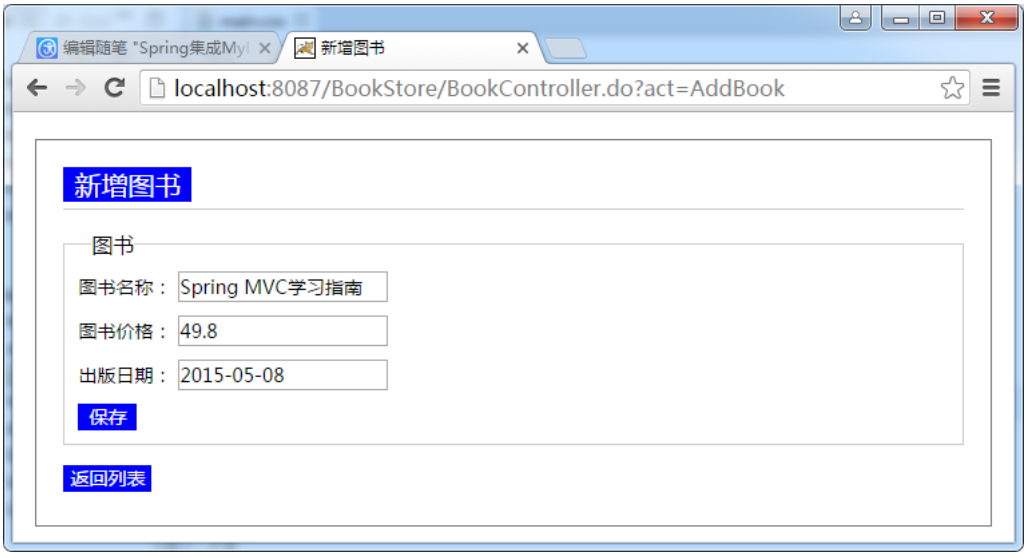
笔记

Spring学习总结（六）——Spring整合MyBatis完整示例

目录

为了梳理前面学习的内容《Spring整合MyBatis（Maven+MySQL）一》与《Spring整合MyBatis（Maven+MySQL）二》，做一个完整的示例完成一个简单的图书管理功能，主要使用到的技术包含Spring、MyBatis、Maven、MySQL及简单MVC等。最后的运行效果如下所示：

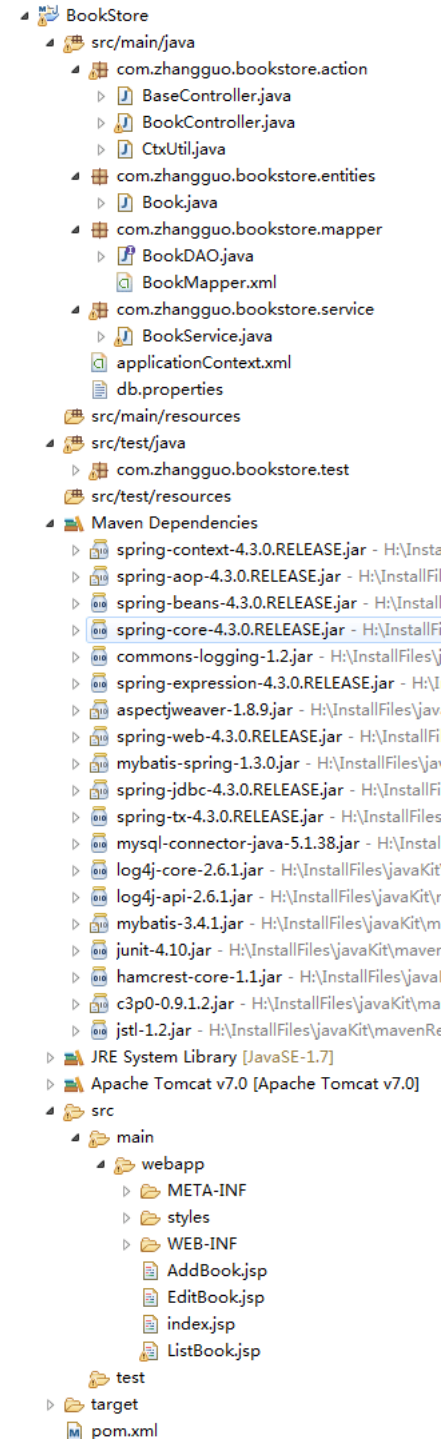


项目结构如下：

MyBatis(4)
MySQL(4)
更多

随笔分类(139)
Android(1)
C#(8)
IT(5)
jQuery(1)
Spring(12)
Spring MVC(7)
第二学期(17)
服务器端Web开发与设计
工具(2)
面试题(2)
前端(37)
数据库(9)
项目(3)
一学期(21)

随笔档案(126)
2017年7月 (2)
2017年6月 (4)
2017年5月 (9)
2017年4月 (5)
2017年3月 (7)
2017年2月 (1)
2017年1月 (4)



一、新建一个基于Maven的Web项目

1.1、创建一个简单的Maven项目，项目信息如下：

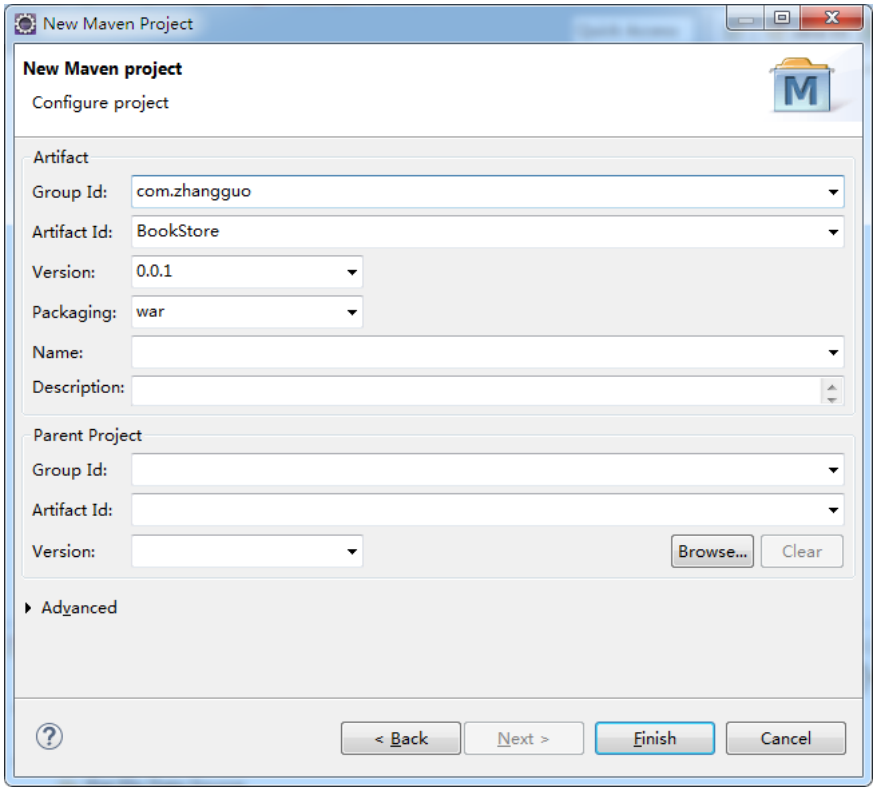
2016年12月 (19)
2016年11月 (11)
2016年10月 (11)
2016年9月 (9)
2016年8月 (7)
2016年7月 (14)
2014年2月 (1)
2014年1月 (1)
2013年5月 (1)
2012年9月 (1)
2011年10月 (3)
2011年9月 (8)
2011年8月 (4)
2011年7月 (4)

相册(2)
打赏(2)
文章中的图片

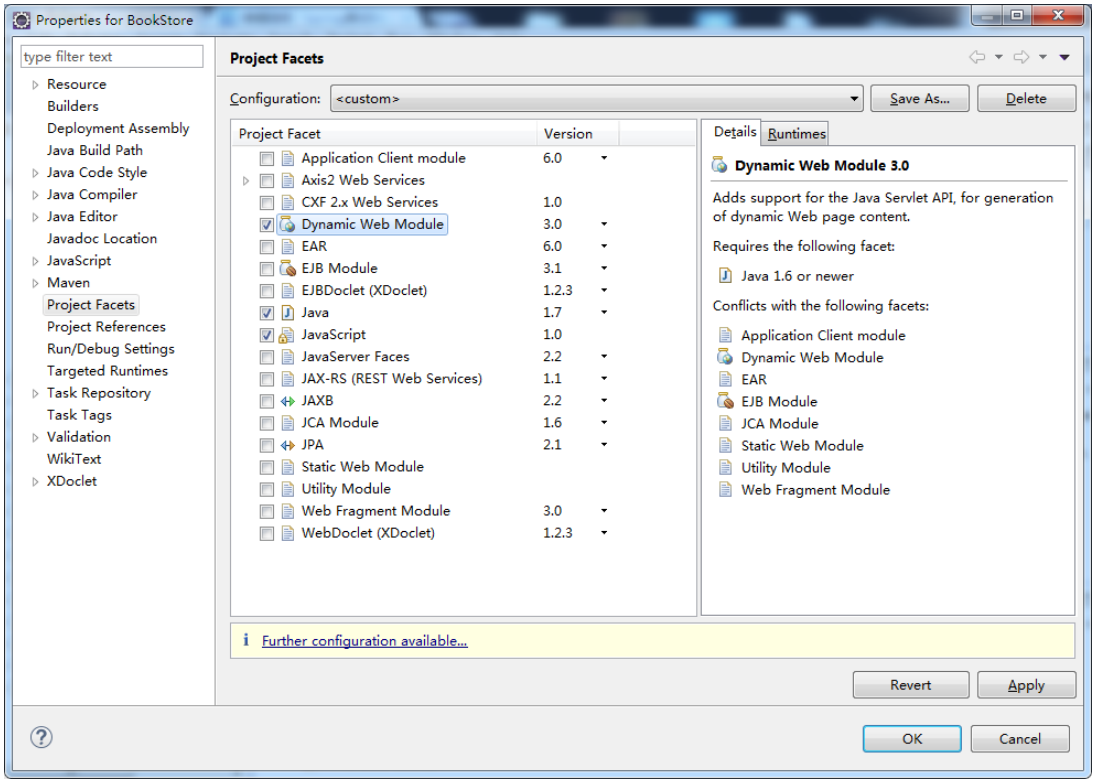
友情链接
南方IT学院

积分与排名
积分 - 208002
排名 - 996

最新评论
1. Re:一种绝对提高开发
张果老师，能否将电影利
我邮箱。ronghh1989@
麻烦了，万分感谢



1.2、修改层面信息，在项目上右键选择属性，再选择“Project Facets”，先设置java运行环境为1.7，先去掉"Dynamic Web Module"前的勾，然后保存关闭；再打开勾选上"Dynamic Web Module"，版本选择“3.0”；这里在左下解会出现一个超链接，创建“Web Content”，完成关闭。



2. Re:Spring MVC 学习
pring+Spring MVC+M
卧槽。受益匪浅啊！

3. Re:30分钟搞定后台登
后台PSD源文件、素材网
@艾题免得已发送...

4. Re:30分钟搞定后台登
后台PSD源文件、素材网
848001189@qq.com
份。谢谢楼主

5. Re:一种绝对提高开发
@黑米面包派@baobac
爷爷@永远的小鄂@De
g发送到了你们指定的邮

阅读排行榜

1. 4种解决json日期格式
3)

2. WebSocket与消息推

3. 后台管理UI的选择(29

4. 弹出层之2：JQuery.l

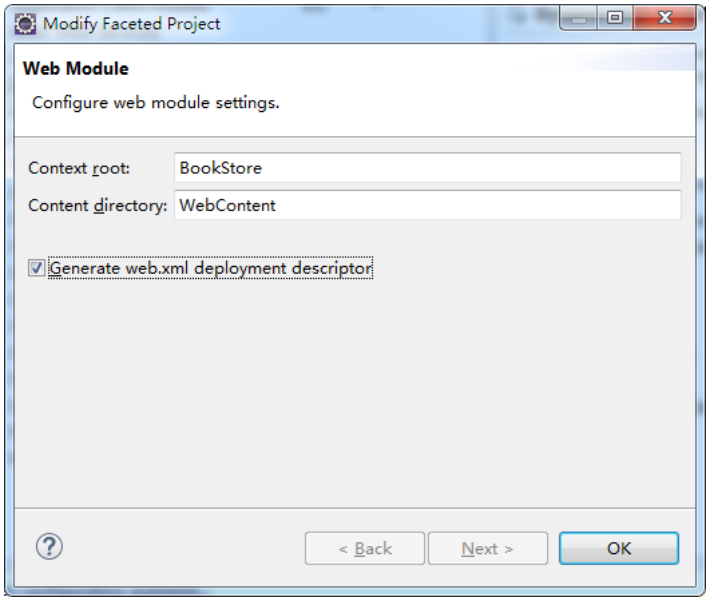
5. 弹出层之1：JQuery.l

6. 一种绝对提高开发水平

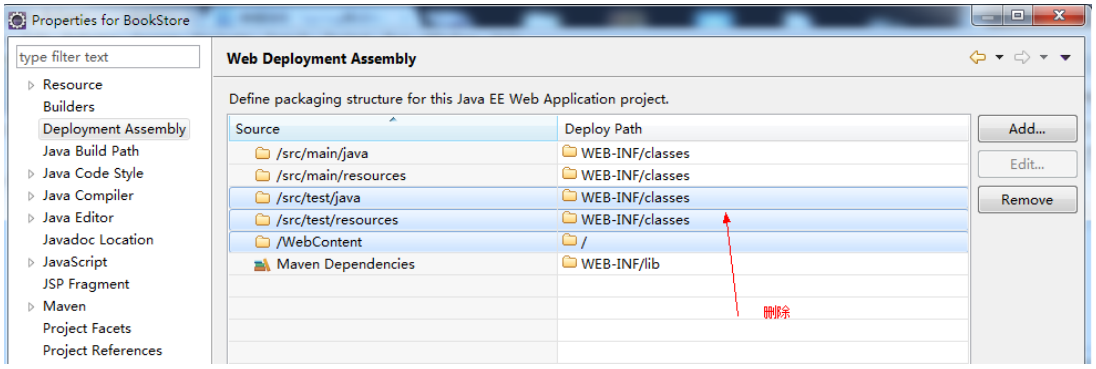
7. Node.js开发Web后台

8. DotNet 资源大全中文
最新版）(12461)

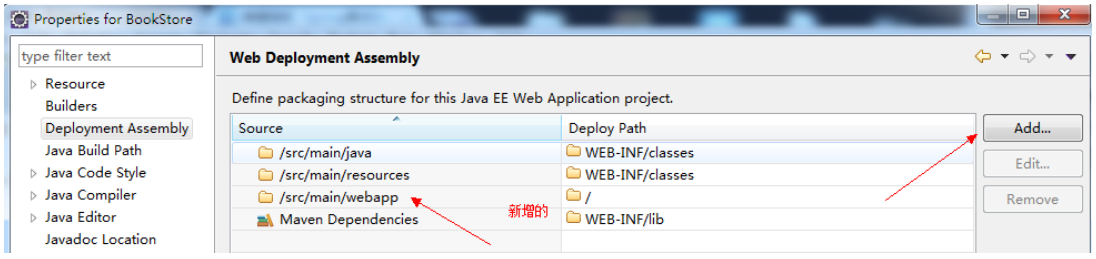
9. Spring学习总结（五
合MyBatis（Maven+M
9)



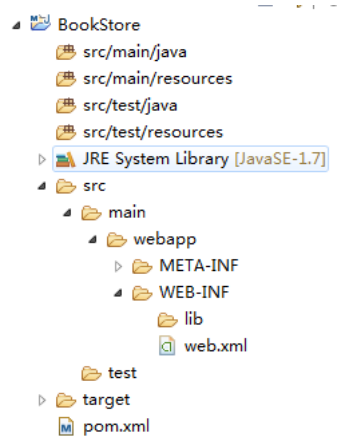
1.3、修改项目的部署内容。项目上右键属性，选择“Deplyment Assembly”,删除不需要发布的内容如：带“test”的两个目录，WebContent目录，再添加一个main下的webapp目录。



修改后的结果如下所示：



1.4、修改项目内容。将WebContent下的内容复制到/src/main/webapp下，再删除WebContent目录，修改后的结果如下所示：



1.5、添加“服务器运行时（Server Runtime）”，添加后的结果如下：

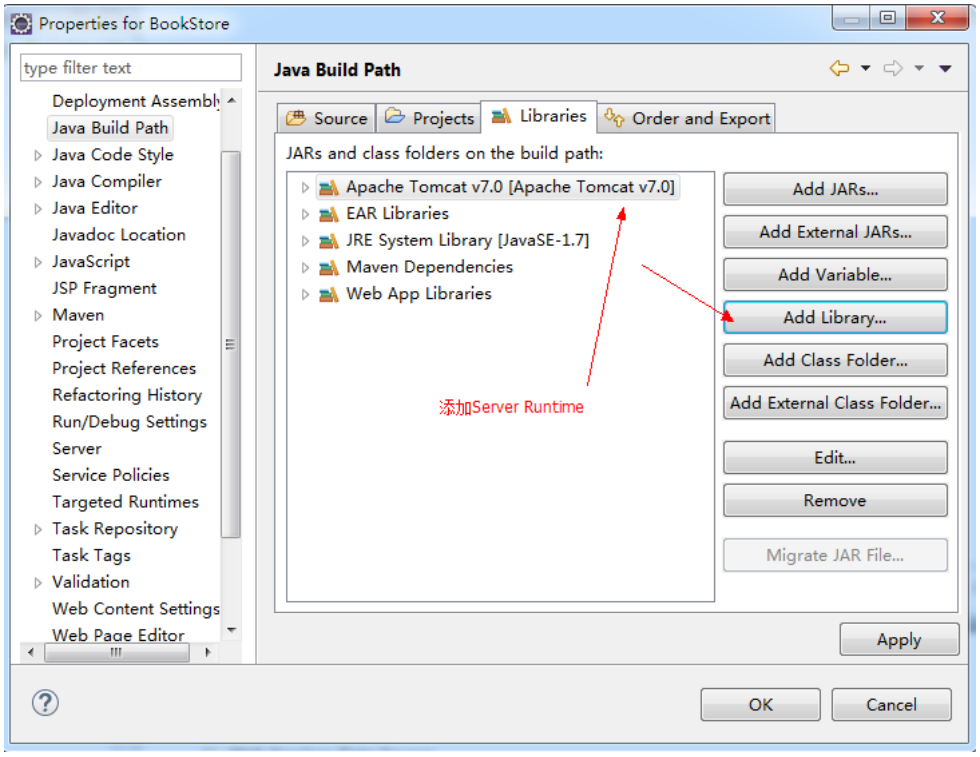
10. Spring MVC 学习总
ring+Spring MVC+My
054)

评论排行榜

1. 30分钟搞定后台登录
PSD源文件、素材网站)
2. 一种绝对提高开发水
3. DotNet 资源大全中文
最新版) (211)
4. 后台管理UI的选择(18
5. 你不知道网络安全有多
6. 一个小时学会MySQL
7. JavaScript资源大全中
e最新版) (74)
8. 从一个故事开始谈项目
9. Java资源大全中文版
版) (40)
10. Spring集成MyBatis

推荐排行榜

1. DotNet 资源大全中文
最新版) (287)
2. 一个小时学会MySQL
3. 一种绝对提高开发水
4. 30分钟搞定后台登录
PSD源文件、素材网站)
5. 后台管理UI的选择(14
6. JavaScript资源大全中
e最新版) (134)
7. Java资源大全中文版
版) (72)
8. 从一个故事开始谈项目



二、创建数据库与表

启动MySQL，创建数据库，新建表books，插入测试数据，完成后的表如下所示：

创建表的sql脚本如下：

```
/*
Navicat MySQL Data Transfer

Source Server         : localhost
Source Server Version : 50536
Source Host           : localhost:3306
Source Database       : db1

Target Server Type    : MYSQL
Target Server Version : 50536
File Encoding         : 65001

Date: 2016-07-06 22:05:07
*/

SET FOREIGN_KEY_CHECKS=0;
```

```
-- ----------------------------
-- Table structure for `books`
-- ----------------------------
DROP TABLE IF EXISTS `books`;
CREATE TABLE `books` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '编号',
  `title` varchar(100) NOT NULL COMMENT '书名',
  `price` decimal(10,2) DEFAULT NULL COMMENT '价格',
  `publishDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '出版日期',
  PRIMARY KEY (`id`),
  UNIQUE KEY `title` (`title`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of books
-- ----------------------------
INSERT INTO `books` VALUES ('1', 'Java编程思想', '98.50', '2005-01-02 00:00:00');
INSERT INTO `books` VALUES ('2', 'HeadFirst设计模式', '55.70', '2010-11-09 00:00:00');
INSERT INTO `books` VALUES ('3', '第一行Android代码', '69.90', '2015-06-23 00:00:00');
INSERT INTO `books` VALUES ('4', 'C++编程思想', '88.50', '2004-01-09 00:00:00');
INSERT INTO `books` VALUES ('5', 'HeadFirst Java', '55.70', '2013-12-17 00:00:00');
INSERT INTO `books` VALUES ('6', '疯狂Android', '19.50', '2014-07-31 00:00:00');
```

需特别注意的是书名是唯一键。

三、添加依赖包

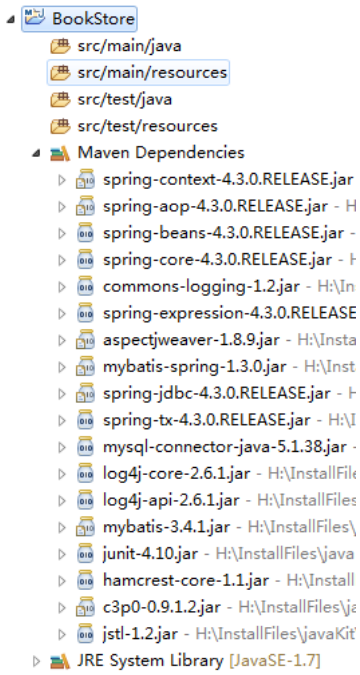
项目主要依赖的jar包有Spring核心包、Spring AOP包、MyBatis ORM包、MyBatis-Spring适配包、JSTL、JUnit、Log4j2等，具体的pom.xml文件如下：

```
<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.zhangguo</groupId>
  <artifactId>BookStore</artifactId>
  <version>0.0.1</version>
  <packaging>war</packaging>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <spring.version>4.3.0.RELEASE</spring.version>
  </properties>

  <dependencies>
    <!--Spring框架核心库 -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!-- aspectJ AOP 织入器 -->
    <dependency>
      <groupId>org.aspectj</groupId>
```

```
    <artifactId>aspectjweaver</artifactId>
    <version>1.8.9</version>
</dependency>
<!-- Spring Web -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
</dependency>
<!--mybatis-spring适配器 -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.0</version>
</dependency>
<!--Spring java数据库访问包，在本例中主要用于提供数据源 -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${spring.version}</version>
</dependency>
<!--mysql数据库驱动 -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.38</version>
</dependency>
<!--log4j日志包 -->
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.6.1</version>
</dependency>
<!-- mybatis ORM框架 -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.1</version>
</dependency>
<!-- JUnit单元测试工具 -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.10</version>
</dependency>
<!--c3p0 连接池 -->
<dependency>
    <groupId>c3p0</groupId>
    <artifactId>c3p0</artifactId>
    <version>0.9.1.2</version>
</dependency>
<!-- jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
```

```
    </dependency>
  </dependencies>
</project>
```

如果是第一次依赖相关的包，则需要下载时间，请耐心等待，如果下载失败请手动下载后复制到本地的资源库中。依赖后的项目结果如下：



四、新建POJO实体层

为了实现与数据库中的books表进行关系映射新建一个Book类，具体代码如下：

```
package com.zhangguo.bookstore.entities;

import java.util.Date;

/**
 * 图书实体
 */
public class Book {
    /**
     * 编号
     */
    private int id;
    /**
     * 书名
     */
    private String title;
    /**
     * 价格
     */
    private double price;
    /**
     * 出版日期
     */
    private Date publishDate;
```

```
    public Book(int id, String title, double price, Date publishDate) {
        this.id = id;
        this.title = title;
        this.price = price;
        this.publishDate = publishDate;
    }

    public Book() {
    }

    public int getId() {
        return id;
    }

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

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Date getPublishDate() {
        return publishDate;
    }

    public void setPublishDate(Date publishDate) {
        this.publishDate = publishDate;
    }
}
```

五、新建MyBatis SQL映射层

这个项目中我们采用接口与xml结束的形式完成关系与对象间的映射，在接口中定义一些数据访问的方法，在xml文件中定义实现数据访问需要的sql脚本。图书数据访问映射接口如下：

```
package com.zhangguo.bookstore.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Param;
```

```
import com.zhangguo.bookstore.entities.Book;

/**
 * 图书数据访问接口
 */
public interface BookDAO {
    /**
     * 获得所有图书
     */
    public List<Book> getAllBooks();
    /**
     * 根据图书编号获得图书对象
     */
    public Book getBookById(@Param("id") int id);
    /**
     * 添加图书
     */
    public int add(Book entity);
    /**
     * 根据图书编号删除图书
     */
    public int delete(int id);
    /**
     * 更新图书
     */
    public int update(Book entity);
}
```

为MyBatis ORM创建的映射文件BookMapper.xml（命名尽量都遵循一个规则，便于扫描，这里约定以实体名+Mapper）如下：

```
<?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">
<!--命名空间应该是对应接口的包名+接口名 -->
<mapper namespace="com.zhangguo.bookstore.mapper.BookDAO">
    <!--id应该是接口中的方法，结果类型如没有配置别名则应该使用全名称 -->
    <!--获得所有图书 -->
    <select id="getAllBooks" resultType="Book">
        select id,title,price,publishDate from books
    </select>
    <!--获得图书对象通过编号 -->
    <select id="getBookById" resultType="Book">
        select id,title,price,publishDate from books where id=#{id}
    </select>
    <!-- 增加 -->
    <insert id="add">
        insert into books(title,price,publishDate)
        values(#{title},#{price},#{publishDate})
    </insert>
    <!-- 删除 -->
    <delete id="delete">
        delete from books where id=#{id}
    </delete>
```

```
    <!-- 更新 -->
    <update id="update">
        update books set title=#{title},price=#{price},publishDate=#{publishDate}
        where id=#{id}
    </update>
</mapper>
```

六、完成Spring整合MyBatis配置

6.1、在源代码的根目录下新建 db.properties文件，用于存放数据库连接信息，文件内容如下：

```
#mysql jdbc
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/db1?useUnicode=true&characterEncoding=UTF-8
jdbc.uid=root
jdbc.pwd=root
```

6.2、在源代码的根目录下新建 applicationContext.xml文件，用于整合MyBatis与Spring，该文件是整个项目的控制中心，非常关键，具体的内容如下：

```
<?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:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

    <!--1 引入属性文件，在配置中占位使用 -->
    <context:property-placeholder location="classpath*:db.properties" />

    <!--2 配置C3P0数据源 -->
    <bean id="datasource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-
method="close">
        <!--驱动类名 -->
        <property name="driverClass" value="${jdbc.driver}" />
        <!-- url -->
        <property name="jdbcUrl" value="${jdbc.url}" />
        <!-- 用户名 -->
        <property name="user" value="${jdbc.uid}" />
        <!-- 密码 -->
        <property name="password" value="${jdbc.pwd}" />
        <!-- 当连接池中的连接耗尽的时候c3p0一次同时获取的连接数 -->
        <property name="acquireIncrement" value="5"></property>
        <!-- 初始连接池大小 -->
```

```
        <property name="initialPoolSize" value="10"></property>
        <!-- 连接池中连接最小个数 -->
        <property name="minPoolSize" value="5"></property>
        <!-- 连接池中连接最大个数 -->
        <property name="maxPoolSize" value="20"></property>
    </bean>

    <!--3 会话工厂bean sqlSessionFactoryBean -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <!-- 数据源 -->
        <property name="dataSource" ref="datasource"></property>
        <!-- 别名 -->
        <property name="typeAliasesPackage" value="com.zhangguo.bookstore.entities"></property>
        <!-- sql映射文件路径 -->
        <property name="mapperLocations"
value="classpath*:com/zhangguo/bookstore/mapper/*Mapper.xml"></property>
    </bean>

    <!--4 自动扫描对象关系映射 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!--指定会话工厂，如果当前上下文中只定义了一个则该属性可省去 -->
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"></property>
        <!-- 指定要自动扫描接口的基础包，实现接口 -->
        <property name="basePackage" value="com.zhangguo.bookstore.mapper"></property>
    </bean>

    <!--5 声明式事务管理 -->
    <!--定义事物管理器，由spring管理事务 -->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="datasource"></property>
    </bean>
    <!--支持注解驱动的事务管理，指定事务管理器 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>

    <!--6 容器自动扫描IOC组件  -->
    <context:component-scan base-package="com.zhangguo.bookstore"></context:component-scan>

    <!--7 aspectj支持自动代理实现AOP功能 -->
    <aop:aspectj-autoproxy proxy-target-class="true"></aop:aspectj-autoproxy>
</beans>
```

共有7处配置，第7处配置非必要，另外关于事务管理可以选择AOP拦截式事务管理。

七、创建服务层

创建BookService服务类，完成图书管理业务，有些项目中也叫业务层，这里我们叫服务层，具体实现如下：

```
package com.zhangguo.bookstore.service;

import java.util.List;
import javax.annotation.Resource;

import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;

import com.zhangguo.bookstore.entities.Book;
import com.zhangguo.bookstore.mapper.BookDAO;

@Service
public class BookService{

    @Resource
    BookDAO bookdao;

    public List<Book> getAllBooks() {
        return bookdao.getAllBooks();
    }

    public Book getBookById(int id){
        return bookdao.getBookById(id);
    }

    public int add(Book entity) throws Exception {
        if(entity.getTitle()==null||entity.getTitle().equals("")){
            throw new Exception("书名必须不为空");
        }
        return bookdao.add(entity);
    }

    @Transactional
    public int add(Book entity1,Book entityBak){
        int rows=0;
        rows=bookdao.add(entity1);
        rows=bookdao.add(entityBak);
        return rows;
    }

    public int delete(int id) {
        return bookdao.delete(id);
    }

    /**
     * 多删除
     */
    public int delete(String[] ids){
        int rows=0;
        for (String idStr : ids) {
            int id=Integer.parseInt(idStr);
            rows+=delete(id);
        }
        return rows;
    }

    public int update(Book entity) {
        return bookdao.update(entity);
    }

}
```

服务层不只是一个dao的接力棒，认为他可有可无，其实是因为我们现在的的示例中没有涉及到更多的复杂业务，所以显得比较空，实现开发可能有更多的业务逻辑要在这里处理。另外给bookdao成员变量注解为自动装配，service类注解为IOC组件。

八、JUnit测试服务类

为了确保服务类中的每个方法正确，先使用JUnit进行单元测试，测试代码如下：

```
package com.zhangguo.bookstore.test;

import static org.junit.Assert.*;
import java.util.Date;
import java.util.List;
import org.junit.BeforeClass;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.zhangguo.bookstore.entities.Book;
import com.zhangguo.bookstore.service.BookService;

public class TestBookService {

   static BookService bookservice;

   @BeforeClass
   public static void before(){
      ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
      bookservice=ctx.getBean(BookService.class);
   }

   @Test
   public void testGetAllBooks() {
      List<Book> books=bookservice.getAllBooks();
      assertNotNull(books);
   }

   @Test
   public void testAdd() {
      Book entity=new Book(0, "Hibernate 第七版", 78.1, new Date());
      try {
         assertEquals(1, bookservice.add(entity));
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   @Test
   public void testDeleteInt() {
      assertEquals(1, bookservice.delete(9));
   }

   @Test
   public void testDeleteStringArray() {
      String[] ids={"7","11","12"};
```

```
        assertEquals(3, bookservice.delete(ids));
    }

    @Test
    public void testUpdate() {
        Book entity=new Book(7, "Hibernate 第二版", 79.1, new Date());
        try {
            assertEquals(1, bookservice.update(entity));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Test
    public void testGetBookById()
    {
        assertNotNull(bookservice.getBookById(1));
    }

    @Test
    public void testAddDouble(){
        //因为书名相同，添加第二本会失败，用于测试事务
        Book entity1=new Book(0, "Hibernate 第八版", 78.1, new Date());
        Book entity2=new Book(0, "Hibernate 第八版", 78.1, new Date());
        assertEquals(2, bookservice.add(entity1, entity2));
    }
}
```

所有的测试均通过，有一个想法就是能否测试完成后数据库还原，如删除的数据在测试后不被真正删除。

九、加载Spring容器与获得容器对象

9.1、修改web.xml文件，添加加载Spring容器用的监听器，修改后的结果如下：

```
<?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_3_0.xsd"
    id="WebApp_ID" version="3.0" >
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <listener>
        <description>Spring容器加载监听器</description>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <context-param>
        <description>设置Spring加载时的配置文件位置，默认位置在web-inf/lib下</description>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath*:applicationContext.xml</param-value>
    </context-param>
</web-app>
```

类org.springframework.web.context.ContextLoaderListener处在Spring-web.jar包中，要记得在pom.xml中添加依赖，测试是否加载成功的简单办法是：重新启动tomcat查看控制信息。

9.2、为了方便获得Spring容器实例，定义一个CtxUtil工具类，工具类的代码如下：

```
package com.zhangguo.bookstore.action;

import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.stereotype.Component;

/**
 * Spring容器上下文工具类，用于获取当前的Spring容器
 * 实现了接口ApplicationContextAware且该类被Spring管理
 *则会自动调用setApplicationContext方法获取Spring容器对象
 */
@Component
public class CtxUtil implements ApplicationContextAware {

   public static ApplicationContext ctx;

   @Override
   public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {
      ctx=applicationContext;
   }
   /**
    * 根据类型获得bean
    */
   public static <T> T getBean(Class<T> clazz){
      return ctx.getBean(clazz);
   }
   /**
    * 根据名称名称获得bean
    */
   public static Object getBean(String name){
      return ctx.getBean(name);
   }

}
```

十、简单MVC控制器封装

为了实现一个简单的MVC基础控制器，定义了一个叫BaseController的Servlet，可以让其它的Servlet继承该Servlet获得部分MVC功能，具体代码如下：

```
package com.zhangguo.bookstore.action;

import java.io.IOException;
import java.lang.reflect.*;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet基类
 * 自定义控制器基类
 */
public class BaseController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=utf-8");

        // 获得要执行的方法名
        String act = request.getParameter("act");

        // 如果用户没有提供方法名
        if (act == null || act.equals("")) {
            // 默认方法
            act = "execute";
        }

        // 根据方法名获得方法信息获得方法信息
        Method method;
        try {
            // 在对象中获得类型信息,在类型中获得方法通过方法名，与参数类型
            method = this.getClass().getMethod(act, HttpServletRequest.class, HttpServletResponse.class);
            // 调用方法,在当前对象中调用，传递参数request与response,获得返回结果
            String targetUri = method.invoke(this, request, response) + "";
            // 如果返回的url是以redirect开始，则是重定向
            if (targetUri.startsWith("redirect:")) {
                response.sendRedirect(targetUri.substring(9, targetUri.length()));
            } else {
                // 转发
                request.getRequestDispatcher(targetUri).forward(request, response);
            }
        } catch (Exception e) {
            response.sendError(400, e.getMessage());
            e.printStackTrace();
        }
    }

    public void execute(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.sendError(400, "请使用参数act指定您要访问的方法");
    }
}
```

十一、完成图书管理功能

11.1、定义BookController控制器

该控制器继承BaseController，中间每一个参数为（HttpServletRequest request,HttpServletResponse response）的方法都充当一个Action，代码如下：

```
package com.zhangguo.bookstore.action;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.zhangguo.bookstore.entities.Book;
import com.zhangguo.bookstore.service.BookService;

@WebServlet("/BookController.do")
public class BookController extends BaseController {
    private static final long serialVersionUID = 1L;

    BookService bookservice;

    @Override
    public void init() throws ServletException {
        bookservice = CtxUtil.getBean(BookService.class);
    }

    // 图书列表Action
    public String ListBook(HttpServletRequest request, HttpServletResponse response) {
        request.setAttribute("books", bookservice.getAllBooks());
        return "ListBook.jsp";
    }

    // 删除图书Action
    public String Delete(HttpServletRequest request, HttpServletResponse response) {
        int id = Integer.parseInt(request.getParameter("id"));
        request.setAttribute("message", bookservice.delete(id) > 0 ? "删除成功！" : "删除失败！");
        request.setAttribute("books", bookservice.getAllBooks());
        return "ListBook.jsp";
    }

    // 多删除图书Action
    public String Deletes(HttpServletRequest request, HttpServletResponse response) {
        String[] ids = request.getParameterValues("ids");
        if (ids!=null&&ids.length > 0) {
            request.setAttribute("message", bookservice.delete(ids) > 0 ? "删除成功！" : "删除失败！");
        } else {
            request.setAttribute("message", "请选择删除项！");
        }
        request.setAttribute("books", bookservice.getAllBooks());
        return "ListBook.jsp";
    }

    // 添加图书Action
```

```
public String AddBook(HttpServletRequest request, HttpServletResponse response) {
    return "AddBook.jsp";
}

// 保存添加图书Action
public String AddBookPost(HttpServletRequest request, HttpServletResponse response) {
    try {
        String title = request.getParameter("title");
        double price = Double.parseDouble(request.getParameter("price"));

        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date publishDate = simpleDateFormat.parse(request.getParameter("publishDate"));

        Book entity = new Book(0, title, price, publishDate);
        if (bookservice.add(entity) > 0) {
            request.setAttribute("model", new Book());
            request.setAttribute("message", "添加成功！");
        } else {
            request.setAttribute("model", entity);
            request.setAttribute("message", "添加失败！");
        }
    } catch (Exception exp) {
        request.setAttribute("message", exp.getMessage());
        exp.printStackTrace();
    }
    return "AddBook.jsp";
}

    //编辑图书Action
    public String EditBook(HttpServletRequest request, HttpServletResponse response) {
        int id = Integer.parseInt(request.getParameter("id"));
        Book model=bookservice.getBookById(id);
        request.setAttribute("model", model);
        return "EditBook.jsp";
    }

    // 保存编辑图书Action
    public String EditBookPost(HttpServletRequest request, HttpServletResponse response) {
        try {
            int id=Integer.parseInt(request.getParameter("id"));

            String title = request.getParameter("title");
            double price = Double.parseDouble(request.getParameter("price"));

            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
            Date publishDate = simpleDateFormat.parse(request.getParameter("publishDate"));

            Book entity = new Book(id, title, price, publishDate);
            request.setAttribute("message", bookservice.update(entity) > 0 ? "更新成功！" : "更新失败！");
            request.setAttribute("model", entity);
        } catch (Exception exp) {
            request.setAttribute("message", exp.getMessage());
            exp.printStackTrace();
        }
        return "EditBook.jsp";
    }
```

```
}
```

11.2、图书列表与删除

定义视图ListBook.jsp，用于完成图书管理，实现图书的列表、删除与多删除功能，页面脚本如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link href="styles/main.css" type="text/css" rel="stylesheet" />
<title>图书管理</title>
</head>
<body>
   <div class="main">
      <h2 class="title"><span>图书管理</span></h2>
      <form action="BookController.do?act=Deletes" method="post">
      <table border="1" width="100%" class="tab">
         <tr>
            <th><input type="checkbox" id="chbAll"></th>
            <th>编号</th>
            <th>书名</th>
            <th>价格</th>
            <th>出版日期</th>
            <th>操作</th>
         </tr>
         <c:forEach var="book" items="${books}">
            <tr>
               <th><input type="checkbox" name="ids" value="${book.id}"></th>
               <td>${book.id}</td>
               <td>${book.title}</td>
               <td>${book.price}</td>
               <td><fmt:formatDate value="${book.publishDate}" pattern="yyyy年MM月dd日"/></td>
               <td>
               <a href="BookController.do?act=Delete&id=${book.id}" class="abtn">删除</a>
               <a href="BookController.do?act=EditBook&id=${book.id}" class="abtn">编辑</a>
               </td>
            </tr>
         </c:forEach>
      </table>
      <p style="color: red">${message}</p>
      <p>
         <a href="BookController.do?act=AddBook" class="abtn">添加</a>
         <input type="submit"  value="删除选择项" class="btn"/>
      </p>
   </form>
   </div>
</body>
</html>
```

运行时效果如下图所示：

15 0

打赏

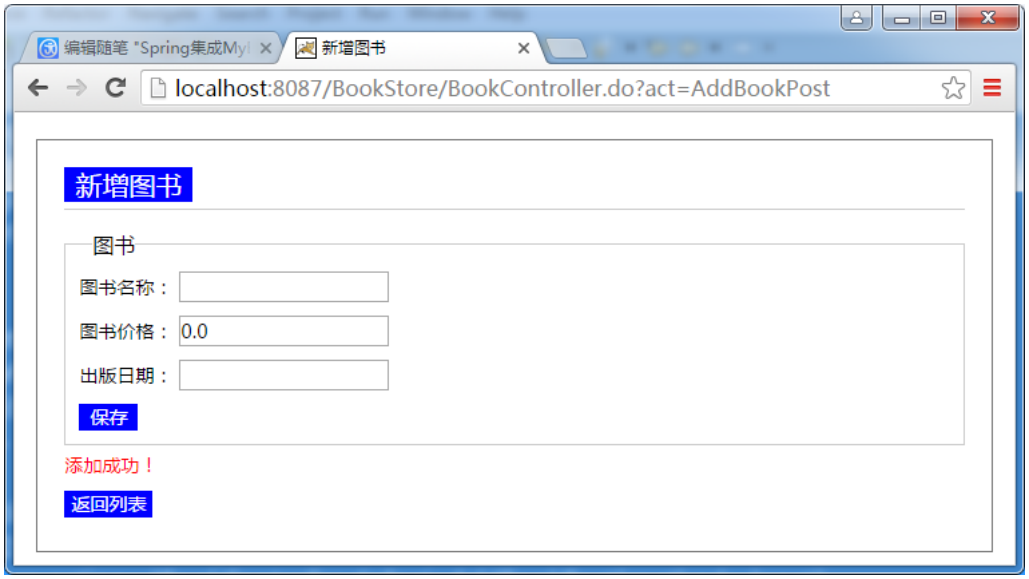
11.3、新增图书功能

定义页面AddBook.jsp完成添加图书功能，在控制器中有两个Action对应新增功能，一个是AddBook，完成页面展示；另一个是AddBookPost处理保存事件，页面脚本如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link href="styles/main.css" type="text/css" rel="stylesheet" />
<title>新增图书</title>
</head>
<body>
    <div class="main">
        <h2 class="title"><span>新增图书</span></h2>
        <form action="BookController.do?act=AddBookPost" method="post">
        <fieldset>
            <legend>图书</legend>
            <p>
                <label for="title">图书名称：</label>
                <input type="text" id="title" name="title"  value="${model.title}"/>
            </p>
            <p>
                <label for="title">图书价格：</label>
                <input type="text" id="price" name="price" value="${model.price}"/>
            </p>
            <p>
                <label for="title">出版日期：</label>
                <input type="text" id="publishDate" name="publishDate"  value="${model.publishDate}"/>
            </p>
```

```
        <p>
          <input type="submit" value="保存" class="btn">
        </p>
      </fieldset>
      </form>
      <p style="color: red">${message}</p>
      <p>
        <a href="BookController.do?act=ListBook" class="abtn">返回列表</a>
      </p>
    </div>
</body>
</html>
```


运行成功时的状态如下：



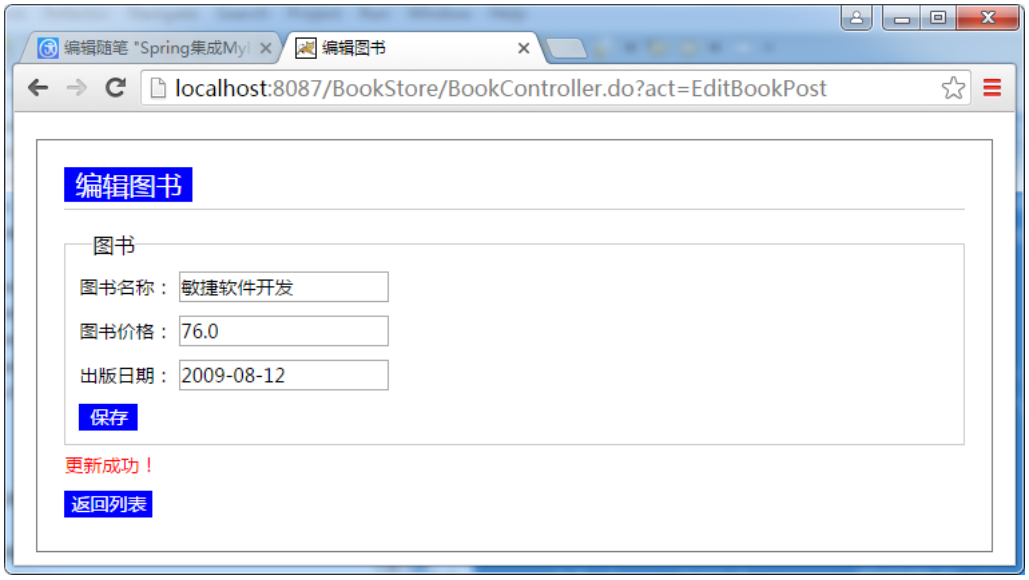
11.4、编辑图书功能

定义页面EditBook.jsp完成更新图书功能，在控制器中有两个Action对应更新功能，一个是EditBook，完成页面展示与加载要编辑图书实体的信息；另一个是EditBookPost处理保存事件，页面脚本如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<link href="styles/main.css" type="text/css" rel="stylesheet" />
<title>编辑图书</title>
</head>
<body>
  <div class="main">
    <h2 class="title"><span>编辑图书</span></h2>
    <form action="BookController.do?act=EditBookPost" method="post">
    <fieldset>
      <legend>图书</legend>
      <p>
        <label for="title">图书名称：</label>
```

```
            <input type="text" id="title" name="title"  value="${model.title}"/>
        </p>
        <p>
            <label for="title">图书价格：</label>
            <input type="text" id="price" name="price" value="${model.price}"/>
        </p>
        <p>
            <label for="title">出版日期：</label>
            <input type="text" id="publishDate" name="publishDate"  value="<fmt:formatDate
value="${model.publishDate}" pattern="yyyy-MM-dd"/>"/>
        </p>
        <p>
          <input type="hidden" id="id" name="id" value="${model.id}"/>
          <input type="submit" value="保存" class="btn">
        </p>
      </fieldset>
      </form>
      <p style="color: red">${message}</p>
      <p>
        <a href="BookController.do?act=ListBook" class="abtn">返回列表</a>
      </p>
    </div>
</body>
</html>
```

运行时的状态如下所示：



11.5、首页与样式

定义index.jsp页面，让其转发到指定的控制器（有点类似路由功能了），页面代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<jsp:forward page="BookController.do?act=ListBook"></jsp:forward>
```

定义了一个简陋的样式main.css，样式表脚本如下：

```
按 Ctrl+C 复制代码
```

```
@CHARSET "UTF-8";

* {
    margin: 0;
    padding: 0;
    font-family: microsoft yahei;
    font-size: 14px;
}

body {
    padding-top: 20px;
}

.main {
    width: 90%;
    margin: 0 auto;
    border: 1px solid #777;
    padding: 20px;
}

.main .title {
    font-size: 20px;
    font-weight: normal;
    border-bottom: 1px solid #ccc;
    margin-bottom: 15px;
```

按 Ctrl+C 复制代码

十二、总结与示例下载

这个Demo起到了承前启后的作用，通过该示例将前面学习过的Spring IOC、MyBatis、JSP、Servlet、Maven及Spring整合MyBatis的内容进行巩固，也为后面学习Spring MVC作好了铺垫。示例中隐约的实现了一些MVC的功能，这远远不够，在URL的处理、表单验证、自动映射表单等方面还可以完善，只想有一个抛砖引玉的作用就满意了，谢谢您的阅读，谢谢！

示例下载

github下载与预览

分类: Spring, 服务器端Web开发与设计

标签: Maven, MyBatis, Spring, MVC

« 上一篇：Spring MVC 学习总结（一）——MVC概要与环境配置
» 下一篇：Spring MVC 学习总结（二）——控制器定义与@RequestMapping详解

posted @ 2016-07-12 08:29 张果 阅读(5667) 评论(7) 编辑 收藏

评论列表

#1楼 2016-07-12 08:45 Vinton.Liu

好

支持(0) 反对(0)

#2楼 2016-07-12 09:16 小饼干

写的不错，加油继续

支持(0) 反对(0)

#3楼 2016-09-30 15:14 tiposhenxiu

好文要顶

支持(1) 反对(0)

15 0

#4楼[楼主] 2016-09-30 15:25 张果

@ tiposhenxiu
谢谢

支持(0) 反对(0)

#5楼 2016-12-09 14:01 huangqili

很好的例子 之前写的模块都通熟易懂

支持(1) 反对(0)

#6楼[楼主] 2016-12-09 14:32 张果

@ huangqili
谢谢！

支持(0) 反对(0)

#7楼 2017-04-25 16:48 青音

挺好的，受用了

支持(0) 反对(0)

打赏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。