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

**Spring Initializr**

* URL: <https://start.spring.io>
* Steps:  
  1️⃣ Choose project type (Maven / Gradle)  
  2️⃣ Select language (Java)  
  3️⃣ Choose Spring Boot version  
  4️⃣ Add dependencies (e.g. Spring Web)  
  5️⃣ Generate the project → unzip → open in IDE

**➜ Main Class**

java

CopyEdit

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

✅ @SpringBootApplication:

* Marks as Spring Boot app
* Combines:
  + @Configuration
  + @EnableAutoConfiguration
  + @ComponentScan

✅ SpringApplication.run():

* Launches the app
* Starts embedded server (Tomcat)

**📌 Reference:** [Spring Initializr](https://start.spring.io)

**✅ 2. Explain the need and benefits of Spring Boot**

✅ Why use Spring Boot?  
⭐ Makes Java development easy  
⭐ Avoids tedious configuration  
⭐ Reduces development time  
⭐ No need for XML config  
⭐ Embedded Tomcat server—no deploy to external server  
⭐ Opinionated defaults—quick start  
⭐ Production-ready features: metrics, health checks

✅ Example of avoided boilerplate:

* No need to define DispatcherServlet in XML
* No need to configure ViewResolvers manually

**📌 Reference:** [JournalDev Spring Boot Tutorial](https://www.journaldev.com/7969/spring-boot-tutorial)

**✅ 3. Demonstrate loading bean from spring configuration file**

✅ Use classic Spring (XML-based) configuration

**➜ Spring configuration XML (beans.xml)**

xml

CopyEdit

<?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="student" class="com.example.Student">

<constructor-arg name="id" value="101"/>

<constructor-arg name="name" value="John Doe"/>

</bean>

</beans>

**➜ Java Bean**

java

CopyEdit

package com.example;

public class Student {

private int id;

private String name;

public Student(int id, String name) {

this.id = id;

this.name = name;

}

// getters and toString()

}

**➜ Loading Bean**

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

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

System.out.println(student);

}

}

✅ Key elements:

* <bean>: defines a Spring bean
* id: identifier
* class: fully qualified class name
* <constructor-arg> or <property>: dependency injection
* ApplicationContext: IoC container
* context.getBean(): retrieves bean

✅ Scopes:

* Singleton (default): one shared instance
* Prototype: new instance on each request

**✅ 4. Demonstrate inclusion of logging in Spring Boot Application**

✅ Spring Boot uses **SLF4J** with **Logback** by default.

**➜ Add to application.properties**

properties

server.port=8081

logging.level.root=INFO

logging.level.com.example.demo=DEBUG

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

**➜ Using Logger in Code**

java

CopyEdit

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 DemoController {

private static final Logger logger = LoggerFactory.getLogger(DemoController.class);

@GetMapping("/")

public String home() {

logger.trace("TRACE level log");

logger.debug("DEBUG level log");

logger.info("INFO level log");

logger.warn("WARN level log");

logger.error("ERROR level log");

return "Hello, Spring Boot Logging!";

}

}

✅ Log Levels:

* TRACE
* DEBUG
* INFO
* WARN
* ERROR