Laborator 8: AOP Introducere

CONFIG CLASS

1. Importați proiectul LAB8_START.
2. Creați pachetul com.apbbdoo.lab8.config
3. Adăugați clasa Lab8Config:

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.apbdoo.lab8")
public class Lab8Config {

}
```

@Configuration indică o clasă care permite declararea uneia sau mai multor metode adnotate @Bean și care va fi utilizată de către container pentru inițializarea bean-urilor

@ComponentScan definește pachetele în care containerul va căuta configurarea bean-urilor.

@EnableWebMvc este adnotarea echivalentă tag-ului <mvc:annotation-driven /> care importă configurarea Spring MVC din WebMvcConfigurationSupport.

4. Configurați în clasa Lab8Config un bean de tipul ViewResolver

```
@Bean(name = "templateEngine")
    public SpringTemplateEngine templateEngine(ServletContext
servletContext){
        SpringTemplateEngine springTemplateEngine = new
SpringTemplateEngine();
        springTemplateEngine.setDialect(new
SpringStandardDialect());

springTemplateEngine.setTemplateResolver(templateResolver(servlet
Context));
        return springTemplateEngine;
    }
```

```
@Bean(name = "ServletContextTemplateResolver")
    public ServletContextTemplateResolver
templateResolver(ServletContext servletContext){
        ServletContextTemplateResolver templateResolver = new
ServletContextTemplateResolver(servletContext);
        templateResolver.setPrefix("WEB-INF/views/");
        templateResolver.setSuffix(".html");
        templateResolver.setTemplateMode("HTML5");
        return templateResolver;

    }
```

```
@Bean(name = "viewResolver")
    public ViewResolver viewResolver(ServletContext
servletContext){
        ThymeleafViewResolver thymeleafViewResolver = new
ThymeleafViewResolver();
        thymeleafViewResolver.setCharacterEncoding("UTF-8");

thymeleafViewResolver.setTemplateEngine(templateEngine(servletCon
text));
        return thymeleafViewResolver;
    }
```

5. Adăugați clasa ServletConfig

```
public class ServletConfig extends
AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { Lab8Config.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }
}
```

6. Adăugați clasa de configurare pentru proxy-ul AOP:

```
@Configuration
@EnableAspectJAutoProxy
@ComponentScan(basePackages = "com.apbdoo.lab8")
public class AOPConfig {

}
```

7. Adaugați pachetul com.apbdoo.aspects și clasa com.apbdoo.aspects.LoggingAspect. Adăugați adnotările necesare.

```
@Slf4j
public class LoggingAspect {
    public void beforeSaveCustomerAdvice(){
        log.info("before add account ...");
    }
}
```

8. Adaugați un advice de tip @Before care să determine execuția unei metode care să înregistreze un mesaj de info doar pentru apelarea metodei saveCustomer a clasei ServiceCustomerImpl.

9. Adaugați un advice de tip @Around care să utlizeze la expresia pointcut wildcardul *.

10. Adăugați metodelor create argumente de tip JoinPoint pentru a loga signatura metodelor apelate și argumentele acestora.

11. Ordonați aspectele astfel încât cele care folosesc parametrul JoinPoint să se execute primele. Se vor crea clase separate adnotate cu @Order.