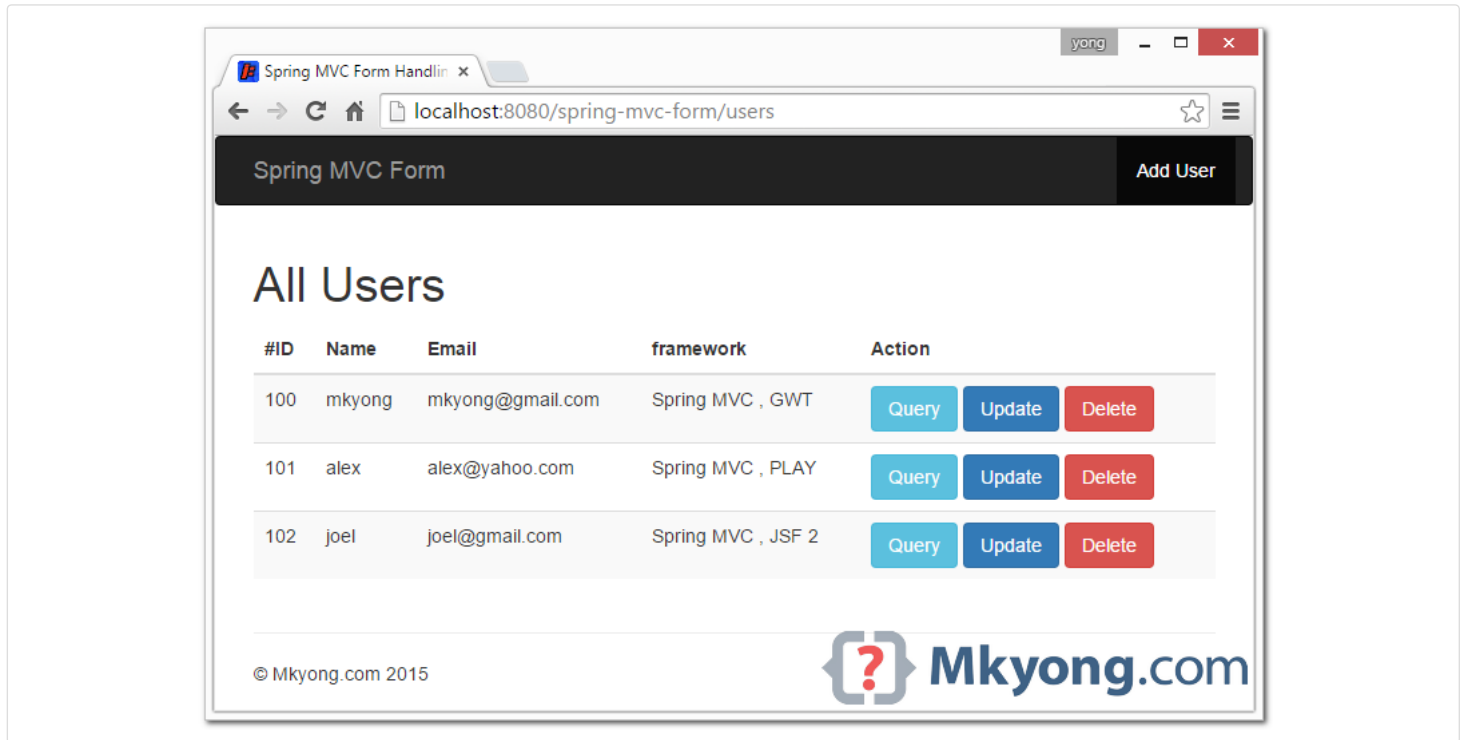


# Spring MVC form handling example

By mkyong (<http://www.mkyong.com/author/mkyong/>) | August 10, 2010 | Updated : July 10, 2015 | Viewed : 605,516 times +2,778 pv/w



In this tutorial, we will show you a Spring MVC form handling project to do the following stuff :

1. Form value binding – JSP and Model.
2. Form validation and display error message.
3. Form POST/REDIRECT/GET pattern, and add messages to flash attribute.
4. CRUD operations, add, get, update and delete with an HTML form.

## Technologies used :

1. Spring 4.1.6.RELEASE
2. Maven 3
3. Bootstrap 3
4. HSQLDB driver 2.3.2
5. Logback 1.1.3
6. JDK 1.7
7. JSTL 1.2
8. Eclipse IDE

## What you'll build :

A simple user management project, you can list, create, update and delete an user, via HTML forms. You'll also see how to perform the form validation and display the error message conditionally. This project is styling with Bootstrap 3, and data are stored in the HSQL embedded database.

## The URI structure :

URI	Method	Action
/users	GET	Listing, display all users
/users	POST	Save or update user
/users/{id}	GET	Display user {id}
/users/add	GET	Display add user form

/users/{id}/update	GET	Display update user form for {id}
/users/{id}/delete	POST	Delete user {id}

**Note**

In the old days, before Spring 3.0, we use `SimpleFormController` (<http://docs.spring.io/spring/docs/3.0.x/api/org/springframework/web/portlet/mvc/SimpleFormController.html>) to do the form handling. As Spring 3.0, this class is deprecated in favor of Spring annotated `@Controller`.

## Spring MVC Form Binding

Before you start the tutorial, you need to understand how the Spring MVC form binding works.

1.1 In controller, you add an object into a model attribute.

```
@RequestMapping(value = "/users/add", method = RequestMethod.GET)
public String showAddUserForm(Model model) {

    User user = new User();
    model.addAttribute("userForm", user);
    //...
}
```

1.2 In HTML form, you use `spring:form` tag and bind the controller object via `modelAttribute`.

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

<form:form method="post" modelAttribute="userForm" action="${userActionUrl}">
    <form:input path="name" type="text" /> <!-- bind to user.name-->
    <form:errors path="name" />
</form:form>
```

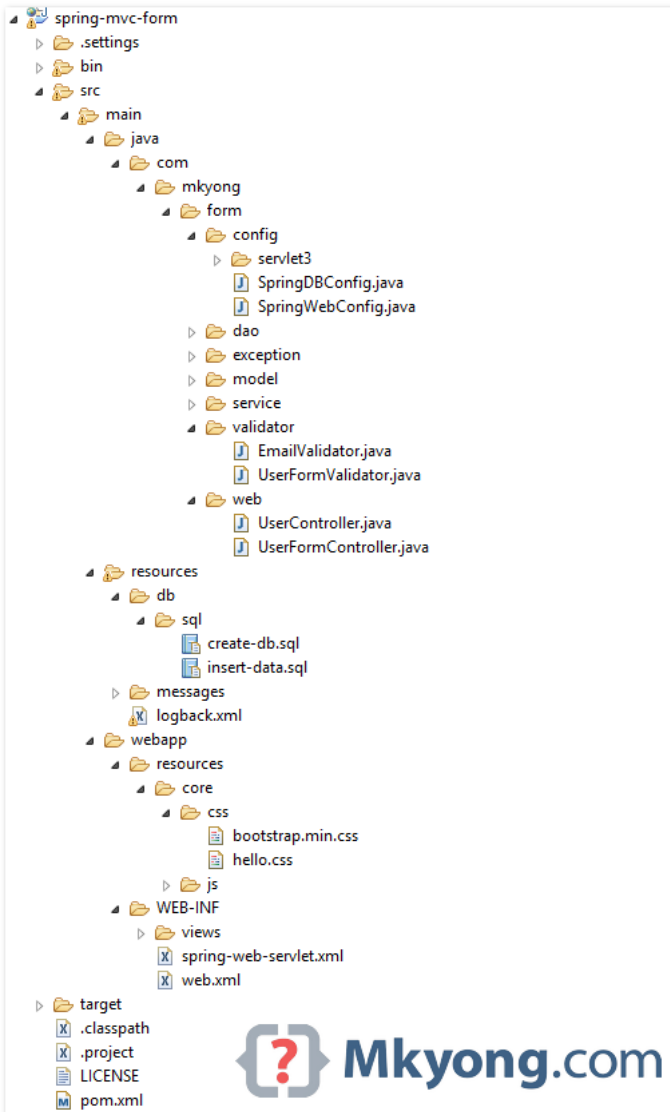
1.3 When the HTML form is "POST", you get the value via `@ModelAttribute`.

```
@RequestMapping(value = "/users", method = RequestMethod.POST)
public String saveOrUpdateUser(@ModelAttribute("userForm") User user,
    BindingResult result, Model model) {
    //...
}
```

Done. Let start the tutorial.

## 1. Project Directory

This is the final project directory structure. A standard Maven project.



(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form->

handling-directory.png)

## 2. Project Dependencies

To develop a Spring MVC project, you need `spring-webmvc`.

`pom.xml`

```
<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mkyong.form</groupId>
  <artifactId>spring-mvc-form</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>SpringMVC + Form Handling Example</name>

  <properties>
    <jdk.version>1.7</jdk.version>
    <spring.version>4.1.6.RELEASE</spring.version>
    <hsqldb.version>2.3.2</hsqldb.version>
    <logback.version>1.1.3</logback.version>
    <jcl.slf4j.version>1.7.12</jcl.slf4j.version>
    <jstl.version>1.2</jstl.version>
    <servletapi.version>3.1.0</servletapi.version>
  </properties>

  <dependencies>

    <!-- Spring Core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
      <exclusions>
        <exclusion>
          <groupId>commons-logging</groupId>
          <artifactId>commons-logging</artifactId>
        </exclusion>
      </exclusions>
    </dependency>

    <!-- Spring Web -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- Spring JDBC -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>

    <!-- HyperSQL DB -->
    <dependency>
      <groupId>org.hsqldb</groupId>
      <artifactId>hsqldb</artifactId>
      <version>${hsqldb.version}</version>
    </dependency>

    <!-- logging -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>jcl-over-slf4j</artifactId>
      <version>${jcl.slf4j.version}</version>
    </dependency>

    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>${logback.version}</version>
    </dependency>

    <!-- jstl -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>${jstl.version}</version>
    </dependency>
```

```

<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>${servletapi.version}</version>
  <scope>provided</scope>
</dependency>

</dependencies>

<build>
  <plugins>

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.3</version>
      <configuration>
        <source>${jdk.version}</source>
        <target>${jdk.version}</target>
      </configuration>
    </plugin>

    <!-- embedded Jetty server, for testing -->
    <plugin>
      <groupId>org.eclipse.jetty</groupId>
      <artifactId>jetty-maven-plugin</artifactId>
      <version>9.2.11.v20150529</version>
      <configuration>
        <scanIntervalSeconds>10</scanIntervalSeconds>
        <webApp>
          <contextPath>/spring-mvc-form</contextPath>
        </webApp>
      </configuration>
    </plugin>

    <!-- configure Eclipse workspace -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-eclipse-plugin</artifactId>
      <version>2.9</version>
      <configuration>
        <downloadSources>true</downloadSources>
        <downloadJavadocs>true</downloadJavadocs>
        <wtpversion>2.0</wtpversion>
        <wtpContextName>spring-mvc-form</wtpContextName>
      </configuration>
    </plugin>

  </plugins>
</build>
</project>

```

### 3. Controller

Spring `@Controller` class to show you how to bind the form value via `@ModelAttribute`, add a form validator, add message into flash attribute, populate values for dropdown and checkboxes and etc. Read the comments for self-explanatory.

```
UserController.java
```

```
package com.mkyong.form.web;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.mkyong.form.model.User;
import com.mkyong.form.service.UserService;
import com.mkyong.form.validator.UserFormValidator;

@Controller
public class UserController {

    private final Logger logger = LoggerFactory.getLogger(UserController.class);

    @Autowired
    private UserService userService;

    @Autowired
    UserFormValidator userFormValidator;

    //Set a form validator
    @InitBinder
    protected void initBinder(WebDataBinder binder) {
        binder.setValidator(userFormValidator);
    }

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String index(Model model) {
        logger.debug("index()");
        return "redirect:/users";
    }

    // list page
    @RequestMapping(value = "/users", method = RequestMethod.GET)
    public String showAllUsers(Model model) {

        logger.debug("showAllUsers()");
        model.addAttribute("users", userService.findAll());
        return "users/list";
    }

    // save or update user
    // 1. @ModelAttribute bind form value
    // 2. @Validated form validator
    // 3. RedirectAttributes for flash value
    @RequestMapping(value = "/users", method = RequestMethod.POST)
    public String saveOrUpdateUser(@ModelAttribute("userForm") @Validated User user,
        BindingResult result, Model model,
        final RedirectAttributes redirectAttributes) {

        logger.debug("saveOrUpdateUser() : {}", user);
    }
}
```

```

    if (result.hasErrors()) {
        populateDefaultModel(model);
        return "users/userform";
    } else {

        // Add message to flash scope
        redirectAttributes.addFlashAttribute("css", "success");
        if(user.isNew()){
            redirectAttributes.addFlashAttribute("msg", "User added successfully!");
        }else{
            redirectAttributes.addFlashAttribute("msg", "User updated successfully!");
        }

        userService.saveOrUpdate(user);

        // POST/REDIRECT/GET
        return "redirect:/users/" + user.getId();

        // POST/FORWARD/GET
        // return "user/list";

    }

}

// show add user form
@RequestMapping(value = "/users/add", method = RequestMethod.GET)
public String showAddUserForm(Model model) {

    logger.debug("showAddUserForm()");

    User user = new User();

    // set default value
    user.setName("mkyong123");
    user.setEmail("test@gmail.com");
    user.setAddress("abc 88");
    user.setNewsletter(true);
    user.setSex("M");
    user.setFramework(new ArrayList<String>(Arrays.asList("Spring MVC", "GWT")));
    user.setSkill(new ArrayList<String>(Arrays.asList("Spring", "Grails", "Groovy")));
    user.setCountry("SG");
    user.setNumber(2);
    model.addAttribute("userForm", user);

    populateDefaultModel(model);

    return "users/userform";

}

// show update form
@RequestMapping(value = "/users/{id}/update", method = RequestMethod.GET)
public String showUpdateUserForm(@PathVariable("id") int id, Model model) {

    logger.debug("showUpdateUserForm() : {}", id);

    User user = userService.findById(id);
    model.addAttribute("userForm", user);

    populateDefaultModel(model);

    return "users/userform";

}

// delete user
@RequestMapping(value = "/users/{id}/delete", method = RequestMethod.POST)
public String deleteUser(@PathVariable("id") int id,
    final RedirectAttributes redirectAttributes) {

    logger.debug("deleteUser() : {}", id);

    userService.delete(id);

    redirectAttributes.addFlashAttribute("css", "success");
    redirectAttributes.addFlashAttribute("msg", "User is deleted!");
}

```

```

        return "redirect:/users";
    }

    // show user
    @RequestMapping(value = "/users/{id}", method = RequestMethod.GET)
    public String showUser(@PathVariable("id") int id, Model model) {

        logger.debug("showUser() id: {}", id);

        User user = userService.findById(id);
        if (user == null) {
            model.addAttribute("css", "danger");
            model.addAttribute("msg", "User not found");
        }
        model.addAttribute("user", user);

        return "users/show";
    }

    private void populateDefaultModel(Model model) {

        List<String> frameworksList = new ArrayList<String>();
        frameworksList.add("Spring MVC");
        frameworksList.add("Struts 2");
        frameworksList.add("JSF 2");
        frameworksList.add("GWT");
        frameworksList.add("Play");
        frameworksList.add("Apache Wicket");
        model.addAttribute("frameworkList", frameworksList);

        Map<String, String> skill = new LinkedHashMap<String, String>();
        skill.put("Hibernate", "Hibernate");
        skill.put("Spring", "Spring");
        skill.put("Struts", "Struts");
        skill.put("Groovy", "Groovy");
        skill.put("Grails", "Grails");
        model.addAttribute("javaSkillList", skill);

        List<Integer> numbers = new ArrayList<Integer>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);
        model.addAttribute("numberList", numbers);

        Map<String, String> country = new LinkedHashMap<String, String>();
        country.put("US", "United Stated");
        country.put("CN", "China");
        country.put("SG", "Singapore");
        country.put("MY", "Malaysia");
        model.addAttribute("countryList", country);
    }
}

```

## 4. Form Validator

### 4.1 Spring validator example.

UserFormValidator.java



```
package com.mkyong.form.validator;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

import com.mkyong.form.model.User;

import com.mkyong.form.service.UserService;

@Component
public class UserFormValidator implements Validator {

    @Autowired
    @Qualifier("emailValidator")
    EmailValidator emailValidator;

    @Autowired
    UserService userService;

    @Override
    public boolean supports(Class<?> clazz) {
        return User.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {

        User user = (User) target;

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "NotEmpty.userForm.name");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "email", "NotEmpty.userForm.email");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "address", "NotEmpty.userForm.address");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "password", "NotEmpty.userForm.password");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "confirmPassword", "NotEmpty.userForm.confirmPassword");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "sex", "NotEmpty.userForm.sex");
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "country", "NotEmpty.userForm.country");

        if(!emailValidator.valid(user.getEmail())){
            errors.rejectValue("email", "Pattern.userForm.email");
        }

        if(user.getNumber()==null || user.getNumber()<=0){
            errors.rejectValue("number", "NotEmpty.userForm.number");
        }

        if(user.getCountry().equalsIgnoreCase("none")){
            errors.rejectValue("country", "NotEmpty.userForm.country");
        }

        if (!user.getPassword().equals(user.getConfirmPassword())) {
            errors.rejectValue("confirmPassword", "Diff.userform.confirmPassword");
        }

        if (user.getFramework() == null || user.getFramework().size() < 2) {
            errors.rejectValue("framework", "Valid.userForm.framework");
        }

        if (user.getSkill() == null || user.getSkill().size() < 3) {
            errors.rejectValue("skill", "Valid.userForm.skill");
        }

    }
}
```

validation.properties

```
NotEmpty.userForm.name = Name is required!
NotEmpty.userForm.email = Email is required!
NotEmpty.userForm.address = Address is required!
NotEmpty.userForm.password = Password is required!
NotEmpty.userForm.confirmPassword = Confirm password is required!
NotEmpty.userForm.sex = Sex is required!
NotEmpty.userForm.number = Number is required!
NotEmpty.userForm.country = Country is required!
Valid.userForm.framework = Please select at least two frameworks!
Valid.userForm.skill = Please select at least three skills!
Diff.userForm.confirmPassword = Passwords do not match, please retype!
Pattern.userForm.email = Invalid Email format!
```

To run the Spring Validator, add the validator via `@InitBinder` and annotate the model with `@Validated`

```
@InitBinder
protected void initBinder(WebDataBinder binder) {
    binder.setValidator(userFormValidator);
}

@RequestMapping(value = "/users", method = RequestMethod.POST)
public String saveOrUpdateUser(... @Validated User user,
    ...) {

    //...

}
```

Or run it manually.

```
@RequestMapping(value = "/users", method = RequestMethod.POST)
public String saveOrUpdateUser(... User user,
    ...) {
    userFormValidator.validate(user, result);
    //...
}
```

## 5. HTML Forms

All the HTML forms are css styling with Bootstrap framework, and using Spring form tags to do the display and form binding.

### 5.1 User Listings.

```
list.jsp
```

```

<%@ page session="false"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ 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 lang="en">

<jsp:include page="../fragments/header.jsp" />

<body>

    <div class="container">

        <c:if test="${not empty msg}">
            <div class="alert alert-${css} alert-dismissible" role="alert">
                <button type="button" class="close" data-dismiss="alert"
                    aria-label="Close">
                    <span aria-hidden="true">×</span>
                </button>
                <strong>${msg}</strong>
            </div>
        </c:if>

        <h1>All Users</h1>

        <table class="table table-striped">
            <thead>
                <tr>
                    <th>#ID</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>framework</th>
                    <th>Action</th>
                </tr>
            </thead>

            <c:forEach var="user" items="${users}">
                <tr>
                    <td>
                        ${user.id}
                    </td>
                    <td>${user.name}</td>
                    <td>${user.email}</td>
                    <td>
                        <c:forEach var="framework" items="${user.framework}"
                            varStatus="loop">
                            ${framework}
                        <c:if test="${not loop.last}">,</c:if>
                        </c:forEach>
                    </td>
                    <td>
                        <spring:url value="/users/${user.id}" var="userUrl" />
                        <spring:url value="/users/${user.id}/delete" var="deleteUrl" />
                        <spring:url value="/users/${user.id}/update" var="updateUrl" />

                        <button class="btn btn-info"
                            onclick="location.href='${userUrl}'">Query</button>
                        <button class="btn btn-primary"
                            onclick="location.href='${updateUrl}'">Update</button>
                        <button class="btn btn-danger"
                            onclick="this.disabled=true;post('${deleteUrl}')">Delete</button>
                    </td>
                </tr>
            </c:forEach>
        </table>

    </div>

    <jsp:include page="../fragments/footer.jsp" />

</body>
</html>

```

show.jsp

```
<%@ page session="false"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>
<html lang="en">

<jsp:include page="../fragments/header.jsp" />

<div class="container">

    <c:if test="${not empty msg}">
        <div class="alert alert-${css} alert-dismissible" role="alert">
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">x</span>
            </button>
            <strong>${msg}</strong>
        </div>
    </c:if>

    <h1>User Detail</h1>
    <br />

    <div class="row">
        <label class="col-sm-2">ID</label>
        <div class="col-sm-10">${user.id}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Name</label>
        <div class="col-sm-10">${user.name}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Email</label>
        <div class="col-sm-10">${user.email}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Address</label>
        <div class="col-sm-10">${user.address}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Newsletter</label>
        <div class="col-sm-10">${user.newsletter}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Web Frameworks</label>
        <div class="col-sm-10">${user.framework}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Sex</label>
        <div class="col-sm-10">${user.sex}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Number</label>
        <div class="col-sm-10">${user.number}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Country</label>
        <div class="col-sm-10">${user.country}</div>
    </div>

    <div class="row">
        <label class="col-sm-2">Skill</label>
        <div class="col-sm-10">${user.skill}</div>
    </div>

</div>
```

```
</div>  
  
<jsp:include page="../../fragments/footer.jsp" />  
  
</body>  
</html>
```

```
userform.jsp
```

```
<%@ page session="false"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>
<html lang="en">

<jsp:include page="../fragments/header.jsp" />

<div class="container">

    <c:choose>
        <c:when test="${userForm['new']}">
            <h1>Add User</h1>
        </c:when>
        <c:otherwise>
            <h1>Update User</h1>
        </c:otherwise>
    </c:choose>
    <br />

    <spring:url value="/users" var="userActionUrl" />

    <form:form class="form-horizontal" method="post"
        modelAttribute="userForm" action="${userActionUrl}">

        <form:hidden path="id" />

        <spring:bind path="name">
            <div class="form-group ${status.error ? 'has-error' : ''}">
                <label class="col-sm-2 control-label">Name</label>
                <div class="col-sm-10">
                    <form:input path="name" type="text" class="form-control"
                        id="name" placeholder="Name" />
                    <form:errors path="name" class="control-label" />
                </div>
            </div>
        </spring:bind>

        <spring:bind path="email">
            <div class="form-group ${status.error ? 'has-error' : ''}">
                <label class="col-sm-2 control-label">Email</label>
                <div class="col-sm-10">
                    <form:input path="email" class="form-control"
                        id="email" placeholder="Email" />
                    <form:errors path="email" class="control-label" />
                </div>
            </div>
        </spring:bind>

        <spring:bind path="password">
            <div class="form-group ${status.error ? 'has-error' : ''}">
                <label class="col-sm-2 control-label">Password</label>
                <div class="col-sm-10">
                    <form:password path="password" class="form-control"
                        id="password" placeholder="password" />
                    <form:errors path="password" class="control-label" />
                </div>
            </div>
        </spring:bind>

        <spring:bind path="confirmPassword">
            <div class="form-group ${status.error ? 'has-error' : ''}">
                <label class="col-sm-2 control-label">confirm Password</label>
                <div class="col-sm-10">
                    <form:password path="confirmPassword" class="form-control"
                        id="password" placeholder="password" />
                    <form:errors path="confirmPassword" class="control-label" />
                </div>
            </div>
        </spring:bind>

        <spring:bind path="address">
            <div class="form-group ${status.error ? 'has-error' : ''}">
```

```

<label class="col-sm-2 control-label">Address</label>
<div class="col-sm-10">
  <form:textarea path="address" rows="5" class="form-control"
    id="address" placeholder="address" />
  <form:errors path="address" class="control-label" />
</div>
</div>
</spring:bind>

<spring:bind path="newsletter">
  <div class="form-group ${status.error ? 'has-error' : ''}">
    <label class="col-sm-2 control-label">Newsletter</label>
    <div class="col-sm-10">
      <div class="checkbox">
        <label>
          <form:checkbox path="newsletter" id="newsletter" />
        </label>
        <form:errors path="newsletter" class="control-label" />
      </div>
    </div>
  </div>
</spring:bind>

<spring:bind path="framework">
  <div class="form-group ${status.error ? 'has-error' : ''}">
    <label class="col-sm-2 control-label">Web Frameworks</label>
    <div class="col-sm-10">
      <form:checkboxes path="framework" items="${frameworkList}"
        element="label class='checkbox-inline'" />
      <br />
      <form:errors path="framework" class="control-label" />
    </div>
  </div>
</spring:bind>

<spring:bind path="sex">
  <div class="form-group ${status.error ? 'has-error' : ''}">
    <label class="col-sm-2 control-label">Sex</label>
    <div class="col-sm-10">
      <label class="radio-inline">
        <form:radio button path="sex" value="M" /> Male
      </label>
      <label class="radio-inline">
        <form:radio button path="sex" value="F" /> Female
      </label> <br />
      <form:errors path="sex" class="control-label" />
    </div>
  </div>
</spring:bind>

<spring:bind path="number">
  <div class="form-group ${status.error ? 'has-error' : ''}">
    <label class="col-sm-2 control-label">Number</label>
    <div class="col-sm-10">
      <form:radiobuttons path="number" items="${numberList}"
        element="label class='radio-inline'" />
      <br />
      <form:errors path="number" class="control-label" />
    </div>
  </div>
</spring:bind>

<spring:bind path="country">
  <div class="form-group ${status.error ? 'has-error' : ''}">
    <label class="col-sm-2 control-label">Country</label>
    <div class="col-sm-5">
      <form:select path="country" class="form-control">
        <form:option value="NONE" label="--- Select ---" />
        <form:options items="${countryList}" />
      </form:select>
      <form:errors path="country" class="control-label" />
    </div>
    <div class="col-sm-5"></div>
  </div>
</spring:bind>

<spring:bind path="skill">

```



```

<div class="form-group ${status.error ? 'has-error' : ''}">
  <label class="col-sm-2 control-label">Java Skills</label>
  <div class="col-sm-5">
    <form:select path="skill" items="${javaSkillList}"
      multiple="true" size="5" class="form-control" />
    <form:errors path="skill" class="control-label" />
  </div>
  <div class="col-sm-5"></div>
</div>
</spring:bind>

<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
    <c:choose>
      <c:when test="${userForm['new']}">
        <button type="submit" class="btn-lg btn-primary pull-right">Add
          </button>
      </c:when>
      <c:otherwise>
        <button type="submit" class="btn-lg btn-primary pull-right">Update
          </button>
      </c:otherwise>
    </c:choose>
  </div>
</div>
</form:form>

</div>

<jsp:include page="../../fragments/footer.jsp" />

</body>
</html>

```

## 6. Database Stuff

### 6.1 Create a table and insert some data for testing.

create-db.sql

```

CREATE TABLE users (
  id INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 100, INCREMENT BY 1) PRIMARY KEY,
  name VARCHAR(30),
  email VARCHAR(50),
  address VARCHAR(255),
  password VARCHAR(20),
  newsletter BOOLEAN,
  framework VARCHAR(500),
  sex VARCHAR(1),
  NUMBER INTEGER,
  COUNTRY VARCHAR(10),
  SKILL VARCHAR(500)
);

```

insert-data.sql

```

INSERT INTO users (name, email, framework) VALUES ('mkyong', 'mkyong@gmail.com', 'Spring MVC, GWT');
INSERT INTO users (name, email) VALUES ('alex', 'alex@yahoo.com', 'Spring MVC, GWT');
INSERT INTO users (name, email) VALUES ('joel', 'joel@gmail.com', 'Spring MVC, GWT');

```

### 6.2 Start a HSQLDB embedded database, create a datasource and jdbcTemplate.

SpringDBConfig.java

```
package com.mkyong.form.config;

import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabase;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseBuilder;
import org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseType;

@Configuration
public class SpringDBConfig {

    @Autowired
    DataSource dataSource;

    @Bean
    public NamedParameterJdbcTemplate getNamedParameterJdbcTemplate() {
        return new NamedParameterJdbcTemplate(dataSource);
    }

    @Bean
    public DataSource getDataSource() {
        EmbeddedDatabaseBuilder builder = new EmbeddedDatabaseBuilder();
        EmbeddedDatabase db = builder.setName("testdb")
            .setType(EmbeddedDatabaseType.HSQL)
            .addScript("db/sql/create-db.sql")
            .addScript("db/sql/insert-data.sql").build();
        return db;
    }
}
```

### 6.3 User object.

```
user.java
```

```
package com.mkyong.form.model;

import java.util.List;

public class User {
    // form:hidden - hidden value
    Integer id;

    // form:input - textbox
    String name;

    // form:input - textbox
    String email;

    // form:textarea - textarea
    String address;

    // form:input - password
    String password;

    // form:input - password
    String confirmPassword;

    // form:checkbox - single checkbox
    boolean newsletter;

    // form:checkboxes - multiple checkboxes
    List<String> framework;

    // form:radio - radio button
    String sex;

    // form:radio - radio button
    Integer number;

    // form:select - form:option - dropdown - single select
    String country;

    // form:select - multiple=true - dropdown - multiple select
    List<String> skill;

    //Check if this is for New or Update
    public boolean isNew() {
        return (this.id == null);
    }

    //...
}
```

## 7. Services and DAO

### UserService.java

```
package com.mkyong.form.service;

import java.util.List;
import com.mkyong.form.model.User;

public interface UserService {

    User findById(Integer id);

    List<User> findAll();

    void saveOrUpdate(User user);

    void delete(int id);

}
```

### UserServiceImpl.java

```
package com.mkyong.form.service;

import java.util.List;

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

import com.mkyong.form.dao.UserDao;
import com.mkyong.form.model.User;

@Service("userService")
public class UserServiceImpl implements UserService {

    UserDao userDao;

    @Autowired
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }

    @Override
    public User findById(Integer id) {
        return userDao.findById(id);
    }

    @Override
    public List<User> findAll() {
        return userDao.findAll();
    }

    @Override
    public void saveOrUpdate(User user) {

        if (findById(user.getId())==null) {
            userDao.save(user);
        } else {
            userDao.update(user);
        }

    }

    @Override
    public void delete(int id) {
        userDao.delete(id);
    }

}
```

#### UserDao.java

```
package com.mkyong.form.dao;

import java.util.List;

import com.mkyong.form.model.User;

public interface UserDao {

    User findById(Integer id);

    List<User> findAll();

    void save(User user);

    void update(User user);

    void delete(Integer id);

}
```

#### UserDaoImpl.java

```

package com.mkyong.form.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.support.GeneratedKeyHolder;
import org.springframework.jdbc.support.KeyHolder;
import org.springframework.stereotype.Repository;
import org.springframework.util.StringUtils;

import com.mkyong.form.model.User;

@Repository
public class UserDaoImpl implements UserDao {

    NamedParameterJdbcTemplate namedParameterJdbcTemplate;

    @Autowired
    public void setNamedParameterJdbcTemplate(
        NamedParameterJdbcTemplate namedParameterJdbcTemplate) {
        this.namedParameterJdbcTemplate = namedParameterJdbcTemplate;
    }

    @Override
    public User findById(Integer id) {

        Map<String, Object> params = new HashMap<String, Object>();
        params.put("id", id);

        String sql = "SELECT * FROM users WHERE id=:id";

        User result = null;
        try {
            result = namedParameterJdbcTemplate
                .queryForObject(sql, params, new UserMapper());
        } catch (EmptyResultDataAccessException e) {
            // do nothing, return null
        }

        return result;
    }

    @Override
    public List<User> findAll() {

        String sql = "SELECT * FROM users";
        List<User> result = namedParameterJdbcTemplate.query(sql, new UserMapper());
        return result;
    }

    @Override
    public void save(User user) {

        KeyHolder keyHolder = new GeneratedKeyHolder();

        String sql = "INSERT INTO USERS(NAME, EMAIL, ADDRESS, PASSWORD, NEWSLETTER, FRAMEWORK, SEX, NUMBER, COUNTRY, SKILL) "
            + "VALUES (:name, :email, :address, :password, :newsletter, :framework, :sex, :number, :country, :skill)";

        namedParameterJdbcTemplate.update(sql, getSqlParameterByModel(user), keyHolder);
        user.setId(keyHolder.getKey().intValue());
    }
}

```

```

@Override
public void update(User user) {

    String sql = "UPDATE USERS SET NAME=:name, EMAIL=:email, ADDRESS=:address, "
        + "PASSWORD=:password, NEWSLETTER=:newsletter, FRAMEWORK=:framework, "
        + "SEX=:sex, NUMBER=:number, COUNTRY=:country, SKILL=:skill WHERE id=:id";

    namedParameterJdbcTemplate.update(sql, getSqlParameterByModel(user));

}

@Override
public void delete(Integer id) {

    String sql = "DELETE FROM USERS WHERE id= :id";
    namedParameterJdbcTemplate.update(sql, new MapSqlParameterSource("id", id));

}

private SqlParameterSource getSqlParameterByModel(User user) {

    MapSqlParameterSource paramSource = new MapSqlParameterSource();
    paramSource.addValue("id", user.getId());
    paramSource.addValue("name", user.getName());
    paramSource.addValue("email", user.getEmail());
    paramSource.addValue("address", user.getAddress());
    paramSource.addValue("password", user.getPassword());
    paramSource.addValue("newsletter", user.isNewsletter());

    // join String
    paramSource.addValue("framework", convertListToDelimitedString(user.getFramework()));
    paramSource.addValue("sex", user.getSex());
    paramSource.addValue("number", user.getNumber());
    paramSource.addValue("country", user.getCountry());
    paramSource.addValue("skill", convertListToDelimitedString(user.getSkill()));

    return paramSource;
}

private static final class UserMapper implements RowMapper<User> {

    public User mapRow(ResultSet rs, int rowNum) throws SQLException {
        User user = new User();
        user.setId(rs.getInt("id"));
        user.setName(rs.getString("name"));
        user.setEmail(rs.getString("email"));
        user.setFramework(convertDelimitedStringToList(rs.getString("framework")));
        user.setAddress(rs.getString("address"));
        user.setCountry(rs.getString("country"));
        user.setNewsletter(rs.getBoolean("newsletter"));
        user.setNumber(rs.getInt("number"));
        user.setPassword(rs.getString("password"));
        user.setSex(rs.getString("sex"));
        user.setSkill(convertDelimitedStringToList(rs.getString("skill")));
        return user;
    }
}

private static List<String> convertDelimitedStringToList(String delimitedString) {

    List<String> result = new ArrayList<String>();

    if (!StringUtils.isEmpty(delimitedString)) {
        result = Arrays.asList(StringUtils.delimitedListToStringArray(delimitedString, ","));
    }
    return result;
}

private String convertListToDelimitedString(List<String> list) {

    String result = "";
    if (list != null) {
        result = StringUtils.arrayToCommaDelimitedString(list.toArray());
    }
    return result;
}

```

```
        return result;  
    }  
}
```

## 8. Spring Configuration

Mixing Spring XML and JavaConfig.

### SpringWebConfig.java

```
package com.mkyong.form.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.context.support.ResourceBundleMessageSource;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  
import org.springframework.web.servlet.view.InternalResourceViewResolver;  
import org.springframework.web.servlet.view.JstlView;  
  
@EnableWebMvc  
@Configuration  
@ComponentScan({ "com.mkyong.form.web", "com.mkyong.form.service", "com.mkyong.form.dao",  
    "com.mkyong.form.exception", "com.mkyong.form.validator" })  
public class SpringWebConfig extends WebMvcConfigurerAdapter {  
  
    @Override  
    public void addResourceHandlers(ResourceHandlerRegistry registry) {  
        registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");  
    }  
  
    @Bean  
    public InternalResourceViewResolver viewResolver() {  
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();  
        viewResolver.setViewClass(JstlView.class);  
        viewResolver.setPrefix("/WEB-INF/views/jsp/");  
        viewResolver.setSuffix(".jsp");  
        return viewResolver;  
    }  
  
    @Bean  
    public ResourceBundleMessageSource messageSource() {  
        ResourceBundleMessageSource rb = new ResourceBundleMessageSource();  
        rb.setBasenames(new String[] { "messages/messages", "messages/validation" });  
        return rb;  
    }  
}
```

### spring-web-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"  
    xmlns:context="http://www.springframework.org/schema/context"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:mvc="http://www.springframework.org/schema/mvc"  
    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">  
  
    <!-- Scan the JavaConfig -->  
    <context:component-scan base-package="com.mkyong.form.config" />  
  
</beans>
```

web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">

  <display-name>Spring3 MVC Application</display-name>

  <servlet>
    <servlet-name>spring-web</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>spring-web</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <error-page>
    <error-code>500</error-code>
    <location>/WEB-INF/views/jsp/error.jsp</location>
  </error-page>

  <error-page>
    <error-code>404</error-code>
    <location>/WEB-INF/views/jsp/error.jsp</location>
  </error-page>

  <error-page>
    <location>/WEB-INF/views/jsp/error.jsp</location>
  </error-page>

</web-app>
```

## 9. Demo

Download the project and type `mvn jetty:run`

```
$ mvn jetty:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building SpringMVC + Form Handling Example 1.0-SNAPSHOT
[INFO] -----
//...
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 10 seconds.
```

**9.1 List all users.** <http://localhost:8080/spring-mvc-form/users>



Spring MVC Form Handling Example

localhost:8080/spring-mvc-form/users

Spring MVC Form

**Add User**

`/users/add`

### All Users

#ID	Name	Email	framework	Action
100	mkyong	mkyong@gmail.com	Spring MVC , GWT	<a href="#">Query</a> <a href="#">Update</a> <a href="#">Delete</a>
101	alex	alex@yahoo.com	Spring MVC , PLAY	<a href="#">Query</a> <a href="#">Update</a> <a href="#">Delete</a>
102	joel	joel@gmail.com	Spring MVC , JSF 2	<a href="#">Query</a> <a href="#">Update</a> <a href="#">Delete</a>

© Mkyong.com 2015

`/users/{id}`

`/users/{id}/update`

`/users/{id}/delete`

Mkyong.com


(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form-handling-demo1.png>)

**9.2 Add an user.** <http://localhost:8080/spring-mvc-form/users/add>

Spring MVC Form

Add User

# Add User



Name

Email

Password

confirm Password

Address

Newsletter

☒

Web Frameworks

☒ Spring MVC ☐ Struts 2 ☐ JSF 2 ☒ GWT ☐ Play ☐ Apache Wicket

Sex

☒ Male ☐ Female

Number

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

Country

Java Skills

Hibernate

Spring

Struts

Groovy

Grails

Add

© Mkyong.com 2015


(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form-handling-demo-add.png>)

### 9.3 Form validation.

Spring MVC Form

Add User

# Add User



**Name**

mkyong123

**Email**

Email

Email is required!  
Invalid Email format!

**Password**

password

Password is required!

**confirm Password**

password

Confirm password is required!

**Address**

abc 88

**Newsletter**

☒

**Web Frameworks**

☒ Spring MVC ☐ Struts 2 ☐ JSF 2 ☐ GWT ☐ Play ☐ Apache Wicket

Please select at least two frameworks!

**Sex**

☒ Male ☐ Female

**Number**

☐ 1 ☒ 2 ☐ 3 ☐ 4 ☐ 5

**Country**

Singapore

**Java Skills**

Hibernate  
Spring  
Struts  
Groovy  
Grails

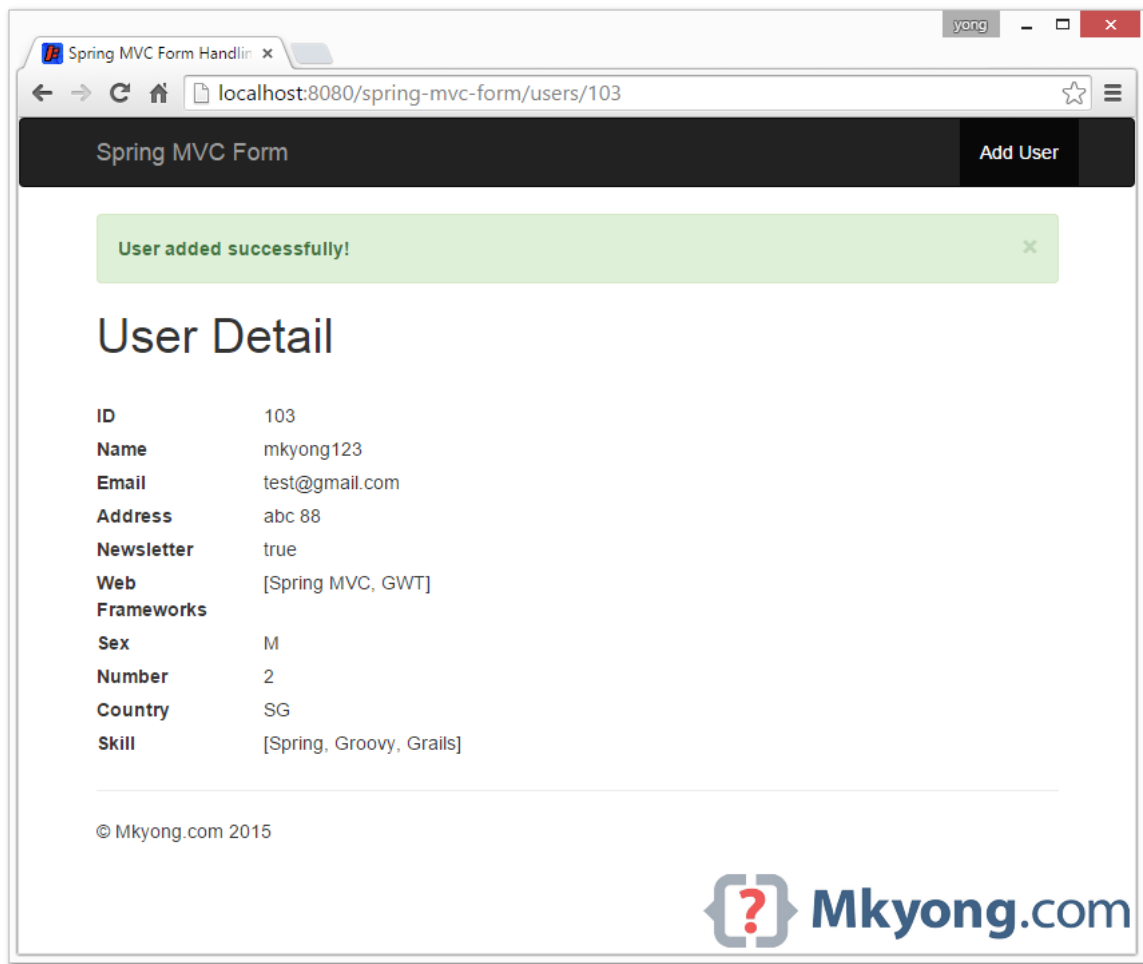
Please select at least three skills!

Add

© Mkyong.com 2015

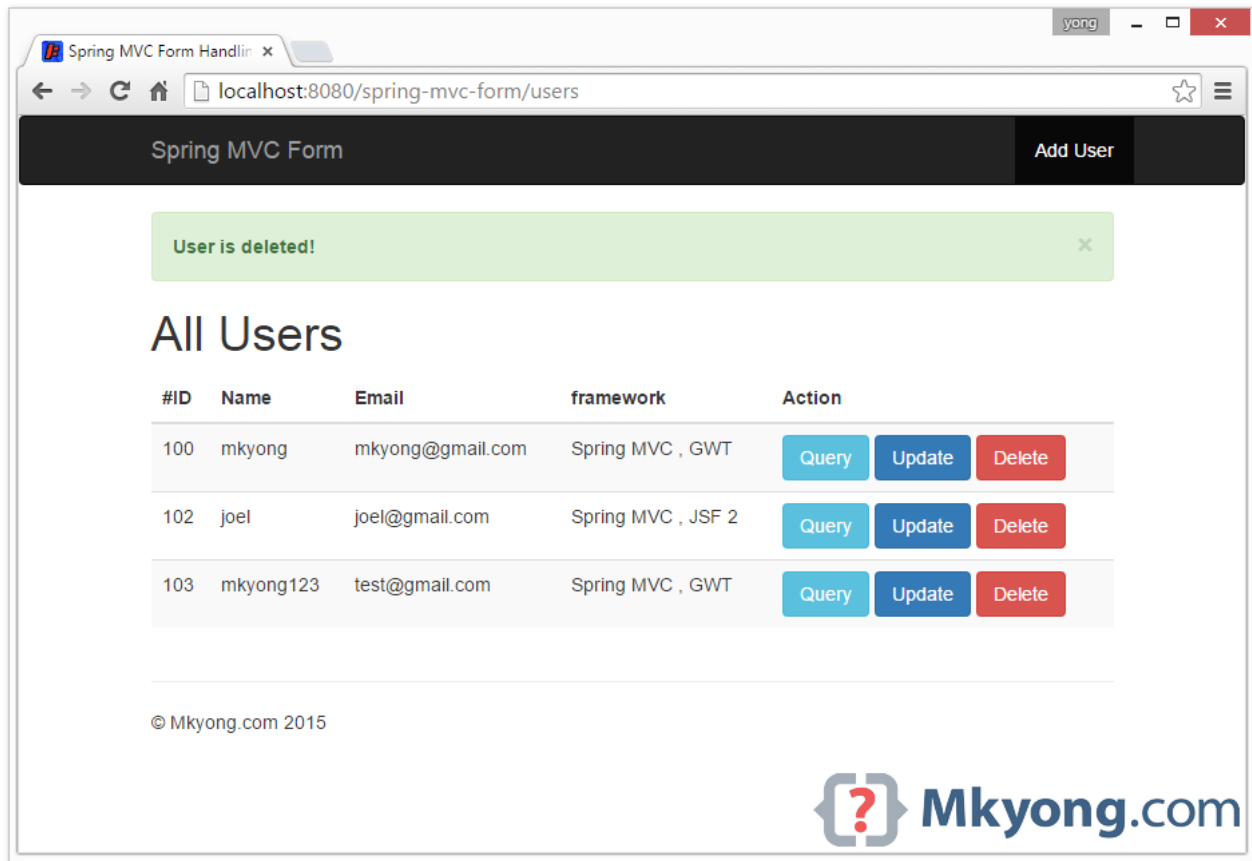
(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form-handling-demo-add-error.png>)

9.4 New user is added. <http://localhost:8080/spring-mvc-form/users>



(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form-handling-demo-add-done.png>)

**9.5 Delete an user.** <http://localhost:8080/spring-mvc-form/users/104/delete>



(<http://www.mkyong.com/wp-content/uploads/2010/08/spring-mvc-form-handling-demo-delete.png>)

## Download Source Code

Download it - [spring4-form-handle-example.zip](http://www.mkyong.com/wp-content/uploads/2010/08/spring4-form-handle-example.zip) (<http://www.mkyong.com/wp-content/uploads/2010/08/spring4-form-handle-example.zip>) (80 kb)

Github - TBD

## References

1. Using Spring's form tag library (<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/view.html#view-jsp-formtaglib>)
2. Spring MVC 3 Validation (<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/validation.html#validation-mvc>)
3. RedirectAttributes JavaDoc (<http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/mvc/support/RedirectAttributes.html>)

Tags : [form handling](http://www.mkyong.com/tag/form-handling/) (<http://www.mkyong.com/tag/form-handling/>) [spring mvc](http://www.mkyong.com/tag/spring-mvc/) (<http://www.mkyong.com/tag/spring-mvc/>)

## Share this article on

Twitter ([https://twitter.com/intent/tweet?text=Spring MVC form handling example&url=http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example/&via=mkyong](https://twitter.com/intent/tweet?text=Spring+MVC+form+handling+example&url=http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example/&via=mkyong)) Facebook (<https://www.facebook.com/sharer/sharer.php?u=http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example/>) Google+ (<https://plus.google.com/share?url=http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example/>)



DO YOU REALLY KNOW YOUR MOBILES?



How many times does the average person unlock their phone a day? **15**

▼


**A** 20

**B** 110

**C** 55

**D** 201

Your Score **0** Question **1/10**



**Secure her future.**  
Always seat her in the correct seat.  
Visit [safercar.gov/therightseat](http://safercar.gov/therightseat)

Child Car Safety  
NHTSA

## About the Author



**mkyong**

Founder of Mkyong.com (<http://mkyong.com>), love Java and open source stuff. Follow him on Twitter (<https://twitter.com/mkyong>), or befriend him on Facebook (<http://www.facebook.com/java.tutorial>) or Google Plus (<https://plus.google.com/110948163568945735692?rel=author>). If you like my tutorials, consider make a donation to these charities (<http://www.mkyong.com/blog/donate-to-charity/>).

## Related Posts


- 142k Spring MVC InternalResourceViewResolver example (</spring-mvc/spring-mvc-internalresourceviewresolver-example/>)
- 133k Spring MVC form errors tag example (</spring-mvc/spring-mvc-form-errors-tag-example/>)
- 126k Spring MVC hidden value example (</spring-mvc/spring-mvc-hidden-value-example/>)
- 398k Spring MVC file upload example (</spring-mvc/spring-mvc-file-upload-example/>)
- 329k Spring MVC form handling annotation example (</spring-mvc/spring-mvc-form-handling-annotation-example/>)
- 36k Spring 3 MVC and RSS feed example (</spring-mvc/spring-3-mvc-and-rss-feed-example/>)
- 5k ClassNotFoundException : com.sun.syndication.feed.WireFeed (</spring-mvc/classnotfoundexception-com-sun-syndication-feed-wirefeed/>)
- 163k Spring MVC and List Example (</spring-mvc/spring-mvc-and-list-example/>)
- 279k Gradle - Spring 4 MVC Hello World Example (</spring-mvc/gradle-spring-mvc-web-project-example/>)
- 31k Google App Engine + JDO + Spring MVC, CRUD example (</google-app-engine/google-app-engine-jdo-spring-mvc-crud-example/>)

## Popular Posts

- JAXB hello world example (</java/jaxb-hello-world-example/>) 1m
- How to get current timestamps in Java (</java/how-to-get-current-timestamps-in-java/>) 678k
- JSON.simple example - Read and write JSON (</java/json-simple-example-read-and-write-json/>) 1.2m
- Spring Security Custom Login Form Example (</spring-security/spring-security-form-login-example/>) 671k
- Android ImageView example (</android/android-imageview-example/>) 663k
- Android Tutorial (</tutorials/android-tutorial/>) 1.1m
- Spring MVC Tutorial (</tutorials/spring-mvc-tutorials/>) 1.8m
- How to read XML file in Java - (DOM Parser) (</java/how-to-read-xml-file-in-java-dom-parser/>) 1.8m
- Jackson 2 - Convert Java Object to / from JSON (</java/jackson-2-convert-java-object-to-from-json/>) 763k
- How to convert Java object to / from JSON (Jackson) (</java/how-to-convert-java-object-to-from-json-jackson/>) 1.6m

Java XML Tutorial (/tutorials/java-xml-tutorials/)	767k
How to write to file in Java - BufferedWriter (/java/how-to-write-to-file-in-java-bufferedwriter-example/)	1.2m
How to create a Web Application Project with Maven (/maven/how-to-create-a-web-application-project-with-maven/)	799k
JSF 2.0 hello world example (/jsf2/jsf-2-0-hello-world-example/)	743k
Android custom dialog example (/android/android-custom-dialog-example/)	723k
Java - How to get current date time (/java/java-how-to-get-current-date-time-date-and-calender/)	2.2m
How to convert Java object to / from JSON (Gson) (/java/how-do-convert-java-object-to-from-json-format-gson-api/)	1.2m
Spring Security Tutorial (/tutorials/spring-security-tutorials/)	1.1m
How to read file in Java - BufferedReader (/java/how-to-read-file-from-java-bufferedReader-example/)	1.4m
Spring JdbcTemplate Querying examples (/spring/spring-jdbcTemplate-querying-examples/)	710k

### Leave a Reply



Sort by: newest | oldest | [most voted](#)



Kiran

Perfect code for any beginner to understand the MVC design and their separations of layers.

Guest



🕒 4 days 11 hours ago



Tushar Girase

Awesome design man I was searching for kind of website which will give me the best practice to develop a website.

Guest



🕒 7 months 8 days ago



Problematic

Hey thanx for tutorial :) Sadly , Spring is throwing an UnsatisfiedDependencyException :( This is full message Error creating bean with name 'springDBConfig': Unsatisfied dependency expressed through field 'dataSource'; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'getDataSource' defined in com.form.config.SpringDBConfig: Bean instantiation via factory method failed; nested exception is org.springframework.beans.BeanInstantiationException: Failed to instantiate [javax.sql.DataSource]: Circular reference involving containing bean 'springDBConfig' – consider declaring the factory method as static for independence from its containing instance. Factory method 'getDataSource' threw exception; nested exception is org.springframework.jdbc.datasource.init.ScriptStatementFailedException: Failed to execute SQL script statement #2 of class path resource [db/sql/insert-data.sql]: INSERT INTO users... Read more »

Guest



🕒 7 months 23 days ago



Soumya Ranjan Das Srd

Thanks sir.

Guest



🕒 8 months 3 days ago



Dynamic checkboxes

Guest

Thank you for your post, please create new post about dynamic checkboxes coming from database using many to many annotation in Spring MVC

  0  [REPLY](#)

🕒 8 months 10 days ago



Dávid Hreško

Guest

How can i change embeded hsqldb to mysql with hibernate ? ...is it possible ? ...answer please :)

  0  [REPLY](#)

🕒 9 months 28 days ago



Tom

Guest

I have this tag: in the form for updating user info, use\_id is an Integer object, but when click submit, user\_id=null, if I change user\_id to an int variable then result is user\_id=0. So I can't update user info. Can you help solve this problem?

  0  [REPLY](#)

🕒 10 months 22 days ago



Basu Dapalapur

Guest

I am trying to Add one new column in the same users table , Its not working, DROP TABLE users IF EXISTS; CREATE TABLE users ( id INTEGER GENERATED BY DEFAULT AS IDENTITY(START WITH 100, INCREMENT BY 1) PRIMARY KEY, name VARCHAR(30), email VARCHAR(50), address VARCHAR(255), password VARCHAR(20), newsletter BOOLEAN, framework VARCHAR(500), sex VARCHAR(1), NUMBER INTEGER, COUNTRY VARCHAR(10), SKILL VARCHAR(500), mobileNumber VARCHAR(15) ); private SqlParameterSource getSqlParameterByModel(User user) { // Unable to handle List or Array // BeanPropertySqlParameterSource MapSqlParameterSource paramSource = new MapSqlParameterSource(); paramSource.addValue("id", user.getId()); ..... //paramSource.addValue("mobileNumber", user.getMobileNumber()); // join String .... return paramSource; } private static final class UserMapper implements... Read more »

  0  [REPLY](#)

🕒 11 months 13 days ago



Meena Kannan

Guest

Hi Can anybody pls help me to setup this project in eclipse?

  0  [REPLY](#)

🕒 1 year 1 month ago



kolivanov84

Guest

Hi, I'm following your tutorial, I already created a database mysql without using EmbeddedDatabaseBuilder class. How can I create the db connection? How can I create a mysql datasource using SpringDBConfig class?

  0  [REPLY](#)

🕒 1 year 1 month ago



Zacson Skaria

Guest

Can you please share the header.jsp and footer.jsp ?

  0  [REPLY](#)

🕒 1 year 1 month ago



Vinicius Telles

Guest

MK great example. Congratulations.

  0  [REPLY](#)

🕒 1 year 2 months ago



NaN

Guest

validation.properties might be interesting to show as a file in this tutorial. It's in the download. But unexplained.





🕒 1 year 4 months ago



Guest

NaN

Hi mkyong,

First of, you make some great tutorials mate.

Just a little note: I am using your code in a Spring 4.2.4 MVC

I ran in to a strange issue after upgrading from 3.x which forced me to look into this bit of code in your list.jsp at the delete button you have:

Delete

you need to add \$. to the post bit like so

Delete

Thanks again for the great examples.



🕒 1 year 4 months ago ^



Guest

NaN

I was wrong about adding "\$.".

I forgot to add hello.js ... to my config.

Maybe you should explain why this "hello.js" is needed since RedirectAttributes seem to need.

Could be cool if that wasn't needed.



🕒 1 year 4 months ago



Guest

Rudy

Hi, Why am I getting 404 error error when I try to access <http://localhost:8080/spring-mvc-form/users> (<http://localhost:8080/spring-mvc-form/users>)

🕒 1 year 6 months ago



Guest

sushil gc

Very nice tutorial.. Mr mkyong could help what to do with the onclick with delete using thymeleaf. I understand how post function on hello.js works but dont not know how to integrate spring:url in thymeleaf



🕒 1 year 7 months ago



Guest

Viswanathan Saravanan

(http://saravan.asuscomm.com)

test="\${userForm['new']}"

Where this variable is defined ? in other words where r u fetching this variable value?



🕒 1 year 8 months ago ^



Guest

NaN

Look in User.java

this boolean is the trigger for it:

```
public boolean isNew() {  
    return (this.id == null);  
}
```



🕒 1 year 4 months ago



Guest

NaN

I saw your question and thought... Yeah how was that again...

Look in User.java

```
public boolean isNew() {
    return (this.id == null);
}
```

👍 0 🗨️ REPLY

🕒 1 year 4 months ago



USRO

ok thanks....

Guest

👍 0 🗨️ REPLY

🕒 1 year 8 months ago



Edward Beckett

(<http://www.edwardbeckett.com/>)

Guest

Just wanted to give you props man... love your site ;)

👍 0 🗨️ REPLY

🕒 1 year 10 months ago



mkyong

This article is fully updated with the Spring @Controller

Guest

1. Download the source and start the embedded Jetty server.

`mvn jetty:run`

2. Deployed URL

`http://localhost:8080/spring-mvc-form/users`

*P.S Previously, this article was using the deprecated SimpleFormController*

👍 0 🗨️ REPLY

🕒 1 year 10 months ago



Spring MVC Tutorial

(<http://www.mkyong.com/tutorials/spring-mvc-tutorials/>)

[...] Form handling example Form handling in Spring MVC, XML based version.  
[...]

👍 0 🗨️ REPLY

🕒 1 year 11 months ago



Shwetank Arya

why your application url is not like – `http://localhost:8080/SpringMVCFORM/CustomForm.htm`  
(`http://localhost:8080/SpringMVCFORM/CustomForm.htm`)?

Guest

👍 0 🗨️ REPLY

🕒 2 years 1 month ago



Puneeth Shivalingaiah

Thanks again Yong. For clarity and concise...

Guest

👍 0 🗨️ REPLY

🕒 2 years 2 months ago



anass312

I got this error :

Caused by: `org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'com.mkyong.customer.controller.CustomerController#0' defined in ServletContext resource [/WEB-INF/mvc-dispatcher-servlet.xml]: Initialization of bean failed; nested exception is java.lang.Error: Unresolved compilation problem:The method supports(Class) of type CustomerValidator must override a superclass method`  
please any help.

Guest

👍 0 🗨️ REPLY

🕒 2 years 4 months ago



stupid hater

stupid tutorial not enough details or explanation stop stealing from others and remake without testing

Guest



🕒 2 years 10 months ago ^



mkyong

I never steal tutorial, all examples are unique and well tested in my development environment.

Guest



🕒 1 year 10 months ago



Kishore

difference between "successView" (will redirect to "/WEB-INF/pages/CustomerSuccess.jsp") and return new ModelAndView("CustomerSuccess","customer",customer);

Guest



🕒 3 years 1 month ago



nitesh

HTTP Status 404 – /spring-mvc-formhandling/CustomerForm.jsp continuously getting this even i have change web.xml also

Guest



🕒 3 years 5 months ago ^



Victor

try localhost:8080/customer.htm

Guest



🕒 2 years 2 months ago



Jolly Tilo

Hey Myong, 1 You forgot to specify the /welcome.htm in the bean for the xxxx-xxxx-servlet.xml. This causes A LOT for confusion making this tutorial USELESS and a headache. With this any xxxx.htm that was included would go to this Controller. 2. Also you did not specify a "How it works" portion. Making this NOT a tutorial but just a sample code with lacking explanation. 3. Lastly, yo did not explain WHY THE HELL should the for go to , than ? Yes it goes for CustomerForm first but WHY? There nothing seems to specifically indicate it in the code. THIS... Read more »

Guest



🕒 3 years 7 months ago ^



janc

Just the opposite, I found this tutorial very useful!

And for your question #3, you just have to do a big digging to find out that all the /customer\* goes to CustomerForm because of ControllerClassNameHandlerMapping in your dispatcher servlet xml.

Guest



🕒 3 years 2 months ago



Tapanesh Dash

Please god shake don't tell useless..... If u found something then mention it .....Mkyong is doing good work. ....It will hurt to the person who is doing for our shake .....Give respect .....So that u will get respect . But thanks for finding the reasons ..... :) . It was typo mistake .

Guest

Same thing u can ask politely ? .....



🕒 3 years 5 months ago



Ariel

Hi Yong, in mvc-dispatcher-servlet.xml, you have declared CustomerForm, CustomerSuccess, and CustomerValidator as the properties of CustomerController. How can we declare these as properties of CustomerController. I couldn't understand this. Can you help me in this regard.

Guest