Aspect Oriented Programming

Cosa imparerai

- Cos’è l’Aspect Oriented Programming e quali sono i principi alla base di questo paradigma di programmazione

Aspect Oriented Programming

Nella programmazione una delle metodologie più utilizzate è l'**Object Oriented Programming** (OOP).

L'OOP consente di scrivere software modulare, facilmente manutenibile e riutilizzabile.

I concetti alla base dell'OOP sono:

- classi
- interfacce
- ereditarietà
- polimorfirsmo
- incapsulamento

Aspect Oriented Programming

Tuttavia esistono comportamenti che per loro natura sono trasversali (cioè utilizzati da più o tutte le classi della nostra applicazione).

Questi comportamenti (un tipico esempio è il *logging*) sono presenti in più componenti di un'applicazione.

Anche se possiamo gestire tali comportamenti con utilizzando determinati design pattern, comunque dobbiamo inserire del codice nelle classi (ad es. l'interfaccia da implementare, una variabile d'istanza, ecc…).

E se invece riuscissimo a gestire questi comportamenti in maniera diversa?

L'**Aspect Oriented Programming** nasce proprio con questo obiettivo.

Aspect Oriented Programming

Cos'è L'AOP

L'Aspect Oriented Programming (AOP) è un paradigma di programmazione complementare all'OOP.

L'AOP è in grado di descrivere dei comportamenti che sono trasversali all'applicazione.

Con l'AOP, questi comportamenti vengono separati ed isolati dal dominio applicativo in moduli dedicati al singolo aspetto.

L'AOP garantisce:

- ❑ Separazione dei comportamenti trasversali dal dominio applicativo
- ❑ Riutilizzo del codice

Aspect Oriented Programming

I concetti alla base di questo paradigma sono:

- **Crosscutting Concern**
- **Aspect**
- **Join Point**
- **Advice**
- **Pointcut**
- **Target object**
- **AOP proxy**

Aspect Oriented Programming

Crosscutting Concern

È un comportamento trasversale all'applicazione che si riflette in più punti.

Esempi: il logging, l'autenticazione, la gestione delle transazioni…

Aspect

È una classe implementa le logiche applicative di un determinato comportamento (*Crosscutting Concern*).

Grazie agli *Aspect* possiamo aggiungere comportamenti agli oggetti della nostra applicazione senza modificare il codice sorgente.

Un *Aspect* comprende:

- gli *Advice* che implementano le logiche applicative
- i *Pointcut* che ne regolano l'esecuzione

Aspect Oriented Programming

Join Point

Il *Join Point* è un punto preciso nell'esecuzione dell'applicazione.

Esempi di *Join Point* sono il verificarsi di un'eccezione, la modifica del valore di una variabile. l'invocazione di un metodo, ecc…

Advice

Un *Advice* è un metodo della classe *Aspect* che implementa l'operazione che un *Aspect* deve eseguire ogni volta che ad un *Join Point* si verifica una determinata condizione.

Possiamo immaginare gli Advice come dei Servlet Filter…

Aspect Oriented Programming

Advice

Esistono diversi tipi di Advice:

Tipo di Advice	Quando viene eseguito
BeforeAdvice	prima venga eseguito un *Join Point*
AfterReturning	dopo l'esecuzione del *Join Point* e se non sono state generate eccezioni
AfterThrowing	dopo l'esecuzione del *Join Point* che ha generato l'eccezione
AfterAdvice	dopo l'esecuzione del Join Point, indipendentemente dall'esito
AroundAdvice	consente di prendere il controllo del *Join Point*, specificando anche quando e se eseguire il *Join Point*

Aspect Oriented Programming

Pointcut

Un *Pointcut* è un'espressione che definisce le regole con cui associamo l'esecuzione di un *Advice* ad un determinato *Join Point*.

In pratica indica che al verificarsi di un determinato *Join Point* (ad es. l'esecuzione del metodo *salvaOrdine()* della classe *OrderService*) venga applicato un determinato *Advice* (ad es. un metodo che implementa l'*Advice AfterReturning*).

Target Object

Rappresenta l'oggetto sul quale viene applicato l'Advice.

AOP proxy

È un oggetto creato dal framework AOP per implementare gli aspect.

Aspect Oriented Programming

Cosa si intende per proxy?

Per capire bene il funzionamento dell'AOP in Spring è importante capire come funziona un proxy.

Un proxy è una classe che racchiude un oggetto ed espone tutti i metodi pubblici dell'oggetto stesso.

In sostanza, **un proxy è un wrapper** (o contenitore…).

Quando invochiamo un metodo esposto dal proxy, l'esecuzione reale viene delegata all'oggetto contenuto.

Spring usa i proxy sia per delegare l'esecuzione di metodi, sia per gestire i vari *advice*.

Aspect Oriented Programming

Cosa si intende per proxy?

Facciamo un esempio pratico di proxy…

1. Creo un'interfaccia che definisce i metodi da implementare

```
public interface Catalogo {
    public List<String> getProdotti();

    public String getProdotto(Long id);
}
```

2. Creo la classe che implementa la logica di business

```
public class CatalogoImpl implements Catalogo {

    public List<String> getProdotti() {
        List<String> a = new ArrayList<String>();
        a.add("Prod 1");
        a.add("Prod 2");
        a.add("Prod 3");

        return a; /* logica applicativa... */
    }

    public String getProdotto(Long id) {
        return "Carta A4"; /* logica applicativa... */
    }

}
```

3. Creo la classe wrapper (che fa da proxy)

```
public class CatalogoProxy implements Catalogo {
    private Catalogo catalogo;

    public CatalogoProxy(Catalogo catalogo) {
        super();
        this.catalogo = catalogo;
    }

    public List<String> getProdotti() {
        return catalogo.getProdotti();
    }

    public String getProdotto(Long id) {
        return catalogo.getProdotto(id);
    }
}
```

4. Creo un'istanza della classe proxy e vediamo che verranno invocati i metodi della classe contenuta.

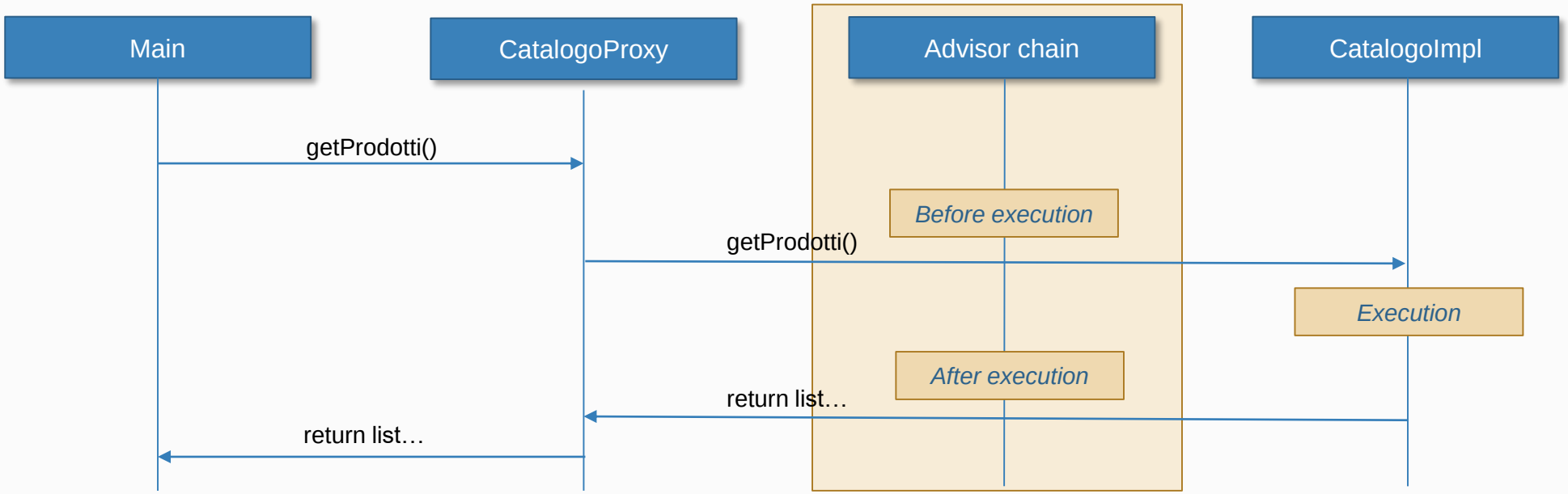
```
public class Main {

    public static void main(String[] args) {
        CatalogoProxy cp = new CatalogoProxy(new CatalogoImpl());

        System.out.println(cp.getProdotti());
        System.out.println(cp.getProdotto(1231));
    }
}
```

```
[Prod 1, Prod 2, Prod 3]
Carta A4
```

Aspect Oriented Programming

Cosa si intende per proxy?

Facciamo un esempio pratico di proxy…



È importante notare che le chiamate interne ad CatalogoImpl non attivano mai l'esecuzione di un advice AOP.

L'esecuzione di un advice avviene a livello di proxy.

La classe CatalogoImpl non saprà mai che l'invocazione del suo metodo è stata intercettata…

Di cosa abbiamo parlato in questa lezione

- Cos’è l’Aspect Oriented Programming e quali sono i principi alla base di questo paradigma di programmazione