

Основы веб-разработки на Spring Framework

# Spring Boot

Spring Boot. Архитектура. Модель. Обзор решений.

## Оглавление

[Spring Boot](#)

[Быстрая разработка приложений со Spring Boot](#)

[Стартеры](#)

[Указание версии Java](#)

[Spring Boot Maven plugin](#)

[Свойства \(application.properties\)](#)

[Особенности различных применений](#)

[Статический контент](#)

[Практическое задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

# Spring Boot

Spring Boot упрощает создание Spring-приложений: для большинства из них требуется очень небольшая настройка конфигурации, но при этом их можно настраивать по своему усмотрению. С помощью Spring Boot можно создавать Java-приложения, которые запускаются из командной строки через `java -jar`, или более традиционно — путем развертывания `war`-архива на сервере приложений.

Для того, чтобы создать с нуля полноценное Spring Boot приложение можно воспользоваться сервисом `start.spring.io` — это веб-сервис, призванный построить каркас приложения с определенной структурой каталогов и файлов, которая понятна современным средствам автоматизированной сборки проектов (например, Gradle или Maven).

После генерации проекта сервисом получаем `zip`-архив, содержащий его готовую файловую структуру:

1. Файл **`pom.xml`** — файл описания проекта.
2. Папка **`src`** — содержит все исходные файлы.
3. Папка **`src/main`** — исходные файлы разрабатываемого приложения.
4. Папка **`src/main/java`** — исходный Java-код.
5. Папка **`src/main/resources`** — файлы, используемые при компиляции и исполнении.
6. Папка **`src/test`** — исходные файлы для автоматического тестирования.
7. Папка **`src/test/java`** — исходные файлы Java для автоматического тестирования.

В `pom`-файле, описывающем проект, видим и стартер по умолчанию, и запрошенные ранее зависимости.

```
<?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/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.3.RELEASE</version>
        <relativePath/>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>

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

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

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>

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

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

```

Инициализатор **Spring Boot** внутри **IntelliJ IDEA Ultimate** также использует описанный выше сервис, чтобы создать файл **pom.xml** для нового Spring-проекта.

## Быстрая разработка приложений со Spring Boot

Из стартового набора рассмотрим еще два файла:

- **DemoApplication.java** — класс начальной загрузки и конфигурации Spring;
- **application.properties** — файл настройки свойств Spring Boot.

```
@RestController
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @RequestMapping("/")
    public String helloWorld() {
        return "Hello World";
    }
}
```

У класса **DemoApplication** двойное назначение:

1. Получает управление при запуске **jar**.
2. Отвечает за конфигурирование приложения благодаря аннотации **@SpringBootApplication**.

Аннотация **@SpringBootApplication** в числе прочих наследуется от аннотаций **@Configuration**, **@ComponentScan**, **@EnableAutoConfiguration**:

- **@Configuration** — объявляет класс классом Java-based конфигурации;
- **@ComponentScan** — позволяет выполнять компонентное сканирование;
- **@EnableAutoConfiguration** — это вызов «магии»: она сообщает Spring Boot, что необходимо «угадать», как вы хотите настроить фреймворк, основываясь на добавленных зависимостях.

Поскольку стартер **spring-boot-starter-web** добавил **Tomcat** и **Spring MVC**, автоконфигурация предполагает, что вы разрабатываете веб-приложение.

## Стартеры

Стартеры — это наборы удобных дескрипторов зависимостей, которые можно включить в приложение. Например, чтобы использовать Spring и JPA для доступа к базе данных, нужно просто добавить зависимость **spring-boot-starter-data-jpa** в проект. Стартеры содержат множество зависимостей, необходимых для быстрого запуска проекта с помощью согласованного и поддерживаемого набора управляемых транзитивных зависимостей. Все официальные стартеры следуют единой схеме именования: **spring-boot-starter-\***, где **\*** — конкретный тип приложения.

Вот несколько популярных стартеров Spring Boot:

- **spring-boot-starter-web** — используется для создания веб-служб RESTful с использованием Spring MVC и Tomcat в качестве встроенного контейнера приложений;
- **spring-boot-starter-thymeleaf** — подключение шаблонизатора Thymeleaf;
- **spring-boot-starter-data-jpa** — подключает модуль Spring Data JPA;
- **spring-boot-starter-security** — подключает модуль Spring Security для обеспечения безопасности веб-приложения;
- **spring-boot-starter-jersey** — альтернатива Spring-boot-starter-web, в которой используется встроенный сервер приложений Jersey, а не Tomcat;
- **spring-boot-starter-jdbc** — реализует пул соединений JDBC, основан на реализации пула JDBC Tomcat.

Вернемся к pom-файлу, в котором мы наследуем свой проект от специального стартера:

```
...
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.3.RELEASE</version>
</parent>
...
```

**spring-boot-starter-parent** — это специальный стартер, предоставляющий настройки Maven по умолчанию и раздел управления зависимостями, чтобы вы могли опустить теги версии для Spring-зависимостей.

## Указание версии Java

По-умолчанию, в **spring-boot-starter-parent** указана совместимость с Java 6. Для того, чтобы изменить версию на более новую, достаточно в pom.xml в элементе <properties> прописать элемент <java.properties> с соответствующей версией.

```
<properties>
  <java.version>1.8</java.version>
</properties>
```

## Spring Boot Maven plugin

По умолчанию инициализатор **Spring Boot** в секции <plugins> pom-файла указывает плагин **spring-boot-maven-plugin**:

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

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

Данный плагин решает несколько полезных задач:

- перепackовывает выходной jar-файл проекта в исполняемый **uber-jar**, включающий в себя все jar-файлы зависимостей проекта, что очень удобно при развертывании;
- находит точку входа в проекте — класс, содержащий функцию **main()**, указывает его в манифесте переупакованного jar-файла (uber-jar);

```
Main-Class: org.springframework.boot.loader.JarLauncher
Start-Class: com.example.demo.DemoApplication
```

- при указании war-упаковки перепackовывает выходной war-файл проекта в war-файл, включающий в себя все jar-файлы зависимостей;

```
<packaging>war</packaging>
```

- предоставляет определитель версии зависимостей, который устанавливает номер версии для соответствия зависимостям Spring Boot. Это избавляет от необходимости указывать их версию. Кроме того, вы всегда можете переопределить данное поведение и указать версию по своему усмотрению.

## Свойства (application.properties)

Наследование от **spring-boot-starter-parent** включает использование файла свойств **application.properties**, в котором могут быть указаны свойства проекта — например, порт, на котором слушает встроенный web-сервер, или настройки Spring-бинов.

Указание 8189-го порта (а не на 8080, как указано по умолчанию) выглядит следующим образом:

```
server.port=8189
```

Ранее для гибкого указания свойств JDBC-соединения (не посредством Java-конфигурирования) приходилось описывать бины в файле **applicationContext.xml**. Так по-прежнему можно делать, если вы привыкли или любите XML.

Spring Boot, зная, что вы используете реализацию JDBC-драйвера (в нашем примере это **H2 Database : com.h2database**), сам создает бины DataSource, свойства которых выставляет по умолчанию. Чтобы выставить другие значения, их можно прописать в файле свойств.

Например, по умолчанию H2-драйвер использует URL-подключения к in-memory БД **jdbc:h2:mem:testdb**. Указываем в файле свойств:

```
spring.datasource.url=jdbc:h2:~/spring.h2
spring.datasource.driver-class-name=org.h2.Driver
```

# Особенности различных применений

## Статический контент

По умолчанию Spring Boot будет отдавать статический контент из каталога `/static` (или `/public`, или `/resources`, или `/META-INF/resources`) в `classpath` или из каталога `ServletContext`. Он использует `ResourceHttpRequestHandler` из Spring MVC, поэтому вы можете изменить это поведение, добавив свой `WebMvcConfigurerAdapter` и переопределив метод `addResourceHandlers`.

По умолчанию статический контент отображается на корень (`/`) веб-приложения с точки зрения браузера. Но можно настроить иное отображение, используя свойство `spring.mvc.static-path-pattern`. Например, для отображения статического контента в `/resources/**`:

```
spring.mvc.static-path-pattern=/resources/**
```

Физическое расположение статического контента можно указать, используя свойство `spring.resources.static-locations`. Можно указать список из нескольких локаций.

В дополнение к обычным статическим ресурсам можно использовать упакованные в формате [WebJar](#). Для этого необходимо дополнить `pom.xml` соответствующими зависимостями:

```
<dependencies>
...
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>jquery</artifactId>
  <version>2.1.1</version>
</dependency>
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>3.2.0</version>
</dependency>
...
</dependencies>
```

Далее мы можем ссылаться на эти статические ресурсы следующим образом:

```
<link rel='stylesheet' href='/webjars/bootstrap/3.2.0/css/bootstrap.min.css'>
<script type="text/javascript"
src="/webjars/bootstrap/3.2.0/js/bootstrap.min.js"></script>
```

Заметьте, что версию внешнего ресурса (js-библиотеки, css-файлов) приходится указывать ссылкой на ресурс и следить, чтобы она соответствовала версии, указанной в зависимости, — а это неудобно. Решение — подключить в зависимость локатор JS/CSS-библиотек:

```
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>webjars-locator</artifactId>
```

```
<version>RELEASE</version>
</dependency>
```

Теперь указание версии можно опустить:

```
<link rel='stylesheet' href='/webjars/bootstrap/css/bootstrap.min.css'>
<script type="text/javascript"
src="/webjars/bootstrap/js/bootstrap.min.js"></script>
```

## Практическое задание

1. Перенести функциональность, реализованную на прошлых занятиях, на платформу Spring Boot.

## Дополнительные материалы

1. [Обратная сторона Spring](#).
2. [Spring Boot. Boot up your development \(Sergey Morenets\)](#).

## Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Официальная документация проекта Spring Boot](#).
2. [Spring Boot basics](#).