Spring MVC Validation

The Spring MVC Validation is used to restrict the input provided by the user. To validate the user's input, the Spring 4 or higher version supports and use Bean Validation API. It can validate both server-side as well as client-side applications.

Bean Validation API

The Bean Validation API is a Java specification which is used to apply constraints on object model via annotations. Here, we can validate a length, number, regular expression, etc. Apart from that, we can also provide custom validations.

As Bean Validation API is just a specification, it requires an implementation. So, for that, it uses Hibernate Validator. The Hibernate Validator is a fully compliant JSR-303/309 implementation that allows to express and validate application constraints.

Validation Annotations

Let's see some frequently used validation annotations.

Annotation	Description
@NotNull	It determines that the value can't be null.
@Min	It determines that the number must be equal or greater than the specified value.
@Max	It determines that the number must be equal or less than the specified value.
@Size	It determines that the size must be equal to the specified value.
@Pattern	It determines that the sequence follows the specified regular expression.

Spring MVC Validation Example

In this example, we create a simple form that contains the input fields. Here, (*) me
field. Otherwise, the form generates an error.

1. Add dependencies to pom.xml file.

pom.xml

```xml
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.1.1.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.tomcat/tomcat-jasper -->
<dependency>
    <groupId>org.apache.tomcat</groupId>
    <artifactId>tomcat-jasper</artifactId>
    <version>9.0.12</version>
</dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>3.0-alpha-1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
 <!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
<dependency>
    <groupId>org.hibernate.validator</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>
```

2. Create the bean class

Employee.java

```java
package com.javatpoint;
import javax.validation.constraints.Size;

public class Employee {

    private String name;
    @Size(min=1,message="required")
    private String pass;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPass() {
        return pass;
    }
    public void setPass(String pass) {
        this.pass = pass;
    }
}
```

3. Create the controller class

In controller class:

- The **@Valid** annotation applies validation rules on the provided object.
- The **BindingResult** interface contains the result of validation.

```
package com.javatpoint;

import javax.validation.Valid;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class EmployeeController {

    @RequestMapping("/hello")
    public String display(Model m)
    {
        m.addAttribute("emp", new Employee());
        return "viewpage";
    }
    @RequestMapping("/helloagain")
    public String submitForm( @Valid @ModelAttribute("emp") Employee e, Bindin
    {
        if(br.hasErrors())
        {
            return "viewpage";
        }
        else
        {
        return "final";
        }
    }
}
```

4. Provide the entry of controller in the web.xml file

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"                    xmlns="http://java.sun.com/xm
/javaee"                    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee                    http://java.sun.com/xml/ns/javaee/w
app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>SpringMVC</display-name>
   <servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```

5. Define the bean in the xml file

spring-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    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/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!-- Provide support for component scanning -->
    <context:component-scan base-package="com.javatpoint" />
    <!--Provide support for conversion, formatting and validation -->
    <mvc:annotation-driven/>
    <!-- Define Spring MVC view resolver -->
     <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
     </bean>
</beans>
```

6. Create the requested page

index.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<body>
<a href="hello">Click here...</a>
</body>
</html>
```

7. Create the other view components

viewpage.jsp

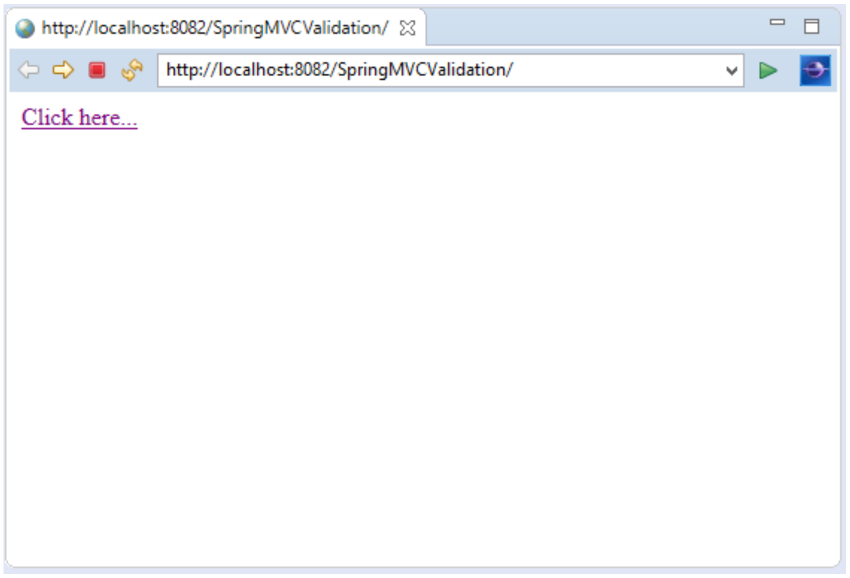
```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<head>
<style>
.error{color:red}
</style>
</head>
<body>
<form:form action="helloagain" modelAttribute="emp">
Username: <form:input path="name"/> <br><br>
Password(*): <form:password path="pass"/>
```

```
<form:errors path="pass" cssClass="error"/><br><br>
<input type="submit" value="submit">
</form:form>
</body>
</html>
```

final.jsp

```
<html>
<body>
Username: ${emp.name} <br><br>
Password: ${emp.pass}
</body>
</html>
```

Output:


http://localhost:8082/SpringMVCValidation/
http://localhost:8082/SpringMVCValidation/

Click here...

Let's submit the form without entering the password.