Contents

[Persiapan 1](#_Toc168047680)

[Spring Starter Project 3](#_Toc168047681)

[Membuat Object 7](#_Toc168047682)

[Di spring boot 7](#_Toc168047683)

[AutoWired 8](#_Toc168047684)

[Menggunakan XML 8](#_Toc168047685)

[Singleton Object vs Prototype 10](#_Toc168047686)

[Singleton 10](#_Toc168047687)

[Setter Injection 11](#_Toc168047688)

[Primitive type(int, Sting, dll) 11](#_Toc168047689)

[Reference Type 13](#_Toc168047690)

[Constructor Injector 14](#_Toc168047691)

[Autowired di bean 15](#_Toc168047692)

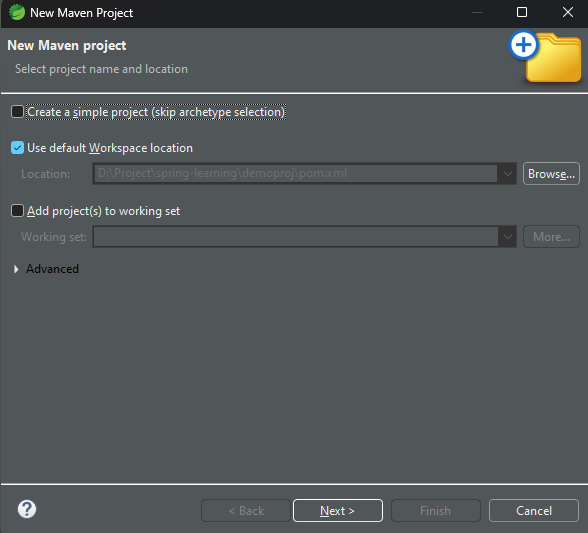
[byName 17](#_Toc168047693)

[byType 18](#_Toc168047694)

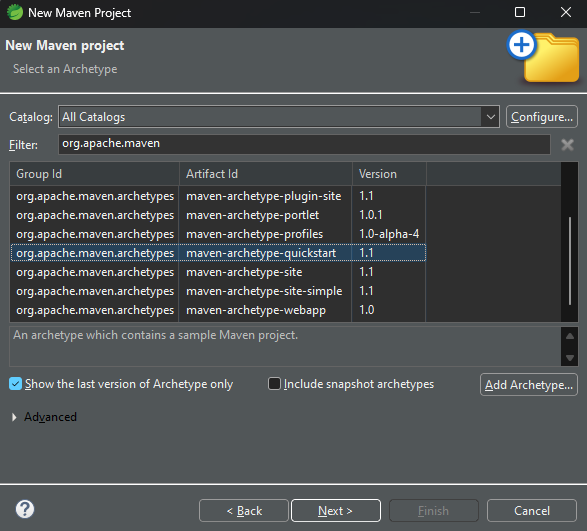
# Persiapan

Sebelum memulai project download dulu kebutuhan dependencies, Download Dependencies, menggunakan maven project

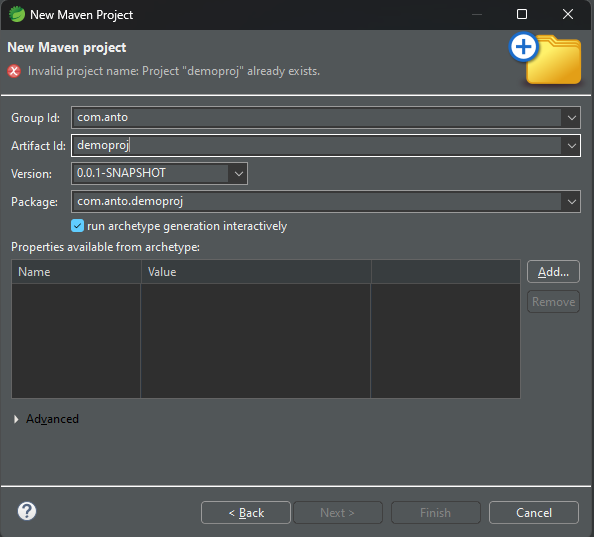
File > new maven project, next



Pilih maven-archtype-quickstart dan next



Isi Group-id = com.anto, artifact Id = demoproj, lalu Finish



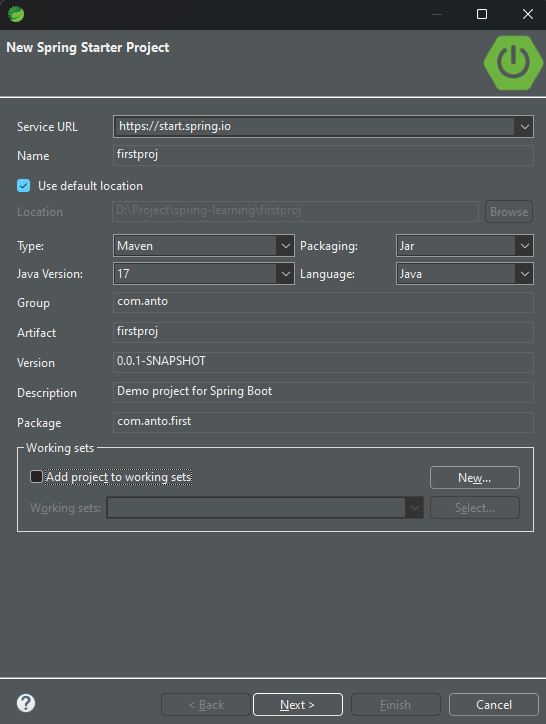
Akan muncul console, ketik Y

Untuk tambah dependencies buka pom.xml

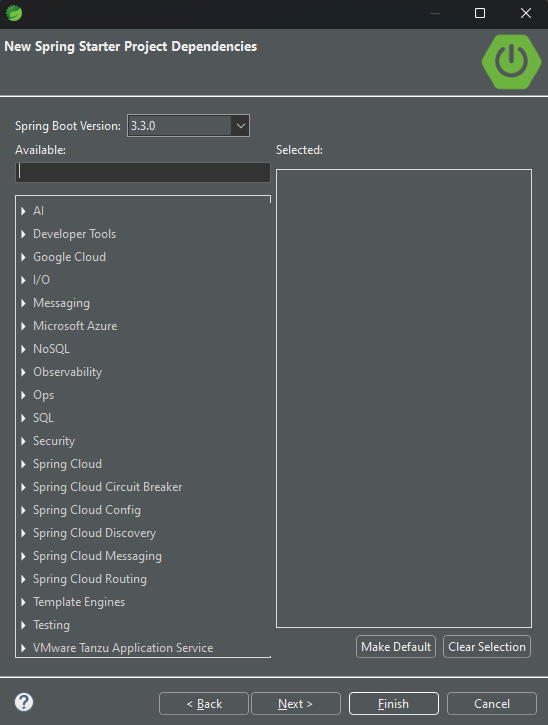
Untuk tau spring dependencies nya, buka <https://mvnrepository.com/search?q=spring> , pilih spring context, untuk versi tutorial ini kita menggunakan versi 6.1.8

Copy paste code dependency yg ada di tab Maven ke pom.xml di dalam tab <dependencies>

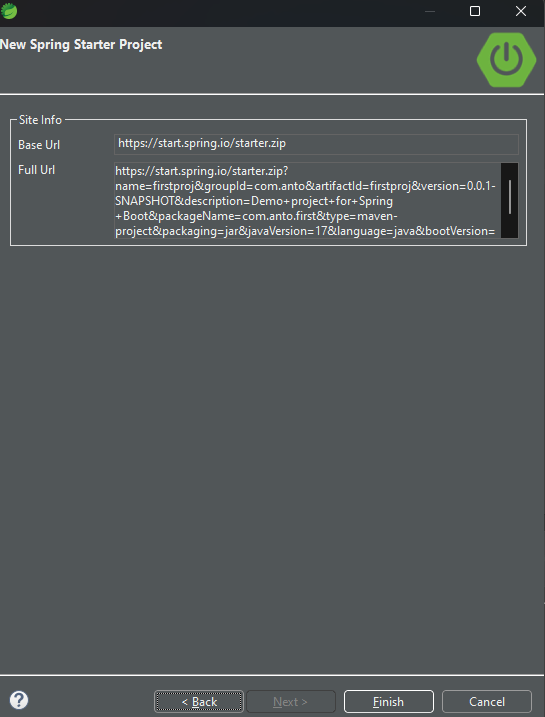
# Spring Starter Project



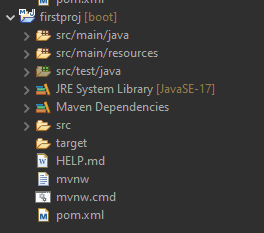
Isi Name, Type Maven, Group = com.anto, Artifact = fistproj, package = com.anto.first lalu next



Untuk ini, sementara langsung next saja



Finish, lalu tunggu proses download selesai



Buka src/main/java, lalu com.anto.first, FirstprojApplication.java

# Membuat Object

## Di spring boot

FirstprojApplication.java

package com.anto.first;

import org.springframework.boot.SpringApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class FirstprojApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(FirstprojApplication.class, args);

// old method to create object

// Alien obj = new Alien();

// inject object in spring boot

Alien obj = context.getBean(Alien.class);

obj.code();

}

}

Pertama buat class bernama Alien terlebih dahulu, bisa dengan cara klik kanan agar muncul suggestion



Alien.java

package com.anto.first;

import org.springframework.stereotype.Component;

*@Component*

public class Alien

{

public void code()

{

System.***out***.println("Im Coding...");

}

}

Tambahkan @Component di atas nama class dan buat 1method, kemudian di FirstprojApplication import ApplicationContext

## AutoWired

Cara lain membuat object yg di handle spring dengan cara menambahkan @Autowired

Alien.java

package com.anto.first;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

*@Component*

public class Alien

{

*@Autowired*

Laptop lap; // object ini akan otomatis dibuat oleh spring

public void code()

{

lap.compile();

}

}

Laptop.java

package com.anto.first;

import org.springframework.stereotype.Component;

*@Component*

public class Laptop {

public void compile()

{

System.***out***.println("Compiling....");

}

}

## Menggunakan XML

App.java

package com.anto.SpringDemo;

// import org.springframework.beans.factory.BeanFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

// import org.springframework.core.io.FileSystemResource;

public class App

{

public static void main( String[] args )

{

ApplicationContext context= new ClassPathXmlApplicationContext("spring.xml");

Alien obj = (Alien)context.getBean("alien");

obj.code();

}

}

Alien.java

package com.anto.SpringDemo;

public class Alien

{

public void code()

{

System.***out***.println("Im coding...");

}

}

spring.xml

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

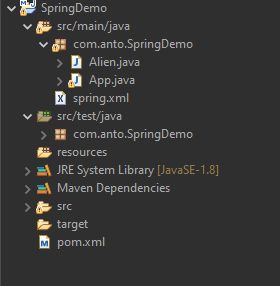
xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*></**bean**>

</**beans**>

Struktur file



\*untuk file spring.xml nya, bisa drag and drop file .xml ke folder src/main/java agar struktur nya seperti diatas

# Singleton Object vs Prototype

## Singleton

Alien.java

package com.anto.SpringDemo;

public class Alien

{

int age = 18;

public Alien()

{

System.***out***.println("Alien object created...");

}

public void code()

{

System.***out***.println("Im coding...");

}

}

App.java

package com.anto.SpringDemo;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main( String[] args )

{

ApplicationContext context= new ClassPathXmlApplicationContext("spring.xml");

Alien obj1 = (Alien)context.getBean("alien");

obj1.code();

obj1.age = 20;

System.***out***.println(obj1.age); // 20

Alien obj2 = (Alien)context.getBean("alien");

obj2.code();

System.***out***.println(obj2.age); // 20

}

}

Pada contoh diatas, meski kita define age =18 di class Alien, saat di main, kita deklarasi ulang age =20.

Saat membuat obj2 juga, age = 20 karena by default Spring beans menggunakan konsep singleton, obj1 dan obj2 me-refer ke object yg sama

Scope Prototype

Agar kita bisa menggunakan 2 objek pada file .xml tambahkan scope=*"prototype"*

Bila tidak kita tulis scope nya apa, berarti menggunakan “singleton”

spring.xml

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"* scope=*"prototype"*></**bean**>

</**beans**>

App.java

public static void main( String[] args )

{

ApplicationContext context= new ClassPathXmlApplicationContext("spring.xml");

Alien obj1 = (Alien)context.getBean("alien");

obj1.code();

obj1.age = 20;

System.***out***.println(obj1.age); // 20

Alien obj2 = (Alien)context.getBean("alien");

obj2.code();

System.***out***.println(obj2.age); // 18

}

Dengan menggunakan scope prototype kita dapat membuat 2 object yg berbeda

# Setter Injection

## Primitive type(int, Sting, dll)

Kita dapat men-set value properti pada bean xml, syarat nya di class harus dibuat setter getter dan default constructor

App.java

package com.anto.SpringDemo;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main( String[] args )

{

ApplicationContext context= new ClassPathXmlApplicationContext("spring.xml");

Alien obj1 = (Alien)context.getBean("alien");

obj1.code();

System.***out***.println(obj1.getAge()); // 10

}

}

Alien.java

package com.anto.SpringDemo;

public class Alien

{

private int age = 18;

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Alien()// default constructor

{

System.***out***.println("Alien object created...");

}

public void code()

{

System.***out***.println("Im coding...");

}

}

spring.xml

<?**xml** version=*"1.0"* encoding=*"UTF-8"*?>

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*>

<**property** name=*"age"* value=*"10"*></**property**>

</**bean**>

</**beans**>

## Reference Type

Me-inisialisasi type ref dari suatu class/object

App.java

public class App

{

public static void main( String[] args )

{

ApplicationContext context= new ClassPathXmlApplicationContext("spring.xml");

Alien obj1 = (Alien)context.getBean("alien");

obj1.code();

System.***out***.println(obj1.getAge()); // 10

}

}

Alien.java

public class Alien

{

private int age = 18;

private Laptop laptop; //memanggil class Laptop

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Laptop getLaptop() {

return laptop;

}

public void setLaptop(Laptop laptop) {

this.laptop = laptop;

}

public Alien()

{

System.***out***.println("Alien object created...");

}

public void code()

{

System.***out***.println("Im coding...");

laptop.compile(); // menggunakan method compile() yg ada di class laptop

}

}

Laptop.java

package com.anto.SpringDemo;

public class Laptop

{

public void compile()

{

System.***out***.println("Code Compiled...");

}

}

spring.xml

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*>

<**property** name=*"age"* value=*"10"*></**property**>

<**property** name=*"laptop"* ref=*"laptop"*></**property**>

</**bean**>

<**bean** id=*"laptop"* class=*"com.anto.SpringDemo.Laptop"*> <!-- id disini mengacu pada property dengan ref="laptop" -->

</**bean**>

pada Alien.java kita ingin menggunakan method compile() milik class Laptop, agar Laptop di kenal kita harus menambahkan bean dan property di spring.xml

# Constructor Injector

Menambahkan constructor with field di bean

Alien.java

public class Alien

{

private int age;

private Laptop laptop; //memanggil class Laptop

public Alien(int age) {// contructor with Field

this.age = age;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Laptop getLaptop() {

return laptop;

}

public void setLaptop(Laptop laptop) {

this.laptop = laptop;

}

public void code()

{

System.***out***.println("Im coding...");

laptop.compile(); // menggunakan method compile() yg ada di class laptop

}

}

Spring.xml

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*>

<**constructor-arg** value=*"12"*></**constructor-arg**>

<**property** name=*"laptop"* ref=*"laptop"*></**property**>

</**bean**>

Saat program di jalankan, maka constructor(file xml) akan mengset variable this.age dengan value yg ada di XML

Kapan pakai Constructor injector & Setter injector?

Gunakan Constructor Injector kalau nilainya wajib ada/ required

Gunakan Setter Injector kalau nilainya tidak wajib

# Autowired di bean

Pertama buat interface Computer dan ubah Alien.java

Computer.java

ppackage com.anto.SpringDemo;

public interface Computer {

void compile();

}

Buat Class Desktop.java dan meng-implements interface Computer

package com.anto.SpringDemo;

public class Desktop implements Computer

{

public void compile()

{

System.***out***.println("Code Compiled in Desktop");

}

}

Ubah class Laptop.java dengan meng-implement Computer

package com.anto.SpringDemo;

public class Laptop implements Computer

{

public void compile()

{

System.***out***.println("Code Compiled in Laptop");

}

}

Ubah Alien.java dengan memanggil property/interface Computer, juga buat setter dan getter

package com.anto.SpringDemo;

public class Alien

{

private int age;

private Computer com;

public Alien()

{

System.***out***.println("Alien object created...");

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public Computer getCom() {

return com;

}

public void setCom(Computer com) {

this.com = com;

}

public void code()

{

System.***out***.println("Im coding...");

com.compile();

}

}

Di spring.xml, set property untuk com

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*>

<**property** name=*"age"* value=*"10"*></**property**>

<**property** name=*"com"* ref=*"desktop"*></**property**>

</**bean**>

<**bean** id=*"laptop"* class=*"com.anto.SpringDemo.Laptop"*>

</**bean**>

<**bean** id=*"desktop"* class=*"com.anto.SpringDemo.Desktop"*>

</**bean**>

Bila di run maka outputnya:

Code Compiled in Laptop

10

Kenapa “Code compiled in Laptop” karena pada spring.xml kita membuat property com yg me-ref ke laptop

Bila kita ubah ref nya ke desktop, <**property** name=*"com"* ref=*"desktop"*></**property**> maka outputnya akan “Code Compiled in Desktop”

Bila kita membuat nama ref yg sama juga dengan nama property (sama2 com):

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"*>

<**property** name=*"age"* value=*"10"*></**property**>

<**property** name=*"com"* ref=*"com"*></**property**>

</**bean**>

<**bean** id=*"com"* class=*"com.anto.SpringDemo.Laptop"*>

</**bean**>

Outputnya juga tetap sama “Code compiled in Laptop”, bila nama nya sama kita bisa singkat penulisan nya

Autowired, kita memberitau spring, jika ingin mengakses properti, dan jika ada bean yg available, otomatis konek. Ada 2 macam Autowired

## byName

pada contoh diatas, Alien.java property name private Computer com; adalah com dan bean name/id dari <**bean** id=*"com"* juga com, sehingga tanpa define tag <property> di xml, program akan otomatis di konek/sambungkan ke autowired byName

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"* autowire=*"byName"*>

<**property** name=*"age"* value=*"10"*></**property**>

</**bean**>

<**bean** id=*"com"* class=*"com.anto.SpringDemo.Laptop"*>

</**bean**>

## byType

pada contoh diatas, Alien.java type property private Computer com; adalah Computer maka di xml nya, program akan mencari class yg meng-implement Computer, pada contoh di bawah class nya adalah Desktop

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"* autowire=*"byType"*>

<**property** name=*"age"* value=*"10"*></**property**>

</**bean**>

<**bean** id=*"desktop"* class=*"com.anto.SpringDemo.Desktop"*>

</**bean**>

Output nya “Code Compiled in Desktop”

Jika ada 2 class di tag bean yg implement Computer, maka akan error, untuk mengatasinya tambahkan primary=*"true"*

<**bean** id=*"alien"* class=*"com.anto.SpringDemo.Alien"* autowire=*"byType"*>

<**property** name=*"age"* value=*"10"*></**property**>

</**bean**>

<**bean** id=*"com"* class=*"com.anto.SpringDemo.Laptop"* primary=*"true"*>

</**bean**>

<**bean** id=*"desktop"* class=*"com.anto.SpringDemo.Desktop"*>

</**bean**>

Sehingga program akan mennjalankan yg Laptop