跨域问题的解决方案

什么是跨域？

当一个请求 url 的协议、域名、端口三者之间任意一个与当前页面的 url 不同即为跨域。

当前页面url	被请求页面url	是否跨域	原因
http://www.test.com/	http://www.test.com/index.html	否	同源（协议、域名、端口号相同）
http://www.test.com/	https://www.test.com/index.html	跨域	协议不同（http/https）
http://www.test.com/	http://www.baidu.com/	跨域	主域名不同（test/baidu）
http://www.test.com/	http://blog.test.com/	跨域	子域名不同（www/blog）
http://www.test.com:8080/	http://www.test.com:7001/	跨域	端口号不同（8080/7001）

为什么会出现跨域问题？

出于浏览器的同源策略限制。同源策略（Sameoriginpolicy）是一种约束，它是浏览器最核心也是最基本的安全功能，如果缺少了同源策略，则浏览器的正常功能可能会受到影响。可以说 web 是构建在同源策略基础上的，浏览器只是针对同源策略的一种实现。同源策略会阻止一个域的 JavaScript 脚本和另外一个域的内容进行交互。所谓同源（即指在同一个域）就是两个页面具有相同的协议（protocol）、主机（host）和端口号（port）。

目前常用的跨域解决方案主要有三种，详细如下：

方案一：

在方法上加入注解@CrossOrigin。

方案二：

配置过滤器。

允许整个项目跨域访问，可以通过 Filter 来进行过滤。

```
public class SimpleCORSFilter implements Filter{

    @Override
    public void destroy() {

    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse res,
            FilterChain chain) throws IOException, ServletException {
            HttpServletResponse response = (HttpServletResponse) res;
            response.setHeader("Access-Control-Allow-Origin", "*");
            response.setHeader("Access-Control-Allow-Methods", "POST, GET, OPTIONS,
DELETE");
            response.setHeader("Access-Control-Max-Age", "3600");
            response.setHeader("Access-Control-Allow-Headers", "x-requested-with");
            chain.doFilter(req, res);
```

```
    }

    @Override
    public void init(FilterConfig arg0) throws ServletException {

    }

}
```

在 web.xml 文件中加入下面的配置

```
<filter>
        <filter-name>cors</filter-name>
        <filter-class>com.ssm.web.filter.SimpleCORSFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>cors</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</filter>
```

方案三：

后台配置同源 Cors（推荐）

```
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;
/**
 * 实现基本的跨域请求
 * @author linhongcun
 *
 */
@Configuration
public class CorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        final UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new
UrlBasedCorsConfigurationSource();
        final CorsConfiguration corsConfiguration = new CorsConfiguration();
        /*是否允许请求带有验证信息*/
        corsConfiguration.setAllowCredentials(true);
        /*允许访问的客户端域名*/
        corsConfiguration.addAllowedOrigin("*");
```

```
        /*允许服务端访问的客户端请求头*/
        corsConfiguration.addAllowedHeader("*");
        /*允许访问的方法名,GET POST 等*/
        corsConfiguration.addAllowedMethod("*");
        urlBasedCorsConfigurationSource.registerCorsConfiguration("/**", corsConfiguration);
        return new CorsFilter(urlBasedCorsConfigurationSource);
    }
}
```