SSM整合

Spring整合其他几个

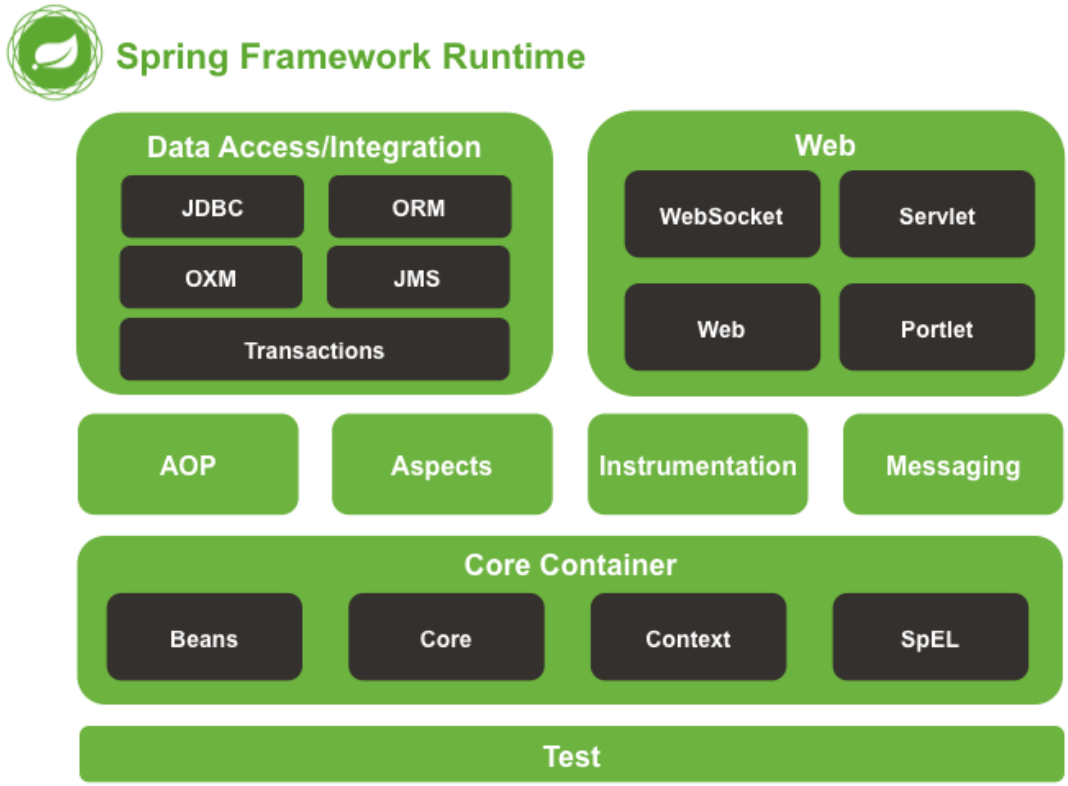
- SpringMVC
- MyBatis

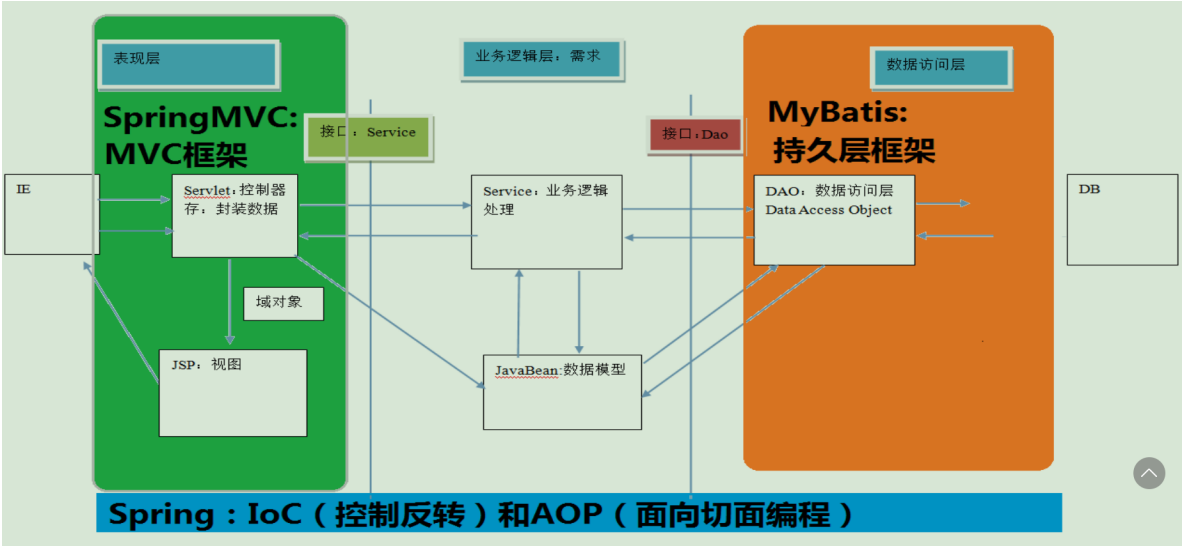
SpringIOC容器管理：

- service
- dao

SpringMVC：

- controller





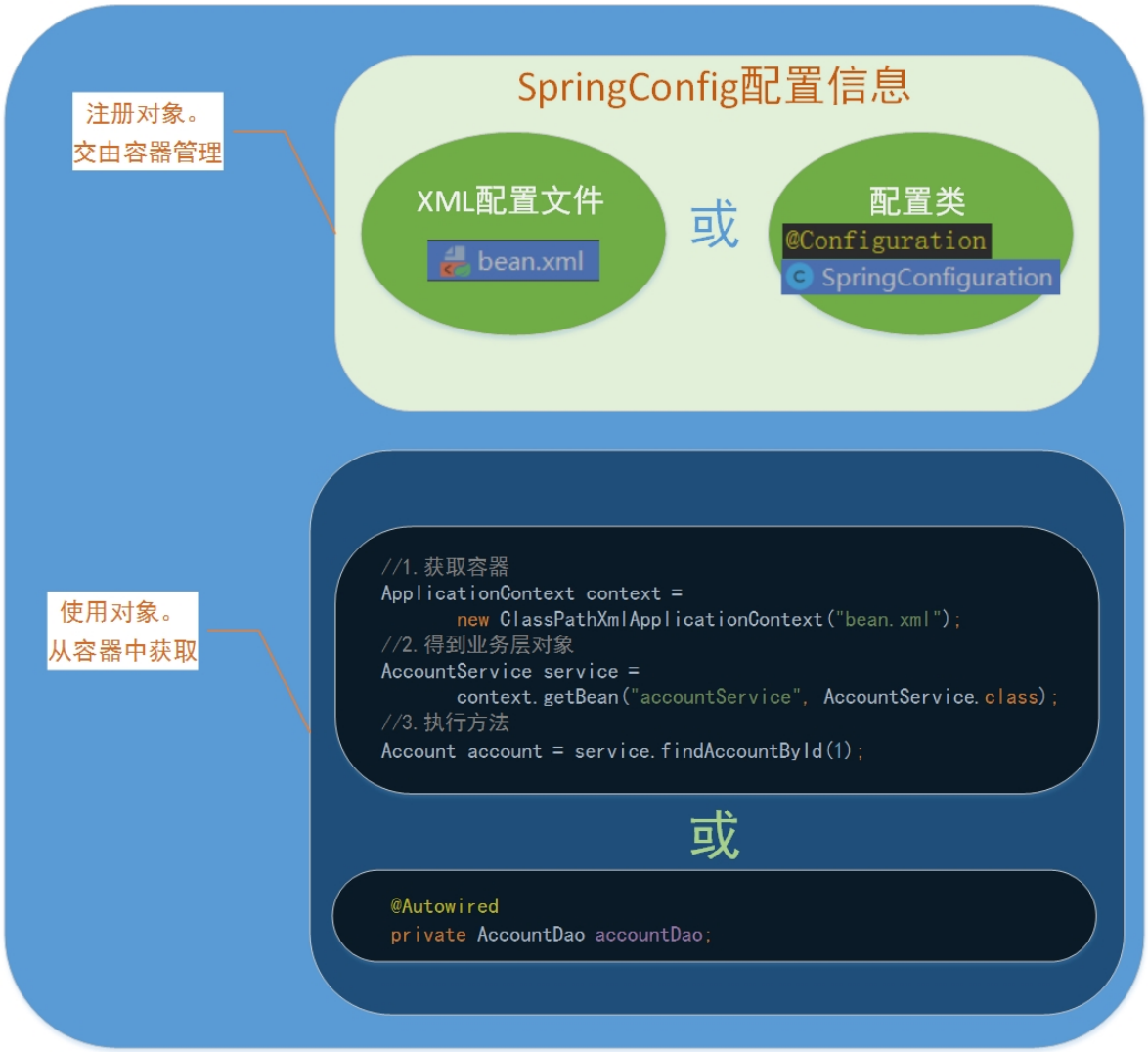
Spring需要

- service
- dao

依赖

- aspectjweaver
- spring-aop

- spring-context



配置文件|配置类

- 配置文件|配置类
    - 注册对象，交由容器管理

获取&使用 对象

- 从容器中获取对象，使用

SpringMVC需要

依赖

- spring-web

- spring-webmvc
- servlet-api
- jsp-api

- jstl

web.xml

- web.xml
    - 前端控制器
        - 加载SpringMVC.xml配置文件
    - 编码过滤器

SpringMVC.xml

- SpringMVC.xml(和Spring的配置文件是一样的)
    - 扫描包
    - 视图解析器
    - mvc注解支持
    - 类型转换
    - 静态资源不过滤
    - 拦截器注册
    - 文件解析器
    - 异常处理器

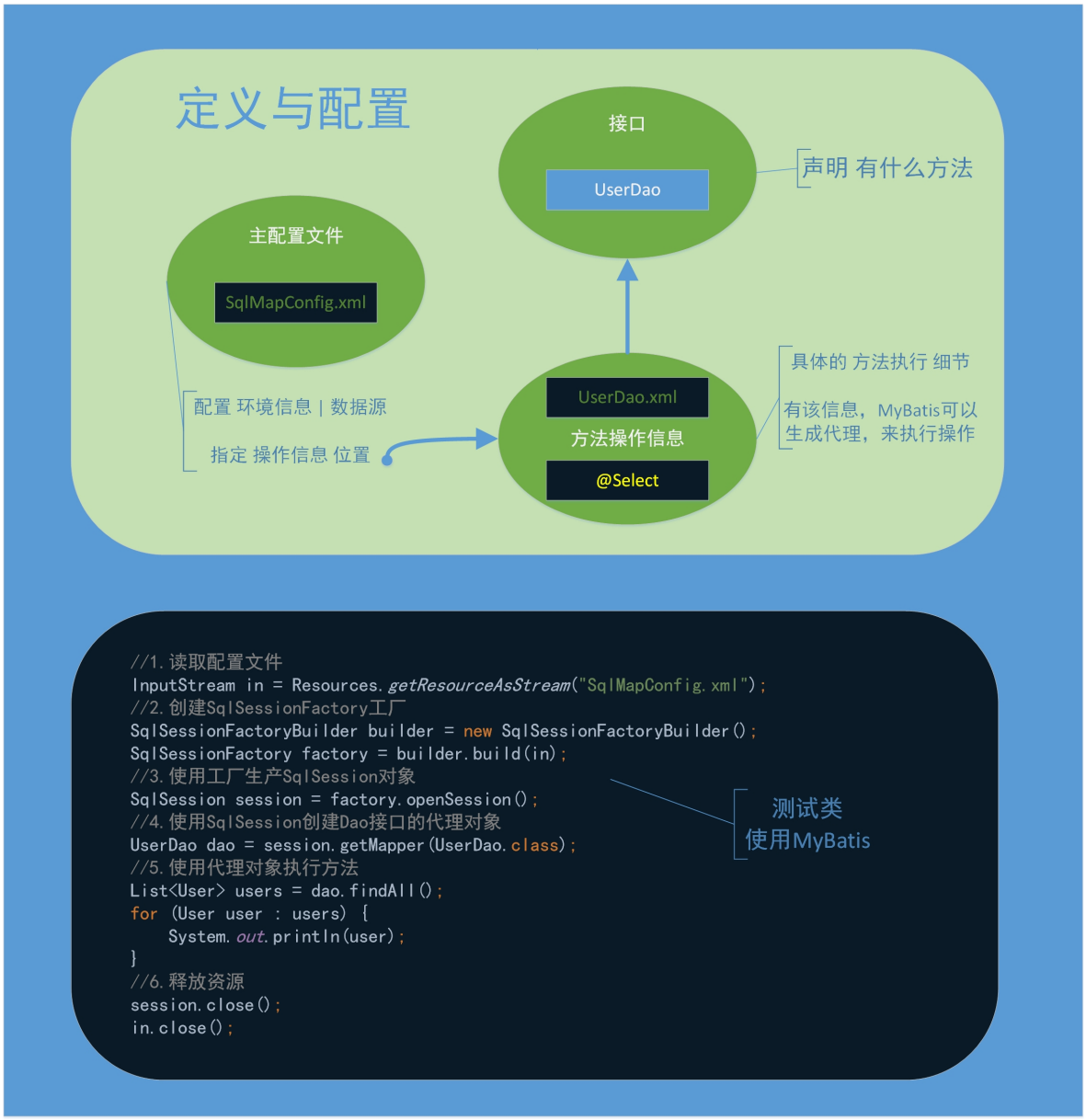
Controller

- Controller.java
    - @RequestMapping
    - @RequestParam
    - @RequestBody
    - @PathVariable
    - @RequestHeader
    - @CookieValue
    - @ModelAttribute
    - @SessionAttribute
    - @ReponseBody

MyBatis需要

依赖

- mysql-connector-java
- mybatis



- 主配置文件
- 接口：声明方法
- xml|注解：方法操作信息
- 测试类使用

主配置文件SqlMapConfig.xml

- 习惯上 SqlMapConfig.xml
- 数据源配置
- 指定映射信息位置
- 实体类别名

接口

- 声明方法

映射配置

- xml文件 或 接口注解
- SQL执行 相关信息

获取session&使用

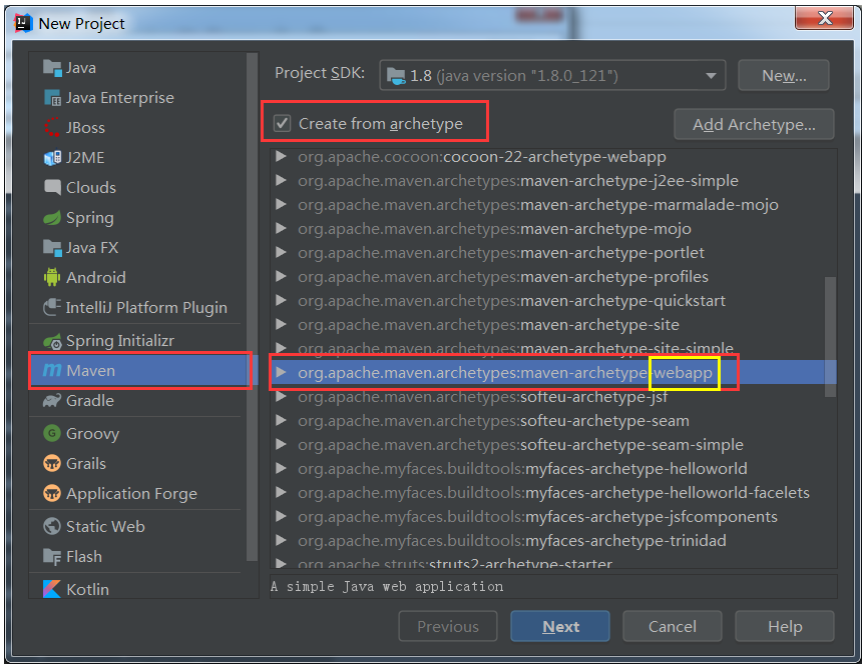
- 读取配置主配置文件，in
- Builder->factory->session
- 代理对象，执行操作

新建Maven-webapp工程

构建项目

骨架构建：Create from archetype

- org.apache.maven.archetypes:maven-archetype-**webapp**



解决webapp项目构建慢的问题：Properties+

- archetypeCatalog
- internal

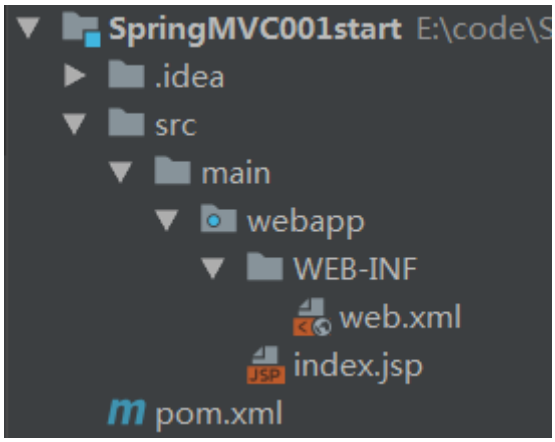
完善项目结构

构建完的项目，**目录结构是不全的。**

- `src`
    - `main`
        - `webapp`
            - `WEB-INF`

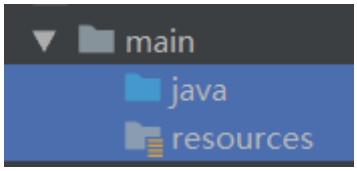
        - web.xml
    - index.jsp

- pom.xml

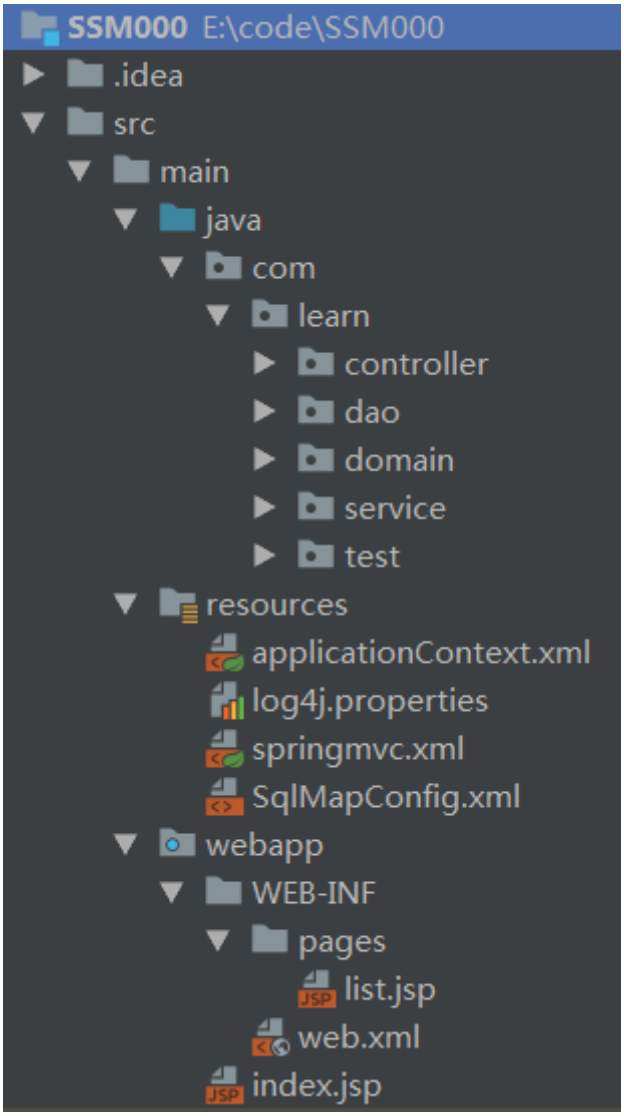


main下 新建 ：

- java
    - 右键：Mark Directory as
    - Sources Root ：源码根目录
- resources
    - 右键：Mark Directory as
    - Resources Root：资源根目录



最终结构



- `src`
    - `main`
        - `java`...com.learn
            - `controller`
            - `dao`
            - `domain`
            - `service`
            - `test`
        - `resources`
        - `webapp`
            - `WEB-INF`
            - index.jsp
- pom.xml

pom依赖

- aspectjweaver
- spring-aop

- spring-context

- spring-web
- spring-webmvc
- servlet-api
- jsp-api

- jstl

- spring-test

- mysql-connector-java
- c3p0

- spring-tx
- spring-jdbc

- mybatis
- mybatis-spring

- junit

- log4j
- slf4j-api
- slf4j-log4j12

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.grape</groupId>
  <artifactId>SSM000</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>SSM000 Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
```

```
    <spring.version>5.0.2.RELEASE</spring.version>
    <slf4j.version>1.6.6</slf4j.version>
    <log4j.version>1.2.12</log4j.version>
    <mysql.version>5.1.6</mysql.version>
    <mybatis.version>3.4.5</mybatis.version>
  </properties>


  <dependencies>

    <!-- spring -->
    <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>1.6.8</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>${spring.version}</version>
    </dependency>

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

    <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>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.0</version>
      <scope>provided</scope>
    </dependency>

    <!-- 这个... -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>

```

```
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-tx</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <!-- 用5.几的就有问题，可能我MySQL的版本高，不兼容吧 -->
      <version>8.0.15</version>
    </dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>compile</scope>
    </dependency>

    <!-- log start -->
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>${log4j.version}</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>${slf4j.version}</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>${slf4j.version}</version>
    </dependency>
    <!-- log end -->

    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>${mybatis.version}</version>
    </dependency>
    <!-- 用于整合 -->
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis-spring</artifactId>
      <version>1.3.0</version>
```

```
    </dependency>
    <dependency>
      <groupId>c3p0</groupId>
      <artifactId>c3p0</artifactId>
      <version>0.9.1.2</version>
      <type>jar</type>
      <scope>compile</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>SSM000</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
      <plugins>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_war_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

独立Spring

SpringIOC容器管理：

- service
- dao

`domain`.Account

```java
/**
 * 账户
 */
public class Account implements Serializable {

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

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getMoney() {
        return money;
    }

    public void setMoney(Double money) {
        this.money = money;
    }

    @Override
    public String toString() {
        return "Account{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", money=" + money +
                '}';
    }
}
```

`service.`AccountService

```java
public interface AccountService {

    /**
     * 查询所有账户
     * @return
     */
```

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

    /**
     * 保存账户信息
     * @param account
     */
    public void saveAccount(Account account);

}
```

`service.impl.` AccountServiceImpl

- 加注解@Service
- 注册，交给IOC容器进行管理

```
@Service("accountService")
public class AccountServiceImpl implements AccountService {

    @Override
    public List<Account> findAll() {
        System.out.println("业务层：查询所有账户");
        return null;
    }

    @Override
    public void saveAccount(Account account) {
        System.out.println("业务层：保存账户");
    }
}
```

`resources.` applicationContext.xml

开启注解扫描，Spring只负责 业务层 、持久层。

- service
- dao

```
<?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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置 哪些注解 不扫描 -->
        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>
</beans>
```

`test.`**TestSpring**

```
public class TestSpring {

    @Test
    public void run1() {
        //加载配置文件
        ApplicationContext context = new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        //获取对象
        AccountService service = context.getBean("accountService", AccountService.class);
        //调用方法
        service.findAll();
    }

}
```

独立SpringMVC

`resources.`**springmvc.xml**

- 开启注解扫描，SpringMVC负责web层
    - controller
- 配置 视图解析器
- 静态资源 (不过滤)
- 开启SpringMVC注解支持

```
<?xml version="1.0" encoding="UTF-8"?>
<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">

    <!-- 开启注解扫描,只扫描Controller注解 -->
    <context:component-scan base-package="com.learn">
        <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 配置视图解析器 -->
    <bean id="internalResourceViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>


    <!-- 设置静态资源 不过滤 -->
    <mvc:resources location="/css/" mapping="/css/**" />
    <mvc:resources location="/images/" mapping="/images/**" />
    <mvc:resources location="/js/" mapping="/js/**" />

    <!-- 开启SpringMVC注解的支持 -->
    <mvc:annotation-driven></mvc:annotation-driven>
</beans>
```

`webapp.WEB-INF.` web.xml

- 前端控制器
    - 加载springmvc.xml配置文件
    - 启动服务器，创建servlet
- 编码过滤器

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>

<!-- 配置前端控制器 -->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

```
    <!-- 加载SpringMVC.xml配置文件 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!-- 启动服务器 创建该Servlet对象 -->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

<!-- 解决中文乱码的过滤器 -->
  <filter>
    <filter-name>characterEncodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <!-- 设置编码集 -->
    <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>
```

`webapp.` index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <a href="account/findAll">测试SpringMVC 测试查询</a>
</body>
</html>
```

`webapp.WEB-INF.pages.` list.jsp

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %>
<%-- 引入 jstl的标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h3>查询了所有的账户信息</h3>
</body>
</html>
```

`controller.` AccountController

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

    @RequestMapping("/findAll")
    public String findAll() {
        System.out.println("表现层：查询所有账户信息");
        return "list";//跳到list.jsp
    }

}
```

独立MyBatis

`dao.` AccountDao

```java
/**
 * 账户dao接口
 */
public interface AccountDao {

    /**
     * 查询所有账户
     * @return
     */
    @Select("select * from account")
    List<Account> findAll();

    /**
     * 保存账户信息
     * @param account
     */
    @Insert("insert into account (name,money) values (#{name}, #{money})")
    void saveAccount(Account account);
```

```
19
20 }
```

`resources.` SqlMapConfig.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE configuration
3          PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4          "http://mybatis.org/dtd/mybatis-3-config.dtd">
5  <configuration>
6      <environments default="mysql">
7          <environment id="mysql">
8              <transactionManager type="JDBC"/>
9              <dataSource type="POOLED">
10                 <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
11                 <property name="url"
   value="jdbc:mysql://localhost:3306/db"/>
12                 <property name="username" value="root"/>
13                 <property name="password" value="xxxx"/>
14             </dataSource>
15         </environment>
16     </environments>
17
18     <!-- 引入映射配置:xml|注解 -->
19     <mappers>
20         <!-- <mapper class="com.learn.dao.AccountDao"/> -->
21         <!-- 该包下所有的dao接口都可以使用 -->
22         <package name="com.learn.dao"/>
23     </mappers>
24 </configuration>
```

`test.` MyBatis

```
1  public class TestMyBatis {
2
3      /**
4       * 测试查询
5       * @throws Exception
6       */
7      @Test
8      public void testFindAll() throws Exception {
9          //1.加载配置文件
10         InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
11         //2.创建工厂
12         SqlSessionFactory factory = new
   SqlSessionFactoryBuilder().build(in);
13         //3.创建session
14         SqlSession session = factory.openSession();
15
16         //4.获取代理对象
17         AccountDao dao = session.getMapper(AccountDao.class);
18         //5.查询
19         List<Account> accounts = dao.findAll();
```

```
        for (Account account : accounts) {
            System.out.println(account);
        }

        //释放资源
        session.close();
        in.close();
    }

    @Test
    public void testSaveAccount() throws Exception {
        //1.加载配置文件
        InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
        //2.创建工厂
        SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(in);
        //3.创建session
        SqlSession session = factory.openSession();

        //4.获取代理对象
        AccountDao dao = session.getMapper(AccountDao.class);

        Account account = new Account();
        account.setName("史睿生");
        account.setMoney(3000d);

        dao.saveAccount(account);
        session.commit();

        //释放资源
        session.close();
        in.close();
    }

}
```

Spring整合SpringMVC

- 在controller中 能 调用 service(由IOC管理)代表 整合成功
- ？：启动tomcat，根据 web.xml 做初始化
    - 前端控制器
        - 加载springmvc.xml
        - **？Spring.xml的配置文件没加载过**

Controller

- controller里调用service(IOC管理)，整合成功
- 加入 成员 accountService，注入
- 调用service方法

```
/**
 * web层
```

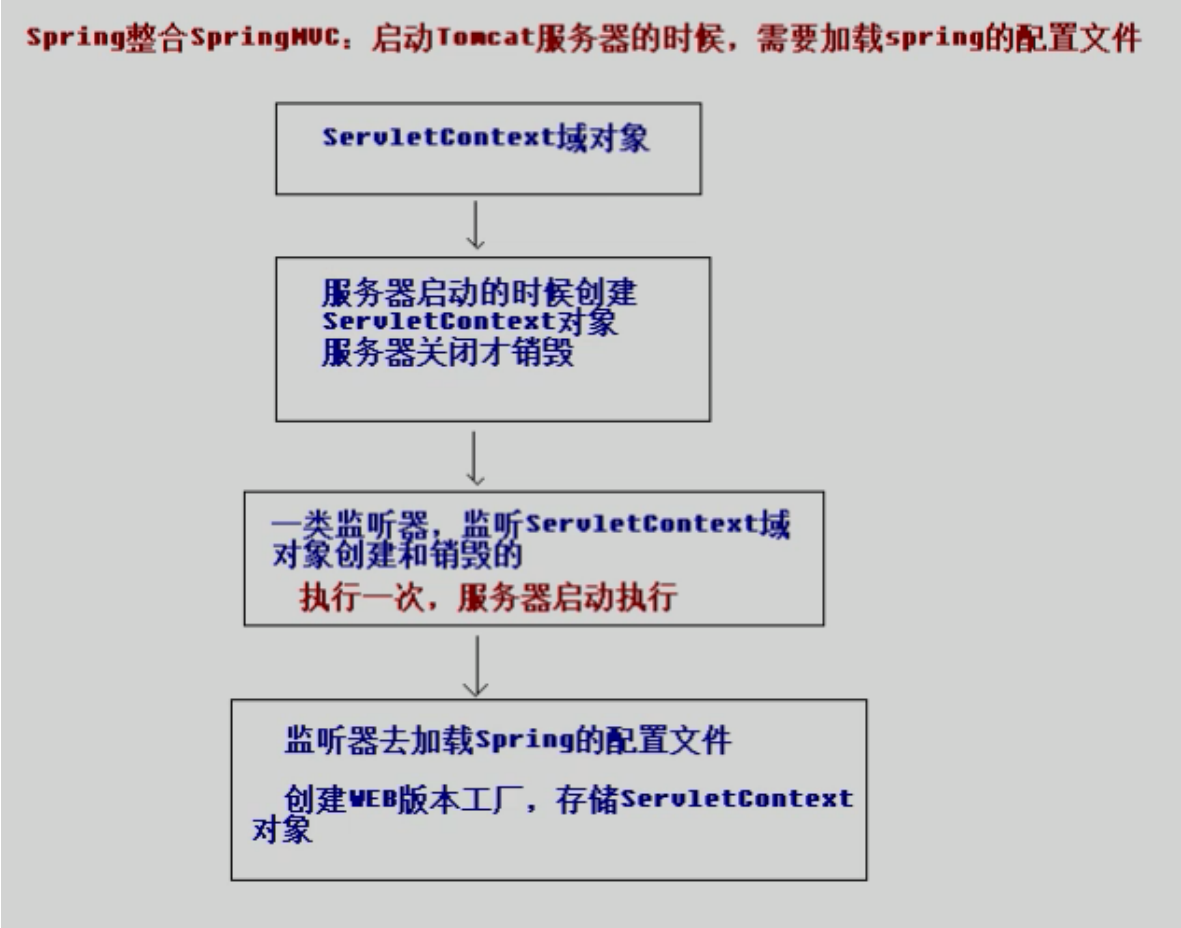
```
 */
@Controller
@RequestMapping("/account")
public class AccountController {
    @Autowired
    private AccountService accountService;

    @RequestMapping("/findAll")
    public String findAll() {
        System.out.println("表现层：查询所有账户信息");
        //调用service的方法
        accountService.findAll();
        return "list";//跳到list.jsp
    }

}
```

web.xml

- 监听器
    - 依赖 spring-web
    - 这里用于 加载 Spring.xml 其实是 applicationContext.xml




```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
```

```
  <display-name>Archetype Created Web Application</display-name>

<!-- 监听器 加载Spring.xml -->
  <!-- 默认只加载WEB-INF目录下的applicationContext.xml配置文件：1、复制一份过来2、传参数 -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <!-- 设置文件的路径 -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>


<!-- 配置前端控制器 -->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 加载SpringMVC.xml配置文件 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!-- 启动服务器 创建该Servlet对象 -->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

<!-- 解决中文乱码的过滤器 -->
  <filter>
    <filter-name>characterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <!-- 设置编码集 -->
    <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>
```

Spring整合MyBatis

Service能调用到dao对象。

dao 注册到 IOC中：

- 以前：测试类，自己加载SqlMapConfig.xml，生成代理
- 现在：代理对象 存到 IOC容器中

SqlMapConfig.xml废弃，内容转移到 applicationContext.xml

`resources.` applicationContext.xml

不用SqlMapConfig.xml了。

- 连接池
- 工厂对象SqlSessionFactory
    - 通过org.mybatis.spring.SqlSessionFactoryBean
- AccountDao接口所在包
    - 通过org.mybatis.spring.mapper.MapperScannerConfigurer
- 该有的都有了，别的就不需要我们操心了，Spring会自己做的

```
<?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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让
SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置要忽略的注解 -->
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

<!-- Spring整合MyBatis框架 SqlMapConfig的转移? -->
    <!-- 配置连接池 C3P0 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
        <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/db"/>
        <property name="user" value="root"/>
        <property name="password" value="xxxx"/>
```

```
    </bean>

    <!-- 配置SqlSessionFactory工厂对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <!-- 配置AccountDao接口所在的包 映射扫描配置的类 -->
    <bean id="mapperScanner"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.learn.dao"></property>
    </bean>

</beans>
```

`dao.`AccountDao

- 加个注解，注册一下。@Repository

```
/**
 * 账户dao接口
 */
@Repository//整合配置后才有用
public interface AccountDao {

    /**
     * 查询所有账户
     * @return
     */
    @Select("select * from account")
    List<Account> findAll();

    /**
     * 保存账户信息
     * @param account
     */
    @Insert("insert into account (name,money) values (#{name}, #{money})")
    void saveAccount(Account account);

}
```

`service.impl.`AccountServiceImpl

- service 能 调用 dao 说明 整合成功
- 加入 成员 dao
- 调用dao方法

```
@Service("accountService")
public class AccountServiceImpl implements AccountService {

    @Autowired
    private AccountDao accountDao;
```

```
6
7       @Override
8       public List<Account> findAll() {
9           System.out.println("业务层：查询所有账户");
10          return accountDao.findAll();
11      }
12
13      @Override
14      public void saveAccount(Account account) {
15          System.out.println("业务层：保存账户");
16          accountDao.saveAccount(account);
17      }
18  }
```

测试

`controller.` **AccountController**

- 能查到数据了
- 把数据 展示到 页面上

- Model
- 存数据

```
1   /**
2    * web层
3    */
4   @Controller
5   @RequestMapping("/account")
6   public class AccountController {
7       @Autowired
8       private AccountService accountService;
9
10      @RequestMapping("/findAll")
11      public String findAll(Model model) {
12          System.out.println("表现层：查询所有账户信息");
13          //调用service的方法
14          List<Account> accounts = accountService.findAll();
15          model.addAttribute("accounts", accounts);
16          return "list";//跳到list.jsp
17      }
18
19  }
```

`webapp.WEB-INF.pages.` **list.jsp**

- 展示 查询到的数据
- isELIgnored="false" 开启表达式
- 使用 jstl

```
```

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %>
<%-- 引入 jstl的标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h3>查询了所有的账户信息</h3>

    <c:forEach items="${accounts}" var="account">
        ${account.name}
    </c:forEach>

</body>
</html>
```

加入Spring声明式事务管理

`resources.`**applicationContext.xml**

- 事务管理器
    - DataSourceTransactionManager
- 事务通知
    - 方法名 & 事务
- AOP

```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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置要忽略的注解 -->
        <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>
```

```
<!-- Spring整合MyBatis框架 SqlMapConfig的转移? -->
    <!-- 配置连接池 C3P0 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
        <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/db"/>
        <property name="user" value="root"/>
        <property name="password" value="xxxx"/>
    </bean>

    <!-- 配置SqlSessionFactory工厂对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <!-- 配置AccountDao接口所在的包 映射扫描配置的类 -->
    <bean id="mapperScanner"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.learn.dao"></property>
    </bean>

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

    <!-- 配置事务通知 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <!-- find开头的方法 只读事务 -->
            <tx:method name="find*" read-only="true"/>

            <tx:method name="*" isolation="DEFAULT"></tx:method>
        </tx:attributes>
    </tx:advice>

    <!-- 配置AOP增强 -->
    <aop:config>
        <aop:advisor advice-ref="txAdvice" pointcut="execution(*
com.learn.service.impl.*ServiceImpl.*(..))"></aop:advisor>
    </aop:config>

</beans>
```

`webapp.` index.jsp

试试保存

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
```

```
    <title>Title</title>
</head>
<body>
    <a href="account/findAll">测试SpringMVC 测试查询</a>

    <h3>测试保存</h3>
    <form action="account/saveAccount" method="post">
        姓名:<input type="text" name="name"><br>
        金额:<input type="text" name="money"><br>
        <input type="submit" value="保存">
    </form>

</body>
</html>
```

`controller.` AccountController

- save方法，接收 表单为 对象
- 保存
- 重定向 到 list.jsp

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

    /*
        这里能调用 service 就算 springmvc 和 spring整合成功了
        这里启动方式是 tomcat，web.xml只配置了加载springmvc.xml，别的不扫描

        加载springConfig.xml
     */

    @Autowired
    private AccountService accountService;


    @RequestMapping("/findAll")
    public String findAll(Model model) {
        System.out.println("表现层：查询所有账户信息");
        List<Account> accounts = accountService.findAll();
        model.addAttribute("accounts", accounts);//存到域对象里
        return "list";//跳到list.jsp
    }

    @RequestMapping("/saveAccount")
    public void saveAccount(Account account, HttpServletRequest request,
HttpServletResponse response) throws IOException {
        System.out.println("表现层：保存账户");
        accountService.saveAccount(account);

response.sendRedirect(request.getContextPath()+"/account/findAll");//保存完
就去查
```

```
        return;
    }

}
```

java.com.learn

controller

AccountController

1SpringMVC

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

    @RequestMapping("/findAll")
    public String findAll() {
        System.out.println("表现层：查询所有账户信息");
        return "list";//跳到list.jsp
    }

}
```

2Spring整合SpringMVC

- controller里调用service(IOC管理)，整合成功
- 加入 成员 accountService，注入
- 调用service方法

```
/**
 * web层
 */
@Controller
@RequestMapping("/account")
public class AccountController {
    @Autowired
    private AccountService accountService;

    @RequestMapping("/findAll")
    public String findAll() {
        System.out.println("表现层：查询所有账户信息");
        //调用service的方法
        accountService.findAll();
        return "list";//跳到list.jsp
    }

}
```

3Spring整合MyBatis

- 能查到数据了
- 把数据 展示到 页面上

```
/**
 * web层
 */
@Controller
@RequestMapping("/account")
public class AccountController {
    @Autowired
    private AccountService accountService;

    @RequestMapping("/findAll")
    public String findAll(Model model) {
        System.out.println("表现层：查询所有账户信息");
        //调用service的方法
        List<Account> accounts = accountService.findAll();
        model.addAttribute("accounts", accounts);
        return "list";//跳到list.jsp
    }



}
```

4声明式事务

- save方法，接收 表单为 对象
- 保存
- 重定向 到 list.jsp

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

    /*
        这里能调用 service 就算 springmvc 和 spring整合成功了
        这里启动方式是 tomcat，web.xml只配置了加载springmvc.xml，别的不扫描

        加载springConfig.xml
     */

    @Autowired
    private AccountService accountService;


    @RequestMapping("/findAll")
    public String findAll(Model model) {
        System.out.println("表现层：查询所有账户信息");
        List<Account> accounts = accountService.findAll();
        model.addAttribute("accounts", accounts);//存到域对象里
        return "list";//跳到list.jsp
```

```
    }

    @RequestMapping("/saveAccount")
    public void saveAccount(Account account, HttpServletRequest request,
HttpServletResponse response) throws IOException {
        System.out.println("表现层：保存账户");
        accountService.saveAccount(account);

response.sendRedirect(request.getContextPath()+"/account/findAll");//保存完
就去查
        return;
    }

}
```

dao

AccountDao.java

1MyBatis

```
/**
 * 账户dao接口
 */
public interface AccountDao {

    /**
     * 查询所有账户
     * @return
     */
    @Select("select * from account")
    List<Account> findAll();

    /**
     * 保存账户信息
     * @param account
     */
    @Insert("insert into account (name,money) values (#{name}, #{money})")
    void saveAccount(Account account);

}
```

2Spring整合MyBatis

- 加个注解注册一下@Repository

```
/**
 * 账户dao接口
 */
@Repository//整合配置后才有用
public interface AccountDao {

    /**
     * 查询所有账户
     * @return
```

```
     */
    @Select("select * from account")
    List<Account> findAll();

    /**
     * 保存账户信息
     * @param account
     */
    @Insert("insert into account (name,money) values (#{name}, #{money})")
    void saveAccount(Account account);

}
```

domain

Account.java

```
/**
 * 账户
 */
public class Account implements Serializable {

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

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getMoney() {
        return money;
    }

    public void setMoney(Double money) {
        this.money = money;
    }

    @Override
    public String toString() {
        return "Account{" +
                "id=" + id +
                ", name='" + name + '\'' +
```

```
                ", money=" + money +
                '}';
    }
}
```

service

AccountService.java

```
public interface AccountService {

    /**
     * 查询所有账户
     * @return
     */
    public List<Account> findAll();

    /**
     * 保存账户信息
     * @param account
     */
    public void saveAccount(Account account);

}
```

impl

AccountServiceImpl.java

```
public class AccountServiceImpl implements AccountService {

    @Override
    public List<Account> findAll() {
        System.out.println("业务层：查询所有账户");
        return null;
    }

    @Override
    public void saveAccount(Account account) {
        System.out.println("业务层：保存账户");
    }
}
```

1Spring

- 加注解@Service
- 注册，交给IOC容器进行管理

```
@Service("accountService")
public class AccountServiceImpl implements AccountService {

    @Override
```

```
    public List<Account> findAll() {
        System.out.println("业务层：查询所有账户");
        return null;
    }

    @Override
    public void saveAccount(Account account) {
        System.out.println("业务层：保存账户");
    }
}
```

2Spring整合MyBatis

- service 能 调用 dao 说明 整合成功
- 加入 成员 dao
- 调用dao方法

```
@Service("accountService")
public class AccountServiceImpl implements AccountService {

    @Autowired
    private AccountDao accountDao;

    @Override
    public List<Account> findAll() {
        System.out.println("业务层：查询所有账户");
        return accountDao.findAll();
    }

    @Override
    public void saveAccount(Account account) {
        System.out.println("业务层：保存账户");
        accountDao.saveAccount(account);
    }
}
```

test

TestSpring

```
public class TestSpring {

    @Test
    public void run1() {
        //加载配置文件
        ApplicationContext context = new
ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        //获取对象
        AccountService service = context.getBean("accountService",
AccountService.class);
        //调用方法
        service.findAll();
    }

}
```

TestMyBatis

```
public class TestMyBatis {

    /**
     * 测试查询
     * @throws Exception
     */
    @Test
    public void testFindAll() throws Exception {
        //1.加载配置文件
        InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
        //2.创建工厂
        SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(in);
        //3.创建session
        SqlSession session = factory.openSession();

        //4.获取代理对象
        AccountDao dao = session.getMapper(AccountDao.class);
        //5.查询
        List<Account> accounts = dao.findAll();
        for (Account account : accounts) {
            System.out.println(account);
        }

        //释放资源
        session.close();
        in.close();
    }

    @Test
    public void testSaveAccount() throws Exception {
        //1.加载配置文件
        InputStream in = Resources.getResourceAsStream("SqlMapConfig.xml");
        //2.创建工厂
        SqlSessionFactory factory = new
SqlSessionFactoryBuilder().build(in);
        //3.创建session
```

```
        SqlSession session = factory.openSession();

        //4.获取代理对象
        AccountDao dao = session.getMapper(AccountDao.class);

        Account account = new Account();
        account.setName("史睿生");
        account.setMoney(3000d);

        dao.saveAccount(account);
        session.commit();

        //释放资源
        session.close();
        in.close();

    }

}
```

resources

applicationContext.xml

1.Spring

- 开启注解扫描，Spring只负责 业务层 、持久层。
    - service
    - dao

```
<?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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置 哪些注解 不扫描 -->
```

```
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>
</beans>
```

2Spring整合MyBatis

- 连接池
- 工厂对象SqlSessionFactory
- AccountDao接口所在包

```
<?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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让
SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置要忽略的注解 -->
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

<!-- Spring整合MyBatis框架 SqlMapConfig的转移? -->
    <!-- 配置连接池 C3P0 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
        <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/db"/>
        <property name="user" value="root"/>
        <property name="password" value="xxxx"/>
    </bean>

    <!-- 配置SqlSessionFactory工厂对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <!-- 配置AccountDao接口所在的包 映射扫描配置的类 -->
    <bean id="mapperScanner"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
```

```
        <property name="basePackage" value="com.learn.dao"></property>
    </bean>

</beans>
```

3Spring声明式事务管理

- 事务管理器
- 事务通知
- AOP

```
<?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:aop="http://www.springframework.org/schema/aop"
       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

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"
>
<!-- 开启注解扫描，要扫描的是service和dao层的注解，要忽略web层注解，因为web层让
SpringMVC框架去管理 -->
    <context:component-scan base-package="com.learn">
        <!-- 配置要忽略的注解 -->
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

<!-- Spring整合MyBatis框架 SqlMapConfig的转移? -->
    <!-- 配置连接池 C3P0 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver"/>
        <property name="jdbcUrl" value="jdbc:mysql://localhost:3306/db"/>
        <property name="user" value="root"/>
        <property name="password" value="xxxx"/>
    </bean>

    <!-- 配置SqlSessionFactory工厂对象 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
    </bean>

    <!-- 配置AccountDao接口所在的包 映射扫描配置的类 -->
    <bean id="mapperScanner"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
```

```
        <property name="basePackage" value="com.learn.dao"></property>
    </bean>

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

    <!-- 配置事务通知 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <!-- find开头的方法 只读事务 -->
            <tx:method name="find*" read-only="true"/>

            <tx:method name="*" isolation="DEFAULT"></tx:method>
        </tx:attributes>
    </tx:advice>

    <!-- 配置AOP增强 -->
    <aop:config>
        <aop:advisor advice-ref="txAdvice" pointcut="execution(*
com.learn.service.impl.*ServiceImpl.*(..))"></aop:advisor>
    </aop:config>

</beans>
```

springmvc.xml

1SpringMVC

- 开启注解扫描，SpringMVC负责web层
    - controller
- 配置 视图解析器
- 静态资源 (不过滤)
- 开启SpringMVC注解支持

```
<?xml version="1.0" encoding="UTF-8"?>
<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">

```

```
    <!-- 开启注解扫描,只扫描Controller注解 -->
    <context:component-scan base-package="com.learn">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!-- 配置视图解析器 -->
    <bean id="internalResourceViewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>


    <!-- 设置静态资源 不过滤 -->
    <mvc:resources location="/css/" mapping="/css/**" />
    <mvc:resources location="/images/" mapping="/images/**" />
    <mvc:resources location="/js/" mapping="/js/**" />

    <!-- 开启SpringMVC注解的支持 -->
    <mvc:annotation-driven></mvc:annotation-driven>
</beans>
```

SqlMapConfig.xml

1MyBatis

- 数据源|环境
- 映射 位置 指定

```
<?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>
    <environments default="mysql">
        <environment id="mysql">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.cj.jdbc.Driver"/>
                <property name="url"
value="jdbc:mysql://localhost:3306/db"/>
                <property name="username" value="root"/>
                <property name="password" value="xxxx"/>
            </dataSource>
        </environment>
    </environments>

    <!-- 引入映射配置:xml|注解 -->
    <mappers>
        <!-- <mapper class="com.learn.dao.AccountDao"/> -->
        <!-- 该包下所有的dao接口都可以使用 -->
        <package name="com.learn.dao"/>
    </mappers>
</configuration>
```

log4j.properties

```
# Set root category priority to INFO and its only appender to CONSOLE.
#log4j.rootCategory=INFO, CONSOLE            debug   info   warn error
fatal
log4j.rootCategory=info, CONSOLE, LOGFILE

# Set the enterprise logger category to FATAL and its only appender to
CONSOLE.
log4j.logger.org.apache.axis.enterprise=FATAL, CONSOLE

# CONSOLE is set to be a ConsoleAppender using a PatternLayout.
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ISO8601} %-6r [%15.15t]
%-5p %30.30c %x - %m\n

# LOGFILE is set to be a File appender using a PatternLayout.
log4j.appender.LOGFILE=org.apache.log4j.FileAppender
log4j.appender.LOGFILE.File=d:\axis.log
log4j.appender.LOGFILE.Append=true
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=%d{ISO8601} %-6r [%15.15t]
%-5p %30.30c %x - %m\n


```

webapp

index.jsp

1SpringMVC

做测试用的，Spring有测试类，这个需要jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <a href="account/findAll">测试SpringMVC 测试查询</a>
</body>
</html>
```

2声明式事务

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
```

```
    <title>Title</title>
</head>
<body>
    <a href="account/findAll">测试SpringMVC 测试查询</a>

    <h3>测试保存</h3>
    <form action="account/saveAccount" method="post">
        姓名:<input type="text" name="name"><br>
        金额:<input type="text" name="money"><br>
        <input type="submit" value="保存">
    </form>

</body>
</html>
```

WEB-INF

web.xml

1SpringMVC

- 前端控制器
    - 加载springmvc.xml配置文件
    - 启动服务器，创建servlet
- 编码过滤器

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>


<!-- 配置前端控制器 -->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 加载SpringMVC.xml配置文件 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!-- 启动服务器 创建该Servlet对象 -->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

<!-- 解决中文乱码的过滤器 -->
  <filter>
    <filter-name>characterEncodingFilter</filter-name>
```

```
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <!-- 设置编码集 -->
    <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>
```

2Spring整合SpringMVC

- 配置 监听器 ContextLoaderListener
    - 加载Spring.xml
    - 依赖 pom中 spring-web
- 问题：Listener 默认只加载 WEB-INF下 的文件
    - 复制一份 到WEB-INF下
    - 传参数

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>

<!-- 监听器 加载Spring.xml -->
  <!-- 默认只加载WEB-INF目录下的applicationContext.xml配置文件：1、复制一份过来 2、传参数 -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <!-- 设置配置文件的路径 -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>


<!-- 配置前端控制器 -->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 加载SpringMVC.xml配置文件 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
```

```
    <!-- 启动服务器 创建该Servlet对象 -->
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

<!-- 解决中文乱码的过滤器 -->
  <filter>
    <filter-name>characterEncodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <!-- 设置编码集 -->
    <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>
```

pages

list.jsp

1SpringMVC

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%-- 引入 jstl的标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h3>查询了所有的账户信息</h3>
</body>
</html>
```

2Spring整合MyBatis

- 展示 查询到的数据
- isELIgnored="false" 开启表达式
- 使用 jstl

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
isELIgnored="false" %>
<%-- 引入 jstl的标签库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h3>查询了所有的账户信息</h3>

    <c:forEach items="${accounts}" var="account">
        ${account.name}
    </c:forEach>

</body>
</html>
```

其他知识

- resources目录 等于 java工程的 src目录下，类路径下
- 会 到 WEB-INF/classes/..

EOF