Spring Security

Beginner to Guru

Overview of CORS

Cross-Origin Resource Sharing - CORS

- Cross-Origin Resources are common in web applications
- A typical web page may use Cross-Origin Resources for:
  - Images / videos
  - Content Delivery Network (CDN)
  - Web fonts
- Some Cross-Origin Requests are forbidden (mainly Ajax)

Cross-Origin Resource Sharing - CORS

- CORS is a standard to enable Cross-Origin Requests when needed
- CORS is a browser standard, it is the browser implementing the security restrictions
  - Server provides data, browser implements
- Browser displaying page from myapplication.com, Javascript makes request from api.myapplication.com
  - Browser performs HTTP GET at api.myapplication.com ; with HTTP Header “Origin: myapplication.com"

Cross-Origin Resource Sharing - CORS

- Server Responds with header “Access-Control-Allow-Origin: myapplication.com"
  - This indicates the api domain can be accessed from root domain
  - Can be an asterisk (*) - wildcard for all domains
- An Example of this could be:
  - a ReactJS application running at myapplication.com
  - Spring Boot application running at api.myapplication.com

CORS Headers

- **Request Headers:**
  - Origin
  - Access-Control-Request-Method
  - Access-Control-Headers
- **Response Headers:**
  - Access-Control-Allow-Origin
  - Access-Control-Allow-Credentials
  - Access-Control-Expose-Headers
  - Access-Control-Max-Age
  - Access-Control-Allow-Methods
  - Access-Control-Allow-Headers

CORS Preflight

- Browsers can also do a ‘preflight’
  - A check to see if action is allowed
- Browser makes a HTTP OPTIONS request using request headers for method and/or headers
- Server responds with HTTP 204 if okay, error if not

Spring Framework CORS Support

- Spring MVC / WebFlux
  - Majority of CORS support is built into Spring MVC / WebFlux
  - Configuration can be very granular
- Spring Security
  - Works in conjunction with Spring MVC / Webflux
  - Alternatively can Intercept requests in security filter chain

SPRING FRAMEWORK

GURU