3. Профили

В этом уроке мы поговорим о профилях.

Представим ситуацию, что разработчик решил задачу и написал код. Стандартные пути для выкатки этой задачи на продуктовый сервер:

1. Проверить работу локально (например, запустить приложение на своем компьютере и проверить работу endpoints).
2. Запустить тесты.
3. Выкатить задачу на какой-то стенд, например стенд с названием dev, куда разработчики заливают все задачи для первоначального тестирования и отладки.
4. После проверки на dev - сделать деплой на prod.

Во всех случаях (локальное окружение, прогон тестов, dev и prod стенды) нужны различные настройки: разные хосты и креды для подключения к БД, разные наименования хостов для каких-то внутренних интеграций (например, url для запросов в какой-то микросервис могут выглядеть по разному на dev и prod стенде). Чтобы не указывать каждый раз это вручную, такие настройки описываются где-то один раз для разных стендов (это и называется профили) и далее Spring можно запустить с указанием конкретного профиля (стенда), причем все настройки применятся автоматически.

Для того, чтобы в Spring указать конкретный профиль, нужно либо указать аннотацию @Profile, либо сделать отдельный файл application.properties/application.yaml. Обычно это делают, указывая название профиля после application через дефис: application-local.properties, application-dev.properties, application-prod.properties и т.д.

4. Пример работы с профилями в Spring Boot

Давайте разберем, как работать с профилями на примере. За основу возьмем стандартный файл с настройками в Spring application.properties

1. **Создаем разные файлы настроек с разными профилями со следующим содержимым. Данные файлы обычно в той же директории, где находится стандартный application.properties:**

**application.properties:** (файл настроек без профиля, по умолчанию)

spring.application.name=MyApp  
server.port=8080

**application-local.properties:** (файл настроек без профиля для локального запуска)

spring.application.name=MyAppLocal  
server.port=8080

**application-test.properties:** (файл настроек без профиля для запуска тестов)

spring.application.name=MyAppTest  
server.port=8080

**application-dev.properties:** (dev стенд)

spring.datasource.url=jdbc:h2:mem:testdb  
logging.level.root=DEBUG

**application-prod.properties:** (prod сервер)

spring.datasource.url=jdbc:postgresql://prod-server/mydb  
logging.level.root=INFO

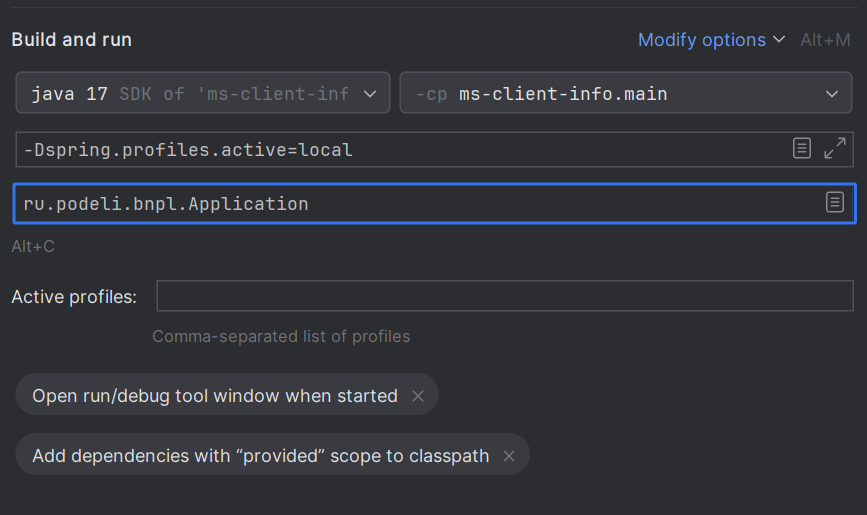
1. **Используем аннотацию @Profile в коде (опционально):**

@Service  
@Profile("dev")  
**public** **class** DevOnlyService {  
    *// Этот бин будет создан только если активен профиль dev*  
}

1. **Запускаем приложение с нужным профилем. Пример из практики: запуск приложения на стенде dev через kubernetes:**

* Через командную строку:
* java -jar myapp.jar --spring.profiles.active=dev
* Или через переменную окружения:
* export SPRING\_PROFILES\_ACTIVE=dev  
  ./gradlew bootRun

Точно также можно запускать приложение локально. Я, как разработчик, часто тестирую свои приложения с локальным профилем через создание файла application.yml (или application.properties) и указание профиля прямо из Idea:



1. **Spring Boot сам подхватывает нужные настройки:**

* Если активен профиль dev, будут применены настройки из application-dev.properties.
* Если активен профиль prod, будут применены настройки из application-prod.properties.

Итого, профили позволяют легко переключаться между разными конфигурациями без изменения основного кода приложения. Это очень сильно упрощает разработку - от локальной проверки, до запуска приложения на проде.