**Spring Form Validation Using Annotation Tutorial**

In this example you will learn the changes you need to make to the previous validation example to make it work with Spring annotations. When using annotated controller class the validate() method will not be automatically called. The onSubmit() method will be called when the form is submitted. Here you need to first call the validate() method of the UserValidator class, then check for any error using the hasErrors() method, if there are any errors then, forward the user to the user registration form and dispaly the errors, if there are no errors then display the success page. Here is the UserController class.

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

04.import org.springframework.stereotype.Controller;

05.import org.springframework.ui.ModelMap;

06.import org.springframework.validation.BindingResult;

07.import org.springframework.web.bind.annotation.ModelAttribute;

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

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

10.import org.springframework.web.bind.annotation.SessionAttributes;

11.

12.import com.tutorials4u.domain.User;

13.import com.tutorials4u.service.UserService;

14.import com.tutorials4u.validator.UserValidator;

15.

16.@Controller

17.@RequestMapping("/userRegistration.htm")

18.@SessionAttributes("user")

19.public class UserController {

20.

21.private UserService userService;

22.private UserValidator userValidator;

23.

24.@Autowired

25.public UserController(UserService userService, UserValidator userValidator) {

26.this.userService = userService;

27.this.userValidator = userValidator;

28.}

29.

30.@RequestMapping(method = RequestMethod.GET)

31.public String showUserForm(ModelMap model) {

32.User user = new User();

33.model.addAttribute("user", user);

34.return "userForm";

35.}

36.

37.@RequestMapping(method = RequestMethod.POST)

38.public String onSubmit(@ModelAttribute("user") User user,

39.BindingResult result) {

40.userValidator.validate(user, result);

41.if (result.hasErrors()) {

42.return "userForm";

43.} else {

44.userService.add(user);

45.return "redirect:userSuccess.htm";

46.}

47.}

48.

49.}

Here the UserService and the UserValidator classes are injected using the @Autowired annotation. The Spring bean configuration file has the following entries.

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

02.<beans xmlns="http://www.springframework.org/schema/beans"

03.xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

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

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

06.xsi:schemaLocation="http://www.springframework.org/schema/beans

07.http://www.springframework.org/schema/beans/spring-beans.xsd

08.http://www.springframework.org/schema/context

09.http://www.springframework.org/schema/context/spring-context.xsd">

10.

11.<bean id="viewResolver" class="org.springframework.web.servlet.view. InternalResourceViewResolver" p:prefix="/WEB-INF/jsp/"p:suffix=".jsp" />

12.

13.<bean id="messageSource" class="org.springframework.context.support. ResourceBundleMessageSource" p:basename="messages" />

14.

15.<context:component-scan base-package="com.tutorials4u.web" />

16.

17.<bean id="userService" class="com.tutorials4u.service.UserServiceImpl" />

18.

19.

20.<bean id="userValidator" class="com.tutorials4u.validator.UserValidator" />

21.

22.</beans>