|  |  |
| --- | --- |
|  | **Spring MVC Form Validations and Resource Handling**  **Student Registration Form Example** |
|  | we will learn about using Spring Form Tags, Form Validation using JSR-303 validation annotations, hibernate-validators, providing internationalization support using MessageSourceand accessing static resources (e.g. CSS, javascript, images) in our views using ResourceHandlerRegistry, all using annotation-based configuration. |
|  | we will create a simple application containing a student registration form, on form submission validating the user-input via JSR-303 validation annotations, overriding default messages using internationalized validation messages through properties files and also access the static resources. |
|  | JSR303 is a specification and hibernate-validator we are using in this post is an implementation, which also provides few of it’s own validation annotations. |
|  | **Following technologies being used:**   * Spring 4.0.6.RELEASE * validation-api 1.1.0.Final * hibernate-validator 5.1.2.Final * Bootstrap v3.1.0 * Maven 3 * JDK 1.6 * Tomcat 7.0.54 * Eclipse JUNO Service Release 2 |
| **1.** | **Create the directory structure** |
|  |  |
| 2. | Update pom.xml to include required dependencies |
|  | <?xml version="1.0"?>  <project xsi:schemaLocation="<a class="vglnk" href="<http://maven.apache.org/POM/4.0.0>" rel="nofollow"><span>http</span><span>://</span><span>maven</span><span>.</span><span>apache</span><span>.</span><span>org</span><span>/</span><span>POM</span><span>/</span><span>4</span><span>.</span><span>0</span><span>.</span><span>0</span></a> <a class="vglnk" href="<http://maven.apache.org/xsd/maven-4.0.0.xsd>" rel="nofollow"><span>http</span><span>://</span><span>maven</span><span>.</span><span>apache</span><span>.</span><span>org</span><span>/</span><span>xsd</span><span>/</span><span>maven</span><span>-</span><span>4</span><span>.</span><span>0</span><span>.</span><span>0</span><span>.</span><span>xsd</span></a>"      xmlns="<a class="vglnk" href="<http://maven.apache.org/POM/4.0.0>" rel="nofollow"><span>http</span><span>://</span><span>maven</span><span>.</span><span>apache</span><span>.</span><span>org</span><span>/</span><span>POM</span><span>/</span><span>4</span><span>.</span><span>0</span><span>.</span><span>0</span></a>" xmlns:xsi="<a class="vglnk" href="<http://www.w3.org/2001/XMLSchema-instance>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>w3</span><span>.</span><span>org</span><span>/</span><span>2001</span><span>/</span><span>XMLSchema</span><span>-</span><span>instance</span></a>">        <modelVersion>4.0.0</modelVersion>      <groupId>com.websystique.springmvc</groupId>      <artifactId>Spring4MVCFormValidationExample</artifactId>      <packaging>war</packaging>      <version>1.0.0</version>      <name>Spring4MVCFormValidationExample</name>        <properties>          <springframework.version>4.0.6.RELEASE</springframework.version>          <hibernate.validator.version>5.1.2.Final</hibernate.validator.version>          <javax.validation.version>1.1.0.Final</javax.validation.version>      </properties>        <dependencies>          <!-- Spring dependencies -->          <dependency>              <groupId>org.springframework</groupId>              <artifactId>spring-core</artifactId>              <version>${springframework.version}</version>          </dependency>          <dependency>              <groupId>org.springframework</groupId>              <artifactId>spring-web</artifactId>              <version>${springframework.version}</version>          </dependency>          <dependency>              <groupId>org.springframework</groupId>              <artifactId>spring-webmvc</artifactId>              <version>${springframework.version}</version>          </dependency>            <!-- jsr303 validation dependencies-->          <dependency>              <groupId>javax.validation</groupId>              <artifactId>validation-api</artifactId>              <version>${javax.validation.version}</version>          </dependency>          <dependency>              <groupId>org.hibernate</groupId>              <artifactId>hibernate-validator</artifactId>              <version>${hibernate.validator.version}</version>          </dependency>            <!-- Servlet dependencies -->          <dependency>              <groupId>javax.servlet</groupId>              <artifactId>javax.servlet-api</artifactId>              <version>3.1.0</version>          </dependency>          <dependency>              <groupId>javax.servlet.jsp</groupId>              <artifactId>javax.servlet.jsp-api</artifactId>              <version>2.3.1</version>          </dependency>          <dependency>              <groupId>javax.servlet</groupId>              <artifactId>jstl</artifactId>              <version>1.2</version>          </dependency>      </dependencies>        <build>          <pluginManagement>              <plugins>                  <plugin>                      <groupId>org.apache.maven.plugins</groupId>                      <artifactId>maven-compiler-plugin</artifactId>                      <version>3.2</version>                      <configuration>                          <source>1.6</source>                          <target>1.6</target>                      </configuration>                  </plugin>                  <plugin>                      <groupId>org.apache.maven.plugins</groupId>                      <artifactId>maven-war-plugin</artifactId>                      <version>2.4</version>                      <configuration>                          <warSourceDirectory>src/main/webapp</warSourceDirectory>                          <warName>Spring4MVCFormValidationExample</warName>                          <failOnMissingWebXml>false</failOnMissingWebXml>                      </configuration>                  </plugin>              </plugins>          </pluginManagement>          <finalName>Spring4MVCFormValidationExample</finalName>      </build>  </project> |
|  | First thing to notice here is the maven-war-plugin declaration. As we are using full annotation configuration, we don’t even include web.xml, so we will need to configure this plugin in order to avoid maven failure to build war package. On Validation part, validation-api represents the specification, while hibernate-validator is an implementation of this specification. hibernate-validator also provides few of it’s own annotations (@Email, @NotEmpty, etc..) which are not part of the specification. |
|  | Along with that, we have also included JSP/Servlet/Jstl dependencies which we will be needing as we are going to use servlet api’s and jstl view in our code. In general, containers might already contains these libraries, so we can set the scope as ‘provided’ for them in pom.xml. |
| 3. | Create POJO / Domain Object |
|  | This domain object will acts as a backing bean to the form holding data user will provide via form submission. We will annotate the properties(with validation annotations) which we want to be validated. |
|  | package keyur;    import java.io.Serializable;  import java.util.ArrayList;  import java.util.Date;  import java.util.List;    import javax.validation.constraints.NotNull;  import javax.validation.constraints.Past;  import javax.validation.constraints.Size;    import org.hibernate.validator.constraints.Email;  import org.hibernate.validator.constraints.NotEmpty;  import org.springframework.format.annotation.DateTimeFormat;    public class Student implements Serializable {        @Size(min=3, max=30)      private String firstName;        @Size(min=3, max=30)      private String lastName;        @NotEmpty      private String sex;        @DateTimeFormat(pattern="dd/MM/yyyy")      @Past @NotNull      private Date dob;        @Email @NotEmpty      private String email;        @NotEmpty      private String section;        @NotEmpty      private String country;        private boolean firstAttempt;        @NotEmpty      private List<String> subjects = new ArrayList<String>();        public String getFirstName() {          return firstName;      }        public void setFirstName(String firstName) {          this.firstName = firstName;      }        public String getLastName() {          return lastName;      }        public void setLastName(String lastName) {          this.lastName = lastName;      }        public String getSex() {          return sex;      }        public void setSex(String sex) {          this.sex = sex;      }        public Date getDob() {          return dob;      }        public void setDob(Date dob) {          this.dob = dob;      }        public String getEmail() {          return email;      }        public void setEmail(String email) {          this.email = email;      }        public String getSection() {          return section;      }        public void setSection(String section) {          this.section = section;      }        public String getCountry() {          return country;      }        public void setCountry(String country) {          this.country = country;      }        public boolean isFirstAttempt() {          return firstAttempt;      }        public void setFirstAttempt(boolean firstAttempt) {          this.firstAttempt = firstAttempt;      }        public List<String> getSubjects() {          return subjects;      }        public void setSubjects(List<String> subjects) {          this.subjects = subjects;      }        @Override      public String toString() {          return "Student [firstName=" + firstName + ", lastName=" + lastName                  + ", sex=" + sex + ", dob=" + dob + ", email=" + email                  + ", section=" + section + ", country=" + country                  + ", firstAttempt=" + firstAttempt + ", subjects=" + subjects                  + "]";      }    } |
| 4. | Add controller |
|  | Add the controller which will serve the GET and POST request. |
|  | package keyur;    import java.util.ArrayList;  import java.util.List;    import javax.validation.Valid;    import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.validation.BindingResult;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;    import com.websystique.springmvc.model.Student;    @Controller  @RequestMapping("/")  public class HelloWorldController {        /\*       \* This method will serve as default GET handler.       \*       \*/      @RequestMapping(method = RequestMethod.GET)      public String newRegistration(ModelMap model) {          Student student = new Student();          model.addAttribute("student", student);          return "enroll";      }        /\*       \* This method will be called on form submission, handling POST request       \* It also validates the user input       \*/      @RequestMapping(method = RequestMethod.POST)      public String saveRegistration(@Valid Student student, BindingResult result, ModelMap model){            if(result.hasErrors()) {                  return "enroll";          }            model.addAttribute("success", "Dear "+ student.getFirstName()+" , your Registration completed successfully");          return "success";      }        /\*       \* Method used to populate the Section list in view.       \* Note that here you can call external systems to provide real data.       \*/      @ModelAttribute("sections")      public List<String> initializeSections() {            List<String> sections = new ArrayList<String>();          sections.add("Graduate");          sections.add("Post Graduate");          sections.add("Research");          return sections;      }        /\*       \* Method used to populate the country list in view.       \* Note that here you can call external systems to provide real data.       \*/      @ModelAttribute("countries")      public List<String> initializeCountries() {            List<String> countries = new ArrayList<String>();          countries.add("USA");          countries.add("CANADA");          countries.add("FRANCE");          countries.add("GERMANY");          countries.add("ITALY");          countries.add("OTHER");          return countries;      }        /\*       \* Method used to populate the subjects list in view.       \* Note that here you can call external systems to provide real data.       \*/      @ModelAttribute("subjects")      public List<String> initializeSubjects() {            List<String> subjects = new ArrayList<String>();          subjects.add("Physics");          subjects.add("Chemistry");          subjects.add("Life Science");          subjects.add("Political Science");          subjects.add("Computer Science");          subjects.add("Mathmatics");          return subjects;      }    } |
|  | @Controller indicates that this class is a controller handling the requests with pattern mapped by @RequestMapping.Here with ‘/’, it is serving as default controller. Method newRegistration is fairly simple, annotated with @RequestMethod.GET serving default GET requests, adding the model object to serve as data-holder of form , and presenting the page containing the blank form. |
|  | Method initializeSections, initializeCountries & initializeSubjects are simply creating request level objects whose values will be used in view/jsp. |
|  | Method saveRegistration is annotated with @RequestMethod.POST, and will handle the form-submission POST requests.Notice the parameters and their orders in this method. @Valid asks spring to validate the associated object(student). BindingResult contains the outcome of this validation and any error that might have occurred during this validation. Notice that BindingResult must come right after the validated object else spring won’t be able to validate and an exception been thrown. |
| 5. | Add Configuration Class |
|  | package keyur;    import org.springframework.context.MessageSource;  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.ViewResolver;  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;    @Configuration  @EnableWebMvc  @ComponentScan(basePackages = "com.websystique.springmvc")  public class HelloWorldConfiguration extends WebMvcConfigurerAdapter {        /\*       \* Configure View Resolver       \*/      @Bean      public ViewResolver viewResolver() {          InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();          viewResolver.setViewClass(JstlView.class);          viewResolver.setPrefix("/WEB-INF/views/");          viewResolver.setSuffix(".jsp");          return viewResolver;      }        /\*       \* Configure ResourceHandlers to serve static resources like CSS/ Javascript etc...       \*       \*/      @Override      public void addResourceHandlers(ResourceHandlerRegistry registry) {          registry.addResourceHandler("/static/\*\*").addResourceLocations("/static/");      }        /\*       \* Configure MessageSource to provide internationalized messages       \*       \*/        @Bean      public MessageSource messageSource() {          ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();          messageSource.setBasename("messages");          return messageSource;      }    } |
|  | @Configuration indicates that this class contains one or more bean methods annotated with @Beanproducing bean manageable by spring container.@EnableWebMvc is equivalent to mvc:annotation-driven in XML. It enables support for @Controller-annotated classes that use @RequestMapping to map incoming requests to specific method. @ComponentScan is equivalent to context:component-scan base-package="..." providing with where to look for spring managed beans/classes. |
|  | Method viewResolver configures a view resolver to identify the real view. Method addResourceHandlers configures the ResourceHandler for static resources. CSS, JavaScript, images etc are static resources your pages needs.Above configuration says that all resource requests starting with /static/ will be served from /static/ folder under webapp. In this example, we will put all the css files under /static/css inside webapp directory. |
|  | Method messageSource configures a Message bundle to support [internationalized] messages from properties file. Notice the parameter provided (messages) to baseName method. Spring will search for a file named messages.properties in application class path. Let’s add that file: |
|  | src/main/resources/messages.properties |
|  | Size.student.firstName=First Name must be between {2} and {1} characters long  Size.student.lastName=Last Name must be between {2} and {1} characters long  NotEmpty.student.sex=Please specify your gender  NotNull.student.dob=Date of birth can not be blank  Past.student.dob=Date of birth must be in the past  Email.student.email=Please provide a valid Email address  NotEmpty.student.email=Email can not be blank  NotEmpty.student.country=Please select your country  NotEmpty.student.section=Please select your section  NotEmpty.student.subjects=Please select at least one subject  typeMismatch=Invalid format |
|  | Notice that above message follows a specific pattern |
|  | {ValidationAnnotationClass}.{modelObject}.{fieldName} |
|  | Above configuration in XML format will be |
|  | <beans xmlns="<a class="vglnk" href="<http://www.springframework.org/schema/beans>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>beans</span></a>"      xmlns:context="<a class="vglnk" href="<http://www.springframework.org/schema/context>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>context</span></a>"      xmlns:xsi="<a class="vglnk" href="<http://www.w3.org/2001/XMLSchema-instance>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>w3</span><span>.</span><span>org</span><span>/</span><span>2001</span><span>/</span><span>XMLSchema</span><span>-</span><span>instance</span></a>"      xmlns:mvc="<a class="vglnk" href="<http://www.springframework.org/schema/mvc>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>mvc</span></a>"      xsi:schemaLocation="<a class="vglnk" href="<http://www.springframework.org/schema/beans>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>beans</span></a> <a class="vglnk" href="<http://www.springframework.org/schema/beans/spring-beans-4.0.xsd>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>beans</span><span>/</span><span>spring</span><span>-</span><span>beans</span><span>-</span><span>4</span><span>.</span><span>0</span><span>.</span><span>xsd</span></a>      <a class="vglnk" href="<http://www.springframework.org/schema/mvc>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>mvc</span></a> <a class="vglnk" href="<http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>mvc</span><span>/</span><span>spring</span><span>-</span><span>mvc</span><span>-</span><span>4</span><span>.</span><span>0</span><span>.</span><span>xsd</span></a>      <a class="vglnk" href="<http://www.springframework.org/schema/context>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>context</span></a> <a class="vglnk" href="<http://www.springframework.org/schema/context/spring-context-4.0.xsd>" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>schema</span><span>/</span><span>context</span><span>/</span><span>spring</span><span>-</span><span>context</span><span>-</span><span>4</span><span>.</span><span>0</span><span>.</span><span>xsd</span></a>">        <context:component-scan base-package="com.websystique.springmvc" />      <mvc:annotation-driven/>        <mvc:resources mapping="/static/\*\*" location="/static/" />      <mvc:default-servlet-handler />          <bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">          <property name="basename">              <value>messages</value>          </property>      </bean>          <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">          <property name="prefix">              <value>/WEB-INF/views/</value>          </property>          <property name="suffix">              <value>.jsp</value>          </property>      </bean>    </beans> |
| 6. | Add Views (Simple JSP Pages) |
|  | We will add two simple jsp pages. First one will contain a Form to receive input from user, and second one will show the success message to user once form input is validated successfully.  Below is the peace of code used to include static resources(bootstrap.css in our case) |
|  | <link href="<c:url value='/static/css/bootstrap.css' />" rel="stylesheet"></link> |
|  | Notice the path to static resource. Since we have already configured ResourceHandlers in previous Step with /static/\*\*, css file will be searched inside /static/ folder.  Complete JSP Files are shown below : |
|  | WEB-INF/views/enroll.jsp |
|  | |  | | --- | | <%@ page language="java" contentType="text/html; charset=ISO-8859-1"      pageEncoding="ISO-8859-1"%>  <%@ taglib prefix="form" uri="<a class="vglnk" href="[http://www.springframework.org/tags/form"%](http://www.springframework.org/tags/form%22%25)" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>springframework</span><span>.</span><span>org</span><span>/</span><span>tags</span><span>/</span><span>form</span><span>"%</span></a>>  <%@ taglib prefix="c" uri="<a class="vglnk" href="<http://java.sun.com/jsp/jstl/core>" rel="nofollow"><span>http</span><span>://</span><span>java</span><span>.</span><span>sun</span><span>.</span><span>com</span><span>/</span><span>jsp</span><span>/</span><span>jstl</span><span>/</span><span>core</span></a>" %>    <html>    <head>      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">      <title>Student Enrollment Form</title>      <link href="<c:url value='/static/css/bootstrap.css' />" rel="stylesheet"></link>      <link href="<c:url value='/static/css/custom.css' />" rel="stylesheet"></link>  </head>    <body>        <div class="form-container">        <h1>Enrollment Form</h1>        <form:form method="POST" modelAttribute="student" class="form-horizontal">            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="firstName">First Name</label>                  <div class="col-md-7">                      <form:input type="text" path="firstName" id="firstName" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="firstName" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="lastName">Last Name</label>                  <div class="col-md-7">                      <form:input type="text" path="lastName" id="lastName" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="lastName" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="sex">Sex</label>                  <div class="col-md-7" class="form-control input-sm">                      <form:radiobutton path="sex" value="M" />Male                      <form:radiobutton path="sex" value="F" />Female                      <div class="has-error">                          <form:errors path="sex" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="dob">Date of birth</label>                  <div class="col-md-7">                      <form:input type="text" path="dob" id="dob" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="dob" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="email">Email</label>                  <div class="col-md-7">                      <form:input type="text" path="email" id="email" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="email" class="help-inline"/>                      </div>                  </div>              </div>          </div>              <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="section">Section</label>                  <div class="col-md-7" class="form-control input-sm">                      <form:radiobuttons path="section" items="${sections}" />                      <div class="has-error">                          <form:errors path="section" class="help-inline"/>                      </div>                  </div>              </div>          </div>              <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="country">Country</label>                  <div class="col-md-7">                      <form:select path="country" id="country" class="form-control input-sm">                          <form:option value="">Select Country</form:option>                          <form:options items="${countries}" />                      </form:select>                      <div class="has-error">                          <form:errors path="country" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="firstAttempt">First Attempt ?</label>                  <div class="col-md-1">                      <form:checkbox path="firstAttempt" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="firstAttempt" class="help-inline"/>                      </div>                  </div>              </div>          </div>              <div class="row">              <div class="form-group col-md-12">                  <label class="col-md-3 control-lable" for="subjects">Subjects</label>                  <div class="col-md-7">                      <form:select path="subjects" items="${subjects}" multiple="true" class="form-control input-sm"/>                      <div class="has-error">                          <form:errors path="subjects" class="help-inline"/>                      </div>                  </div>              </div>          </div>            <div class="row">              <div class="form-actions floatRight">                  <input type="submit" value="Register" class="btn btn-primary btn-sm">              </div>          </div>      </form:form>      </div>  </body>  </html> |   WEB-INF/views/success.jsp   |  | | --- | | <%@ page language="java" contentType="text/html; charset=ISO-8859-1"      pageEncoding="ISO-8859-1"%>  <%@ taglib prefix="c" uri="<a class="vglnk" href="<http://java.sun.com/jsp/jstl/core>" rel="nofollow"><span>http</span><span>://</span><span>java</span><span>.</span><span>sun</span><span>.</span><span>com</span><span>/</span><span>jsp</span><span>/</span><span>jstl</span><span>/</span><span>core</span></a>" %>  <html>  <head>      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">      <title>Student Enrollment Detail Confirmation</title>      <link href="<c:url value='/static/css/custom.css' />" rel="stylesheet"></link>  </head>  <body>      <div class="success">          Confirmation message : ${success}          <br>          We have also sent you a confirmation mail to your email address : ${student.email}.      </div>  </body>  </html> | |  | |
| **7.** | **Add Initializer class** |
|  | package keyur;    import javax.servlet.ServletContext;  import javax.servlet.ServletException;  import javax.servlet.ServletRegistration;    import org.springframework.web.WebApplicationInitializer;  import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;  import org.springframework.web.servlet.DispatcherServlet;    public class HelloWorldInitializer implements WebApplicationInitializer {        public void onStartup(ServletContext container) throws ServletException {            AnnotationConfigWebApplicationContext ctx = new AnnotationConfigWebApplicationContext();          ctx.register(HelloWorldConfiguration.class);          ctx.setServletContext(container);            ServletRegistration.Dynamic servlet = container.addServlet(                  "dispatcher", new DispatcherServlet(ctx));            servlet.setLoadOnStartup(1);          servlet.addMapping("/");      }    } |
|  | By extending AbstractAnnotationConfigDispatcherServletInitializer base class, as shown below: |
|  | package com.websystique.springmvc.configuration;    import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;    public class HelloWorldInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {        @Override      protected Class<?>[] getRootConfigClasses() {          return new Class[] { HelloWorldConfiguration.class };      }        @Override      protected Class<?>[] getServletConfigClasses() {          return null;      }        @Override      protected String[] getServletMappings() {          return new String[] { "/" };      }    } |
| **8.** | Build and Deploy the application |
|  | One thing to keep in mind that the Spring java based configuration api’s like WebApplicationInitializer depends on Servlet 3.0 containers.So make sure you don’t have any web.xml with servlet declaration less than 3.0. For our case, we have removed web.xml file from our application. |
|  | Now build the war (either by eclipse as was mentioned in last tutorial) or via maven command line( mvn clean install). Deploy the war to a Servlet 3.0 container . Since here i am using Tomcat, i will simply put this war file into tomcat webapps folder and click on start.bat inside tomcat bin directory. |
|  | Run the application |
|  | You get the initial page like below : |
|  | 3.jpg |
|  | Now if you try to submit, you will get validation errors (with our user-defined message in message.properties) |
|  | Now Provide your sample inputs |
|  | Submit the form now: |
|  | That’s it. |