SpringBoot过滤器

过滤器的优先级在于拦截器之上.

****通过过滤器注册配置类使用过滤器****

# SpringBoot[过滤器注册配置类]

|  |
| --- |
| /\*\*  \* 过滤器设置到配置中  \* **@author** 25772  \*  \*/  @Configuration  **public** **class** WebComponent2Config {  @Bean  **public** FilterRegistrationBean someFilterRegistration1() {  //新建过滤器注册类  FilterRegistrationBean registration = **new** FilterRegistrationBean();  // 添加我们写好的过滤器  registration.setFilter(**new** SessionFilter());  // 设置过滤器的URL模式,过滤所有的URL  registration.addUrlPatterns("/\*");  System.***err***.println(registration);  **return** registration;  }  } |

# SpringBoot[过滤器代码实现]

|  |
| --- |
| **public** **class** SessionFilter **implements** Filter {  //标示符：表示当前用户未登录(可根据自己项目需要改为json样式)  String NO\_LOGIN = "您还未登录";  //不需要登录就可以访问的路径(比如:注册登录等)  String[] includeUrls = **new** String[]{"/login","register"};  @Override  **public** **void** doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) **throws** IOException, ServletException {    HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  HttpSession session = request.getSession(**false**);  String uri = request.getRequestURI();  System.***out***.println("filter url:"+uri);  //是否需要过滤  **boolean** needFilter = isNeedFilter(uri);  String requestURI = request.getRequestURI();  //不需要  **if**(requestURI.contains("index")){  System.***err***.println(**true**);  filterChain.doFilter(servletRequest, servletResponse);  //需要.  }**else**{  System.***err***.println(**false**);  response.getWriter().write("未知URL");  }  }  /\*\*  \* **@Author**: xxxxx  \* **@Description**: 是否需要过滤  \* **@Date**: 2018-03-12 13:20:54  \* **@param** uri  \*/  **public** **boolean** isNeedFilter(String uri) {  **for** (String includeUrl : includeUrls) {  **if**(includeUrl.equals(uri)) {  **return** **false**;  }  }  **return** **true**;  }  @Override  **public** **void** init(FilterConfig filterConfig) **throws** ServletException {  System.***err***.println("init");  }  @Override  **public** **void** destroy() {  System.***err***.println("destory");  }  } |