导航
博客园
首页
新随笔
管理

< 2021年1月 >

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

springboot配置监听器、过滤器和拦截器

监听器：listener是servlet规范中定义的一种特殊类。用于监听servletContext、HttpSession和servletRequest等域对象的创建和销毁事件。监听域对象的属性发生修改的事件。用于在事件发生前、发生后做一些必要的处理。其主要可用于以下方面：1、统计在线人数和在线用户2、系统启动时加载初始化信息3、统计网站访问量4、记录用户访问路径。

过滤器：Filter是Servlet技术中最实用的技术，Web开发人员通过Filter技术，对web服务器管理的所有web资源：例如Jsp, Servlet, 静态图片文件或静态html 文件等进行拦截，从而实现一些特殊的功能。例如实现URL级别的权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能。它主要用于对用户请求进行预处理，也可以对HttpServletResponse进行后处理。使用Filter的完整流程：Filter对用户请求进行预处理，接着将请求交给Servlet进行处理并生成响应，最后Filter再对服务器响应进行后处理。

拦截器：Interceptor 在AOP（Aspect-Oriented Programming）中用于在某个方法或字段被访问之前，进行拦截然后在之前或之后加入某些操作。比如日志，安全等。一般拦截器方法都是通过动态代理的方式实现。可以通过它来进行权限验证，或者判断用户是否登陆，或者是像12306 判断当前时间是否是购票时间。

三大器在springboot中使用时，首先实现相应的接口定义类，然后通过配置类将其加入到spring容器中，从而实现相应的功能。代码如下：

1、过滤器类

```
package com.example.demo;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;

public class MyFilter implements Filter {

    @Override
    public void doFilter(ServletRequest servletRequest, Servle
        System.out.println(servletRequest.getParameter("name")
        HttpServletRequest hrequest = (HttpServletRequest)serv
        HttpServletResponseWrapper wrapper = new HttpServletRe
        if(hrequest.getRequestURI().indexOf("/index") != -1 ||
                hrequest.getRequestURI().indexOf("/asd") != -1
                hrequest.getRequestURI().indexOf("/online") !=
                hrequest.getRequestURI().indexOf("/login") != -
                ) {
            filterChain.doFilter(servletRequest, servletRespon
```

统计
随笔 - 106
文章 - 0
评论 - 55
引用 - 0

我的标签
设计模式(22)
springboot(19)
java(8)
mybatis(5)
netty(5)
spring(5)
redis(4)
spring data jpa(3)
数据库中间件(3)
rabbitmq(3)
更多

积分与排名
积分 - 146776
排名 - 5383

最新评论
1. Re:springboot+security+JWT实现单点登录

楼主对单点登录是有什么误解吗？文章里的内容只是很清晰的讲解了security，并没有讲到单点登录吧。

--Byron_2015

2. Re:springboot+security+JWT实现单点登录

简单明了，可以根据需要自行DIY的Demo，感谢

--Himura13

3. Re:java实现HTTP请求的三种方式

写的太好了，我要写个小程序，正好用上，感谢

--月饼馅饺子

4. Re:java实现HTTP请求的三种方式

@君临-行者无界 像将对象转为JSON字符串塞到Body中，然后计算一个token塞进请求Header中，拼装url，将返回的Response的JSON字符串解析成Java对象，等等这些工作还是要程序...

--公子骏

5. Re:java实现HTTP请求的三种方式

@君临-行者无界 @公子骏 okhttp不香吗？ okhttp还是比较底层的框架...

--公子骏

阅读排行榜

```
        }else {
            wrapper.sendRedirect("/login");
        }
    }
    @Override
    public void destroy() {
    }
    @Override
    public void init(FilterConfig filterConfig) throws Servlet
    }
}
```

2、监听器类

```
package com.example.demo;

import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

public class MyHttpSessionListener implements HttpSessionListe

    public static int online = 0;
    @Override
    public void sessionCreated(HttpSessionEvent se) {
        System.out.println("创建session");
        online ++;
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        System.out.println("销毁session");

    }



}
```

3、拦截器类

```
package com.example.demo;

import java.io.PrintWriter;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.HandlerMapping;
import org.springframework.web.servlet.ModelAndView;

public class MyInterceptor implements HandlerInterceptor {
    //在请求处理之前进行调用（Controller方法调用之前
```

1. java实现HTTP请求的三种方式(191929)
2. springboot配置监听器、过滤器和拦截器(91551)
3. java解析Excel（xls、xlsx两种格式）(78751)
4. springboot自定义消息转换器HttpMessageConverter(43634)
5. springboot+自定义注解实现灵活的切面配置(41731)

```
    @Override
    public boolean preHandle(HttpServletRequest httpServletReq
        System.out.println("preHandle被调用");
        Map map =(Map)httpServletRequest.getAttribute(HandlerM
        System.out.println(map.get("name"));
        System.out.println(httpServletRequest.getParameter("us
        if(map.get("name").equals("zhangsan")) {
            return true;    //如果false，停止流程，api被拦截
        }else {
            PrintWriter printWriter = httpServletResponse.getW
            printWriter.write("please login again!");
            return false;
        }
    }
    @Override
    public void postHandle(HttpServletRequest httpServletReque
        System.out.println("postHandle被调用");
    }

    @Override
    public void afterCompletion(HttpServletRequest httpServlet
        System.out.println("afterCompletion被调用");
    }
}
```

4、配置类

```
package com.example.demo;

import org.springframework.boot.web.servlet.FilterRegistrationB
import org.springframework.boot.web.servlet.ServletListenerRegi
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.Inter
import org.springframework.web.servlet.config.annotation.ViewC
import org.springframework.web.servlet.config.annotation.WebMv

@Configuration
public class MywebConfig implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry regis
        registry.addViewController("/zxc/foo").setViewName("fo
    }
    @Override
    public void addInterceptors(InterceptorRegistry registry)
        registry.addInterceptor(new MyInterceptor())
                .addPathPatterns("/asd/**");
    }
    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Bean
    public FilterRegistrationBean filterRegist() {
        FilterRegistrationBean frBean = new FilterRegistrationB
        frBean.setFilter(new MyFilter());
        frBean.addUrlPatterns("/*");
        System.out.println("filter");
        return frBean;
```

```
    }
    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Bean
    public ServletListenerRegistrationBean listenerRegist() {
        ServletListenerRegistrationBean srb = new ServletListe
        srb.setListener(new MyHttpSessionListener());
        System.out.println("listener");
        return srb;
    }
}
```

5、控制层

```
package com.example.demo;

import java.util.Date;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class UserController {

    private final Logger logger = LoggerFactory.getLogger(User

    @Value("${application.message:Hello World}")
    private String message ;

    @GetMapping("/asd/{name}")
    public String welcome(@PathVariable String name,Map<String
        model.put("time", new Date());
        model.put("message", this.message);
        return "welcome";
    }

    @RequestMapping("/login")
    @ResponseBody
    public Object foo() {
        logger.info("打印日志----------------------");
        return  "login";
    }

    @RequestMapping("/index")
    @ResponseBody
    public Object index(HttpServletRequest request) {
        HttpSession  session = request.getSession(true);
        session.setAttribute("zxc", "zxc");
```

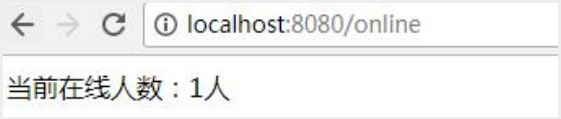
```
        return  "index";
    }

    @RequestMapping("/online")
    @ResponseBody
    public Object online() {
        return  "当前在线人数：" + MyHttpSessionListener.online +
    }
}
```

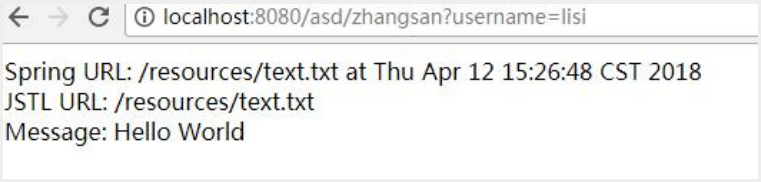
6、程序主入口及application.properties文件默认。

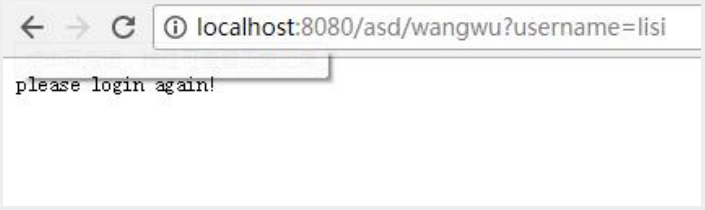
项目启动后，首先测试过滤器，所有访问路径都需经过我们自定义的filter，如果路径中包含/index、/asd、/online、/login则直接通过，否则则被重定向至/login。

然后测试监听器，先访问http://localhost:8080/index,创建session对象（因为session对象创建需要显示的调用getsession方法），然后访问http://localhost:8080/online，然后换一个浏览器做相同操作，如下：



最后测试拦截器，代码展示了如何获取@PathVariable注解的请求参数以及普通请求参数，首先访问http://localhost:8080/asd/zhangsan?username=lisi，响应正常，如果访问路径上的名字不是zhangsan，则被拦截，例如http://localhost:8080/asd/wangwu?username=lisi，如下：





在实际项目中我们还可以在配置的时候设置过滤器、拦截器的执行顺序及其它的参数，同时filter和listener还有对应的注解方式：@WebFilter和@WebListener,在使用注解方式时不要忘了在主程序加上@ServletComponentScan注解，这样才能在程序启动时将对应的bean加载进来。

标签: springboot

君临-行者无界
关注 - 6
粉丝 - 66
+加关注

« 上一篇： Netty构建Http服务器
» 下一篇： springboot实现服务器端消息推送（websocket + sockjs + stomp）

posted on 2018-04-12 15:37 君临-行者无界 阅读(91569) 评论(5) 编辑 收藏

评论

#1楼 2018-05-15 15:54 **HolleJava-Like**

您好，方便发一下您的demo吗?这是我的邮箱17601626176@163.com，您的文章帮到了我，能否发一下demo让我研究一下

支持(0) 反对(0)

#2楼 2019-05-22 10:06 **Kevin_zheng**

get. 博主如果再介绍下 三者的区别更好了。现在 就有点分不清

支持(0) 反对(0)

#3楼 2019-06-13 15:58 **欧阳海瞬**

希望介绍一下 每步 的注释

支持(0) 反对(0)

#4楼 2020-06-15 17:52 **二胖子666**

博主 ，http://localhost:8081/spring-boot-rabbitmq/asd/zhangsan MyInterceptor 为什么会被调用两次 ，第二次拦截不能登录。最后发现 @RestController 用这个注解，@Controller 会把return "welcome"; 当路径跳转请求

支持(0) 反对(0)

#5楼 2020-09-04 10:31 **卯仙**

@二胖子666
他应该还有个welcome界面呢

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

登录后才能发表评论，立即 登录 或 注册， 访问 网站首页