Authentication using OAuth2

1. Essential concepts
2. Using OAuth2 in Spring Boot

Demo project: **`demo-security-oauth2`**

1. Essential Concepts

- Overview of Spring Boot Security
- Spring authentication and authorization
- Overview of OAuth2

Overview of Spring Boot Security

- Spring encapsulates security, offering the following benefits:
  - Portable
    - Portable across web containers (and standalone)
    - No need for platform-specific declarations or role-mappings
  - Comprehensive
    - Supports common web authentication techniques
  - Elegant
    - Security is decoupled from application logic
    - Achieved via Spring AOP and security filters

Spring Authentication and Authorization

- Spring authentication:
  - Establishes a security context
  - Security context contains info about the authenticated principal

- Spring authorization:
  - Examines the security attributes of a secured item
  - Gets principal information from the security context
  - Grants or denies access to the secured item

Overview of OAuth2

- OAuth2 is a client framework
  - Enables access to a user's resources…
  - With the user's consent…
  - Without exposing the user's username/password

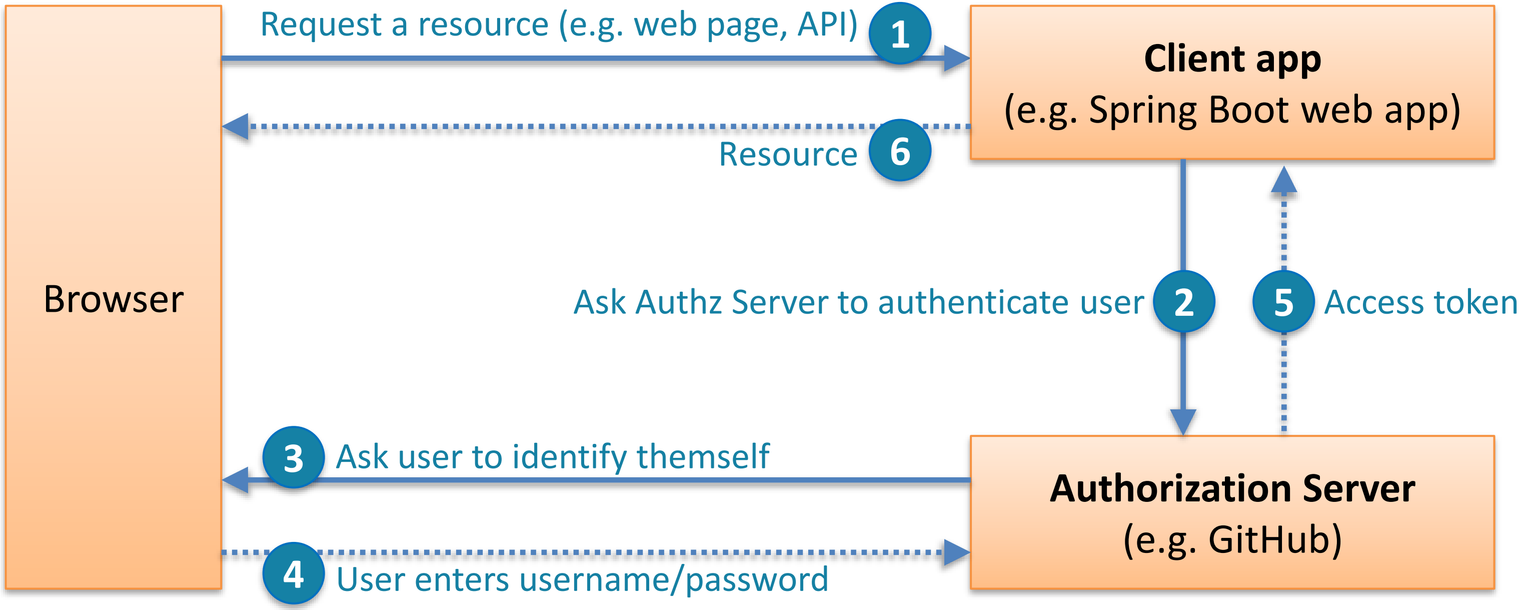
- For example:
  - The user tries to access an endpoint in a Spring Boot web app
  - The web app redirects to a social-media login page, where the user is challenged to authenticate themselves
  - The social media provider returns an "access token" to the web app, which represents the user's authenticated identity

2. Using OAuth2 in Spring Boot

- Overview
- Spring Boot dependency for OAuth2
- Resources in the demo app
- Specifying authentication rules
- Registering your client app with GitHub
- Adding GitHub credentials to your client app
- Running the client app
- Displaying additional info about the user

Overview

- In this section we'll show an example of how to how to use OAuth2 in a Spring Boot client app


Request a resource (e.g. web page, API)
1
Client app
(e.g. Spring Boot web app)
Resource
6
Browser
Ask Authz Server to authenticate user
2
5
Access token
3
Ask user to identify themself
Authorization Server
(e.g. GitHub)
4
User enters username/password

olsen software

Spring Boot Dependency for OAuth2

- Take a look in the demo app
  - `demo-security-oauth2`

- Note the pom file includes the OAuth2 dependency
  - This automatically pulls in the Spring Security library too

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
```

pom.xml

Resources in the Demo App

- The demo app has several resources the user might request
  - `src/main/resources/static/index.html`
  - `src/main/java/mypackage/Controller1.java`
  - `src/main/java/mypackage/Controller2.java`

- By default Spring Boot protects all URLs in your web app
  - i.e. the user must be authenticated to access any URL

Specifying Authentication Rules

- You can specify authentication rules as follows:

```
@Configuration
public class SecurityConfig {

   @Bean
   public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

      return http
              .authorizeHttpRequests()
                 .requestMatchers("/", /index.html", "/controller1").permitAll()
                 .requestMatchers("/controller2").authenticated()
                 .anyRequest().authenticated()
              .and()
                 .oauth2Login()
              .and()
                 .build();
   }
}
```

SecurityConfig.java

Registering your Client App with GitHub (1 of 3)

- The previous slide specified we want to use OAuth2 to perform authentication

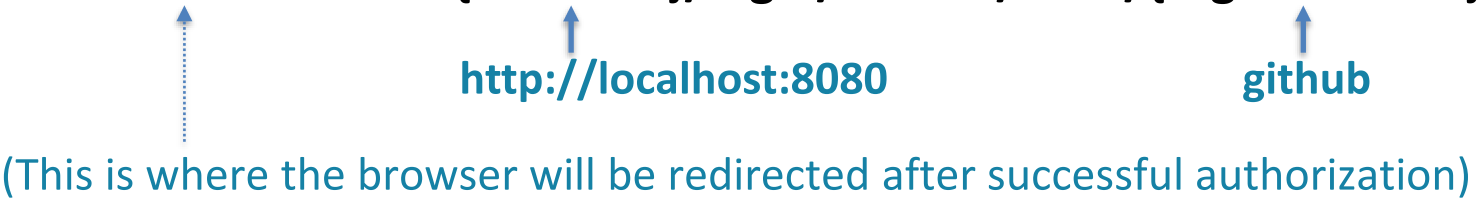
- You must tell OAuth2 how to contact an authorization server in order to do that task
  - E.g. GitHub

- The first step is to register your application with GitHub
  - So GitHub is aware of your application…
  - So GitHub will be willing to authenticate users on its behalf

Registering your Client App with GitHub (2 of 3)

- To register your client app with GitHub:
  - Sign in to https://github.com/settings/developers
  - Click **OAuth apps**, then click **New OAuth App**

- Then specify the following info:
  - Application name **My Cool App**
  - Homepage URL **http://localhost:8080**
  - Authz callback URL **{homeUrl}/login/oauth2/code/{registrationId}**



http://localhost:8080

github

(This is where the browser will be redirected after successful authorization)

Registering your Client App with GitHub (3 of 3)

- You will then be able to enter additional info about the app
  - E.g. the owner of the app
  - E.g. a logo for the app
  - E.g. add the app to GitHub Marketplace

- You must also grab the following credentials from GitHub, which you will need to add into your client app (see next slide)
  - **Client ID**
  - **Client secret**

Adding GitHub Credentials to your Client App

- In your Spring Boot app, add the GitHub credentials (from previous slide) to your `application.properties` file

```
spring.security.oauth2.client.registration.github.client-id=<client-id>
spring.security.oauth2.client.registration.github.client-secret=<client-secret>
```

- This is what will happen when a user accesses a web resource:
  - If the web resource is protected…
  - OAuth2 will contact GitHub, passing the client credentials above
  - GitHub will challenge the user to authenticate themself
  - GitHub will return an access token 👍 to your client app

Running the Client App

- Run the client app and try to access the following resources:
  - `/` (no authentication needed)
  - `/index.html` (no authentication needed)
  - `/controller1` (no authentication needed)
  - `/controller2` (you'll be redirected to GitHub to authenticate)

Displaying Additional Info About the User

- During authentication, GitHub also returned info about the user
  - You can access this info in your client app as follows:

```
@RestController
public class Controller2 {

    @GetMapping(path="/controller2-with-info")
    public String getWithName(@AuthenticationPrincipal OAuth2User principal) {
        return "Hello from Controller2, " +
                principal.getAttribute("name") + ", " +
                principal.getAttribute("company");
    }
    …
}
```

Controller2.java

- To see this in action, ping `/controller2-with-info`

Summary

- Essential concepts
- Using OAuth2 in Spring Boot

Exercise

- We've seen how to use OAuth2 to perform authentication, in the following project:
  - `demo-security-oauth2`

- Spring Boot also supports other authentication techniques, e.g. forms authentication, see here:
  - `demo-security-forms`