SpringMVC的HandlerInterceptor处理器拦截器

# SpringMVC的Interceptor拦截器

SpringMVC中的Interceptor拦截器相当重要，也是相当有用的。

## 主要作用：拦截用户的请求并进行相应的处理。

## 主要的应用场景：

### 日志记录

### 权限检查

判断用户是否登录等。

### 性能监控

### 通用行为

## 拦截器的实现方法

SpringMVC中的Interceptor拦截器是通过实现Spring提供的HandlerInterceptor接口来实现的：

### Interceptor拦截器需要实现HandlerInterceptor接口；

### Interceptor拦截器继承HandlerInterceptor接口的实现类，如SpringMVC已经实现的HandlerInterceptorAdapter类。

# HandlerInterceptor接口

## package

存在于包**org.springframework.web.servlet**中。

Interface HandlerInterceptor

## 功能

拦截器是链式调用。

Workflow interface that allows for customized handler execution chains. Applications can register any number of existing or custom interceptors for certain groups of handlers, to add common preprocessing behavior without needing to modify each handler implementation.

## 三个方法

preHandle、postHandle、afterCompletion

### preHandle：在请求处理之前进行调用。

default boolean **preHandle**(HttpServletRequest request, HttpServletResponse response, java.lang.Object handler)

Intercept the execution of a handler.

返回类型为boolean，当返回false时就不会再执行后续的拦截器或控制器；只有当返回true时，才会继续执行。

### postHandle：请求进行处理之后进行调用（渲染对应视图之前）。

default void postHandle(HttpServletRequest request, HttpServletResponse response, java.lang.Object handler, ModelAndView **modelAndView**)

Intercept the execution of a handler.

### afterCompletion：整个请求结束之后(渲染了对应的视图之后)。

default void afterCompletion(HttpServletRequest request, HttpServletResponse response, java.lang.Object handler, java.lang.Exception ex)

Callback after completion of request processing, that is, after rendering the view.

## SpringMVC中拦截器的配置

在springmvc.xml中通过**标签mvc:interceptors**进行配置。

可以通过bean直接配置拦截所有请求的Interceptor拦截器；对特定请求可以通过标签mvc:interceptor配置特定的拦截器。

拦截器的执行顺序：按照配置拦截器的先后，顺序执行对应拦截器的perHandle方法，逆顺序执行postHandle和afterHandle方法。



# AsyncHandlerInterceptor接口

## package

public interface **AsyncHandlerInterceptor**

extends HandlerInterceptor

## 功能

相比HandlerInterceptor接口多了一个方法： **afterConcurrentHandlingStarted**。

## 特有方法

除了afterCompletion, postHandle, preHandle三个方法外，定义了

default void **afterConcurrentHandlingStarted**(HttpServletRequest request, HttpServletResponse response, java.lang.Object handler)

Called **instead of postHandle and afterCompletion** when the handler is being executed concurrently.

# HandlerInterceptorAdapter拦截器适配器

## package

存在于org.springframework.web.servlet.handler包中。

public **abstract** class **HandlerInterceptorAdapter**

extends java.lang.Object

implements **AsyncHandlerInterceptor**

## 功能

Abstract adapter class for the **AsyncHandlerInterceptor** interface, for simplified implementation of pre-only/post-only interceptors.

HandlerInterceptorAdapter只是对HandlerInterceptor接口的抽象实现(abstract)，作为一个适配器使用。其中preHandler方法始终返回true，其他方法都是空。

## 方法



# 异常处理

## 在J2EE项目中，每个异常若单独处理，工作量大，最好是统一处理异常；

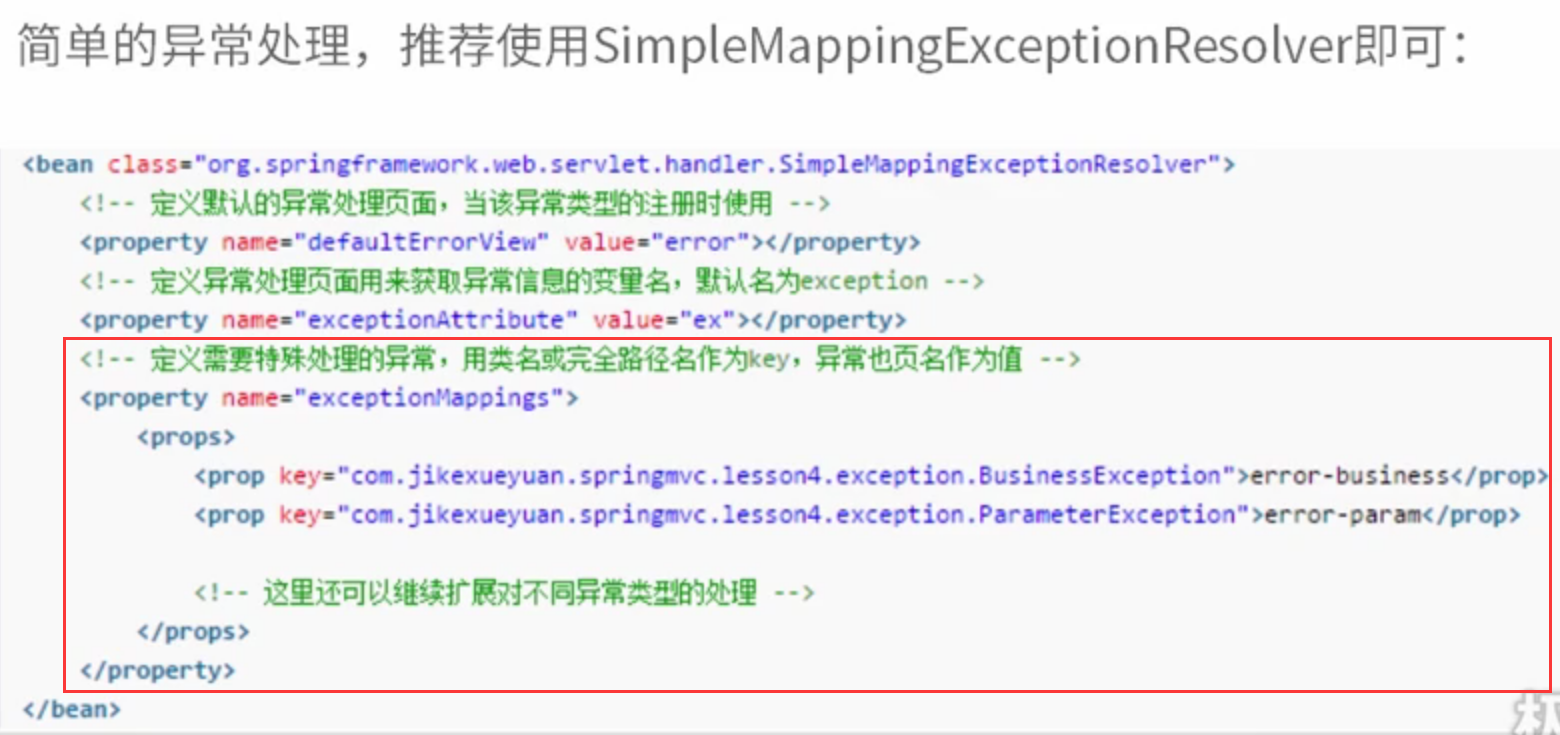
## SpringMVC中的异常处理器

### 使用SpringMVC提供的简单异常处理器SimpleMappingExceptionResolver；

### 自定义异常处理器：实现Spring的异常处理接口HandlerExceptionResolver；

### 使用@ExceptionHandler注解实现异常处理

## SimpleMappingExceptionResolver简单异常处理



## 自定义异常处理器

实现HandlerExceptionResolver接口：

第4个参数表示：对哪种类型的异常进行处理。



在配置文件中，配置该异常处理器：



## 使用@ExceptionHandler注解(推荐使用,零配置)

使用@ExceptionHandler注解实现异常处理：

### 增加BaseController类，并在类中使用@ExceptionHandler注解声明异常处理

### 再让所有需要异常处理的Controller都继承该类。



# 总结

应当掌握：

### 自定义SpringMVC中的拦截器；

### 在项目中做好异常处理逻辑。