endava

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

WEB TECHNOLOGIES USING JAVA

COURSE 3 – SPRING CORE.

AGENDA

- BEAN SCOPES
- EAGER VERSUS LAZY INSTANTIATION
- ASPECTS

BEAN SCOPES

- Simple bean scopes:
  - singleton
  - prototype

- Web aware bean scopes:
  - request
  - session
  - application

SINGLETON SCOPE

- Singleton beans:
  - the default Spring bean scope
  - one instance per Spring context
  - used for immutable beans because they are shared among multiple concerns
  - autowiring by constructor helps obtaining immutable beans

PROTOTYPE SCOPE

- Prototype beans:
  - one instance per every usage of the bean
  - may be mutable
  - @Scope(BeanDefinition.SCOPE_PROTOTYPE)
  - don’t inject a prototype bean into a singleton bean (the prototype bean will be created only once)

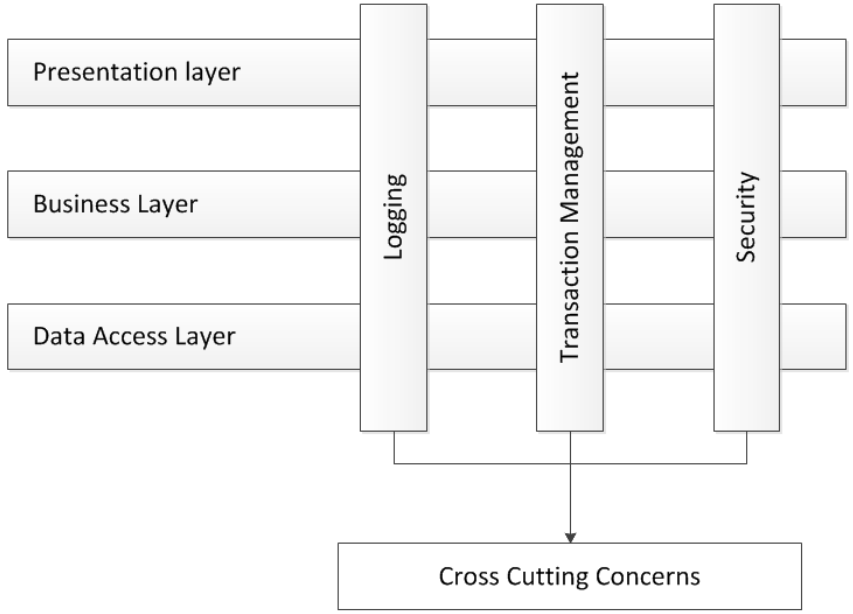
EAGER VERSUS LAZY INSTANTIATION

- Eager instantiation:
  - Spring creates the bean when the context is created
  - default approach
  - better performance
  - fail-fast approach for bean creation

- Lazy instantiation:
  - Spring creates the bean when it is first needed
  - @Lazy
  - avoids creation of unnecessary beans for certain parts of the application

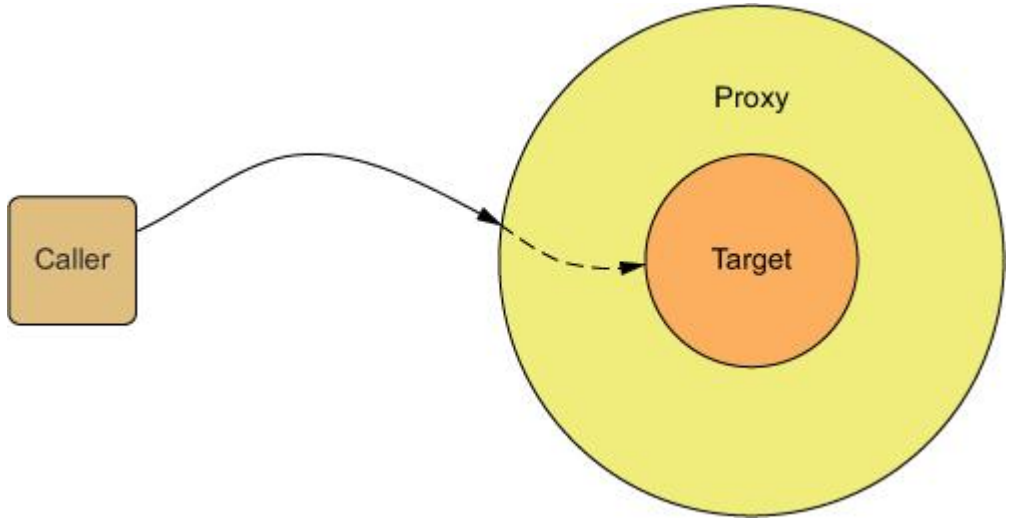
ASPECTS

- AOP (Aspect Oriented Programming)
- a way in which a framework intercepts methods calls and possibly alters the execution of methods
- decouples the code
- modularization of cross cutting concerns
- some Spring implementations of aspects:
  - transactions
  - security
  - caching



ASPECTS

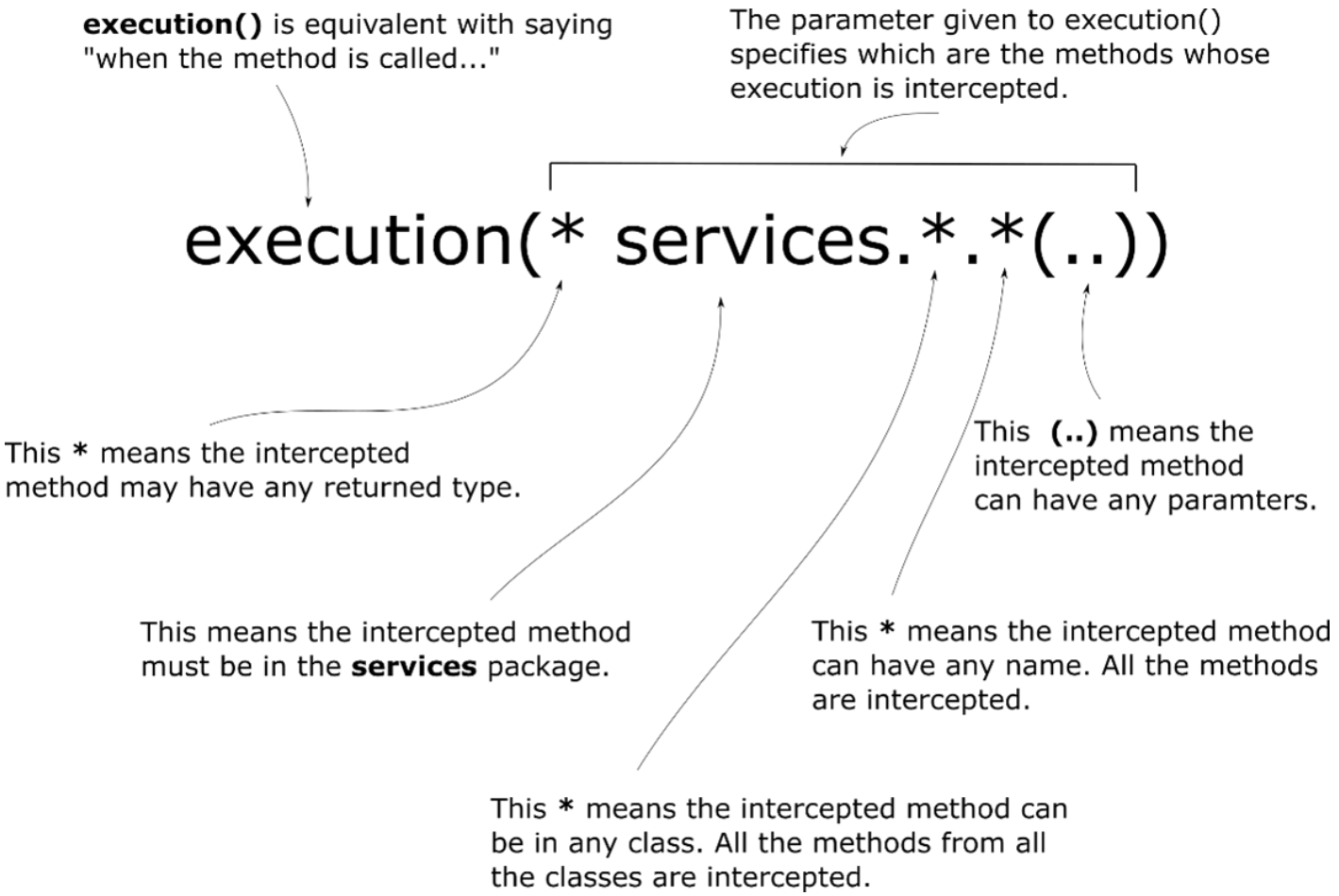
- Aspect: logic the framework executes when you call specific methods.
- Key concepts:
  - the object containing the methods you want to enrich with functionality -> **target object**
  - the logic you want Spring to execute when you call specific methods -> **aspect**
  - when should Spring execute this logic (before the method call, after the method call etc) -> **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

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

ASPECTS



ASPECTS

- Advices:
  - **@Around** – calls the aspect logic before, after or instead the execution of the intercepted method
  - **@Before** – calls the aspect logic before the execution of the intercepted method.
  - **@AfterReturning** – calls the aspect logic after the method successfully returns. The aspect method isn't called if the intercepted method throws an exception.
  - **@AfterThrowing** – calls the aspect logic if the intercepted method throws an exception.
  - **@After** – calls the aspect logic always after the intercepted method execution, both if the method successfully returned or threw an exception.

BIBLIOGRAPHY

- Aspect Oriented Programming with Spring
- Youtube Spring playlist, by Laurentiu Spilca
- Spring in Action, by Craig Walls
- Spring Aspect, by Ramnivas Laddad

THANK YOU!