Spring: IoC

Peter Alagna Jr.

Inversion of Control (IoC)

In **Spring**, this is also known as **Dependency Injection** (DI).

- Remove **dependencies** in your code by injecting **objects** via configuration and/or annotations.

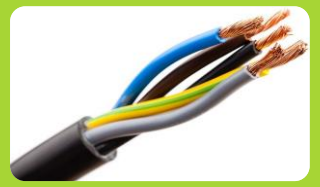
```java
public class CustomerServiceImplementation implements CustomerService {

    private CustomerRepository customerRepository =
            new CustomerRepositoryHibernate();

    @Override
    public List<Customer> findAll() {
        return customerRepository.findAll();
    }
}
```

```java
public class CustomerServiceImplementation implements CustomerService {

    private CustomerRepository customerRepository;

    @Override
    public List<Customer> findAll() {
        return customerRepository.findAll();
    }
}
```

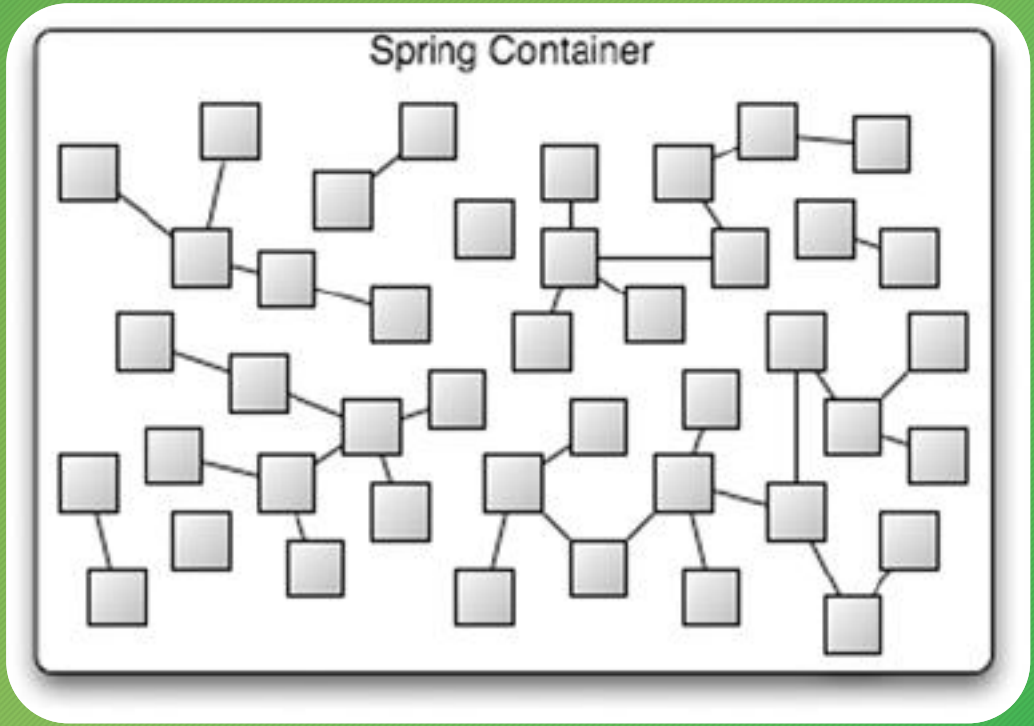
Dependency Injection

REVATURE

Wiring



Association between **beans** within the Spring **container**.



REVATURE

Types of Injection

- Setter Injection.
  - <property name="attributeName" ref="beanName" />
- Constructor Injection.
  - Guaranteed contract.
  - You need a constructor defined for each situation.
  - Index based instead of name based.
  - <constructor-arg index="*0*" *ref="beanName"* />
- As an additional:

You can also inject primitive values.

- Can use **expression language** if configured properly.
- Spring injects these values with its **setters**.
- <property name="id" value="123" />

REVATURE

Autowiring

Automatically wire beans:

- By type.
  - Problems if you have two beans of the same class.
  - <bean … autowire="byType"></bean>
- By name.
  - Solves by type issue.
  - <bean … autowire="byName"></bean>
- By constructor.
  - <bean … autowire="constructor"></bean>

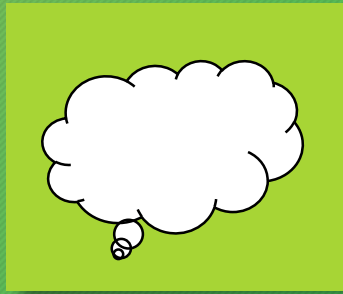
Bean Naming Convention

↓

ClassName : className

REVATURE

Materials



- Spring IoC: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html
- Inversion of Control: https://martinfowler.com/articles/injection.html

REVATURE