Introdução a Web Services REST com Spring Framework

Parte 9 → Configurando o CORS – Cross Origin Resource Sharing

Prof. Me. Jorge Luís Gregório

www.jlgregorio.com.br

Agenda

- O que é CORS?
- Implementando CORS por meio de anotações
- Implementando CORS globalmente

O que é CORS?

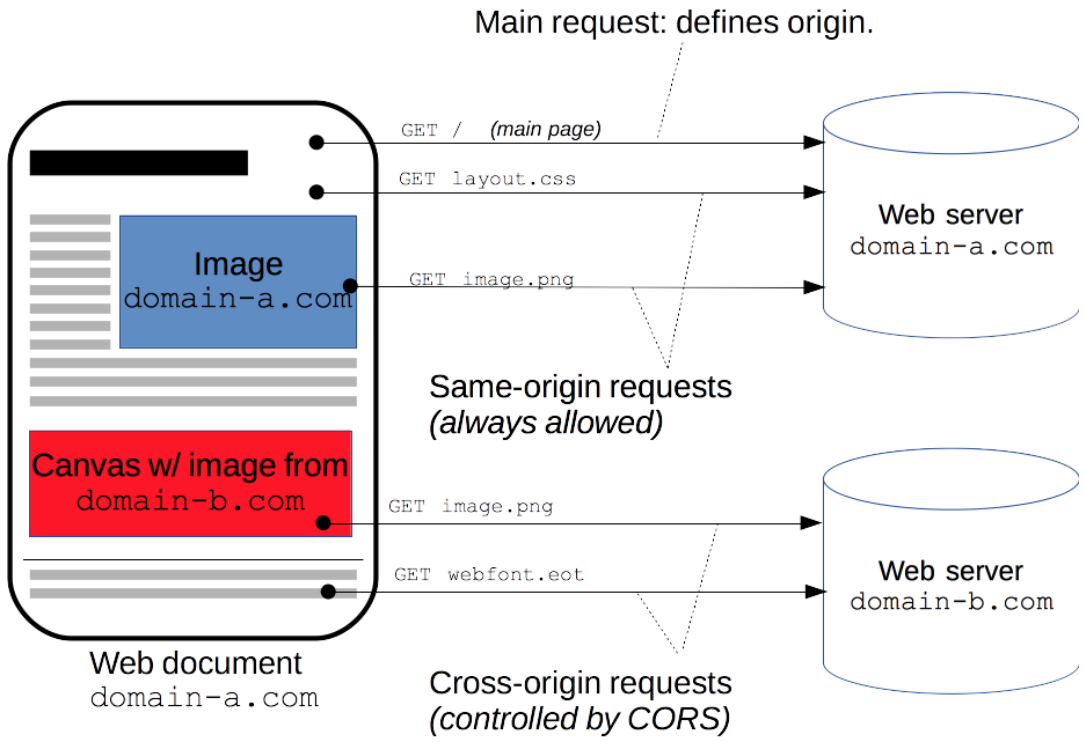
O que é CORS?

- **CORS - Cross-Origin Resource Sharing** (*Compartilhamento de recursos com origens diferentes*) é um mecanismo que usa cabeçalhos adicionais HTTP para informar a um navegador que permita que um aplicativo Web seja executado em uma origem (domínio) com permissão para acessar recursos selecionados de um servidor em uma origem distinta. Um aplicativo Web executa uma requisição cross-origin HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem.

MDN Web Docs (2021)

Na prática o que isso significa?

- Os navegadores web modernos implementam uma política de segurança chamada *Same-Origin* (*Same-Origin Policy*);
- Isso significa que o recurso (*resource*) de um site só pode ser consumido por outro site se ambos estiverem no mesmo domínio;
- Isso cria um grande problema, pois é muito comum APIs REST consumir outros serviços que estejam em outros domínios;
- Precisamos habilitar o CORS para que determinados serviços em determinados domínios possam consumir nossa API.



Implementando CORS por meio de anotações

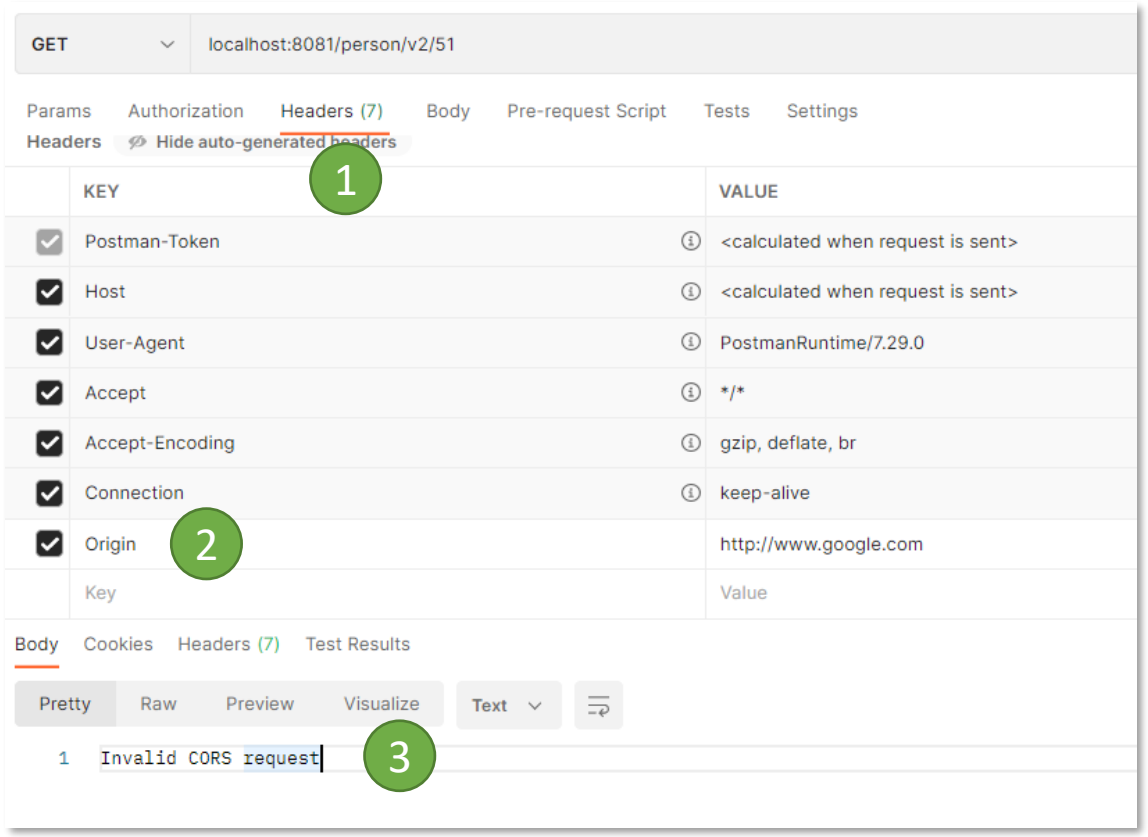
Implementando CORS via anotações

- Uma das maneiras de configurar o CORS em uma aplicação Spring é via anotações;
- A anotação **@CrossOrigin** pode ser feita a nível de classe ou para cada método específico;
- No exemplo a seguir, a anotação foi realizada a nível de classe;
- Note que a propriedade ***origins*** é usada para determinarmos quais são as origens aceitas nas requisições;
- Nesse exemplo, estamos aceitando para todo o *endpoint /person/v2* apenas requisições das origens *localhost:8081* e do meu site 😉.
- **ATENÇÃO!** Caso o atributo ***origins*** seja ignorado, isso significa que a requisição irá aceitar todas as origens;
- A mesma ideia pode ser aplicada via métodos. Faça os testes!

```
@CrossOrigin(origins = {"http://localhost:8081", "http://www.jlgregorio.com.br"})
@RestController
@RequestMapping("/person/v2")
public class PersonControllerV2 {
   ...
   ...
}
```

Testando

- Para testar de maneira mais eficiente, deveríamos fazer uma aplicação JS para fazer a requisição via Browser;
- Entretanto, podemos simular um requisição CORS usando o Postman:
  - Antes de fazer uma requisição ao *endpoint* anotado, abra a guia **Headers** (1) e adicione o cabeçalho **Origin** (2);
  - O valor deste cabeçalho deve ser um dos valores configurados na anotação, caso contrário será retornado um erro de CORS (3)



Implementando CORS globalmente

CORS para todos!

- Em muitos casos, ficar anotando todos os controllers e/ou métodos pode ser uma tarefa cansativa e que causa erros;
- Assim, o ideal é fazer com que todas as requisições da API sejam tratadas segundo o CORS;
- Para isso, crie um novo pacote chamado ***config*** dentro do pacote principal da aplicação;
- Agora crie uma classe Java com o nome ***WebConfig***;
- O código da classe está a seguir.... →

Adicionando a configuração

- No exemplo a seguir estamos adicionando um ***CorsRegistry*** que irá interceptar todas as requisições dos métodos e origens especificadas;

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

   public void addCorsMappings(CorsRegistry registry){
      registry.addMapping("/**")
            .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS", "HEAD", "TRACE", "CONNECT");
            .allowedOrigins("http://localhost:8081", "http://www.jlgregorio.com.br");

   }
   ...
   ...
   ...
}
```

Conclusão

- Para evitar problemas de CORS, considerando os *clients* de sua API, é necessário definir as políticas que serão aplicadas;
- Lembre-se que o CORS existe devido a Same-Domain Policy, uma diretiva de segurança dos navegadores modernos;
- Logo, é imperativo que sua API configure o CORS de maneira adequada.

Referências

- MDN Web Docs. **Cross-Origin Resource Sharing (CORS)**. 2021. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS>. Acesso em: 10 fev. 2022.
- MDN Web Docs. **Same-Origin Policy. 2022**. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy>. Acesso em: 10 fev. 2022.

Sobre mim

JORGE LUÍS GREGÓRIO

- Professor da Faculdade de Tecnologia “Prof. José Camargo” – Fatec Jales, e da Escola Técnica Estadual Dr. José Luiz Viana Coutinho – Etec Jales;
- Articulista do Jornal de Jales – Coluna “Fatecnologia”;
- Apresentador do Tech Trends, podcast oficial da Fatec Jales;
- Bacharel em Sistemas de Informação; Especialista em Desenvolvimento de Software para Web e Mestre em Ciência da Computação.
- Trabalha com tecnologia desde 1998, tendo atuado como analista de suporte; administrador de redes de computadores; desenvolvedor de software, *webdesigner* e professor.

- Site oficial: www.jlgregorio.com.br
- Perfil do LinkedIn: www.linkedin.com/in/jlgregorio81
- Currículo Lattes: http://lattes.cnpq.br/3776799279256689