CORS

CORS

- **CORS -** Cross-origin resource sharing. O'zaro domainlar bo'yicha manbalarni almashish.

- Cross-origin resource sharing (**CORS**) is a mechanism that allows JavaScript on a web page to make AJAX requests to another domain. different from the domain from where it originated. By default, such web requests are forbidden in browsers, and they will result into *same origin security policy* errors.
- Cross-origin resource sharing (CORS) bu javaScripga boshqa bir domain dan turib JavaScript ga AJAX request larni boshqa bir domainga jo’natish qilish imkonini beradi. Yani Fornt boshqa domain da turibdi va u boshqa domain dagi serverga murojar qilishi mumkin.
- Default holatda server boshqa domain dan kelgan requestlarni qabul qilmaydi.
- Biz backent da o’zimiz nastroyka qilib CORS ga ruxsat berishimiz kerak.

Global CORS configuration

- In spring boot application, it is recommended to just declare a WebMvcConfigurer bean. There allowed all origion

```
@Configuration
public class MyConfiguration {
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**");
            }
        };
    }
}
```

Global CORS configuration 2

- You can easily change any properties, as well as only apply this CORS configuration to a specific path pattern:

```
@Override
public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/api/**")
                .allowedOrigins("http://domain2.com")
                .allowedMethods("PUT", "DELETE")
                        .allowedHeaders("header1", "header2", "header3")
                .exposedHeaders("header1", "header2")
                .allowCredentials(false).maxAge(3600);
}
```

CORS with Spring Security

- To enable CORS support through Spring security, configure CorsConfigurationSource bean and use HttpSecurity.cors() configuration.
- In next slide

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http.cors().and()
      //other config
  }

  @Bean
  CorsConfigurationSource corsConfigurationSource()
  {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOrigins(Arrays.asList("https://example.com"));
    configuration.setAllowedMethods(Arrays.asList("GET","POST"));
    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", configuration);
    return source;
  }
}
```

Ajax GET request example

```
<!DOCTYPE html>
<html lang="en">
<body>

<h1>Result: </h1>
<div id="result">dasda</div>
</body>

</html>
<script>
    var xhttps = new XMLHttpRequest();

    xhttps.onreadystatechange = function () {
        document.getElementById("result").innerText = this.responseText;
    };
    xhttps.open("GET", "http://localhost:8080/category", false);
    xhttps.send();
</script>
```

Ajax POST request example

```
<!DOCTYPE html>
<html lang="en">
<body>
<h1>Result: </h1>
<div id="result">dasda</div>
</body>

</html>
<script>
    var xhttps = new XMLHttpRequest();
    var userObj = {name: "Alijon", surname: "Aliyev"};
    var jsonBody = JSON.stringify(userObj);

    xhttps.onreadystatechange = function () {
        document.getElementById("result").innerText = this.responseText;
    };

    xhttps.open("POST", "http://localhost:8080/category/test", true);
    xhttps.setRequestHeader('Content-Type', 'application/json');
    xhttps.send(jsonBody);
</script>
```

Links

- https://spring.io/blog/2015/06/08/cors-support-in-spring-framework
- https://spring.io/guides/gs/rest-service-cors/
- https://howtodoinjava.com/spring-boot2/spring-cors-configuration/
- https://howtodoinjava.com/java/servlets/java-cors-filter-example/