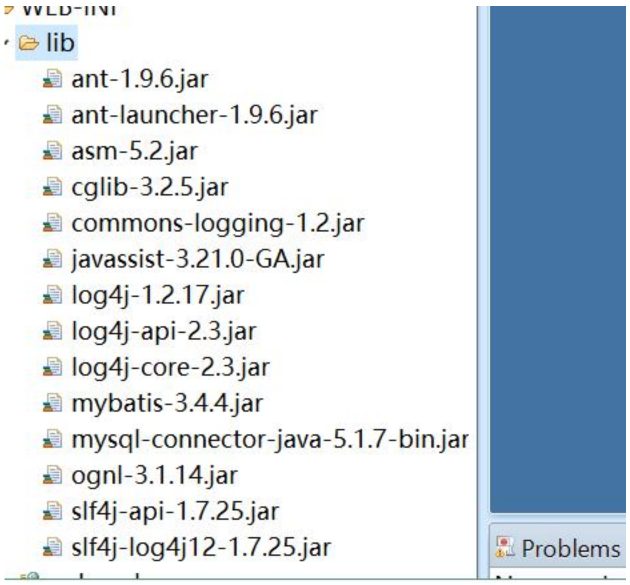
## 本章学习目标

- 单独使用 Mybatis
- MyBatis 整合 Spring - 有 Mapper 实现类
- MyBatis 整合 Spring - 没有 Mapper 实现类
- MyBatis 整合 Spring - Mapper 接口扫描
- MyBatis 整合 Spring - 整合 JDBC 事务
- 整合 SpringMVC

# 1. 单独使用 Mybatis

## 1.1. 导入必须包



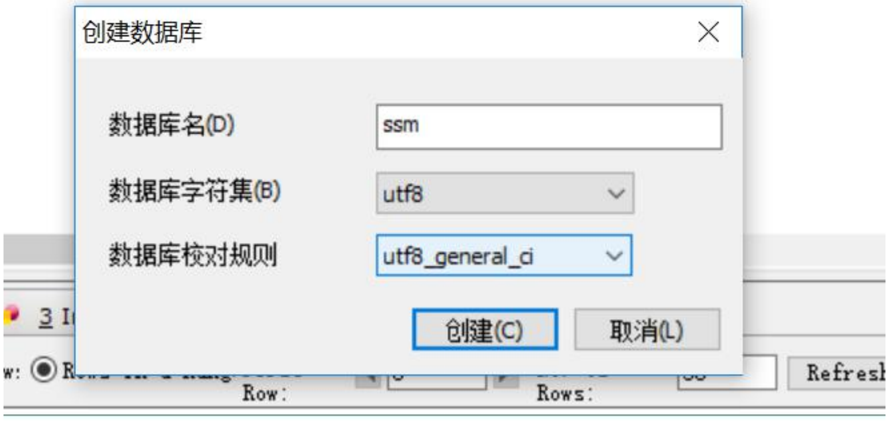
```
lib
  ant-1.9.6.jar
  ant-launcher-1.9.6.jar
  asm-5.2.jar
  cglib-3.2.5.jar
  commons-logging-1.2.jar
  javassist-3.21.0-GA.jar
  log4j-1.2.17.jar
  log4j-api-2.3.jar
  log4j-core-2.3.jar
  mybatis-3.4.4.jar
  mysql-connector-java-5.1.7-bin.jar
  ognl-3.1.14.jar
  slf4j-api-1.7.25.jar
  slf4j-log4j12-1.7.25.jar
```

加入 log4j.properties:

```
### direct log messages to stdout ###

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.err

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n
```

```
### direct messages to file mylog.log ###

log4j.appender.file=org.apache.log4j.FileAppender

log4j.appender.file.File=c\:mylog.log

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n


### set log levels - for more verbose logging change 'info' to 'debug' ###


log4j.rootLogger=debug, stdout
```

## 1.2. 建立数据库和表


创建数据库

数据库名(D) ssm

数据库字符集(B) utf8

数据库校对规则 utf8_general_ci

```
CREATE TABLE t_customer(

	id INT PRIMARY KEY AUTO_INCREMENT,

	NAME VARCHAR(20),

	gender CHAR(1),

	telephone VARCHAR(20),

	address VARCHAR(50)

);
```

## 1.3. 建立实体类

```
package cn.sm1234.domain;


public class Customer {


    private Integer id;

    private String name;

    private String gender;

    private String telephone;

    private String address;

    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 String getGender() {

        return gender;

    }

    public void setGender(String gender) {

        this.gender = gender;

    }
    public String getTelephone() {

        return telephone;

    }

    public void setTelephone(String telephone) {

        this.telephone = telephone;

    }

    public String getAddress() {

        return address;

    }

    public void setAddress(String address) {

        this.address = address;

    }


}
```

## 1.4. 建立 Mapper 接口

```
package cn.sm1234.dao;


import cn.sm1234.domain.Customer;


public interface CustomerMapper {


    /**

     * 添加客户

     */

    public void saveCustomer(Customer customer);

}
```

## 1.5. 建立 sql 映射文件

```
<?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">

<!-- 该文件编写 mybatis 中的 mapper 接口里面的方法提供对应的 sql 语句 -->

<mapper namespace="cn.sm1234.dao.CustomerMapper">


    <!-- 添加客户 -->

    <insert id="saveCustomer" parameterType="cn.sm1234.domain.Customer">

        INSERT INTO ssm.t_customer

            (

            NAME,

            gender,

            telephone,

            address

            )

            VALUES

            (

            #{name},

            #{gender},

            #{telephone},

            #{address}

            )

    </insert>
```

```
</mapper>
```

## 1.6. 建立 sqlMapConfig.xml 文件

```
<?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>

    <!-- 和 spring 整合后 environments 配置将废除 -->

    <environments default="development">

        <environment id="development">

            <!-- 使用 jdbc 事务管理 -->

            <transactionManager type="JDBC" />

            <!-- 数据库连接池 -->

            <dataSource type="POOLED">

                <property name="driver" value="com.mysql.jdbc.Driver" />

                <property name="url"


    value="jdbc:mysql://localhost:3306/ssm?characterEncoding=utf-8" />

                <property name="username" value="root" />

                <property name="password" value="root" />

            </dataSource>

        </environment>

    </environments>


    <!-- 查找 sql 映射文件 -->

    <mappers>

        <mapper resource="mapper/CustomerMapper.xml"/>
    </mappers>

</configuration>
```

## 1.7. 编写测试类

```
package cn.sm1234.test;


import java.io.IOException;

import java.io.InputStream;


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.junit.Test;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;


public class MyBatisTest {


    @Test

    public void test() throws Exception{

        //1.创建 SqlSessionFactoryBuilder

        SqlSessionFactoryBuilder builer = new SqlSessionFactoryBuilder();

        //加载 sqlMapConfig.xml 文件

        InputStream is = Resources.getResourceAsStream("sqlMapConfig.xml");


        //2.创建 sqlSessionFactory
        SqlSessionFactory factory = builer.build(is);


        //3.打开 SqlSession

        SqlSession sqlSession = factory.openSession();


        ///4.获取 Mapper 接口的对象

        CustomerMapper customerMapper = sqlSession.getMapper(CustomerMapper.class);


        //5.操作

        Customer customer = new Customer();

        customer.setName("小张");

        customer.setGender("男");

        customer.setTelephone("020-3333333");

        customer.setAddress("广州天河城广场");


        customerMapper.saveCustomer(customer);


        //6.提交事务

        sqlSession.commit();


        //7.关闭资源

        sqlSession.close();

    }

}
```

# 2. MyBatis 整合 Spring - 有 Mapper 实现类

## 2.1. 导入必须包

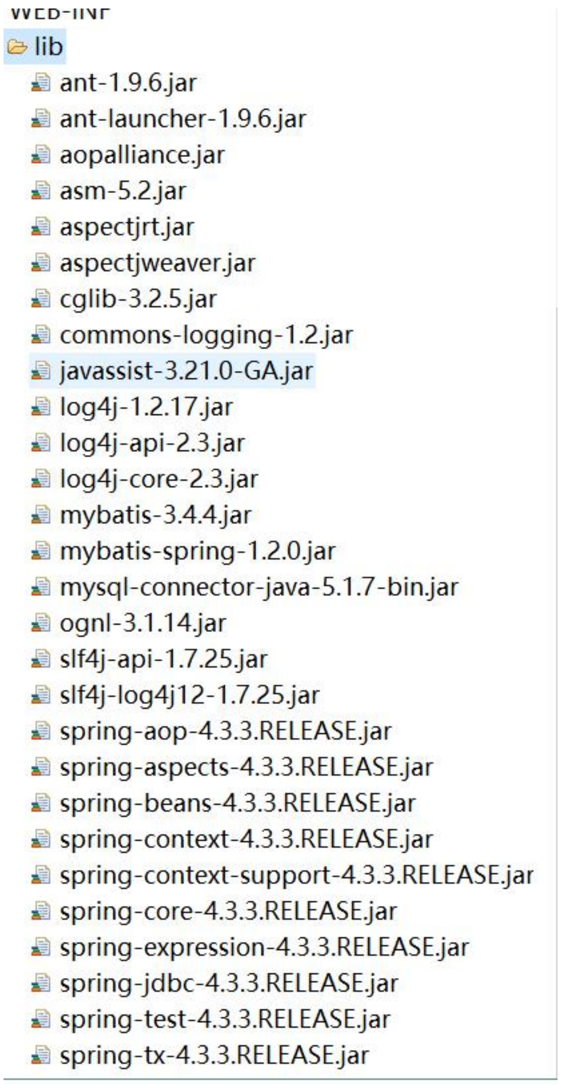
mybatis-spring

spring-ioc

spring-aop

spring-tx

spring-context



WEB-INF
lib
ant-1.9.6.jar
ant-launcher-1.9.6.jar
aopalliance.jar
asm-5.2.jar
aspectjrt.jar
aspectjweaver.jar
cglib-3.2.5.jar
commons-logging-1.2.jar
javassist-3.21.0-GA.jar
log4j-1.2.17.jar
log4j-api-2.3.jar
log4j-core-2.3.jar
mybatis-3.4.4.jar
mybatis-spring-1.2.0.jar
mysql-connector-java-5.1.7-bin.jar
ognl-3.1.14.jar
slf4j-api-1.7.25.jar
slf4j-log4j12-1.7.25.jar
spring-aop-4.3.3.RELEASE.jar
spring-aspects-4.3.3.RELEASE.jar
spring-beans-4.3.3.RELEASE.jar
spring-context-4.3.3.RELEASE.jar
spring-context-support-4.3.3.RELEASE.jar
spring-core-4.3.3.RELEASE.jar
spring-expression-4.3.3.RELEASE.jar
spring-jdbc-4.3.3.RELEASE.jar
spring-test-4.3.3.RELEASE.jar
spring-tx-4.3.3.RELEASE.jar

## 2.2. 编写 Mapper 的实现类

接口:

```
package cn.sm1234.dao;


import cn.sm1234.domain.Customer;


public interface CustomerMapper {


    /**

     * 添加客户

     */

    public void saveCustomer(Customer customer);

}
```

实现:

```
package cn.sm1234.dao.impl;


import org.apache.ibatis.session.SqlSession;

import org.mybatis.spring.support.SqlSessionDaoSupport;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;


public class CustomerMapperImpl extends SqlSessionDaoSupport implements CustomerMapper

{


    public void saveCustomer(Customer customer) {
        SqlSession sqlSession = this.getSqlSession();

        sqlSession.insert("saveCustomer",customer);

    }


}
```

## 2.3. 编写 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: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">


    <!-- 读取 jdbc.properties -->

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


    <!-- 创建 DataSource -->

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
```

```
        <property name="url" value="${jdbc.url}"/>

        <property name="driverClassName" value="${jdbc.driverClass}"/>

        <property name="username" value="${jdbc.user}"/>

        <property name="password" value="${jdbc.password}"/>

        <property name="maxActive" value="10"/>

        <property name="maxIdle" value="5"/>

    </bean>


    <!-- 创建 SqlSessionFactory 对象 -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

        <!-- 关联连接池 -->

        <property name="dataSource" ref="dataSource"/>

        <!-- 加载 sql 映射文件 -->

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

    </bean>


    <!-- 创建 CustomerMapperImpl 对象，注入 SqlSessionFactory -->

    <bean id="customerMapper" class="cn.sm1234.dao.impl.CustomerMapperImpl">

        <!-- 关联 sqlSessionFactory -->

        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>

    </bean>


</beans>
```

## 2.4. 编写测试类

```
package cn.sm1234.test;


import org.junit.Test;
```

```java
import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;



public class MyBatisSpringTest {


    @Test

    public void test(){

        //1.加载 spring 配置

        ApplicationContext ac = new

ClassPathXmlApplicationContext("applicationContext.xml");


        //2.获取对象

        CustomerMapper customerMapper = (CustomerMapper)ac.getBean("customerMapper");


        //3.调用方法

        Customer customer = new Customer();

        customer.setName("小美");

        customer.setGender("女");

        customer.setTelephone("020-666666");

        customer.setAddress("广州体育中心");


        customerMapper.saveCustomer(customer);

    }

}
```

# 3. MyBatis 整合 Spring - 没有 Mapper 实现类

## 3.1. 删除 CustomerMapperImpl 类

## 3.2. 修改 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: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">

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

    <!-- 创建 DataSource -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="url" value="${jdbc.url}"/>
        <property name="driverClassName" value="${jdbc.driverClass}"/>

        <property name="username" value="${jdbc.user}"/>

        <property name="password" value="${jdbc.password}"/>

        <property name="maxActive" value="10"/>

        <property name="maxIdle" value="5"/>

    </bean>


    <!-- 创建 SqlSessionFactory 对象 -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

        <!-- 关联连接池 -->

        <property name="dataSource" ref="dataSource"/>

        <!-- 加载 sql 映射文件 -->

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

    </bean>


    <!-- 配置 Mapper 接口 -->

    <bean id="customerMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">

        <!-- 关联 Mapper 接口 -->

        <property name="mapperInterface" value="cn.sm1234.dao.CustomerMapper"/>

        <!-- 关联 SqlSessionFactory -->

        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>

    </bean>


</beans>
```

## 3.3. 运行测试类

```
package cn.sm1234.test;


import org.junit.Test;
```

```java
import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;



public class MyBatisSpringTest {


    @Test

    public void test(){

        //1.加载 spring 配置

        ApplicationContext ac = new

ClassPathXmlApplicationContext("applicationContext.xml");


        //2.获取对象

        CustomerMapper customerMapper = (CustomerMapper)ac.getBean("customerMapper");


        //3.调用方法

        Customer customer = new Customer();

        customer.setName("老王");

        customer.setGender("男");

        customer.setTelephone("020-888888");

        customer.setAddress("广州体育中心");


        customerMapper.saveCustomer(customer);

    }

}

package cn.sm1234.test;
```

```
import org.junit.Test;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;



public class MyBatisSpringTest {


    @Test

    public void test(){

        //1.加载 spring 配置

        ApplicationContext ac = new

ClassPathXmlApplicationContext("applicationContext.xml");


        //2.获取对象

        CustomerMapper customerMapper = (CustomerMapper)ac.getBean("customerMapper");


        //3.调用方法

        Customer customer = new Customer();

        customer.setName("老王");

        customer.setGender("男");

        customer.setTelephone("020-888888");

        customer.setAddress("广州体育中心");


        customerMapper.saveCustomer(customer);

    }
}
```

## 4. MyBatis 整合 Spring - Mapper 接口扫描（推荐）

### 4.1. 修改 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: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">

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

    <!-- 创建 DataSource -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
```

```
        <property name="url" value="${jdbc.url}"/>

        <property name="driverClassName" value="${jdbc.driverClass}"/>

        <property name="username" value="${jdbc.user}"/>

        <property name="password" value="${jdbc.password}"/>

        <property name="maxActive" value="10"/>

        <property name="maxIdle" value="5"/>

    </bean>


    <!-- 创建 SqlSessionFactory 对象 -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

        <!-- 关联连接池 -->

        <property name="dataSource" ref="dataSource"/>

        <!-- 加载 sql 映射文件 -->

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

    </bean>


    <!-- Mapper 接口的扫描 -->

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

        <!-- 配置 Mapper 接口所在包路径  -->

        <property name="basePackage" value="cn.sm1234.dao"/>

    </bean>


</beans>
```

# 5. MyBatis 整合 Spring - 整合 JDBC 事务

## 5.1. 修改 applicationContext.xml

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

```xml
<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">


    <!-- 读取 jdbc.properties -->

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


    <!-- 创建 DataSource -->

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">

        <property name="url" value="${jdbc.url}"/>

        <property name="driverClassName" value="${jdbc.driverClass}"/>

        <property name="username" value="${jdbc.user}"/>

        <property name="password" value="${jdbc.password}"/>

        <property name="maxActive" value="10"/>

        <property name="maxIdle" value="5"/>

    </bean>


    <!-- 创建 SqlSessionFactory 对象 -->

    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
```

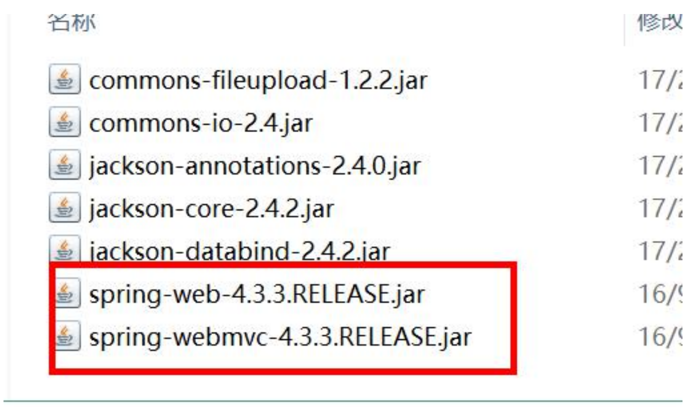
```
        <!-- 关联连接池 -->

        <property name="dataSource" ref="dataSource"/>

        <!-- 加载 sql 映射文件 -->

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

    </bean>


    <!-- Mapper 接口的扫描 -->

    <!--

        注意：如果使用 Mapper 接口包扫描，那么每个 Mapper 接口在 Spring 容器中的 id 名称为类
名： 例如 CustomerMapper -> customerMapper

     -->

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

        <!-- 配置 Mapper 接口所在包路径  -->

        <property name="basePackage" value="cn.sm1234.dao"/>

    </bean>


    <!-- 开启 Spring 的 IOC 注解扫描 -->

    <context:component-scan base-package="cn.sm1234"/>


    <!-- 开启 Spring 的事务 -->

    <!-- -事务管理器 -->

    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

        <property name="dataSource" ref="dataSource"/>

    </bean>

    <!-- 启用 Spring 事务注解 -->

    <tx:annotation-driven transaction-manager="transactionManager"/>


</beans>
```

## 5.2. 在业务方法添加注解

```
package cn.sm1234.service.impl;


import javax.annotation.Resource;


import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;


import cn.sm1234.dao.CustomerMapper;

import cn.sm1234.domain.Customer;

import cn.sm1234.service.CustomerService;


@Service("customerService")

@Transactional

public class CustomerServiceImpl implements CustomerService {

    //注入 Mapper 对象

    @Resource

    private CustomerMapper customerMapper;


    public void saveCustomer(Customer customer) {

        customerMapper.saveCustomer(customer);

        //模拟异常

        int i = 100/0;

        customerMapper.saveCustomer(customer);

    }


}
```

# 6. 整合 SpringMVC

## 6.1. 导入 spring-mvc 包


名称 修改
commons-fileupload-1.2.2.jar 17/
commons-io-2.4.jar 17/
jackson-annotations-2.4.0.jar 17/
jackson-core-2.4.2.jar 17/
jackson-databind-2.4.2.jar 17/
spring-web-4.3.3.RELEASE.jar 16/
spring-webmvc-4.3.3.RELEASE.jar 16/

## 6.2. 配置 web.xml

1）启动 spring，加载 applicationContext.xml

2）启动 springMVC，加载 spring-mvc.xml

```
<!-- 启动 SpringMVC -->
    <servlet>
        <servlet-name>DispatcherServlet</servlet-name>

    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class
>
        <!-- 参数：读取 spring-mvc.xml -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring-mvc.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>DispatcherServlet</servlet-name>
        <url-pattern>*.action</url-pattern>
    </servlet-mapping>



    <!-- 启动 spring -->
    <listener>

    <listener-class>org.springframework.web.context.ContextLoaderListener</listener
-class>
    </listener>
    <!-- 修改路径 -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>
```

## 6.3. 配置 spring-mvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:contenxt="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">


    <!-- 扫描 Controller 所在的包 -->
    <contenxt:component-scan base-package="cn.sm1234.controller"/>


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


    <!-- 视图解析器:简化在 Controller 类编写的视图路径 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!-- 前缀 -->
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <!-- 后缀 -->
        <property name="suffix" value=".jsp"/>
    </bean>


</beans>
```

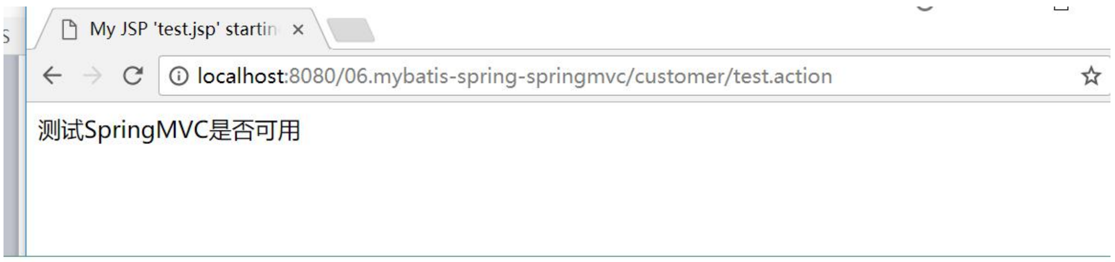
## 6.4. 编写 Controller

```
package cn.sm1234.controller;


import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;


@Controller
@RequestMapping("/customer")
public class CustomerController {
    @RequestMapping("/test")
    public String test(){
        return "test";
    }
}
```

## 6.5. 编写页面

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'test.jsp' starting page</title>

     <meta http-equiv="pragma" content="no-cache">
     <meta http-equiv="cache-control" content="no-cache">
     <meta http-equiv="expires" content="0">
     <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
     <meta http-equiv="description" content="This is my page">
     <!--
     <link rel="stylesheet" type="text/css" href="styles.css">
     -->

  </head>

  <body>
```

```
    测试 SpringMVC 是否可用

  </body>

</html>
```



My JSP 'test.jsp' startin
localhost:8080/06.mybatis-spring-springmvc/customer/test.action

测试SpringMVC是否可用

# 7. SSM 整合-客户添加

## 7.1. 在 CustomerController 里面添加方法

```
package cn.sm1234.controller;


import javax.annotation.Resource;


import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;


import cn.sm1234.domain.Customer;

import cn.sm1234.service.CustomerService;


@Controller

@RequestMapping("/customer")

public class CustomerController {


    //注入业务对象
    @Resource
    private CustomerService customerService;
    
    /*@RequestMapping("/test")
    public String test(){
        return "test";
    }*/
    
    /**
     * 跳转到 input.jsp
     */
    @RequestMapping("/input")
    public String input(){
        return "input";
    }
    
    /**
     *保存客户
     */
    @RequestMapping("/save")
    public String save(Customer customer){
        customerService.saveCustomer(customer);
        return "succ";
    }
    
    
}
```

## 7.2. 编写 input.jsp 录入客户页面

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>

  <head>

    <title>客户录入页面</title>


    <meta http-equiv="pragma" content="no-cache">

    <meta http-equiv="cache-control" content="no-cache">

    <meta http-equiv="expires" content="0">

    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">

    <meta http-equiv="description" content="This is my page">

    <!--

    <link rel="stylesheet" type="text/css" href="styles.css">

    -->


  </head>


  <body>

  <form action="${pageContext.request.contextPath}/customer/save.action"

method="post">

    客户姓名：<input type="text" name="name"/><br/>

    客户性别：

    <input type="radio" name="gender" value="男"/>男

    <input type="radio" name="gender" value="女"/>女

    <br/>

    客户手机：<input type="text" name="telephone"/><br/>
    客户住址：<input type="text" name="address"/><br/>

    <input type="submit" value="保存">

  </form>


  </body>

</html>
```

这时发现页面传参到 Controller，中文数据乱码，这时可以在 web.xml 加多编码过滤器：

```
    <!-- 配置 SpringMVC 编码过滤器 -->

    <filter>

        <filter-name>CharacterEncodingFilter</filter-name>


    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-cl
ass>

        <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>
```