# Spring

Spring is Dependency Injection framework

### Example Of Tight-coupling

**public** **class** ComplexService{

SortAlgorithm sortAlogrithm = **new** BubbleSortAlgorithm();

}

For ComplexService the ShoringAlgorithm is necessary

Public class BubbleSortAlgorithm implements SortAlgorithm{

When we use ComplexService, BubbleShortAlogorithm Object will be created automatically.

### Example Of loosely-coupling

**public** **class** ComplexService{

SortAlgorithm sortAlogrithm;

**public** ComplexService(SortAlgorithm sortAlogrithm){

}

}

SortAlgorithm sortAlogrithm = **new** BubbleSortAlgorithm();

ComplexService complexService = **new** ComplexService(sortAlogrithm);

Now Spring FrameWork does exactly the same Job.

@Componenet

**public** **class** ComplexService{

@Autowired

SortAlgorithm sortAlogrithm;

}

@Componenet

Public class BubbleSortAlgorithm implements SortAlgorithm{}

#### Dependency Injection

Now spring understands you want to create

ComplexService obj and BubbleShortAlogrithm obj

While creating obj, who to manage dependency

#### Beans

ComplexService obj and BubbleShortAlogrithm obj

Spring managing obj are called beans

All **beans** are singletons, so whenever **ApplicationContext** gets **created**, they are all pre-loaded.

**Singleton** class is a class that can have only one object.

After first time, if we try to instantiate the Singleton class, new variable also points to the first instance.

So one change will affects all.

<bean scope=”**prototype**”> then it will create more objects.

#### Inversion of control

Above examples 1, and 2 we are taking control of class that create dependent object,

Giving that control to the framework is known as IOC (**inversion of control**).

#### IOC Container

Anything that is implementing IOC is IOC container.

Example: Application Context

Bean Factory

#### Application Context

Is where all the beans are created, and managed.

All the core logic of spring-frame work happens here.

public class SpringIn5StepsApplication {

public static void main(String[] args) {

SortAlgotirhm sortObj= new BubbleSortAlgorithm();

BinarySearch bsObj= new BinarySearch(sortObj);

int result= bsObj.binarySearch(**~~myArray~~**, **~~serchNum~~**);

System.out.println(result);

}

}

public class BinarySearch {

private SortAlgorithm sortA;

public BinarySearch(SortAlgorithm sortA){

this.sortA = sortA;

}

public int binarySearch(int[] numbers, int num){

int[] sortedNums = sortA.sort(numbers);

// need **sortedNums**

But don’t want **specific implementation**

public interface SortAlgorithm {

public int[] sort(int[] numbers) {

public class QuickSortAlgorithm implements SortAlgorithm {

public int[] sort(int[] numbers){

// sorting by quick

public class BubbleSortAlgorithm implements SortAlgorithm {

public int[] sort(int[] numbers){

// sorting by bubble

### Dependency injection with spring

@SpringBootApplication

public class SpringIn5StepsApplication {

public static void main(String[] args) {

// what are the beans? @component

// what are the dependencies of a been? @Autowired

// where to search for beans 🡺 to spring we need to tell the package to search for beans

// Spring boot 🡺 no need to tell where to search for beans

@SpringBootApplication get this scan as this package & it’s

sub packages for beans

ApplicationContext ac

ApplicationContext ac= SpringApplication.run(….args);

BinarySearch bs= ac.getBean(BinarySearch.class);

int result= bs.binarySearch(new array, numSerch);

System.out.println(result);

}

get BinarySearch.class type of bean

@Component

public class BinarySearch {

@Autowired

private SortAlgorithm sortA;

public BinarySearch(SortAlgorithm sortA){

this.sortA = sortA;

}

public int binarySearch(int[] numbers, int num){

int[] sortedNums = sortA.sort(numbers);

//sortedNums need to for step

public interface SortAlgorithm {

public int[] sort(int[] numbers) {

@Component

public class QuickSortAlgorithm implements SortAlgorithm {

public int[] sort(int[] numbers){

@Component

public class QuickSortAlgorithm implements SortAlgorithm {

public int[] sort(int[] numbers){

Searching directory [com/tharikan/basic/springTest] for files matching pattern [com/tharikan/basic/springTest /\*\*/\*.class]

Identified candidate component class: file [com/tharikan/basic/springTest BinarySearchImpl.class]

Identified candidate component class: file [/in28Minutes/git/getting-started-in-5-steps/spring-in-5-steps/target/classes/com/in28minutes/spring/basics/springin5steps/BubbleSortAlgorithm.class]

Creating instance of bean 'binarySearchImpl'

Creating instance of bean 'bubbleSortAlgorithm'

Finished creating instance of bean 'bubbleSortAlgorithm'

Constuctor - Autowiring by type from bean name 'binarySearchImpl' via constructor

to bean named 'bubbleSortAlgorithm'

Setter - Autowiring by type from bean name 'binarySearchImpl' to bean named 'bubbleSortAlgorithm'

No Setter or Constructor - Autowiring by type from bean name 'binarySearchImpl' to bean named 'bubbleSortAlgorithm'

Finished creating instance of bean 'binarySearchImpl'

### Aspect oriented programming (AOP)

Call this kind of methods..(exaple in order to log )

Web layer

Business layer

Data Layer

Something may go through all (Transection, login, ..)

\*.save\* \*.get\* \*StudentImpl.get\*

Identify which methods need to intercepted (point-cut)

Logger.logmsg(Logger.ingo + methodName + parametors)

This exact please need to be called when that point-cut methods calls (an aspect)

### Jpa(java persistent Api)

Persist means :: data serving

Persistence is “do something (don’t give up)”

Javax.persistence = jpa

**JPA** is not a tool or framework; rather, it defines a set of concepts that can be implemented by any tool or framework.

Event how application server provides implementation to the jpa

so without any other frameworks(like hibernate) we can work with jpa.

### Hibernate

It’s ORM(object relational mapping) Framework. Which implements jpa.

(class🡺 table, variables🡺columns, objects🡺rows)