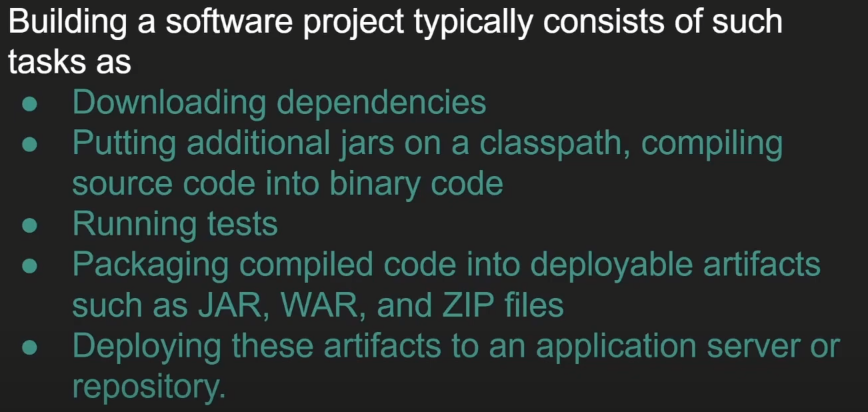
**Spring**

**What is Maven?**

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



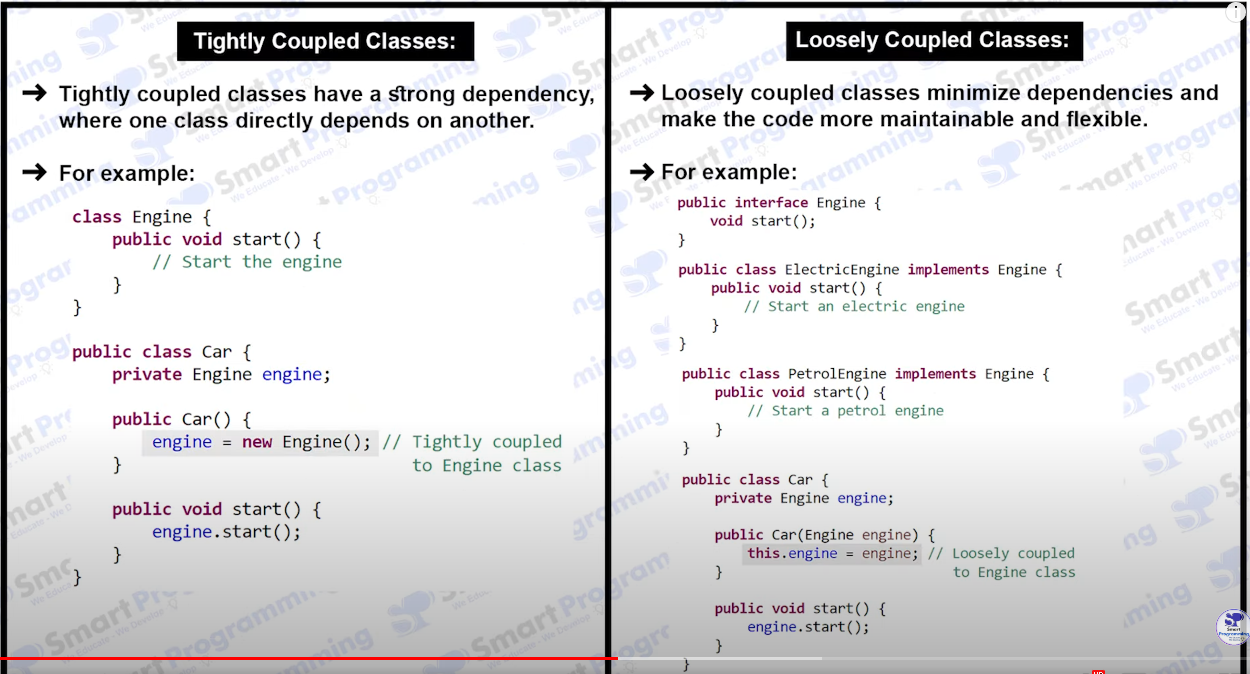
* Maven will do All above points automatically for us.

**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 2 Ways 🡪 Setter Method and Constructor Method.



**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 in 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.geBean(“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.geBean(“student”);**

**sout(student);**

**sout(student.getAddress);**

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

* Create Some Java Pojo **with Contructor**
* Create XML file in 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.geBean(“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.geBean(“student”);**

**sout(student);**

**sout(student.getAddress);**

**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.

**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.

**Spring MVC – Model View Controller**

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

It is build on Top of Servlet API.

**Interview Question and Answers**

**Why will you choose Spring Boot over Spring Framework?**

Spring Boot is nothing but an Spring Framework for Rapid Application Development with extra support of configuration and embedded server.

* Dependency Resolution
* Avoid additional configuration
* Embedded Tomcat, Jetty
* Provide Production ready features such as metrics, health checks

**What is RAD Rapid Application Development means?**

Rapid application development is modified Waterfall model which focus on development application in short span of Time.

Basically there is five phase of development in RAD

**Business Model** – In this we need to create business model for the product.

**Data Model** – After creating or understand business model we need to design data model. Where we create data objects and establishing relation in this data objects.

**Process Model** – with the help of both business model and Data model we need to process on this object like Adding, deleting, retrieving, Modifying.

**Application Generation** – Here we create actual Product prototype and ask the client to Agree on this build code project or not then we will take next step.

**Testing and Turnover** – Now we can test over product on client side if it works properly then ok If changes are required from client side the start over again whole processes known as Turnover.

**How we can change the server port no. in Spring Boot?**

In application.propertys file we can write server.port=8081.

**Can we replace the embed tomcat server in spring boot application?**

Yes, whith adding the dependency like **spring-boot-starter-jetty**.

**Can we disable web server in spring boot application?**

Yes, we can write in application.property file “**spring.main.web-application-type=none**”.

**How to disable Auto-configuration class?**

We can add @EnableAutoConfiguration(exclude=”excudingclasssname.class”).

**What is the use of Profile in Spring Boot?**

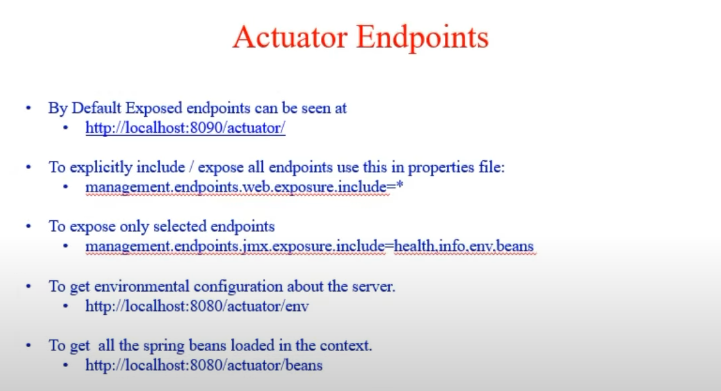
Dev environment

Prod environment

UAT or QA environment

**What is Spring Actuator? What are Advantages?**

* Actuator is a manufacturing term actually that is use for referring to mechanical devices to moving or controlling something.
* In Spring Boot, Actuator is addition feature that help us control manage and monitor our application spatially when its push to production.
* It include Auditing, Health, and matrices gathering means how many hit did we get in HTTP and many more.
* We can enable this feature bay adding the dependency “**spring-boor-starter-actuator**”.



What all Spring Boot starters you have used or what all modules you have worked on?

How will you run Spring Boot application?

What is the purpose of the @SpringBootApplication annotation in spring boot application?

Can I directly use above 3 annotation in my main class, instead of using @SpringBootApplication annotation, if yes will my application work as expected

What is AutoConfiguration in Spring Boot?

How can disable a specific auto configuration class in Spring Boot?

How can you customize the default configuration in Spring Boot?

How Spring Boot run() method works internally?

What is CommandLineRunner in Spring Boot?