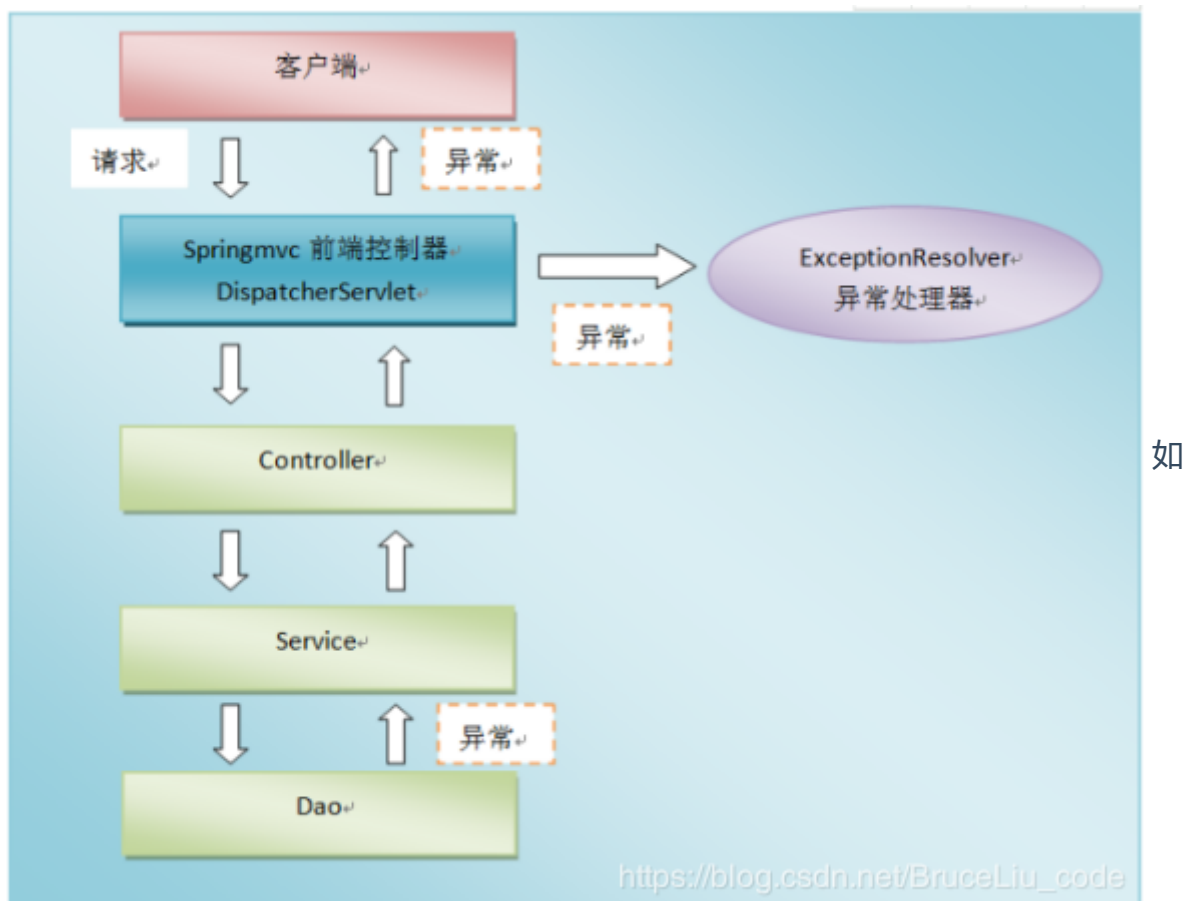


我们知道，系统中异常包括：编译时异常和运行时异常RuntimeException，前者通过捕获异常从而获取异常信息，后者主要通过规范代码开发、测试通过手段减少运行时异常的发生。在开发中，不管是dao层、service层还是controller层，都有可能抛出异常，在springmvc中，能将所有类型的异常处理从各处理过程解耦出来，既保证了相关处理过程的功能较单一，也实现了异常信息的统一处理和维持。

## 1.异常处理思路

在springmvc中，异常处理的思路



上图所示，系统的dao、service、controller出现异常都通过throws Exception向上抛出，最后由springmvc前端控制器交由异常处理器进行异常处理。springmvc提供全局异常处理器（一个系统只有一个异常处理器）进行统一异常处理。明白了springmvc中的异常处理机制，下面就开始分析springmvc中的异常处理。

## 2.异常处理结构体系

Spring MVC通过HandlerExceptionResolver处理程序的异常,包括处理映射,数据绑定及处理器执行时发生异常。HandlerExceptionResolver仅有一个接口方法:

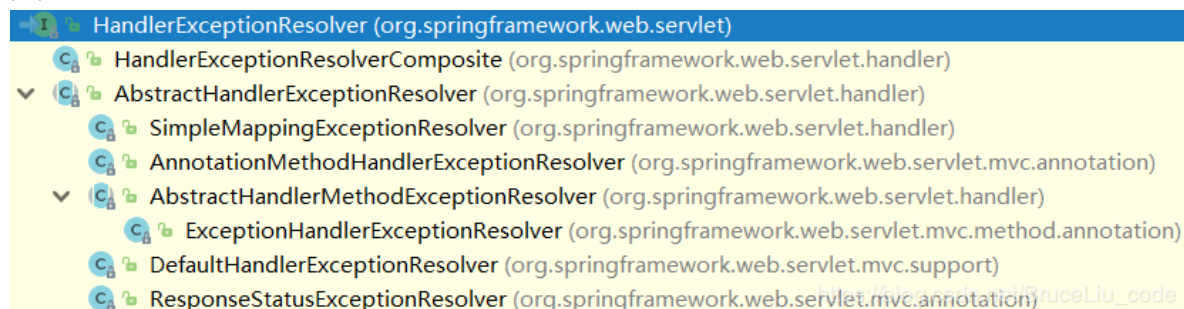
```
1  /**
2   * Interface to be implemented by objects that can resolve
   exceptions thrown during
```

```

3    * handler mapping or execution, in the typical case to error
  views. Implementors are
4    * typically registered as beans in the application context.
5    *
6    * <p>Error views are analogous to JSP error pages but can be
  used with any kind of
7    * exception including any checked exception, with potentially
  fine-grained mappings for
8    * specific handlers.
9    *
10   * @author Juergen Hoeller
11   * @since 22.11.2003
12   */
13  public interface HandlerExceptionResolver {
14
15      /**
16       * Try to resolve the given exception that got thrown during
  handler execution,
17       * returning a {@link ModelAndView} that represents a
  specific error page if appropriate.
18       * <p>The returned {@code ModelAndView} may be {@linkplain
  ModelAndView#isEmpty() isEmpty}
19       * to indicate that the exception has been resolved
  successfully but that no view
20       * should be rendered, for instance by setting a status code.
21       * @param request current HTTP request
22       * @param response current HTTP response
23       * @param handler the executed handler, or {@code null} if
  none chosen at the
24       * time of the exception (for example, if multipart
  resolution failed)
25       * @param ex the exception that got thrown during handler
  execution
26       * @return a corresponding {@code ModelAndView} to forward
  to, or {@code null}
27       * for default processing
28       */
29      ModelAndView resolveException(
30          HttpServletRequest request, HttpServletResponse
  response, Object handler, Exception ex);
31
32  }

```

当发生异常时, Spring MVC将调用 `resolveException()`方法,并转到ModelAndView对应视图中,作为一个异常报告页面,反馈给用户! `HandlerExceptionResolver`拥有4个常见实现类:



## 3.异常处理方案

### 3.1.DefaultHandlerExceptionHandler

Spring MVC默认装配了DefaultHandlerExceptionHandler,它会将Spring MVC框架的异常转换为相应的相应状态码! 异常和相应状态码对应表

异常类型	响应状态码
ConversionNotSupportedException	500(Web服务器内部错误)
HttpMediaTypeNotAcceptableException	406(无和请求accept匹配的MIME类型)
HttpMediaTypeNotSupportedException	415(不支持MIME类型)
HttpMessageNotReadableException	400
HttpMessageNotWritableException	500
HttpRequestMethodNotSupportedException	405
MissingServletRequestParameterException	400

在web.xml响应状态码配置一个对应页面

```
1 <error-page>
2     <error>404</error>
3     <location>/static/404.html</location>
4 </error-page>
```

注意: 静态资源注意会被DispatcherServlet拦截!

### 3.2.SimpleMappingExceptionHandler

如果希望对所有的异常进行统一的处理, 比如当指定的异常发生时, 把它映射到要显示的错误的网页中, 此时用SimpleMappingExceptionHandler进行解析。

DispatcherServlet中没有实现SimpleMappingExceptionHandler的Bean, 所有需要在springmvc的配置文件中配置。

- 示例如下:

```

1  @Controller
2  public class DemoServlet2 {
3
4      @RequestMapping("/testSimpleMappingExceptionHandler")
5      public String testSimpleMappingExceptionHandler() {
6          String[] values = new String[10];
7          // 下标越界了
8          System.out.println(values[11]);
9          return "success";
10     }
11 }

```

发送index.jsp中的超链接请求，控制器捕获请求后处理控制器逻辑，由于在逻辑中，数组越界，会抛出ArrayIndexOutOfBoundsException异常。

- 处理异常

```

1  <!--注解驱动 -->
2  <mvc:annotation-driven />
3  <!-- 配置使用SimpleMappingExceptionHandler来映射异常 -->
4  <bean
5      class="org.springframework.web.servlet.handler.SimpleMappingException
6          tionResolver">
7      <!-- 给异常命名一个别名 -->
8      <property name="exceptionAttribute" value="ex">
9          </property>
10     <property name="exceptionMappings">
11         <props>
12             <!-- 一定要异常的全类名。 表示出现
13             ArrayIndexOutOfBoundsException异常，就跳转到error.jsp视图 -->
14             <prop
15                 key="java.lang.ArrayIndexOutOfBoundsException">error</prop>
16         </props>
17     </property>
18 </bean>

```

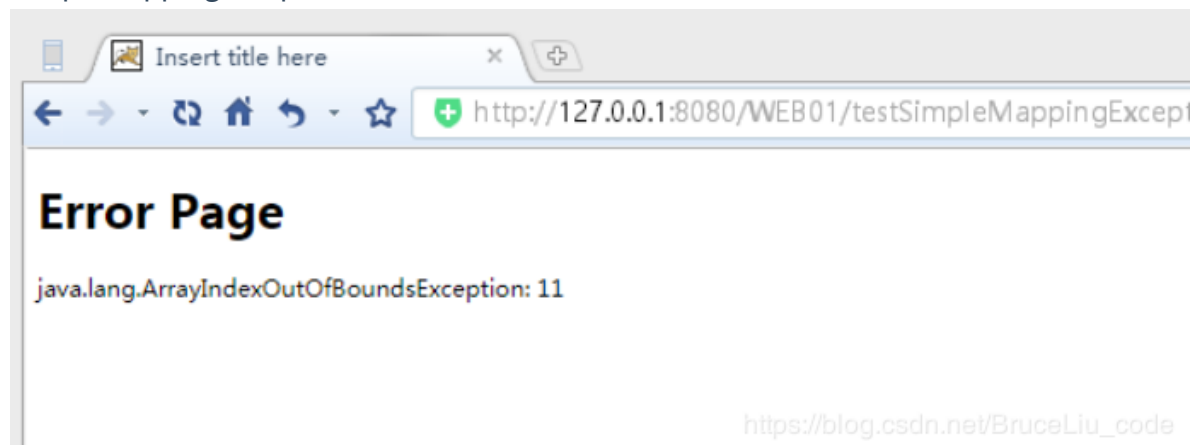
另外在/WEB-INF/jsp下新建一个error.jsp视图。因为上面配置的InternalResourceViewResolver视图解析器默认把error字符串解析为error.jsp视图。error.jsp内容为：

```

1  <%@ page language="java" contentType="text/html; charset=UTF-
   8"pageEncoding="UTF-8"%>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
3  <html>
4  <head>
5  <meta http-equiv="Content-Type" content="text/html; charset=UTF-
   8">
6  <title>Insert title here</title>
7  </head>
8  <body>
9      <h1>Error Page</h1>
10     ${requestScope.ex}
11 </body>
12 </html>

```

下面重新发送index.jsp中的超链接请求后，控制器截获请求并处理请求时，数组越界抛出一个ArrayIndexOutOfBoundsException一个异常，此时由SimpleMappingExceptionHandler异常解析！



### 3.3.AnnotationMethodHandlerExceptionHandlerResolver

Spring MVC 默认注册了 AnnotationMethodHandlerExceptionHandlerResolver,它允许通过 @ExceptionHandler注解指定处理特定异常的方法!

```

1  @Controller
2  public class DemoController1 {
3
4      @ExceptionHandler(value = { RuntimeException.class })
5      public ModelAndView handleArithmeticException2(Exception ex)
6      {
7          System.out.println("[出异常了]:" + ex);
8          ModelAndView mv = new ModelAndView("error");
9          mv.addObject("exception", ex);
10         return mv;
11     }
12 }

```

```

12     @ExceptionHandler(value = { ArithmeticException.class })
13     public ModelAndView handleArithmeticException(Exception ex) {
14         System.out.println("出异常了，算术异常:" + ex);
15         ModelAndView mv = new ModelAndView("error");
16         mv.addObject("exception", ex);
17         return mv;
18     }
19
20     @RequestMapping("/testExceptionHandler1")
21     public String test1() {
22         String s=null;
23         System.out.println(s.length());
24         return "success";
25     }
26
27     @RequestMapping("/testExceptionHandler2")
28     public String test2() {
29         int i=100/0;
30         return "success";
31     }
32 }

```

目标方法内抛出了一个ArithmeticException异常，将由继承关系最近的异常处理捕捉到，即由handleArithmeticException捕捉到。若将handleArithmeticException方法注释掉，则发生ArithmeticException异常将由handleArithmeticException2进行处理。

缺点:

- 使用该注解有一个不好的地方就是：进行异常处理的方法必须与出错的方法在同一个Controller里面。
- 不能全局控制异常。每个类都要写一遍。

### 3.4.全局异常处理

上文说到 @ExceptionHandler 需要进行异常处理的方法必须与出错的方法在同一个Controller里面。那么当代码加入了@ControllerAdvice，则不需要必须在同一个controller中了。这也是Spring 3.2带来的新特性。从名字上可以看出大体意思是控制器增强。也就是说，@controlleradvice + @ExceptionHandler 也可以实现全局的异常捕捉。请确保此WebExceptionHandler类能被扫描到并装载进Spring容器中。

```

1  @Controller
2  @ControllerAdvice
3  public class WebExceptionHandler {
4
5      @ExceptionHandler(Exception.class)
6      public ModelAndView handleException(Exception ex) {
7          System.out.println("全局异常:ex = " + ex);

```

```
8         ModelAndView modelAndView = new ModelAndView();
9
10        modelAndView.setViewName("error");
11        modelAndView.addObject("exception", ex);
12        return modelAndView;
13    }
14 }
```

此处可以捕捉全局异常,但是不要忘了在spring配置的时候扫描该类!

若在其他由@Controller标记的Handler类中的Handle方法抛出异常,且没有在Handler类中定义@ExceptionHandler方法,则会去由@ControllerAdvice标记的类中去找,若也找不到,则在页面抛出异常。