1)Difference between Gradle and Maven

**Gradle:** Gradle is an open-source tool that helps us to create software with mechanization. This tool is widely used for the creation of different kinds of software due to its high performance. It works on Java and a Groovy-based Domain-Specific Language (DSL) for developing the project structure. Gradle supports the creation of mobile and web applications with testing and deploying on various platforms.

* Gradle is based on developing domain-specific language projects.
* It uses a Groovy-based Domain-specific language (DSL) for creating project structure.
* Developing applications by adding new features to them.
* It avoids compilation.
* This tool is highly customizable as it supports a variety of IDE’s.

**Maven**: Maven is an open-source project management tool that helps us to create different software in the lifecycle used by this tool. This tool focuses on the standardization (i.e.) development of the software in a standard layout within a short duration of time. With this, we can create Java projects but is compatible to be used for other languages too. Maven uses Extensible Markup language (XML) for the structuring of the application.

* Maven is based on developing pure Java language-based software.
* It uses Extensible Markup Language (XML) for creating project structure
* Developing applications in a given time limit.
* It is necessary to compile.
* This tool serves a limited amount of developers and is not that customizable.

2)Difference between yaml and properties file.

**YAML File:** YAML is a configuration language. Languages like Python, Ruby, Java heavily use it for configuring the various properties while developing the applications.

* Human Readable
* It Supports key/val, basically map, List and scalar types
* Its usage is quite prevalent in many languages like Python, Ruby, and Java
* It has Hierarchical Structure
* Spring Framework doesn’t support @PropertySources with .yml files
* If you are using spring profiles, you can have multiple profiles in one single .yml file

**Properties File:** This file extension is used for the configuration application. These are used as the Property Resource Bundles files in technologies like Java, etc.

* Human Readable
* Supports key/val, but doesn’t support values beyond the string
* It is primarily used in java
* It has Non-Hierarchical Structure
* It supports @PropertySources with .properties file
* Each profile need one separate .properties file

3)What Is profiles in spring boot.

* A profile is a set of configuration settings.
* Spring Boot allows to define profile specific property files in the form of

**application-{profile}.properties.**

* To work with Profiles, Spring Framework has provided the following two properties.

1)spring.profiles.default

2)spring.profiles.active

* spring.profiles.active represents active profile.
* spring.profiles.default represents default profile.
* The @Profile annotation indicates that a component is eligible for registration when the specified profile or profiles are active. The default profile is called default.
* Enterprise application development is complex. You have multiple environments
* Dev
* QA
* Stage
* Production
* We have different application configuration in each of the environments.

5)What is entity and different types of mappings.

An entity is a lightweight persistence domain object. An entity represents a table in a relational database, and each entity instance corresponds to a row in that table. The primary programming artifact of an entity is the entity class, although entities can use helper classes.

An entity class must follow these requirements.

* The class must be annotated with the javax.persistence.Entity annotation.
* The class must have a public or protected, no-argument constructor. The class may have other constructors.
* The class must not be declared final. No methods or persistent instance variables must be declared final.
* If an entity instance is passed by value as a detached object, such as through a session bean’s remote business interface, the class must implement the Serializable interface.
* Entities may extend both entity and non-entity classes, and non-entity classes may extend entity classes.
* Persistent instance variables must be declared private, protected, or package-private and can be accessed directly only by the entity class’s methods. Clients must access the entity’s state through accessor or business methods.

6)Loggings in spring boot application.

* Logging in spring boot is very flexible and easy to configure. Spring boot supports various logging providers through some simple configuration.
* In spring we will look various logging options and configurations supported by Spring boot.
* **Default Zero Configuration Logging:** If we do not provide any logging specific configuration, we will still see logs printed in “console”. These are because of default logging support provided in spring boot which uses Logback.
* **Logback Logging:** The default logging is good enough for most usecases. But sometimes in enterprise applications, we need more fine control over logging with other complex requirements. In that case, having a dedicated logging configuration is suitable.

Spring boot by default uses logback, so to customize it’s behavior, all we need to add only logback.xml in classpath and define customization over the file.

* **Log4j2 Logging:**

Step 1: Exclude logback and include log4j2

Spring boot uses logback as default. So if we have to use any other logging framework e.g. log4j2, we must exclude logback from classpath of the application. Also, add spring-boot-starter-log4j2 to classpath.

Step 2: Add log4j2 configuration file

Now, add log4j2 specific configuration file in It can be named as any of the following:

log4j2-spring.xml

log4j2.xml