SpringBoot配置跨域Cors

01、同源策略

同源策略[same origin policy]是浏览器的一个安全功能，不同源的客户端脚本在没有明确授权的情况下，不能读写对方资源。 同源策略是浏览器安全的基石。

什么是源

源[origin]就是协议、域名和端口号。例如：http://www.baidu.com:80这个URL。

什么是同源

若地址里面的协议、域名和端口号均相同则属于同源。

是否是同源的判断

例如判断下面的`URL`是否与 http://www.a.com/test/index.html 同源

- http://www.a.com/dir/page.html 同源
- http://www.child.a.com/test/index.html 不同源，域名不相同
- https://www.a.com/test/index.html 不同源，协议不相同
- http://www.a.com:8080/test/index.html 不同源，端口号不相同

哪些操作不受同源策略限制

1. 页面中的链接，重定向以及表单提交是不会受到同源策略限制的；
2. 跨域资源的引入是可以的。但是`JS`不能读写加载的内容。如嵌入到页面中的`<script src="..."></script>`，`<img>`，`<link>`，`<iframe>`等。

02、跨域

受前面所讲的浏览器同源策略的影响，不是同源的脚本不能操作其他源下面的对象。想要操作另一个源下的对象就需要跨域。 在同源策略的限制下，非同源的网站之间不能发送 `AJAX` 请求。

03、如何解决跨域

- 降域

  可以通过设置 `document.damain='a.com'`，浏览器就会认为它们都是同一个源。想要实现以上任意两个页面之间的通信，两个页面必须都设置 `documen.damain='a.com'`。

- `JSONP`跨域
- `CORS` 跨域

为了解决浏览器同源问题，`W3C` 提出了跨源资源共享，即 `CORS`(Cross-Origin Resource Sharing)。

04、CORS 简介

`CORS` 做到了如下两点：

- 不破坏即有规则
- 服务器实现了 `CORS` 接口，就可以跨源通信

基于这两点，`CORS` 将请求分为两类：简单请求和非简单请求。

1、简单请求

在`CORS`出现前，发送`HTTP`请求时在头信息中不能包含任何自定义字段，且 `HTTP` 头信息不超过以下几个字段：

- `Accept`

- `Accept-Language`
- `Content-Language`
- `Last-Event-ID`
- `Content-Type` 只限于 [`application/x-www-form-urlencoded` 、`multipart/form-data`、`text/plain`] 类型

一个简单的请求例子：

一个简单的请求例子：

```
GET /test HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, sdch, br
Origin: http://www.examples.com
Host: www.examples.com
```

对于简单请求，`CORS`的策略是请求时在请求头中增加一个`Origin`字段，服务器收到请求后，根据该字段判断是否允许该请求访问。

1. 如果允许，则在 HTTP 头信息中添加 `Access-Control-Allow-Origin` 字段，并返回正确的结果；
2. 如果不 允许，则不在 HTTP 头信息中添加 `Access-Control-Allow-Origin` 字段 。

除了上面提到的 `Access-Control-Allow-Origin` ，还有几个字段用于描述 `CORS` 返回结果：

1. `Access-Control-Allow-Credentials`： 可选，用户是否可以发送、处理 `cookie`;
2. `Access-Control-Expose-Headers`：可选，可以让用户拿到的字段。有几个字段无论设置与否都可以拿到的，包括：`Cache-Control`、`Content-Language`、`Content-Type`、`Expires`、`Last-Modified`、`Pragma` 。

2、非简单请求

对于非简单请求的跨源请求，浏览器会在真实请求发出前，增加一次`OPTION`请求，称为预检请求(`preflight request`)。预检请求将真实请求的信息，包括请求方法、自定义头字段、源信息添加到 HTTP 头信息字段中，询问服务器是否允许这样的操作。

例如一个`DELETE`请求：

```
OPTIONS /test HTTP/1.1
Origin: http://www.examples.com
Access-Control-Request-Method: DELETE
Access-Control-Request-Headers: X-Custom-Header
Host: www.examples.com
```

与 `CORS` 相关的字段有：

1. 请求使用的 `HTTP` 方法 `Access-Control-Request-Method` ；
2. 请求中包含的自定义头字段 `Access-Control-Request-Headers` 。

服务器收到请求时，需要分别对 `Origin`、`Access-Control-Request-Method`、`Access-Control-Request-Headers` 进行验证，验证通过后，会在返回 `HTTP`头信息中添加 ：

```
Access-Control-Allow-Origin: http://www.examples.com
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
Access-Control-Allow-Headers: X-Custom-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 1728000
```

他们的含义分别是：

1. Access-Control-Allow-Methods: 真实请求允许的方法
2. Access-Control-Allow-Headers: 服务器允许使用的字段
3. Access-Control-Allow-Credentials: 是否允许用户发送、处理 cookie
4. Access-Control-Max-Age: 预检请求的有效期，单位为秒。有效期内，不会重复发送预检请求

当预检请求通过后，浏览器会发送真实请求到服务器。这就实现了跨源请求。

05、Spring Boot 配置 CORS

1、使用`@CrossOrigin`注解实现

#如果想要对某一接口配置 `CORS`，可以在方法上添加 `@CrossOrigin` 注解：

```
@CrossOrigin(origins = {"http://localhost:9000", "null"})
@RequestMapping(value = "/test", method = RequestMethod.GET)
public String greetings() {
    return "{\"project\":\"just a test\"}";
}
```

#如果想对一系列接口添加 CORS 配置，可以在类上添加注解，对该类声明所有接口都有效：

```
@CrossOrigin(origins = {"http://localhost:9000", "null"})
@RestController
@SpringBootApplication
public class SpringBootCorsTestApplication {

}
```

#如果想添加全局配置，则需要添加一个配置类：

```
@Configuration
public class WebMvcConfig extends WebMvcConfigurerAdapter {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("POST", "GET", "PUT",
"OPTIONS", "DELETE")
                .maxAge(3600)
                .allowCredentials(true);
    }
}
```

另外，还可以通过添加 Filter 的方式，配置 CORS 规则，并手动指定对哪些接口有效。

```
@Bean
public FilterRegistrationBean corsFilter() {
    UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowCredentials(true);
config.addAllowedOrigin("http://localhost:9000");
    config.addAllowedOrigin("null");
    config.addAllowedHeader("*");
    config.addAllowedMethod("*");
    source.registerCorsConfiguration("/**", config); // CORS
配置对所有接口都有效
    FilterRegistrationBean bean =
newFilterRegistrationBean(new CorsFilter(source));
    bean.setOrder(0);
    return bean;
}

```