自：https://blog.csdn.net/u010657094/article/details/78816566

**spring boot 错误处理**

2017年12月15日 21:02:58 [回归心灵](https://me.csdn.net/u010657094) 阅读数：1395

本文是Spring boot官方文档[*Error Handling*](https://docs.spring.io/spring-boot/docs/2.0.0.M7/reference/htmlsingle/#boot-features-error-handling)部分的译文

另一篇博客更详细的讲解了其内在原理及应用。(<http://blog.csdn.net/u010657094/article/details/78993929>)

### [27.1.9 Error Handling](https://docs.spring.io/spring-boot/docs/2.0.0.M7/reference/htmlsingle/" \l "boot-features-error-handling" \t "_blank)

Spring Boot provides an /error mapping by default that handles all errors in a sensible way, and it is registered as a ‘global’ error page in the servlet container. For machine clients it will produce a JSON response with details of the error, the HTTP status and the exception message. For browser clients there is a ‘whitelabel’ error view that renders the same data in HTML format (to customize it just add a View that resolves to ‘error’). To replace the default behaviour completely you can implement ErrorController and register a bean definition of that type, or simply add a bean of type ErrorAttributes to use the existing mechanism but replace the contents.

Spring Boot 以一个合理的方式提供了默认的映射 /error ，用于处理所有错误。并且它被作为一个全局的错误页注册在servlet容器中。对于机器客户端，它将产生一个带有详细错误信息的JSON响应，HTTP状态和异常信息。对于浏览器客户端，将会返回一个白色的错误标签页，同样的错误信息以HTML格式渲染（想要自定义这个页面，只需添加一个解析“error”的View）。要完全地替换这个默认的行为，你可以实现 ErrorController ，并且注册一个bean定义这个类型，或者简单的增加一个 ErrorAttributes 类型的bean来使用现有的机制，但替换掉内容。

The BasicErrorController can be used as a base class for a custom ErrorController. This is particularly useful if you want to add a handler for a new content type (the default is to handle text/html specifically and provide a fallback for everything else). To do so, extend BasicErrorController, add a public method with a @RequestMapping that has a produces attribute, and create a bean of your new type.

对于自定义的 ErrorController 来说，BasicErrorController能够被作为一个基类使用。如果你想为一个新的内容类型（默认的是处理特定的 text/html 类型和为其他所有的事情提供一个备用方案）添加一个处理器，它是非常有用的。要做到这一点仅仅需要扩展 BasicErrorController 和增加一个带有一个@RequestMapping 的公共方法，该方法有一个 produces 属性，并且创建一个你的新类型的bean。

You can also define a @ControllerAdvice to customize the JSON document to return for a particular controller and/or exception type.

你也可以定义一个 @ControllerAdvice 来为特定的controller或者异常类型返回自定义的JSON文档。

@ControllerAdvice(basePackageClasses = FooController.class)

public class FooControllerAdvice extends ResponseEntityExceptionHandler {

@ExceptionHandler(YourException.class)

@ResponseBody

ResponseEntity<?> handleControllerException(HttpServletRequest request, Throwable ex) {

HttpStatus status = getStatus(request);

return new ResponseEntity<>(new CustomErrorType(status.value(), ex.getMessage()), status);

}

private HttpStatus getStatus(HttpServletRequest request) {

Integer statusCode = (Integer) request.getAttribute("javax.servlet.error.status\_code");

if (statusCode == null) {

return HttpStatus.INTERNAL\_SERVER\_ERROR;

}

return HttpStatus.valueOf(statusCode);

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20

In the preceding example, if YourException is thrown by a controller defined in the same package as FooController, a JSON representation of the CustomErrorType POJO is used instead of the ErrorAttributes representation.

在上面的例子中，如果 YourException 由一个和 FooController 在同一个文件夹下的controller抛出，那么将使用 CustomerErrorType POJO类型的json表示，而不是 ErrorAttributes 的表示。

### [Custom Error Pages](https://docs.spring.io/spring-boot/docs/2.0.0.M7/reference/htmlsingle/" \l "boot-features-error-handling-custom-error-pages" \t "_blank)

If you want to display a custom HTML error page for a given status code, you can add a file to an /error folder. Error pages can either be static HTML (that is, added under any of the static resource folders) or built by using templates. The name of the file should be the exact status code or a series mask.

如果你想为一个给定的状态码显示一个自定义的HTML错误页，那么你可以在 /error 文件夹下添加一个文件。错误页面页可以是一个静态的HTML（也就是，在任何的静态资源文件夹下添加），也可以使用模板。这个文件的名字应该是精确的状态码或者是表示一个系列的模糊名称。

For example, to map 404 to a static HTML file, your folder structure would be as follows:

例如，为了映射 404 状态码到一个静态HTML文件，你的文件夹结构应该像下面这样：

src/

+- main/

+- java/

| + <source code>

+- resources/

+- public/

+- error/

| +- 404.html

+- <other public assets>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

To map all 5xx errors by using a FreeMarker template, your folder structure would be as follows:

为了映射所有的 5XX 错误，使用一个 Freemarker模板， 其结构应该如下：

src/

+- main/

+- java/

| + <source code>

+- resources/

+- templates/

+- error/

| +- 5xx.ftl

+- <other templates>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

For more complex mappings, you can also add beans that implement the ErrorViewResolver interface, as shown in the following example:

对于更多的复杂的映射，你还可以使用实现 ErrorViewResolver 接口添加 beans 的方式。

public class MyErrorViewResolver implements ErrorViewResolver {

@Override

public ModelAndView resolveErrorView(HttpServletRequest request,

HttpStatus status, Map<String, Object> model) {

// Use the request or status to optionally return a ModelAndView

return ...

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

You can also use regular Spring MVC features such as [@ExceptionHandler](https://docs.spring.io/spring/docs/5.0.2.RELEASE/spring-framework-reference/web.html#mvc-exceptionhandlers) methods and [@ControllerAdvice](https://docs.spring.io/spring/docs/5.0.2.RELEASE/spring-framework-reference/web.html#mvc-ann-controller-advice). The ErrorControllerthen picks up any unhandled exceptions.

你也可以使用常规的Spring MVC特性，像 [@ExceptionHandler methods](https://docs.spring.io/spring/docs/5.0.0.RELEASE/spring-framework-reference/web.html#mvc-exceptionhandlers) 和 [@ControllerAdvice](https://docs.spring.io/spring/docs/5.0.0.RELEASE/spring-framework-reference/web.html#mvc-ann-controller-advice)。ErrorController将获得任何未处理的异常。

### [Mapping Error Pages outside of Spring MVC](https://docs.spring.io/spring-boot/docs/2.0.0.M7/reference/htmlsingle/" \l "boot-features-error-handling-mapping-error-pages-without-mvc" \t "_blank)

For applications that do not use Spring MVC, you can use the ErrorPageRegistrar interface to directly register ErrorPages. This abstraction works directly with the underlying embedded servlet container and works even if you do not have a Spring MVC DispatcherServlet.

对于没有使用Spring MVC 的应用程序，你可以使用 ErrorPageRegistrar 接口来直接注册 ErrorPages。这种抽象将直接与内嵌的Servlet容器一起工作，即使你没有Spring MVC 的 DispatcherServlet 也将工作。

@Bean

public ErrorPageRegistrar errorPageRegistrar(){

return new MyErrorPageRegistrar();

}

// ...

private static class MyErrorPageRegistrar implements ErrorPageRegistrar {

@Override

public void registerErrorPages(ErrorPageRegistry registry) {

registry.addErrorPages(new ErrorPage(HttpStatus.BAD\_REQUEST, "/400"));

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

N.B. if you register an ErrorPage with a path that will end up being handled by a Filter (e.g. as is common with some non-Spring web frameworks, like Jersey and Wicket), then the Filter has to be explicitly registered as an ERROR dispatcher, e.g.

注：如果你注册一个最终会由一个Filter处理的路径的ErrorPage（例如：常见的一些非Spring web 框架，像 Jersey 和 Wicket），那么必须将Filter显示的注册为一个 ERROR 调度器。

@Bean

public FilterRegistrationBean myFilter() {

FilterRegistrationBean registration = new FilterRegistrationBean();

registration.setFilter(new MyFilter());

...

registration.setDispatcherTypes(EnumSet.allOf(DispatcherType.class));

return registration;

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

(the default FilterRegistrationBean does not include the ERROR dispatcher type).

（默认的 FilterRegistrationBean 不包括 ERROR 调度器类型）

### [Error Handling on WebSphere Application Server](https://docs.spring.io/spring-boot/docs/2.0.0.M7/reference/htmlsingle/" \l "boot-features-error-handling-websphere" \t "_blank)

When deployed to a servlet container, Spring Boot uses its error page filter to forward a request with an error status to the appropriate error page. The request can only be forwarded to the correct error page if the response has not already been committed. By default, WebSphere Application Server 8.0 and later commits the response upon successful completion of a servlet’s service method. You should disable this behavior by setting com.ibm.ws.webcontainer.invokeFlushAfterService to false.

当部署到一个Servlet容器时，Spring Boot使用其错误页面过滤器转发带有错误状态的请求到合适的错误页。如果response还没有提交，那么请求只能被转发的正确的错误页面。默认的，WebSphere Application Server 8.0及更高版本在成功完成servlet的服务方法后提交响应。你应该禁用这个行为，通过设置 com.ibm.ws.webcontainer.invokeFlushAfterService 为 false。