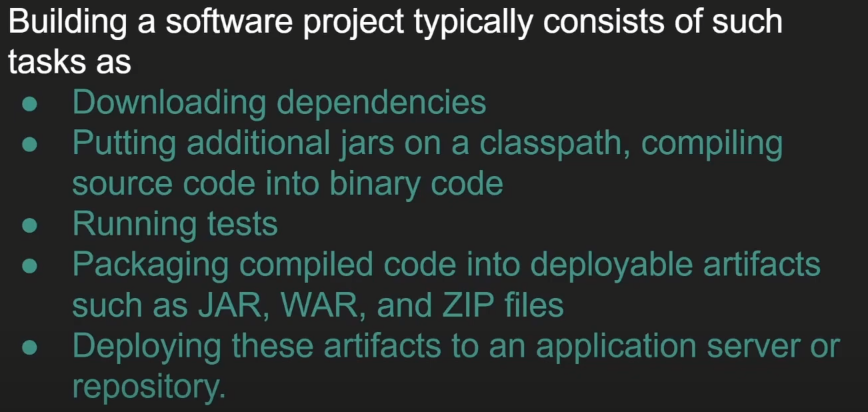
**Spring**

**What is Maven?**

* Maven is Build Management Tool.
* Build mean’s .java file get complied in .class file and then packaged into .jar / .war.



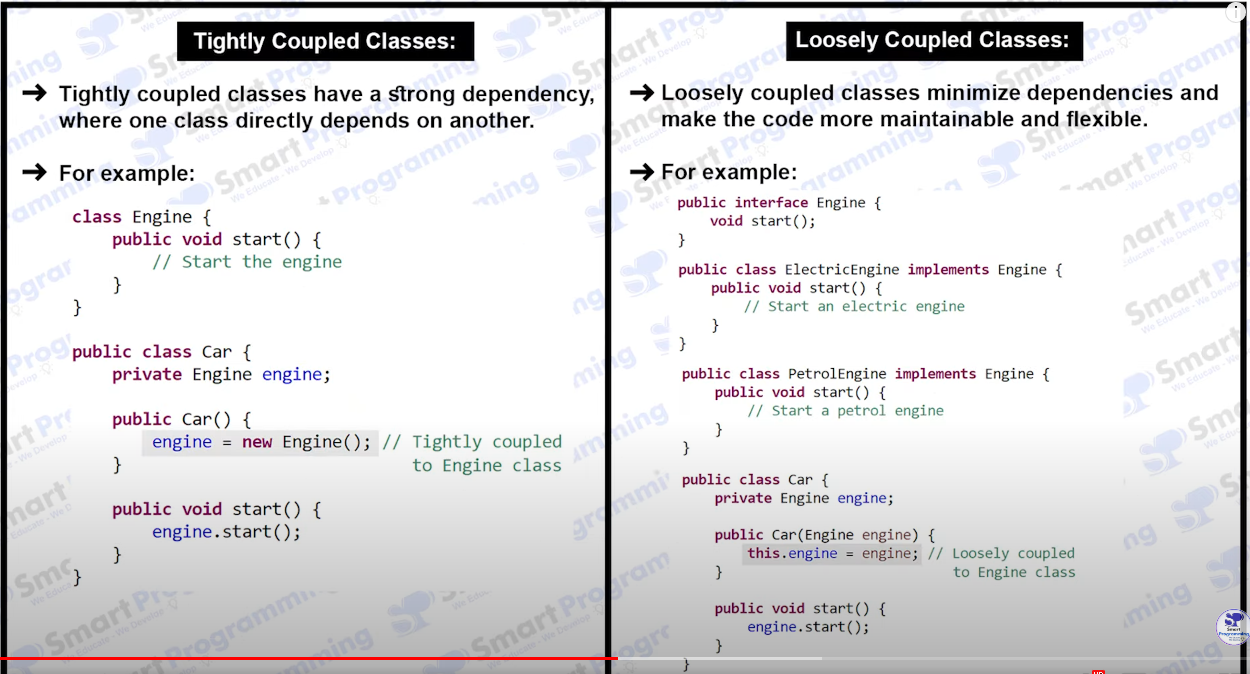
* Maven will do all above points automatically for us.

**What is Spring Framework?**

* It is a Frameworks of Framework.
* It helps developers to work their application rather than worrying about non-functional code.
* In simple way it will help to focus on our Business Logic rather than Worrying about other non-functional requirements.
* It minimizes boilerplate java code.
* **Spring Framework is a Dependency Injection Framework to make Java application Loosely Coupled.**
* Spring Provide IOC (Inversion Of Control) which help Dependency Injection.

**What is Dependency Injection?**

* It is Design Pattern.
* It inject one object into another object.
* Simply means one object dependency will inject in another Object.
* It is used to achieve loose coupling in Java.
* We can achieve Dependency Injection by 3 Ways 🡪 Setter Method and Constructor Method and Filed Level.



**Inversion Of Control (IOC)**

* Create the Object.
* Hold them in memory.
* And those Object Inject them in Another Object as Require.
* Complete Life cycle of Object creation to destruction is maintained by IOC Container.
* We need provide Beans information and Configuration Information to IOC.

**What is Spring Container?**

* Core Component(Heart) Like JVM
* Responsibilities
* Manage bean object
* Manage bean life cycle
* Dependency injection
* AOP
* Transaction Manager

Type – BeanFactory (Old) – Lazy Loading – when call getBean()

ApplicationContext (New) – Interface – Eager Loading – when application loading

**ApplicationContext**

– It represent bean container

– ApplicationContext (New) – Interface – Eager Loading – when application loading

– It extend BeanFactory

* ClassPathXMLApplicationContext
* AnnotationConfigApplicationContext
* FileSystemXMLApplicationContext

**SetterInjection - <property> Tag**

* Create Some Java Pojo **with Setter Method**
* Create XML file for Configuration which will provide instruction to spring container with name as config.xml
* Add in this xml file <bean> tag
* For Example –

**Primitive Type Injection**

If we have **Student.class**

Then add this tag in XML **config.xml** file.

**<bean class=”com.example.Student” name=”student”>**

**<property name=”studentName” value = “ABC” />**

**</bean>**

In main class write functionality for using this bean

**ApplicationContext context = new ClassPathXMLApplicationContext(“config.xml”);**

**Student student = (Student) context.getBean(“student”);**

**sout(student);**

**Reference Type Injection**

If we have **Student.class and Address.class**

Then add this tag in XML **config.xml** file.

**<bean class=”com.example.Student” name=”student”>**

**<property name=”studentName” value = “ABC” />**

**<property name=”studentAddress” ref = “address” />**

**</bean>**

**<bean class=”com.example.Address” name=”address”>**

**<property name=”city” value = “XYZ city”>**

**</bean>**

In main class write functionality for using this bean

**ApplicationContext context = new ClassPathXMLApplicationContext(“config.xml”);**

**Student student = (Student) context.getBean(“student”);**

**sout(student);**

**sout(student.getAddress);**

**ConstructorInjection - <constructor-arg> Tag**

* Create Some Java Pojo **with Contructor.**
* Create XML file for Configuration which will provide instruction to spring container with name as config.xml
* Add in this xml file <bean> tag
* For Example –

**Primitive Type Injection**

If we have **Student.class**

Then add this tag in XML **config.xml** file.

**<bean class=”com.example.Student” name=”student”>**

**< constructor-arg name=”studentName” value = “ABC” />**

**</bean>**

In main class write functionality for using this bean

**ApplicationContext context = new ClassPathXMLApplicationContext(“config.xml”);**

**Student student = (Student) context.getBean(“student”);**

**sout(student);**

**Reference Type Injection**

If we have **Student.class and Address.class**

Then add this tag in XML **config.xml** file.

**<bean class=”com.example.Student” name=”student”>**

**< constructor-arg name=”studentName” value = “ABC” />**

**< constructor-arg name=”studentAddress” ref = “address” />**

**</bean>**

**<bean class=”com.example.Address” name=”address”>**

**< constructor-arg name=”city” value = “XYZ city”>**

**</bean>**

In main class write functionality for using this bean

**ApplicationContext context = new ClassPathXMLApplicationContext(“config.xml”);**

**Student student = (Student) context.getBean(“student”);**

**sout(student);**

**sout(student.getAddress);**

**Ambiguity Problem in with Constructor Injection**

If we have class with method overloading constructor

For Example

Public void add(int a, int b)

{

System.out.println(a+b);

}

Public void add(double a, double b)

{

System.out.println(a+b);

}

Now we configure the bean in config.xml file

**<bean class=”com.example.add” name=”add”>**

**< constructor-arg value = “10”>**

**< constructor-arg value = “20”>**

**</bean>**

Now Here we get Ambiguity because it will search first match which int method but we have two method it can get confused.

Now we need to prevent this with “type” argument.

**<bean class=”com.example.add” name=”add”>**

**< constructor-arg value = “10” type=”int”>**

**< constructor-arg value = “20” type=”int”>**

**</bean>**

**Life Cycle of Spring Bean**

* Loading Bean Definition
* Bean Object Instantiation
* Bean Initialization
* Bean Destruction

**Scope of Bean: - @Scope** – It will tell scope of the bean means if we want to declare the scope of the Bean.

In that commonly we use singleton or prototype.

Type of Scope

**Singleton** – by default scope of spring bean – it will return same object reference every time in entire application.

**Prototype** – it will return always new object reference – it will create new object every time

**Request** – for HTTP request objects – it will create new instance for each HTTP request.

**Session** – for only session objects – it will create single instance of the bean per web socket session

**Globalsession**

**Spring Expression Language – SpEL**

It supports Parsing and executing expression with the help of @Value annotation.

For Example - @Value(“#{Expression}”)

@Value(“#{11 + 22}”) – it will return 33

@Value(“#{1 > 0}”) – it will return True.

**Spring JDBC**

Spring JDBC is a powerful mechanism to connect to the database and execute SQL queries.

JDBC is API to perform operations with Database.

**Spring MVC – Model View Controller**

It is sub framework of Spring Framework which is used to build a Web Application.

It is built on Top of Servlet API.

It follows Model – View – Controller Design pattern.

**Spring MVC Interceptor**

In simple word interceptor works process some operation in between sending the Request.

**Stereotype Annotation**

@Component Annotation

First we need to create POJO class with setter and getter and constructor.

Class Student

{

\\ Statements getter and setters

\\ toString Method

}

Now we create configuration file

Create XML file for Configuration which will provide instruction to spring container with name as config.xml

**Config.xml**

<context:component-scan base-package = “com.springcore.strereotype” />

After that we need to write annotation on class

@Component

Class Student

{

@Value(“Set Values for fields”)

\\ Statements getter and setters

\\ toString Method

}

