Вебинар

Старт в Spring Boot: создание простых приложений

Шибков Константин

4 марта 2023

образовательная платформа Skillbox

Спикер

Шибков Константин

- Java разработчик
- куратор курса «Java разработчик»
- пишу микросервисы на Spring Boot
- люблю иммутабельные классы
- пишу в канале ТГ @three_monitors

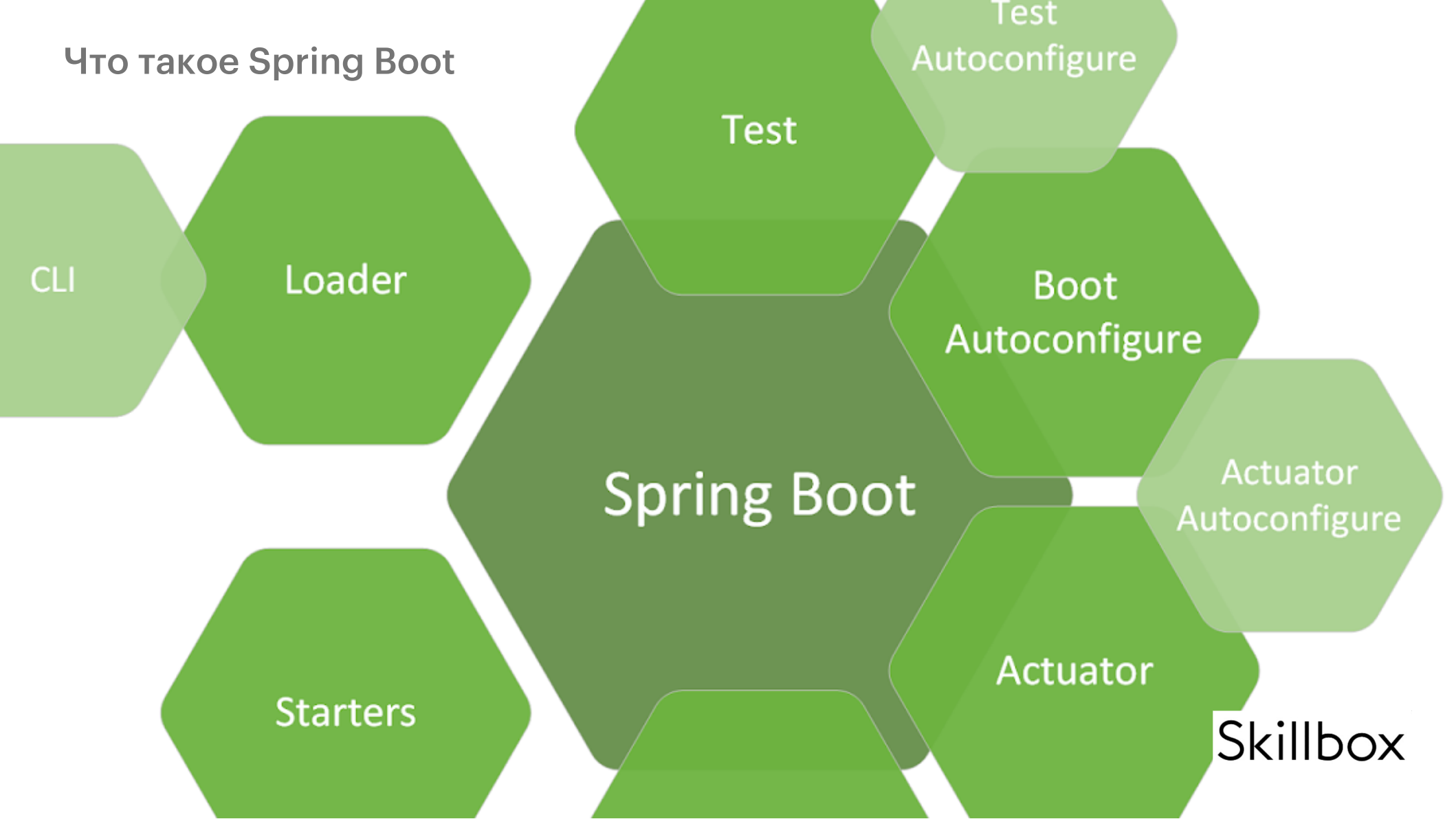
Skillbox

Темы вебинара

- Что такое Spring Boot
- Основные концепции
- Создание проекта Spring Boot
- Обзор настроек и конфигураций
- Dependency Injection / Spring IoC Container
- Аннотации Spring Boot
- Создание простого приложения

Skillbox

Что такое Spring Boot



Skillbox

Упрощенная конфигурация

Большинство компонентов уже имеет стандартные настройки для компонентов. Подключаете нужную зависимость и она начинает работать.

При необходимости можно переопределить стандартные настройки своими.

Это позволяет избежать написания большого количества шаблонного кода, аннотаций и XML-конфигурации.

Что такое Spring Boot

Одна зависимость - готовое приложение

Spring Boot приложение уже имеет встроенный HTTP сервер и набор удобных аннотаций. Снова не требует лишних действий от разработчика.

Быстрое развертывание

Spring Boot поддерживает множество платформ для развертывания приложений, включая облачные сервисы, что позволяет быстро развернуть приложение на любой платформе.

Очень легко интегрировать приложение Spring Boot с его экосистемой, такой как Spring JDBC, Spring ORM, Spring Data, Spring Security и т.д.


МАГИЯ!

Что нужно знать для Spring Boot?

Основы Java, такие как классы, объекты, переменные, методы, коллекции, наследование, интерфейсы и т.д.

Maven или Gradle: Spring Boot использует системы управления зависимостями Maven или Gradle. Минимально надо знать как подключать новые зависимости.

SQL: для работы с базой данных вам нужно знать SQL, язык запросов к базам данных.


НАЧНЕМ!

Создаем новое приложение

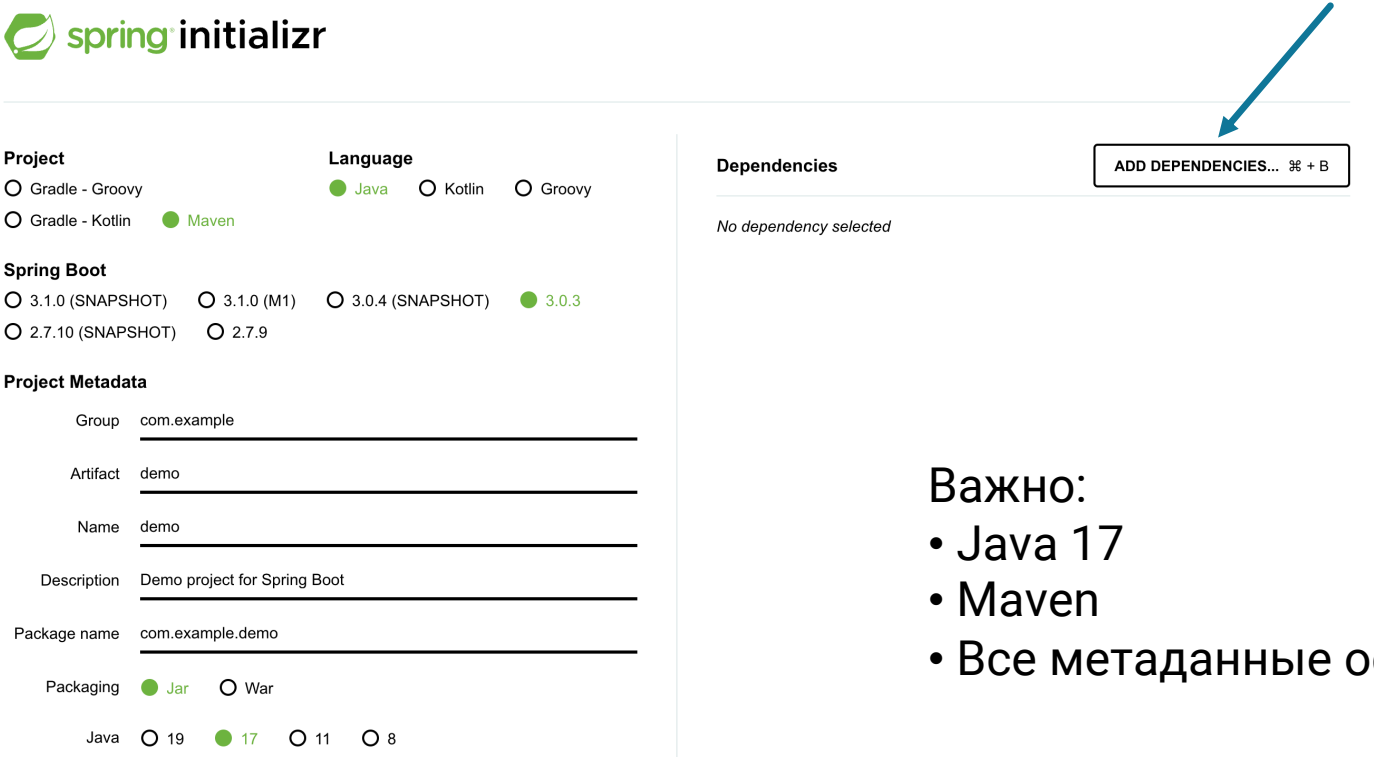
Для создания проекта нам не потребуется среда разработки.

Будем использовать Spring Initializr - конструктор стартового проекта.

1. Откройте браузер и перейдите по адресу **https://start.spring.io/**

Создаем новое приложение

ничего не добавляем
будет чистый Boot



Важно:
- Java 17
- Maven
- Все метаданные оставляем

Skillbox

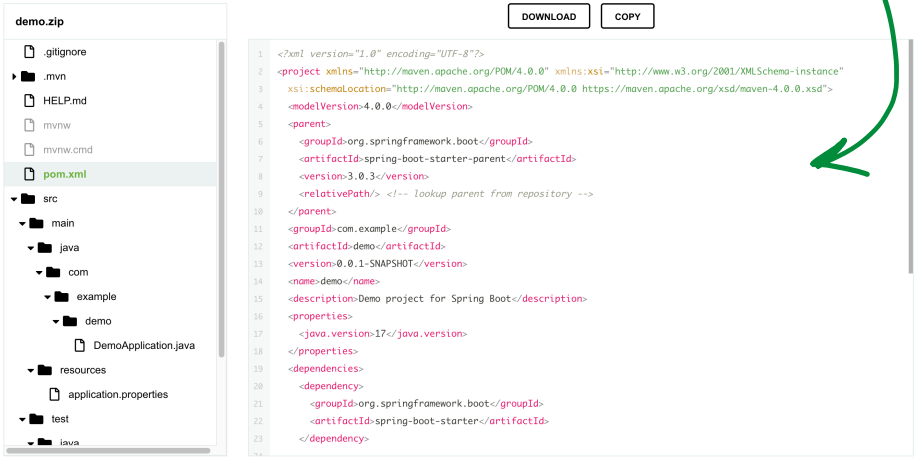
Создаем новое приложение

скачать архив
с проектом

отправить или сохранить
ссылку с настройками,
позволяет сохранить
шаблон

скачиваем

посмотреть
содержимое
проекта



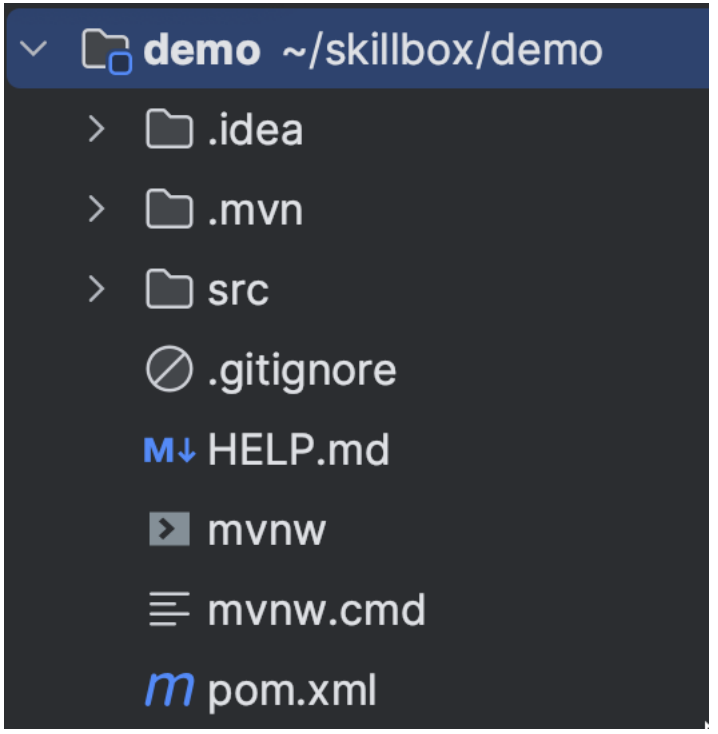
Skillbox

Создаем новое приложение

Скачивайте архив demo.zip

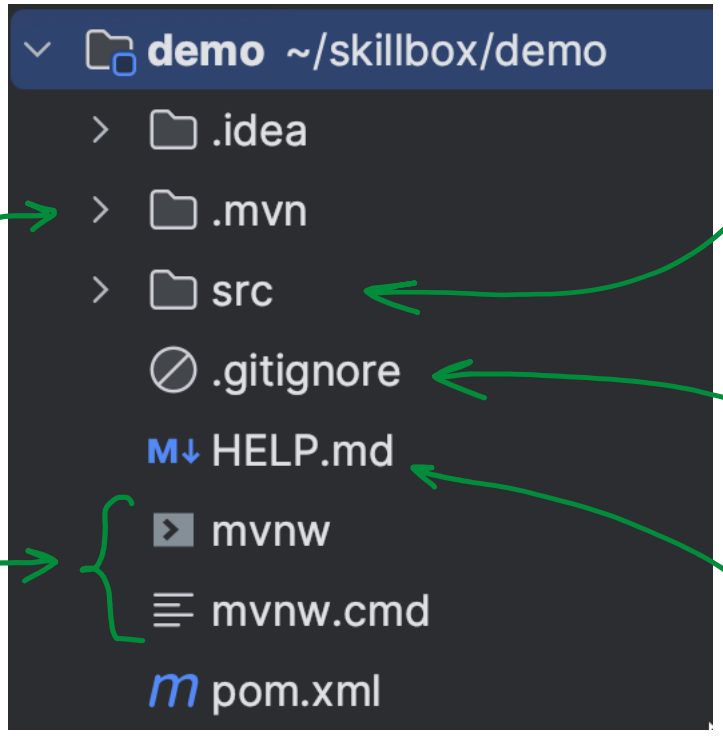
Распакуйте архив

Откройте папку проект в IDEA



Что внутри проекта?

настройки IDEA (это уже создала среда разработки)



исходный код проекта

и к Git уже готово

Maven Wrapper
не надо даже Maven
устанавливать отдельно!

шаблон описания проектам

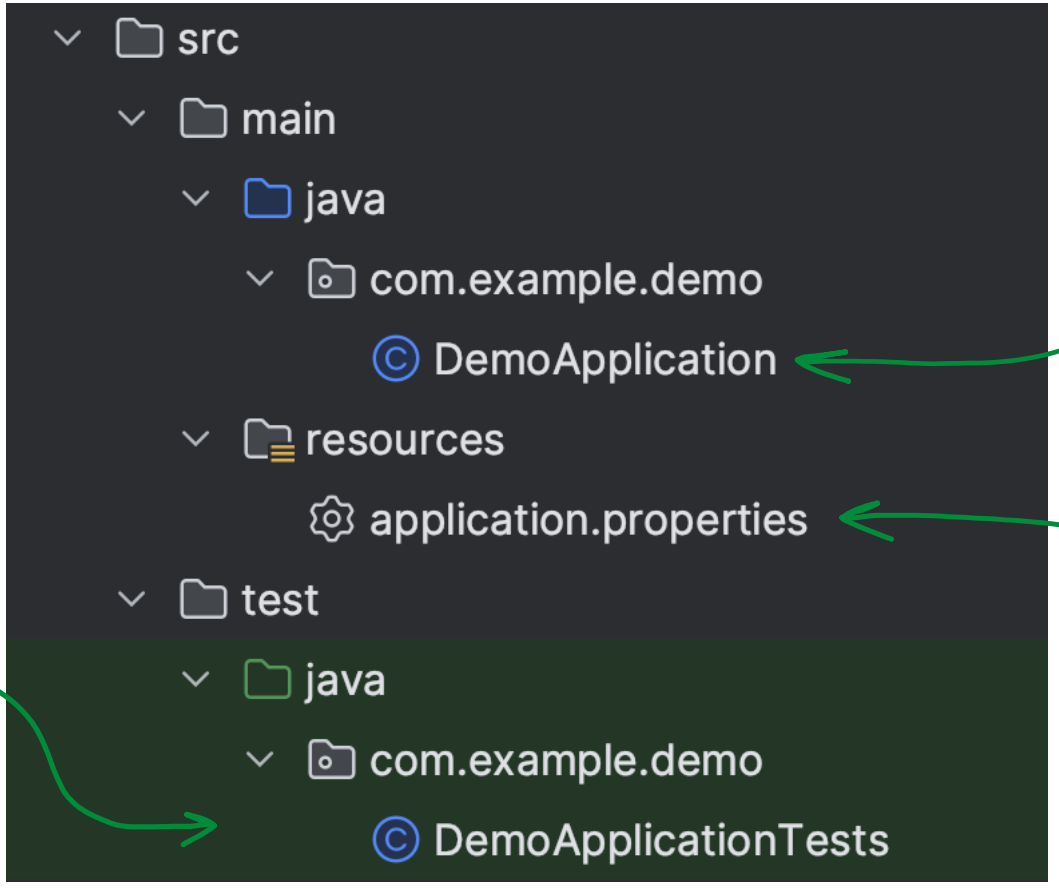
- Windows
```
mvnw.cd clean compile
```

- Linux/macOs/WSL2
```
./mvnw clean compile
```

конфигурация Maven

Skillbox

Что внутри проекта?



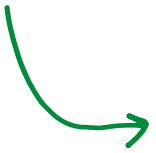
Класс с main()

настройки
Spring Boot проекта

тест запуска приложения
+ уже все есть для
написания тестов

Skillbox

Запускаем!



DemoApplication.java

```
1  package com.example.demo;
2
3  import ...
5
   1 usage
6  @SpringBootApplication
7  public class DemoApplication {
8
       no usages
9      public static void main(String[] args) {
10         SpringApplication.run(DemoApplication.class, args);
11     }
12
13 }
14
```

Skillbox

@SpringBootApplication

Все работает!

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.0.3)

2023-03-02T00:23:09.475+05:00  INFO 23479 --- [           main] com.example.demo.DemoApplication         : Starting DemoApplication using Java 17.0.4 with PID 23479 (/Users/shibk
2023-03-02T00:23:09.477+05:00  INFO 23479 --- [           main] com.example.demo.DemoApplication         : No active profile set, falling back to 1 default profile: "default"
2023-03-02T00:23:09.708+05:00  INFO 23479 --- [           main] com.example.demo.DemoApplication         : Started DemoApplication in 0.433 seconds (process running for 0.609)

Process finished with exit code 0
```

Но сразу завершает работу - ведь ему нечего делать!

Skillbox

Что такое аннотация?

Специальный тип метаданных, которые могут быть добавлены к классам, методам, переменным и другим элементам кода. Чаще всего используются для наделения свойствами и поведением сущностей, который нельзя объединить другими способами. Плюс в большинстве случаев это улучшает читаемость кода.

```
@SpringBootTest
class DemoApplicationTests {

    no usages
    @Test
    void contextLoads() {
    }

}
```

Skillbox

Что такое аннотация?

Аннотации могут использоваться для:

- пометить класс, метод - позволяет выделить эти классы среди других не прибегая к интерфейсам.

```
@Component @Service
```

- добавляет поведение - перед или после работы методы мы можем производить нужные действия.

```
@Min(5) @NotNull @Size(min = 1, max = 100)
```

- кодогенерация - создание шаблонного кода, яркий пример @Lombok и его аннотации

- документирование кода - позволяет разработчиком лучше понимать код.

```
@Deprecated @Override @Documented
```

Skillbox

@SpringBootApplication

Все эти аннотации объединены в **@SpringBootApplication**, что позволяет создать приложение Spring Boot с минимальным количеством кода.

Помимо объединения других аннотаций, **@SpringBootApplication** также наследует свойства **@Configuration**, поэтому любые настройки, определенные в классе с этой аннотацией, также будут использованы при запуске приложения.

```
Indicates a configuration class that declares one or more @Bean methods and also
triggers auto-configuration and component scanning. This is a convenience annotation
that is equivalent to declaring @Configuration, @EnableAutoConfiguration and
@ComponentScan.
Since:  1.2.0
Author: Phillip Webb, Stephane Nicoll, Andy Wilkinson

@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, clas
    @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExc
public @interface SpringBootApplication {
```

Skillbox

Spring IoC Container

Spring Dependency Container (также известный как Spring Inversion Of Control Container) - это механизм, который управляет объектами и их зависимостями в приложении, чтобы сделать их доступными для использования в других частях приложения.

BeanFactory и ApplicationContext это части Spring IoC Container.

По простому

Spring на себя берет заботы по созданию объектов, внедрение нужные зависимостей через конструкторы, геттеры.

А еще сам определяет порядок создания объектов, умеет внедрять значения из файлов конфигураций, среды окружения.

Нам остается только сказать что мы хотим увидеть в том или ином объекте.

Spring IoC Container

без Spring

- Сами создаем все объекты, выбираем конкретные реализации
- Читаем файлы конфигураций, параметры командной строки, переменные окружения и вставляем в новый объект
- Сложнее расширять код
- Писать много кода для создания всех объектов

используя Spring

- Объявляем в классах желаемый тип объекта
- Указываем куда и какое значение параметра внедрить
- Никаких наследований, код покрывается аннотациями
- Используется подход Инверсия Управления (IoC) - вся ответственность за жизненным циклом лежит на Spring, в приложении не используются new для компонентов.

Что такое Spring Bean?

Bean - это просто Java-объект, который создается, управляется и внедряется контейнером Spring. Контейнер Spring обеспечивает различные функции для бинов, такие как создание, инициализация, уничтожение и внедрение зависимостей.

В контексте Spring, бин определяется с помощью специальных аннотаций (@Component, @Service, @Repository и другие) или XML-конфигурации.

Аннотация метода

```
@Configuration
public class AppConfig {

    @Bean
    public MyBean myBean() {
        return new MyBean();
    }
```

Аннотация класса

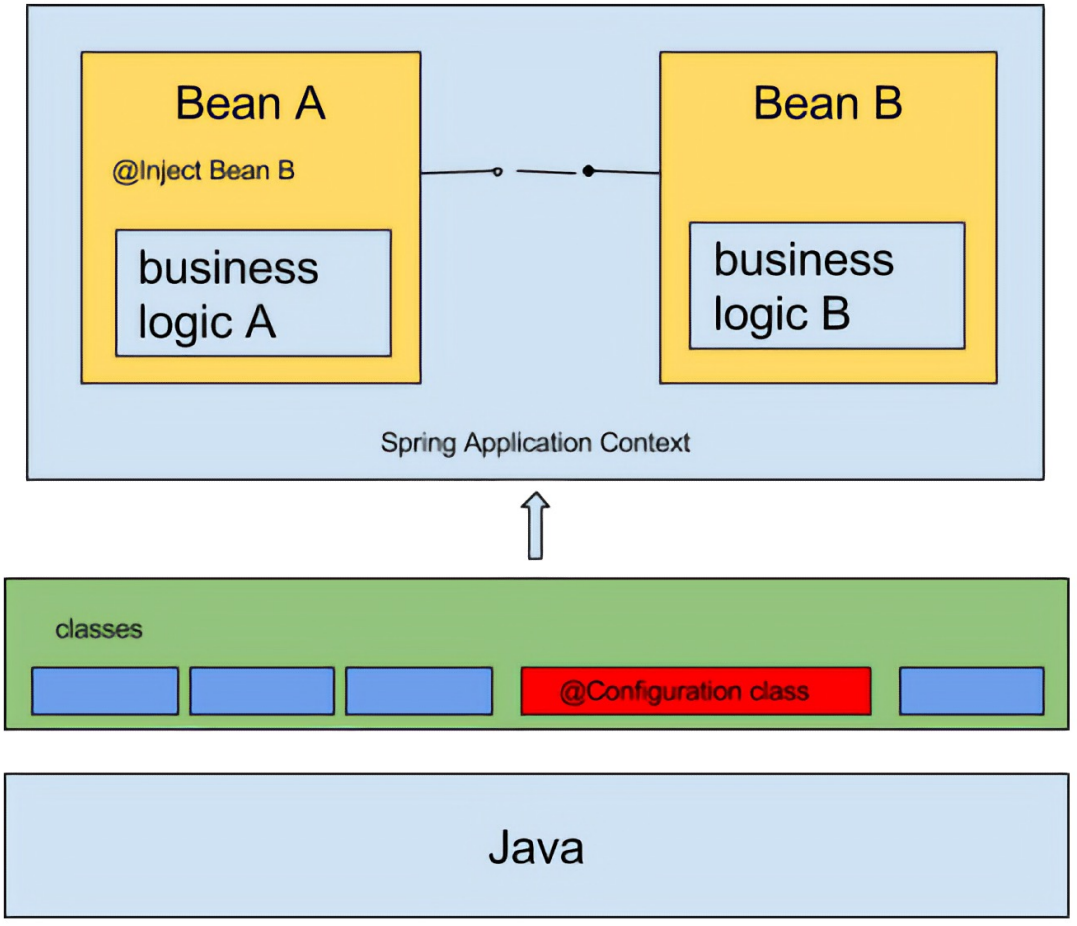
```
@Component
public class MyBean {
    // ...
}
```

XML конфигурации

```
<bean id="myBean" class="com.example.MyBean">
    <!-- Зависимости для MyBean -->
</bean>
```

Skillbox

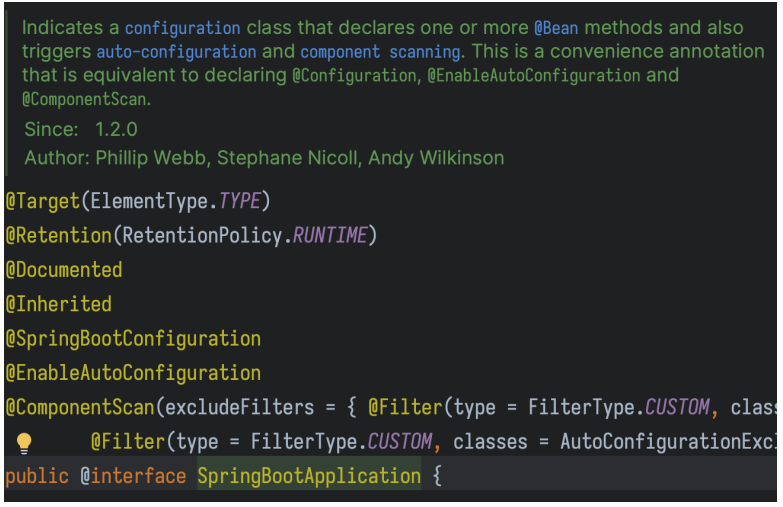
Схема Spring приложения



Skillbox

@SpringBootApplication

это композитная аннотация в Spring, которая объединяет несколько других аннотаций, чтобы упростить создание и запуск Spring Boot приложения. Эта аннотация является сокращением для трех других аннотаций: **@Configuration**, **@EnableAutoConfiguration** и **@ComponentScan**.



@Configuration

Аннотация используется для указания, что класс содержит методы, которые объявляют и конфигурируют бины (**bean**) в приложении.

Бины - это объекты, управляемые Spring IoC контейнером,
которые можно получить и использовать
в приложении через DI (Dependency Injection).

Методы, помеченные аннотацией **@Bean**,
возвращают экземпляры бинов и
могут иметь конфигурационные параметры.

@EnableAutoConfiguration

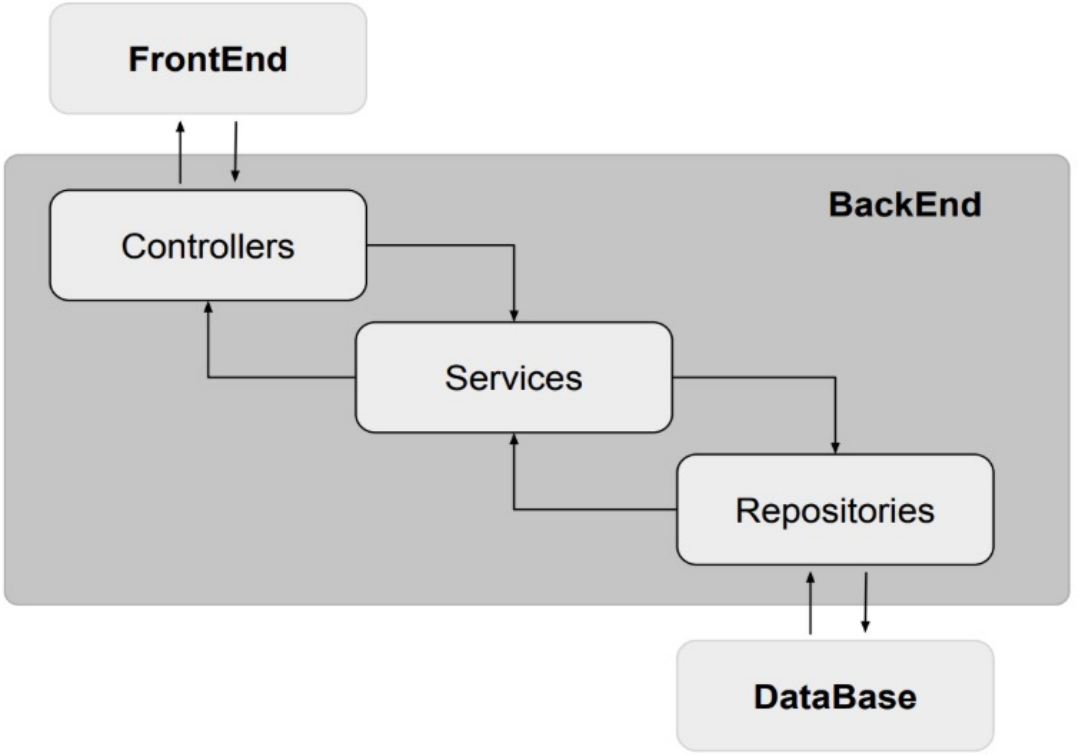
сообщает Spring, что нужно автоматически настроить приложение на основе классов находящихся в classpath, включая классы, определенные в библиотеках, которые были добавлены как зависимости в **pom.xml**.

Например, если приложение использует базу данных,
Spring Boot автоматически создаст бины для соединения с базой данных на основе настроек в **application.properties** или **application.yml**

@ComponentScan

сообщает Spring, где искать компоненты, которые могут быть включены в приложение. Компоненты могут быть помечены другими аннотациями, такими как **@Controller**, **@Service**, **@Repository** и **@Component**, которые сообщают Spring, что класс должен быть обработан и добавлен в контейнер Spring.

Структура приложения



Ошибки

```
Consider defining a bean of type 'com.example.demo.springboot.repository.Counter'
in your configuration.
```

Spring не нашел **Bean** типа **Counter** в **ApplicationContext**, причины:
- у класса **Counter** или его наследника не установлена аннотация
- у вас нет наследника такого класса если **Counter** интерфейс
- вам класс **Counter** класс имеет аннотацию, но расположен выше по иерархии папок относительно **Main** класса где выполняется **SpringApplication.run()**

```
Parameter 0 of constructor in com.example.demo.springboot.repository.
Counter required a bean of type 'com.example.demo.springboot.config.CounterSettings'
that could not be found.
```

Spring не нашел Bean типа **CounterSettings** в **ApplicationContext** при создании объекта **Counter** через конструктор, причины:
- у класса **CounterSettings** или его наследника не установлена аннотация
- у вас нет наследника такого класса если **CounterSettings** интерфейс
- вам класс **CounterSettings** класс имеет аннотацию, но расположен выше по иерархии относительно **Main** класса где выполняется **SpringApplication.run()**

Ошибки

```
Parameter 0 of constructor in com.example.demo.springboot.config.CounterSettings
required a bean of type 'int' that could not be found.
```

Spring при создании объекта типа **CounterSettings** не определил, какое значение аргумента использовать, возможно:

- это значение должно браться из конфигурации. @Value, @ConfigurationProperties,
- в классе **CounterSettings** переопределить значение по умолчанию

```
Parameter 0 of constructor in com.example.demo.springboot.repository.
Counter required a bean of type 'com.example.demo.springboot.config.CounterSettings'
that could not be found.
```

Spring не нашел Bean типа **CounterSettings** в **ApplicationContext** при создании объекта **Counter** через конструктор, причины:

- у класса **CounterSettings** или его наследника не установлена аннотация
- у вас нет наследника такого класса если **CounterSettings** интерфейс
- вам класс **CounterSettings** класс имеет аннотацию, но расположен выше по иерархии относительно **Main** класса где выполняется **SpringApplication.run()**

Подключение БД Postgres

Подключи три новые зависимости Hibernate, Spring JPA, Postgres

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

<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>6.1.7.Final</version>
</dependency>

<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.5.4</version>
</dependency>
```

Подключение БД Postgres

Создаем @Entity класс, JpaRepository и создаем нового наследника Storage

```java
@Entity
@Table(name = "customers")
@NoArgsConstructor
@Getter
@Setter
@Builder
@AllArgsConstructor
public class PersonCodeEntity {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    no usages
    @Column(name = "code")
    private String code;
}
```

```java
@Repository
@RequiredArgsConstructor
public class DbStorage implements Storage {
    3 usages
    private final PersonRepository personRepository;

    1 usage  Konstantin Shibkov
    @Override
    public int save(String text) {
        var personCode = new PersonCodeEntity( id: null, text);
        personRepository.save(personCode);
        return personCode.getId();
    }

    1 usage  Konstantin Shibkov
    @Override
    public Optional<String> findById(int id) {
        return personRepository.findById(id)
                .map(PersonCodeEntity::getCode);
    }
```

```java
public interface PersonRepository extends JpaRepository<PersonCodeEntity, Integer> {
}
```

Skillbox

Подключение БД Postgres

Создаем локально базу, например через Docker

```
docker run \
--detach \
--name=sk202303_postgres \
--env="POSTGRES_PASSWORD=12345678" \
--publish 5999:5432 \
postgres
```

Подключение БД Postgres

Вносим данные для подключения в конфиг приложения resources/application.yml

```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5999/
    username: postgres
    password: 12345678
  jpa:
    hibernate:
      ddl-auto: update
```

Ссылки

- Плагин клиент для Community
https://plugins.jetbrains.com/plugin/14723-restfulbox

- Плагин для обзора БД для Community
https://plugins.jetbrains.com/plugin/1800-database-navigator

- Плагин Pojo -> JSON (не работает с Java 17 😓)
https://plugins.jetbrains.com/plugin/12066-pojo-to-json

- конфиги быстрого запуска популярных Docker контейнеров
https://sendel.ru/posts/popular-docker-images/

Спасибо за внимание!

@sendel

Если не успели задать вопрос – пишите в телеграм

Мой канал в Telegram

@three_monitors

Skillbox