University of Sri Jayawardenapura
Faculty of Technology
Department of Information and Communication Technology

ITS 4243 – Micro-services and Cloud Computing

Part 1

1. What is Spring Boot and why is it used?

Spring Boot – Spring boot is an open-source Java framework built on top of Spring Framework that makes the developing of production-ready applications easy.

Why is it used?

- Fast development – reduces boilerplate code and setup time, therefore developers can focus on business logic.
- Microservices architecture – uses microservices architecture which is suitable for developing distributed systems with independent services.
- Embedded servers – Spring boot comes with in-built servers like Tomcat, Jetty or Undertow. So do not need to deploy separate WAR files to external servers, just have to run the app as a stand-alone JAR.
- Production-Ready features – Spring boot includes in-built support for monitoring, metrics, health checks and logging.
- Strong ecosystem and community – Spring boot is a part of the large Spring Ecosystem that includes, Spring Data, Spring Security, Spring Cloud and more. This large community and well-formed documentations makes Spring boot more efficient.

2. The difference between Spring Framework and Spring Boot:

Aspect	Spring Framework	Spring Boot
Purpose	Spring framework provides a wide range of tools to build java applications.	Spring boot simplifies Spring development by reducing boilerplate

		and configuration.
Configuration	This requires manual setup using XML, or annotations.	Uses auto-configuration to set up components based on dependencies.
Server deployment	Need external server deployment like Tomcat	Has embedded servers, so no need to deploy externally.
Complexity	More flexible but need a deep understanding and setup.	Easier to use with opinionated defaults.
Use case	When the users need full control over their application.	When users need to build and deploy applications quickly with minimal steps.

3. Inversion of Control and Dependency Injection:

Inversion of Control (IoC)

Inversion of Control is one of the most important design principles of Spring Boot. This principle transferred the control of object creation and management from the application code to a framework. Instead of the code controlling the flow and creating objects, the framework takes that responsibility. The framework creates and manage the objects, the code is just use them.

This is achieved through IoC Containers, which are instantiates beans, configures them and manages their life cycle.

Types of IoC Containers:

- BeanFactory
  - This is the most basic version of the IoC container. It suitable for lightweight applications where advanced features are not required and it provides basic support for dependency injection and bean life cycle management.
- ApplicationContext
  - This is an extension of BeanFactory with more enterprise features like event propagation, internationalization, and more. This type is mostly used over BeanFactory due to its advance features.

Dependency Injection

Dependency Injection is a pattern used to implement IoC. This allows developer to inject dependencies from the outside rather than creating them inside the class.

There are three types of dependency injections in Spring:

- Constructor injection
- Setter injection
- Field injection

4. The purpose of application.properties / application.yml

- When customizing the behavior of an application, configuration plays a huge role. In traditional way, developers have to hard-code values in the code, but Spring boot provides a more flexible way to configure application settings with application.properties file.
- application.properties or application.yml file manages environment-specific settings like database configuration, server port, logging level and more. This file help to maintain the application's flexibility and efficiency.

5. What is a REST API and list HTTP methods used?

- REST API is an interface which follows the concepts of REST architecture. This enables clients to interact with the servers through HTTP requests and get, and manipulate those data.
- The responses are getting in JSON or XML format.
- As the HTTP methods used in REST API, the followings can be listed.
    - GET – used to retrieve (get) data from the server.
    - POST – used to create a new resource in the server.
    - PUT – this method used to update an existing record completely in the server.
    - PATCH – enable the partial updating of existing resources.
    - DELETE – to remove existing resources, this method can be used.

6. What is Spring Data JPA? What is an Entity and a Repository?

Spring Data JPA is a part of Spring ecosystem, that simplifies database access by using Java Persistence API (JPA). Developers do not need to write boilerplate SQL or JDBC codes, they can interact with relational databases using java objects.

Spring Data JPA provides:

- Automatic implementation of repository interfaces
- Powerful query methods
- Pagination and sorting
- Integration with Spring boot

Entity:

Entity is a java class which denotes a table in the relational database. This specifies with @Entity annotation in the code and the primary key of the table is denoted as @Id in the code.

Repository:

A repository in Spring Data JPA is an interface that offers the operations of accessing and manipulating data in a database. This enables developers to perform CRUD operations with the database without SQL commands.

7.What is the difference of @Component, @Service, @Repository, @Controller, @RestController?

The classes in Spring are marked with these annotations to be automatically detected and wired but they serve a different purpose in the application architecture:

Annotation	Purpose	Layer	Special behavior
@Component	Marks a class as a Spring-managed bean	Generic	No specialization; base for other annotations
@Service	Indicates business logic or service layer	Service	Semantic clarity
@Repository	Marks data access objects (DAOs)	Persistence	Enables exception translation for database errors
@Controller	Handles HTTP requests and returns views	Web (MVC)	Works with view resolvers
@RestController	Combines @Controller and @ResponseBody	Web (REST)	Returns JSON/ XML directly instead of views

8. What is @Autowired? When should we avoid it?

The annotation, called Autowired, is a Spring annotation that is utilized to automatically inject dependencies into an application context class. It also makes the wiring of beans easier because it allows Spring to resolve and deliver the necessary collaborators. Although it is convenient, it works best with the concept of the constructor injection to facilitate immutability and testability.

When should avoid ?

- Avoid field injection
  - @Autowired field injection adds complexity to testing since it does not have any easily available mechanism to inject mocks, breaks encapsulation by exposing the internal fields, and is instead not recommendable and replaced with the more explicit technique of constructor injection which encourages clearer dependencies and immutability.
- Avoid in large applications without explicit configuration
  - When there are several beans of the same type, it can be ambiguous when you use the annotation of Autowired without any qualifiers, so it is better to use the annotation of qualifier to eliminate the conflicts and make the selection of the beans explicit.
- Avoid overuse
  - Too much autowiring will obscure the structure of an application and result in a hard to maintain codebase, often it is better to use configuration classes or manual wiring to define a complex structure.
- Avoid in static contexts
  - Autowired may not be applied to any static field or any method since Spring dependency injection is applied solely to components at the instance level. Keeping the beans in the static context, it is preferable to look at them through ApplicationContext or some other programmatic tools offered by Spring.

9. Explain how Exception Handling works in Spring Boot (@ControllerAdvice).

Exception handling:

Exception handling has a major role in Spring Boot, when it comes to constructing robust web applications. The @ControllerAdvice annotation offers a centralized means of handling exceptions thrown by controllers, at the global level, instead of having error-handling logic scattered over various controllers. Basically, it is a global exception handler whereby you define reusable methods for catching and responding to exceptions consistently.

How it works:

1. Spring's exception handler process starts when an exception occurs in a controller method. (Example: during request processing)
2. Firstly, it checks whether the "@ExceptionHandler" method is available within the same controller.
3. If not, it checks for "@ExceptionHandler" methods in classes defined as "@ControllerAdvice".
4. If it is present, the handler method will generate a custom response for the exception such as an error page, JSON error object or a redirection.

10. What is the role of Maven/ Gradle in a Spring Boot project?

Maven and Gradle can be introduced as two popular build automation tools used in Java projects, which include Spring boot applications. They perform the managing of project dependencies, compilation, testing, packaging and deployment of the project. Spring boot is compatible for both, to provide plugins and starters that makes the configuration easy. Maven uses XML-based configuration which is named as pom.xml, and Gradle uses a Groovy or Kotlin based DSL named as build.gradle.

Key roles in Spring boot project:

- Dependency management
  - Automatically download and manage libraries via pom.xml or build.gradle
- Project build and compilation
  - Spring Boot's Maven or Gradle plugins enable features like executable JARs (via "spring-boot: run" in Maven or "bootRun" in Gradle), which bundle the app with an embedded Tomcat server for easy deployment.
- Testing and Quality assurance
  - Tests can be run during builds to ensure the code quality before packaging. (Example: "mvn test" , "./gradlew test")
- Packaging and deployment
  - Maven and Gradle handle packaging by compiling code, resolving dependencies, and bundling the application into executable JAR or WAR files.
  - They handle deployment across environments and integrate smoothly with CI/CD pipelines for automated delivery.