**Spring**

What is Spring?

Spring is a Java Framework which allow us to do Dependency Injection so that we can build loosely coupled applications.

Spring Framework makes the development of JavaEE application easy. It provides so many modules to do that such as Spring MVC, Spring JDBC, Spring Core and so on.

What is Dependency Injection?

Dependency Injection is a design pattern that we can use to build applications.

Example: -

Graphical user interface, application

Description automatically generated

Here, Ramu has a dependency on Geeta or Geeta is a dependency for Ramu.

Earlier we used to create an object of Geeta class by using new keyword in Ramu class but the issue with that approach is application become tightly coupled.

Spring framework’s ability called Dependency Injection can be utilized to get rid of this issue.

Spring will create an instance of Geeta and inject it.

Here, we are not the one who are creating the object of Geeta class, spring is doing that for us.

What is Inversion of Control?

The process in which we give control of object creation to spring and spring create and inject objects for all the dependency in runtime. That is called Inversion of Control.

We need to tell spring what kind object that need to be created for the dependency, for that we either use XML file or Annotation. (What kind of object that need to be create and injected for Geeta)

Diagram, timeline

Description automatically generated

This is just one ability of Spring framework.

Spring also provides other module, services, API such as

Spring JDBC, Spring ORM, Spring MVC, Spring Security.

**Spring Modules**



**Spring Core**

The ***Core***and ***Bean*** provides features to fundamental parts of the framework. This includes the *Ioc* as well the *Dependency Injection* (DI).

A sophisticated implementation of the factory pattern, called BeanFactory*,*is provided by the **Bean** module.

The ***Context*** module uses the base provided by the Core and Beans modules. It inherits from these modules. On top of that it provides some additional features like internationalization, Event propagation, Resource loading. It has access rights for any objects that define and configure. In the Context module, the major point is the ApplicationContext interface.

In the Spring framework, a powerful language tool for querying is given by the ***Expression Language Module***. It also provides manipulation of an object graph at run-time execution.

**Next Modules**

AOP

Aspect

Instrumentation

Messaging

AOP (Aspect Oriented Programming): -

1. Allow us to define method interceptors and point cuts; Helps in decoupling code.

To Be Continued…

**Spring IOC Container**

A picture containing icon

Description automatically generated

* It is a component of spring framework, which we get with spring framework. Like with JVM we get garbage collector.
* Spring IOC container is responsible for
  + Object creation
  + Holding objects in memory
  + Injecting one object in another object (dependency injection)
* IOC container control and maintain the lifecycle of object.
* We need to give two information to IOC container
  + Which all Beans (POJO classes) that spring need to manage.
  + Configuration file (contain information dependency of one Bean on another)
* With the help of these two-information spring IOC container create objects and perform dependency injection and then our application code can utilize those objects.
* ***ApplicationContext*** is the interface used to represent spring IOC container.
* ***BeanFactory*** is also a representation of spring IOC container but we prefer ApplicationContext because it inherit from BeanFactory so it has all of its features and additionally it also have some additional features.
* ApplicationContext is an interface so we can create object out of it, so in case we need to create object of it then we have to use any class who is implementing this interface.

Diagram

Description automatically generated

[Details Link](https://www.youtube.com/watch?v=1jGGsuQoLjY&list=PL0zysOflRCekeiERASkpi-crREVensZGS&index=4)

**Ways of Dependency Injection**

[Link](https://www.youtube.com/watch?v=bICqNfzUG4c&list=PL0zysOflRCekeiERASkpi-crREVensZGS&index=5)

**How to start a project**

Text

Description automatically generated

**Lifecycle Methods of Spring Bean**

* Spring provides two important methods to every bean.

public void init()

public void destroy()

Diagram

Description automatically generated with medium confidence

inti() function will consists of all the initialization related code such as loading configuration, connecting with database, webservice etc.

destroy() function is for cleanup code. Let’s say we created a database connection and did some transactions with it. After finishing all the transaction now, we want to close the connection so we will write code for that in destroy() function.

Complete Lifecycle looks like this: -

* Spring Bean (Java class with some members, constructor, and getter setter).
* Configuration XML File

A picture containing diagram

Description automatically generated

Configuration Techniques for Beans

* Xml
* Spring Interface
* Annotation

**Autowiring in Spring**

Feature of spring framework in which spring container inject the dependency automatically.

Autowiring can’t be used for primitive and string value. It works with reference only.

Wiring stands for link two objects or injecting one class object in another class so that it can work properly. When spring does that for us automatically then we call it Autowiring.

Diagram, application

Description automatically generated



