

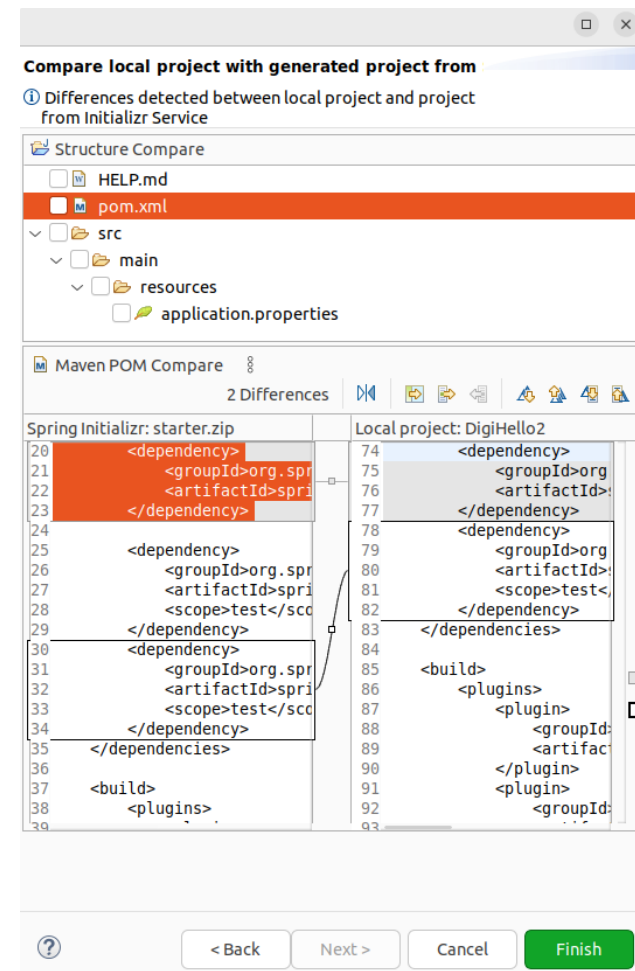
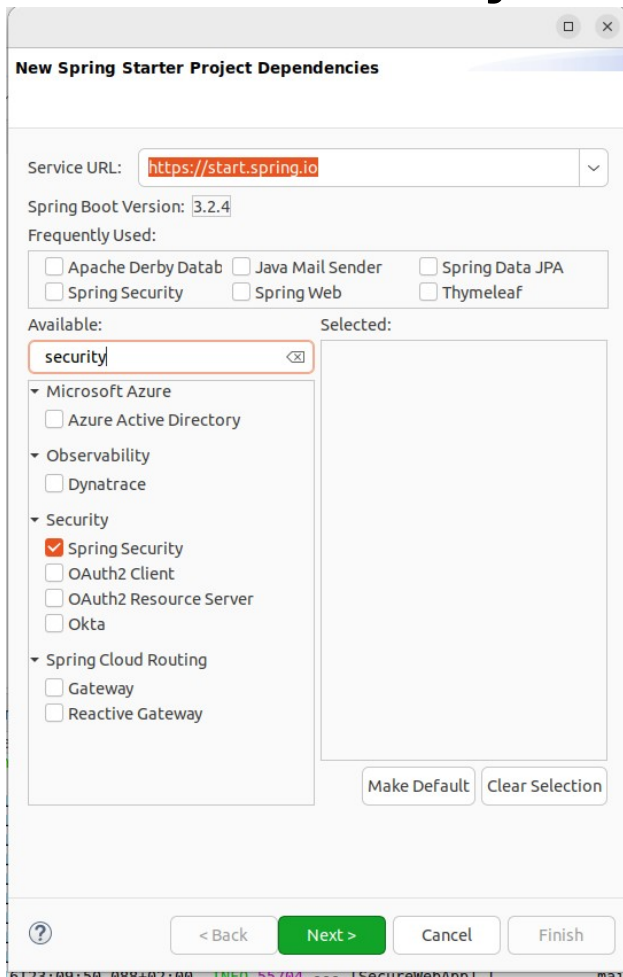
Spring Boot Security

Objectifs

- Sécuriser notre application Spring MVC

Ajout dépendance

- Dans STS, ajouter le starter security (merge)

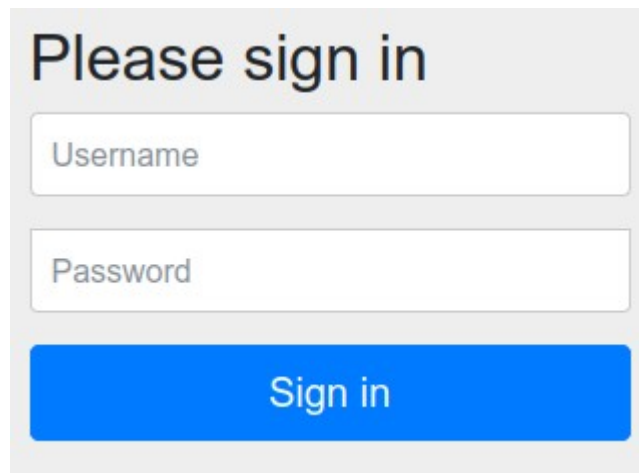


Accès

- Redémarrer le projet et visualiser le mdp :

Using generated security password: 1a0cd347-14b4-416c-a72f-d4644c29044a

- Accéder à votre application

A sign-in form with a light gray background. It features the text "Please sign in" at the top. Below this are two input fields: "Username" and "Password". At the bottom is a blue button with the text "Sign in".

Please sign in

Username

Password

Sign in

Accès

- Accéder à votre application en utilisant :
 - UserName : user
 - Password : celui trouvé dans la console

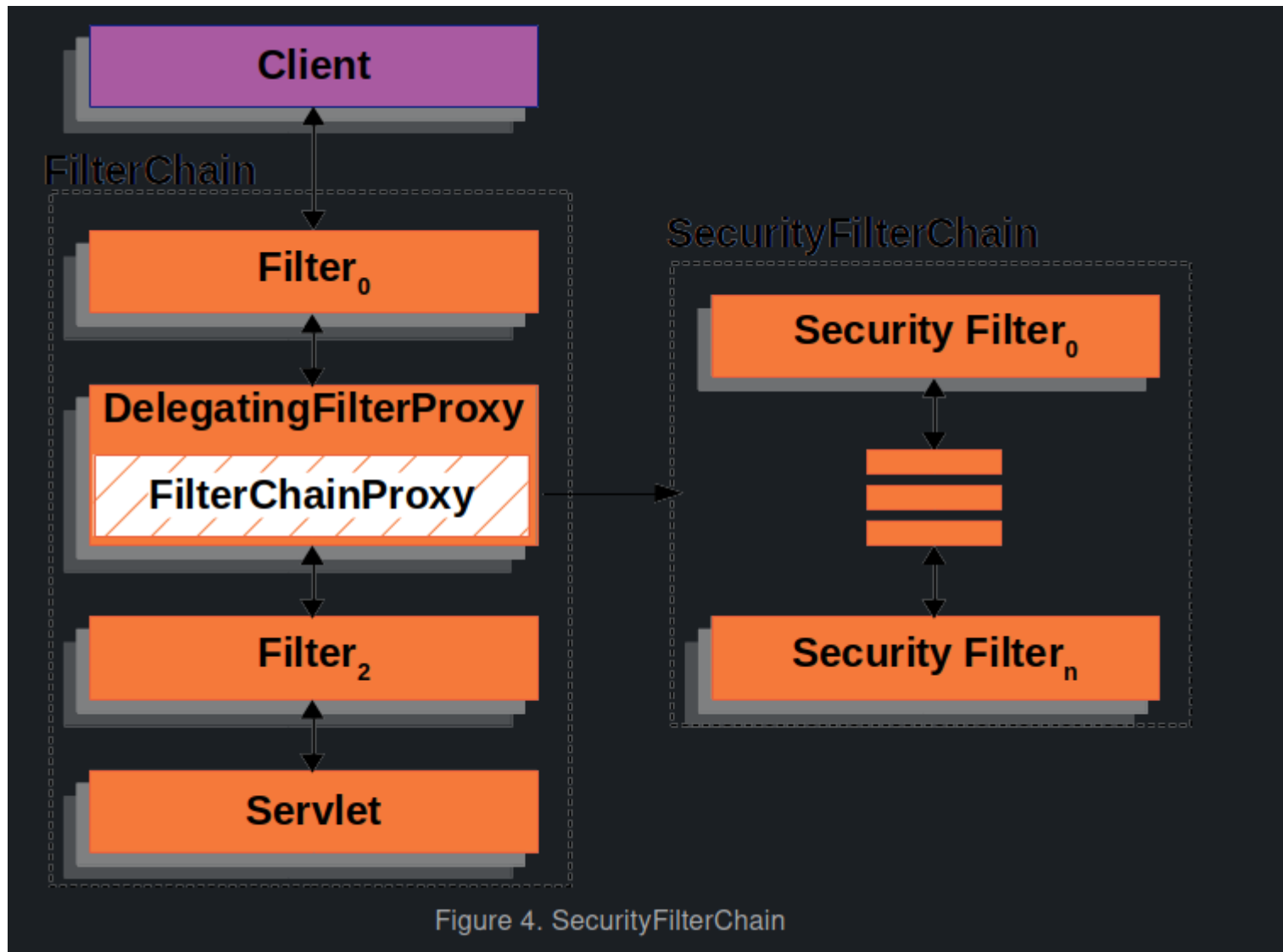
User admin

- Dans application.properties ajouter un user :
 - `spring.security.user.name=admin`
 - `spring.security.user.password=admin`
- Redémarrer et tester
 - Pratique pour le dev...

Concepts clés

- Définir la source des utilisateurs
- Créer les règles d'accès des users
- Associer les parties de l'application aux règles
- Appliquer la sécurisation d'ensemble

Concepts clés



Concepts clés

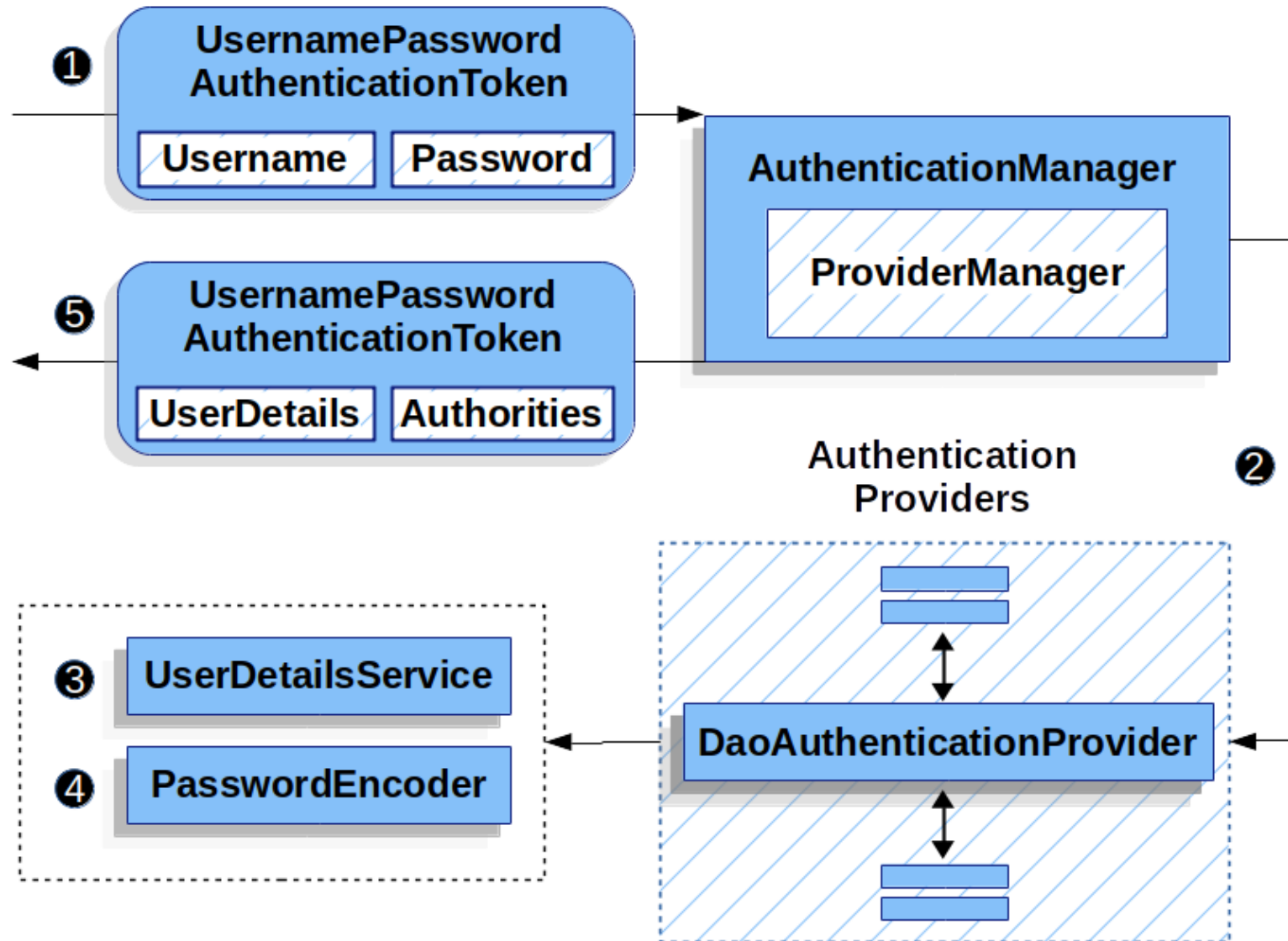


Figure 1. DaoAuthenticationProvider Usage

Configuration

- Définir un bean de configuration spring
 - Annotez une classe @Configuration
 - Déclarer un UserDetailsService @Bean (testez...)

```
@Configuration
public class SecurityConfig {
    @Bean
    public UserDetailsService userDetailsService() {
        UserDetailsManager userDetailsManager =
            new InMemoryUserDetailsManager();
        userDetailsManager.createUser(
            User.withDefaultPasswordEncoder()
                .username("user")
                .password("password")
                .roles("USER")
                .build());
        userDetailsManager.createUser(
            User.withDefaultPasswordEncoder()
                .username("admin")
                .password("password")
                .roles("ADMIN")
                .build());
        return userDetailsManager;
    }
}
```

Stockage en BDD

- Définir une entité JPA UserAccount

```
@Entity
public class UserAccount {
    @Id
    @GeneratedValue
    private Long id;
    private String username;
    private String password;
    @ElementCollection(fetch = FetchType.EAGER)
    private List<GrantedAuthority> authorities = //
        new ArrayList<>();
}
```

Stockage en BDD

- Créer le repository de UserAccount
- Créer le service
- Initialiser deux users avec ROLE_USER et ROLE_ADMIN.

```
public UserAccount(String username, String password, String... authorities) {  
    this.username = username;  
    this.password = password;  
    this.authorities = Arrays.stream(authorities)  
        .map(SimpleGrantedAuthority::new)  
        .map(GrantedAuthority.class::cast)  
        .toList();  
}
```

Stockage en BDD

- Mapper UserAccount / UserDetails de spring :

```
/**
 * To map useraccount to userdetails
 */
public class UserMapper {

    public static UserDetails toUserDetails(UserAccount userAccount) {
        return User.builder().username(userAccount.getUsername()).password(userAccount.getPassword()).authorities(userAccount.getAuthorities()).build();
    }
}
```

- Configurer le Password Encoder :

```
@Bean
PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

- Configurer le UserDetailsService :

```
@Bean
UserDetailsService userDetailsService(UserAccountRepository userAccountRepository) {
    return username -> UserMapper.toUserDetails(userAccountRepository.findByUsername(username));
}
```

Stockage en BDD

- Testez votre application et regardez en BDD

jdbc:h2:~/Documents/workspace-spring-t

+ DEPARTMENT
+ TOWN
+ USER_ACCOUNT
+ USER_ACCOUNT_AUTHORITIES

SELECT * FROM USER_ACCOUNT;

ID	PASSWORD	USERNAME
1	\$2a\$10\$X3uhEx20Y/S6q1INvooL8.ncvOj/SN.47VTLFsownGxfd1mBF4j0u	admin
2	\$2a\$10\$5liBx9ccEMgeYWKU3/maa.dce1z3Flj.RdVpQjQmeiBmvh/OW16B.	user

(2 enregistrements, 3 ms)

SELECT * FROM USER_ACCOUNT_AUTHORITIES;

USER_ACCOUNT_ID	AUTHORITIES
1	aced0005737200426f72672e737072696e676672616d65776f726b2e73656375726974792e636f72652e617574686f726974792e53696d706c654772616e746564417574686f7269747900000000
2	aced0005737200426f72672e737072696e676672616d65776f726b2e73656375726974792e636f72652e617574686f726974792e53696d706c654772616e746564417574686f7269747900000000

(2 enregistrements, 3 ms)

Sécuriser les vues et les routes

- Objectifs :
 - Permettre l'accès non authentifié à la page d'accueil
 - Autoriser les utilisateurs connectés à voir la liste des villes
 - Autoriser seulement les admin à supprimer des villes ou à créer des données
- Il faut intercaler un nouveau filtre dans la chaîne de sécurité

Sécuriser les routes : SecurityFilterChain

- Configurer un nouveau Bean :

```
@Bean
SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http.authorizeHttpRequests(request -> request
        .requestMatchers("/", "login").permitAll()
        .requestMatchers("/logout").authenticated()
        .requestMatchers("/townList").authenticated()
        .requestMatchers("/deleteTown/**").hasRole("ADMIN")
        .anyRequest().denyAll()
    )
    .httpBasic(Customizer.withDefaults())
    .formLogin(Customizer.withDefaults());
    return http.build();
}
```

- Testez ! (se déconnecter en utilisant /logout)

Sécuriser les méthodes

- On peut sécuriser la route de delete :

```
.requestMatchers("/deleteTown/**").hasRole("ADMIN")
```

- Pour assurer : activer la sécurité fine

```
@Configuration  
@EnableMethodSecurity  
public class SecurityConfig {
```

- Sécuriser une méthode de suppression

```
@PreAuthorize("hasRole('ADMIN')")  
public boolean deleteTown(Long id) {  
    Town result = townRepository.findById(id).get();  
    if (result!=null) {  
        townRepository.deleteById(id);  
        return true;  
    }else {  
        return false;  
    }  
}
```

Sécuriser les méthodes

- Test : Dé-sécuriser volontairement /deleteTown
 - Mettre la route en .authenticated()
 - Tester la suppression en user et en admin

Utiliser les infos du user

- Injecter le userdetail dans le model de la vue

```
@GetMapping
public String getIndex(Model model, Authentication authentication) {
    model.addAttribute("authentication", authentication);
    return "index";
}
```

- Afficher les infos

```
<h3>Profil utilisateur :</h3>
<ul>
<li th:text="'Username : '+${authentication.name}">Username</li>
<li th:text="'Authorities : '+${authentication.authorities}">Authorities</li>
</ul>
```

A vos projets...

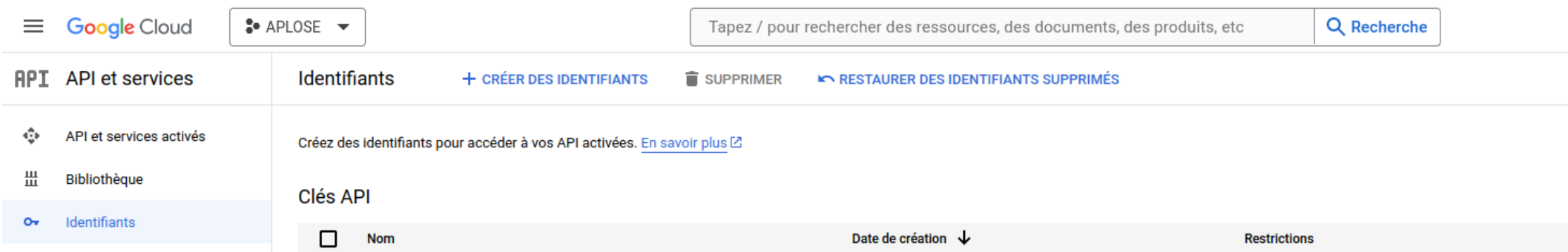
- TP : sur votre projet, activer et configurez Spring Security

Utiliser Oauth (de Google)

- La base d'utilisateurs de Google est immense
- Autre possibilité : FB, X, ...
- Pour et contre...

Utiliser Oauth (de Google)

- Étapes :
 - Créer un projet sur Google Cloud Dash Board
<https://console.cloud.google.com/home/dashboard>
 - Activer les apis Google que vous voulez (youtube data api v3 par exemple)
 - Créer des « Credentials » : Oauth client



The screenshot shows the Google Cloud console interface. At the top, there's a navigation bar with the Google Cloud logo, a dropdown menu showing 'APLOSE', and a search bar with the placeholder text 'Tapez / pour rechercher des ressources, des documents, des produits, etc' and a 'Recherche' button. Below this, the left sidebar shows a menu with 'API et services', 'API et services activés', 'Bibliothèque', and 'Identifiants' (which is highlighted). The main content area is titled 'Identifiants' and includes links for '+ CRÉER DES IDENTIFIANTS', 'SUPPRIMER', and 'RESTAURER DES IDENTIFIANTS SUPPRIMÉS'. It contains a message: 'Créez des identifiants pour accéder à vos API activées. [En savoir plus](#)'. Below this, there's a section for 'Clés API' with a table header showing columns for 'Nom', 'Date de création', and 'Restrictions'.

	Nom	Date de création	Restrictions
--	-----	------------------	--------------

Utiliser Oauth (de Google)

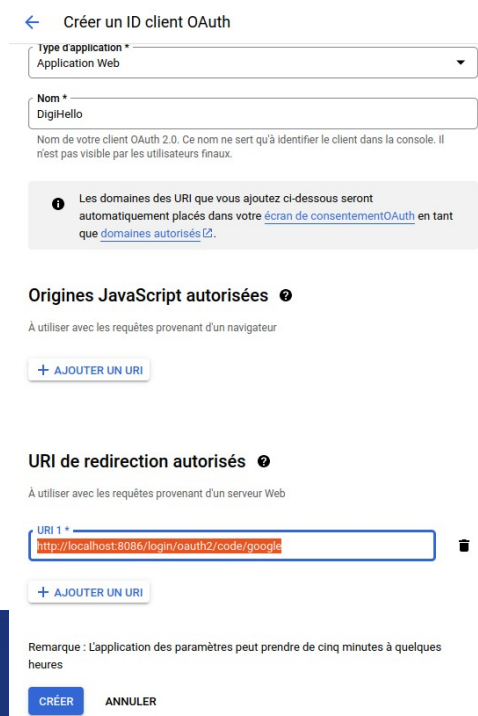
- Étapes (suite) :

- Type : Application Web, Nom : DigiHello

- Url de redirection autorisée :

<http://localhost:8086/login/oauth2/code/google>

- « Créer »



The screenshot shows the 'Créer un ID client OAuth' (Create an OAuth client ID) page in the Google Cloud Console. The page is in French and contains the following elements:

- Créer un ID client OAuth**: The main heading of the page.
- Type d'application ***: A dropdown menu set to 'Application Web' (Web Application).
- Nom ***: A text input field containing 'DigiHello'.
- Informations**: A section with a note: 'Nom de votre client OAuth 2.0. Ce nom ne sert qu'à identifier le client dans la console. Il n'est pas visible par les utilisateurs finaux.'
- Origines JavaScript autorisées**: A section with a note: 'À utiliser avec les requêtes provenant d'un navigateur'. It includes a '+ AJOUTER UN URI' button.
- URI de redirection autorisés**: A section with a note: 'À utiliser avec les requêtes provenant d'un serveur Web'. It includes a text input field for 'URI 1 *' containing the URL 'http://localhost:8086/login/oauth2/code/google', a trash icon, and a '+ AJOUTER UN URI' button.
- Remarque**: A note at the bottom stating: 'L'application des paramètres peut prendre de cinq minutes à quelques heures'.
- Buttons**: 'CRÉER' (Create) and 'ANNULER' (Cancel) buttons at the bottom.

Utiliser Oauth (de Google)

- Étapes (suite) :
 - Passez sur l'écran de consentement

DIGIHELLO

Tapez / pour rechercher des ressources,

Modifier l'enregistrement de l'application

✓ Écran de consentement OAuth

✓ Niveaux d'accès

3 Utilisateurs tests

4 Résumé

OAuth

lisation

Utilisateurs tests

Quand l'état de publication est défini sur "Test", seuls les utilisateurs tests peuvent accéder à l'application. La limite d'utilisateurs autorisés avant la validation de l'application est de 100 pour toute sa durée de vie. [En savoir plus](#)

+ ADD USERS

Filtre

Saisissez le nom ou la valeur de la propriété

?

Informations utilisateur

oandrade33000@gmail.com

ENREGISTRER ET CONTINUER

ANNULER

Utiliser Oauth (de Google)

- Étapes (suite) :
 - Créer un projet Spring Boot avec :
Oauth2 client, Spring Web, Spring Reactive Web, ...
 - Paramétrer Spring Security


```
spring:  
  security:  
    oauth2:  
      client:  
        registration:  
          google:  
            clientId: 1091754...  
            clientSecret: GOCrX...  
            scope: openid,profile,email,https://www.googleapis.com/auth/youtube
```

Utiliser Oauth (de Google)


- Étapes (suite) :
 - Ajoutez la logique de connexion Oauth et les requêtes API voulues : recherchez et implémentez...




Utiliser Oauth (de Google)

 Se connecter avec Google




digihello veut accéder à votre compte Google

 oandrade33000@gmail.com

Sélectionnez les services auxquels **digihello** a accès

 Gérer votre compte YouTube. [En savoir plus](#) ☒

Comme vous utilisez Se connecter avec Google, digihello pourra

-  Créer une relation entre vous et vos informations personnelles sur Google ☒
-  Consulter vos informations personnelles, y compris celles que vous avez choisi de rendre disponibles publiquement ☒
-  Afficher l'adresse e-mail principale associée à votre compte Google ☒

Utiliser Oauth (de Google)

Activités | Navigateur Web Firefox | 17 avril 22:55






Kindle | API ID client pour Application | API ID client pour Application | localhost:8080/?continue&continue | Utiliser OAuth 2.0 pour le | (931) Olivier ANDRADE S | +

localhost:8080/?continue&continue

Archimédys | APOSE | Andorens | APOSE | Agenda | Mutu02 | ChatGPT | FB | LinkedIn | DIGINAMIC | Udem | Dev Dolibarr | FOSSBilling - Free and... | MineCraftDev | >> | Autres marque-pages

TestOauth

Your Videos

Id	Published	Thumbnail	Title	Description
m2E_-VRIqlg	2020-04-06T08:42:47Z		Installation Serveur de Dolibarr (v11.0.3) en moins de 15 minutes	Dans cette présentation, je vous présente simplement l'installation serveur d'une instance Dolibarr.
-c-qB6DLkKM	2019-01-18T11:18:08Z		Matinale présentation de l'ERP CRM libre Dolibarr @Apose @PoleREALIS #Montpellier	Pour ceux qui n'ont pu se libérer et souhaitent assister à la matinale Dolibarr de mardi, j'ai pu f
nao9H96Vu5Q	2017-04-20T16:06:46Z		Matinale Dolibarr 5 compta et gestion libre d'entreprise	
xonXx7ZLrto	2019-12-28T19:31:50Z		Présentation du projet Cocoon Start, le coworking des entrepreneurs écoresponsables	Il était une fois, Sandrine et Olivier, deux terriens amoureux de la nature... Ils avaient du mal à
zFrk6BgFWWI	2017-03-13T15:05:43Z		Matinale Gratuite Apose #Montpellier - Gestion de projets agile et outils d'aide à l'organisation	Cette vidéo est le premier enregistrement d'une matinale Apose à Montpellier. Retrouver l'informati

je copié

```
ava (17 avr. 2024, 22:55:25) [p  
g.springframework.securi  
estWrapper[ org.springfr  
enticationToken [Princip  
ated OAuth2Authenticati  
atcherServlet 'dispatch  
tHandler [classpath [MET  
to [Is Secure]
```

Sécuriser une API

- Pour une api seule (sans page spring web mvc) : pas de page de login webmvc
- L'application doit fournir un webservice d'authentification qui va générer une « Token »
- Cette token sera échangée entre le client et le serveur à chaque requête (header, queryparam,...).
- Peut contenir des informations (claims).
- C'est le serveur qui garanti la durée de validité et la sécurité de la token
- Standard de fait : JWT (Json Web Token)

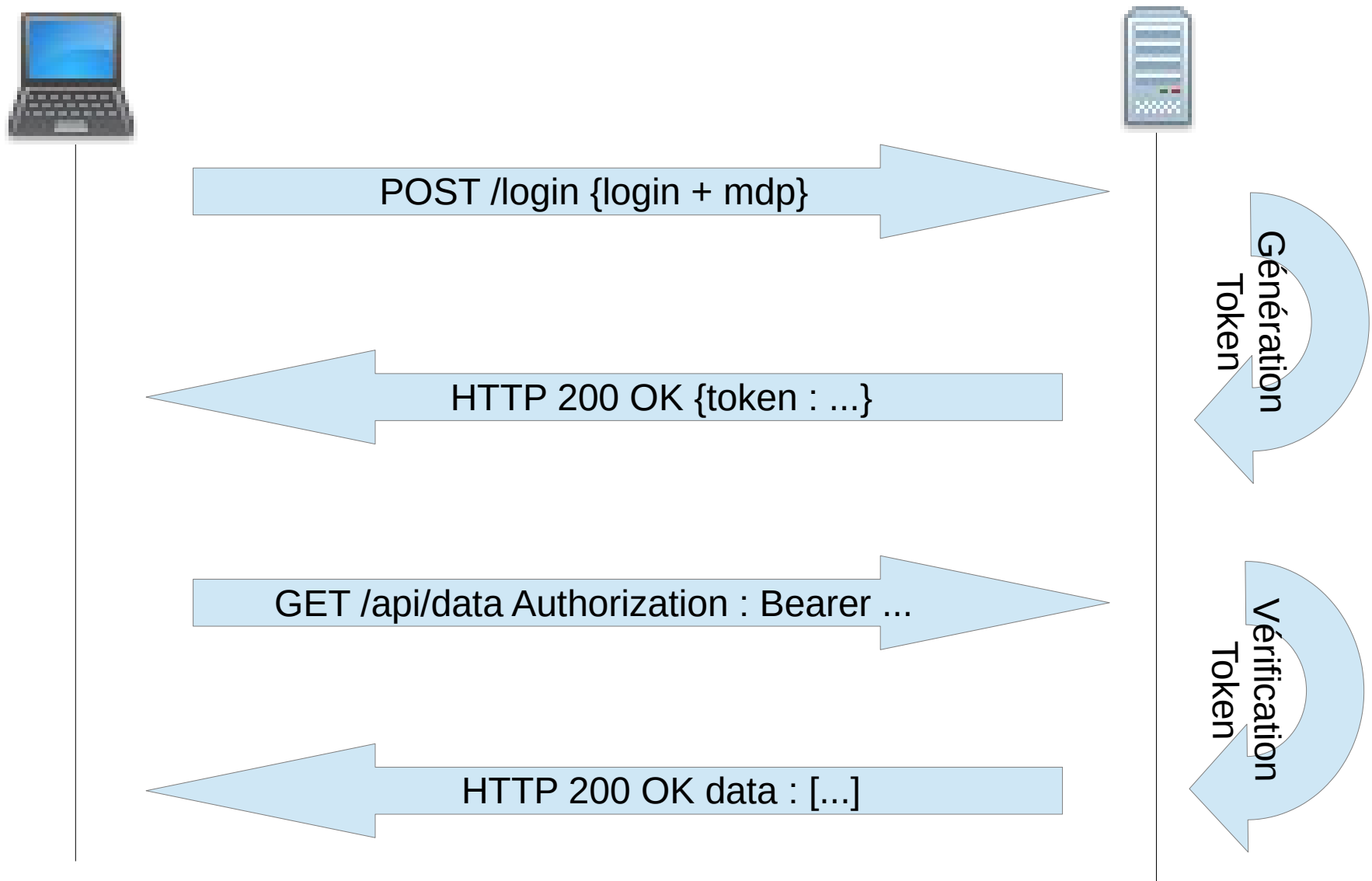
JWT Json Web Token

- Permet la vérification de l'intégrité et de l'authenticité des données échangées entre front et back
- Trois parties composent le jeton :
 - Header (description du jeton, algo, etc...)
 - Payload (infos échangées)
 - Signature
- Attention :
 - clé privée du serveur à protéger
 - HTTPS obligatoire car vol de token possible...
 - Révocation impossible donc expiration courte.

Authentication / Autorisation

- La première phase de sécurisation est l'authentification
 - Avec un login/mpd ou oauth l'utilisateur de l'application (web, mobile) se connecte
 - Le serveur d'authentification renvoie une token JWT encodé et signé, la token est sauvegardée en BDD sur le user.
 - Le client web stocke localement la token
- Le client demande une ressource (REST par exemple) en envoyant la token JWT dans le header
 - Le serveur autorise ou non l'accès à la ressource.

Authentication / Autorisation



Sécuriser une API

- TP : créer un projet Spring Boot pour une api REST sécurisée Spring Security + JWT
 - Dans STS créer un projet avec Initializr
 - Ajouter les dépendances Maven
 - Spring-boot-starter-web,
 - Spring-boot-security,
 - Spring-boot-starter-data-jpa,
 - H2, modelmapper, lombok, jjwt-api, jjwt-impl, jjwt-jackson

Sécuriser une API

- TP : suivre le tuto

<https://medium.com/spring-boot/spring-boot-3-spring-security-6-jwt-authentication-authorization-98702d6313a5>

- Tester votre projet :

- Avec postman : POST //api/v1/login avec le json
{"username":"admin","password":"admin"}
- Récupérer la token et avec postman : GET
apiv1/ping et Authorisation : Bearer token

Sécuriser une API

The screenshot displays a REST client interface with two panels. The top panel shows a GET request to `http://localhost:8080/api/v1/ping` with the Authorization tab selected. The authorization type is set to Bearer Token, and the token is `eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbilslmlhdCI6MTcyMDk3MTU2MywiZXBwIjozNzlwOTcxNjlfQ.5hXm0RDIL2vEhmSw2nqcg4u3FhX-bnoszC-IMyB_EKk`. The bottom panel shows the response body as `Welcome Admin`.

HTTP `http://localhost:8080/api/v1/ping` Save

GET `http://localhost:8080/api/v1/ping` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer Token Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

`eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbilslmlhdCI6MTcyMDk3MTU2MywiZXBwIjozNzlwOTcxNjlfQ.5hXm0RDIL2vEhmSw2nqcg4u3FhX-bnoszC-IMyB_EKk`

GET `http://localhost:8080/api/v1/ping`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token Token

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

`eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pbil...`

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize Text ⌵ ⌵

1 `Welcome Admin`