1. 回答：在和其它框架整合时，spring框架起的主要作用是什么？

利用IOC、AOP特性整合其他框架实现各种功能

**降低组件之间的耦合度,实现软件各层之间的解耦。**

**可以使用容器提供的众多服务，如：事务管理服务、消息服务等等。当我们使用容器管理事务时，开发人员就不再需要手工控制事务.也不需处理复杂的事务传播。**

**容器提供单例模式支持，开发人员不再需要自己编写实现代码。**

**容器提供了AOP技术，利用它很容易实现如权限拦截、运行期监控等功能。**

**容器提供的众多辅作类，使用这些类能够加快应用的开发，如： JdbcTemplate、 HibernateTemplate。**

**Spring对于主流的应用框架提供了集成支持，如：集成Hibernate、JPA、Struts等，这样更便于应用的开发。**

（2）将springmvc+mybatis实例改造为spring+springmvc+mybatis

实现（参考整合实例中ssm），上传改动的代码。

部署地址<http://oneseek.cn/student-springmvc-maven-0.0.1-SNAPSHOT/>

全部代码<https://gitee.com/Chuangzw/JavaEE_Learn/tree/master/%E5%B7%A5%E7%A8%8B/student-mybatis/student-mybatis>

## Bean.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:aop="http://www.springframework.org/schema/aop"

xmlns:c="http://www.springframework.org/schema/c" xmlns:context="http://www.springframework.org/schema/context"

xmlns:jdbc="http://www.springframework.org/schema/jdbc" xmlns:p="http://www.springframework.org/schema/p"

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/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd

http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

<context:component-scan base-package="service">

</context:component-scan>

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

<!-- Initialization for data source -->

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

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

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

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

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

<!-- 最大连接数量 -->

<property name="maxActive" value="150" />

<!-- 最小空闲连接 -->

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

<!-- 最大空闲连接 -->

<property name="maxIdle" value="20" />

<!-- 初始化连接数量 -->

<property name="initialSize" value="30" />

<!-- 连接被泄露时是否打印 -->

<property name="logAbandoned" value="true" />

<!-- 是否自动回收超时连接 -->

<property name="removeAbandoned" value="true" />

<!-- 超时等待时间(以秒为单位) -->

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

</bean>

<!-- 配置mybatis的sqlSessionFactory -->

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

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

<!-- 自动扫描mappers.xml文件 -->

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

</property>

<!-- mybatis配置文件 -->

<property name="configLocation" value="classpath:mybatis-config.xml">

</property>

</bean>

<!-- 配置事务管理器 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

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

</bean>

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

<!-- DAO接口所在包名，Spring会自动创建各自接口的动态代理类 -->

<property name="basePackage" value="dao" />

<!--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>

</bean>

<!-- 配置事务通知属性 -->

<tx:advice id="txAdvice" transaction-manager="transactionManager">

<!-- 定义事务传播属性 -->

<tx:attributes>

<tx:method name="insert\*" propagation="REQUIRED" />

<tx:method name="tran\*" propagation="REQUIRED" />

<tx:method name="update\*" propagation="REQUIRED" />

<tx:method name="edit\*" propagation="REQUIRED" />

<tx:method name="save\*" propagation="REQUIRED" />

<tx:method name="add\*" propagation="REQUIRED" />

<tx:method name="new\*" propagation="REQUIRED" />

<tx:method name="set\*" propagation="REQUIRED" />

<tx:method name="remove\*" propagation="REQUIRED" />

<tx:method name="delete\*" propagation="REQUIRED" />

<tx:method name="change\*" propagation="REQUIRED" />

<tx:method name="get\*" propagation="REQUIRED" read-only="true" />

<tx:method name="find\*" propagation="REQUIRED" read-only="true" />

<tx:method name="load\*" propagation="REQUIRED" read-only="true" />

<tx:method name="\*" propagation="REQUIRED" read-only="true" />

</tx:attributes>

</tx:advice>

<!-- 配置事务切面 -->

<aop:config>

<aop:pointcut id="serviceOperation"

expression="execution(\* service..\*.\*(..))" />

<aop:advisor advice-ref="txAdvice" pointcut-ref="serviceOperation" />

</aop:config>

</beans>

## Jdbc.properties

jdbc.driverClassName=com.mysql.cj.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/student?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=utf-8&serverTimezone=GMT&useSSL=false

jdbc.username=root

jdbc.password=mysql

## BjController

package controller;

import java.util.List;

import org.apache.ibatis.session.SqlSession;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import dao.BjDao;

import service.BjService;

import vo.Bj;

import javax.annotation.Resource;

@Controller

@RequestMapping("/bj")

public class BjController {

@Resource(name="bjService")

BjService bjService;

@RequestMapping("/list")

public String list(Model model) {

try {

List<Bj> bjList=bjService.queryAll();

model.addAttribute("bjList", bjList);

System.out.println(bjList);

return "bj/list";

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return "error";

}

}

@RequestMapping("/preadd")

public String preadd(){

return "bj/add";

}

@RequestMapping(value="/add",method=RequestMethod.POST)

public String add(Bj bj,Model model){

try {

bjService.add(bj);

model.addAttribute("info", "添加成功");

return "bj/add";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/edit")

public String edit(int id,Model model){

try {

Bj bj=bjService.findById(id);

model.addAttribute("bj", bj);

return "bj/edit";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/update")

public String update(Bj bj,Model model){

try {

bjService.update(bj);

model.addAttribute("info", "更新成功");

return "bj/edit";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/delete/{id}")

public String delete(@PathVariable("id") int id){

try {

bjService.delete(id);

return "redirect:/bj/list.do";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

}

## StudentController

package controller;

import java.io.PrintWriter;

import java.util.List;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletResponse;

import org.apache.ibatis.session.SqlSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import dao.BjDao;

import dao.StudentDao;

import service.BjService;

import service.StudentService;

import vo.Bj;

import vo.Student;

@Controller

@RequestMapping("/student")

public class StudentController {

@Resource(name="studentService")

StudentService studentService;

@Resource(name = "bjService")

BjService bjService;

@RequestMapping("/list")

public String list(Model model) {

try {

List<Student> studentList = studentService.queryAll();

model.addAttribute("studentList",studentList);

return "student/list";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/preadd")

public String preadd(Model model) {

try {

List<Bj> bjList=bjService.queryAll();

model.addAttribute("bjList",bjList);

return "student/add";

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/add")

public String add(Student student,HttpServletResponse response) {

try {

studentService.add(student);

response.setContentType("text/html;charset=UTF-8");

PrintWriter out=response.getWriter();

out.print("添加成功");

return null;

} catch (Exception e) {

e.printStackTrace();

return "error";

}

}

@RequestMapping("/edit")

public String edit(int id,Model model) {

try {

Student student = studentService.findById(id);

List<Bj> bjList=bjService.queryAll();

model.addAttribute("student",student);

model.addAttribute("bjList",bjList);

return "student/edit";

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return "error";

}

}

@RequestMapping("/update")

public String update(Student student,HttpServletResponse response) {

try {

studentService.update(student);

response.setContentType("text/html;charset=UTF-8");

PrintWriter out=response.getWriter();

out.print("更新成功");

return null;

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return "error";

}

}

@RequestMapping("/delete")

public String delete(int id,Model model) {

try {

studentService.delete(id);

List<Student> studentList = studentService.queryAll();

model.addAttribute("studentList", studentList);

return "student/list";

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return "error";

}

}

}

## BjServiceImpl

package service.impl;

import dao.BjDao;

import org.springframework.stereotype.Service;

import service.BjService;

import vo.Bj;

import javax.annotation.Resource;

import java.util.List;

@Service(value = "bjService")

public class BjServiceImpl implements BjService {

@Resource

private BjDao bjDao;

@Override

public List<Bj> queryAll() {

return bjDao.queryAll();

}

@Override

public void add(Bj bj) {

bjDao.add(bj);

}

@Override

public Bj findById(int id) {

return bjDao.findById(id);

}

@Override

public void update(Bj bj) {

bjDao.update(bj);

}

@Override

public void delete(int id) {

bjDao.delete(id);

}

}

## StudentServiceImpl

package service.impl;

import dao.BjDao;

import dao.StudentDao;

import org.springframework.stereotype.Service;

import service.StudentService;

import vo.Student;

import javax.annotation.Resource;

import java.util.List;

@Service(value = "studentService")

public class StudentServiceImpl implements StudentService {

@Resource

StudentDao studentDao;

@Resource

BjDao bjDao;

@Override

public List<Student> queryAll() {

return studentDao.queryAll();

}

@Override

public void add(Student student) {

studentDao.add(student);

}

@Override

public Student findById(int id) {

return studentDao.findById(id);

}

@Override

public void update(Student student) {

studentDao.update(student);

}

@Override

public void delete(int id) {

studentDao.delete(id);

}

}

## BjService

package service;

import vo.Bj;

import java.util.List;

public interface BjService{

public List<Bj> queryAll();

public void add(Bj bj);

public Bj findById(int id);

public void update(Bj bj);

public void delete(int id);

}

## StudentService

package service;

import vo.Bj;

import vo.Student;

import java.util.List;

public interface StudentService {

public List<Student> queryAll();

public void add(Student student);

public Student findById(int id);

public void update(Student student);

public void delete(int id);

}