



Spring MISC

Michael Inden

Freiberuflicher Consultant und Trainer

Speaker Intro



- Michael Inden, Jahrgang 1971
- Diplom-Informatiker, C.v.O. Uni Oldenburg
- ~8 ¼ Jahre **SSE** bei Heidelberger Druckmaschinen AG in Kiel
- ~6 ¾ Jahre **TPL, SA** bei IVU Traffic Technologies AG in Aachen
- ~4 ¼ Jahre **LSA / Trainer** bei Zühlke Engineering AG in Zürich
- ~3 Jahre **TL / CTO** bei Direct Mail Informatics / ASMIQ in Zürich
- **Freiberuflicher Consultant, Trainer und Konferenz-Speaker**
- Autor und Gutachter beim dpunkt.verlag

E-Mail: michael_inden@hotmail.com

Blog: <https://jaxenter.de/author/minden>

<https://www.wearedevelopers.com/magazine/java-records>

Kurse: **Bitte spricht mich an!**

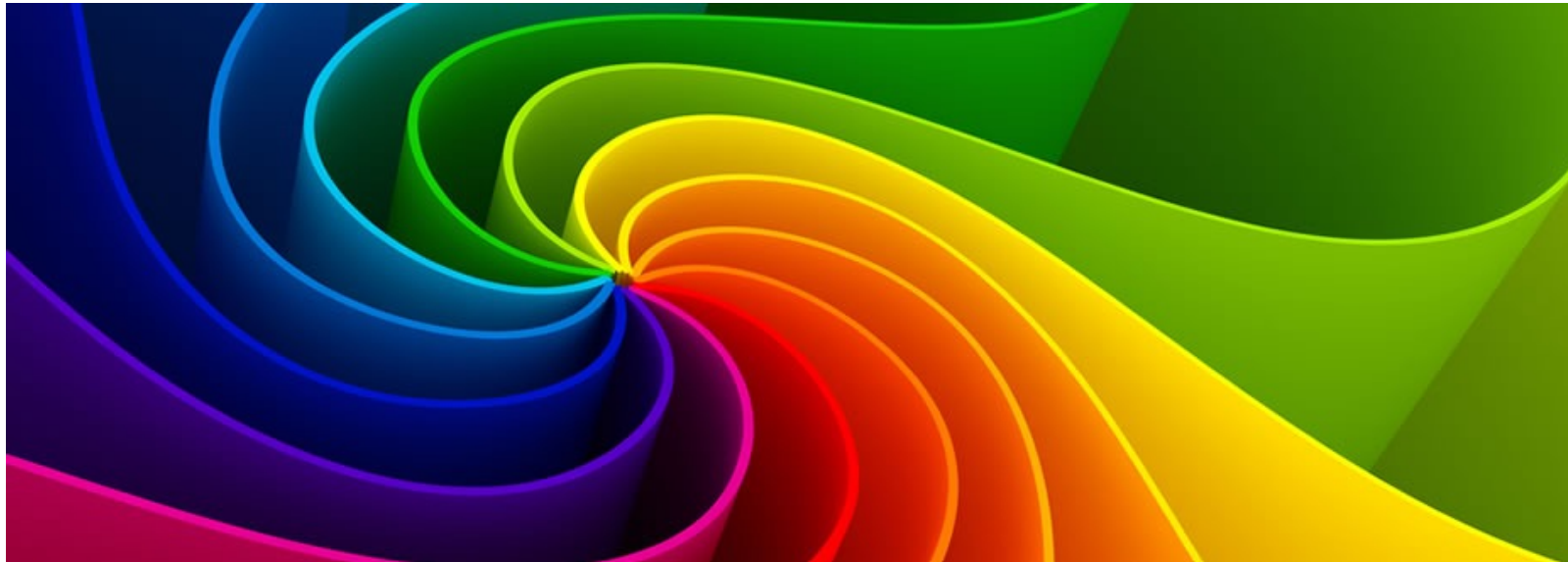




- **AOP in Spring**
 - Einführung
 - AOP und Proxies
 - **Profiles**
 - Einführung
 - Beans exkludieren / inkludieren => CONFIG DEV / TEST / PROD
-



AOP in Spring





- **AOP (Aspect Oriented Programming)**
 - **XML-Konfiguration und AspectJ Annotations möglich**
 - **Was ist AOP?**
 - Programmierparadigma, um die Modularität zu erhöhen
 - Trennung von Cross Cutting Concerns vom Applikationscode
 - wird erreicht, indem zusätzliches Verhalten zu bestehendem Code hinzugefügt wird, ohne den Code selbst zu ändern.
 - Applikationscode und das «neue» Verhalten separat deklarieren.
 - **Das AOP-Framework von Spring hilft uns bei der Implementierung von Cross Cutting Concerns (Logging, Auditing, Security, ...)**
-



- **AOP (Aspect Oriented Programming)**
- **XML-Konfiguration und AspectJ Annotations möglich**

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



- «nackte» Businessklasse

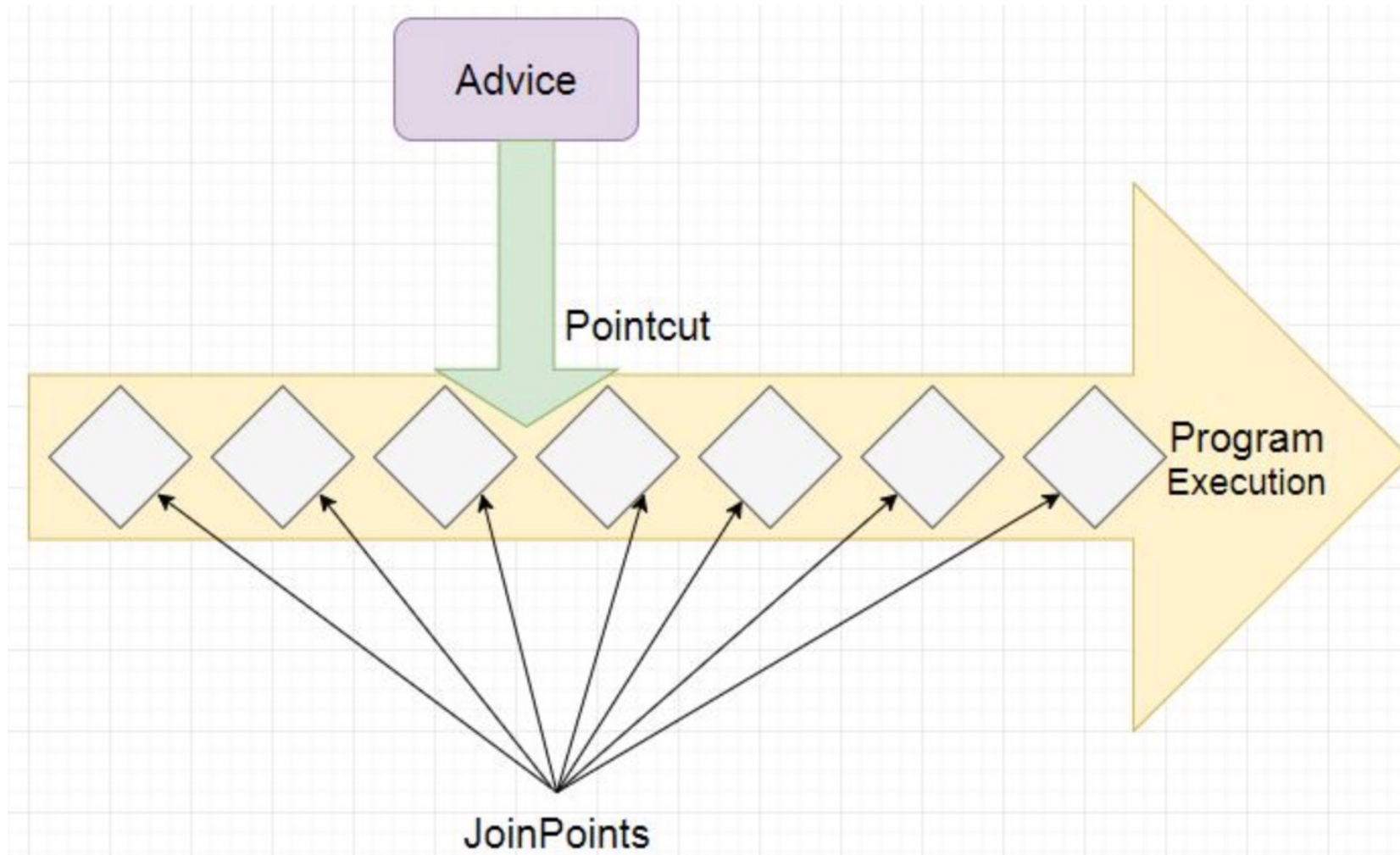
```
public class SampleAdder {  
    public int add(int a, int b) {  
        return a + b;  
    }  
}
```

- Logging-Aspekt

```
public class AdderAfterReturnAspect {  
    private Logger logger = LoggerFactory.getLogger(this.getClass());  
  
    public void afterReturn(Object returnValue) throws Throwable {  
        logger.info("value return was {}", returnValue);  
    }  
}
```

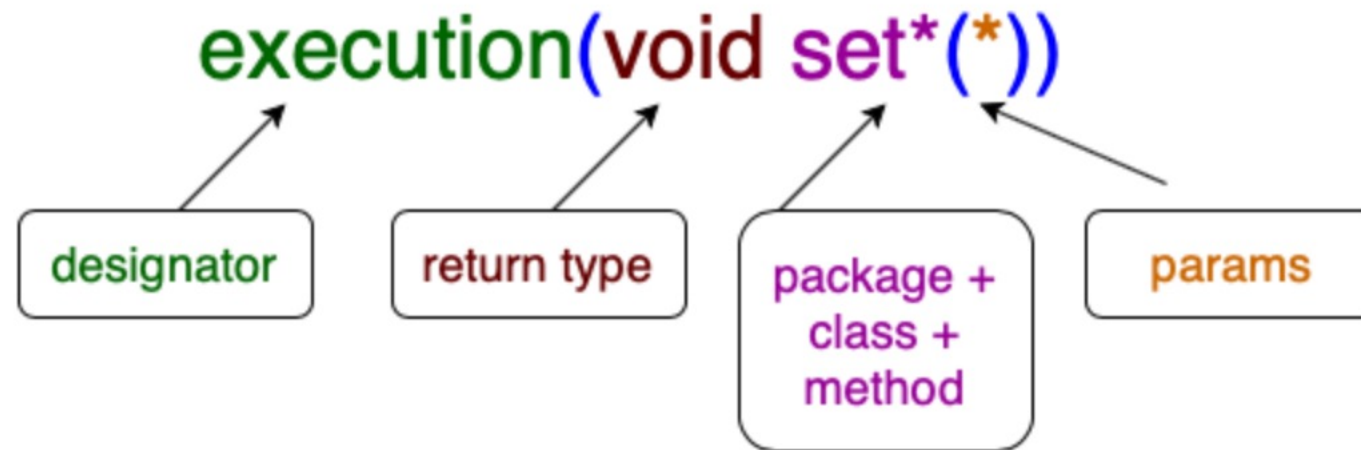


- **Aspect:** eine Modularisierung eines Anliegens, das sich über mehrere Klassen erstreckt. Sowohl Transaktionsverwaltung als auch Logging sind gute Beispiele für Cross Cutting Concerns.
 - **Join point:** Ein Punkt während der Ausführung eines Programms, wie z.B. die Ausführung einer Methode oder die Behandlung einer Ausnahme. In Spring AOP ist ein Join point immer eine Methodenausführung dar.
 - Ein wichtiger Begriff in AOP ist Advice. Es handelt sich um die Aktion, die ein Aspekt an einem bestimmten Joinpoint durchführt.
-

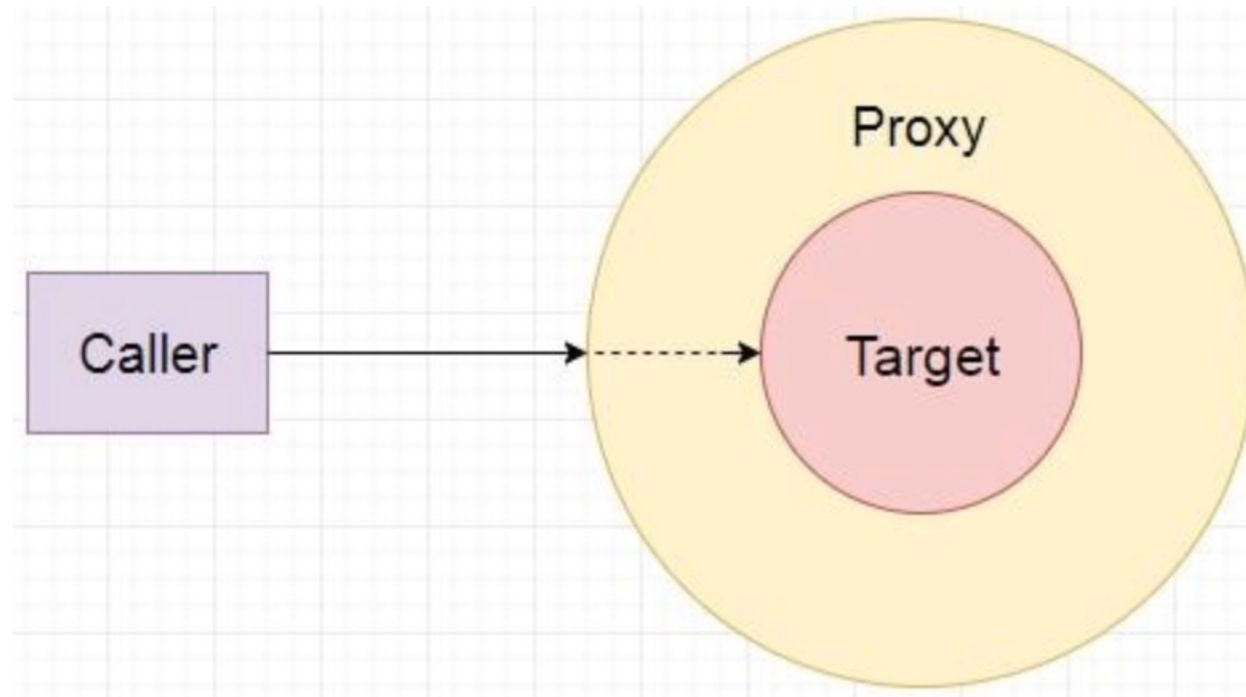




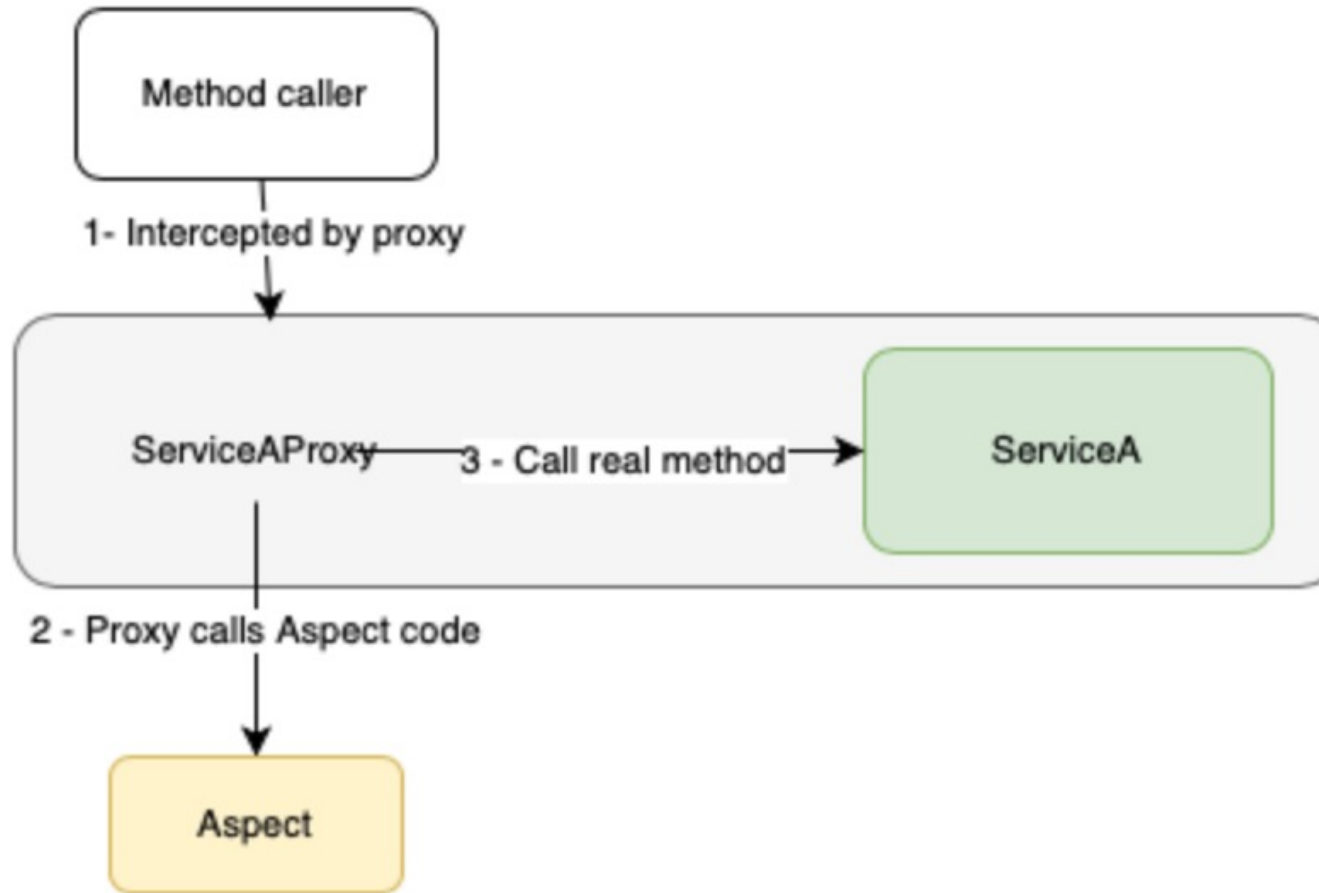
- Pointcut ist ein Prädikat oder ein Ausdruck, der mit Verknüpfungspunkten übereinstimmt.
- Spring verwendet standardmäßig die Pointcut-Definition von AspectJ.



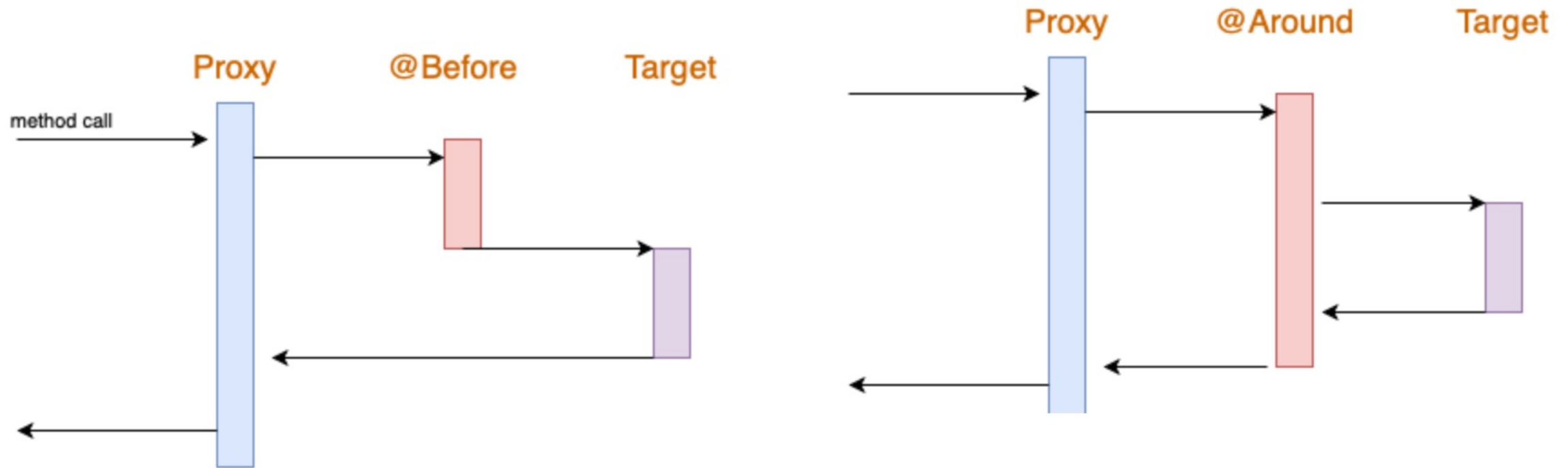
AOP Unter der Motorhaube I



AOP Unter der Motorhaube II



AOP Unter der Motorhaube III: Advices Before / Around





- **Before advice:** wird vor einem JoinPoint ausgeführt
 - **After returning advice:** Advice, die nach der normalen Beendigung eines Join-Points ausgeführt werden, z. B. wenn eine Methode zurückkehrt, ohne eine Exception auszulösen.
 - **After throwing advice:** wird ausgeführt, wenn eine Methode durch das Auslösen einer Ausnahme beendet wird.
 - **After (finally) advice:** wird nach finally ausgeführt
-



- **Around advice:** umklammern einen JoinPoint, z. B. einen Methodenaufruf
 - **leistungsfähigste Art von advices:** Sie können ein benutzerdefiniertes Verhalten vor und nach dem Methodenaufruf ausführen.
 - **Auch für die Entscheidung verantwortlich, ob sie zum JoinPoint weitergeht oder die Ausführung der Methode abkürzt, indem sie ihren eigenen Rückgabewert zurückgibt oder eine Ausnahme auslöst.**
-

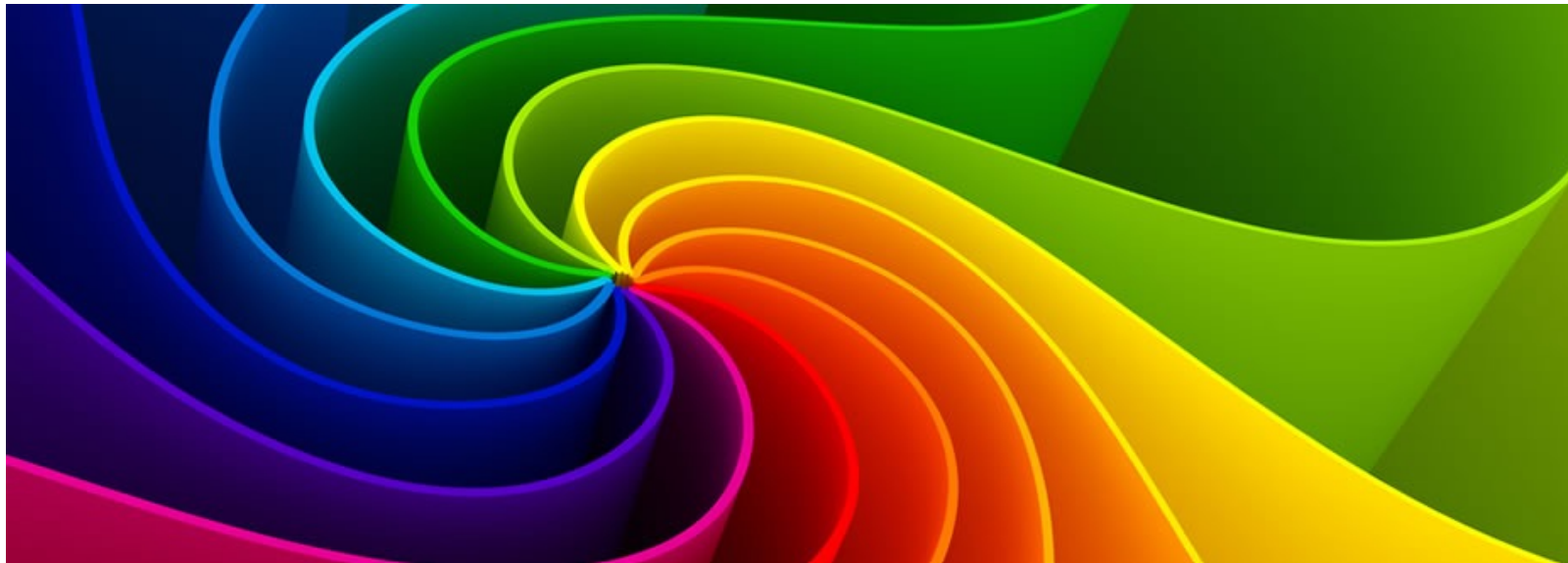


DEMO

«spring-aop-slides-examples»



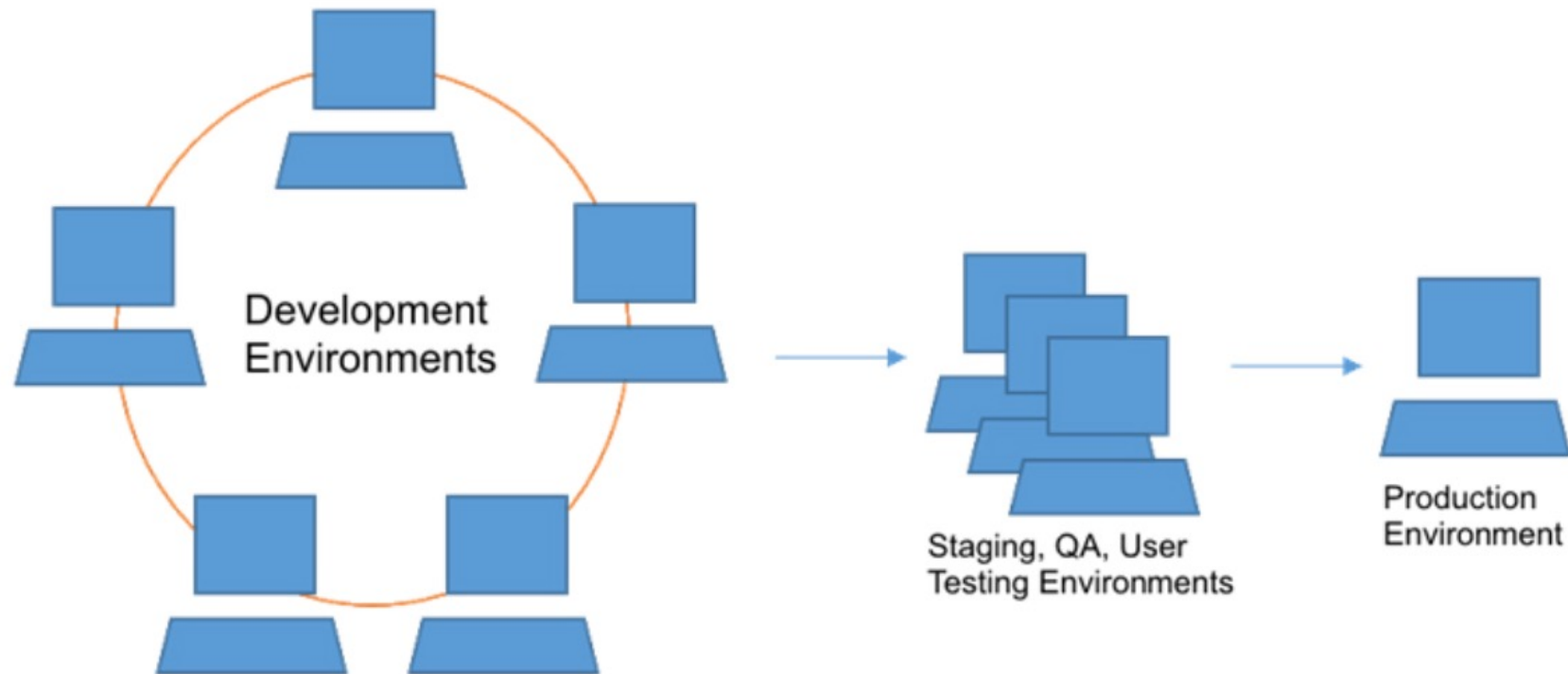
Profiles in Spring



Profiles



- Bei der Erstellung von Anwendungen kann es vorkommen, dass wir verschiedene Anwendungskonfigurationen benötigen.
- Abhängig von der verwendeten Umgebung (Entwicklung, Test, Produktion usw.).



Profiles -- Beispiele



- **DATENBANK:** Zum Beispiel können wir für das Entwicklungsprofil die H2-Datenbank verwenden. Für das Produktionsprofil können wir dann etwa eine MySQL-Datenbank verwenden.
- **PORTS:** Im Dev soll die Applikation beispielsweise auf 8080 laufen. In der Produktion soll etwa 9999 als Port genutzt werden.
- **CACHING:** Ist das Profil "dev" aktiviert, kann ein einfacher Cache-Manager genutzt werden, etwa einer, der auf Maps basiert. Im Profil "prod" kann dann ein ausgeklügelterer Cache-Manager wie etwa EhCache zum Einsatz kommen.

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



- **Spring Boot bietet eine einfache Möglichkeit, verschiedene Konfigurationen mit Profiles zu bereitzustellen und man kann bei Bedarf leicht zwischen den Profilen wechseln.**
 - **Der wichtigste Anwendungsfall für Profile ist die Konfiguration unserer Anwendung für eine von mehreren Umgebungen.**
 - **Es ist eine gute Idee, für jede Umgebung eine spezifische Konfigurationen zu definieren:**
 - `application-dev.properties`
 - `application-test.properties`
 - `application-prod.properties`
-



- **Definition der einzelnen Konfigurationen:**
 - `application-dev.properties`
 - `application-test.properties`
 - `application-prod.properties`
- **Master der einzelnen Konfigurationen:** Die Datei `application.properties` ist der Master für alle Eigenschaften. Hier geben wir an, welches Profil aktiv ist, indem wir die Eigenschaft `spring.profiles.active` definieren:

```
#spring.profiles.active=dev  
spring.profiles.active=prod
```

```
spring.application.name = Spring Profiles  
app.message = This is the Primary Application Property
```



- **DEV mit H2**

```
app.message = This is the DEV Environment Property file
```

```
spring.datasource.url=jdbc:h2:mem:test  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.dbuser=sa  
spring.datasource.dbpassword=
```

```
#NICE:  
spring.h2.console.enabled=true
```



- **TEST mit MySQL**

app.message = This is the TEST Environment property file

spring.datasource.url=jdbc:mysql://localhost:3306/testDB

spring.datasource.username=root

spring.datasource.password=root1234

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect



- **PROD mit MySQL**

app.message = This is the PRODUCTION Environment property file

```
spring.datasource.url=jdbc:mysql://localhost:3306/prodDB
```

```
spring.datasource.username=root
```

```
spring.datasource.password=root1234
```

```
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
```

```
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```

Beans abhängig vom Profile definieren



```
@Configuration
public class ProfileConfig {
    @Profile("DEV")
    @Bean(name = "myBean")
    public String createDevBean() {
        return "My bean is configured with Development configurations!!";
    }

    @Profile("PROD")
    @Bean(name = "myBean")
    public String createProdBean() {
        return "My bean is configured with Production configurations!!";
    }
}
```

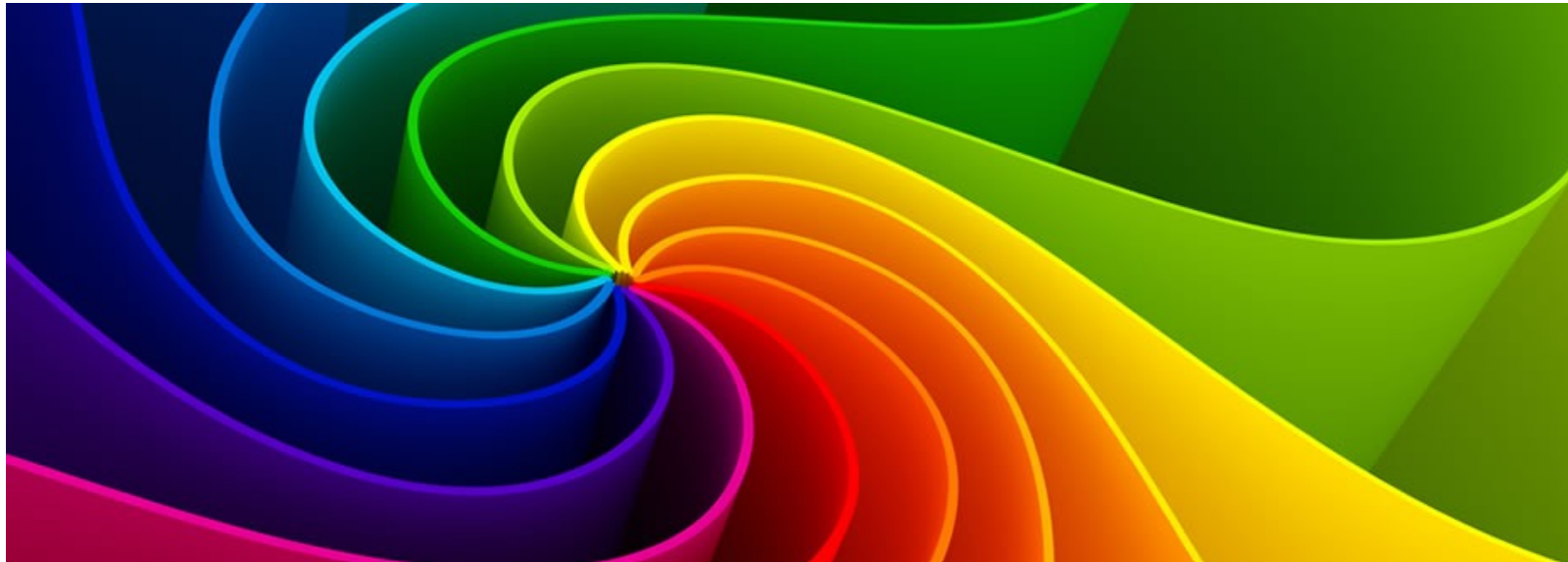


DEMO

«spring-profiles-slides-examples»



Utils in Spring





DEMO

«spring-boot-utils-app»



Questions?



- <https://www.baeldung.com/aspectj>
 - <https://blog.doubleslash.de/separation-of-concerns-mit-spring-aop/>
 - <https://www.javatpoint.com/spring-aop-tutorial>
 - <https://www.javatpoint.com/spring-aop-aspectj-xml-configuration-example>
 - [https://www.tutorialspoint.com/spring/aop with spring.htm](https://www.tutorialspoint.com/spring/aop_with_spring.htm)
 - <https://www.torsten-horn.de/techdocs/Spring-DI-AOP.html#Spring-AOP-Demo>
 - <https://medium.com/javarevisited/spring-core-including-aspect-oriented-programming-in-your-skills-2c37eaa75c2a>
 - <https://www.javainuse.com/spring/spring-boot-aop>
-



Thank You
