Gestion des avis (Partie 11)

# TP01 du module 10 – AOP

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Déploiement d’un WAR |

|  |
| --- |
| **Durée estimée** |
| 45 minutes |

Contexte

Dans les itérations de ce module, nous allons ajouter des aspects.

* Si vous n’aviez pas fini le TP du module 09. Vous trouverez les classes solutions dans un zip des ressources.
* Attention, la configuration Spring Boot n’est pas incluse
* Ainsi que les propriétés pour gérer OAuth 2.0 dans le fichier de configuration de Spring Boot : application.yml

Solution

# Configuration des logs

* Nous voulons créer un WAR. Il faut éviter que l’application perde trop de temps à tracer des logs.
  + Pour éviter de saturer les ressources, nous allons changer le niveau de trace de MongoTemplate et le passer en ERROR uniquement
* Dans le fichier de configuration de Spring Boot : application.yml :

#Log

logging:

level:

root: WARN

org:

springframework:

data:

mongodb:

core:

MongoTemplate: ERROR

# Création d’un WAR et déploiement dans Tomcat

* Le WAR doit s’appeler eni-gestion-avis-1.0.0-plain
* L’API doit être totalement sécurisée. Toutes les URLs obligent à se connecter
* Penser à la configuration OAuth en production
  + URL racine sera maintenant : http://localhost:8080/eni-gestion-avis
  + Appeler l’OAuth Apps : ENI-Gestion-Avis-VOTRE\_NOM

## Dans build.gradle :

* Ajouter L’id war

plugins {

id 'java'

// DEPLOIEMENT WAR TOMCAT

id 'war'

id 'org.springframework.boot' *version* '3.1.4'

id 'io.spring.dependency-management' *version* '1.1.3'

}

* Et la production sous Tomcat :
  + Starter pour les serveurs web Java

// DEMO DEPLOIEMENT WAR TOMCAT

// Starter Tomcat

providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'

* Mettre à jour le numéro de version pour correspondre au nom du war à générer

*version* = '1.0.0'

* Ajouter un bloc de déclaration après le bloc java nommé : taks.war pour définir le nom du war désiré

*tasks*.war{

archiveBaseName.set("eni-gestion-avis")

}

* Faire un « Refresh Gradle Project »

# Configuration du ServletContext pour Spring Boot et Tomcat :

* Ajouter sur la classe principale de SpringBoot :
  + L’héritage de SpringBootServletInitializer
  + Et redéfinir la méthode « configure » qui permettra d’associer ServletContext et Spring Boot et ainsi configurer l’exécution dans un Tomcat :

**package** fr.eni.gestionavis;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.boot.builder.SpringApplicationBuilder;

**import** org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication

**public** **class** GestionAvisApplication **extends** SpringBootServletInitializer {

@Override

**protected** SpringApplicationBuilder configure(SpringApplicationBuilder application) {

**return** application.sources(GestionAvisApplication.**class**);

}

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(GestionAvisApplication.**class**, args);

}

}

# Contraintes de sécurité

Le DispatchServlet peut rentrer en contrainte avec le SecurityFilterChain.

* Dès qu’on utilise requestMatchers
* Car les 2 peuvent réagir à des URLs

Pour éviter cela, il y a 2 solutions :

1. Faire que l’application Web soit totalement sécurisée : anyRequest().authenticated()
2. Utiliser AntPathRequestMatcher pour préciser que le contexte de sécurité s’en occupe

Dans ce cas, nous voulons que la totalité de l’API soit sécurisée. Il faut donc appliquer la première stratégie

* Supprimer la ligne :

.requestMatchers("/").permitAll()

# Configuration OAuth 2 en production :

Nous avions créé une configuration sur GitHub pour l’autorisation avec OAuth 2 pour des URLs du type http://localhost:8080

* Maintenant, notre URL racine correspondra à http://localhost:8080/eni-gestion-avis
* Il faut donc recréer une configuration OAuth pour la production
* Sous votre compte GitHub
* Allez sur vos paramètres
* Allez vers le bas sur l’option « Developer settings »
  + Sélectionner OAuth Apps
* Donner le nom ENI-Gestion-Avis-VOTRE\_NOM à votre application
  + Mettre l’URL de « localhost » : http://localhost:8080/eni-gestion-avis
  + L’URL d’autorisation 🡪 permet de gérer le jeton transmis par GitHub sur le contexte du projet (« localhost »)
    - http://localhost:8080/eni-gestion-avis/login/oauth2/code/github
* Vous obtiendrez une vue récapitulative
  + Où, GitHub aura créé un identifiant Client ID
* Il nous faut générer un « Client Secret »
  + GitHub vous demandera de ressaisir votre mot de passe GitHub pour valider
  + Il nous faut les 2 informations : Client ID et Client Secret pour effectuer toutes les opérations de connexion.
* Modifier les paramètres de OAuth 2.0 dans le fichier de configuration de Spring Boot : application.properties ou application.yml
  + Conseil, mettre en commentaire ceux de DEV

#configuration OAuth2 - GitHub

security:

oauth2:

client:

registration:

github:

#DEV

#client-id: 11a51…

#client-secret: 4f64…

#PROD

client-id: 2ad48…

client-secret: 517ba…

# Création du WAR :

Dans la vue « Gradle Tasks »

* Rechercher votre projet
* Ouvrir : build
* Faire un clic droit sur war 🡪 Run Gradle Tasks
* Dans la vue « Gradle Executions » vous verrez la progression du build
  + Une fois terminée
  + Allez dans un explorer sous votre projet dans votre workspace
  + Dans le répertoire « buid » 🡪 « libs » ; le WAR sera là

# Déploiement du WAR dans Tomcat :

* Copier eni-gestion-avis-1.0.0-plain dans le répertoire webapps de Tomcat
  + Dans notre exemple : C:\java\_outils\apache-tomcat-10.1.8\webapps
  + Pour que l’URL racine du déploiement soit : http://localhost:8080/eni-gestion-avis
  + Il faut renommer votre WAR en eni-gestion-avis.war
* Lancer une fenêtre de commande
  + Placez-vous dans le répertoire bin de votre Tomcat :
  + Exemple de commande pour le répertoire C:\java\_outils\apache-tomcat-10.1.8\bin

cd c:\java\_outils\apache-tomcat-10.1.8\bin

* Démarrer Tomcat en appelant la commande :

startup.bat

* Lorsque le démarrage est terminé
  + Tomcat va décompresser le WAR
  + Et exécuter l’application web
* Dans la console, vous verrez apparaitre la trace de démarrage de Spring Boot :



* Vous pouvez maintenant, vérifier le comportement de votre application déployée

# Vérifier l’exécution du WAR

* L’URL racine : http://localhost:8080/eni-gestion-avis
  + Elle est restreinte au permission OAuth

|  |  |
| --- | --- |
|  |  |