1. what is the difference between maven and gradle?

|  |  |
| --- | --- |
| **Gradle** | **Maven** |
| It is a build automation system that uses a Groovy-based DSL (domain-specific language ) | It is a software project management  system that is primarily used for  java projects. |
| It does not use an XML file for declaring the project configuration. | It uses an XML file for  declaring the project, its dependencies,  the build order, and its required plugin. |
| It is based on a graph of task dependencies that do the work. | It is based on the phases of the fixed  and linear model. |
| In Gradle, the main goal is to add functionality to the project. | In maven, the main goal is related  to the project phase. |
| It avoids the work by tracking input and output tasks and only runs the tasks that have been changed. Therefore it gives a faster performance. | It does not use the build cache; thus,  its build time is slower than Gradle. |
| Gradle is highly customizable; it provides a wide range of IDE support custom builds. | Maven has a limited number  of parameters and requirements, so  customization is a bit complicated. |
| Gradle avoids the compilation of Java. | The compilation is mandatory in Maven. |

1. what is difference between yaml and properties file?

| YAML(.yml) | .properties |
| --- | --- |
| Spec can be found here | It doesn’t really actually have a spec. The closest thing it has to a spec is actually the javadoc. |
| Human Readable (both do quite well in human readability) | Human Readable |
| Supports key/val, basically map, List and scalar types (int, string etc.) | Supports key/val, but doesn’t support values beyond the string |
| Its usage is quite prevalent in many languages like Python, Ruby, and Java | It is primarily used in java |
| Hierarchical Structure | Non-Hierarchical Structure |
| Spring Framework doesn’t support @PropertySources with .yml files | supports @PropertySources with .properties file |
| If you are using spring profiles, you can have multiple profiles in one single .yml file | Each profile need one separate .properties file |
| While retrieving the values from .yml file we get the value as whatever the respective type (int, string etc.) is in the configuration | While in case of the .properties files we get strings regardless of what the actual value type is in the configuration |

## 3. what is profiles in spring boot?

## Spring Boot profiles

The development process of an application has different stages; the typical ones are development, testing, and production. Spring Boot profiles group parts of the application configuration and make it be available only in certain environments.

A profile is a set of configuration settings. Spring Boot allows to define profile specific property files in the form of application-{profile}.properties. It automatically loads the properties in an application.properties file for all profiles, and the ones in profile-specific property files only for the specified profile. The keys in the profile-specific property override the ones in the master property file.

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; all the beans that do not have a profile set belong to this profile.

There are plenty of ways of defining active profiles in Spring Boot, including command line arguments, Maven settings, JVM system parameters, environment variables, spring.profiles.active property, and SpringApplication methods.

## Spring Boot profiles example

In the following application, we have three profiles (local, dev, prod) and two profile-specific property files. We use the spring.profiles.active to set active profiles and SpringApplicationBuilder's profiles method to add new active profiles.

pom.xml

src

├── main

│   ├── java

│   │   └── com

│   │   └── zetcode

│   │   └── Application.java

│   └── resources

│   ├── application-dev.properties

│   ├── application-prod.properties

│   └── application.properties

└── test

└── java

1. what is entity and different types of mappings?

An entity is a lightweight persistence domain object. Typically, 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.

1. logging in spring boot application?

[10:00 pm, 09/11/2021] +91 95813 11214: 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