**Spring Boot 中处理跨域**

1. **跨域背景**
2. **什么是跨域**

**Url的一般格式：协议 + 域名（子域名 + 主域名） + 端口号 + 资源地址**

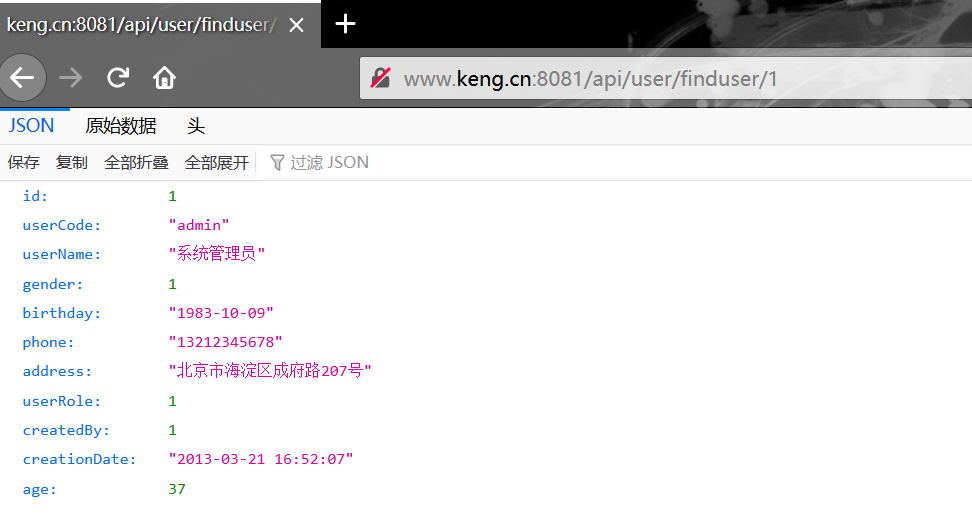
**示例：http://www.keng.cn:8081/say**

**只要协议，子域名，主域名，端口号这四项组成部分中有一项不同，就可以认为是不同的域，不同的域之间互相访问资源，就被称之为跨域。也叫同源策略，同源策略是由Netscape提出的一个著名的安全策略。**

1. **HTML 5中新增的跨域资源访问（Cross-Origin Resource Sharing）特性可以让我们在开发后端系统的时候决定资源是否允许被跨域访问。Spring从4.2版本开始就提供了跨域的支持，开箱即用。这里介绍如何在Spring Boot开发中解决跨域的问题，主要分为注解驱动和接口编程的方式。**
2. **模拟跨域场景**
3. **修改本地hosts文件(C:\Windows\System32\drivers\etc)**



1. **运行demo14项目，输入网址：**[**http://www.keng.cn:8081/api/user/finduser/1**](http://www.keng.cn:8081/api/user/finduser/1)



1. **创建新的Spring Boot项目demo15**
2. **添加thymeleaf依赖**

|  |
| --- |
| **<dependency>**  **<groupId>org.springframework.boot</groupId>**  **<artifactId>spring-boot-starter-thymeleaf</artifactId>**  **</dependency>** |

1. **然后在resources/templates下新建index.html**

|  |
| --- |
| **<!DOCTYPE html>**  **<html lang="en">**  **<head>**  **<meta charset="UTF-8"/>**  **<title>跨域测试</title>**  **<script src="http://libs.baidu.com/jquery/1.11.3/jquery.min.js"></script>**  **</head>**  **<body>**  **<div id="hello"></div>**  **</body>**  **<script>**  **$(function () {**  **$.get("http://www.keng.cn:8081/api/user/finduser/1", function (data) {**  **$("#hello").html(data.userName);**  **})**  **})**  **</script>**  **</html>** |

1. **创建TestController类**

|  |
| --- |
| **@Controller**  **public class TestController {**  **@RequestMapping("index")**  **public String index () {**  **return "index";**  **}**  **}** |

1. **启动demo15，输入访问路径：**[**http://localhost:8080/index**](http://localhost:8080/index)



1. **因为我们在http://localhost:8080/域名下试图访问www.keng.cn:8081下的接口，这就存在跨域问题。**
2. **解决跨域问题**

**CORS全称为Cross Origin Resource Sharing（跨域资源共享）, 每一个页面需要返回一个名为Access-Control-Allow-Origin的http头来允许外域的站点访问，你可以仅仅暴露有限的资源和有限的外域站点访问。**

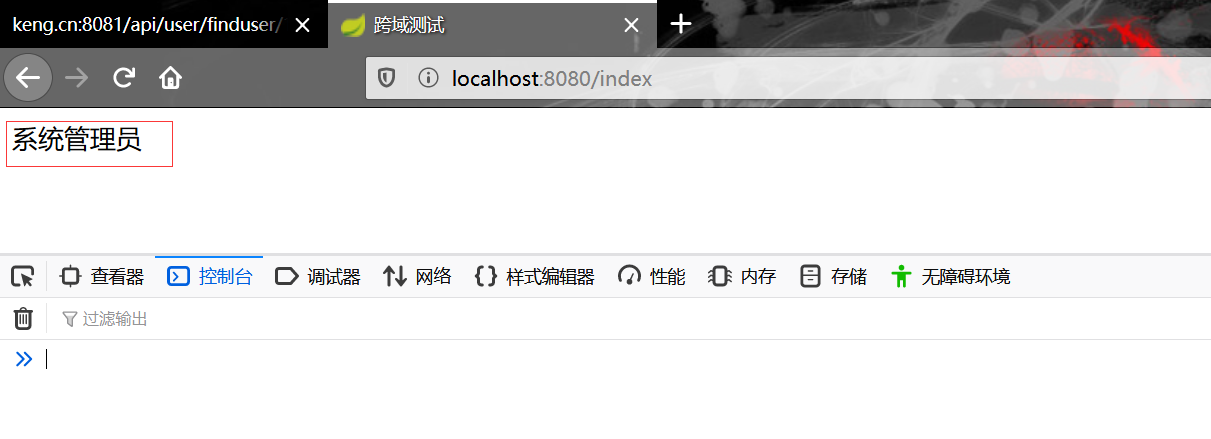
1. **方案1：@CrossOrigin注解**
2. **Spring 4.2后提供了@CrossOrigin注解，该注解可以标注于方法或者类上，包含了以下属性:**



1. **改造demo14中的UserController类**

|  |
| --- |
| **@ApiOperation(value = "获取用户信息", notes = "根据用户id获取用户信息")**  **@ApiImplicitParam(name = "id", value = "用户id", required = true, paramType = "path")**  **@GetMapping("/finduser/{id}")**  **@CrossOrigin("http://localhost:8080")**  **public User searchUserById(@PathVariable("id") Integer id){**  **return userService.queryUserById(id);}** |

1. **分别启动demo14和demo15**



1. **跨域成功**
2. **方案二：CORS全局配置-实现WebMvcConfigurer接口，建跨域配置类**

|  |
| --- |
| **import org.springframework.context.annotation.Configuration;**  **import org.springframework.format.FormatterRegistry;**  **import org.springframework.http.HttpHeaders;**  **import org.springframework.http.converter.HttpMessageConverter;**  **import org.springframework.validation.MessageCodesResolver;**  **import org.springframework.validation.Validator;**  **import org.springframework.web.method.support.HandlerMethodArgumentResolver;**  **import org.springframework.web.method.support.HandlerMethodReturnValueHandler;**  **import org.springframework.web.servlet.HandlerExceptionResolver;**  **import org.springframework.web.servlet.config.annotation.\*;**  **import java.util.List;**  **/\*\***  **\* 跨域配置**  **\*/**  **@Configuration**  **public class CorsConfig implements WebMvcConfigurer {**  **@Override**  **public void addCorsMappings(CorsRegistry registry) {**  **registry.addMapping("/\*\*")**  **.allowedOrigins("http://localhost:8080")**  **.allowedMethods("GET")**  **.allowCredentials(true)**  **.exposedHeaders(HttpHeaders.SET\_COOKIE).maxAge(3600L); //maxAge(3600)表明在3600秒内，不需要再发送预检验请求，可以缓存该结果**  **}** |

1. **方案三：过滤器实现：通过实现Fiter接口在请求中添加一些Header来解决跨域的问题。**

|  |
| --- |
| **import org.springframework.stereotype.Component;**  **import javax.servlet.\*;**  **import javax.servlet.http.HttpServletRequest;**  **import javax.servlet.http.HttpServletResponse;**  **import java.io.IOException;**  **@Component**  **public class CorsFilter implements Filter {**  **@Override**  **public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)**  **throws IOException, ServletException {**  **HttpServletResponse res = (HttpServletResponse) response;**  **res.addHeader("Access-Control-Allow-Credentials", "true");**  **res.addHeader("Access-Control-Allow-Origin", "\*");**  **res.addHeader("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");**  **res.addHeader("Access-Control-Allow-Headers", "Content-Type,X-CAF-Authorization-Token,sessionToken,X-TOKEN");**  **if (((HttpServletRequest) request).getMethod().equals("OPTIONS")) {**  **response.getWriter().println("ok");**  **return;**  **}**  **chain.doFilter(request, response);**  **}**  **@Override**  **public void destroy() {**  **}**  **@Override**  **public void init(FilterConfig filterConfig) throws ServletException {**  **}**  **}** |