# CommandLineRunner in Spring Boot

what is CommandLineRunner in java sprinbgoot,
Give me sample, use case, and benefit to implements  CommandLineRunner in a class

In **Java Spring Boot**, the `CommandLineRunner` interface is used to execute a block of code after the Spring Boot application has started. It is typically used to run tasks like initializing resources, setting up default data, or executing any logic that needs to happen after the application context has been initialized but before the application starts serving requests.

---

## 1. Overview

- **Interface:** `CommandLineRunner`
- **Method to Implement:** `void run(String... args) throws Exception`
- The `run` method is invoked automatically with the command-line arguments passed to the application.

---

## 2. Sample Implementation

Here is an example where a `CommandLineRunner` is used to print a startup message and initialize some mock data.

```
java


 import org.springframework.boot.CommandLineRunner;
 import org.springframework.stereotype.Component;

 @Component
 public class StartupRunner implements CommandLineRunner {

     @Override
     public void run(String... args) throws Exception {
         System.out.println("Application started with command-line arguments: " + String.join
 (", ", args));

         // Mock data initialization
         initializeData();
     }

     private void initializeData() {
         System.out.println("Initializing default data...");
         // Your logic for initializing data
     }
 }
```

---

## 3. Use Cases

Here are common scenarios where `CommandLineRunner` is beneficial:

1. **Database Initialization:**
   - Populate a database with default or mock data.
2. **Data Migration:**
   - Run scripts for migrating data from one format or database to another.
3. **Configuration Checks:**
   - Verify that required configurations or environment variables are set correctly.
4. **Logging Information:**
   - Print startup information like environment details or application-specific metrics.
5. **Service Registration:**
   - Register the application or its services with an external system (e.g., a discovery service).

---

## 4. Benefits of Using CommandLineRunner

1. **Post-Startup Execution:**
   - Ensures that the logic is executed after the application context is fully initialized.
2. **Easy Setup:**
   - Simple to implement and requires no complex configuration.
3. **Access to Spring Context:**
   - You have access to all Spring-managed beans, allowing seamless integration.
4. **Flexibility:**
   - Can handle any type of initialization or one-time startup logic.

---

## 5. Real-World Example

Here's a more practical use case where a `CommandLineRunner` inserts some default records into a database at application startup:

```
java


 import org.springframework.boot.CommandLineRunner;
 import org.springframework.stereotype.Component;
 import org.springframework.beans.factory.annotation.Autowired;

 @Component
 public class DataInitializer implements CommandLineRunner {

     @Autowired
     private UserRepository userRepository;

     @Override
     public void run(String... args) throws Exception {
         if (userRepository.count() == 0) {
             System.out.println("No users found in the database. Initializing default user
 s...");

             User user1 = new User("john.doe", "John Doe", "password123");
             User user2 = new User("jane.doe", "Jane Doe", "password456");

             userRepository.save(user1);
             userRepository.save(user2);

             System.out.println("Default users initialized.");
         } else {
             System.out.println("Users already exist in the database. Skipping initializatio
 n.");
         }
```

```
    }
}
```

---

## 6. Comparison with ApplicationRunner

If you need access to parsed arguments (instead of just a `String[]`), consider using `ApplicationRunner` instead of `CommandLineRunner`. It works similarly but provides an `ApplicationArguments` object.

Example:

```
java


import org.springframework.boot.ApplicationRunner;
import org.springframework.boot.ApplicationArguments;
import org.springframework.stereotype.Component;

@Component
public class ApplicationStartupRunner implements ApplicationRunner {

    @Override
    public void run(ApplicationArguments args) throws Exception {
        System.out.println("Application started with arguments: " + args.getOptionNames());
    }
}
```

---

## Conclusion

The `CommandLineRunner` is a convenient and straightforward way to execute custom logic after the Spring Boot application starts. Its simplicity, combined with integration into the Spring context, makes it an excellent choice for various post-startup tasks.