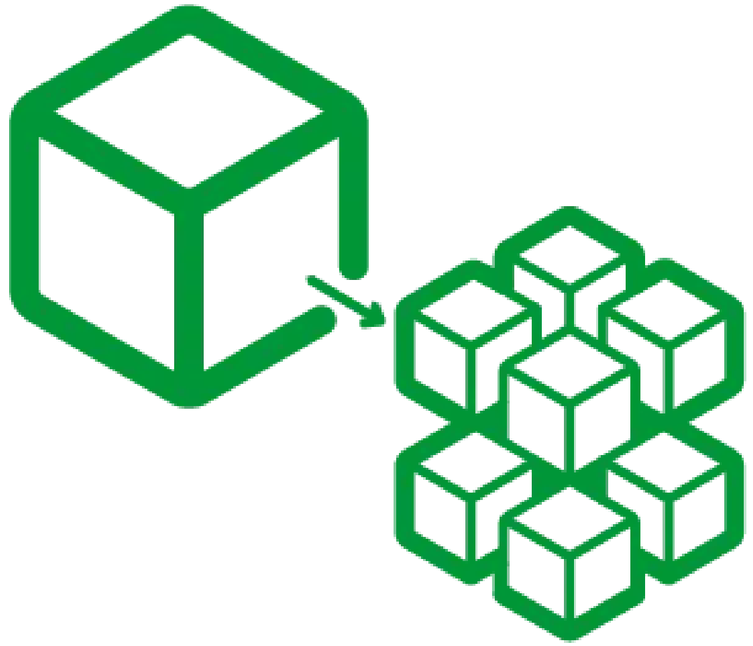
spring®

Marzo 14, 2024

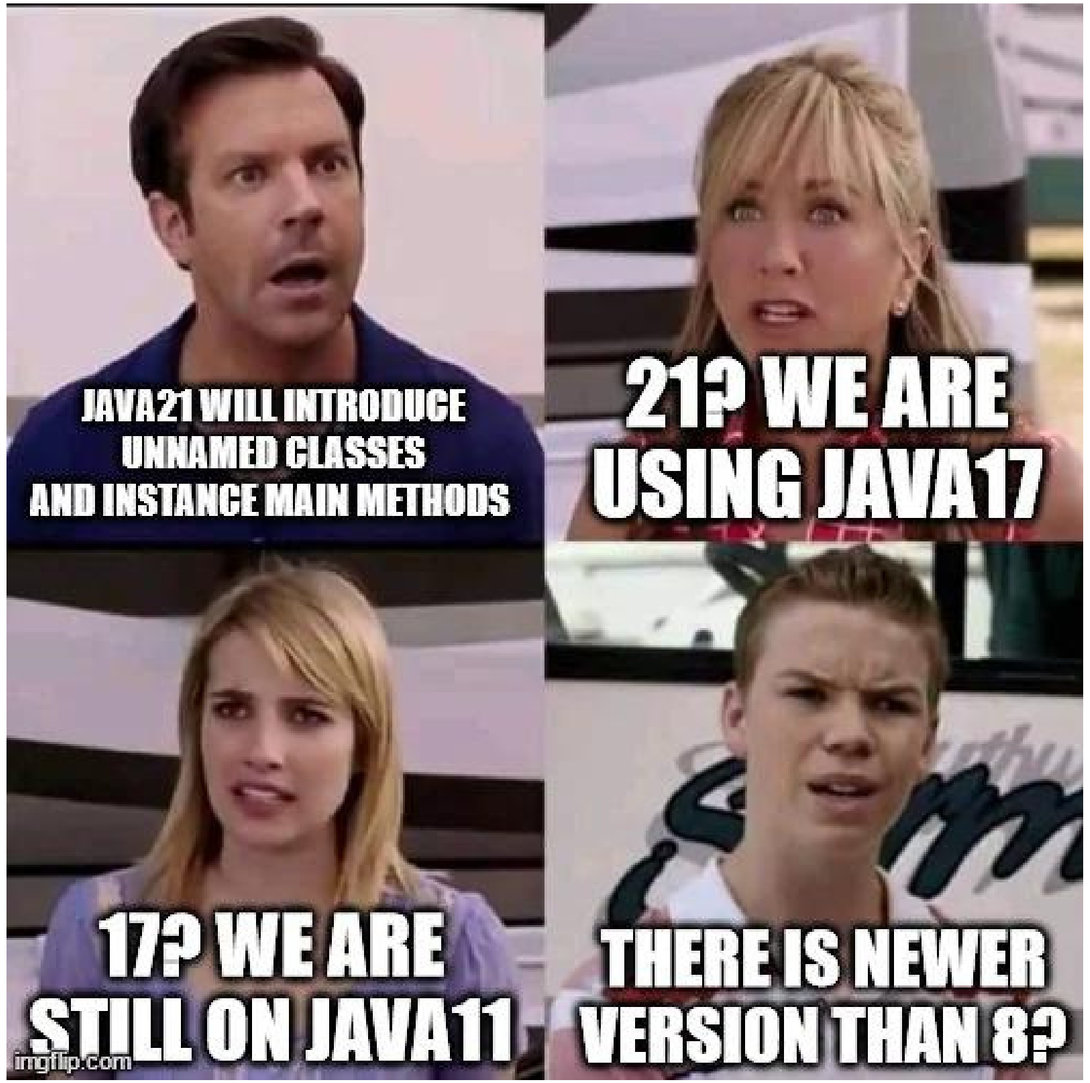
Programación Orientada a Aspectos y su aplicación en Microservicios.



Universidad Tecnológica del Valle del Mezquital.
1er Congreso Estatal de Tecnologías de la Información y Desarrollo de Software 2024

Agenda

- Presentación e Introducción
- ¿Qué son los microservicios?
- ¿Qué es la POA?
- Práctica
- Beneficios, desafíos y consideraciones para su aplicación
- Preguntas y Respuestas


JAVA21 WILL INTRODUCE
UNNAMED CLASSES
AND INSTANCE MAIN METHODS
21? WE ARE
USING JAVA17
17? WE ARE
STILL ON JAVA11
THERE IS NEWER
VERSION THAN 8?
imgflip.com

¿Quién soy?

Egresado de la Universidad Tecnológica del Valle del Mezquital, en el Programa Educativo de TIC.

Desarrollador Java Sr. con +6 años de experiencia diseñando e implementando soluciones tecnológicas para sectores del transporte y tráfico, educación, calzado industrial y financiero. Experto en arquitecturas SOA y Microservicios.

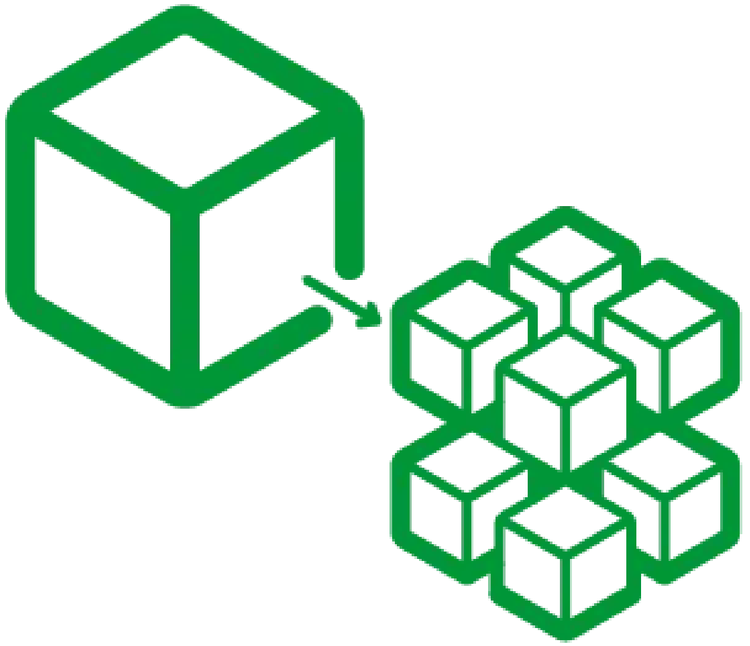
Jefe de Desarrollo de Sistemas en Grupo Financiero Inbursa, responsable de proyectos de transformación tecnológica en la capa de Backend para la App Inbursa Móvil, Nuevo Portal Bancario de Personas Físicas, etc.

Participante activo en comunidades de desarrollo Java, Blockchain & IA en CDMX.

Apasionado por la música, los festivales, hikes y la constante capacitación profesional.

Ing. Ricardo Cruz Ambrosio

¿Qué son los microservicios?



En menos de 15 min.

Microservicios

Los microservicios son una arquitectura de software que se ha vuelto cada vez más popular en los últimos años. En contraste con las arquitecturas monolíticas, donde una aplicación grande se desarrolla como una única unidad, los microservicios descomponen una aplicación en una colección de servicios pequeños e independientes.

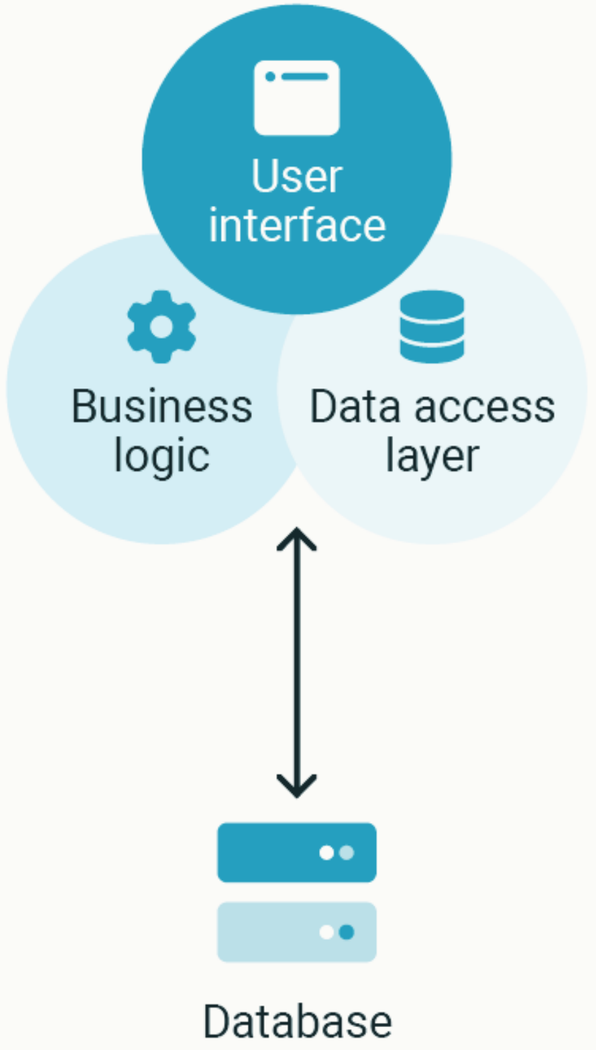
01 **Desacoplamiento**

02 **Escalabilidad.**

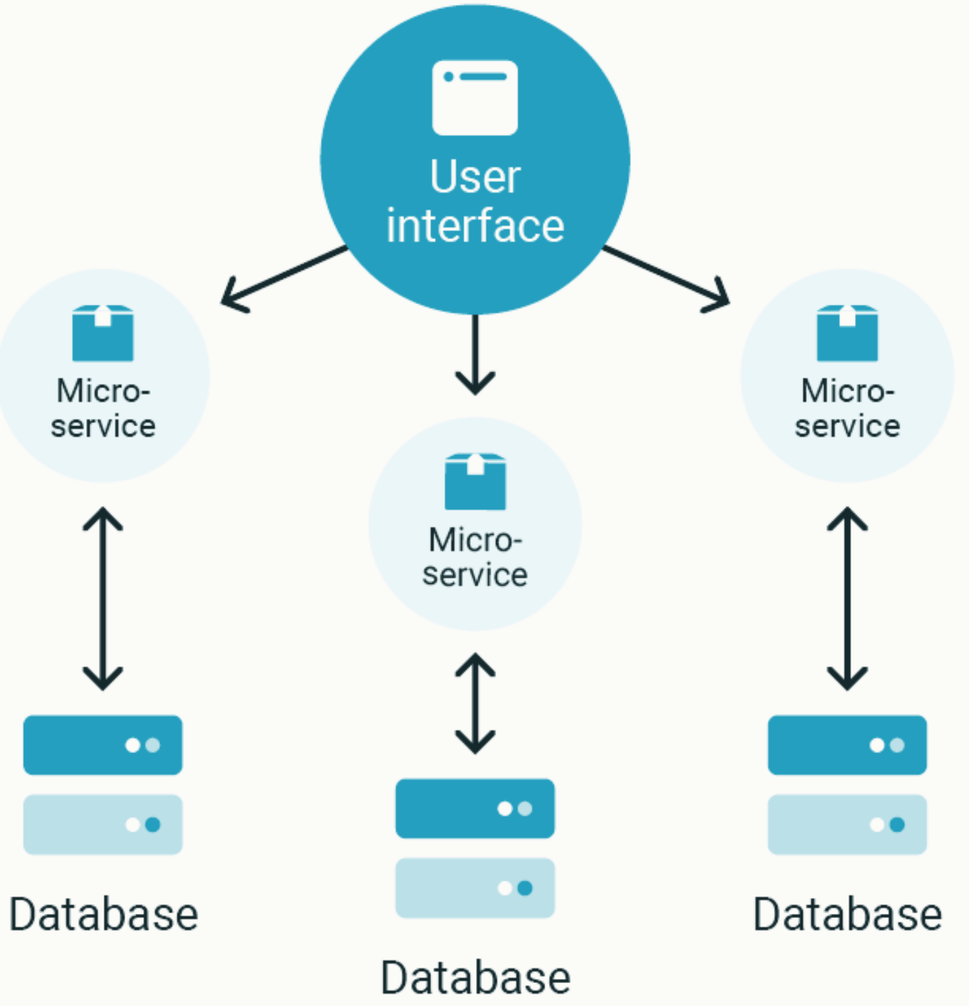
03 **Resiliencia**

03 **Arquitectura Políglota**

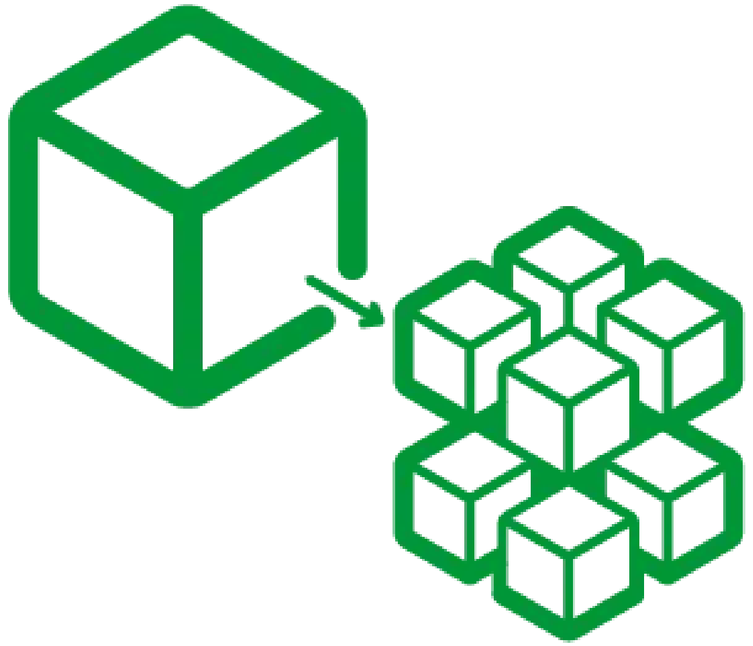
Monolithic Architecture



Microservice Architecture



¿Qué es la Programación Orientada a Aspectos?



AOP

La Programación Orientada a Aspectos (AOP) es un paradigma de programación que permite modularizar aspectos transversales en una aplicación. Estos aspectos transversales son preocupaciones que afectan a múltiples partes de una aplicación, como la seguridad, la auditoría, la transaccionalidad, entre otros. AOP se centra en separar la lógica de negocio de estos aspectos, lo que facilita la reutilización del código y mejora la modularidad y la mantenibilidad del sistema.

¿De qué se compone?

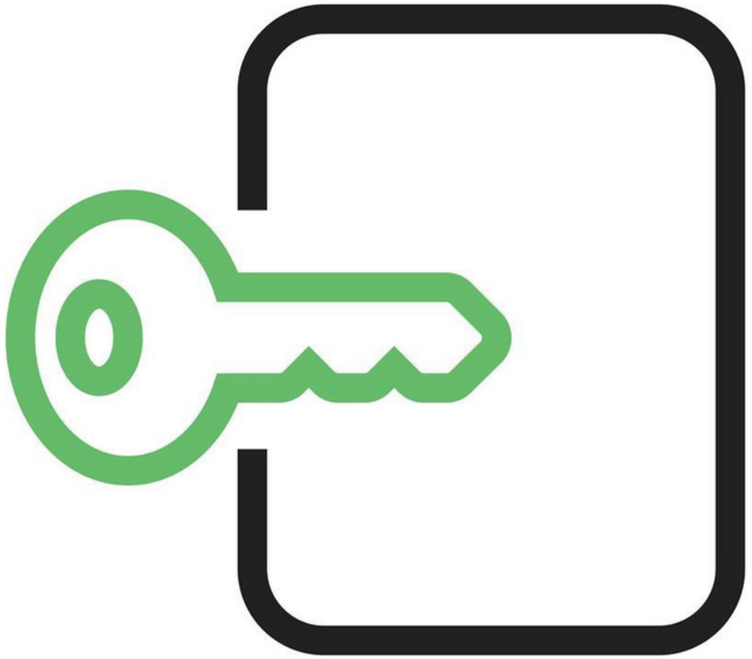
01 **Aspectos**

02 **Puntos de Corte (Poincut)**

03 **Consejos (Advices)**

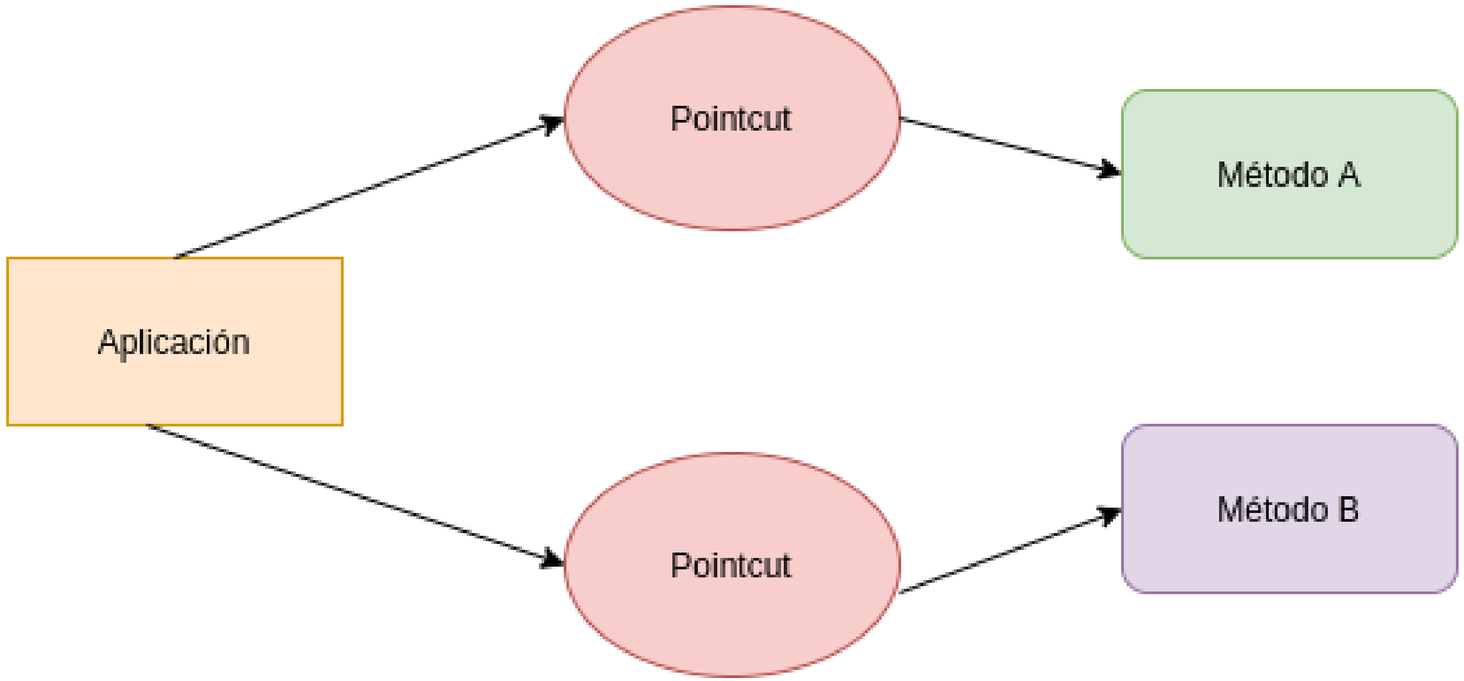
Aspectos

Un aspecto encapsula el comportamiento que se quiere modularizar. Por ejemplo, un aspecto puede ser la seguridad de una aplicación, que incluye la lógica para autenticar usuarios antes de que accedan a ciertas funcionalidades.



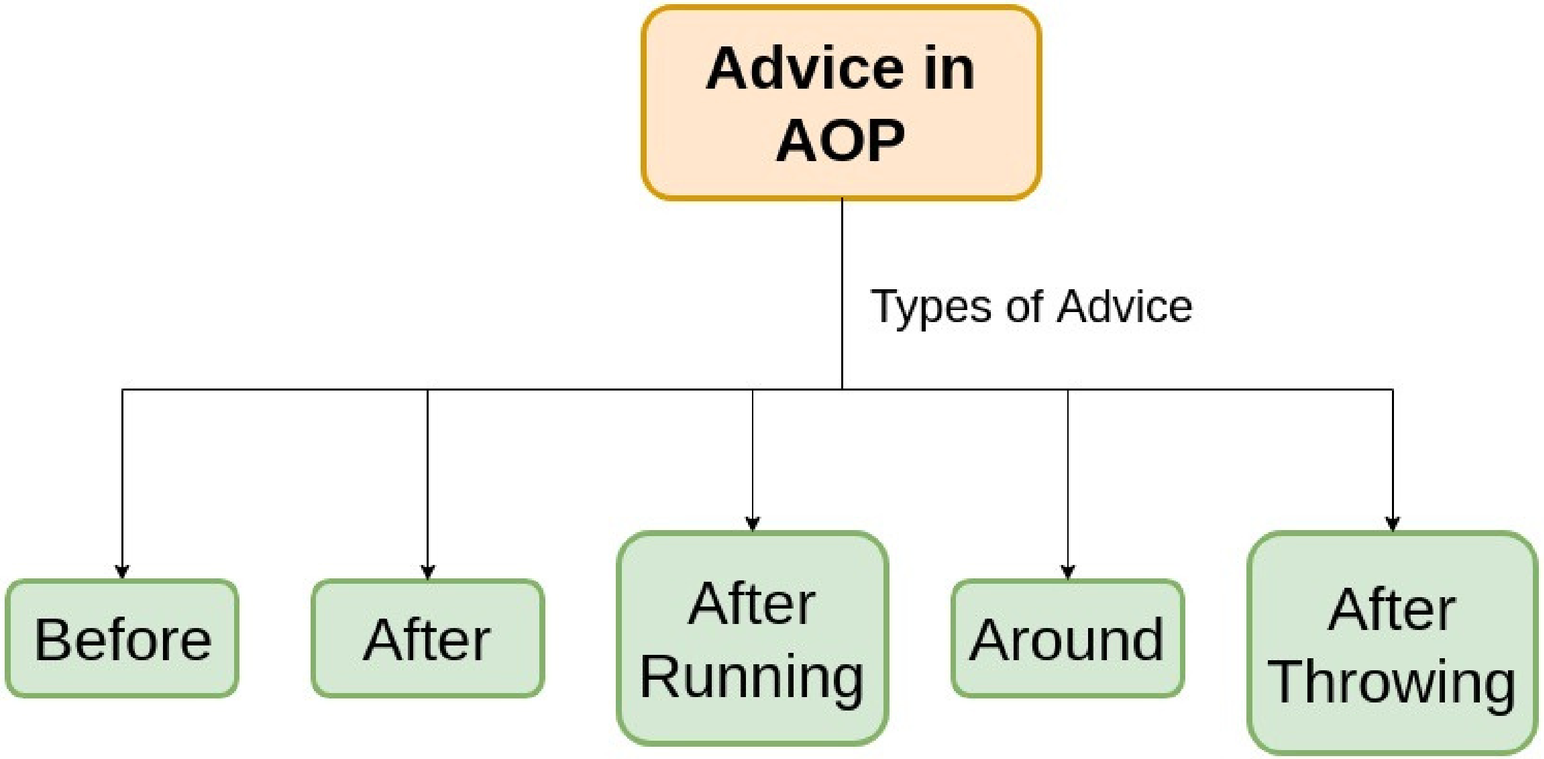
Poincut

Un Pointcut es una expresión que define uno o más puntos de interés en el código base de una aplicación donde se aplicará un aspecto. En otras palabras, un pointcut especifica cuándo y dónde se debe ejecutar el código relacionado con un aspecto.



Advices

Los Advices son bloques de código que se ejecutan en puntos específicos del programa, conocidos como join points, durante la ejecución de una aplicación. Estos advices encapsulan la funcionalidad que se desea agregar o modificar en estos puntos de interés.



Tipos de Advices

Existen varios tipos de advices, cada uno ejecutándose en diferentes momentos durante la ejecución del programa:

- **Before Advice:** Se ejecuta antes de que el join point sea invocado. Es útil para realizar tareas de preparación antes de que se ejecute el método objetivo.
- **After Returning Advice:** Se ejecuta después de que el join point termina de ejecutarse correctamente y devuelve un valor. Es útil para realizar acciones después de que se completa con éxito la ejecución del método.
- **After Throwing Advice:** Se ejecuta después de que el join point lanza una excepción. Es útil para manejar excepciones y realizar acciones específicas en caso de que ocurra un error.

Tipos de Advices

- **After (Finally) Advice:** Se ejecuta después de que el join point termina su ejecución, ya sea con éxito o lanzando una excepción. Es útil para realizar limpieza o liberación de recursos, independientemente del resultado de la ejecución.
- **Around Advice:** Es el más poderoso y flexible de todos los advices. Permite al aspecto tomar el control total del join point, incluyendo la capacidad de modificar los argumentos, el resultado devuelto o incluso evitar la ejecución del método objetivo.

Práctica

Beneficios de AOP

1. **Modularidad Mejorada:** AOP permite separar las preocupaciones transversales, como la seguridad, el registro y la gestión de transacciones, de la lógica del negocio en microservicios individuales. Esto mejora la modularidad del código y facilita su mantenimiento y evolución.
2. **Reutilización de Código:** Al encapsular aspectos comunes en módulos reutilizables, AOP fomenta la reutilización del código en toda la aplicación de microservicios, lo que conduce a un desarrollo más eficiente y menos propenso a errores.
3. **Consistencia en la Implementación:** AOP permite aplicar aspectos de manera consistente en todos los microservicios de una aplicación, lo que garantiza un comportamiento uniforme en áreas como la seguridad, el registro y la gestión de transacciones.
4. **Separación de Preocupaciones:** Al separar las preocupaciones transversales de la lógica del negocio, AOP facilita la comprensión y el mantenimiento del código al enfocarse en un aspecto específico a la vez.

Desafíos y Consideraciones

1. **Granularidad Excesiva:** Una granularidad excesiva en la definición de aspectos puede conducir a una aplicación compleja y difícil de mantener. Es importante encontrar un equilibrio entre la modularidad y la simplicidad en la implementación de aspectos.
2. **Impacto en el Rendimiento:** La introducción de aspectos puede tener un impacto en el rendimiento de la aplicación, especialmente si se utilizan en áreas críticas que se ejecutan con alta frecuencia. Es importante realizar pruebas de rendimiento para evaluar el impacto y optimizar cuando sea necesario.
3. **Gestión de Transacciones Distribuidas:** En entornos de microservicios distribuidos, la gestión de transacciones puede volverse más compleja debido a la naturaleza descentralizada de la arquitectura. AOP puede ayudar a abordar este desafío al proporcionar un mecanismo para coordinar transacciones entre los diferentes servicios.
4. **Dificultad para Depurar:** La introducción de aspectos puede complicar la depuración y el seguimiento de errores, ya que la lógica de los aspectos está dispersa en toda la aplicación. Es importante contar con herramientas adecuadas y prácticas de depuración para facilitar la identificación y resolución de problemas.

Recursos y Herramientas Recomendadas

Aspect Oriented Programming with Spring

Microservices Architecture

Q&A

¡Gracias!

Contacto

Código