Step 1: Configure Development and Production Profiles in Spring Boot

AWS Lightsail Deployment Course

2026-04-08

Table of contents

1 Configure Development and Production Profiles in Spring Boot

Preparing your Spring Boot application for seamless deployment to AWS Lightsail

Welcome to step 1 of deploying Spring Boot apps to AWS Lightsail. Today we'll configure separate application profiles for development and production environments.

---

2 Why Spring Boot Profiles Matter

Development Environment

- Local database connections
- Debug logging enabled
- Development-friendly settings

Production Environment

- Cloud database connections
- Optimized performance settings
- Security-focused configuration

Profiles allow you to segregate application configuration and make it available only in certain environments. This prevents hardcoding production credentials and enables local testing with development databases.

---

3 Base Configuration Setup

Create `application.properties` with common settings:

```
# Application name and basic settings
spring.application.name=my-spring-app
server.servlet.context-path=/

# Default active profile
spring.profiles.active=dev
```

```
# JPA/Hibernate common settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
```

The base configuration contains settings that apply to all environments. Notice we set the default profile to 'dev' for local development convenience.

---

4 Development Profile Configuration

Create `application-dev.properties`:

```
# Local MySQL database configuration
spring.datasource.url=jdbc:mysql://localhost:3306/myapp_dev
spring.datasource.username=dev_user
spring.datasource.password=dev_password

# Development-specific settings
spring.jpa.show-sql=true
logging.level.org.springframework.web=DEBUG
server.port=8080
```

The development profile uses local database connections and enables verbose logging to help with debugging during development.

---

5 Production Profile Configuration

Create `application-prod.properties`:

```
# Production database with environment variables
spring.datasource.url=${DB_URL:jdbc:mysql://localhost:3306/myapp_prod}
spring.datasource.username=${DB_USERNAME:prod_user}
spring.datasource.password=${DB_PASSWORD:secure_password}

# Production optimizations
```

```
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=validate
server.compression.enabled=true
```

Production profile uses environment variables for database configuration, enabling secure credential management without hardcoding sensitive information.

---

6 Environment Variables in Production

Syntax: `${VARIABLE_NAME:default_value}`

Benefits:

- Secure credential management
- Environment-specific overrides
- No hardcoded production secrets
- Easy configuration changes

```
spring.datasource.url=
  ${DB_URL:default}

spring.datasource.username=
  ${DB_USERNAME:user}

spring.datasource.password=
  ${DB_PASSWORD:pass}
```

The ${} syntax allows overriding configuration values with environment variables while providing fallback defaults. This is crucial for cloud deployments.

---

7 Build Configuration Updates

7.1 Maven Dependencies

```xml
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Ensure your build configuration includes the necessary dependencies for MySQL connectivity and JPA data access.

---

8 Testing Your Profiles

8.1 Development Profile

```bash
# Run with development profile (default)
mvn spring-boot:run

# Or explicitly specify
mvn spring-boot:run -Dspring-boot.run.profiles=dev
```

8.2 Production Profile Testing

```bash
# Set environment variables
export DB_URL="jdbc:mysql://localhost:3306/myapp_test"
export DB_USERNAME="test_user"

# Run with production profile
mvn spring-boot:run -Dspring-boot.run.profiles=prod
```

Testing both profiles locally ensures your configuration works correctly before deployment. The production profile test helps verify environment variable substitution works.

9 Building for Production

Create deployment-ready JAR:

```
# Clean and build the project
mvn clean package

# Verify the JAR was created
ls -la target/*.jar
```

Output:

```
-rw-r--r-- 1 user user 45M Dec  1 10:30 my-spring-app-1.0.0.jar
```

The executable JAR contains your application and all dependencies, making it perfect for cloud deployment. This single file can be transferred to your Lightsail instance.

---

10 Configuration Verification Checklist

- Created `application.properties`, `application-dev.properties`, and `application-prod.properties`
- Application starts successfully with development profile
- Executable JAR builds without errors
- Database dependencies configured in build file
- Environment variable syntax working in production profile

Before proceeding to the next step, ensure all these checkpoints are completed. This foundation is crucial for successful deployment.

---

11 Key Benefits Achieved

Development Benefits

- Local development database
- Debug logging enabled
- Fast development cycle
- Easy configuration changes

Production Benefits

- Secure credential management
- Optimized performance settings
- Environment-specific configurations
- Cloud-ready deployment package

This profile configuration provides the foundation for a professional deployment workflow, separating concerns between development and production environments.

---

12 What's Next?

Coming Up in Step 2:

- Create AWS Lightsail instance
- Configure Linux environment
- Set up MySQL database
- Prepare for application deployment

Your Spring Boot application is now **production-ready** with proper profile configuration!

With profiles configured, your application can seamlessly transition from local development to AWS Lightsail. Next, we'll create the cloud infrastructure to host your application.

---

13 Summary

- **Spring Boot profiles** enable environment-specific configurations
- **Environment variables** provide secure credential management
- **Executable JARs** simplify deployment processes
- **Proper configuration** is essential for cloud deployments

Your application is now ready for AWS Lightsail deployment!

You've successfully configured a Spring Boot application with separate development and production profiles, created a deployment-ready JAR, and established the foundation for secure cloud deployment.