UNIVERSITATEA DIN BUCUREȘTI

VIRTUTE ET SAPIENTIA

WEB TECHNOLOGIES USING JAVA

Laboratory 3 - 23.10.2020

AGENDA

- Project examples
- Kahoot quiz
- Aspects - AOP
- Applications

Project examples

- You can find a list of projects examples on:
  https://github.com/silviabt/web-technologies-using-java-lab/blob/main/Project%20eg.pdf

Kahoot quiz

- Go to https://kahoot.it/
- Enter the pin and your full name
- Answer questions :)

Aspects - AOP

- **IoC - Inversion of control** => concept related to the application being controlled by the framework instead of being controlled by the classes that you write
- AOP - a way in which a framework intercepts methods calls and possibly alters the execution of methods
- Aspect: logic the framework executes when you call specific methods.

Aspects - AOP (II)

- Key concepts:
  - the object containing the methods you want to enrich with functionality -> **target object**
  - the logic you want Spring to execute when you call specific methods -> **aspect**
  - when should Spring execute this logic (before the method call, after the method call, instead of the method call) -> **advice**
  - the methods that Spring needs to intercept -> **pointcut**
  - Spring provides a proxy to the target object, which manages the calls to the real method and applies the aspect logic -> **weaving**

Aspects - AOP (III)

- Implementing an aspect:
  - Enable the aspect mechanism: @EnableAspectJAutoProxy on the configuration class
  - Add annotation: @Aspect on a new class, defined as a bean in the Spring context.
  - Define a method that will implement the aspect logic and tell Spring when and which methods to intercept using an advice annotation.
  - Implement the aspect logic.

Aspects - AOP Example (I)

1. Enable the aspect mechanism: @EnableAspectJAutoProxy on the configuration class

```
@Configuration
@ComponentScan(basePackages = "com.ub.webtechnologies")
@EnableAspectJAutoProxy
public class ApplicationConfiguration {
}
```

Aspects - AOP Example (II)

2. Add annotation: @Aspect on a new class, defined as a bean in the Spring context.

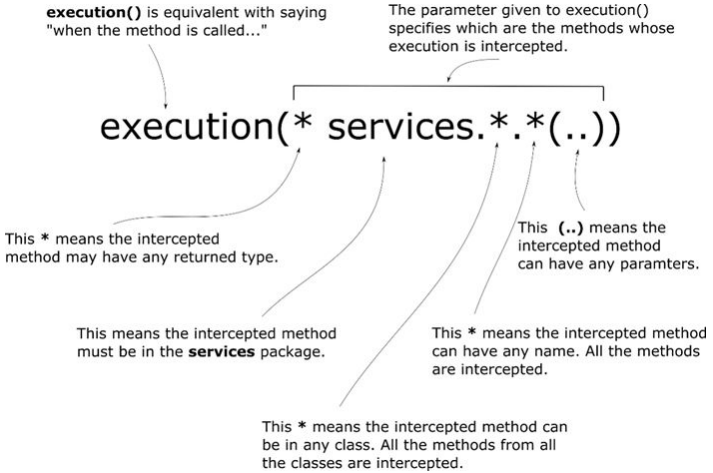
```
@Aspect
@Component
public class UserServiceAspect {
}
```

Aspects - AOP Example (III)

3. Define a method that will implement the aspect logic and tell Spring when and which methods to intercept using an advice annotation.

```
@Aspect
@Component
public class UserServiceAspect {

   @Before("execution(*
com.ub.webtechnologies.service.UserService.createUser(..))")
   public void before() {
   }
}
```



endava

Aspects - AOP Example (IV)

4. Implement the aspect logic.

```
@Aspect
@Component
public class UserServiceAspect {

   @Before("execution(* com.ub.webtechnologies.service.UserService.createUser(..))")
   public void before() {
       System.out.println("Request to create users");
   }
}
```

Applications

1. Define another method in the Advice class that will intercept the createUser method using as advice annotation: **@After.** When is executed the logic inside this method?
2. Define another method in the Advice class that will intercept the createUser method using as advice annotation: **@AfterReturning.** When is executed the logic inside this method?
3. Define another method in the Advice class that will intercept the createUser method using as advice annotation: **@AfterThrowing.** When is executed the logic inside this method?

Applications: Aspects - AOP - @Around Example (I)

- The method `createProfile` is no longer executed

```
@Around("execution(* com.ub.webtechnologies.service.ProfileService.createProfile(..))")
public Object around(ProceedingJoinPoint joinPoint) {
   System.out.println("I will return something else!");

   Profile.ProfileBuilder profileBuilder = new Profile.ProfileBuilder();

   return profileBuilder
           .user(new User.UserBuilder()
                   .name("OtherUser")
                   .build())
           .points(1233)
           .build();
}
```

Applications: Aspects - AOP - @Around Example (II)

- The method `createProfile` is executed and can be executed multiple times

```
@Around("execution(* com.ub.webtechnologies.service.ProfileService.createProfile(..))")
public Object around(ProceedingJoinPoint joinPoint) {
   System.out.println("I will return something else!");
   Object result = null;
   try {
       result = joinPoint.proceed();
       return result;
   } catch (Throwable throwable) {
       throwable.printStackTrace();
   }
   return result;
}
```

THANK YOU

Silvia Butan