Профили

- Среды разработки
- Активация профайла
- Формирование профайла

Приложения, разрабатываемые программистами, запускаются в разных окружениях (или, как говорят, средах), которые требуют разных настроек для своей работы. Например, локально, на своей машине разработчик может использовать одну базу данных, а в продакшене другую. То же самое касается настроек логирования, правил безопасности и всех остальных аспектов работы приложения.

Возможность иметь свои наборы параметров под разные окружения в Spring Boot реализуется с помощью механизмов профайлов (Profiles). С помощью профайлов мы можем включать специфические бины или параметры конфигурации без необходимости изменения исходного кода.

Среды разработки

Как минимум выделяют две среды разработки:

- Продакшен (Production). Место, где запускается проект для его реальной работы. Как правило, это происходит в облаках Amazon, Yandex или Google.
- Девелопмент (Development). Почти всегда локальная машина разработчика. Обычно среда одна, даже если разработчиков больше одного человека.

Кроме указанных, часто, добавляют среду Staging. Это предпродакшен среда, где имитируется окружение, максимально приближенное к продакшену. Эту среду используют для тестирования приложения перед тем как выполнить его деплой на продакшен.

Итого, в типовых ситуациях, мы можем говорить о трех средах, для которых в Spring Boot будут определены три профайла:

- production
- development
- staging

Активация профайла

Активация профайла выполняется двумя основными способами.

- С помощью переменной окружения `SPRING_PROFILES_ACTIVE`

```
export SPRING_PROFILES_ACTIVE=development
java -jar myproject.jar
```

- С помощью аргумента во время запуска приложения

```
java -jar myproject.jar --spring.profiles.active=development
// Или Gradle
./gradlew run --args='--spring.profiles.active=development'
```

Формирование профайла

Аннотация `@Profile` указанная для `@Component` или `@Configuration` указывает на то, что эти классы должны использоваться только в определенном окружении:

```
@Configuration
@Profile("development")
public class DevelopmentConfig {
    // Configuration for the development profile
}
```

Таким образом можно реализовать подмену как настроек, так и реализацию каких-то классов.

Конфигурационные параметры предпочтительно подменять с помощью файлов, специфичных для своих профилей: *application-{profile}.yml* расположенных рядом с оригинальным *application.yml*. Например:

- *application-production.yml*
- *application-staging.yml*

Наличие файлов специфичных для разных окружений не отменяет наличие общего *application.yml*, в котором полезно оставлять общие настройки, чтобы не дублировать каждом профайле.

Далее →