**Introduction:**

Welcome back to today's session is on Spring Boot and my name is Ajay Takur

so before going for spring boot

I want to put some overview on what is the approach to develop an application without using framework?

In Traditional approach for instance let's take java as an language.

In 1990s we used to have GUI based applications.

To develop GUI applications we have leveraged the Java Standard Edition

which provided with core functionalities

like Input/output operations, networking, concurrency.

And Java Standard Edition also includes libraries for handling data structures, working with databases.

let's go to year 2000 so in that time

if you want to create a project in Java for enterprise you will be using lot of Java EE features

and one of them is EJB and it was one of the best technology available at that point

and the only problem is you know messaging and lookup for the entities it becomes difficult to manage

when you are working with entities so they are heavy and you

had to implement special J2EE interfaces and implement special lifecycle methods.

This made it hard to use for the developers.

In earlier versions of EJB (Enterprise JavaBeans), EJB 2.0 was released in the year 2001.

you needed to implement special J2EE (Java 2 Platform, Enterprise Edition) interfaces

to define the lifecycle and callback methods for EJB components.

But EJB 3.0 which was released in year 2006 has come up with

which reduced the need for implementing special interfaces and XML configuration.

But by then lots of industry has started using Spring Framework

and that's where the Spring framework got created in the year 2003.

This picture here shows how the spring is evoluted overtime

Key mile stones and major releases.

The first release of Spring boot was in 2014

which was developed by the pivotal team and been maintaining since then to till date.

And it's almost a decade and for every four years there will be a major release happening.

Move to next slide

Now What is Spring Framework?

It's an Free and Open source Software developed using java language.

It was created by Rod Johnson

Why to use Spring?

**Some Features of Spring:**

**Modular:**

Spring is made up of a number of modules, each of which provides a specific set of features.

You can use just the modules that you need for your application,

which makes your application smaller and faster.

**With this we can say it is also a**

**Lightweight:**

Spring is very lightweight with respect to its jar size and functionality.

Because you don't need all of Spring to use part of it.

For example, you can use Spring JDBC without Spring MVC.

A Spring MVC is a module within the larger Spring Framework

that is used to build web applications in Java.

It follows the MVC design pattern, which separates the application into three main components:

Model: It has got the business data and state.

View: It is responsible for rendering the user interface based on the data provided by the model.

Controller: It handles user input, processes requests and

It also selects the appropriate view to render the response.

**Inversion of Control (IoC) :**

The core part of the Spring Framework is the IoC container,

it manages the instantiation of Java objects and lifecycle of Java objects (beans).

Instead of developers having to create and manage objects manually,

the IoC container takes care of this, making the application more loosely coupled and easier to maintain.

How the IoC container works internally in the Spring Framework:

**First thing is to Define the Bean( Bean is nothing but just java object here)**

Bean definitions can be defined in XML configuration files ie.**applicationContext.xml ,**

Java annotations, or Java configuration classes.

Just need to mention how beans should be created, configured, and managed.

So that IOC container will pick up those configurations at runtime.

**Second thing is Bean Instantiation**

When the application starts up, the IoC container reads the bean definitions

and instantiates the beans as per the configuration.

It creates a bean instance for each bean definition.

For instance: If you have created 10 bean definitions those many

beans objects will be created at runtime

and we can mention whether those beans to be created at the

time of application startup or at the time of request call.

**Dependency Injection :**

if you are working on some advanced project of course you will be using some design patterns

and one of the design patterns which we come across is dependency injection

doesn't matter which language we use maybe Java c-sharp PHP.

if you are working with any programming let's say object-oriented

so we build objects right and those objects are dependent on other objects.

This feature allows you to develop loose coupled applications.

Therefore, the unit testing of these loosely coupled applications becomes easier.

This also allows the developer to swap out some of the modules according to its need.

So the IoC container then injects dependencies into the beans based on their configuration.

Dependencies can be injected using constructor injection, setter injection,

or field injection, depending on the bean configuration.

What is the Scope of the beans

Beans can be singletons, meaning that only one instance of the service is created

and shared throughout the application context.

Or Beans can be Multiple instances meaning as on when

the request comes each time a new instance is created.

**Container:**

Why Container ?

For Bean Lifecycle Management.

The Spring framework designs and manages the lifecycle and configurations of application objects.

The IoC container manages the lifecycle of beans, including their initialization and destruction.

It calls lifecycle callback methods, such as @PostConstruct and

@PreDestroy annotated methods, to perform initialization and cleanup tasks.

so let's say you're writing a business service right you write your business service

that functionality the code containing only the logic that's specific to your business application

and then what you do is annotate that business service that class but something like

at service and that lets spring know that you intend that class to be a service

and it applies a whole lot of things to that class

and it manages the life cycle of the class so it acts as a service so what you do

when you're building your application is you focus on your business problem

and you let Spring handle the rest so it provides that model for you to build those applications.

Let's take for example how @Service works internally in Spring:

1. Component Scanning:

When the Spring application starts up, it scans the classpath for classes

annotated with @Service (along with other stereotype annotations

like @Component, @Repository, and @Controller).

2. Bean Registration:

The classes annotated with @Service are identified as Spring beans,

and instances of these classes are created and managed by the Spring container.

3. Dependency Injection:

Dependencies of the @Service bean, if any, are injected by the Spring container

using either constructor injection, setter injection, or field injection, depending on the configuration.

4. Singleton Scope:

By default, @Service beans are singletons,

meaning that only one instance of the service is created and shared throughout the application context.

This can be overridden by using the @Scope annotation with a different scope, such as prototype.

Transactional Support:

@Service is often used to annotate classes that contain business logic,

and it is common to use the @Transactional annotation within

@Service classes to demarcate transaction boundaries.

**Integration:**

A great thing about this framework is that it doesn’t try to solve the problems that have already been solved.

It has ability to integrate other frameworks For example, this could

include Struts, IBATIS, Hibernate frameworks which provides a solution to greater problems.

So there are so many features where you can achieve any business purpose

and now we know why spring is the best framework now there is one Problem with it

when you are building big applications right and when you say big applications

you will be needing some external Java files so that's one thing you have to work with Java files

the second thing is the configuration and trust me when you're working on the enterprise application

you need to add a lot of configuration and that's what the problem starts you know

because if You're working on a project you want to focus more on the convention

meaning here business logic rather than configuration

because we are developers we don't want to spend much time in the configuration

I'm not saying we can like skip all the configuration but we focus more

on the coding part and that's where spring boot come to picture

Moving to next slide please

With that we will see

**What is Spring Boot?**

Spring boot is a free and open source framework built using java language.

It is used to create an enterprise applications of both monolithic and microservices architecture.

Spring Boot can help developers build and deliver software more quickly and with less code.

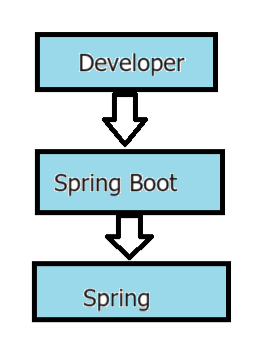
There are two words here spring and boot the first part spring is something

we must be familiar with it's the spring framework it's a framework which lets you write enterprises java applications.

And other part Boot is bootstrapping the spring applications meaning with minimal/zero configurations.

Since Spring boot is built on top of the spring framework, is widely used to develop REST APIs

and it is easy to integrate with any User interface Frameworks like Angular/React JS/Ext JS.



now again spring boot is not a replacement for Spring Framework

because as a developer you are still using Spring Framework behind the scenes okay

but in between just to help you we have spring boot because spring boot

says hey Developer I know you want to work on a project

and spring is the best framework the only thing is they said there are some jar files

which you have to add and there's a configuration you do and don't worry

and they're for you so spring boot says I will give you dependency and

I will give you the configuration awesome right

now the main idea of spring boot is to give you a production ready application

so the moment you put a spring boot project you don't have to do Any configuration it is runnable

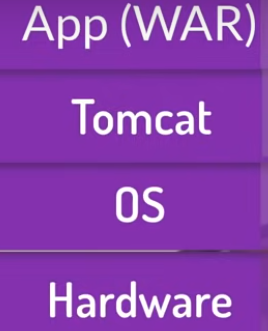
and you can deploy it on the on the server it will do nothing but it will work you don't have to do any configuration

and the more features you want you can add some features and for those features you can do some small configuration

but all the basic configurations will be done by spring boot ok.

**Why to choose Spring Boot?**

**No Server Dependency:**



now there is one more thing you know let's say we have we are not working on spring boot

we are working With normal spring project and we want to make a web application

and if you deploy the application what you will do is of course you need a server

maybe a virtual server or the cloud server

and if you have a server you need to install a Linux server ok so that's done

so we have a physical server then we have a Linux OS and on that OS of course

you'll be installing a web server or maybe a app server it may be tomcat

or the Glass Fish and then on that tomcat we'll be deploying the war file

which you create for web application so many layers right and



nowadays we are moving towards micro services and we want to make it bit easier for us right

and that's why spring boot says hey you don't have to actually do all

those things by yourself it's bringing boot it provides you a embedded server

which means the application jar files which only create ok we will not be making jar file

and Inside the jar file will be having Tomcat Oh we'll be having Tomcat inside a jar file

that's why that's an embedded Tomcat on embedded server you can say so spring boot

says your jar file will have a embedded Tomcat which means you can run your project

and on any JVM you don't basically need a web server there

So the other applications using spring will rely on external web server

whereas spring boot has got embedded application servers to quickly deploy and test out the applications.

now spring boot would provide to so many features two of my favourite is the first one is starter projects

because if you want a project let's say if you want to Make a web application spring boot will say okay

because I'd spring boot starter web it will give you a web project and if you want maybe

if you want to work with JDBC spring boots says okay don't worry we have a spring boot starter JDBC

you can simply use that and it will give you a ready project

what about the configuration spring boot also has spring boot auto configuration

which will do the configuration for you so that

You can focus on the convention part not on configuration

**No XML Configuration:**

So let's say if you want to do some manual configuration is it possible that's just spring boot provides you the XML files

and the answer is no spring boot will not generate any XML file for you okay

because there is no XML file then how would you modify if you want to do some configuration

and that's where we have A special file which is application dot properties

you can add the properties in the form of key value pair which is quite simple now in application.properties file.

And also Spring Boot supports YAML as an alternative to properties files for configuring your application.

YAML files offer a more concise and readable way to configure your application compared to properties files,

especially for complex configurations with nested properties.

In regards to properties file there is a feature called profiling in Spring Boot,

that profiling will allows you to customize the application's configuration based

on the environment in which it is running.

This includes different environments like development , QA, UAT and Prod.

you just need to define different sets of properties for different profiles

For eg: application-dev.properties, application-qa.properties etc.,

Profiles can be activated in several ways, one of them is just mention

spring.profiles.active property in application.properties

And we have other ways to define like Programmatic, Environmental variables etc.,

**RAD is nothing but Rapid Application Development**:

From development perspective we have some Dev tools

This Dev tool will helps in improving the developer productivity And that

can save a lot of time during development by not having to deploy the application

in server every time there is a change made locally.

Some features provided by Dev Tools are:

1. Automatic Restart: When you make a change to your code you can see the results of your changes immediately, without having to manually restart the application.
2. Live reload: You can see the results of your changes in the browser immediately, without having to manually refresh the page.

With Spring Boot, developers can quickly bootstrap their projects and focus on writing business logic instead of dealing with infrastructure and boilerplate configuration.

**Convention Over configuration:**

It follows the "opinionated" approach, which means it provides default configurations and conventions, but you can override them if needed. This helps in reducing the development time and effort.

Since there is no need of writing xml configurations and boiler plate code with that the code maintenance is also easy. Any engineer can go and modify the properties file since it's easy to understand and readability.

**In Memory DB:**

With the spring boot in memory databases are databases that store data in RAM instead of on disk.

As soon as the application is started the embedded databases like H2, Derby can be used to perform operations on data and the data is lost when the application is shutdown.

We can use the in memory DB for

1. Caching: They can store frequently accessed data to reduce the load on backend databases and improve response times.

2. Session Management: In a microservices environment, where sessions need to be managed across multiple instances of a service, an in-memory database can be used to store session data. This ensures that session data is quickly accessible and shared among instances.

3. Transient Data Storage: Microservices often need to store transient data that is not critical for long-term persistence. In-memory databases can be used to store such data, providing fast access and low latency.

**Packaging and Deployment:**

Spring Boot simplifies the packaging and deployment of applications. It includes an embedded server (like Tomcat, Jetty, or Undertow), allowing you to package your application as a standalone JAR or WAR file with an embedded server, making deployment straightforward.

Once deployment is done the Production Workflow needs to be monitored

So for this there is an Actuator which is used to monitor and manage the Spring boot Applications.

Some features provided by Actuator are:

1. Health Checks: These health checks can be used to determine if the application is up and running and if it is responding to requests.
2. Metrics: These metrics can be used to track the number of request being processed the response times and the memory usage.
3. Auditing: Thes Auditing can be used to trach who is accessing your application and what they are doing.
4. Management: Can be used to start, stop and restart the applications.

These Actuator exposes these features via a RESTFUL endpoints you just need to add the Spring-boot-starter-actuator dependency to your project.

**Community Support**:

Spring Boot has a large and active community, with extensive documentation, tutorials, and resources available.

This makes it easy for developers to get started with Spring Boot and look help when needed. This community ensures timely updates, bug fixes, and continuous improvement of the framework.

So overall if you look at the spring boot is very light in terms of its size of the jar, or startup time. For instance if Spring framework only uses 30 MB to run while other frameworks takes 80 MB to run the application.

And Spring boot is also said to be non-invasive means it does not force a programmer to extend or implement any inbuilt APIs.

In the right the applications using Spring boot in ICS BBD are one is Proxy Disclosure which is being migrated from struts to Spring boot.

And other one is Proxy Vote.

And Proxy Edge and Proxy work bench are the upcoming projects in pipeline to use Spring boot.

Here are some Reference materials to go through.

***Slide 6 Where to use Spring Boot?***

**Web Applications:**

Spring Boot can be used to build web applications, which can range from simple to complex web applications.

It provides a wide range of features for web development, such as embedded servers.

The web applications using frameworks such as Spring MVC and Spring boot web flux.

You choose to go with Spring MVC if your communication model is synchronous.

To use Spring MVC in a Spring Boot application, you typically add the spring-boot-starter-web dependency to your project's build file.

And for Spring Boot web flux if your communication model with other systems are asynchronous then you will use spring web flux and its used to create a fully asynchronous and non-blocking applications.

Spring Boot Web Flux follows the Reactive Programming Model

Reactive programming allows you to build applications that are more resilient to failure, more responsive to user input, and more scalable than traditional programming models.

Its non-blocking Input output model. Meaning Asynchronous processing means that the processing of an event does not block the processing of other events.

We use Reactive programming model when dealing with large streams of data that are constantly changing, such as data from sensors or user inputs. It provides a highly interactive for the end user.

**In Microservices Architecture**:

Microservices are those which are designed as converting large module into smaller and independent modules, each module is responsible for a specific business need.

If you are building a microservices-based architecture, Spring Boot provides excellent support.

Its lightweight nature and embedded server make it easy to build and deploy individual microservices.

Each microservice in a microservices architecture is implemented as a separate Spring Boot application.

This allows each microservice to be independently developed, deployed, and scaled. Each Microservice can be developed in different set of tech stack.

**API Development**: To build RESTful APIs supporting both xml and json.

It provides features like Spring MVC for creating web endpoints, and Spring Data REST for easily exposing JPA repositories as RESTful services.

**Batch Processing:**

Spring Boot provides support for batch processing through Spring Batch.

It simplifies the development of batch jobs for processing large volumes of data.

**Integration with Messaging Systems:**

Spring boot making it easy to build messaging-based applications.

It provides abstractions and templates for sending and receiving messages with messaging systems like Apache Kafka, RabbitMQ, and ActiveMQ.

**Integration with NoSQL and Big Data Technologies:**

It provides abstractions to access databases like MongoDB, Redis .

For MongoDB use the dependency called as spring-boot-starter-data-mongodb

**Slide 7 Spring Boot Annotations**

Spring Boot has provided a lots of annotations that can be used to replace all xml configurations which makes developer life easy.

These are some of the basic spring boot and web annotations that are used to configure a web application.

Some of them are class level and method level annotations.

So for @SpringBootApplication has to be one and only single annotation in overall project where in it will act as a entry point for an spring boot application.

It tells Spring boot to scan the class path for other spring components

**@EnableAutoConfiguration**

has to be one and only single annotation present in the main class for the whole application.

It takes cares of certain things like

Look at the class path entries defined by the developer in pom.xml to find libraries, frameworks.

For instance Spring JPA is used to get DB connection and make it readily available to use it in the application code

The boilerplate code and configuration required in your application is reduced.

You can override also the default auto-configurations if required.

By adding your own properties for instance in the application.properties (or application.yml) file in

the src/main/resources directory of your project.

**@Controller**

This is an class level annotation used in controllers are responsible for handling HTTP requests and responses.

It helps in the interaction between the client (browser) and the server (application).

We have n number of controller classes in application and each file has its own responsibilities to perform a task.

**@ResponseBody**

This is useful when you want to return data from a controller be it a JSON, XML.

But most of them prefer to have JSON instead of XML. As the JSON has better advantages over XML usages.

**@RequestMapping**

This annotation is used to map web requests to specific methods in a controller class. It allows you to define how incoming HTTP requests should be handled based on the request URL, HTTP method, and other request parameters.

**@RequestParam:**

It allows you to extract request parameters from the URL or form data and use them in your controller method.

**@PathVariable**

This annotation is used to extract values from the URI path and map them to method parameters in a method for a controller.

These are some of the annotations. Next Slide

**Slide 8 Spring Boot Versions**

The Current stable version of Spring Boot is 3.2.2 as of today and it requires a Java 17 as minimum version.

You can explore the previous releases features from the given link .

Spring boot first version came into existence of 2018.

Every Spring boot version is dependent on Java Version for instance Spring boot 2.x version supports only Java 8/11 versions.

**Slide 9 Market Trends of Frameworks**

There are large number of frameworks that developers leverage to build backend enterprise applications.

But Drop wizard, Quark us, Micronaut on the right are the top 4 back end frameworks which has impressed the world wide developers at large.

Each has its own trade-offs.

**From**

**Framework End:**

Drop wizard is used for only building RESTful web services.

Whereas Spring Boot is to develop web applications, RESTful services, and more.

And From

**Embedded Servers:**

Drop wizard uses Jetty as its embedded servlet container by default.

And Spring Boot supports multiple embedded servlet containers, including Tomcat, Jetty, and Undertow. You can choose the embedded container based on your requirements.

The best framework for you will depend on your specific needs.

**Slide 10 Most of the Companies using Spring Boot**

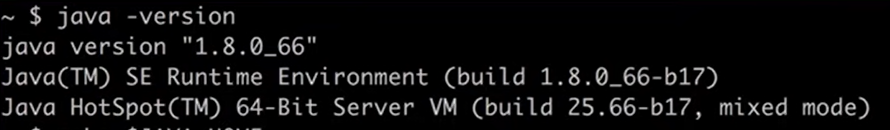
Next Slide

**Setup of Development Env to create Spring boot apps:**

The first thing that you're gonna need is the Java SDK. A minimum of JAV 8 or more is required to go with spring boot.

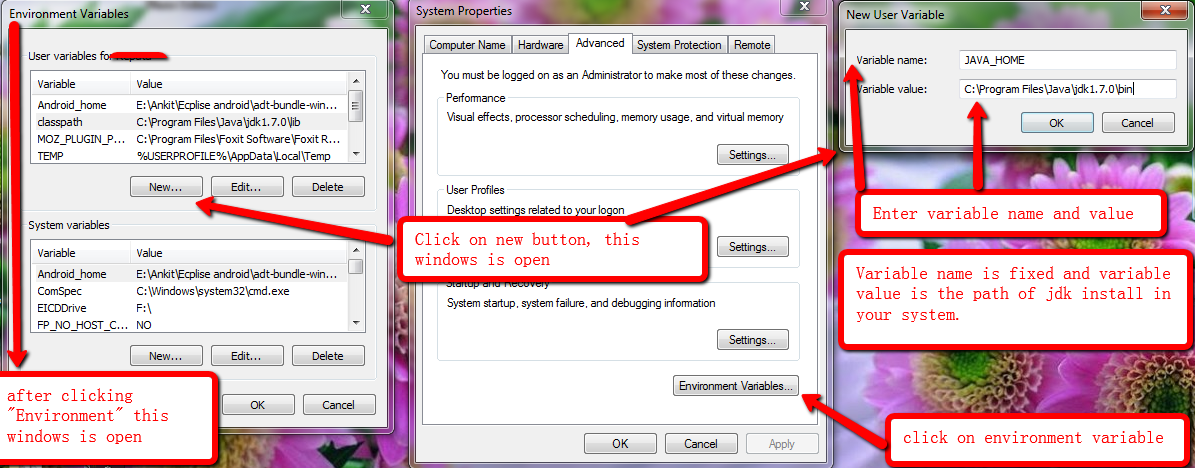
<https://www.oracle.com/java/technologies/downloads/>

so go ahead to the Oracle website and download the java 8 jdk if you don't Have it already and when you're done you should be able to open the command prompt and go to Java - version and you should be able to see the Java version 1.8



it's also helpful to make sure that the Java home is set

you can verify this on Windows by going to the environment variables and making sure that the Java home environment variable



**Spring STS:**

STS is a tool to develop Spring apps it's a flavour of Eclipse which has been tweaked to work with spring applications it has some spring specific features

Download :<https://spring.io/tools/>

Or Choose to download any of the IDE (such as IntelliJ or Eclipse).

**Start Development:**

1. Create a new Spring Boot project using either Spring Initializr ( <https://start.spring.io/>) or the Spring Boot CLI.

The minimum Java version for Spring boot 3.2.2 is Java 17 and also works with Java 21 which is latest version as of today

If you are currently using java 8 or java 11 you will need to upgrade your JDK before you can develop spring Boot 3.0 applications.

However, you can still use the java 8 or java 11 with Spring Boot 2.7.10 or earlier versions.

Spring 2.7.10 is the latest version that supports Java 8.

1. Add necessary dependencies to your project's configuration file (pom.xml for Maven or build.gradle for Gradle) based on your application requirements.

1. Write your application logic, controllers, services, and repositories using Spring Boot's annotations and conventions.

1. Run and test your application using the embedded server provided by Spring Boot.

1. Package your application as a standalone JAR file using the build tool of your choice.

1. Deploy and run your application in a production environment or as a Docker container.

**Conclusion:**

Spring Boot is easy and allows developers to focus on building business logic rather than dealing with infrastructure and configuration.

We will wrap up with this here now.

**QA?**