Classpath

This statement outlines the default locations where Spring Boot will look for `application.properties` and `application.yaml` files during application startup:

1. **From the classpath**: Spring Boot will search for these files in the classpath, which typically includes all the directories containing your application's compiled code and resources.

2. **The classpath root**: If the files are not found directly in the classpath, Spring Boot will look for them at the root of the classpath.

3. **The classpath /config package**: If the files are not found at the root, Spring Boot will search for them in a `/config` package within the classpath.

4. **From the current directory**: If the files are not found in the classpath, Spring Boot will look for them in the current directory where the application is executed.

5. **The current directory**: If the files are not found in the current directory, Spring Boot will search for them at the root of the current directory.

6. **The config/ subdirectory in the current directory**: If the files are not found at the root of the current directory, Spring Boot will look for them in a `config/` subdirectory within the current directory.

7. **Immediate child directories of the config/ subdirectory**: If the files are not found in the `config/` subdirectory, Spring Boot will search for them in any immediate child directories of the `config/` subdirectory.

By default, Spring Boot provides flexibility in locating these configuration files, allowing you to organize them based on your project structure and preferences. This approach simplifies configuration management and promotes convention over configuration.

In the context of a Spring Boot application:

1. **Classpath Root**: This refers to the root directory of the classpath, which is where your compiled Java classes (`*.class` files) and resources are located. In a typical Maven or Gradle project structure, this would be the `target/classes` directory in the `target` directory after the project is built. All resources within this directory, including `application.properties` or `application.yaml`, are accessible from the classpath.

2. **Classpath /config Package**: This refers to a specific subdirectory named `config` within the classpath. For example, if you have a package named `config` within your `src/main/resources` directory, its contents will be included in the classpath. You can place additional configuration files within this package, and Spring Boot will automatically load them.

3. **Current Directory**: This refers to the directory from which the application is being executed. When you run a Spring Boot application using `java -jar`, the current directory is the directory in which you run the `java -jar` command. If you run the application using an IDE, the current directory may vary depending on the IDE settings.

In summary:

- The classpath root contains resources and classes that are directly accessible from the classpath, including the `target/classes` directory after compilation.
- The classpath `/config` package refers to a specific subdirectory within the classpath where additional configuration files can be placed.
- The current directory is the directory from which the application is executed, and it may vary depending on how the application is launched.

Sure, let's consider a typical project structure for a Spring Boot application and how the `application.properties` or `application.yaml` files can be placed within it:

```
my-spring-boot-app/
 ├─── src/
|   ├─── main/
|   |   ├─── java/
|   |   |   └─── com/
|   |   |       └─── example/
|   |   |           └─── myapp/
|   |   |               └─── MySpringBootApplication.java
|   |   └─── resources/
|   |       ├─── application.properties
|   |       └─── application.yaml
|   └─── test/
|       ├─── java/
|       |   └─── com/
|       |       └─── example/
|       |           └─── myapp/
|       |               └─── MySpringBootApplicationTests.java
|       └─── resources/
 └─── target/
```

In this structure:

- `src/main/java`: This directory contains the Java source code of the application, typically organized by package names.
- `src/main/resources`: This directory contains non-Java resources used by the application, such as configuration files, static files, and templates.
  - `application.properties` or `application.yaml`: These files are placed directly in the `resources` directory. Spring Boot will automatically find and load them during application startup.
- `src/test`: This directory contains test code for the application.
- `target`: This directory contains compiled code and other build artifacts generated by the build process.

With this structure, Spring Boot will automatically detect the `application.properties` or `application.yaml` files located in the `resources` directory and use them to configure the application. Additionally, you can also place these configuration files in subdirectories like `config/` within the `resources` directory if you prefer.