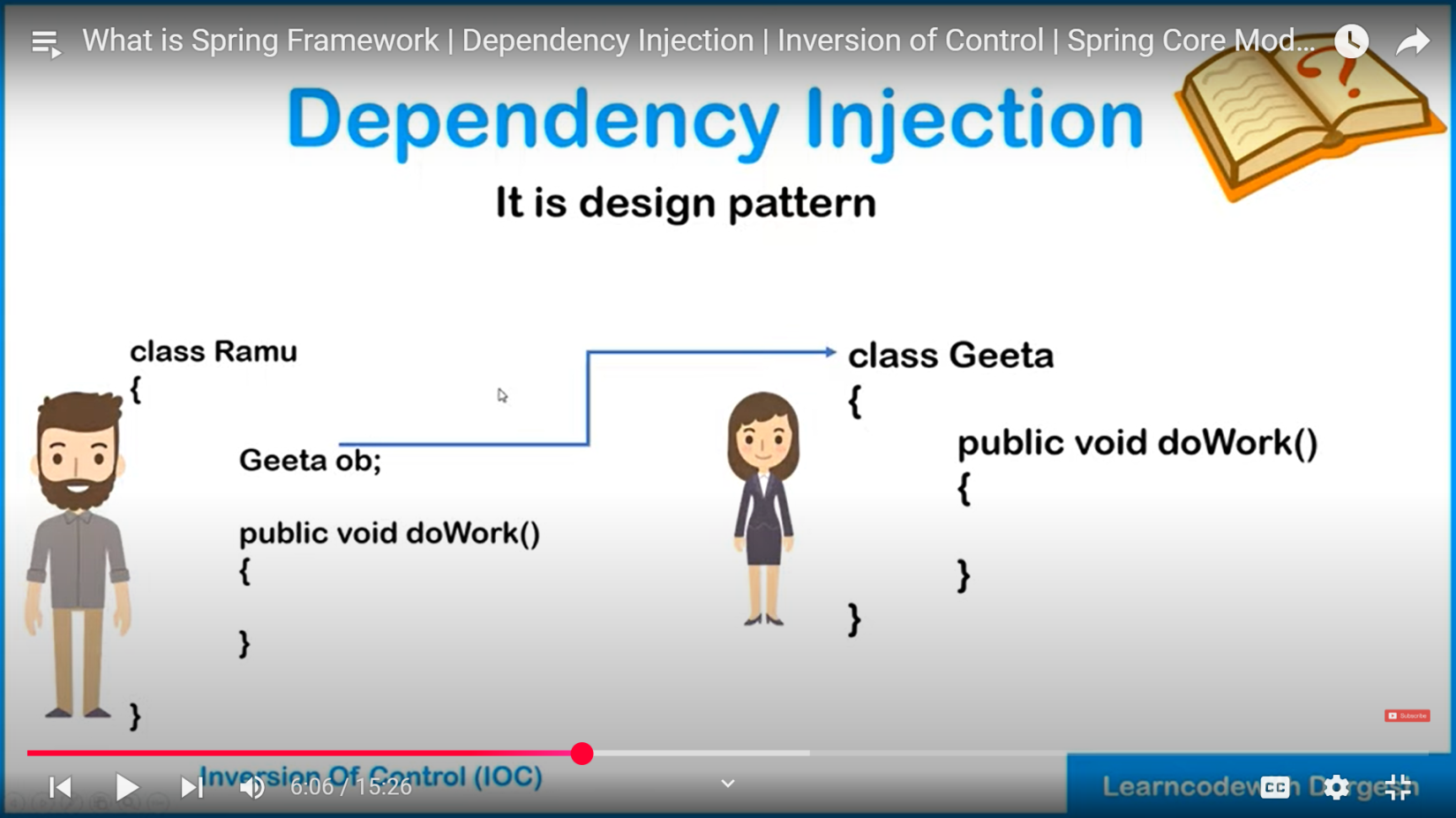
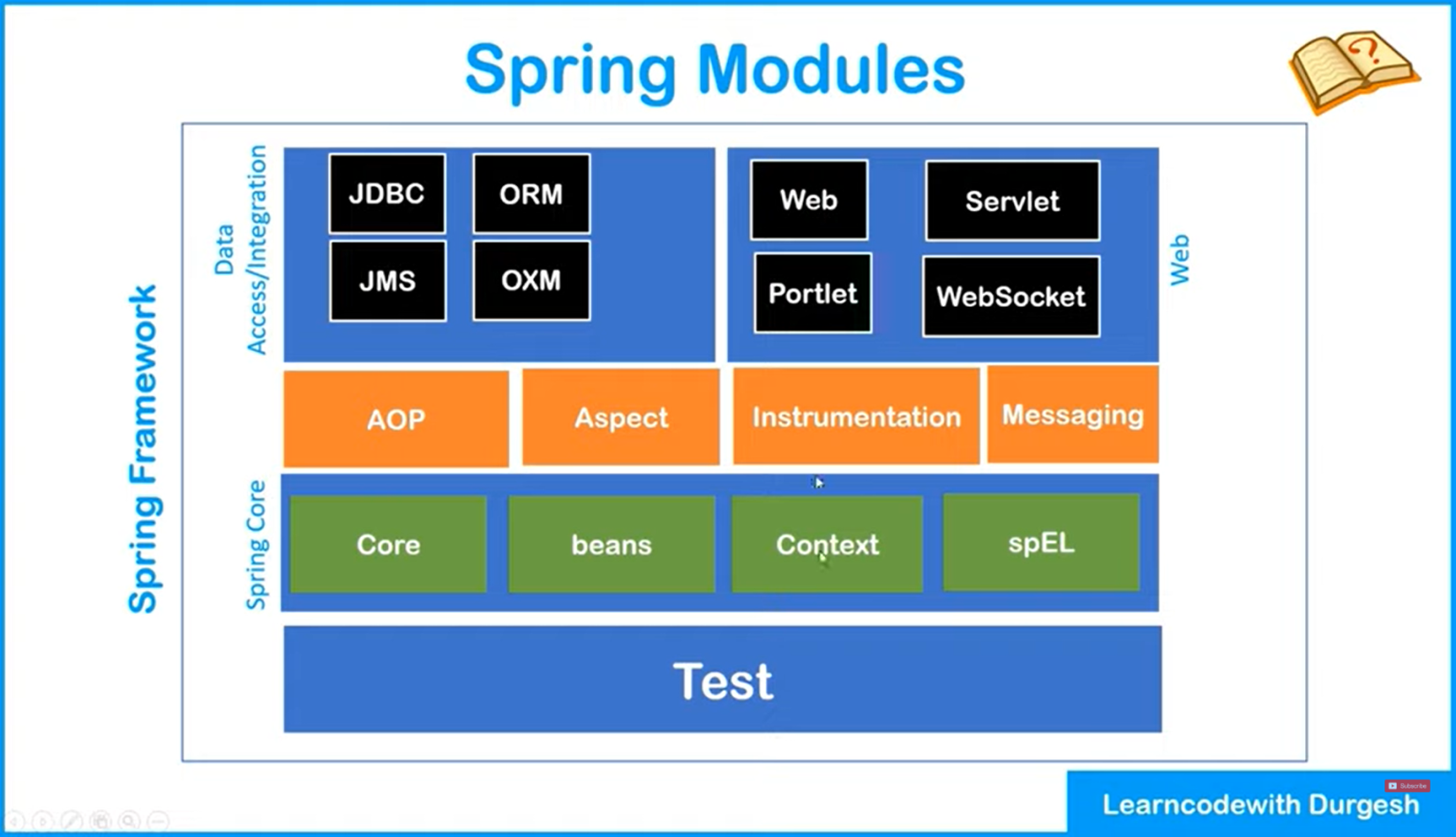
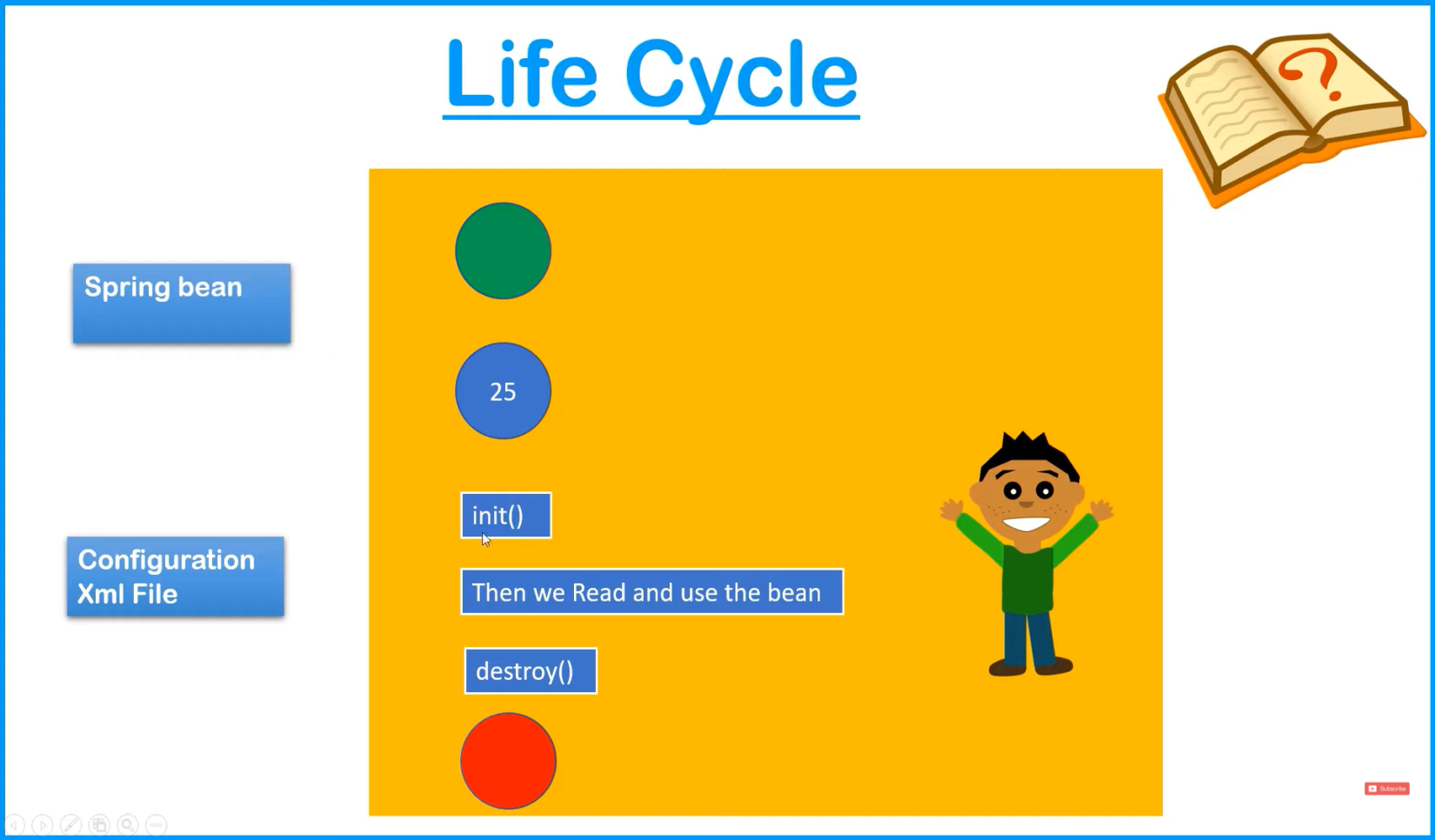
**Spring**

* **Spring** – is a dependency injection framework to make java applications loosely coupled.
* **Dependency Injection** – is a design pattern.   
  In JAVA to make applications we use classes. Suppose onr class needs object of another class. We can do it using Geeta ob = new Geeta(); but this will make it tightly coupled. To remove this, Spring does it internally, creating an object of Geeta and inject it to the object required in class Ramu.



* **Inversion Of Control (**principle**)–** Control on object creation is given to Spring, and it will create objects dynamically at runtime and inject them.
* **AOP –** Aspect Oriented Programming(Aspect, Instrumentation, Messaging) allows to define method interceptors and point cuts to decouple our code. It allows you to modularize cross-cutting concerns like logging, security, and transaction management, separating them from your core business logic, resulting in cleaner and more maintainable code.   
  **Interceptors –** suppose we want to perform certain tasks before or after any method. With its help we can do this.  
  ****
* **Spring IOC Container –** is a predefined program/ component whose job include creation of objects, hold created objects in memory, inject one object to another object, i.e., it maintains the lifecycle of the objects. Classes managed by this is known as Beans. We need to give 2 things to Spring IOC Container : Beans and config(xml file where we define beans and its dependencies)
* **ApplicationContext(interface) –** represents IOC container. It is implemented usually by 3 classes :   
  1. ClasspathXmlApplicationContext – from XML file  
  2.FileSystemApplicationContext  
  3. AnnotationConfigApplicationContext
* **Types of injection –** 1. Setter/ Property injection(calls all setter methods itself),   
  2. Constructor Injection
* **Life Cycle of Spring –** 1. Public void init() // after creating the bean, this is used to initialize several required things.  
  2. public void destroy() // for destroying things   
  ****

**Deletion of object**

**For Clean up purpose. Ex – closing db connection**

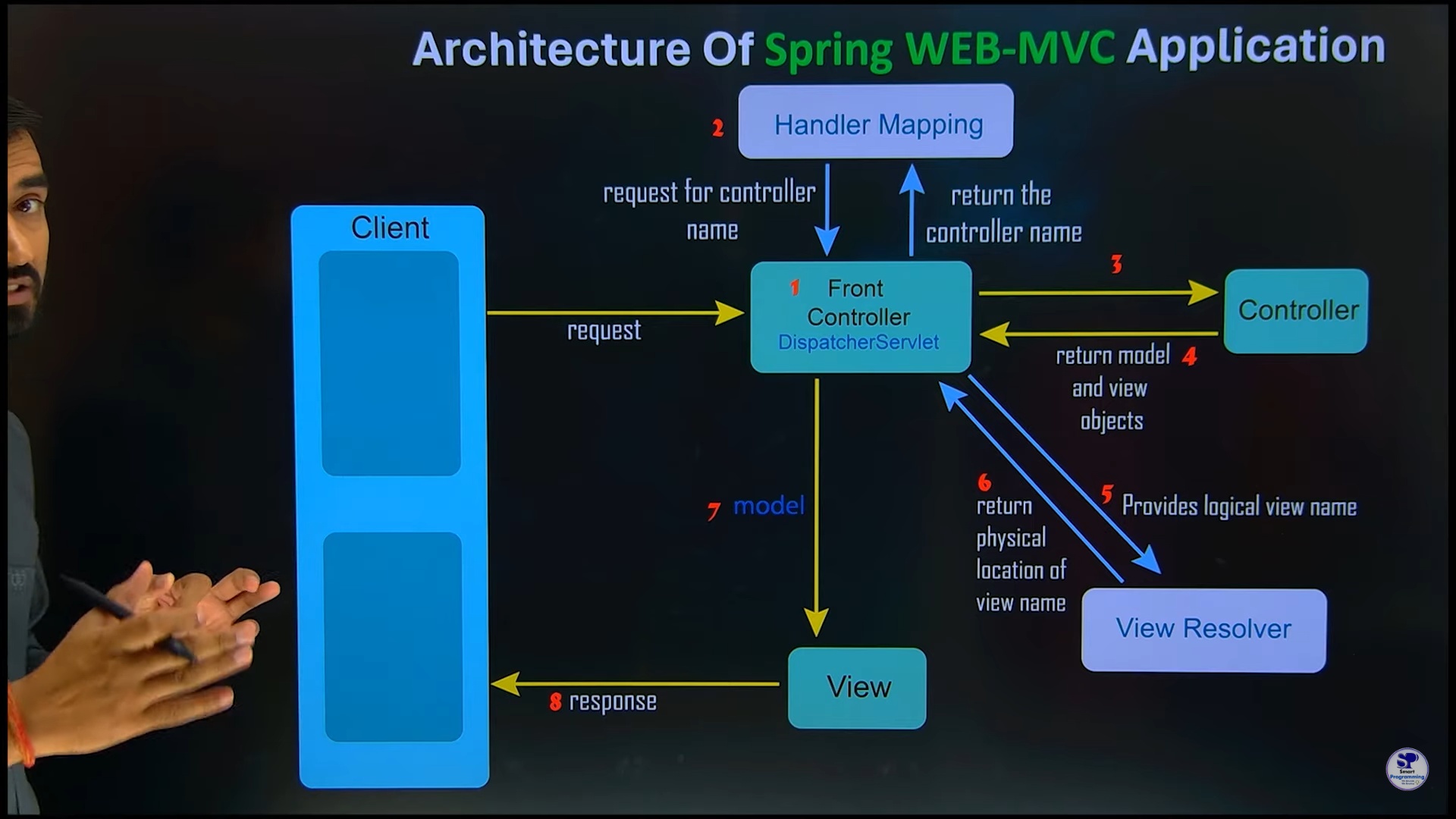
**Creation of object**

**Order**

Property Setting through XML file

* **AutoWiring – is a** feature in which spring container inject the dependencies automatically.

**Spring Boot**



[**Spring Boot**](https://www.geeksforgeeks.org/introduction-to-spring-boot/) is built on top of the conventional spring framework. So, it provides all the features of spring and is easier to use than spring. Spring Boot is a microservice-based framework and makes a production-ready application in very less time. In Spring Boot everything is auto configured. We just need to use proper configuration for utilizing a particular functionality. Spring Boot is very useful if we want to develop REST API.

**Magic –** There is nothing magic. When we add dependency(jar) in the pom.xml file, it pulls all the jars automatically

**Dependency injection** - is a technique used in object-oriented programming (OOP) to reduce the hardcoded dependencies between objects. A dependency in this context refers to a piece of code that relies on another resource to carry out its intended function.

**Bean** - A bean is an object that is instantiated, assembled, and managed by a Spring **IoC container**. Otherwise, a bean is simply one of many objects in your application.

**IoC container :**

**Inversion of Control (IoC) :** A design principle that transfers control of object creation, configuration, and lifecycle management from the application code to a container or framework.

**Dependency injection (DI)** : A specialized form of IoC where objects define their dependencies through constructor arguments, factory method arguments, or properties.

**Spring IoC container :** Responsible for instantiating, configuring, and assembling objects known as beans.

**Beans** - Objects managed by the Spring IoC container, created with configuration metadata supplied to the container. Similar to standard objects just exceptions : a.) all properties are private, b.) a public no-argument constructor, c.)Implements Serializable

The Spring IoC container helps make code more modular, testable, and maintainable. It can be used to:

**Create and provision instances -** The Spring container can create and provision managed instances of a class when needed.

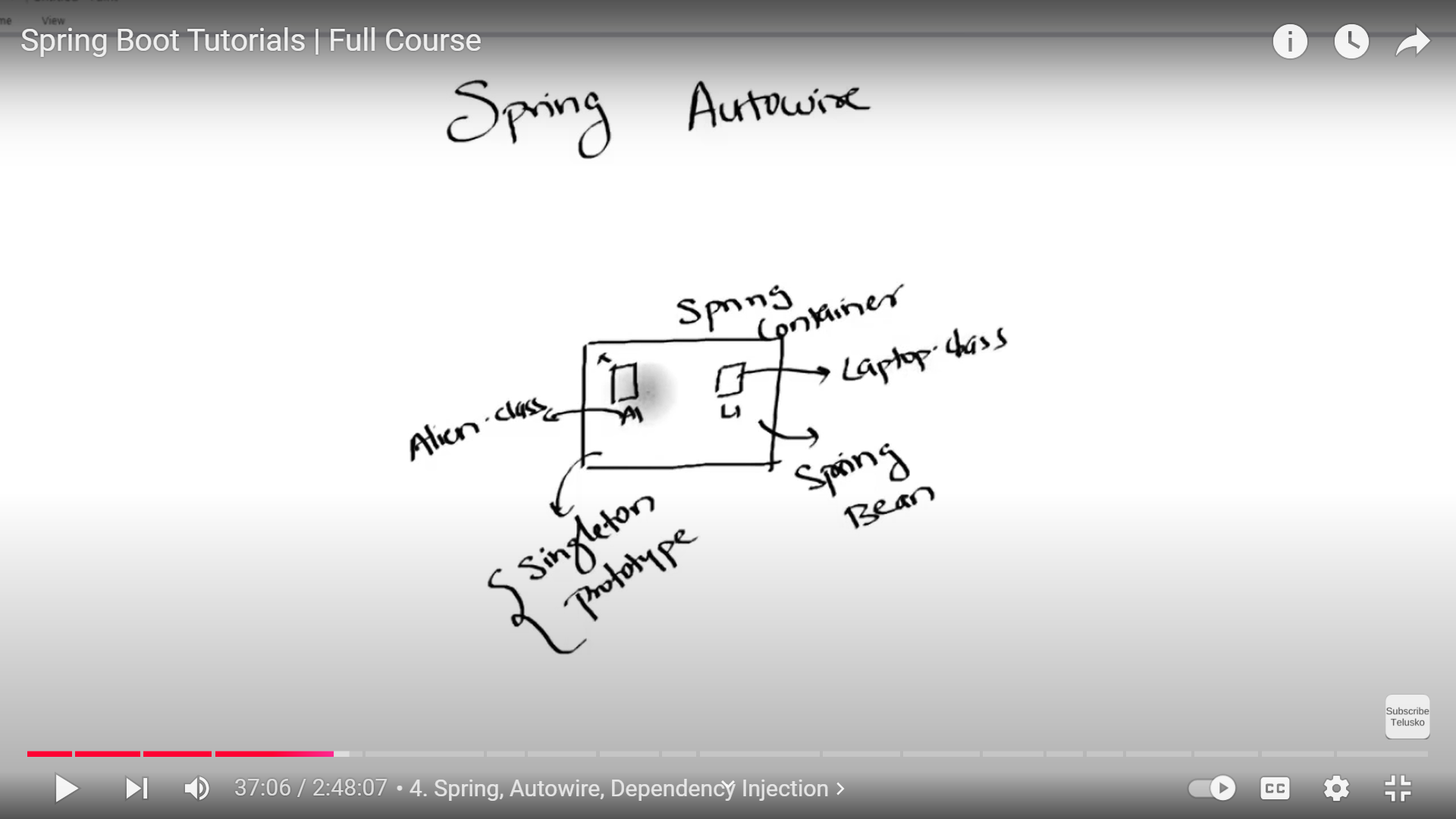
**Adjust ingredients -** If you want a different type of cake, you can tell the Spring container, and it will adjust the ingredients and steps.

**Use test ingredients** - You can ask the Spring container to use test ingredients when trying out new recipes.

**Change components** - You can change the type of cake (or components in your application) by telling the Spring container what you want.

To create a new object in JAVA we do:  
Alien obj = new Alien(), but in Spring we don’t need focus on manual creation of objects but the logic.  
***SpringApplication.run(FirstApplication.class, args);*** // this run method returns an object of ***ConfigurableApplicationContext.***

There are two types of objects in bean - Singleton and prototype. In singleton the object(single instance) is already created for future use but in prototype this is not the case; object will be created when called.



When we mention Component, The object belongs to the Spring Bean. Any objects belonging to Spring Bean can be of two types – Singleton and Prototype.

Autowired is used to make one object understand that there is another object of another class

Content Negotiation –

**ORM** – helps developers to access DB without writing sql instructions. It fetches rows and gives developers in forms of Objects and vice versa. **JPA** is an ORM tool.

**Course – Navin Reddy**

**MAVEN(**project management tool**)**

Normally we need to download jar files to use dependencies in our project, but using **maven** we don’t need to download those dependencies by ourselves but just need to mention the dependency and the version that we require.

Maven has a folder .m2

Whenever a dependency is added in pom.xml file, it is search in the .m2 folder. If not found it downloads dependency from the maven repository

**HIBERNATE** (ORM framework ) (Object Relation Mapping)

– Though JDBC is enough we use this to increase the productivity og the developer. We won’t have to write the sql queries in Hibernate manually

* Directly puts the object in the database. Allows developers to work with data using programming language constructs instead of writing SQL queries
* Hibernate provides 2 caches – L1 & L2

**HQL(*Hibernate Query Language*)** -

|  |  |
| --- | --- |
| SQL | HQL |
| select \* from *student;* | from *student;* |
|  | from *student* where name like ?1 and x = ?2 |
| select name from *student;* | select name from *student;* |

**What is JPA (Java Persistence API)?**

JPA (Java Persistence API) is a **Java specification** that provides a way to manage relational data in Java applications using **ORM (Object-Relational Mapping)**. It acts as an **abstraction layer** over JDBC, making it easier to interact with databases.

**JPA vs Hibernate vs Spring Data JPA**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Definition | Specification for ORM | JPA implementation | Abstraction over JPA |
| Provides | Interfaces & annotations | ORM functionality | Simplifies database interactions |
| Requires Implementation? | Yes | No, it is an implementation | Uses JPA/Hibernate internally |
| Commonly Used With | Hibernate, EclipseLink | JPA | JPA, Hibernate |

JPA uses Domain Specific Language, that will generate certain methods behind the scene. Like findBy*ColumnName*(String value),

**Spring Boot Docker**

1. docker run –it openjdk:latest //use the image of openjdk
2. mvn package //make jar file of the spring boot project  
   //we can run the jar file by `*java -jar target\OAuth-After-Docker.jar`* on terminal
3. put .jar in the container of openjdk
4. to create the image of this container

**MICROSERVICES** (Every microservice is a project)

* **Needed stuffs** – Load Balancer, API Gateway, Registry, Failed Fast
* a **Eureka Server** acts as a service registry, allowing microservices to register themselves and discover other services, facilitating communication and dynamic scaling
* **OpenFeign** is a library provided by the Spring framework to invoke web services to a client

1. Make all the services independent, not dependent on another database(say)
2. We can run multiple instances of the services
3. If we are planning to deploy our application on kubernetes there is generally no use of setting up eureka server, since it performs all the tasks performed by service-registry

