Конфигурация

- Конфигурационный файл
- Стандартные опции
- Нестандартные опции
- Использование опций
  - Аннотация @Value
  - Объект с конфигурацией

Spring Boot работает сразу после установки без какого-либо конфигурирования — для старта этого достаточно. Но со временем появляется все больше задач, в которых нужно менять поведение фреймворка или дополнительных библиотек.

В этом уроке мы разберем основные способы конфигурации фреймворка и ключевые опции, о которых полезно знать.

В конфигурацию фреймворка может входить:

- Включение цветного вывода логов
- Настройки сервера — порта или адреса, на котором сервер должен работать
- Уровни логирования для разных компонентов фреймворка
- Настройки взаимодействия с базой данных
- Конфигурация библиотек

Конфигурационный файл

Основная конфигурация Spring Boot хранится в файле *application.yml*, расположенном в директории *src/main/resources*. По умолчанию этого файла нет. Его нужно создать в тот момент, когда что-то требует конфигурации:

```
---
logging:
  level:
    root: WARN

spring:
  jpa:
```

```
    generate-ddl: true
    show-sql: true
  output:
    ansi:
      enabled: always

server:
  address: 0.0.0.0
```

Сейчас это основной, но не единственный способ настройки. До сих пор во многих статьях и документации можно встретить подход, основанный на файлах *properties*. Сам файл называется *application.properties*. Его располагают в директории *src/main/resources*:

```
logging.level.root=WARN
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.output.ansi.enabled=always
server.address=0.0.0.0
```

Примеры выше содержат одинаковый набор опций. В последнем примере все уровни схлопнуты в одну строчку, где каждый уровень разделен точкой. Эту связь важно понимать, потому что в курсе вы будете видеть примеры в одном формате, а в вашем проекте — использовать другой формат.

Стандартные опции

Какие возможности по конфигурированию Spring Boot у нас есть? Обычно об этом узнают из статей или документации. Опций очень много, поэтому для удобства они сгруппированы по темам. Обратите внимание, что описание свойств сделано в формате *properties* — так удобнее.

Вот некоторые опции, которые стоит установить сразу:

Свойство	Описание
spring.output.ansi.enabled	Включает цветной вывод логов, что упрощает их анализ
spring.jpa.show-sql	Выводит в лог SQL-запросы

Еще полезно настроить уровни логирования и некоторые другие опции. В изучении Spring Boot мы советуем ориентироваться на наши практические задания — в них выставлены важные конфигурационные параметры, которые упрощают работу с фреймворком. Со временем вы сами попробуете многие опции, в том числе связанные с уровнем логирования разных пакетов.

Нестандартные опции

Конфигурация нужна не только для самого Spring Boot — ее можно использовать и для собственных нужд. Этим активно пользуются сторонние библиотеки. Обычно в их документации рекомендуется размещать опции в файлах по названию библиотек: *src/main/resources/name.properties*.

Spring Boot автоматически подхватывает эти файлы. Все опции можно хранить в *application.yml*, но сначала их нужно правильно перенести. Обсудим, как работает перенос. Если библиотека предлагает хранить опции в файле *name.properties*, то в файле *application.yml* эти опции нужно расположить внутри ключа *spring*.

Для примера рассмотрим опции библиотеки *instancio*, которая используется в наших тестах:

```
spring:
  instancio:
    bean:
      validation:
        enabled: true
```

Еще можно добавлять свои опции. Обычно они хранятся на верхнем уровне:

```
myoption:
  key: value
  another:
    inner:
      key: true
```

Структура самих опций может быть любой. Тут действуем по задаче.

Использование опций

Любые опции можно использовать в коде напрямую, в том числе:

- Опции, которые вы добавили сами
- Опции, определенные Spring Boot
- Опции, определенные сторонними библиотеками

Во всех трех случаях подход не меняется. Ниже мы рассмотрим два способа их использования.

Аннотация `@Value`

Самый простой способ использовать опции — это аннотация `@Value`. Она подставляет значение в поле нужного класса — например, контроллера:

```
// Здесь нужно указать полный путь до опции
@Value("${spring.instancio.bean.validation.enabled}")
private Boolean instancioEnabled;

@Value("${myoption.key}")
private String myoptionValue;
```

Объект с конфигурацией

Для примера представим, что опций много, и они нужны в разных местах. В таком случае подход с `@Value` неудобен — возникнет дублирование. Нам придется постоянно указывать полные пути в разных местах приложения и исправлять их вручную, если они поменяются.

Чтобы было удобнее, можно создать класс, объект которого Spring Boot автоматически заполнит данными из конфигурационного файла.

Предположим, что мы будем работать с пользователем по умолчанию. Для этого создадим конфигурацию, в которую поместим имя и возраст пользователя:

```
user:
  full-name: Mask Ilonov
  age: 25
```

Чтобы получить доступ к этим опциям в коде, создадим класс, помеченный несколькими аннотациями:

```
// src/main/java/io/hexlet/spring/component/DefaultUserProperties.java
package io.hexlet.spring.component;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.stereotype.Component;

import lombok.Getter;
import lombok.Setter;

@Component
@ConfigurationProperties(prefix = "user")
@Setter
@Getter
// Имя класса может быть любым
public class DefaultUserProperties {
    private String fullName; // Ключ full-name автоматически превращается в fullName
    private Integer age;
}
```

Здесь мы видим три вида аннотаций:

- Аннотации, которые генерируют геттеры и сеттеры, делая работу удобнее
- Аннотация `@Component`, которая указывает, что Spring Boot должен воспринимать этот класс как часть системы
- Аннотация `@ConfigurationProperties`, которая указывает, что этот класс отвечает за работу со свойствами под ключом *user*

Чтобы использовать этот класс, в любом другом классе создадим свойство, куда Spring Boot передает объект класса `DefaultUserProperties`, заполненный опциями. Дальше его можно использовать в обычном порядке:

```
@Autowired // Аннотация нужна для автоподстановки объекта
private DefaultUserProperties userInfo;

// Где-то в методах используем
userInfo.getFullName();
userInfo.getAge();
```

Далее →