Properties

To include JVM properties in your Spring Boot application, you have a few options depending on how you want to set and manage these properties:

1. **Command-Line Arguments**: You can pass JVM properties directly as command-line arguments when starting your Spring Boot application using the `-D` flag. For example:

   ```bash
   java -jar -Dspring.profiles.active=dev my-application.jar
   ```

   In this example, `-Dspring.profiles.active=dev` sets the `spring.profiles.active` property to `dev`.

2. **Application.properties or application.yml**: You can also define JVM properties in your `application.properties` or `application.yml` file by prefixing them with `spring.boot.`. For example:

   ```properties
   spring.boot.profiles.active=dev
   ```

   Or in YAML format:

   ```yaml
   spring:
     boot:
       profiles:
         active: dev
   ```

   These properties will be picked up and applied by Spring Boot during application startup.

3. **Programmatically Setting Properties**: If you need to set properties dynamically or conditionally during application startup, you can use the `SpringApplication.setDefaultProperties()` method or the `SpringApplication.setAdditionalProfiles()` method in your main application class. For example:

   ```java
   import org.springframework.boot.SpringApplication;
   import org.springframework.boot.autoconfigure.SpringBootApplication;

   @SpringBootApplication
   ```

```
   public class MyApplication {

       public static void main(String[] args) {
           SpringApplication application = new
SpringApplication(MyApplication.class);

application.setDefaultProperties(Collections.singletonMap("spring.profiles.acti
ve", "dev"));
           application.run(args);
       }
   }
```

This approach allows you to set properties based on environment variables, system properties, or any other dynamic logic you require.

Choose the method that best fits your use case and preferences for managing JVM properties in your Spring Boot application.

Different Properties:

We can define the core properties:

logging.level.org.example=info
We are setting up loggging level for package.

We can also put where to keep the log

logging.file.name=my-crazy-stuff.log
logging.file.path=c:/myapps/demo

We can define the web properties:

server.port=8080
server.servlet.context-path=/my-silly-app (make the url as http://localhost:8080/my-silly-app/fortune where fortune is the endpoint)
servlet.session.timeout=15m // default session timeout given in minutes

We can define the actuator properties:

Management.endpoints.web.base-path=/actuator/myapp
We can access the endpoints at http://localhost:8080/actuator/myapp/info

We can also define data properties for the datasource

We can also define security properties