Dependency Injection

Cosa imparerai

- Cosa si intende per Dependency Injection e come si implementa

paolopreite

Dependency Injection

La **Dependency Injection** (**DI**) è un design pattern utilizzata per implementare l'IoC.

Il pattern DI consente la creazione e l'associazione degli oggetti fuori dalla classe che li utilizza.

Il pattern DI può essere implementato in tre modi:

- **Constructor Injection**
- **Setter Injection**
- **Interface Injection**

Spring implementa solo Constructor e Setter Injection!

Dependency Injection

Constructor Injection

La DI basata sul costruttore viene realizzata invocando il costruttore della classe e passando in input l'istanza della classe da creare.

```java
public class Biblioteca {
    private BibliotecaService service;

    public Biblioteca(BibliotecaService serv) {
        super();

        service = serv;
    }

    public List<Libro> getLibriDisponibili() {
        return service.getLibriDisponibili();
    }
}
```

```java
public class BibliotecaFactory {
    public static Biblioteca getBiblioteca() {
        Biblioteca b = new Biblioteca(new BibliotecaServiceImpl());

        return b;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        BibliotecaFactory.getBiblioteca().getLibriDisponibili();
    }
}
```

Dependency Injection

Setter Injection

Questa modalità è realizzata chiamando il metodo setter dell'oggetto da utilizzare.

```java
public class Biblioteca {
    private BibliotecaService service;

    public Biblioteca() {
        super();
    }

    public void setService(BibliotecaService service) {
        this.service = service;
    }

    public List<Libro> getLibriDisponibili() {
        return service.getLibriDisponibili();
    }
}
```

```java
public class BibliotecaFactory {
    public static Biblioteca getBiblioteca() {
        Biblioteca b = new Biblioteca();
        b.setService(new BibliotecaServiceImpl());

        return b;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        BibliotecaFactory.getBiblioteca().getLibriDisponibili();
    }
}
```

Di cosa abbiamo parlato in questa lezione

- Cosa si intende per Dependency Injection e come si implementa