Assignment

1. what is the difference between maven and gradle ?

A

Maven	Gradle
▫ Maven is based on developing pure java language-based software	▫ Gradle is based on developing domain-specific language projects.
▫ It is necessary to compile	▫ It avoids the java Compilation.
▫ This tool is a limited amount of developers and is not customizable	▫ This tool is highly customizable as it supports a variety of IDE's
▫ It does not create local temporary file during software creation hence uses large time	▫ It performs better than mavens as it optimized for tracking only current running task
▫ It supports software development in Scala,c# and ruby	▫ It supports software development in Java,c,c++and Groovy

2. what is difference between yaml and properties file ?

A

Yaml(.yml)	Properties
▫ It contains key and value pairs	▫ Key and values separted by equal or color
▫ Human readable format	▫ Easy to read by humans
▫ Supports Integer, Strings, Maps, Lists	▫ Supports primitive types like strings and numbers
▫ Supports hierarchical structure	▫ Supports flat and non hierarchical structure
▫ Spring Framework doesn't support @PropertySources with .yml files	▫ supports @PropertySources with .properties file

3.what is profiles in spring boot ?

A A profile is a set of configuration settings.

- Profiles are a core feature of the framework — **allowing us to map our beans to different profiles** — for example, *dev*, *test*, and *prod*.
- We can then activate different profiles in different environments to bootstrap only the beans we need.
- Spring Profiles are not limited to deployment environments
- The Spring Boot supports @Profie annotations on the *configurations* and as well as *Bean* Methods. In addition, **Spring Boot supports environment specific properties files**. Because of these properties files properties management becomes really easy.

  For Example , take a look at three different properties file:

  Application.properties

```
spring.profiles.active=dev

spring.datasource.driver-class-name= com.mysql.jdbc.Driver
spring.datasource.username= songs_service_user
```

application-dev.properties

```
spring.datasource.url= jdbc:mysql://dev_db_host:3306/songsDB
spring.datasource.password= <password>
```

application-prod.properties

```
spring.datasource.url= jdbc:mysql://prod_host:3306/songsDB
spring.datasource.password= <password>
```

- These are simple *datasource* related properties. The default properties has common things like *driver* and database *username*. Moreover, Spring Boot reads the default properties file in all profiles. The other two files contains environment specific properties, such as database *url* and database *password*.

- The default properties file has an additional entry spring.profiles.active=dev. If you don't set active profile anywhere else, Spring Boot will use this

5. what is entity and different types of mappings ?

A 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.

Mapping classes are generated during compilation and no runtime processing or reflection is used. ... Mapping classes use simple method invocation, which makes them really easy to debug.

Types of mapping:

1.one-one Association

2.one-many Association

3.Many to one Association

4.Many to Many Association

6. logging in spring boot application ?

A

- 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

- You can enable debug logging by **specifying --debug** when starting the application from the command-line. Spring Boot provides also a nice starting point for logback to configure some defaults, coloring etc. the base. xml file which you can simply include in your logback.

- Spring Boot uses Apache Commons logging for all internal logging. Spring Boot's default configurations provides a support for the use of Java Util Logging, Log4j2, and Logback. Using these, we can configure the console logging as well as file logging.

- If you are using Spring Boot Starters, Logback will provide a good support for logging. Besides, Logback also provides a use of good support for Common Logging, Util Logging, Log4J, and SLF4J.