TP N°8 : Utilisation de l'AOP

SOMMAIRE

I- Objectifs :

- ✓ Compléter le TP n°7 en développant les deux aspects suivants :
  - Un premier aspect pour tracer les appels de méthodes.
  - Un deuxième aspect pour calculer le temps d'exécution de méthodes.
  - Un 3ième aspect pour gérer le droit d'accès.

II- Outils utilisés :

- ✓ Idem que le TP n°7.

III- Développement de l'application

1. pom.xml

Editer le fichier pom.xml du tp n°7 et ajouter la dépendance suivante :

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

2. Développement de l'aspect

a. Développement de l'annotation « LogExecutionTime »

```
package ma.cigma.springsecurity.aop;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface LogExecutionTime {
}
```

b. Développement de l'annotation « Tracabilite »

```
package ma.cigma.springsecurity.aop;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Tracabilite {
}
```

c. Développement de l'aspet

```
package ma.cigma.springsecurity.aop;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class ExampleAspect {
@Around("@annotation(LogExecutionTime)")
public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
long start = System.currentTimeMillis();
Object proceed = joinPoint.proceed();
long executionTime = System.currentTimeMillis() - start;
System.out.println(joinPoint.getSignature() + " executed in " + executionTime + "ms");
return proceed;
}
@Before("@annotation(Tracabilite)")
public void trace(JoinPoint joinPoint) throws Throwable {
String methodeName = joinPoint.getSignature().getName();
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
String dateString = format.format(new Date());
Authentication auth = SecurityContextHolder.getContext().getAuthentication();
String userName=auth.getName();
System.out.println(methodeName +" called by "+ userName +" at " + dateString);
}
}
```

d. Marquer les méthodes à tracer et/ou à calculer le temps d'exécution

Par exemple, annoter la méthode viewemp de la classe `EmpController` par les annotations `@LogExecutionTime` et `@Tracabilite` :

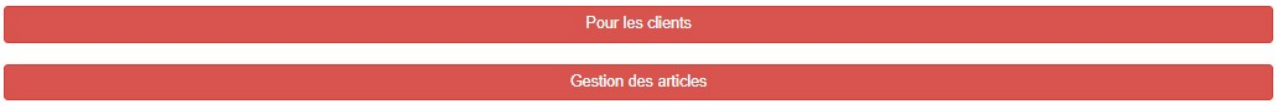
```
@RequestMapping("/list")
@LogExecutionTime
@Tracabilite
public String viewemp(Model m) {
    List<EmpVo> list = service.getEmployees();
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    m.addAttribute("userName", "Welcome " + auth.getName());
    m.addAttribute("list", list);
    return "/admin/emp/list";
}
```

Ici, les deux advices pour calculer le temps d'exécution et pour tracer seront appliqués pour la méthode viewemp.

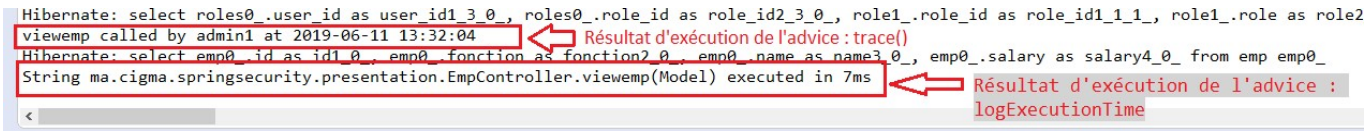
3. Les tests

- Lancer la méthode de démarrage de Spring BOOT et lancer le lien : http://localhost:8080/.

- Entrer le compte utilisateur admin1/admin1 et cliquer sur Login.



- Cliquer sur « Gestion des articles » et vérifier que les deux lignes suivantes ont été ajoutées au niveau de la log :



4. Création d'un aspect pour la gestion de l'authentification

Nous allons créer un aspect pour gérer la régle suivante :Seul l'utilisateur admin1 qui a le droit à supprimer un employé :

- Créer l'annotation suivante :

```
package ma.cigma.springsecurity.aop;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Admin1Profile {

}
```

- Créer la classe suivante :

```
package ma.cigma.springsecurity.aop;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;

import ma.cigma.springsecurity.service.exception.BusinessException;

@Aspect
@Component
public class AuthenticationAspect {
	@Around("@annotation(Admin1Profile)")
	public Object test(ProceedingJoinPoint joinPoint) throws Throwable {
		Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
		String userName = auth.getName();
		if (!userName.equals("admin1")) {
			System.out.println("Only user admin who is authorized to call
this method");
			throw new BusinessException("Only user admin1 who is
authorized to call this method");
		}
		Object proceed = joinPoint.proceed();
		return proceed;
	}
}
```

- Créer la classe suivante :

```java
package ma.cigma.springsecurity.service.exception;

import lombok.Data;

@Data
public class BusinessException extends RuntimeException {
	private static final long serialVersionUID = 1L;
	private String message;
	public BusinessException(String message) {
		this.message = message;
	}
}
```

- Modifier la classe EmpController.java comme suit :

```java
package ma.cigma.springsecurity.presentation;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import ma.cigma.springsecurity.aop.LogExecutionTime;
import ma.cigma.springsecurity.aop.Admin1Profile;
import ma.cigma.springsecurity.aop.Tracabilite;
import ma.cigma.springsecurity.domaine.EmpVo;
import ma.cigma.springsecurity.service.IEmpService;
import ma.cigma.springsecurity.service.exception.BusinessException;

@Controller
@RequestMapping("/admin/emp")
@ControllerAdvice
public class EmpController {

	@Autowired
	private IEmpService service;

	@ExceptionHandler(value = BusinessException.class)
	public String exception(Model m) {
		m.addAttribute("excep", true);
		List<EmpVo> list = service.getEmployees();
		Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
		m.addAttribute("userName", "Welcome " + auth.getName());
		m.addAttribute("list", list);
		return "/admin/emp/list";
	}

	@RequestMapping("/form")
	public String showform(Model m) {
		m.addAttribute("empVo", new EmpVo());
```

```
            return "/admin/emp/edit";
      }

      @RequestMapping(value = "/save", method = RequestMethod.POST)
      @LogExecutionTime
      public String save(@ModelAttribute("empVo") EmpVo emp) {
            service.save(emp);
            return "redirect:/admin/emp/list";
      }

      @RequestMapping("/list")
      @LogExecutionTime
      @Tracabilite
      public String viewemp(Model m) {
            List<EmpVo> list = service.getEmployees();
            Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
            m.addAttribute("userName", "Welcome " + auth.getName());
            m.addAttribute("list", list);
            return "/admin/emp/list";
      }

      @RequestMapping(value = "/edit/{id}")
      public String edit(@PathVariable Long id, Model m) {
            EmpVo emp = service.getEmpById(id);
            m.addAttribute("empVo", emp);
            return "/admin/emp/edit";
      }

      @RequestMapping(value = "/editsave", method = RequestMethod.POST)
      public String editsave(@ModelAttribute("empVo") EmpVo emp) {
            service.save(emp);
            return "redirect:/admin/emp/view";
      }

      @Admin1Profile
      @RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
      public String delete(@PathVariable Long id) {
            service.delete(id);
            return "redirect:/admin/emp/list";
      }

      @RequestMapping("/emp/salary/{salary}")
      public String getBySalary(@PathVariable Double salary, Model m) {
            List<EmpVo> list = service.findBySalary(salary);
            m.addAttribute("list", list);
            return "/admin/emp/list";
      }

      /**
       * Chercher la liste des employés ayant la même fonction
       */
      @RequestMapping("/emp/fonction/{fonction}")
      public String getByFonction(@PathVariable String fonction, Model m) {
            List<EmpVo> list = service.findByFonction(fonction);
            m.addAttribute("list", list);
            return "/admin/emp/list";
      }

      /**
       * Chercher la liste des employés ayant le même salaire et la même
```

```
fonction
	 */
	@RequestMapping("/emp/salary_and_fonction/{salary}/{fonction}")
	public String getBySalaryAndFonction(@PathVariable Double salary,
@PathVariable String fonction, Model m) {
		List<EmpVo> list = service.findBySalaryAndFonction(salary,
fonction);
		m.addAttribute("list", list);
		return "/admin/emp/list";
	}

	/**
	 * Chercher l'employé qui le grand salaire
	 */
	@RequestMapping("/max_salary")
	public String getMaxSalary(Model m) {
		EmpVo empVo = service.getEmpHavaingMaxSalary();
		List<EmpVo> list = new ArrayList<>();
		list.add(empVo);
		m.addAttribute("list", list);
		return "/admin/emp/view";
	}

	/**
	 * Afficher la liste des employés en utilisant la pagination
	 */
	@RequestMapping("/pagination/{pageid}/{size}")
	public String pagination(@PathVariable int pageid, @PathVariable int
size, Model m) {
		List<EmpVo> list = service.findAll(pageid, size);
		m.addAttribute("list", list);
		return "/admin/emp/view";
	}

	/**
	 * Trier les employés par le nom de champs qu'on passe dans l'URL
	 */
	@RequestMapping("/sort/{fieldName}")
	public String sortBy(@PathVariable String fieldName, Model m) {
		List<EmpVo> list = service.sortBy(fieldName);
		m.addAttribute("list", list);
		return "/admin/emp/view";
	}
}
```

- Modifier la page ***templates/admin/emp/list.html*** comme suit :

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
	xmlns:th="http://www.thymeleaf.org">

<head>
<title>Formation Spring Boot : Services Web</title>
<link rel="stylesheet" type="text/css" th:href="@{/css/home.css}" />
<link rel="stylesheet"
	href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script
	src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script
	src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></scrip
t>
```

```
</head>

<body>
	<div class="container">

		<form th:action="@{/logout}" method="get">
			<button class="btn btn-md btn-danger btn-block" name="registration"
				type="Submit">Logout</button>
		</form>

		<div class="panel-group" style="margin-top: 40px">
			<div class="panel panel-primary">
				<div class="panel-heading">
					<span th:utext="${userName}"></span>
				</div>
				<div class="panel-body">
					<div align="center" th:if="${excep}">
						<p style="font-size: 20; color: #FF1C19;">Vous
n'avez pas le
droit d'exécuter cette fonction</p>
					</div>
					<div th:switch="${list}">
						<h1 th:case="null">Aucun employé !</h1>
						<div th:case="*" class="container">
							<h3>Liste des employés :</h3>
							<table>
								<thead>
									<tr>
<th>Name</th>
<th>Salary</th>
<th>Fonction</th>
</tr>
</thead>
<tbody>
<tr th:each="empVo : ${list}">
<td th:text="${empVo.name}"></td>
<td th:text="${empVo.salary}"></td>
<td th:text="${empVo.fonction}"></td>
	<td><a th:href="@{/admin/emp/edit/{id}(id=${empVo.id})}">Edit</a></td>
<td><a th:href="@{/admin/emp/delete/{id}(id=${empVo.id})}">Delete</a></td>
</tr>
</tbody>
</table>
</div>
<p>
		<a href="/admin/emp/form">Ajouter un nouvel employé</a>
</p>
</div>



</div>
<p class="admin-message-text text-center"
th:utext="${adminMessage}"></p>
</div>
</div>

	</div>
</body>
</html>
```

5. Tester

Lancer l'application et s'authentifier avec un compte différent de admin1 et essayer de supprimer un employé. Le resultat est :