



CS544 EA

Applications

Validation: Spring MVC Integration

Pom.xml

- If SpringMVC detects a validator implementation on the classpath
 - It automatically integrates with it

```
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate-validator</artifactId>  
  <version>6.0.9.Final</version>  
</dependency>
```

WebAppInitializer

Regular Initializer
nothing extra needed
for validation

```
public class MyWebAppInitializer implements WebApplicationInitializer {  
  
    @Override  
    public void onStartUp(ServletContext container) throws ServletException {  
        // Create the 'root' Spring application context  
        AnnotationConfigWebApplicationContext rootContext =  
            new AnnotationConfigWebApplicationContext();  
        rootContext.register(WebConfig.class);  
        container.addListener(new ContextLoaderListener(rootContext));  
  
        // Create the dispatcher servlet  
        ServletRegistration.Dynamic appServlet = container.addServlet("mvc",  
            new DispatcherServlet(new GenericWebApplicationContext()));  
        appServlet.setLoadOnStartup(1);  
        appServlet.addMapping("/");  
    }  
}
```

@Configuration

```
@Configuration
@EnableWebMvc
@ComponentScan("cs544")
public class WebConfig implements WebMvcConfigurer{
    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundleMessageSource();
        messageSource.setBasename("i18/errormsg");
        return messageSource;
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver bean = new InternalResourceViewResolver();

        bean.setViewClass(JstlView.class);
        bean.setPrefix("/WEB-INF/view/");
        bean.setSuffix(".jsp");

        return bean;
    }
}
```

Optional bean for
custom validation messages

XML version

```
<bean id="messageSource1"
      class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basename" value="messages" />
</bean>
```

Car Class

With validation annotations

@Entity

```
public class Car {
```

```
    @Id
```

```
    @GeneratedValue
```

```
    private int id;
```

```
    @NotEmpty
```

```
    private String make;
```

```
    @NotEmpty
```

```
    private String model;
```

```
    @Range(min = 1940, max = 2015)
```

```
    private int year;
```

```
    private String color;
```

Controller

```
@Controller
public class CarController {
    @Resource
    private CarService carService;

    @GetMapping("/addCar")
    public String addCar(@ModelAttribute("car") Car car) {
        return "addCar";
    }

    @PostMapping("/addCar")
    public String add(@Valid Car car, BindingResult result, RedirectAttributes attr){
        if (result.hasErrors()) {
            attr.addFlashAttribute("org.springframework.validation.BindingResult.car", result);
            attr.addFlashAttribute("car", car);
            return "redirect:/addCar";
        } else {
            carService.add(car);
            return "redirect:/cars";
        }
    }
}
```

ModelAttribute needed
because view has to have
an object to pull from

Order of args important!
BindingResult has to be
right after @Valid arg

Package name is needed for the
binding result to show after redirecting

View

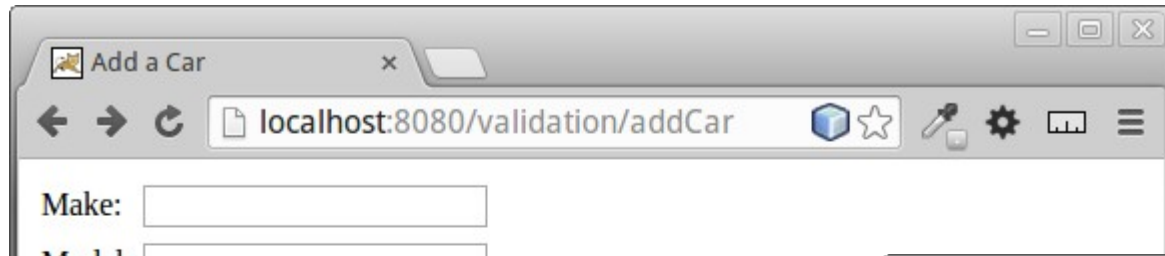
```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Add a Car</title>
    <link href="resources/style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <form:form modelAttribute="car" action="addCar" method="post">
      <form:errors path="*" cssClass="errorblock" element="div" />
      <table>
        <tr>
          <td>Make:</td>
          <td><form:input path="make" /> </td>
          <td><form:errors path="make" cssClass="error" /> </td>
        </tr>
        <!-- Model year and color removed to keep the slide shorter -->
      </table>
      <input type="submit"/>
    </form:form>
  </body>
</html>
```

ModelAttribute or
CommandName refers
to the bean from which
data should be used

Path attribute specifies
which field on that bean

Spring form error tags
display validation errors
for their fields

Result



Add a Car

localhost:8080/validation/addCar

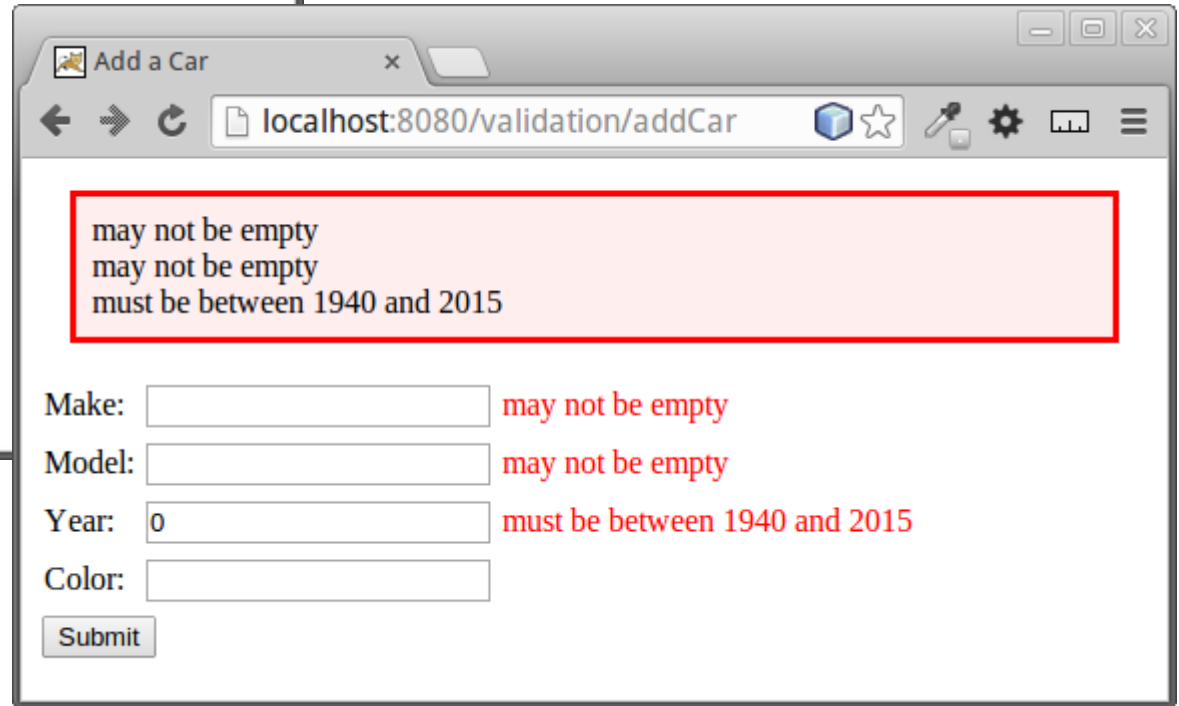
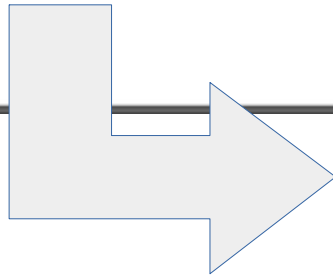
Make:

Model:

Year:

Color:

Submit



Add a Car

localhost:8080/validation/addCar

may not be empty
may not be empty
must be between 1940 and 2015

Make: may not be empty

Model: may not be empty

Year: must be between 1940 and 2015

Color:

Submit

Tag	Description
<form:form>	Creates an HTML form, that can also hold Spring Form Tags
<form:errors>	path attribute can specify which field, shows all without
<form:input>	Wrapper for HTML element
<form:password>	Wrapper for HTML element
<form:hidden>	Wrapper for HTML element
<form:textarea>	Wrapper for HTML element
<form:select>	Wrapper for HTML element
<form:option>	Wrapper for HTML element
<form:options>	Creates multiple option elements from a list
<form:checkbox>	Wrapper for HTML element
<form:checkboxes>	Creates multiple checkboxes from a list
<form:radiobutton>	Wrapper for HTML element
<form:radiobuttons>	Creates multiple radio buttons from a list
<form:label>	Wrapper for HTML element

Spring Form Tags