**1. Demonstrate Creation of Spring Boot Application**

Using Spring Initializr

Visit: <https://start.spring.io>

Choose:

* + Project: Maven
  + Language: Java
  + Spring Boot Version: Latest stable
  + Dependencies: Spring Web
  + Group: com.example
  + Artifact: demo
* Click Generate, unzip the project, and open in IDE (like IntelliJ or Eclipse).

🔹 Code: Main Application

CODE:

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // Combines @Configuration, @EnableAutoConfiguration, and @ComponentScan

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args); // Bootstraps the app

}

}

**2. Explain the Need and Benefits of Spring Boot**

🔹 Why Spring Boot?

* Rapid Development: Auto-configuration reduces setup time.
* Embedded Servers: No need to deploy WARs to external servers.
* No XML Configuration: Use annotations and application.properties/yaml.
* Reduced Boilerplate: Minimizes repetitive code.
* Production Ready: Metrics, health checks, and configurations are included.

🔹 Benefits:

* Simplified dependency management
* Integrated logging and monitoring
* Quick prototyping
* Easy deployment

**3. Demonstrate Loading Bean from Spring Configuration File**

🔹 Step 1: XML Configuration

CODE:

<!-- beans.xml -->

<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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="student" class="com.example.demo.Student">

<constructor-arg value="John"/>

<property name="age" value="21"/>

</bean>

</beans>

🔹 Step 2: Java Class

CODE:

package com.example.demo;

public class Student {

private String name;

private int age;

public Student(String name) {

this.name = name;

}

public void setAge(int age) {

this.age = age;

}

public void display() {

System.out.println("Name: " + name + ", Age: " + age);

}

}

🔹 Step 3: Load Bean

CODE;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

Student student = (Student) context.getBean("student");

student.display();

}

}

**4. Demonstrate Inclusion of Logging in Spring Boot Application**

🔹 Using application.properties

CODE:

# Logging settings

logging.level.root=INFO

logging.level.com.example.demo=DEBUG

logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n

server.port=8081

🔹 Java Code

CODE:

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class LogController {

Logger logger = LoggerFactory.getLogger(LogController.class);

@GetMapping("/log")

public String logExample() {

logger.trace("TRACE log");

logger.debug("DEBUG log");

logger.info("INFO log");

logger.warn("WARN log");

logger.error("ERROR log");

return "Check the console for logs";

}

}