For classnotes and backup videos plz Join in Google Classroom :
https://classroom.google.com/c/NzY3MTQ0MTgzNjk3?cjc=g4bzorum

==========
Spring Boot & Microservices
==========

1) Your trainer

2) Pre-requisites

3) Who should join

4) Course Content (Road map)

5) Course Details

6) Q & A

========
Trainer
========

Name: Mr. Ashok

12+ Years of Exp in IT industry

Product based company from Hyd location (Banking Sector)

10 Years of exp in Software Trainings

Skills : Java + J2EE + Spring + SpringBoot + Microservices

Angular

Linux + AWS + DevOps Tools

Ashok IT started in 2020

==========
Pre-requisites
==========

1) Core Java

- Language Fundamentals
- OOPS
- Collections
- Exception Handling
- File Operations
- Multi Threading
- Lambda Expressions
- Stream API

2) Adv Java

- JDBC
- Servlets API

- MVC

3) Database (SQL)

4) Web Development (HTML & CSS)

==========
Who should join for this course ?
==========

=> Without asking "springboot and microservices quesions" java interview will not be completed.

=> If you don't mention "springboot and microservices" in resume, company will not consider your profile for interview.

- Fresher

- Career Gap

- Exp Developers

Note: To get job as a java developer and to surive in IT industry as a java developer "SpringBoot + Microservices" knowledge is mandatory.

==========
Course Content
==========

Download Syllabus PDF :
https://ashokit.in/courses/springboot_microservices_online_training

Module-1 : Spring Framework (Core Module)

Module-2 : SpringBoot

Module-3 : Spring Data JPA

Module-4 : Spring Web MVC (C2B)

Module-5 : Spring REST (RESTFul Services) (B 2 B)

Module-6 : Spring Cloud

Module-7 : Microservices

Module-8 : Spring Security

Module-9 : Spring Batch + Spring Scheduling

Module-10 : Real-Time Tools (weekend workshops)

- logging
- junit
- kafka
- redis
- swagger
- postman
- docker
- jenkins

Note: After attending this course, you can keep 3 to 4 years of Exp in Spring

Boot and Microservices

==========
Course Details
==========

Course : Spring Boot with Cloud & Microservices

Course Code : 66-SBMS

Trainer : Mr. Ashok

Daily Class Timings : 6 AM To 7 AM (IST)

Class Frequency : Mon to Sat

Course Duration : 3 Months

=> Every day softcopy notes will be shared

Course Fee : 8,000 INR (live classes + soft copy notes)

For Backup Videos : 2,000 INR (1 year validity)

==========
Benefits of attending this course
==========

1) Daily live classes

2) Topic Wise FAQ's

3) Mock Interviews

4) Resume Building

5) Placement Assistance (based on mock interview rating)

6) Community Support

==========
Core Java
==========

JSE : Java standard edition

=> Core java means JSE

=> In core java, fundamentals of java language will be available

=> Core Java is base to become java developer

-> Using core java we can develop standalone apps (CLI and GUI apps)

ex: calculator, notepad, eclipse ide....

CLI = Commandline interface

GUI = Graphical Interface

Note: Only one user can access stand-alone app at a time.

==========
Adv Java (JEE)
==========

JEE : Java Enterprise Edition (Ex: Servlets, JSP...)

=> JEE is developed based on JSE only

=> Using this JEE we can develop web applications

Ex: gmail, facebook, irctc, naukri...

Note: Multiple users can access web application at a time using internet.

=> To run a web application, web server is required

Ex: Tomcat, Jetty, Netty, Web Logic, WebSphere, Glassfish....

==========
Is it recommended to develop web app using adv. java technologies today ?
==========

Note : To develop one web application using "JDBC + Servlets + JSP" we need to write so many lines of code.

ex: perform CRUD Operations in Database Table (Ex: product_tbl) .

For CRUD Operations : 4 Methods required (create, retrieve, update, delete)

in Each method we should write 6 lines of code.

```
// load driver
// get connection
// prepare stmt
// execute query
// process results
// close connection
```

For one table => 4 methods * 6 lines => 24 lines of code to perform DB operations

=> If we have 2000 tables in our project : 2000 * 24 lines => 48,000 lines of code for DB operations.

Note: In every method we are writing same logic (lot of duplicate code we have to write).

Note: In every JDBC method only query will be different remaining all lines are same.

note: What about business logic and presentation logic in project ?

=> If we use Adv Java concepts to develop a project then we should write so many lines of code which takes more time for development, more time for testing and there is a chance of getting more bugs also.

####### To avoid problems of JEE technologies, Frameworks came into picture. ############

==========
What is Framework
==========

=> Framework is a semi developed software. It provides set of pre-defined classes and interfaces.

=> Frameworks are providing common logics for software application development.

---
common logics
---

1) load driver
2) get conn
3) create stmt
4) process result
5) close conn

6) capture form data
7) validate form data
8) convert form data into java obj

=> If we use frameworks then we can focus only on business logics development.

=> Using frameworks we can do more work in less time (productivity)

=> We have several frameworks in java community

1) Struts
2) Hibernate
3) Spring -------> SpringBoot

Note: The above frameworks developed by using JSE and JEE.

==========
What is Hibernate
==========

=> Hibernate is an ORM Framework.

=> ORM means Object Relational Mapping.

=> Using Hibernate we can develop persistence layer (DAO classes).

=> Persistence layer is used to communicate with databases.

=> Hibernate framework developed on top of JDBC.

Note: Our application will communicate with Hibernate then Hibernate will use JDBC internally to connect with database.

java app -------> Hibernate -------> JDBC ---------> Database

==========
What is Struts
==========

=> Struts framework developed by Apache org

=> Struts framework is used to develop only Web Layer in the application.

Note: To develop a project we need to use both struts and hibernate frameworks.

==========

What is Spring
==========

=> Spring is free & open source java based framework.

=> Using spring we can develop entire application.

=> Spring is called as "application development framework"

=> Spring framework provides common logics required for application development.

=> The author of spring framework is "Rod Johnson".

=> Now spring framework is under license of VMWare company.

=> First version of spring released in the year of 2003 .

=> The current version of spring is 6.x version

########## Note: SpringBoot is an extension for Spring Framework. ############

=> Spring Framework developed in modular fashion

1) Spring Core

2) Spring Context

3) Spring AOP

4) Spring DAO / JDBC

5) Spring ORM

6) Spring Web MVC

7) Spring Cloud

8) Spring Security

9) Spring Batch

10) Spring Data

=> Spring Framework is loosely coupled.

Note: It is not mandatory to use all modules of spring framework in one project.

==========
Spring Core Module
==========

=> It is base module of spring framework eco system.

=> Spring core is providing fundamental concepts of spring framework

1) IOC Container

2) Dependency Injection (DI)

3) Auto Wiring

==========
Spring Context Module

==========

=> It provides configuration support for spring application development.

=> Configurations we can do in 2 ways

1) XML (out dated)

2) Annotations (trending)

==========
Spring AOP
==========

=> AOP stands for Aspect Oriented Programming.

=> AOP is used to seperate cross-cutting logics of our application

ex: security, tx, logging, exception handling..

==========
Spring JDBC / DAO module
==========

=> It is used to simplify Database connectivity in java applications.

Note: In Java JDBC, we should write so many lines of boiler plate code to perform DB operations in the project.

```
// load driver
// get conn
// create stmt
// execute query
// close conn
```

=> Spring JDBC provided predefined classes to execute SQL queries directley.

```
JdbcTemplate.execute(query);
```

==========
Spring ORM
==========

=> ORM means Object Relational Mapping.

=> ORM is used to map Java Objects with Relational Database tables.

=> It is used to simplify Persistence layer development with ORM principles.

=> We can represent DB table data in the form of objects.

=> Spring ORM provided predefined methods to perform curd operations by using objects.

```
hibernateTemplate.save(empObj);

List<Emp> list = hibernateTemplate.getAll();
```

Note: Spring ORM internally using Hibernate and Hibernate internally uses JDBC api.

App --> Spring ORM --> Hibernate --> JDBC API --> Database

==========
Spring WEB MVC
==========

=> It is used to develop web applications (C 2 B).

Note: If we develop a web application using servlets, we need to write lot of boiler plate code like below

1) capture form fields data (req.getParameter('key'))
2) validate form data and
3) convert form fields data into object

==========
Spring Cloud
==========

=> It is used to develop Microservices based applications.

=> It provides several services required for microservices management

1) Eureka Server
2) API Gateway
3) Config Server
4) Feign Client

==========
Spring Security
==========

=> It is used to implement security logics in our applications.

=> By using Spring Security module we can implement Authentication and Authorization.

Authentication => who can login into our application

Authorization => after login, which functionality user can access

==========
Spring Batch
==========

=> Spring Batch module is used to implement bulk operations in our applications.

1) generate bank acc stmt and send to customers emails

2) generate credit card bill stmts and send to customers emails

4) Read data from excel file + process it + store into database

==========
Summary
==========

1) SBMS Overview

2) Core Java vs Adv Java Vs Frameworks

3) What is Framework & Why

4) Java Related Frameworks

5) Hibernate vs Struts vs Spring

6) Spring Introduction

7) Spring Architecture

8) Spring Modules Overview

==========
Spring Core module
==========

=> Base module of Spring Framework

=> Providing fundamental concepts of spring framework

1) IOC
2) DI
3) Auto Wiring

Spring Core module is used to manage our classes in the project.

=> In a project we will have several classes

1) Controller Classes (handle request & response)

2) Service Classes (handle business logic)

3) DAO classes (handle DB ops)

=> In project execution process, One java class method should call another java class method

Ex:
1) Controller class method should call service class method

2) Service class method should call DAO class method

=> We have 2 options to access one java class method in another java class

1) Inheritence (IS-A)

2) Composition (HAS-A)

==========
IS-A Relation
==========

=> Extend the properties from one class to another class.

=> Super class methods we can access directley in sub class.

Ex : Car and Engine classes

Car class ----> drive ( ) method

Engine class ----> start ( ) method

Note: If we want to drive the car then we need to start the Engine first. That means Car class functionality is depending on Engine class functionality.

=> Car class drive ( ) method should call Engine class start( ) method.

```
-----------------------------------------
package in.ashokit;

public class Engine {

	public boolean start() {
		// logic
		System.out.println("Engine started...");
		return true;
	}

}
-------------------------------------
package in.ashokit;

public class Car extends Engine {

	public void drive() {
		boolean status = super.start();

		if (status) {
			System.out.println("Journey started...");
		} else {
			System.out.println("Engine having trouble...");
		}
	}
}
-------------------------------------------------------------
```

=> In the above approach Car is extending properties from Engine class.

=> In future Car can't extend props from other classes bcz java doesn't support for multiple inheritence.

=> With IS-A relationship our classes will become tightly coupled.

=> To overcome problems of IS-A relation we can use HAS-A relation.

```
================
HAS-A relation
================
```

=> Create the object and call the method

=> Inside Car class, create object for Engine class and call eng class start ( ) method.

```
-----------------------------------------------------
public class Car {

	public void drive() {

		Engine eng = new Engine();

		boolean status = eng.start();
		if (status) {
```

```
                        System.out.println("Engine started...");
                        System.out.println("Journey started...");
                } else {
                        System.out.println("Engine having trouble...");
                }
        }
}
```

=> If someone modify Engine class constructor then Car class will fail...

=> with HAS-A relation also our java classes becoming tightly coupled.

Note: Always we need to develop our classes with loosely coupling.

=> To make our classes loosely coupled, we should not extend properties and we should not create object directley.

=> To make our classes loosely coupled we can use Spring Core Module concepts

1) IOC Container
2) Dependency Injection

==========
What is IOC Container
==========

=> IOC stands for Inversion of control.

=> IOC is used to manage & colloborate the classes and objects available in the application.

=> IOC will perform Dependency Injection in our application.

=> Injecting Dependent class object into target class object is called as Dependency Injection.

=> By using IOC and DI we can achieve Loosely coupling among the classes in our application.

Note: We need to provide input for IOC regarding our target classes and dependent classes to perform Dependency Injection.

Note: We can do configuration in 2 ways

1) XML Based (outdated -> springboot will not support)

2) Annotations

=> IOC will take our normal java classes as input and it provides Spring Beans as output.

==========
What is Spring Bean
==========

=> The java class which is managed by IOC is called as Spring bean.

==========
First App development using Spring framework

```
==============================================

## Step-1 : Create maven project using IDE (Eclipse/ STS / IntelliJ)

        - select simple project (standalone)
        - groupId : in.ashokit
        - artifactId : 01-Spring-App


## Step-2 : Configure Spring dependency in project pom.xml file to download
required libraries.

                URL : https://mvnrepository.com/

-------------------------------------
        <dependencies>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>6.2.5</version>
                </dependency>
        </dependencies>
-----------------------------------------

## Step-3 :: Create Required java classes

-----------------------------------------

public class Engine {

        public Engine() {
                System.out.println("Engine Constructor :: Executed");
        }
}

-------------------------------

## Step-4 :: Create Spring Bean Configuration file and configure java classes as
spring beans.

                File Location : src/main/resources/spring-beans.xml

-----------------------------------------------------------

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="e" class="in.ashokit.Engine"/>

</beans>
-----------------------------------------------------------

## Step-5 :: Create Main class to test our application.

-----------------------------------------------------------

public class MyApp {

        public static void main(String[] args) {
```

```
                // Start IOC container by giving xml file as input
                ApplicationContext ctxt = new
ClassPathXmlApplicationContext("spring-beans.xml");

                System.out.println("========== IOC Started ==========");

                // getting bean obj from IOC
                Engine e = ctxt.getBean(Engine.class);

                e.start();

        }
}
```

```
============================
What is Dependency Injection
============================
```

=> The process of injecting one class object into another class object is called as dependency injection.

Note: When we want to call one java class method from another java class method then we need Dependency Injection.

Note: IOC is responsible to perform dependency injection.

=> We can perform Dependency Injection in 3 ways

1) Constructor Injection

2) Setter Injection

3) Field Injection

```
==================================
What is Constructor Injection ?
==================================
```

=> Injecting dependent obj into target obj using target class parameterized constructor is called Constructor injection (C.I).

```
        // constructor injection
        public Car(Engine eng) {
                this.eng = eng;
        }
```

Note: To represent constructor injection we will use below syntax

Syntax : <constructor-arg name="" ref=""/>

Ex :

```
<bean id="c" class="in.ashokit.Car">
        <constructor-arg name="eng" ref="e"/>
</bean>

<bean id="e" class="in.ashokit.Engine" />
```

```
==================================
What is Setter Injection ?
==================================
```

=> Injecting dependent obj into target obj using target class setter method is called as setter injection (S.I).

```
// SETTER METHOD with dependent obj as parameter
public void setEng(Engine eng) {
        this.eng = eng;
}
```

Note: To represent setter injection we will use below syntax

Syntax : <property name="" ref=""/>

Ex:

```
<bean id="c" class="in.ashokit.Car">
        <property name="eng" ref="e"/>
</bean>

<bean id="e" class="in.ashokit.Engine" />
```