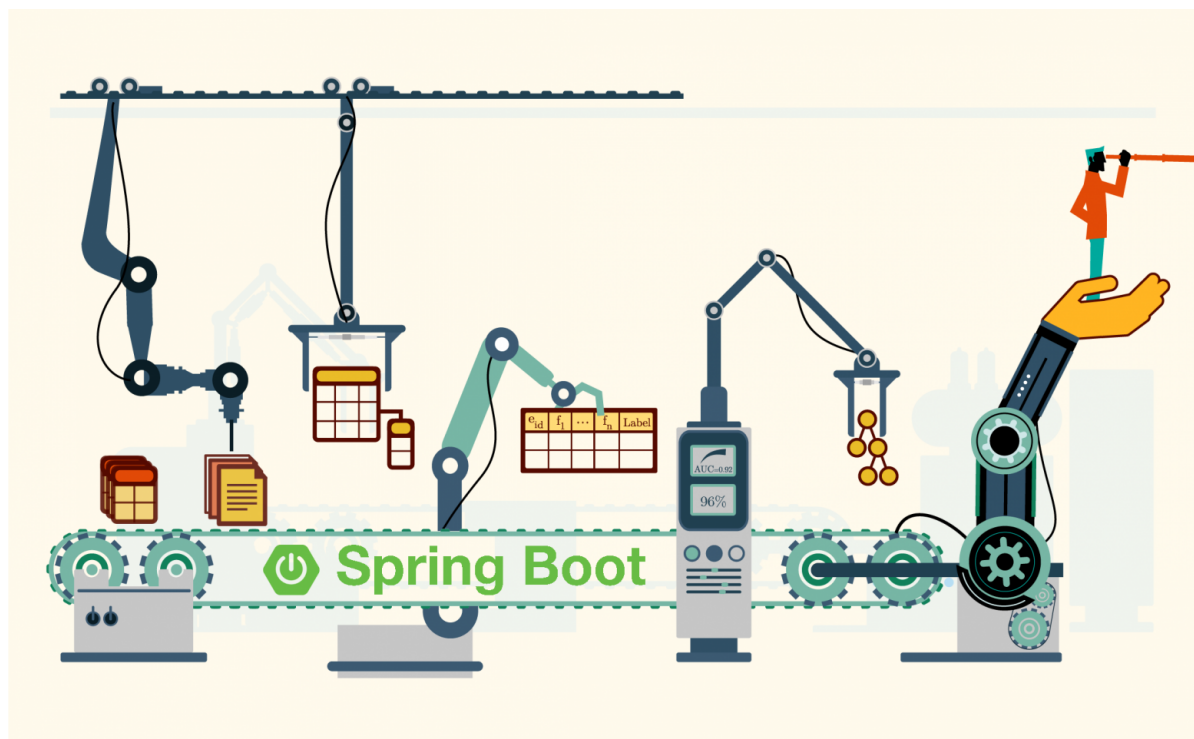


topjava 4 января 2019 в 19:53

Введение в Spring Boot: создание простого REST API на Java

Программирование*, Java*

Из песочницы



Из-за громоздкой конфигурации зависимостей настройка Spring для корпоративных приложений превратилась в весьма утомительное и подверженное ошибкам занятие. Особенно это относится к приложениям, которые используют также несколько сторонних библиотек

Каждый раз, создавая очередное корпоративное Java-приложение на основе Spring, вам необходимо повторять одни и те же рутинные шаги по его настройке:

- В зависимости от типа создаваемого приложения (Spring MVC, Spring JDBC, Spring ORM и т.д.) импортировать необходимые Spring-модули
- Импортировать библиотеку web-контейнеров (в случае web-приложений)
- Импортировать необходимые сторонние библиотеки (например, Hibernate, Jackson), при этом вы должны искать версии, совместимые с указанной версией Spring
- Конфигурировать компоненты DAO, такие, как: источники данных, управление транзакциями и т.д.
- Конфигурировать компоненты web-слоя, такие, как: диспетчер ресурсов, view resolver
- Определить класс, который загрузит все необходимые конфигурации

1. Представляем Spring Boot

Авторы Spring решили предоставить разработчикам некоторые утилиты, которые автоматизируют процедуру настройки и ускоряют процесс создания и развертывания Spring-приложений, под общим названием **Spring Boot**

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на

Suggestions:

Good article about classes

Spring Boot Tutorial

разработчиков минимум усилий по его настройке и написанию кода

2. Особенности Spring Boot

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов

2.1. Простота управления зависимостями

Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых **starter**-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.)

Starter-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволит получить универсальное решение для всех, связанных со Spring технологий, избавляя программиста от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей (пример таких дескрипторов и стартовых пакетов будет показан ниже)

Например, если вы хотите начать использовать Spring Data JPA для доступа к базе данных, просто включите в свой проект зависимость **spring-boot-starter-data-jpa** и все будет готово (вам не придется искать совместимые драйверы баз данных и библиотеки Hibernate)

Если вы хотите создать Spring web-приложение, просто добавьте зависимость **spring-boot-starter-web**, которая подтянет в проект все библиотеки, необходимые для разработки Spring MVC-приложений, таких как **spring-webmvc**, **jackson-json**, **validation-api** и **Tomcat**

Другими словами, **Spring Boot** собирает все общие зависимости и определяет их в одном месте, что позволяет разработчикам просто использовать их, вместо того, чтобы изобретать колесо каждый раз, когда они создают новое приложение

Следовательно, при использовании **Spring Boot**, файл **pom.xml** содержит намного меньше строк, чем при использовании его в Spring-приложениях

Обратитесь к [документации](#), чтобы ознакомиться со всеми **Spring Boot starter-пакетами**

2.2. Автоматическая конфигурация

Второй превосходной возможностью **Spring Boot** является автоматическая конфигурация приложения

После выбора подходящего **starter**-пакета, **Spring Boot** попытается автоматически настроить Spring-приложение на основе добавленных вами **jar**-зависимостей

Например, если вы добавите **Spring-boot-starter-web**, Spring Boot автоматически сконфигурирует такие зарегистрированные бины, как **DispatcherServlet**, **ResourceHandlers**, **MessageSource**

Если вы используете **spring-boot-starter-jdbc**, **Spring Boot** автоматически регистрирует бины **DataSource**, **EntityManagerFactory**, **TransactionManager** и считывает информацию для подключения к базе данных из файла **application.properties**

Если вы не собираетесь использовать базу данных, и не предоставляете никаких подробных сведений о подключении в ручном режиме, Spring Boot автоматически настроит базу в памяти, без какой-либо дополнительной конфигурации с вашей стороны (при наличии H2 или HSQL библиотек)

Suggestions:

Java Data Structures Tutorial

Автоматическая конфигурация может быть полностью переопределена в любой момент с помощью пользовательских настроек

2.3. Встроенная поддержка сервера приложений — контейнера сервлетов

Каждое Spring Boot web-приложение включает встроенный web-сервер. Посмотрите на [список](#) контейнеров сервлетов, которые поддерживаются "из коробки"

Разработчикам теперь не надо беспокоиться о настройке контейнера сервлетов и развертывании приложения на нем. Теперь приложение может запускаться само, как исполняемый jar-файл с использованием встроенного сервера

Если вам нужно использовать отдельный HTTP-сервер, для этого достаточно исключить зависимости по умолчанию. Spring Boot предоставляет отдельные starter-пакеты для разных HTTP-серверов

Создание автономных web-приложений со встроенными серверами не только удобно для разработки, но и является допустимым решением для приложений корпоративного уровня и становится все более полезно в мире микросервисов. Возможность быстро упаковать весь сервис (например, аутентификацию пользователя) в автономном и полностью развертываемом артефакте, который также предоставляет API — делает установку и развертывание приложения значительно проще

3. Требования к установке Spring Boot

Для настройки и запуска Spring Boot приложений требуется следующее:

- Java 8+
- Apache Maven 3.x

4. Создание Spring Boot приложения

Теперь давайте перейдем к практике и реализуем очень простой REST API для приема платежей, используя возможности Spring Boot

4.1. Создание web-проекта с использованием Maven

Создайте Maven-проект в используемой вами IDE, назвав его SpringBootRestService

Обязательно используйте версию Java 8+, поскольку Spring Boot не работает с более ранними версиями

4.2. Конфигурация pom.xml

Вторым шагом необходимо настроить Spring Boot в файле **pom.xml**

Все приложения Spring Boot конфигурируются от **spring-boot-starter-parent**, поэтому перед дальнейшим определением зависимостей, добавьте **starter-parent** следующим образом:

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
```

Suggestions:



































The Best Java course

```
<version>2.1.1.RELEASE</version>  
</parent>
```

Т.к. мы создаем REST API, то необходимо в качестве зависимости использовать **spring-boot-starter-web**, которая неявно определяет все остальные зависимости, такие как **spring-core**, **spring-web**, **spring-webmvc**, **servlet api**, и библиотеку **jackson-databind**, поэтому просто добавьте в pom.xml следующее:

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

Теперь следующие jar-библиотеки автоматически импортируются в ваш проект:

- >  Maven: ch.qos.logback:logback-classic:1.2.3
- >  Maven: ch.qos.logback:logback-core:1.2.3
- >  Maven: com.fasterxml.jackson.core:jackson-annotations:2.9.0
- >  Maven: com.fasterxml.jackson.core:jackson-core:2.9.7
- >  Maven: com.fasterxml.jackson.core:jackson-databind:2.9.7
- >  Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.9.7
- >  Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.9.7
- >  Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.9.7
- >  Maven: com.fasterxml:classmate:1.4.0
- >  Maven: javax.annotation:javax.annotation-api:1.3.2
- >  Maven: javax.validation:validation-api:2.0.1.Final
- >  Maven: org.apache.logging.log4j:log4j-api:2.11.1
- >  Maven: org.apache.logging.log4j:log4j-to-slf4j:2.11.1
- >  Maven: org.apache.tomcat.embed:tomcat-embed-core:9.0.13
- >  Maven: org.apache.tomcat.embed:tomcat-embed-el:9.0.13
- >  Maven: org.apache.tomcat.embed:tomcat-embed-websocket:9.0.13
- >  Maven: org.hibernate.validator:hibernate-validator:6.0.13.Final
- >  Maven: org.jboss.logging:jboss-logging:3.3.2.Final
- >  Maven: org.slf4j:jul-to-slf4j:1.7.25
- >  Maven: org.slf4j:slf4j-api:1.7.25
- >  Maven: org.springframework.boot:spring-boot:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-autoconfigure:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-starter:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-starter-json:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-starter-logging:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-starter-tomcat:2.1.1.RELEASE
- >  Maven: org.springframework.boot:spring-boot-starter-web:2.1.1.RELEASE
- >  Maven: org.springframework:spring-aop:5.1.3.RELEASE
- >  Maven: org.springframework:spring-beans:5.1.3.RELEASE
- >  Maven: org.springframework:spring-context:5.1.3.RELEASE
- >  Maven: org.springframework:spring-core:5.1.3.RELEASE
- >  Maven: org.springframework:spring-expression:5.1.3.RELEASE
- >  Maven: org.springframework:spring-jcl:5.1.3.RELEASE
- >  Maven: org.springframework:spring-web:5.1.3.RELEASE

Следующий шаг — добавляем Spring Boot плагин:

```
<build>  
  <plugins>
```

```

    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>

```

Последний шаг — сделать так, чтобы Maven генерировал исполняемый jar-файл при сборке:

```

<packaging>jar</packaging>

```

Ниже приведен полный файл pom.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
  <modelVersion>4.0.0</modelVersion>
  <groupId>springboot.topjava.ru</groupId>
  <artifactId>SpringBootRestService</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>

  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.1.RELEASE</version>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>

```

Как видите, используя одну зависимость, мы можем создать полностью функциональное web-приложение

4.3. Создание ресурсов REST

Теперь мы собираемся создать контроллер платежей вместе с POJO-классами для запросов и ответов

Напишем класс запроса платежа:

```
package springboot.topjava.ru;

public class PaymentRequest {

    private int userId;
    private String itemId;
    private double discount;

    public String getItemId() {
        return itemId;
    }

    public void setItemId(String itemId) {
        this.itemId = itemId;
    }

    public double getDiscount() {
        return discount;
    }

    public void setDiscount(double discount) {
        this.discount = discount;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }
}
```

А также класс, обрабатывающий базовый ответ, возвращаемый нашим сервисом:

```
package springboot.topjava.ru;

public class BaseResponse {

    private final String status;
    private final Integer code;

    public BaseResponse(String status, Integer code) {
        this.status = status;
        this.code = code;
    }

    public String getStatus() {
        return status;
    }

    public Integer getCode() {
        return code;
    }
}
```

```
}  
}
```

А теперь создадим контроллер:

```
package springboot.topjava.ru;  
  
import org.springframework.web.bind.annotation.*;  
  
@RestController  
@RequestMapping("/payment")  
public class PaymentController {  
  
    private final String sharedKey = "SHARED_KEY";  
  
    private static final String SUCCESS_STATUS = "success";  
    private static final String ERROR_STATUS = "error";  
    private static final int CODE_SUCCESS = 100;  
    private static final int AUTH_FAILURE = 102;  
  
    @GetMapping  
    public BaseResponse showStatus() {  
        return new BaseResponse(SUCCESS_STATUS, 1);  
    }  
  
    @PostMapping("/pay")  
    public BaseResponse pay(@RequestParam(value = "key") String key, @RequestBody PaymentReq  
  
        final BaseResponse response;  
  
        if (sharedKey.equalsIgnoreCase(key)) {  
            int userId = request.getUserId();  
            String itemId = request.getItemId();  
            double discount = request.getDiscount();  
            // Process the request  
            // ....  
            // Return success response to the client.  
            response = new BaseResponse(SUCCESS_STATUS, CODE_SUCCESS);  
        } else {  
            response = new BaseResponse(ERROR_STATUS, AUTH_FAILURE);  
        }  
        return response;  
    }  
}
```

4.4. Создание основного класса приложения

Этот последний шаг заключается в создании класса конфигурации и запуска приложения. Spring Boot поддерживает новую аннотацию **@SpringBootApplication**, которая эквивалентна использованию **@Configuration**, **@EnableAutoConfiguration** и **@ComponentScan** с их атрибутами по умолчанию

Таким образом, вам просто нужно создать класс, аннотированный с помощью **@SpringBootApplication**, а Spring Boot включит автоматическую настройку и отсканирует ваши ресурсы в текущем пакете:

```

package springboot.topjava.ru;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}

```

5. Развертывание приложения Spring Boot

Теперь давайте воспользуемся третьей замечательной особенностью Spring Boot — это встроенный сервер. Все, что нам нужно сделать — это создать исполняемый jar-файл с помощью Maven и запустить его, как обычное автономное приложение:

- Войдите в режим командной строки (команда `cmd`), перейдите в папку с **pom.xml** и введите команду **mvn clean package**
- Maven сгенерирует исполняемый jar-файл с именем **SpringBootRestService-1.0.jar**
- Перейдите в папку **cd target**
- Затем запустите jar-файл: **java -jar SpringBootRestService-1.0.jar**
- Перейдите в браузере по адресу <http://localhost:8080/payment>

Наш REST API запущен и готов обслуживать запросы через порт 8080 (по умолчанию)

В этой статье мы рассмотрели возможности Spring Boot и создали полностью рабочий пример с использованием встроенного сервера

Источник:

<https://dzone.com/articles/introducing-spring-boot>

В переводе обновили информацию:

- Spring-boot-starter-parent изменили версию с 1.5.8.RELEASE на 2.1.1.RELEASE и соответственно был обновлен список библиотек, которые подтягивает Maven
- Убрали объявления репозиторий Spring, зависимости подтягиваются из центрального репозитория
- В классе BaseResponse поля сделали final, добавили конструктор и убрали сеттеры
- В контроллере PaymentController ввели метод showStatus() с @GetMapping для тестирования приложения в браузере
- Заменили @RequestMapping в методах на @GetMapping/@PostMapping
- Также были внесены правки по развертыванию приложения с командной строки

UPDATE:

Как заметил @Lure_of_Chaos, теперь уже все можно сделать автоматически через **SPRING INITIALIZR**. Причем не выходя из любимой JetBrains IntelliJ IDEA.

Теги: java, spring boot, maven, web-разработка, spring-boot-starter, pom.xml, Spring Framework, создание приложения, rest, SpringBootApplication

Хабы: Программирование, Java

Редакторский дайджест



Присылаем лучшие статьи раз в месяц

Электронпочта



9

0

Карма Рейтинг

topjava @topjava

Java-программисты

Сайт Facebook ВКонтакте

Комментарии 19

ПОХОЖИЕ ПУБЛИКАЦИИ

13 февраля 2019 в 18:41

Spring и JDK 8: Вы все еще используете @Param и name/value в Spring MVC аннотациях? Тогда статья для Вас

+16

17K

63

7 +7

11 ноября 2018 в 14:56

Локальное окружение для разработки Spring Boot веб-сервисов с Docker Compose, Consul, Make

+10

16K

66

5 +5

5 июля 2017 в 15:00

Переписываем домашний проект на микросервисы (Java, Spring Boot, Gradle)

+15

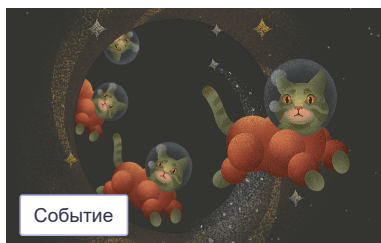
67K

244

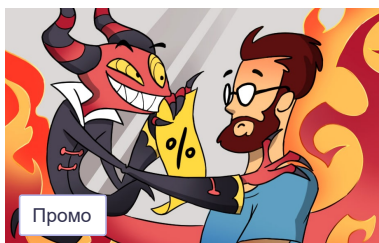
34 +34

МИНУТОЧКУ ВНИМАНИЯ

Разместить



Событие
Конкурс технических статей
Технотекст 2021



Промо
Промокод — твой билет в общество
потребления



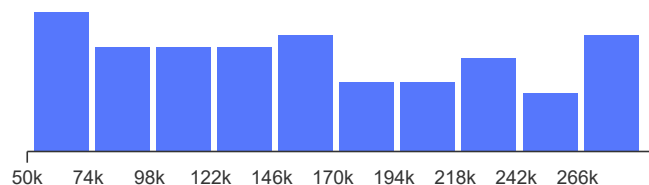
Опрос
Хотите рассказать о себе в наших
социальных сетях?

Suggestions:

Docker Tutorial for Beginners - A Full DevOps
Course on How to Run Applications in Containers

156 176 ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 8 924 анкет, за 1-ое пол. 2022 года. Проверьте «в рынке» ли ваша зарплата или нет!

[Проверить свою зарплату](#)

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 23:15

Как рисовать с помощью SQL?

+50

13K

35

17 +17

сегодня в 15:33

Неклассические контейнеры в C++

+23

2.6K

34

9 +9

сегодня в 13:15

PHP Дайджест № 222/3 (26 марта – 25 апреля 2022)

+18

1.4K

5

2 +2

сегодня в 20:16

Релокация в Индию

+16

2.6K

6

8 +8

сегодня в 13:32

Сколько зарабатывают выпускники МГТУ им.Н.Э. Баумана. Опрос конца 2021 года

+14

9.1K

12

38 +38

Небо — птицам, океаны — рыбам, а вузы — студентам, чтобы они учились и развивались[Мегапост](#)

ЧИТАЮТ СЕЙЧАС

Что происходит внутри IT-отрасли прямо сейчас? IT нас спасет? А если спасет, то кого именно?

10K 31 +31

Релокация в Индию

2.6K 8 +8

Сколько зарабатывают выпускники МГТУ им.Н.Э. Баумана. Опрос конца 2021 года

9.1K 38 +38

Не верьте улыбкам южноазиатов

29K 96 +96

Forbes: после легализации параллельного импорта стоимость техники для конечного потребителя может вырасти на 20-40%

22K 99 +99

IT-2033: ИскИн, агротех, low code и доверие между людьми

Турбо

РАБОТА

Java разработчик
497 вакансий

Все вакансии

Ваш аккаунт

Войти
Регистрация

Разделы

Публикации
Новости
Хабы
Компании
Авторы
Песочница

Информация

Устройство сайта
Для авторов
Для компаний
Документы
Соглашение
Конфиденциальность

Услуги

Реклама
Тарифы
Контент
Семинары
Мегапроекты



Настройка языка

О сайте

Техническая поддержка

Вернуться на старую версию