springboot2 配置 cors 实现跨域

作者：laker
原文链接：https://ld246.com/article/1536320146993
来源网站：链滴

网上的资料乱七八糟，大多是老版本的用法，或者是新版本的添加登录filter之后会出现问题，最后结了多篇文章得出该办法。

版本

springboot： 2.0.4.RELEASE

法一

addCorsMappings允许跨域，简单访问没问题，如果用户没登录被filter拦截的话，返回信息得不到显示跨域失败，原因是filter直接拦截的执行在前，允许跨域执行在后，再添加一个FilterRegistrationBean后解决问题，注意FilterRegistrationBean需要设置成在最前面执行。

跨域的设置全部都在CorsConfig类里

跨域配置

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class CorsConfig implements WebMvcConfigurer {
    private Logger logger = LoggerFactory.getLogger(getClass());

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        //设置允许跨域的路径
        registry.addMapping("/**")
                //设置允许跨域请求的域名
                .allowedOrigins("*")
                //是否允许证书 不再默认开启
                .allowCredentials(true)
                //设置允许的方法
                .allowedMethods("*")
                //跨域允许时间
                .maxAge(3600);
    }
    @Bean
    public FilterRegistrationBean corsFilter() {
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration();
        config.setAllowCredentials(true);
```

```
        config.addAllowedOrigin("*");
        config.addAllowedHeader("*");
        config.addAllowedMethod("*");
        source.registerCorsConfiguration("/**", config);
        FilterRegistrationBean bean = new FilterRegistrationBean(new CorsFilter(source));
        bean.setOrder(0);
        return bean;
    }
}
```

登录filter

```
package com.example.crosserver.filter;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.web.servlet.ServletComponentScan;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Component //将LoginFilter交给容器来处理。也就是让LoginFilter起作用
//这个使用来扫描@WebFilter 的让@WebFilter起作用。当然对于servlet线管注解也是可以的。
// 这个@ServletComponentScan最好卸载Apllication这个上面，通用配置。我这里因为只有一个Fil
er所以没有写在Application上面。
@ServletComponentScan
//针对于什么链接做过滤，filter的名称是为什么。
@WebFilter(urlPatterns = "/*", filterName = "loginFilter")
@Order(1)
public class LoginFilter implements Filter {
    private Logger logger = LoggerFactory.getLogger(getClass());
    private List<String> exclude = Arrays.asList("/gotoLogin", "/user/login");
    private List<String> excludeEndWith = Arrays.asList(".js");

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, Filter
hain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
```

```
        HttpServletRequest req = (HttpServletRequest) servletRequest;
        HttpServletResponse resp = (HttpServletResponse) servletResponse;
        String url = req.getRequestURI().toString();
        logger.info("this is MyFilter,url :" + url);
        HttpSession session = req.getSession();
        Map user = (Map) session.getAttribute("user");
        if (user != null) {
            logger.info("已经登录，放行");
            chain.doFilter(request, resp);
        } else {
            boolean pass = false;
            if(exclude.contains(url)){
                pass = true;
            }else{
                for(String temp : excludeEndWith){
                    if(url.endsWith(temp)){
                        pass = true;
                        break;
                    }
                }
            }
            if (pass) {
                logger.info("用户在登录，放行");
                chain.doFilter(request, resp);
            } else {
                logger.info("用户未登录，重定向");
                Map res = new HashMap();
                res.put("msg", "用户未登录");
                resp.setContentType("application/json;charset=UTF-8");
                PrintWriter out = resp.getWriter();
                ObjectMapper mapper = new ObjectMapper();
                String returnStr = mapper.writeValueAsString(res);
                logger.info(returnStr);
                out.print(returnStr);
                out.flush();
                out.close();
            }
        }
    }

    @Override
    public void destroy() {

    }
}
```

法二

跨域的设置一部分在CorsConfig类中，一部分在filter中，因为对于filter中登录失败的需要单独设置许跨域。

```
package com.example.crosserver.util;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class CorsConfig implements WebMvcConfigurer {
    private Logger logger = LoggerFactory.getLogger(getClass());

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        //设置允许跨域的路径
        registry.addMapping("/**")
                //设置允许跨域请求的域名
                .allowedOrigins("*")
                //是否允许证书 不再默认开启
                .allowCredentials(true)
                //设置允许的方法
                .allowedMethods("*")
                //跨域允许时间
                .maxAge(3600);
    }
}
```

filter

注意：

resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_ORIGIN, "*");不生效，把请求中的origin放到这里后可以了。resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_ORIGIN, req.getHeader(HttpHeaders.ORIGIN));

```
package com.example.crosserver.filter;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.web.servlet.ServletComponentScan;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Component //将LoginFilter交给容器来处理。也就是让LoginFilter起作用
//这个使用来扫描@WebFilter 的让@WebFilter起作用。当然对于servlet线管注解也是可以的。
// 这个@ServletComponentScan最好卸载Apllication这个上面，通用配置。我这里因为只有一个Fil
er所以没有写在Application上面。
@ServletComponentScan
//针对于什么链接做过滤，filter的名称是为什么。
@WebFilter(urlPatterns = "/*", filterName = "loginFilter")
@Order(1)
public class LoginFilter implements Filter {
    private Logger logger = LoggerFactory.getLogger(getClass());
    private List<String> exclude = Arrays.asList("/gotoLogin", "/user/login");
    private List<String> excludeEndWith = Arrays.asList(".js");

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, Filter
hain chain) throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletRequest req = (HttpServletRequest) servletRequest;
        HttpServletResponse resp = (HttpServletResponse) servletResponse;
        String url = req.getRequestURI().toString();
        logger.info("this is MyFilter,url :" + url);
        HttpSession session = req.getSession();
        Map user = (Map) session.getAttribute("user");
        if (user != null) {
            logger.info("已经登录，放行");
            chain.doFilter(request, resp);
        } else {
            boolean pass = false;
            if(exclude.contains(url)){
                pass = true;
            }else{
                for(String temp : excludeEndWith){
                    if(url.endsWith(temp)){
                        pass = true;
                        break;
                    }
                }
            }
            if (pass) {
                logger.info("用户在登录，放行");
                chain.doFilter(request, resp);
            } else {
```

```
            //allow_origin设置成*不起作用，但是设置成原来的origin就可以了，这里很奇怪
            resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_ORIGIN, req.getHeader(Ht
pHeaders.ORIGIN));
            resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_HEADERS, "*");
            resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_METHODS, "*");
            resp.addHeader(HttpHeaders.ACCESS_CONTROL_MAX_AGE, "1800");
            resp.addHeader(HttpHeaders.ACCESS_CONTROL_ALLOW_CREDENTIALS, "true");
            logger.info("用户未登录，重定向");
            Map res = new HashMap();
            res.put("msg", "用户未登录");

            resp.setContentType("application/json;charset=UTF-8");
            PrintWriter out = resp.getWriter();
            ObjectMapper mapper = new ObjectMapper();
            String returnStr = mapper.writeValueAsString(res);
            logger.info(returnStr);
            out.print(returnStr);
            out.flush();
            out.close();
        }
    }
}

@Override
public void destroy() {

}
}
```

原文链接：springboot2 配置 cors 实现跨域