1.使用所学的Spring部分的知识，将Bean类注入到spring容器中进行管理，同时并将Bean成员变量list的值设置为one,two,three.

package com;

public class Bean {

List list;

//省略set/get方法。。。。。。

}

package com;

public class Test {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");

//在控制台遍历输出Bean类成员变量list的全部值

}

}

**要求：**

1.完成applicationContext.xml（文件头可省略）

2.补全测试类Test

<bean id="bean" class="com.Bean">

<property name="list">

<list>

<value>one</value>

<value>two</value>

<value>three</value>

</list>

</property>

</bean>

Bean bean1 = (Bean)ctx.getBean("bean");

List l = bean1.getList();

for(int i=0;i<l.size();i++){

System.out.println(l.get(i));

}

二、使用所学的Spring部分的知识，将Bean类注入到spring容器中进行管理，同时将Bean成员变量map的值设置为1:one,2:two,3:three.

package cn;

public class Bean {

Map map;

//省略set/get方法。。。。。。

}

package cn;

public class Test {

public static void main(String[] args) {

ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");

//在控制台遍历输出Bean类成员变量map的全部key值和相应的value值

}

}

**要求：**

1.完成applicationContext.xml（文件头可省略）

2.补全测试类Test

<bean id="bean" class="cn.Bean">

<property name="map">

<map>

<entry key="1" value="one"/>

<entry key="2" value="two"/>

<entry key="3" value="three"/>

</map>

</property>

</bean>

Bean bean1 = (Bean)ctx.getBean("bean");

Map m = bean1.getMap();

Iterator it = m.keySet().iterator();

while(it.hasNext()){

Object o = it.next();

System.out.println(o+":"+m.get(o));

}

package test;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import assemble.ComplexUser;

public class TestAssemble {

public static void main(String[] args) {

// TODO Auto-generated method stub

ApplicationContext appCon=new ClassPathXmlApplicationContext("applicationContext.xml");

ComplexUser u1=(ComplexUser)appCon.getBean("user1");

System.out.println(u1);

//集合遍历输出

Set<String> sets=u1.getAliasSet();

Iterator<String> it=sets.iterator();

while(it.hasNext()) {

System.out.println(it.next());

}

//列表输出

List<String> users=u1.getHobbyList();

for(int i=0;i<users.size();i++) {

System.out.println(users.get(i));

}

//Map输出

Map<String, String> map=u1.getResidenceMap();

//得到map集合的key键值的set集合

Set<String> setkeys=map.keySet();

//得到key键值的set集合的迭代器

Iterator<String> itkeys=setkeys.iterator();

while(itkeys.hasNext()) {

String key=itkeys.next();

//通过map集合的get(key)方法获取key值对应的value;

String user=map.get(key);

System.out.println(key+":"+user);

}

String[] array=u1.getArray();

for(int i=0;i<array.length;i++) {

System.out.println(array[i]);

}

ComplexUser u2=(ComplexUser)appCon.getBean("user2");

System.out.println(u2);

}

}

**职工信息录入格式为姓名\_年龄，职工信息录入界面如下：**



**index.jsp**

<form action="save" method="post">

职工信息：<input type="text" name="a"/><br/>

<input type="submit" value="插入"/>

</form>

要求：使用Spring MVC的Converter自定义类型转换器相关知识，编写自定义转换器，将上述信息转换为Staff类型的相应域，Person类型定义如下：

package forms;

public class Staff{

private String name;

private int age;

//省略set/get方法

}

其他相关代码：

**StaffController**   
package controllers;

//省略import

@Controller

@RequestMapping("/user")

public class StaffController {

@RequestMapping("/save")

public String save(@RequestParam("a") Staff s, Model model){

return "stafflist";

}

}

**StaffConverter**

package converters;

//省略import

public class StaffConverter implements Converter<String, Staff>{

public Staff convert(String source) {

Staff s = new Staff();

①解析source并存入Staff对象的相应域中

return s;

}

}

**Springmvc-servlet.xml**

**<!--省略扫描controller包 -->**

**<!--省略配置视图解析器 -->**

②注册类型转换器StaffConverter

**要求：**

1. **补全StaffConverter**
2. **补全 springmvc-servlet.xml**

1. String stringValues[] = source.split(",");

if(stringValues != null && stringValues.length == 2){

s.setName(stringValues[0]);

s.setAge(Integer.parseInt(stringValues[1]));

return s;

}else{

throw new IllegalArgumentException(String.format("类型转换失败，需要格式‘张三,18’，但格式是[%s]", source));

}

}

2.

<bean id="conversionService" class="org.springframework.context.support.ConversionServiceFactoryBean">

<property name="converters">

<list>

<bean class="converters.StaffConverter"/>

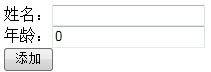
</list>

</property>

</bean>

<mvc:annotation-driven conversion-service="conversionService"/>

**使用JSR 303验证知识实现添加人员的输入校验。验证通过要求姓名不能为空，年龄范围在0到100区间。添加人员界面如下：**



相关代码：

addUser.jsp

<form:form action="addUser" modelAttribute="userForm" method="post">

姓名：<form:input path="name"/><br/>

年龄：<form:input path="age"/><br/>

<input type="submit" value="添加"/><br/>

<form:errors path="\*"/>

</form:form>

UserController

@Controller

@RequestMapping("/User")

public class UserController {

@RequestMapping("/addUser")

public String addUser(@Valid @ModelAttribute UserForm userForm, BindingResult result, Model model) throws Exception {

if(result.hasErrors())

return "addUser";

//省略添加用户代码

return "userList";

}

@RequestMapping("/input")

public String input(Model model){

model.addAttribute("userForm", new UserForm());

return "addUser";

}

}

UserForm

public class UserForm {

①使用注解完成name不为空校验，返回信息的key为userform.name.required

private String name;

②使用注解完成age范围0到100校验，返回信息的key为userform.age.invalid

private int age;

//省略set、get方法

}

springmvc-servlet.xml

<context:component-scan base-package="controller"></context:component-scan>

<bean id="internalResourceViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/"/>

<property name="suffix" value=".jsp" />

</bean>

<bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<property name="basename" value="/WEB-INF/resource/errorMessage"/>

</bean>

**要求：**

③注册校验器、开启Spring的Valid功能

**1、补全**UserForm **（第1、2空）**

**2、补全springmvc-servlet.xml（第3空）**

①@NotBlank(message="{userform.name.required}")

②@Range(max=100,min=0,message="{userform.age.invalid}")

③<bean id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">

<property name="providerClass" value="org.hibernate.validator.HibernateValidator" />

<property name="validationMessageSource" ref="messageSource"/>

</bean>

<mvc:annotation-driven validator="validator"/>