实验八：框架整合

一、 **实验目的：**熟练掌握框架整合

二、 **预习要求：**预习第 14 章框架整合

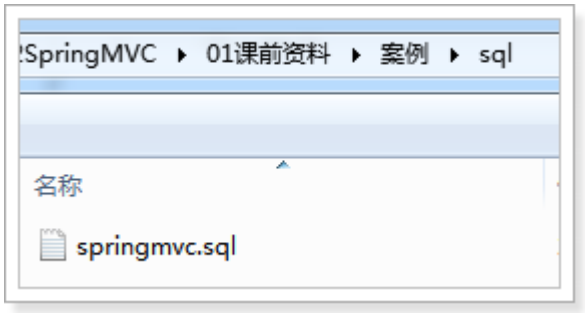
三、 **实验内容：**

1. 框架整合

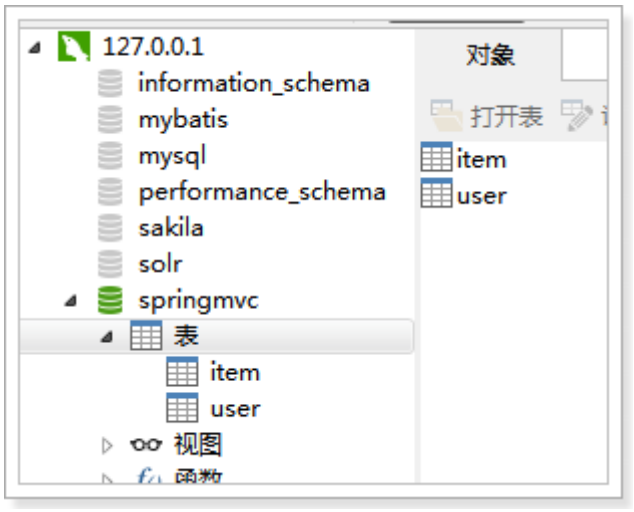
四、 **实验方法和步骤：**

1.1. 创建数据库表

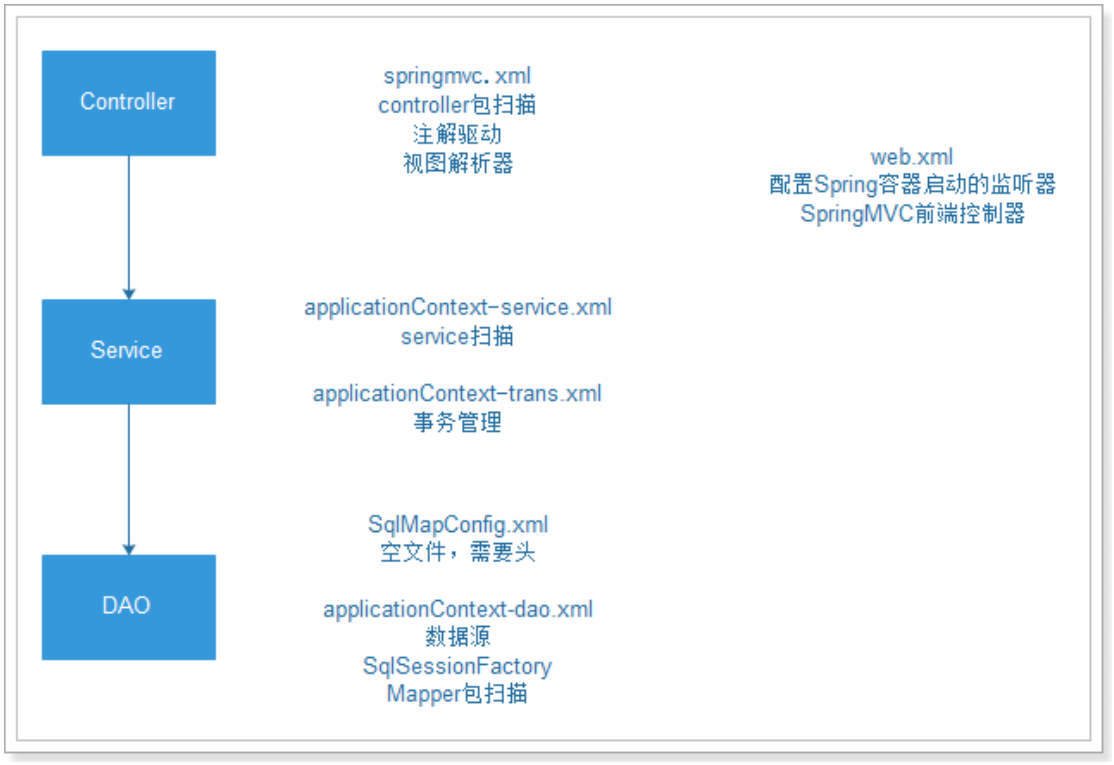
sql 脚本：



创建数据库 springmvc，导入到数据库中：



1.2. 整合思路



Dao 层：

1、applicationContext-dao.xml

a) 数据库连接池

b) SqlSessionFactory 对象，需要 spring 和 mybatis 整合包下的。

c) 配置 mapper 文件扫描器。

Service 层：

1、applicationContext-service.xml 包扫描器，扫描添加了@service 注解的类。

2、applicationContext-trans.xml 配置事务。

Controller 层：

1、Springmvc.xml

a) 包扫描器，扫描添加了@Controller 注解的类。

b) 配置注解驱动

c) 配置视图解析器

Web.xml 文件：

1、配置 spring 监听器

2、配置 SpringMVC 前端控制器。

1.3. 创建工程

导入 Jar 包

1.4. 加入配置文件

在其下创建 mybatis 和 spring 文件夹，用来存放配置文件

1.4.1. sqlMapConfig.xml

整合中不需要 mybatis 核心配置文件

1.4.2. applicationContext-dao.xml

配置数据源、配置 SqlSessionFactory、mapper 扫描器。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
    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/context
http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
    ">

    <!-- 加载配置文件 -->
    <context:property-placeholder
location="classpath:jdbc.properties" />

    <!-- 数据库连接池 -->
    <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 SqlSessionFactory -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 数据库连接池 -->
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!-- 配置 Mapper 扫描 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!-- 配置 Mapper 扫描包 -->
        <property name="basePackage" value="com.haust.dao" />
    </bean>

</beans>
```

1.4.3. jdbc.properties

配置数据库相关信息

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/springmvc?characterEncoding=utf-8
jdbc.username=root
jdbc.password=123
```

1.4.4. applicationContext-service.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
    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/context
http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
    ">
```

```
    <!-- 配置 service 扫描 -->
    <context:component-scan base-package="com.haust.service" />

</beans>
```

1.4.5. applicationContext-trans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
    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/context
http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd
    ">

    <!-- 事务管理器 -->
    <bean id="transactionManager"

    class="org.springframework.jdbc.datasource.DataSourceTransactionM
anager">
        <!-- 数据源 -->
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!-- 通知 -->
    <tx:advice id="txAdvice" transaction-
manager="transactionManager">
        <tx:attributes>
            <!-- 传播行为 -->
            <tx:method name="save*" propagation="REQUIRED" />
            <tx:method name="insert*" propagation="REQUIRED" />
            <tx:method name="delete*" propagation="REQUIRED" />
            <tx:method name="update*" propagation="REQUIRED" />
            <tx:method name="find*" propagation="SUPPORTS" read-
```

```
only="true" />
          <tx:method name="get*" propagation="SUPPORTS" read-
only="true" />
          <tx:method name="query*" propagation="SUPPORTS" read-
only="true" />
       </tx:attributes>
    </tx:advice>

    <!-- 切面 -->
    <aop:config>
       <aop:advisor advice-ref="txAdvice"
          pointcut="execution(* com.haust.service.*.*(..))" />
    </aop:config>

</beans>
```

1.4.6. springmvc.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:p="http://www.springframework.org/schema/p"
    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/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">

    <!-- 配置 controller 扫描包 -->
    <context:component-scan base-package="com.haust.controller" />

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

    <!-- Example: prefix="/WEB-INF/jsp/", suffix=".jsp",
viewname="test" ->
       "/WEB-INF/jsp/test.jsp" -->
    <!-- 配置视图解析器 -->
    <bean
    class="org.springframework.web.servlet.view.InternalResourceViewR
esolver">
```

```
        <!-- 配置逻辑视图的前缀 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- 配置逻辑视图的后缀 -->
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

1.4.7. web.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"
    id="WebApp_ID" version="2.5">
    <display-name>springmvc-web</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- 配置 spring -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext*.xml</param-value>
    </context-param>

    <!-- 使用监听器加载 Spring 配置文件 -->
    <listener>    <listener-
class>org.springframework.web.context.ContextLoaderListener</listene
r-class>
    </listener>
    <!-- 配置 SrpingMVC 的前端控制器 -->
    <servlet>
        <servlet-name>springmvc-web</servlet-name>      <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/springmvc.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc-web</servlet-name>
```

```
        <!-- 配置所有以 action 结尾的请求进入 SpringMVC -->
        <url-pattern>*.action</url-pattern>
    </servlet-mapping>
</web-app>
```

1.5. 加入 jsp 页面

拷贝实验七的 itemList.jsp 到项目中

2. 实现商品列表显示

2.1. 需求

实现商品查询列表，从 mysql 数据库查询商品信息。

2.2. DAO 开发

利用动态代理 Mapper 实现数据层的开发

2.3. ItemService 接口

```
public interface ItemService {

    /**
     * 查询商品列表
     *
     * @return
     */
    List<Item> queryItemList();

}
```

2.4. ItemServiceImpl 实现类

```
@Service
public class ItemServiceImpl implements ItemService {
    @Autowired
    private ItemMapper itemMapper;

    @Override
    public List<Item> queryItemList() {
```

```
        // 从数据库查询商品数据
        List<Item> list = this.itemMapper.selectByExample(null);

        return list;
    }

}
```

2.5. ItemController

```
@Controller
public class ItemController {

    @Autowired
    private ItemService itemService;
    /**
     * 显示商品列表
     *
     * @return
     */
    @RequestMapping("/itemList")
    public ModelAndView queryItemList() {
        // 获取商品数据
        List<Item> list = this.itemService.queryItemList();

        ModelAndView modelAndView = new ModelAndView();
        // 把商品数据放到模型中
        modelAndView.addObject("itemList", list);
        // 设置逻辑视图
        modelAndView.setViewName("itemList");
        return modelAndView;
    }

}
```

3. 用纯注解方式实现 ssm 项目整合

3.1 JdbcConfig

```

/*
等同于
<context:property-placeholder location="classpath*:jdbc.properties"/>
```

```
 */
@PropertySource("classpath:jdbc.properties")
public class JdbcConfig {
    /*
    使用注入的形式，读取 properties 文件中的属性值，
    等同于<property name="*******" value="${jdbc.driver}"/>
     */
    @Value("${jdbc.driverClassName}")
    private String driver;
    @Value("${jdbc.url}")
    private String url;
    @Value("${jdbc.username}")
    private String userName;
    @Value("${jdbc.password}")
    private String password;

    /*定义 dataSource 的 bean， 等同于
   <bean                                                id="dataSource"
class="com.alibaba.druid.pool.DruidDataSource">
    */
    @Bean("dataSource")
    public DataSource getDataSource() {
        //创建对象
        DruidDataSource ds = new DruidDataSource();
               /*
           等同于 set 属性注入<property  name="driverClassName"
value="driver"/>
            */
        ds.setDriverClassName(driver);
        ds.setUrl(url);
        ds.setUsername(userName);
        ds.setPassword(password);
        return ds;
    }
}
```

3.2 MyBatisConfig

```
public class MyBatisConfig {
    /*
    定义 MyBatis 的核心连接工厂 bean，
    等同于<bean class="org.mybatis.spring.SqlSessionFactoryBean">
     参数使用自动装配的形式加载 dataSource，
    为 set 注入提供数据源，dataSource 来源于 JdbcConfig 中的配置
```

```
     */
    @Bean
    public SqlSessionFactoryBean getSqlSessionFactoryBean(
            @Autowired DataSource dataSource) {
        SqlSessionFactoryBean ssfb = new SqlSessionFactoryBean();
        //等同于<property name="dataSource" ref="dataSource"/>
        ssfb.setDataSource(dataSource);
        return ssfb;
    }

    /*
    定义 MyBatis 的映射扫描，
    等                    同                    于                    <bean
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
     */
    @Bean
    public MapperScannerConfigurer getMapperScannerConfigurer() {
        MapperScannerConfigurer msc = new MapperScannerConfigurer();
        //等同于<property name="basePackage" value="com.itheima.dao"/>
        msc.setBasePackage("com.itheima.dao");
        return msc;
    }
}
```

3.3 SpringConfig

```
@Configuration
@Import({MyBatisConfig.class, JdbcConfig.class})
    /*
    等同于<context:component-scan base-package="com.itheima.service">
     */
@ComponentScan(value = "com.itheima.service")
    /*
    将 MyBatisConfig 类和 JdbcConfig 类交给 Spring 管理
     */
public class SpringConfig {
}
```

3.4 SpringMvcConfig

```
@Configuration
```

```
//      等      同      于      <context:component-scan      base-
package="com.itheima.controller"/>
@ComponentScan("com.itheima.controller")
//等同于<mvc:annotation-driven/>, 还不完全相同
@EnableWebMvc
public class SpringMvcConfig {
}
```

3.5 ServletContainersInitConfig

```
public class ServletContainersInitConfig extends
        AbstractAnnotationConfigDispatcherServletInitializer {
    /*
加载 Spring 配置类中的信息,
初始化 Spring 容器
*/
    protected Class<?>[] getRootConfigClasses() {
        return new Class[]{SpringConfig.class};
    }

    /*
 加载 Spring MVC 配置类中的信息,
 初始化 Spring MVC 容器
  */
    protected Class<?>[] getServletConfigClasses() {
        return new Class[]{SpringMvcConfig.class};
    }

    //配置 DispatcherServlet 的映射路径
    protected String[] getServletMappings() {
        return new String[]{"*.action"};
    }
}
```

五、 思考题：

六、 实验作业要求：

⑴实验目的：

⑵实验内容：

⑶实验结果：可以是运行结果截图或其他形式的结果展示

⑷问题及解决：实验中遇到的问题及解决方法。

⑸回答思考题提出的问题。