Spring<XML> est mort

Vive Spring NoXML

par Gildas Cuisinier
@gcuisinier

[sfeir]

whois(@gcuisinier)

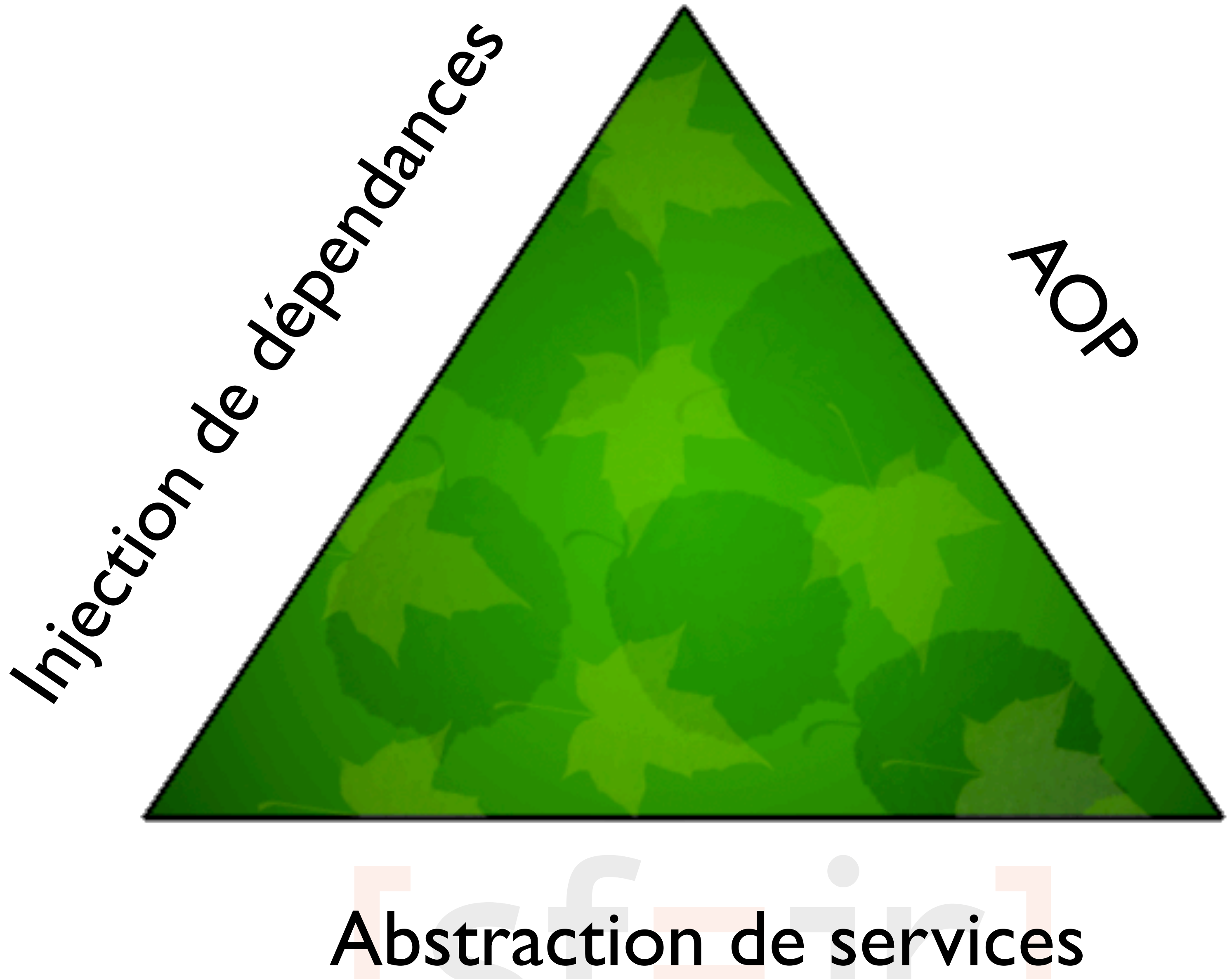
Maître Artisan Développeur, Sfeir Benelux

Initiateur de la section Spring de @Developpez.com

Relecteur de plusieurs livres sur Spring

Evangéliste Spring (JUGs, Devoxx France)

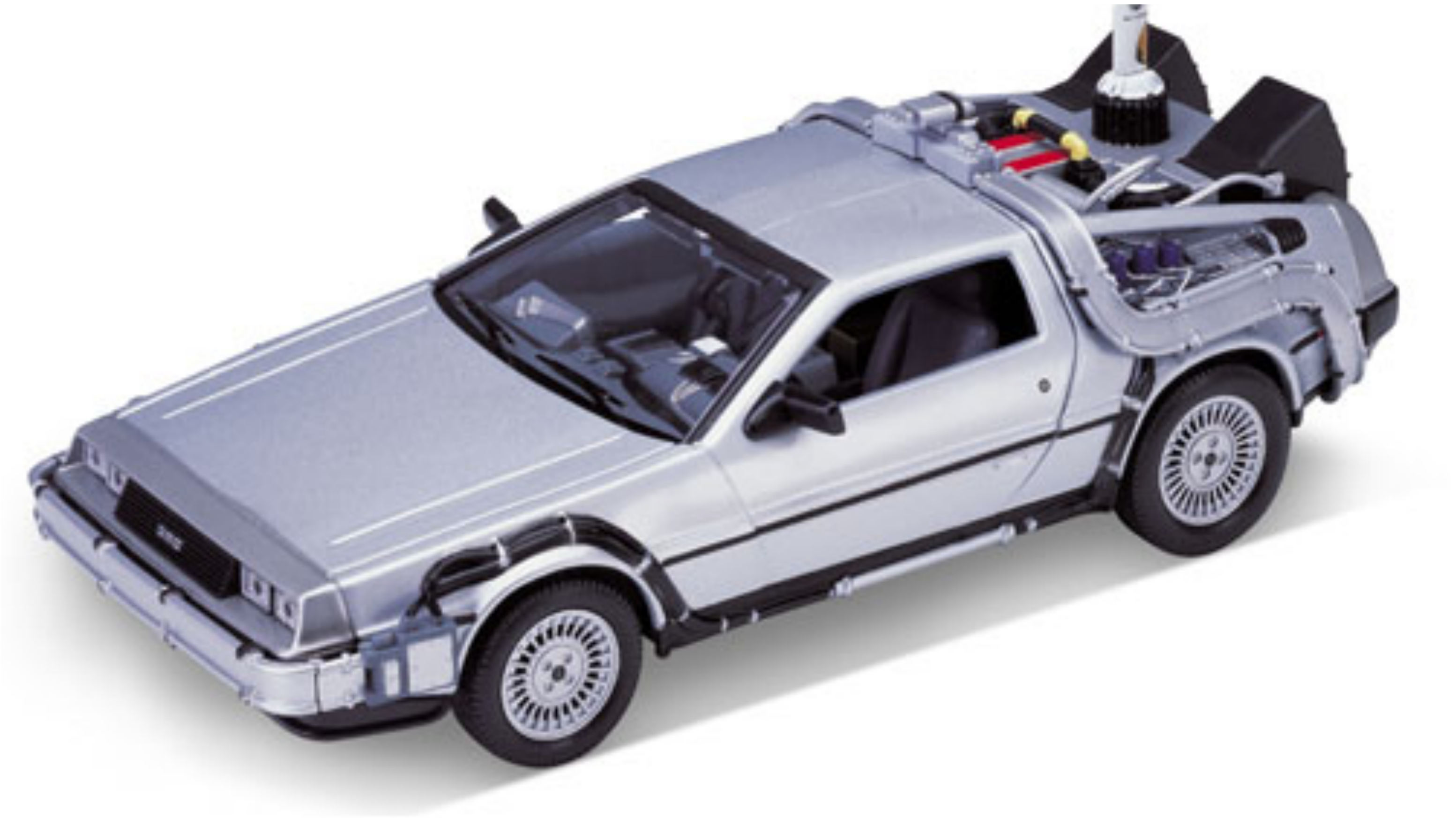
Spring ?



Injection de dépendances

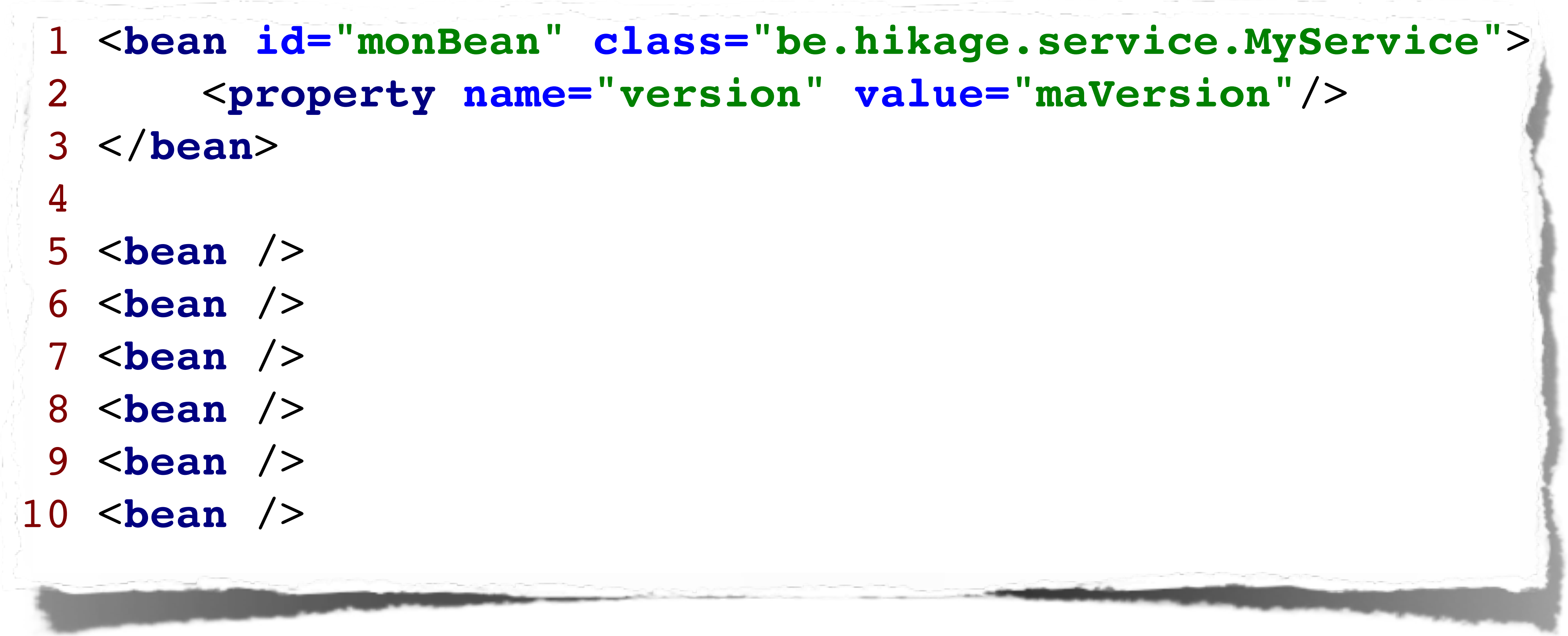
AOP

Abstraction de services



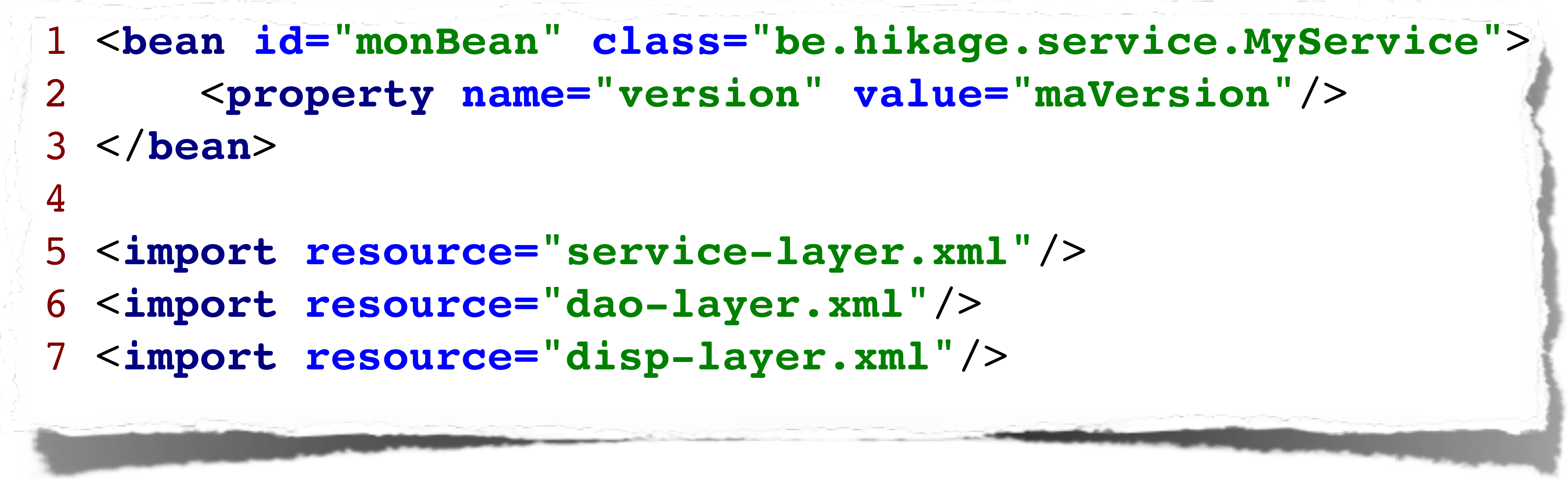
[Retour en 2004]

Spring 1.0



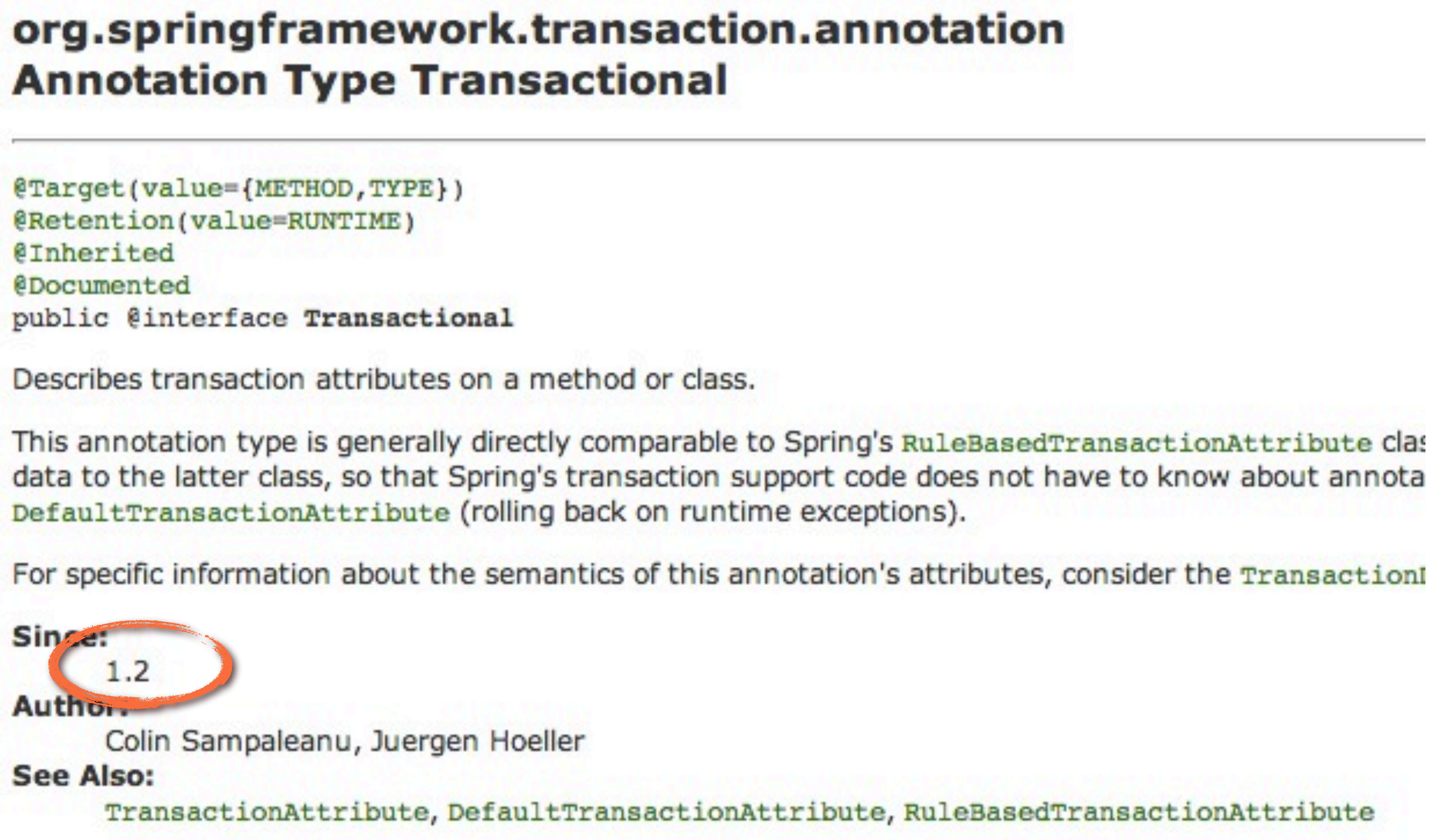
```xml
<bean id="monBean" class="be.hikage.service.MyService">
    <property name="version" value="maVersion"/>
</bean>

<bean />
<bean />
<bean />
<bean />
<bean />
<bean />
```

Spring 1.2



```xml
<bean id="monBean" class="be.hikage.service.MyService">
    <property name="version" value="maVersion"/>
</bean>

<import resource="service-layer.xml"/>
<import resource="dao-layer.xml"/>
<import resource="disp-layer.xml"/>
```

Spring 1.2



org.springframework.transaction.annotation
Annotation Type Transactional

```
@Target(value={METHOD,TYPE})
@Retention(value=RUNTIME)
@Inherited
@Documented
public @interface Transactional
```

Describes transaction attributes on a method or class.

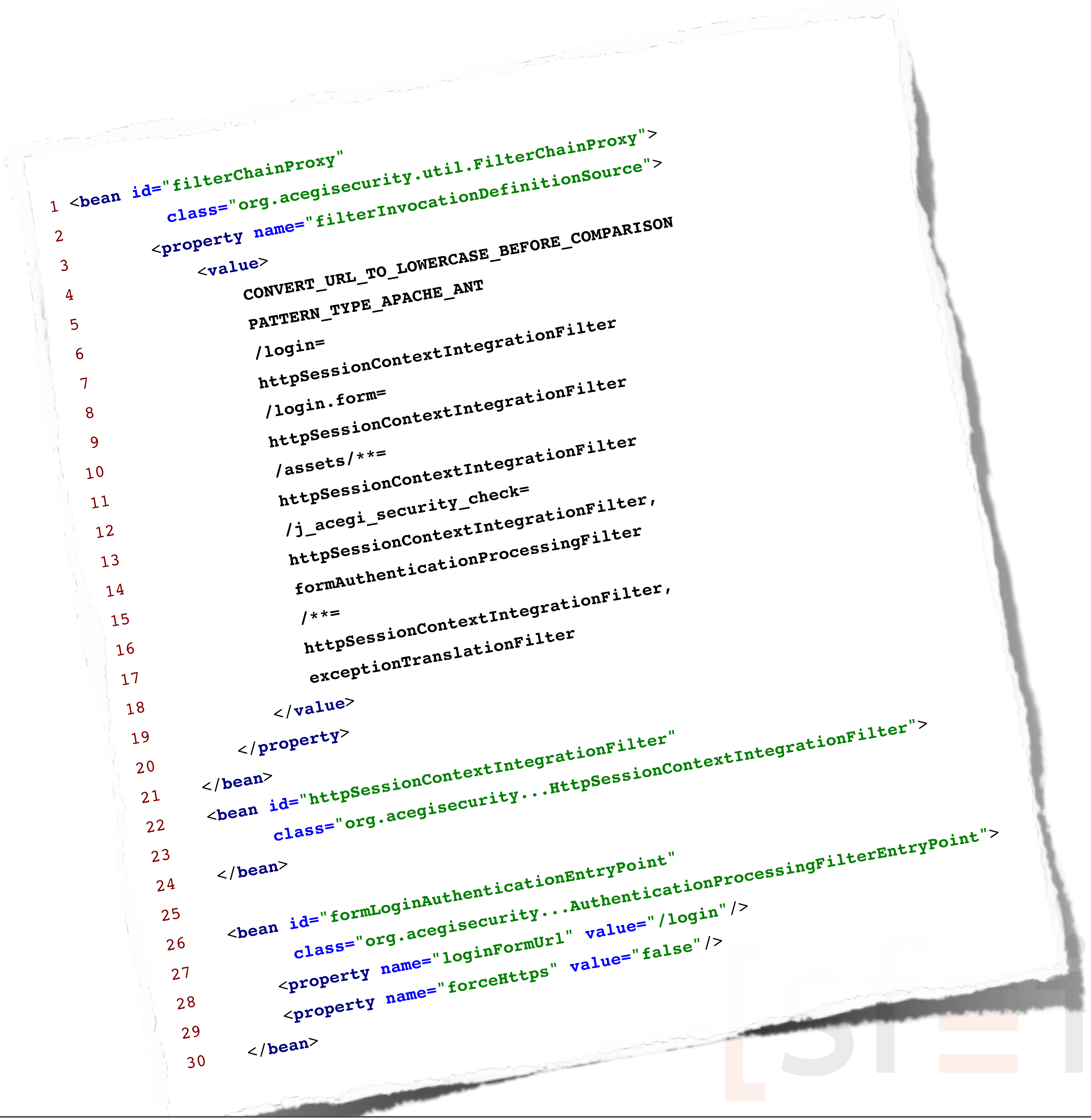
This annotation type is generally directly comparable to Spring's `RuleBasedTransactionAttribute` clas data to the latter class, so that Spring's transaction support code does not have to know about annota `DefaultTransactionAttribute` (rolling back on runtime exceptions).

For specific information about the semantics of this annotation's attributes, consider the `Transaction`

Since:
1.2

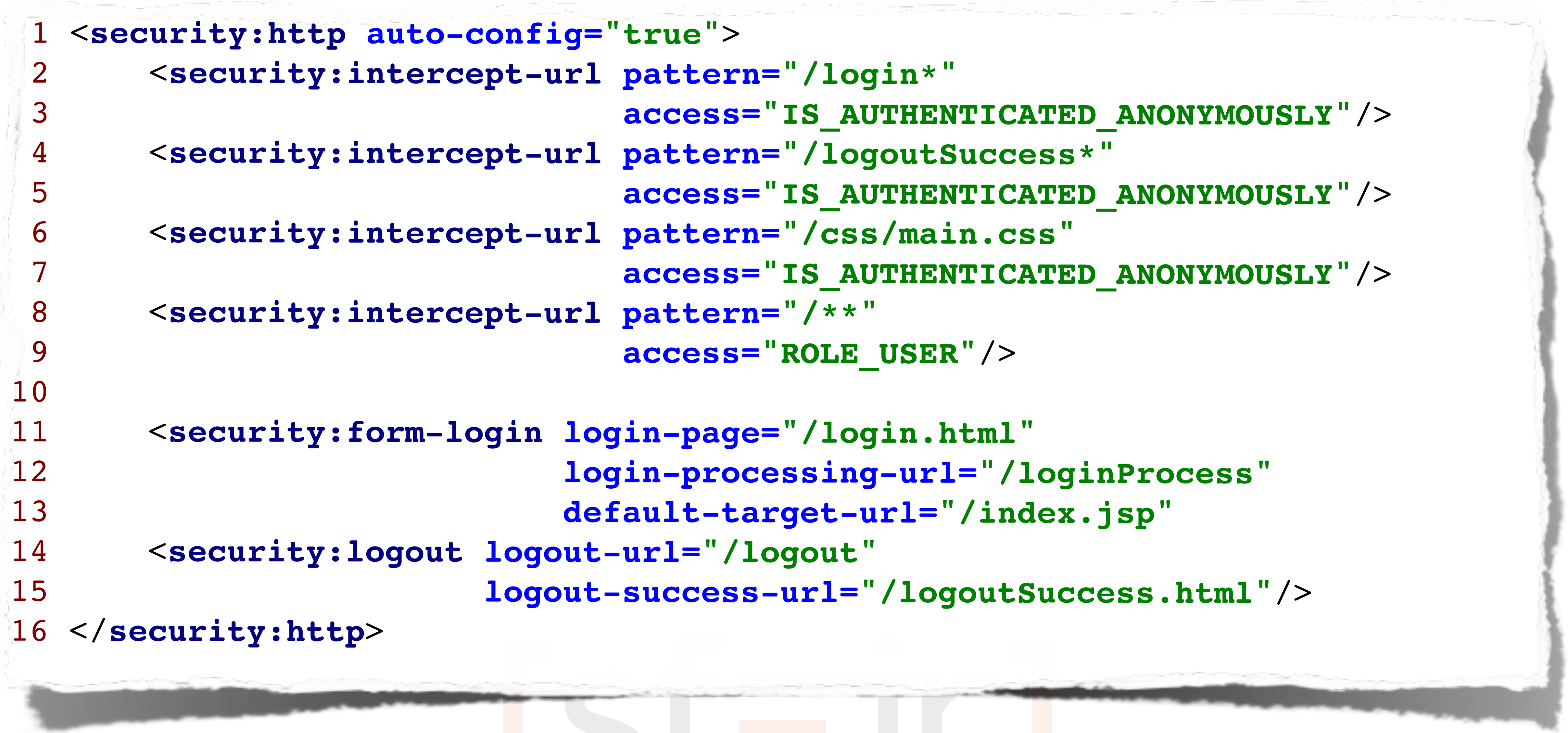
Author:
Colin Sampaleanu, Juergen Hoeller

See Also:
`TransactionAttribute`, `DefaultTransactionAttribute`, `RuleBasedTransactionAttribute`



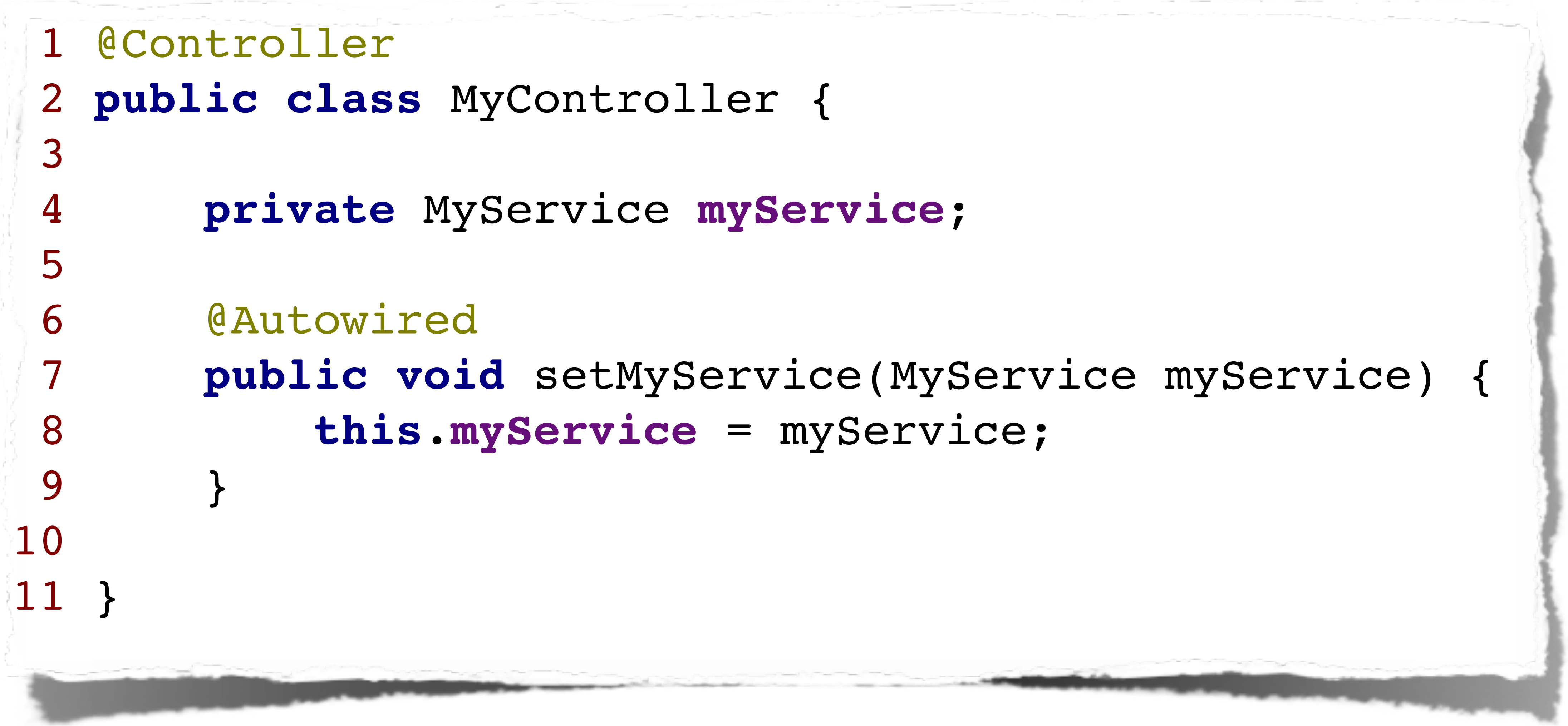
```xml
<bean id="filterChainProxy"
    class="org.acegisecurity.util.FilterChainProxy">
    <property name="filterInvocationDefinitionSource">
        <value>
            CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
            PATTERN_TYPE_APACHE_ANT
            /login=
            httpSessionContextIntegrationFilter
            /login.form=
            httpSessionContextIntegrationFilter
            /assets/**=
            httpSessionContextIntegrationFilter
            /j_acegi_security_check=
            httpSessionContextIntegrationFilter,
            formAuthenticationProcessingFilter
            /**=
            httpSessionContextIntegrationFilter,
            exceptionTranslationFilter
        </value>
    </property>
</bean>
<bean id="httpSessionContextIntegrationFilter"
    class="org.acegisecurity...HttpSessionContextIntegrationFilter">
</bean>

<bean id="formLoginAuthenticationEntryPoint"
    class="org.acegisecurity...AuthenticationProcessingFilterEntryPoint">
    <property name="loginFormUrl" value="/login"/>
    <property name="forceHttps" value="false"/>
</bean>
```

Spring 2.0

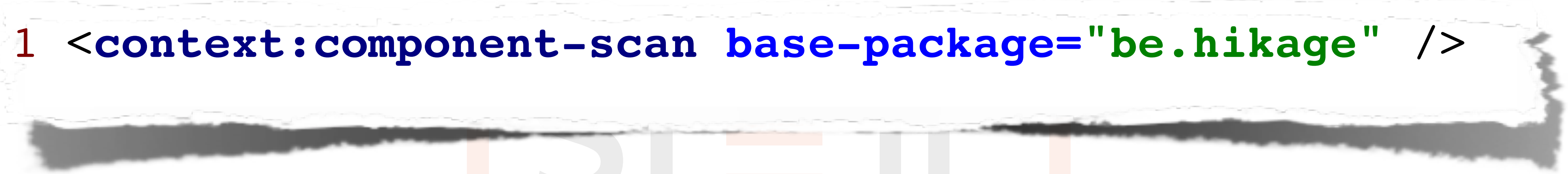


```
<security:http auto-config="true">
    <security:intercept-url pattern="/login*"
                            access="IS_AUTHENTICATED_ANONYMOUSLY"/>
    <security:intercept-url pattern="/logoutSuccess*"
                            access="IS_AUTHENTICATED_ANONYMOUSLY"/>
    <security:intercept-url pattern="/css/main.css"
                            access="IS_AUTHENTICATED_ANONYMOUSLY"/>
    <security:intercept-url pattern="/**"
                            access="ROLE_USER"/>

    <security:form-login login-page="/login.html"
                         login-processing-url="/loginProcess"
                         default-target-url="/index.jsp"
    <security:logout logout-url="/logout"
                     logout-success-url="/logoutSuccess.html"/>
</security:http>
```

Spring 2.5

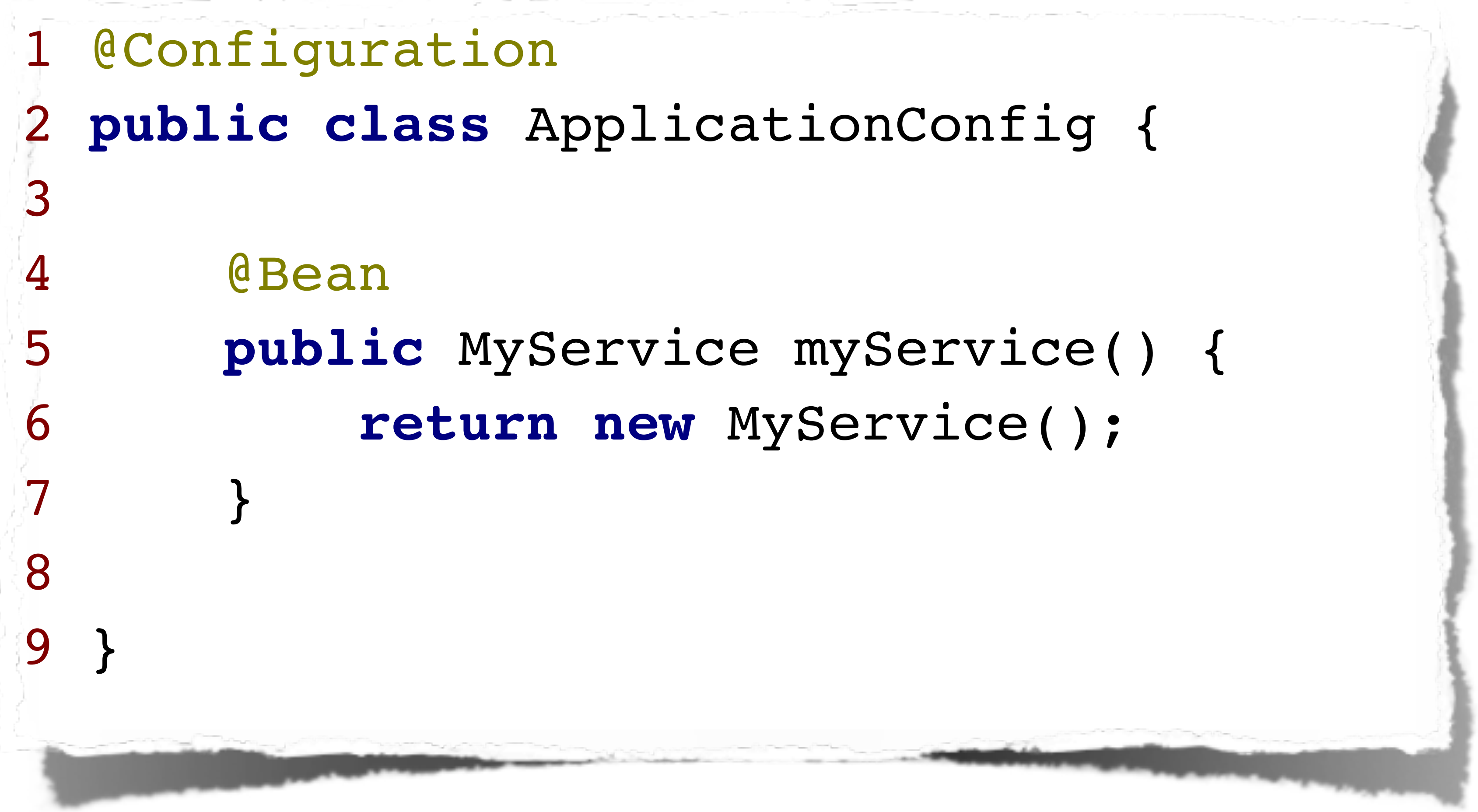


```java
@Controller
public class MyController {

    private MyService myService;

    @Autowired
    public void setMyService(MyService myService) {
        this.myService = myService;
    }

}
```



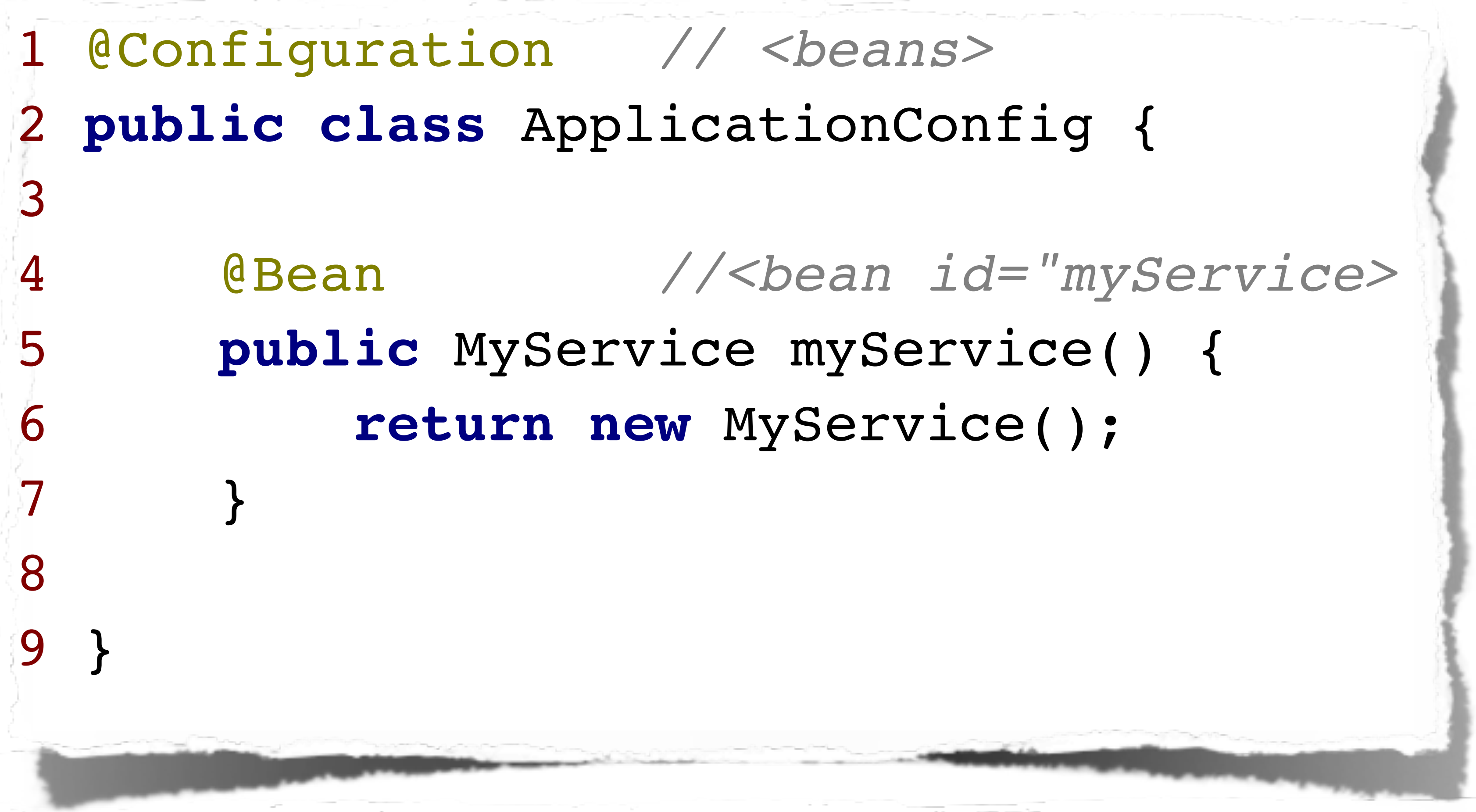
```xml
<context:component-scan base-package="be.hikage" />
```

Spring 3.0

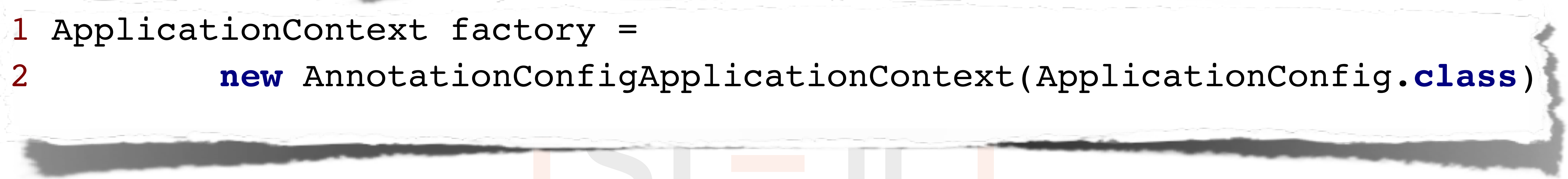


```java
@Configuration
public class ApplicationConfig {

    @Bean
    public MyService myService() {
        return new MyService();
    }

}
```

Spring 3.0



```java
@Configuration    // <beans>
public class ApplicationConfig {

    @Bean          //<bean id="myService>
    public MyService myService() {
        return new MyService();
    }

}
```



```java
ApplicationContext factory =
        new AnnotationConfigApplicationContext(ApplicationConfig.class)
```

Spring 3.0

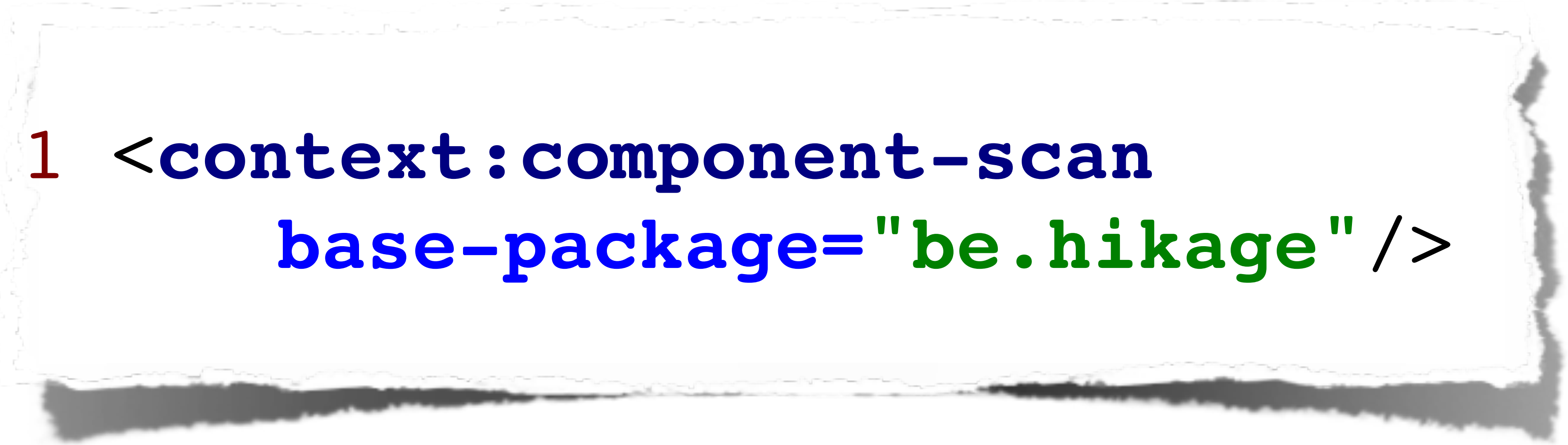
```
@Componen
public                    {

      @S          c           *
      pu          d ex
                  le
      }

}
```

```
<context:co      n          e.hikage" />
<task:annota                   scheduler" .. />
```

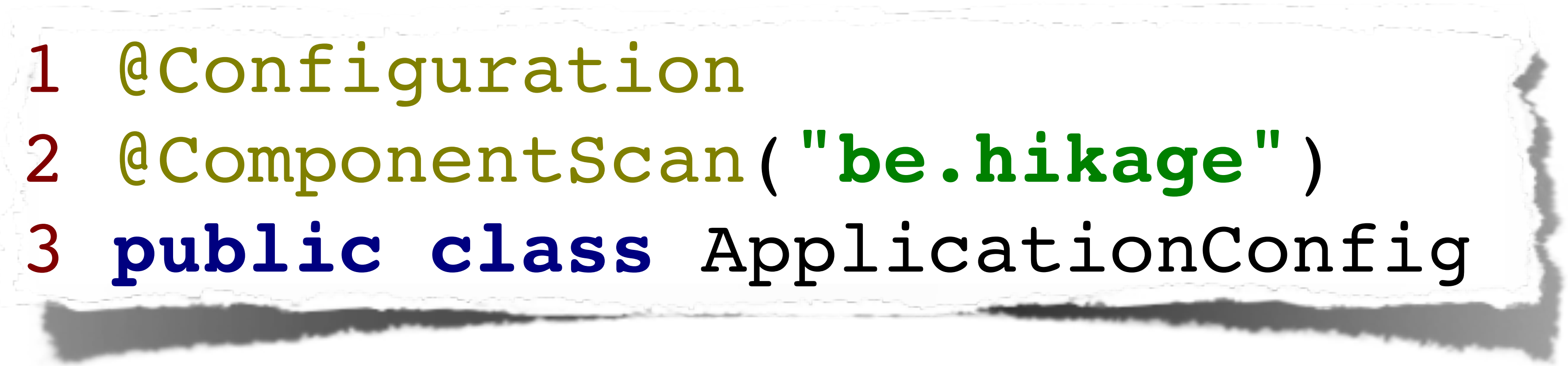
REJECTED • REJECTED • REJECTED •

Spring 3.1
@nnotations ++

@ComponentScan

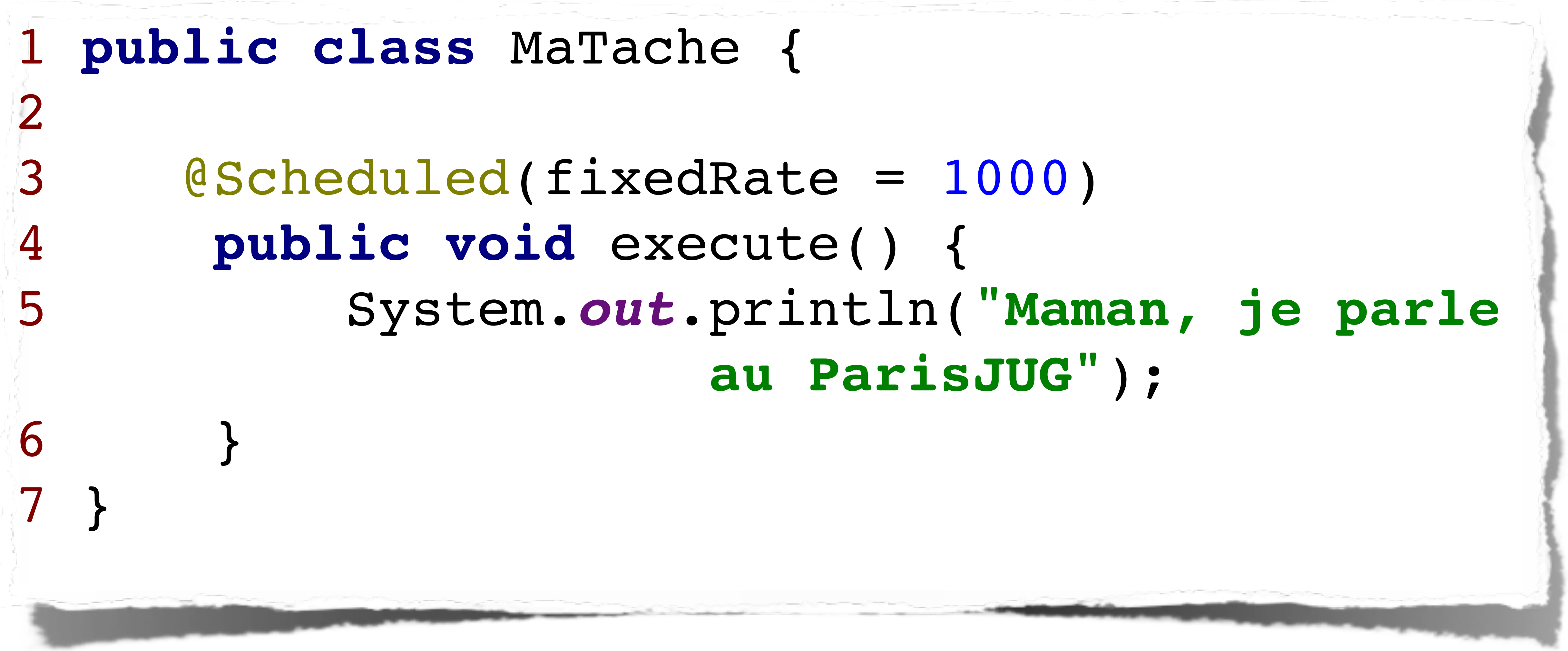


```
<context:component-scan
      base-package="be.hikage"/>
```



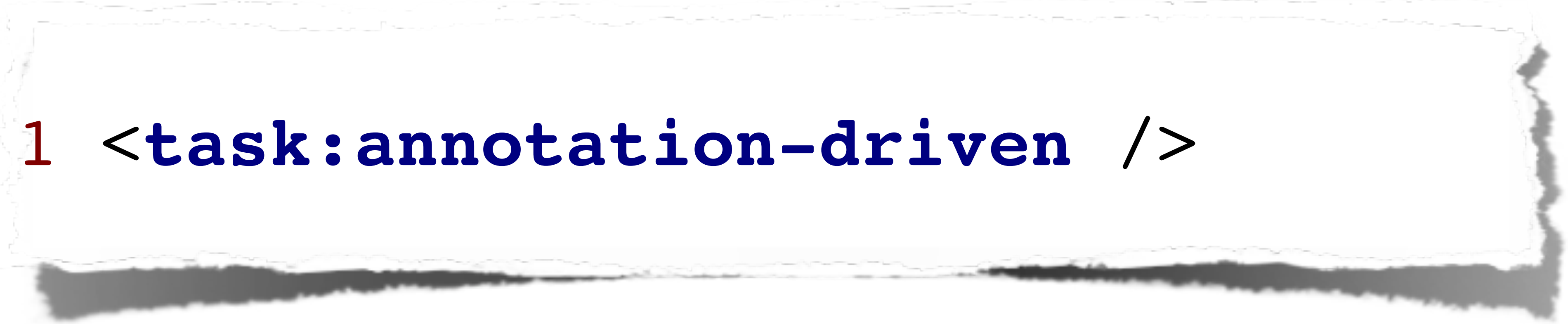
```
@Configuration
@ComponentScan("be.hikage")
public class ApplicationConfig
```

@EnableScheduling

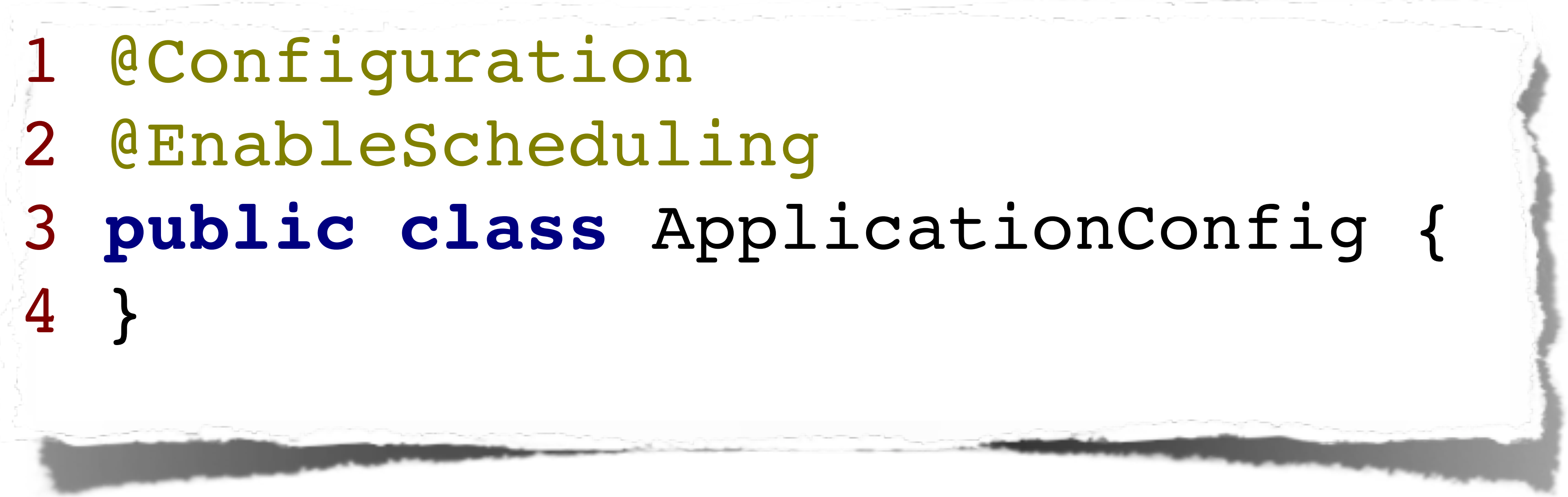


```java
public class MaTache {

    @Scheduled(fixedRate = 1000)
     public void execute() {
          System.out.println("Maman, je parle
                               au ParisJUG");
     }
}
```

@EnableScheduling

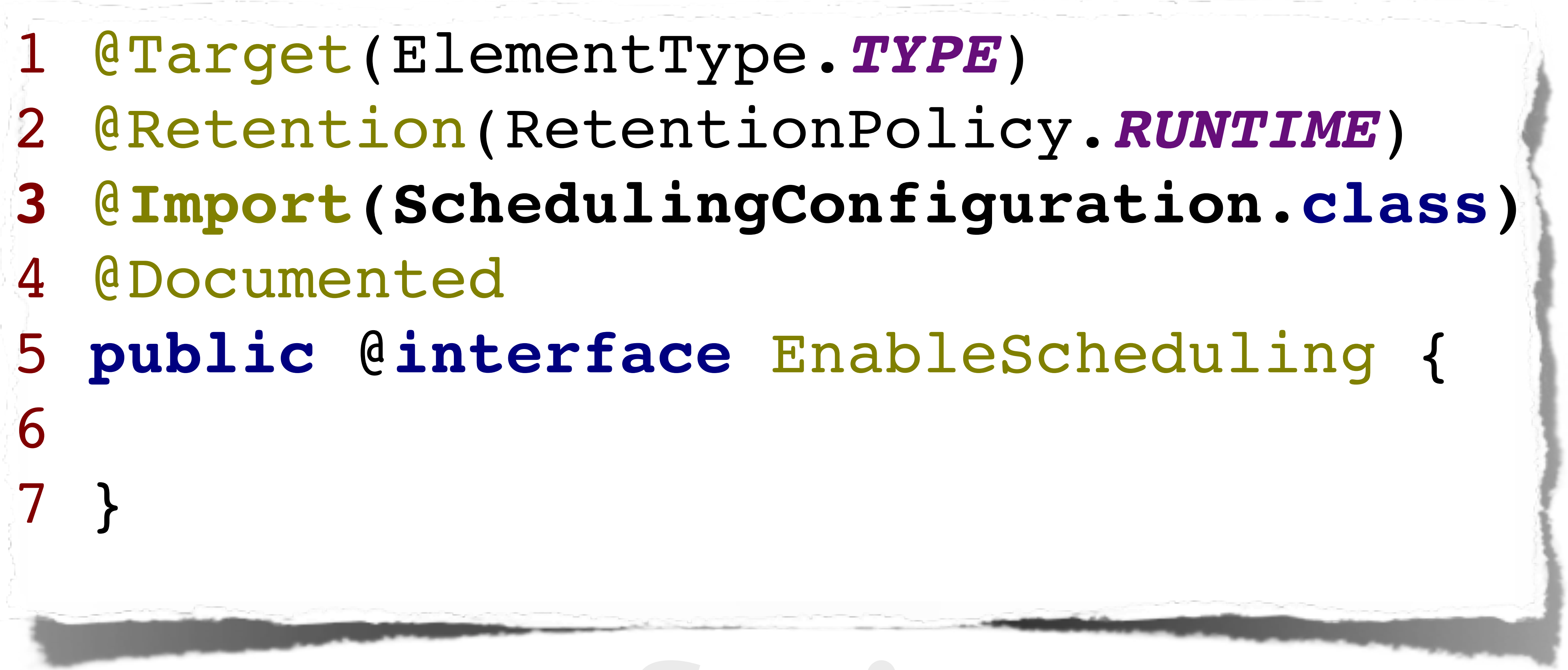


```
<task:annotation-driven />
```



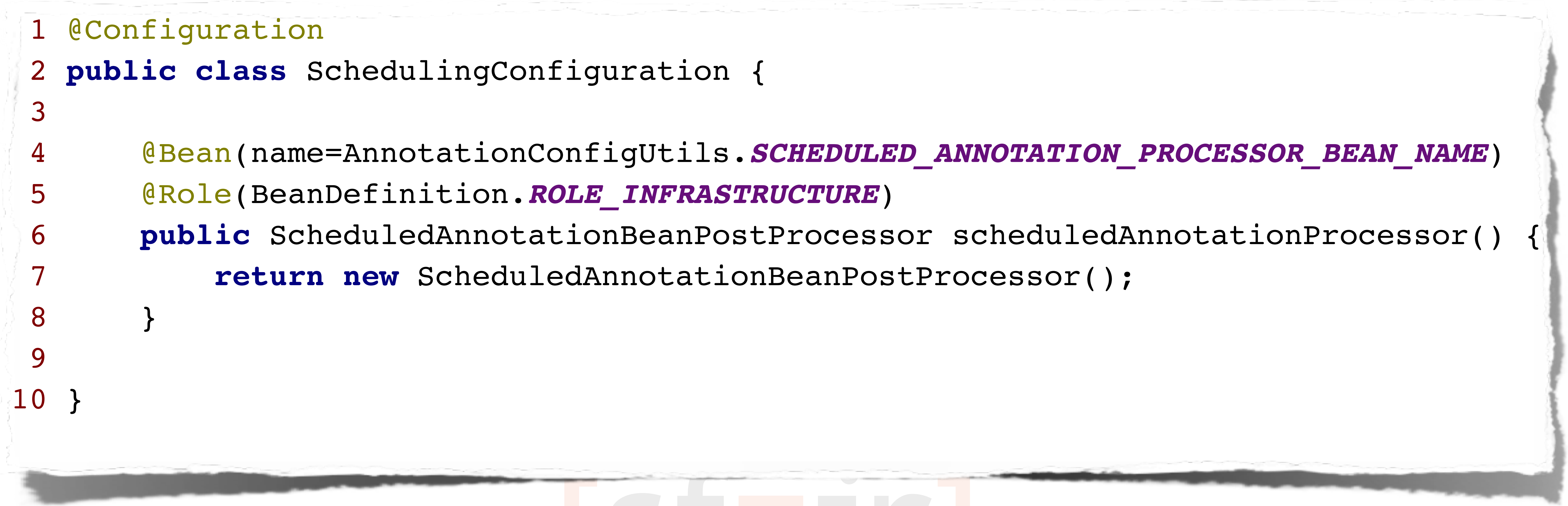
```
@Configuration
@EnableScheduling
public class ApplicationConfig {
}
```

@EnableScheduling



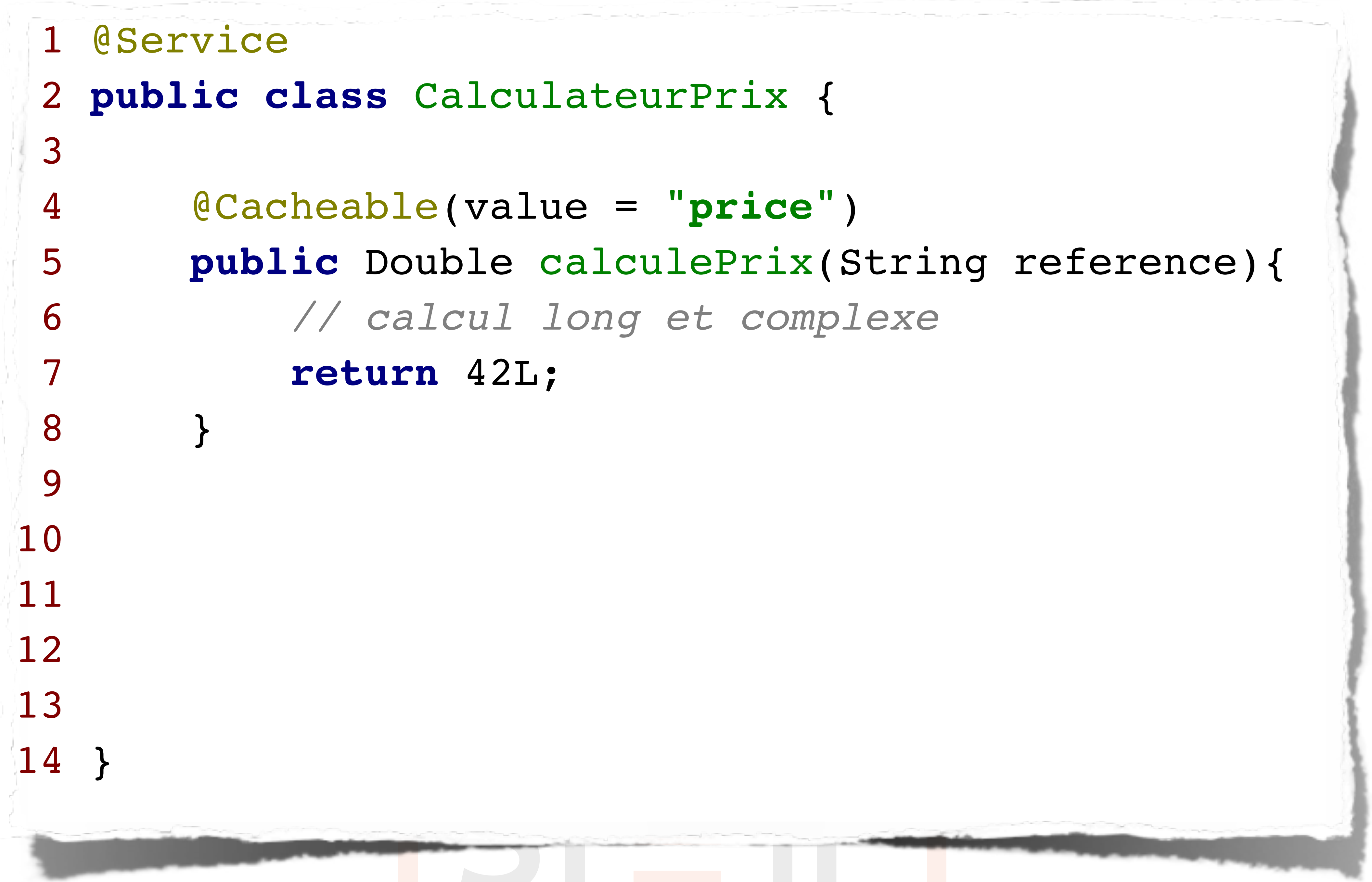
```java
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Import(SchedulingConfiguration.class)
@Documented
public @interface EnableScheduling {

}
```

@EnableScheduling



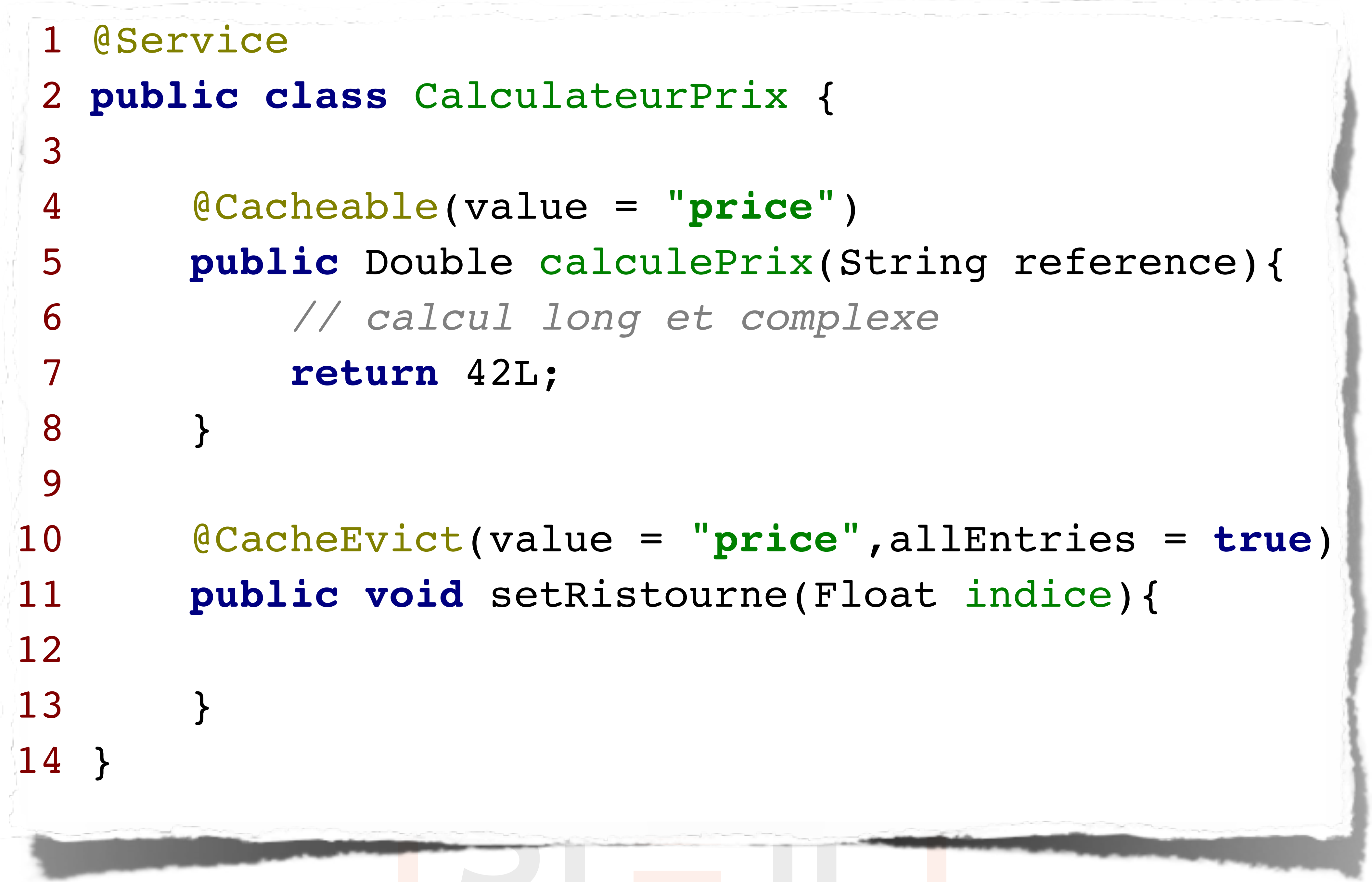
```java
@Configuration
public class SchedulingConfiguration {

    @Bean(name=AnnotationConfigUtils.SCHEDULED_ANNOTATION_PROCESSOR_BEAN_NAME)
    @Role(BeanDefinition.ROLE_INFRASTRUCTURE)
    public ScheduledAnnotationBeanPostProcessor scheduledAnnotationProcessor() {
        return new ScheduledAnnotationBeanPostProcessor();
    }

}
```

@EnableCaching



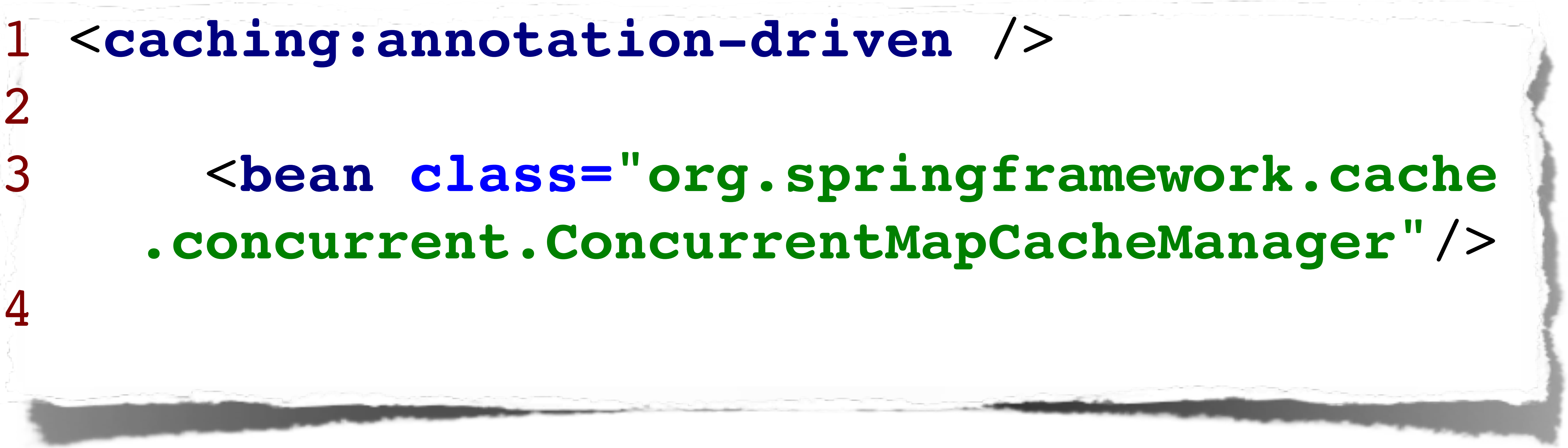
```java
@Service
public class CalculateurPrix {

    @Cacheable(value = "price")
    public Double calculePrix(String reference){
        // calcul long et complexe
        return 42L;
    }





}
```

@EnableCaching



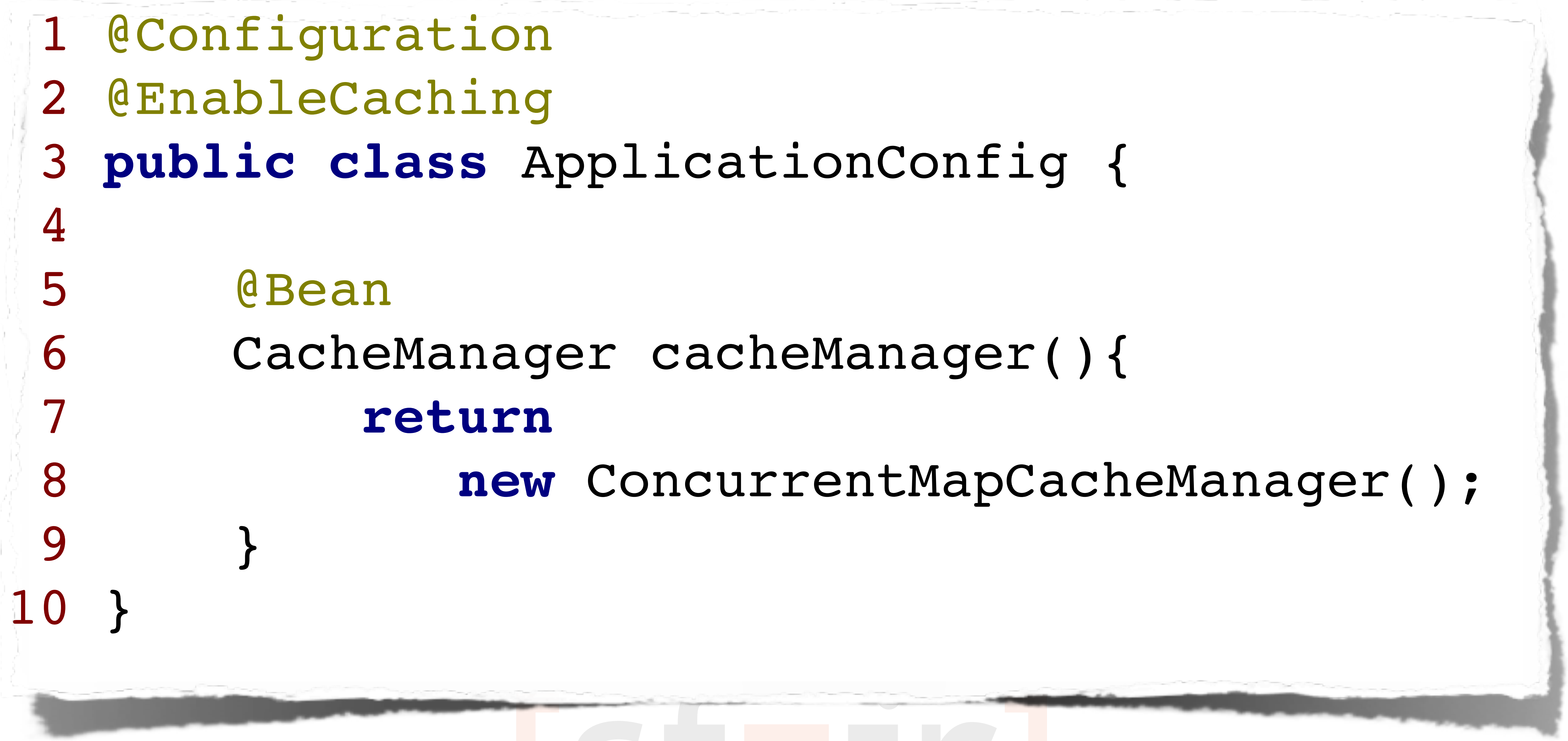
```java
@Service
public class CalculateurPrix {

    @Cacheable(value = "price")
    public Double calculePrix(String reference){
        // calcul long et complexe
        return 42L;
    }

    @CacheEvict(value = "price",allEntries = true)
    public void setRistourne(Float indice){

    }
}
```

@EnableCaching



```
<caching:annotation-driven />

    <bean class="org.springframework.cache.concurrent.ConcurrentMapCacheManager"/>

```

@EnableCaching

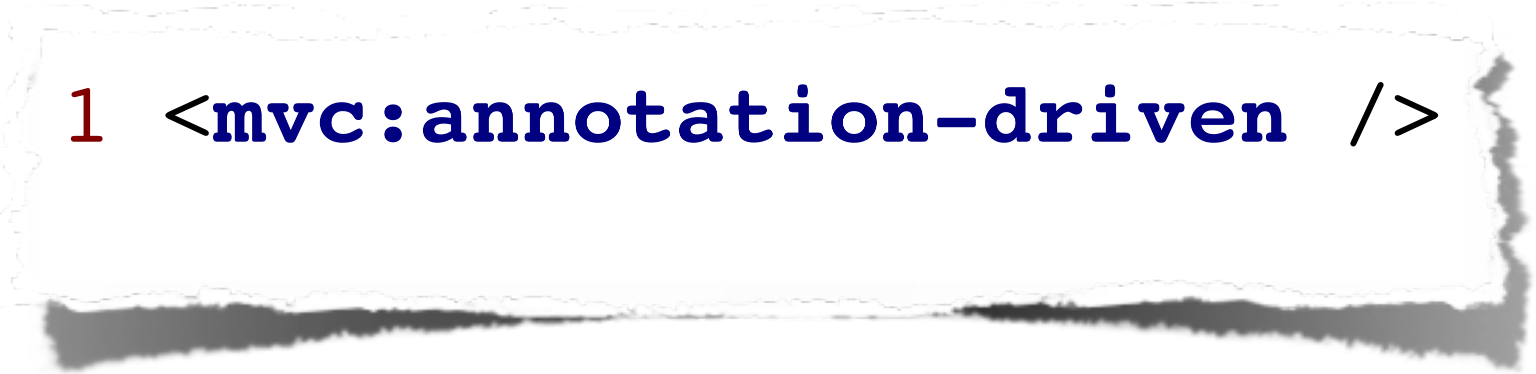


```java
@Configuration
@EnableCaching
public class ApplicationConfig {

    @Bean
    CacheManager cacheManager(){
        return
           new ConcurrentMapCacheManager();
    }
}
```

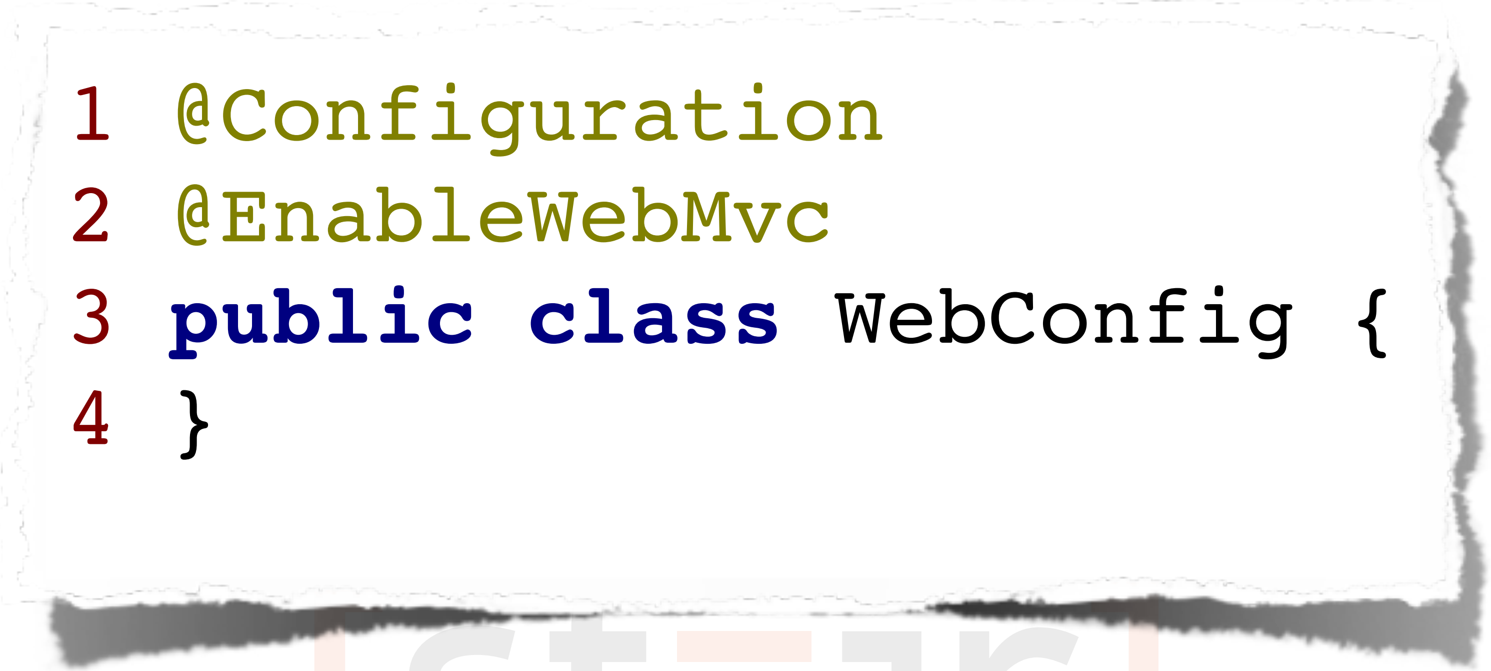
Abstraction de Cache

- Out of the Box
  - ConcurrentHashMap
  - EhCache
- GemFire
- JCache (Spring 3.2/3.3)

@EnableWebMvc

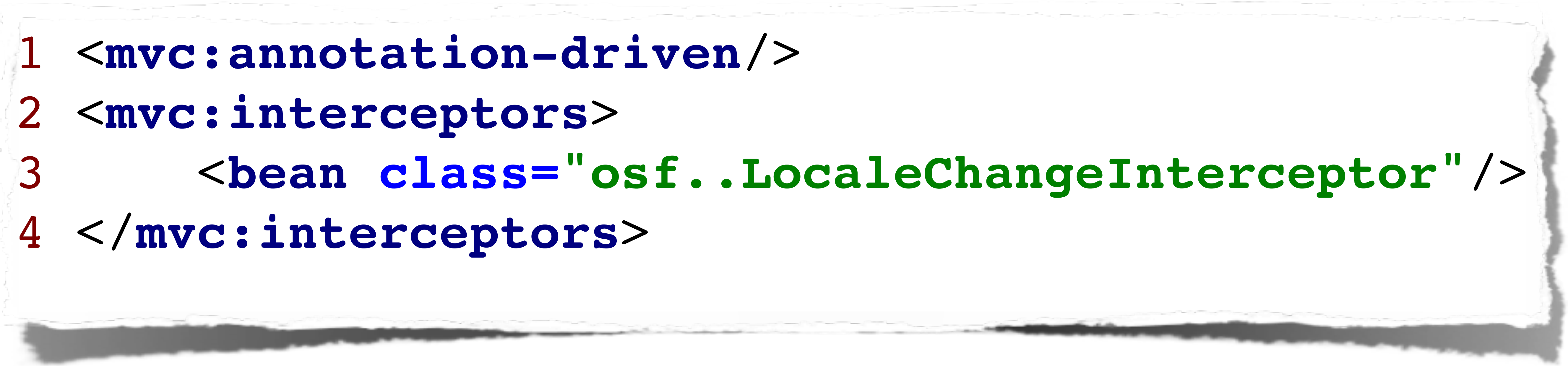


```xml
<mvc:annotation-driven />
```



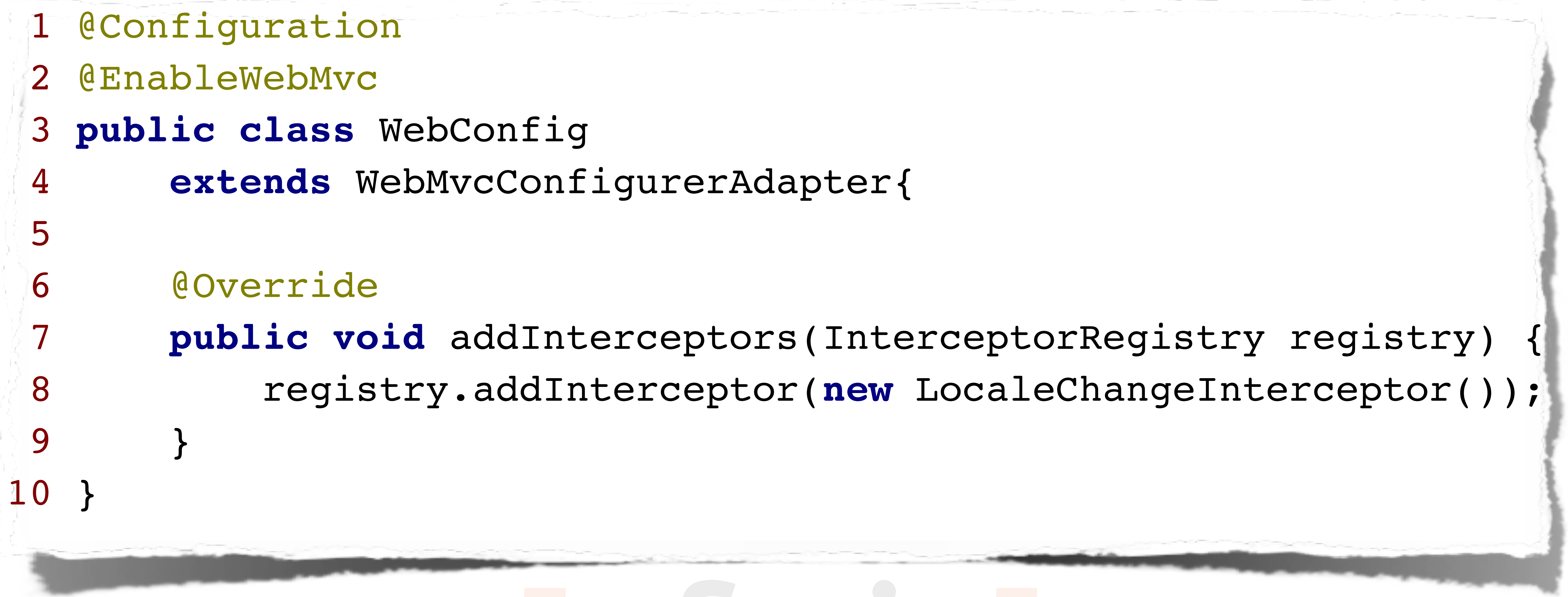
```java
@Configuration
@EnableWebMvc
public class WebConfig {
}
```

@EnableWebMvc



```xml
<mvc:annotation-driven/>
<mvc:interceptors>
    <bean class="osf..LocaleChangeInterceptor"/>
</mvc:interceptors>
```

@EnableWebMvc + WebMvcConfigurerAdapter



```java
@Configuration
@EnableWebMvc
public class WebConfig
    extends WebMvcConfigurerAdapter{

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LocaleChangeInterceptor());
    }
}
```

@EnableWebMvc

☑ Ajout Intercepteur

☑ Configuration de ViewController

☑ Ajout de ResourceHandler

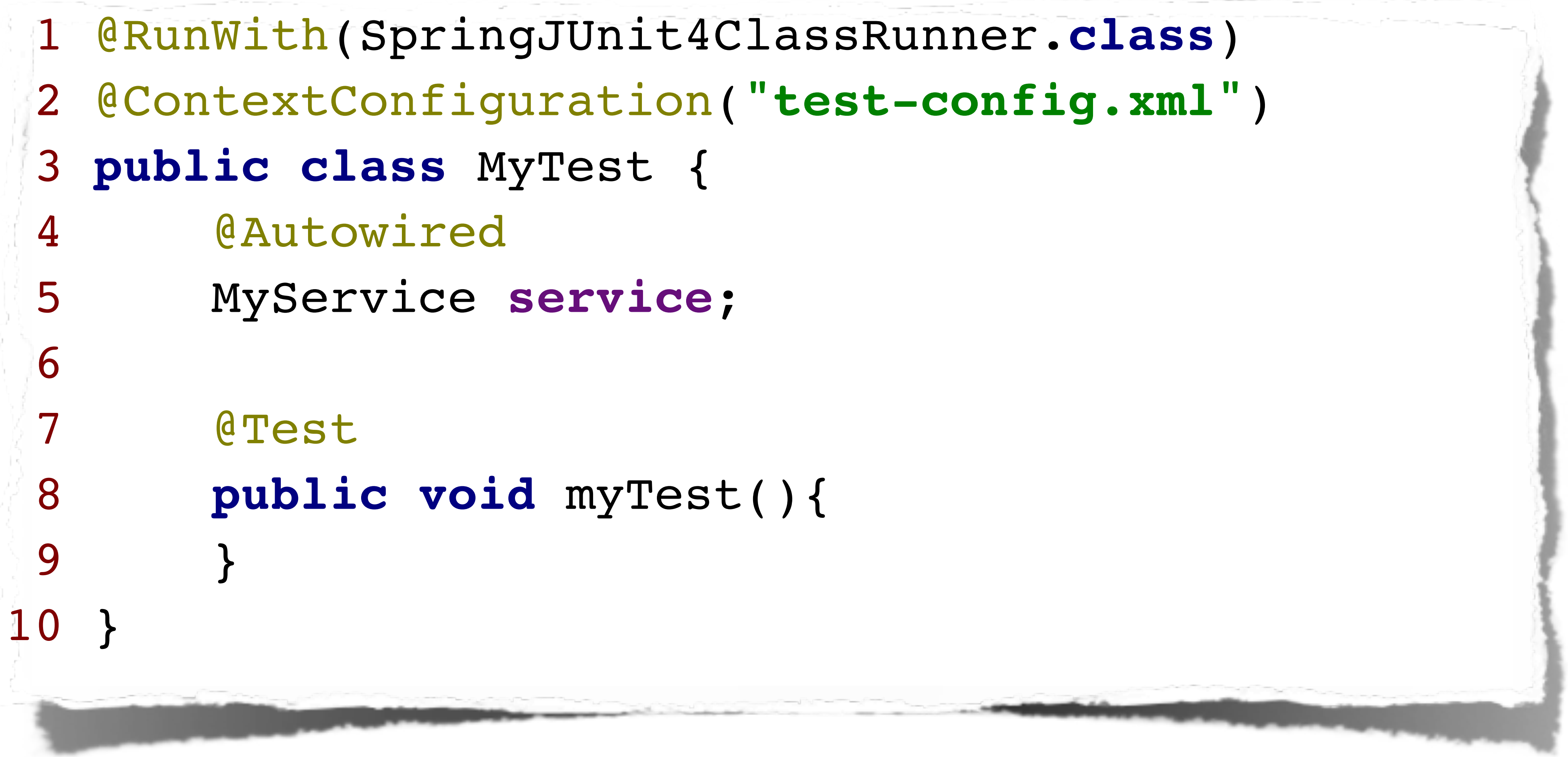
□ Ne permet pas une configuration fine

➡ ~~@EnableWebMvc~~ et étendre WebMvcConfigurationSupport

@Enable*

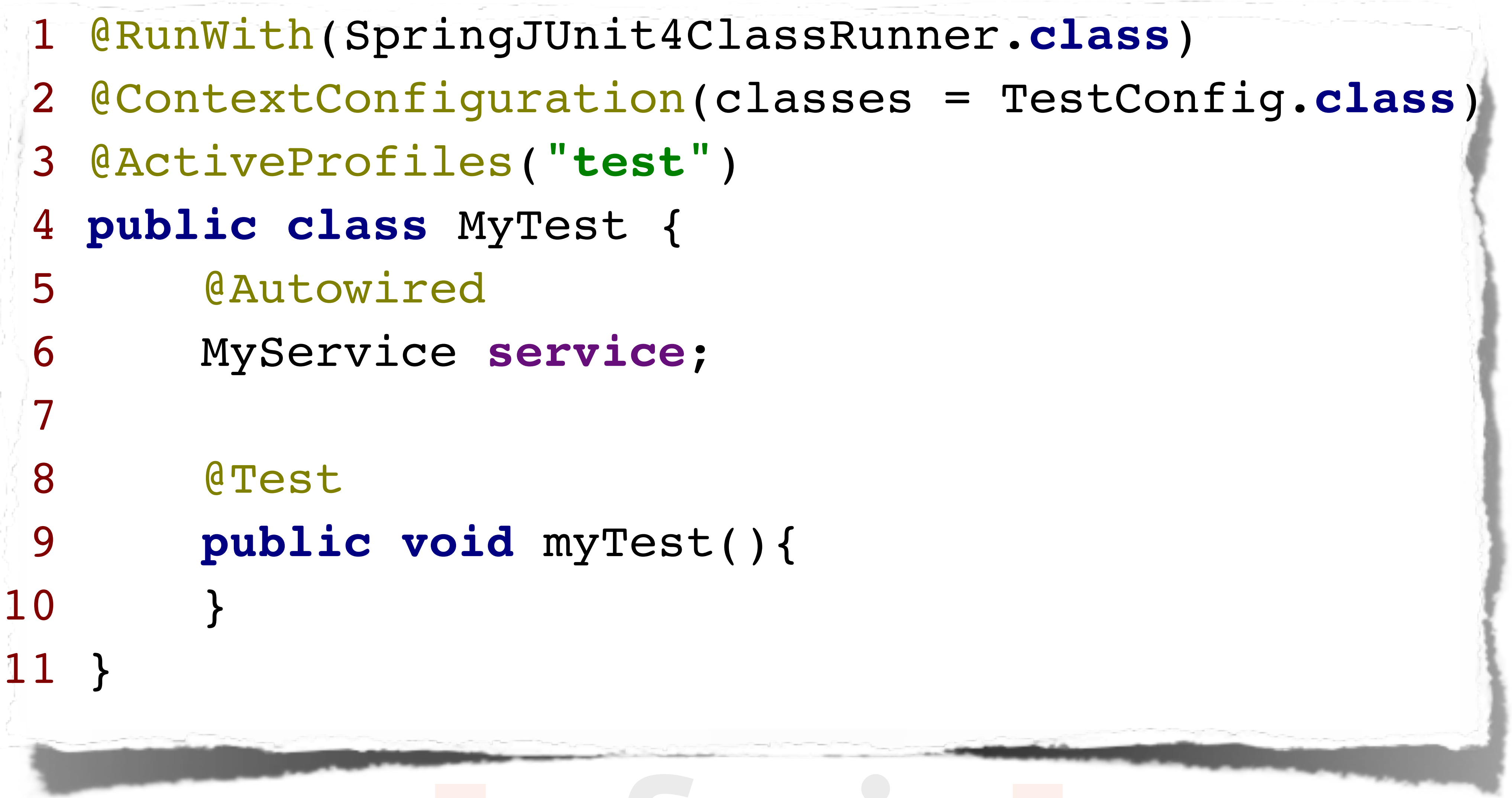
- EnableAsync
- EnableAspectJAutoProxy
- EnableSpringConfigured
- EnableLoadTimeWeaving
- EnableTransactionManagement

Testing 2.5



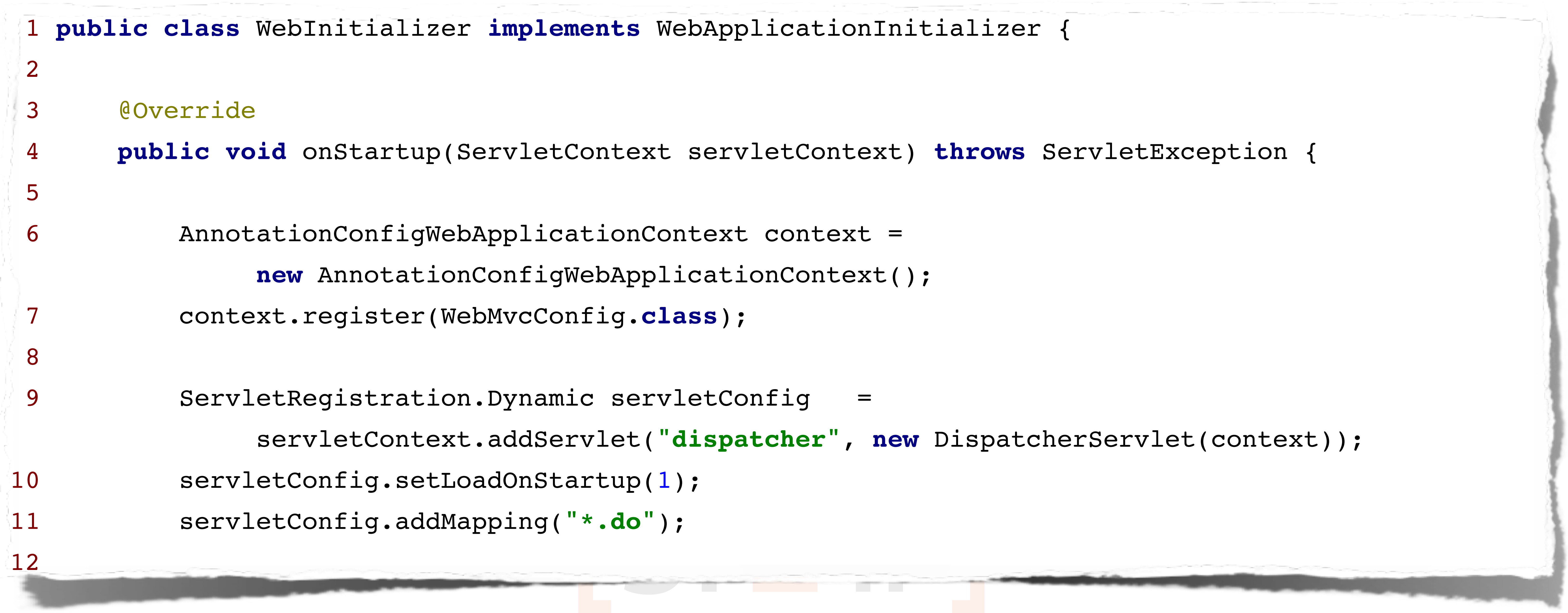
```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("test-config.xml")
public class MyTest {
    @Autowired
    MyService service;

    @Test
    public void myTest(){
    }
}
```

Testing 3.1



```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = TestConfig.class)
@ActiveProfiles("test")
public class MyTest {
    @Autowired
    MyService service;

    @Test
    public void myTest(){
    }
}
```

Servlet 3.0

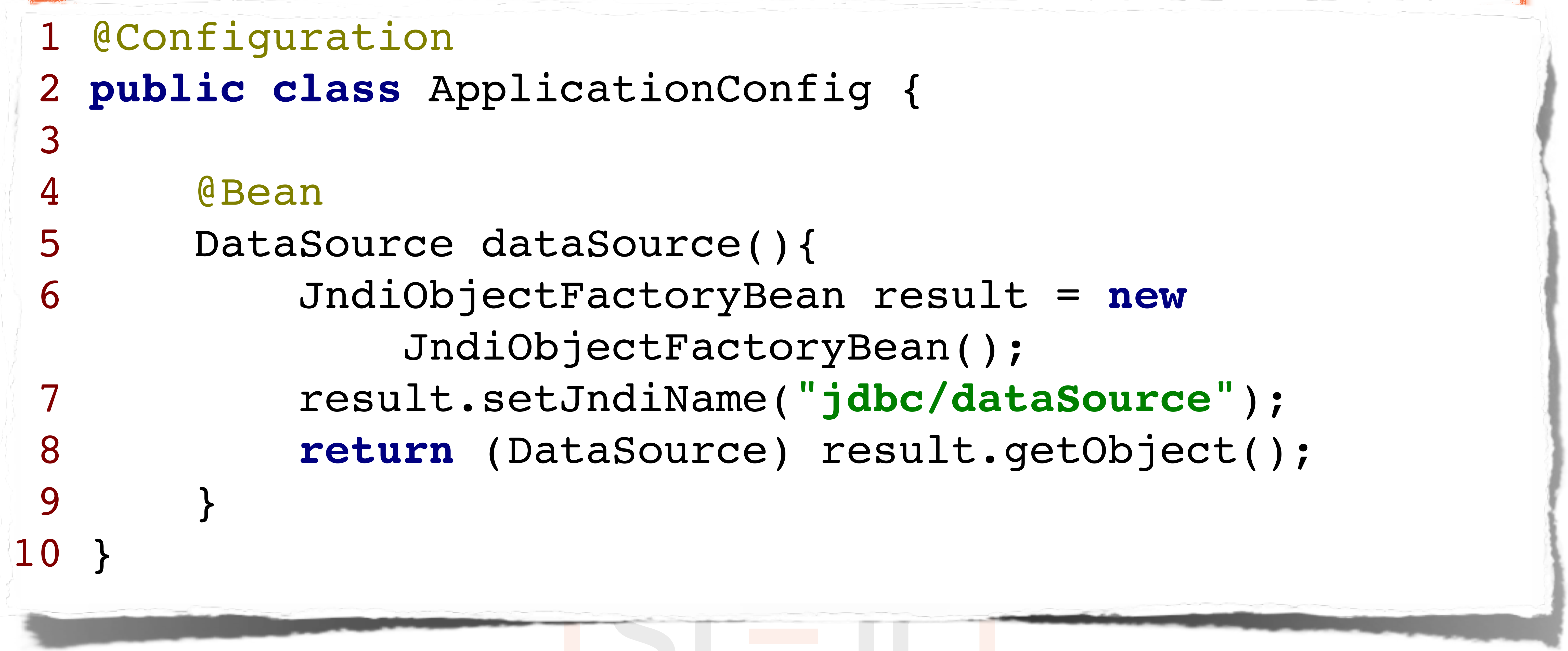


```java
public class WebInitializer implements WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {

        AnnotationConfigWebApplicationContext context =
             new AnnotationConfigWebApplicationContext();
        context.register(WebMvcConfig.class);

        ServletRegistration.Dynamic servletConfig   =
             servletContext.addServlet("dispatcher", new DispatcherServlet(context));
        servletConfig.setLoadOnStartup(1);
        servletConfig.addMapping("*.do");

```

DEMO

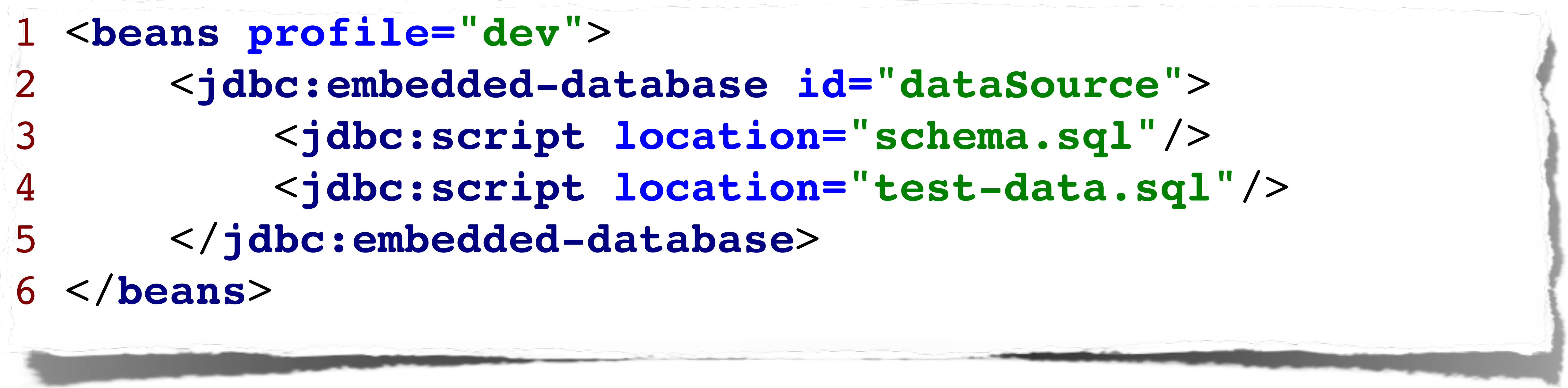
[Spring 3.1 bis]

@Profile



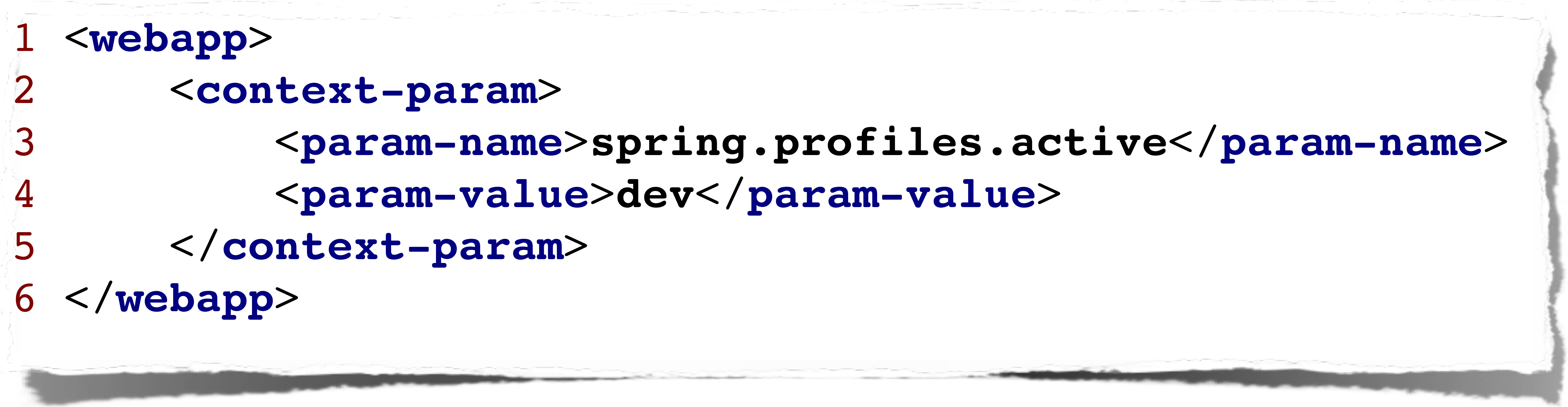
```
@Configuration
public class ApplicationConfig {

    @Bean
    DataSource dataSource(){
        JndiObjectFactoryBean result = new
            JndiObjectFactoryBean();
        result.setJndiName("jdbc/dataSource");
        return (DataSource) result.getObject();
    }
}
```

<Profile />



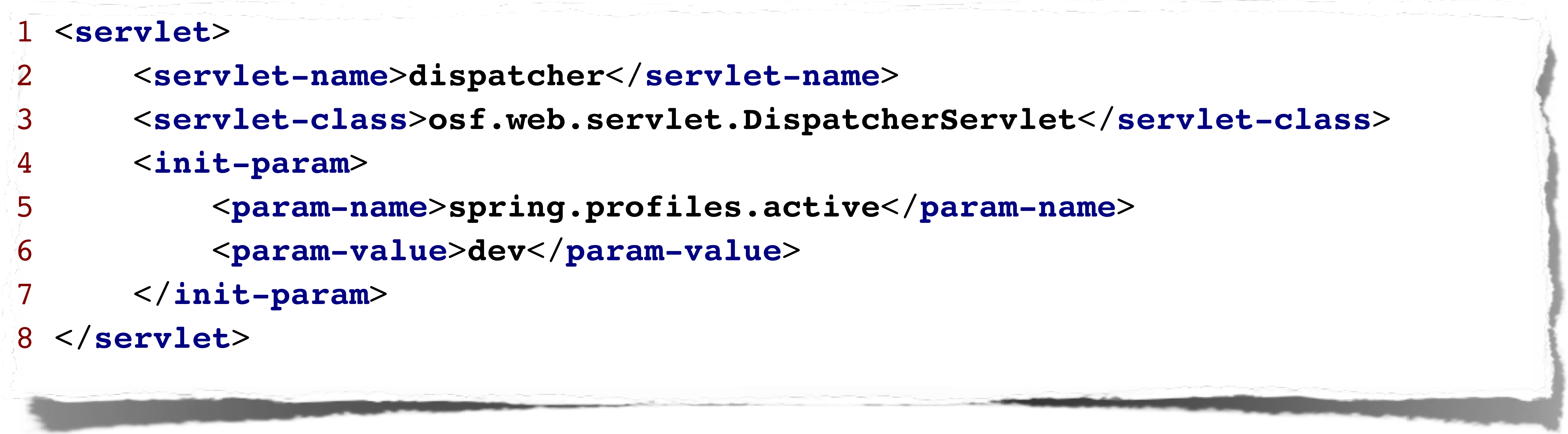
```xml
<beans profile="dev">
    <jdbc:embedded-database id="dataSource">
        <jdbc:script location="schema.sql"/>
        <jdbc:script location="test-data.sql"/>
    </jdbc:embedded-database>
</beans>
```

Activation



```
<webapp>
    <context-param>
        <param-name>spring.profiles.active</param-name>
        <param-value>dev</param-value>
    </context-param>
</webapp>
```

Activation



```xml
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>osf.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>spring.profiles.active</param-name>
        <param-value>dev</param-value>
    </init-param>
</servlet>
```

Spring 3.1, c'est aussi

- Support Hibernate 4
- Namespace c:
- JPA sans Persistence.xml
- ...

Spring 3.2

Spring 3.2

- Spring 3.2 RC1 - Release début Novembre
  - Principalement des améliorations de l’existant
  - Support des contrôleurs asynchrones dans WebMVC
  - Amélioration du testing (REST, MVC)
- Release prévue pour le 12/12/12 ;-)

- Spring 3.3
  - JEE 7 : JPA 2.1, Bean Validation 1.1, JMS 2

[Spring est mort ?]

Spring en 2003

[Spring en 2012]

SpringData
spring Batch
springroo
SpringMobile
Spring Web Services
spring Integration
spring Framework
Spring Web Flow
SpringAndroid
spring Security
spring Flex
SpringSocial

Conclusions

Questions ?