象
        SqlSession sqlSession = sqlSessionFactory.openSession();
        // 获得mapper代理对象
        AccountDao accountDao = sqlSession.getMapper(AccountDao.class);
        // 执行
        List<Account> list = accountDao.findAll();
        for (Account account : list) {
            System.out.println(account);
        }
        // 释放资源
        sqlSession.close();
    }
}
```

1.4 编写spring在ssm环境中可以单独使用

1）相关坐标

```xml
<!--spring坐标-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.13</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
```

2) AccountService接口

```
public interface AccountService {

    public List<Account> findAll();
}
```

3) AccountServiceImpl实现

```
@Service
public class AccountServiceImpl implements AccountService {

    @Override
    public List<Account> findAll() {
        System.out.println("findAll执行了....");
        return null;
    }
}
```

4) spring核心配置文件

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:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!--注解组件扫描-->
    <context:component-scan base-package="com.lagou.service"/>

</beans>
```

5）测试代码

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath:applicationContext.xml")
public class SpringTest {

    @Autowired
    private AccountService accountService;

    @Test
    public void testSpring() throws Exception {
        List<Account> list = accountService.findAll();
        System.out.println(list);
    }
}
```

1.5 spring整合mybatis

1）整合思想

　将mybatis接口代理对象的创建权交给spring管理，我们就可以把dao的代理对象注入到service中，此时也就完成了spring与mybatis的整合了。

2）导入整合包

```
<!--mybatis整合spring坐标-->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.1</version>
</dependency>
```

3）spring配置文件管理mybatis

注意：此时可以将mybatis主配置文件删除。

```
<?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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd">
    <!--注解组件扫描-->
    <context:component-scan base-package="com.lagou.service"/>

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

    <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>

    <!--SqlSessionFactory创建交给spring的IOC容器-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--数据库环境配置-->
        <property name="dataSource" ref="dataSource"/>
        <!--类型别名配置-->
        <property name="typeAliasesPackage" value="com.lagou.domain"/>
        <!--如果要引入mybatis主配置文件，可以通过如下配置-->
        <!--<property name="configLocation"
value="classpath:SqlMapConfig.xml"/>-->
    </bean>


    <!--映射接口扫描配置，由spring创建代理对象，交给IOC容器-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.lagou.dao"/>
    </bean>
</beans>
```

4）修改AccountServiceImpl

```
@Service
public class AccountServiceImpl implements AccountService {

    @Autowired
    private AccountDao accountDao;

    @Override
    public List<Account> findAll() {

        return accountDao.findAll();
    }
}
```

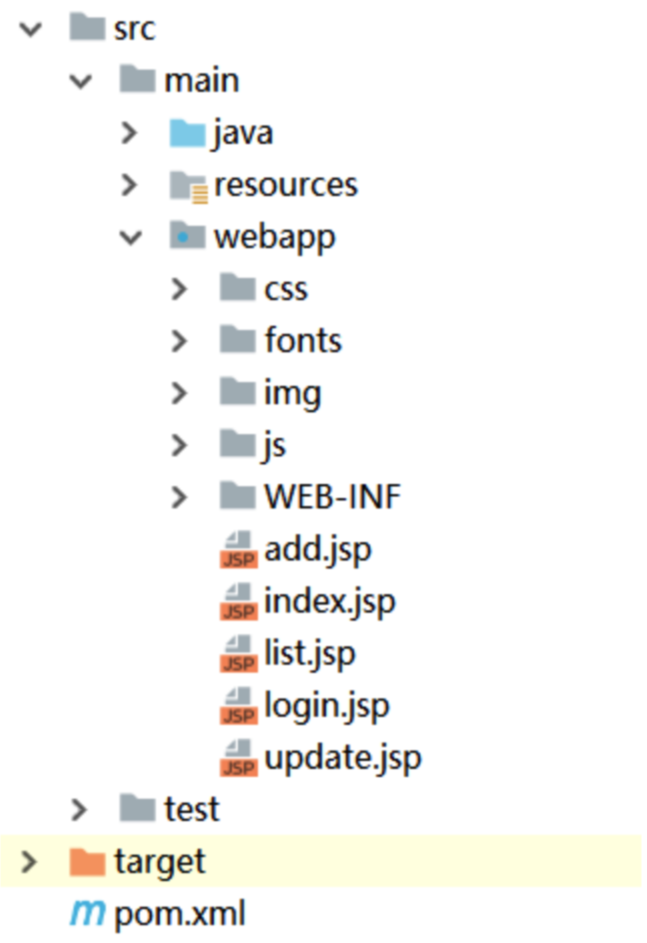
1.6 编写springMVC在ssm环境中可以单独使用

需求：访问到controller里面的方法查询所有账户，并跳转到list.jsp页面进行列表展示

1）相关坐标

```xml
<!--springMVC坐标-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.2</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

2）导入页面资源



```
src
  main
    java
    resources
    webapp
      css
      fonts
      img
      js
      WEB-INF
      add.jsp
      index.jsp
      list.jsp
      login.jsp
      update.jsp
  test
target
pom.xml
```

3）前端控制器DispathcerServlet

```
<?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_3_1.xsd"
         version="3.1">


    <!--前端控制器-->
    <servlet>
        <servlet-name>DispatcherServlet</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring-mvc.xml</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>DispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <!--post中文处理-->
    <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>
    </filter>
    <filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>


</web-app>
```

4）AccountController和 list.jsp

```
@Controller
@RequestMapping("/account")
public class AccountController {


    @RequestMapping("/findAll")
    public String findAll(Model model) {
        List<Account> list = new ArrayList<>();
        list.add(new Account(1,"张三",1000d));
        list.add(new Account(2,"李四",1000d));
        model.addAttribute("list", list);
        return "list";
    }

}
```

```
<c:forEach items="${list}" var="account">
    <tr>
        <td>
            <input type="checkbox" name="ids">
        </td>
        <td>${account.id}</td>
        <td>${account.name}</td>
        <td>${account.money}</td>
        <td>
            <a class="btn btn-default btn-sm" href="update.jsp">修改</a>&nbsp;
            <a class="btn btn-default btn-sm" href="">删除</a>
        </td>
    </tr>
</c:forEach>
```

5）springMVC核心配置文件

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <!--组件扫描-->
    <context:component-scan base-package="com.lagou.controller"/>

    <!--mvc注解增强-->
    <mvc:annotation-driven/>

    <!--视图解析器-->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <!--实现静态资源映射-->
    <mvc:default-servlet-handler/>
</beans>
```

1.7 spring整合springMVC

1）整合思想

spring和springMVC其实根本就不用整合，本来就是一家。

但是我们需要做到spring和web容器整合，让web容器启动的时候自动加载spring配置文件，web容器销毁的时候spring的ioc容器也销毁。

2）spring和web容器整合

ContextLoaderListener加载【掌握】

可以使用spring-web包中的ContextLoaderListener监听器，可以监听servletContext容器的创建和销毁，来同时创建或销毁IOC容器。

```xml
<!--spring 与 web容器整合-->
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>
```

3）修改AccountController

```java
@Controller
@RequestMapping("/account")
public class AccountController {

    @Autowired
    private AccountService accountService;

    @RequestMapping("/findAll")
    public String findAll(Model model) {

        List<Account> list = accountService.findAll();
        model.addAttribute("list", list);
        return "list";
    }
}
```

1.8 spring配置声明式事务

1）spring配置文件加入声明式事务

```xml
<!--事务管理器-->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
</bean>
<!--开启事务注解支持-->
<tx:annotation-driven/>
```

```java
@Service
@Transactional
public class AccountServiceImpl implements AccountService {
}
```

2) add.jsp

```html
<form action="${pageContext.request.contextPath}/account/save" method="post">
    <div class="form-group">
        <label for="name">姓名：</label>
        <input type="text" class="form-control" id="name" name="name"
placeholder="请输入姓名">
    </div>
    <div class="form-group">
        <label for="age">余额：</label>
        <input type="text" class="form-control" id="age" name="age"
placeholder="请输入余额">
    </div>

    <div class="form-group" style="text-align: center">
        <input class="btn btn-primary" type="submit" value="提交" />
        <input class="btn btn-default" type="reset" value="重置" />
        <input class="btn btn-default" type="button" onclick="history.go(-1)"
value="返回" />
    </div>
</form>
```

3) AccountController

```java
@RequestMapping("/save")
public String save(Account account){
    accountService.save(account);
    return "redirect:/account/findAll";
}
```

4) AccountService接口和实现类

```java
 public void save(Account account);
```

```java
@Service
@Transactional
public class AccountServiceImpl implements AccountService {

    @Override
    public void save(Account account) {
        accountDao.save(account);
    }
}
```

5) AccountDao

```java
void save(Account account);
```

6) AccountDao.xml映射

```xml
<insert id="save" parameterType="Account">
    insert into account (name, money) values (#{name}, #{money})
</insert>
```

1.9 修改操作

1.9.1 数据回显

① AccountController

```java
@RequestMapping("/findById")
public String findById(Integer id, Model model) {
    Account account = accountService.findById(id);
    model.addAttribute("account", account);
    return "update";
}
```

② AccountService接口和实现类

```java
Account findById(Integer id);
```

```java
@Override
public Account findById(Integer id) {
    return accountDao.findById(id);
}
```

③ AccountDao接口和映射文件

```
Account findById(Integer id);
```

```
<select id="findById" parameterType="int" resultType="Account">
    select * from account where id = #{id}
</select>
```

④ update.jsp

```
<form action="${pageContext.request.contextPath}/account/update" method="post">
    <input type="hidden" name="id" value="${account.id}">
    <div class="form-group">
        <label for="name">姓名：</label>
        <input type="text" class="form-control" id="name" name="name"
value="${account.name}" placeholder="请输入姓名">
    </div>
    <div class="form-group">
        <label for="money">余额：</label>
        <input type="text" class="form-control" id="money" name="money"
value="${account.money}" placeholder="请输入余额">
    </div>

    <div class="form-group" style="text-align: center">
        <input class="btn btn-primary" type="submit" value="提交" />
        <input class="btn btn-default" type="reset" value="重置" />
        <input class="btn btn-default" type="button" onclick="history.go(-1)"
value="返回" />
    </div>
</form>
```

1.9.2 账户更新

① AccountController

```
@RequestMapping("/update")
public String update(Account account){
    accountService.update(account);
    return "redirect:/account/findAll";
}
```

② AccountService接口和实现类

```
void update(Account account);
```

```
@Override
public void update(Account account) {
    accountDao.update(account);
}
```

③ AccountDao接口和映射文件

```
void update(Account account);
```

```
<update id="update" parameterType="Account">
    update account set name = #{name},money = #{money} where id = #{id}
</update>
```

1.10 批量删除

1) list.jsp

```
<script>
    $('#checkAll').click(function () {
        $('input[name="ids"]').prop('checked', $(this).prop('checked'));
    })

    $('#deleteBatchBtn').click(function () {
        if (confirm('您确定要删除吗?')) {
            if ($('input[name="ids"]:checked').length > 0) {
                $('#deleteBatchForm').submit();
            } else {
                alert('想啥呢？ ')
            }
        }
    })
</script>
```

2) AccountController

```
@RequestMapping("/deleteBatch")
public String deleteBatch(Integer[] ids) {
    accountService.deleteBatch(ids);
    return "redirect:/account/findAll";
}
```

3) AccountService接口和实现类

```
void deleteBatch(Integer[] ids);
```

```
@Override
public void deleteBatch(Integer[] ids) {
    accountDao.deleteBatch(ids);
}
```

4）AccountDao接口和映射文件

```
void deleteBatch(Integer[] ids);
```

```
<delete id="deleteBatch" parameterType="int">
    delete from account
    <where>
        <foreach collection="array" open="id in(" close=")" separator=","
item="id">
            #{id}
        </foreach>
    </where>
</delete>
```