Servlet 编写过滤器 Servlet Cookie 处理

Servlet 异常处理

当一个 Servlet 抛出一个异常时，Web 容器在使用了 exception-type 元素的 **web.xml** 中搜索与抛出异常类型相匹配的配置。

您必须在 web.xml 中使用 **error-page** 元素来指定对特定**异常** 或 HTTP **状态码** 作出相应的 Servlet 调用。

web.xml 配置

假设，有一个 *ErrorHandler* 的 Servlet 在任何已定义的异常或错误出现时被调用。以下将是在 web.xml 中创建的项。

```
<!-- servlet 定义 -->
<servlet>
        <servlet-name>ErrorHandler</servlet-name>
        <servlet-class>ErrorHandler</servlet-class>
</servlet>
<!-- servlet 映射 -->
<servlet-mapping>
        <servlet-name>ErrorHandler</servlet-name>
        <url-pattern>/ErrorHandler</url-pattern>
</servlet-mapping>

<!-- error-code 相关的错误页面 -->
<error-page>
    <error-code>404</error-code>
    <location>/ErrorHandler</location>
</error-page>
<error-page>
    <error-code>403</error-code>
    <location>/ErrorHandler</location>
</error-page>

<!-- exception-type 相关的错误页面 -->
<error-page>
    <exception-type>
          javax.servlet.ServletException
    </exception-type >
    <location>/ErrorHandler</location>
</error-page>

<error-page>
    <exception-type>java.io.IOException</exception-type >
    <location>/ErrorHandler</location>
</error-page>
```

如果您想对所有的异常有一个通用的错误处理程序，那么应该定义下面的 error-page，而不是为每个异常定义单独的 error-page 元素：

```
<error-page>
    <exception-type>java.lang.Throwable</exception-type >
    <location>/ErrorHandler</location>
</error-page>
```

以下是关于上面的 web.xml 异常处理要注意的点：

- Servlet ErrorHandler 与其他的 Servlet 的定义方式一样，且在 web.xml 中进行配置。
- 如果有错误状态代码出现，不管为 404（Not Found 未找到）或 403（Forbidden 禁止），则会调用 ErrorHandler 的 Servlet。
- 如果 Web 应用程序抛出 *ServletException* 或 *IOException*，那么 Web 容器会调用 ErrorHandler 的 Servlet。
- 您可以定义不同的错误处理程序来处理不同类型的错误或异常。上面的实例是非常通用的，希望您能通过实例理解基本的概念。

请求属性 - 错误/异常

以下是错误处理的 Servlet 可以访问的请求属性列表，用来分析错误/异常的性质。

序号	属性 & 描述
1	**javax.servlet.error.status_code** 该属性给出状态码，状态码可被存储，并在存储为 java.lang.Integer 数据类型后可被分析。
2	**javax.servlet.error.exception_type** 该属性给出异常类型的信息，异常类型可被存储，并在存储为 java.lang.Class 数据类型后可被分析。
3	**javax.servlet.error.message** 该属性给出确切错误消息的信息，信息可被存储，并在存储为 java.lang.String 数据类型后可被分析。
4	**javax.servlet.error.request_uri** 该属性给出有关 URL 调用 Servlet 的信息，信息可被存储，并在存储为 java.lang.String 数据类型后可被分析。
5	**javax.servlet.error.exception** 该属性给出异常产生的信息，信息可被存储，并在存储为 java.lang.Throwable 数据类型后可被分析。
6	**javax.servlet.error.servlet_name** 该属性给出 Servlet 的名称，名称可被存储，并在存储为 java.lang.String 数据类型后可被分析。

Servlet 错误处理程序实例

以下是 Servlet 实例，将应对任何您所定义的错误或异常发生时的错误处理程序。

本实例让您对 Servlet 中的异常处理有基本的了解，您可以使用相同的概念编写更复杂的异常处理应用程序：

```
//导入必需的 java 库
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

//扩展 HttpServlet 类
public class ErrorHandler extends HttpServlet {

    // 处理 GET 方法请求的方法
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
 IOException
    {
        Throwable throwable = (Throwable)
        request.getAttribute("javax.servlet.error.exception");
        Integer statusCode = (Integer)
        request.getAttribute("javax.servlet.error.status_code");
        String servletName = (String)
        request.getAttribute("javax.servlet.error.servlet_name");
        if (servletName == null){
            servletName = "Unknown";
        }
        String requestUri = (String)
        request.getAttribute("javax.servlet.error.request_uri");
        if (requestUri == null){
            requestUri = "Unknown";
        }
        // 设置响应内容类型
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();
        String title = "菜鸟教程 Error/Exception 信息";

        String docType = "<!DOCTYPE html>\n";
        out.println(docType +
            "<html>\n" +
             "<head><title>" + title + "</title></head>\n" +
             "<body bgcolor=\"#f0f0f0\">\n");
           out.println("<h1>菜鸟教程异常信息实例演示</h1>");
           if (throwable == null && statusCode == null){
              out.println("<h2>错误信息丢失</h2>");
              out.println("请返回 <a href=\"" +
            response.encodeURL("http://localhost:8080/") +
                "\">主页</a>。");
           }else if (statusCode != null) {
              out.println("错误代码 : " + statusCode);
```

```
         }else{
                 out.println("<h2>错误信息</h2>");
                out.println("Servlet Name : " + servletName +
                                "</br></br>");
                out.println("异常类型 : " +
                                throwable.getClass( ).getName( ) +
                                "</br></br>");
                out.println("请求 URI: " + requestUri +
                                "<br><br>");
                out.println("异常信息: " +
                                    throwable.getMessage( ));
            }
            out.println("</body>");
            out.println("</html>");
     }
     // 处理 POST 方法请求的方法
     public void doPost(HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException {
         doGet(request, response);
     }
}
```

以通常的方式编译 **ErrorHandler.java**，把您的类文件放入<Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes 中。

让我们在 web.xml 文件中添加如下配置来处理异常：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<servlet>
        <servlet-name>ErrorHandler</servlet-name>
        <servlet-class>com.runoob.test.ErrorHandler</servlet-class>
</servlet>
<!-- servlet mappings -->
<servlet-mapping>
        <servlet-name>ErrorHandler</servlet-name>
        <url-pattern>/TomcatTest/ErrorHandler</url-pattern>
</servlet-mapping>
<error-page>
    <error-code>404</error-code>
    <location>/TomcatTest/ErrorHandler</location>
</error-page>
<error-page>
    <exception-type>java.lang.Throwable</exception-type >
    <location>/ErrorHandler</location>
</error-page>
</web-app>
```

现在，尝试使用一个会产生异常的 Servlet，或者输入一个错误的 URL，这将触发 Web 容器调用 **ErrorHandler** 的 Servlet，并显示适当的消息。例如，如果您输入了一个错误的 URL（如：http://localhost:8080/TomcatTest/UnKonwPage），那么它将显示下面的结果：

Servlet 编写过滤器 Servlet Cookie 处理