Spring MVC Tutorial – Contact List Application

Step 19: Custom Error Pages

Step 19: Custom Error Pages

Overview

Let’s face it — the error pages that come by default with Tomcat are pretty ugly. Worse than the bad aesthetics, though, is the fact that the error messages that are displayed by these default pages can give away details of your implementation that are better left unshared. In this step, we will add custom error pages to our application. This allows us to determine the look and feel of the error pages and tightly control the information that gets displayed to the screen.

We’ll accomplish this in three steps:

1. Configure Tomcat to use custom error pages
2. Add a custom controller component to handle errors
3. Add a custom view to display error messages

1. Configuring Tomcat

The first step in implementing custom error pages is to configure Tomcat to use our error pages instead of its default error pages. This configuration is in the **web.xml** file and is straightforward — we simply have to add an error-page entry that points to our custom error controller endpoint. Please add the following entry to your **web.xml** file:

Notes:

1. /error is the Controller endpoint to which we would like to direct all errors

```
<error-page>
    <location>/error</location>
</error-page>
```

2. Creating the Custom Error Controller

The next step in implementing custom error pages is to create a controller that will handle all errors. We must configure the controller so that it handles all requests to the error page location that we configured in the **web.xml** file. In our case, the error page location is **/error**.

Please create a new controller and enter the following code:

Notes:

1. Map this method to the /error endpoint
2. Note that we are use the Spring Model object rather than a Map to store the model information that will be sent to the view component
3. Get useful information about the request and the error message:
    a. HTTP Status Code and HttpStatus object
    b. Any Throwable or Exception information
    c. Error message
4. Construct and format the error message
5. Put the error message in the model object so it can be used by the view

```java
package com.swcguild.contactlistmvc.controller;

import java.text.MessageFormat;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
class ErrorController {

    // #1 - map this end point to /error
    @RequestMapping(value="/error")
    // #2 - note the use of the Spring Model object rather than a Map
    public String customError(HttpServletRequest request,
                              HttpServletResponse  response,
                              Model model) {
        // #3 - retrieve some useful information from the request
        Integer statusCode =
                (Integer) request.getAttribute("javax.servlet.error.status_code");
        HttpStatus httpStatus = HttpStatus.valueOf(statusCode);
        Throwable throwable =
                  (Throwable) request.getAttribute("javax.servlet.error.exception");
        String exceptionMessage = null;
        exceptionMessage = httpStatus.getReasonPhrase();

        String requestUri =
               (String) request.getAttribute("javax.servlet.error.request_uri");
        if (requestUri == null) {
            requestUri = "Unknown";
        }

        // #4 - format the message for the view
        String message = MessageFormat.format("{0} returned for {1}: {2}",
                statusCode, requestUri, exceptionMessage);

        // #5 - put the message in the model object
        model.addAttribute("errorMessage", message);
        return "customError";
    }
}
```

3. Creating the Custom Error View

The final step in implementing custom error pages is to create a view component (JSP) to display the error messages from the error controller. All we do is simply display the error message (in the errorMessage model attribute) to the screen. Please create a new JSP called **customError.jsp** (in your jsp folder) and put the following code in it:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="s" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Company Contacts</title>
        <!-- Bootstrap core CSS -->
        <link href="${pageContext.request.contextPath}/css/bootstrap.min.css" rel="stylesheet">

        <!-- SWC Icon -->
        <link rel="shortcut icon" href="${pageContext.request.contextPath}/img/icon.png">

    </head>
    <body>
        <div class="container">
            <h1>Company Contacts</h1>
            <hr/>
            <div class="navbar">
                <ul class="nav nav-tabs">
                    <li role="presentation" class="active">
                        <a href="${pageContext.request.contextPath}/home">Home</a>
                    </li>
                </ul>
            </div>
            <div>
                <h1>An error has occurred...</h1>

                <h3>${errorMessage}</h3>
            </div>
            <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
            <script src="${pageContext.request.contextPath}/js/jquery-1.11.1.min.js"></script>
            <script src="${pageContext.request.contextPath}/js/bootstrap.js"></script>
    </body>
</html>
```

Wrap-up

In this step, you have learned how to do the following:

1. Configure Tomcat to use custom error pages instead of its default error pages
2. Implement and configure a controller component to create an error message based on what went wrong and to put the error message on the model for use by the view
3. Implement a view component (JSP) to display the error message created by the controller