SpringBoot设置CORS的的本质都是通过设置响应头信息来告诉前端该请求是否支持跨域。

SpringBoot设置CORS的方式主要有以下三种。

1. 配置过滤器CorsFilter

```
@Configuration
public class CorsConfig {

    @Bean
  CorsFilter corsFilter() {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOrigins(Arrays.asList("*"));
    configuration.setAllowedMethods(Arrays.asList("*"));
    configuration.setAllowedHeaders(Arrays.asList("*"));
    configuration.setAllowCredentials(true);
    UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", configuration);
    return new CorsFilter(source);
  }
}
```

2. 实现接口WebMvcConfigurer

```
@Configuration
public class WebMvcConfig implements WebMvcConfigurer {

    @Override
  public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
        .allowedOrigins("*")
        .allowedHeaders("*")
        .allowedMethods("*")
        .allowCredentials(true);
  }
}
```

3. 使用注解@CrossOrigin

@CrossOrigin注解可以用在类或者方法上

用在控制器类上，表示 该类的所有方法都允许跨域

```
@RestController
@CrossOrigin
public class TestController {

    @GetMapping("test")
    public String test() {
        return "success";
    }
}
```

@CrossOrigin注解源码

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface CrossOrigin {

  /**
   * 这origins和value是一样的
   * 允许来源域名的列表，例如 www.baidu.com，匹配的域名是跨域预请求Response头中的Access-Control-Aloow_origin字段值。
   * 不设置确切值时默认支持所有域名跨域访问。
   */
  @AliasFor("origins")
  String[] value() default {};

  @AliasFor("value")
  String[] origins() default {};

  /**
   * 高版本下Spring2.4.4使用originPatterns而不是value和origins
   */
  String[] originPatterns() default {};

  /**
   * 跨域请求中允许的请求头中的字段类型， 该值对应跨域预请求Response头中的Access-Control-Allow-Headers字段值。
   * 不设置确切值默认支持所有的header字段（Cache-Controller、Content-Language、Content-Type、Expires、Last-Modified、Pragma）跨域访问
   */
  String[] allowedHeaders() default {};

  /**
   * 跨域请求请求头中允许携带的除Cache-Controller、Content-Language、Content-Type、Expires、Last-Modified、Pragma这六个基本字段之外的其他字段信息，
   * 对应的是跨域请求Response头中的Access-control-Expose-Headers字段值
```

```
     */
    String[] exposedHeaders() default {};

    /**
     * 跨域HTTP请求中支持的HTTP请求类型（GET、POST...），
     * 不指定确切值时默认与 Controller 方法中的 methods 字段保持一致。
     */
    RequestMethod[] methods() default {};

    /**
     * 浏览器是否将本域名下的cookie信息携带至跨域服务器中。默认携带至跨域服务器中，但要实现cookie共享还需要前端在AJAX请求中打开withCredentials属性。
     * 该值对应的是是跨域请求 Response 头中的 'Access-Control-Allow-Credentials' 字段值。
     */
    String allowCredentials() default "";

    /**
     * 该值的目的是减少浏览器预检请求/响应交互的数量。默认值1800s。设置了该值后，浏览器将在设置值的时间段内对该跨域请求不再发起预请求。
     * 该值对应的是是跨域请求Response头中的Access-Control-Max-Age字段值，表示预检请求响应的缓存持续的最大时间。
     */
    long maxAge() default -1;
}
```