Лабораторная работа №6

Общий вопрос:

Что такое JDBC, JPA и Hibernate? В чем их разница?

- Hibernate – это библиотека, которая предназначена для задач объектно-реляционного отображения. Примерно такое описание будет в википедии. Если простыми словами –  hibernate позволяет разработчику работать с базой данных не напрямую, как мы это делали с помощью библиотеки JDBC в статье [Работа с базой данных,](http://java-master.com/%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B1%D0%B0%D0%B7%D0%BE%D0%B9-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-mysql-%D0%B2-java/) а  с помощью представления таблиц баз данных в виде классов java.

JPA –  спецификация, которая дает возможность сохранять в удобном виде Java-объекты в базе данных. Hibernate – это одна из самых популярных реализаций этой спецификации. Вот такая вот драма. Данные термины даны только в ознакомительных целях так как очень часто запутывают и пугают начинающих разработчиков.

JDBC - платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94), реализованный в виде пакета java.sql, входящего в состав [Java SE](https://ru.wikipedia.org/wiki/Java_SE).

JDBC основан на концепции так называемых драйверов, позволяющих получать соединение с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) по специально описанному [URL](https://ru.wikipedia.org/wiki/URL). Драйверы могут загружаться динамически (во время работы программы). Загрузившись, [драйвер](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) сам регистрирует себя и вызывается автоматически, когда программа требует [URL](https://ru.wikipedia.org/wiki/URL), содержащий протокол, за который драйвер отвечает.

Самый первый способ подключить джава приложение к базе данных – это использование JDBC (Java Database Connectivity).

Sun Microsystems выпустила JDBC как часть JDK в феврале 1997 года. Классы JDBC содержатся в пакетах java.sql и javax.sql для подключения и управления данными в базах данных.

Можно сказать, что JDBC – это мост между миром Java и миром баз данных. Ведь первое, что мы ищем когда хотим подключить базу данных к нашему приложению – jdbc драйвер. Если мы работаем с [Maven](https://java-master.com/maven-%d0%b8-java/), тогда мы ищем зависимость для драйвера к определенной базе.

Одним из недостатков JDBC является то, что код, который получается в конце – выглядит очень большим по объему (хотя работы он выполняет не много). Также, очень сложно приходится, когда объект джава который мы пытаемся сохранить в базе или достать – достаточно большой. Нужно правильно замапить поля базы данных и поля класса Java. Бывает такое, что в процессе работы нужно добавить поле к готовой таблице в базе. И потом нужно находить все запросы в базу в коде приложения чтобы добавить это поле. Чтобы прочувствовать всю “боль” работы с JDBC, нужно с ним поработать. А для тех, кто уже знаком с этим функционалом я думаю объяснения будут излишне.

JPA призвана устранить вышеперечисленные недостатки.

Спецификация Java Persistence API – это технология, которая позволяет удобно мапить объект Java и таблицу базы данных.

 В *JDBC* при написании каждого запроса, Вам необходимо указать в коде все детали, необходимые для операций *CRUD*, такие как имена таблиц, имена столбцов. В *JPA* (которая использует *JDBC* “под капотом”) Вы также указываете эти данные, но уже только один раз, когда навешиваете аннотации на *Java* класс:



Аннотации JPA

Сама по себе спецификация *JPA* не является инструментом или фреймворком; скорее, она определяет набор концепций, которые могут и должны быть реализованы любым другим инструментом.

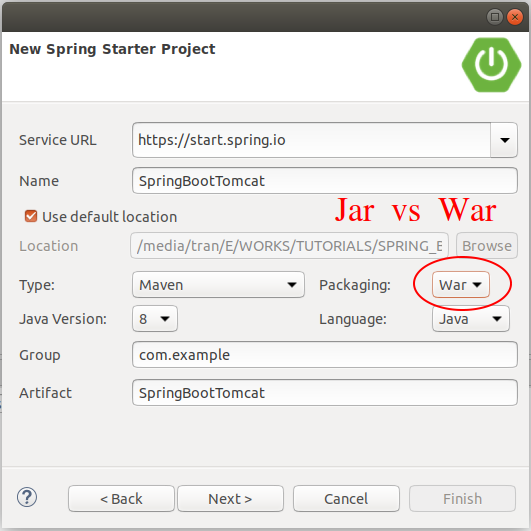
Так как *JPA* – это просто спецификация, Вам нужен инструмент для ее реализации. Этим инструментом может быть *Hibernate*, *TopLink*, *iBatis* и т. д.

Так как Hibernate самый популярный ORM (Object Relational Mapping) фреймворк для работы с базой данных, я оставил свой выбор на нем. Но то же самое касается  и других библиотек ORM о которых Вы скорее всего даже не слышали.

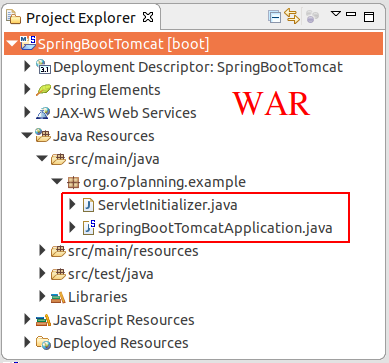
Вы можете рассматривать JPA как интерфейс, а Hibernate – как реализацию. Без Hibernate, от JPA в Вашем коде будет мало полезности. Хотя как Hibernate так и JPA можно использовать по отдельности в связке с другими инструментами.

1. Как развернуть приложение Spring Boot на сервере (tomcat)?

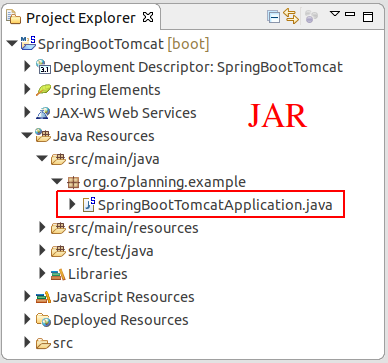
Скачиваем tomcat, устанавливаем. При создании проекта **Spring Boot** у вас есть 2 выбора для упаковки данного проекта в **WAR** или **JAR**.



Когда вы создаете проект **Spring Boot** с опцией **"Packaging = War"**, **Eclipse** создаст для вас 2 класса **"AbcApplication & ServletInitializer"**. Файлы **WAR** подходят для применения (deploy) на **Web Server**.



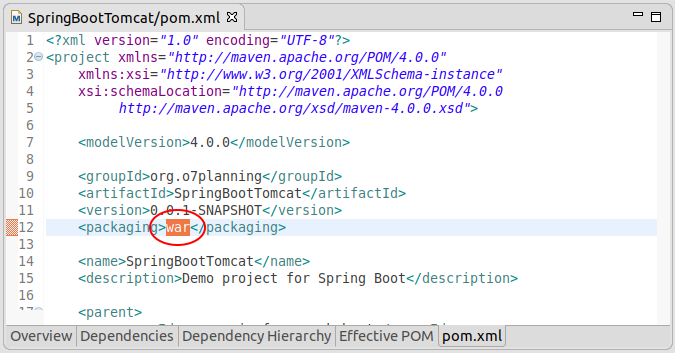
При этом, если вы создаете проект **Spring Boot** с опцией **"Packaging = Jar"**, будет создан только один класс **AbcApplication**. Приложение **Spring Boot** при упаковке в файл **JAR**, может выполнить (Executabe) самостоятельно без необходимости применения какого-либо **Web Server**. Но файл **JAR** не подходит для развертывания на **Web Server**.



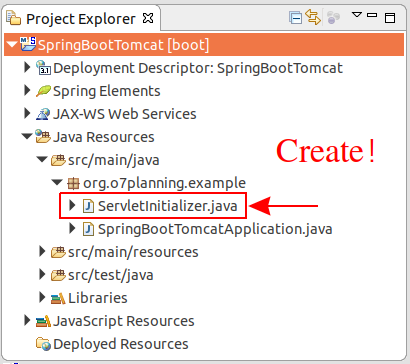
WAR File.

Если у вас уже есть готовый проект **Spring Boot**, вам нужно изменить метод упаковки (Packaging) в **WAR**:

На **Eclipse** откройте файл **pom.xml**, и измените метод упаковки (packaging) в **war**.



Если ваш проект не имеет класс **ServletInitializer**, создайте его.



ServletInitializer.java

package org.o7planning.example;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(SpringBootTomcatApplication.class);

}

}

Добавьте следующий конфигурационный сниппет в **pom.xml**:

\*\* pom.xml \*\*

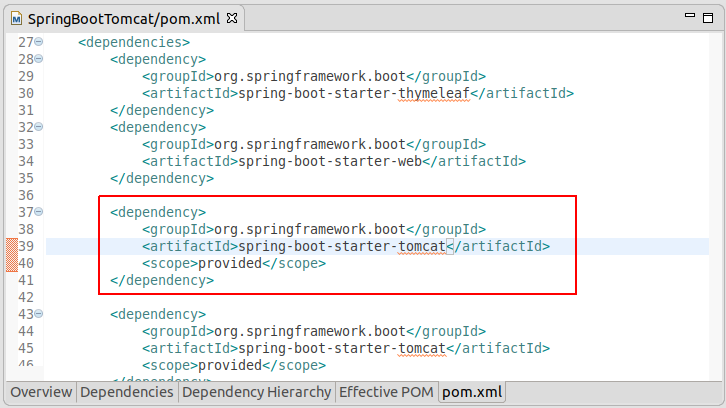
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>



Если имеется более чем 1 класс **Application** в вашем проекте, скажите **Spring Boot** какой класс будет использоваться для вашего приложения.

\*\* pom.xml \*\*

<properties>

<start-class>org.o7planning.example.OtherSpringBootWebApplication</start-class>

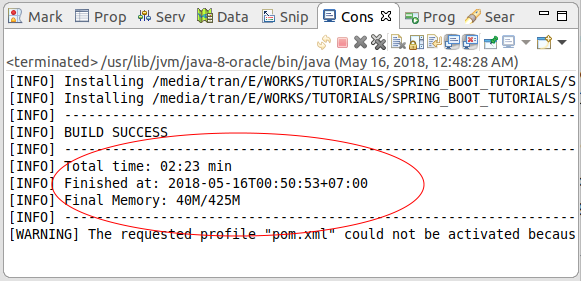
</properties>

**3- Install Spring Boot**

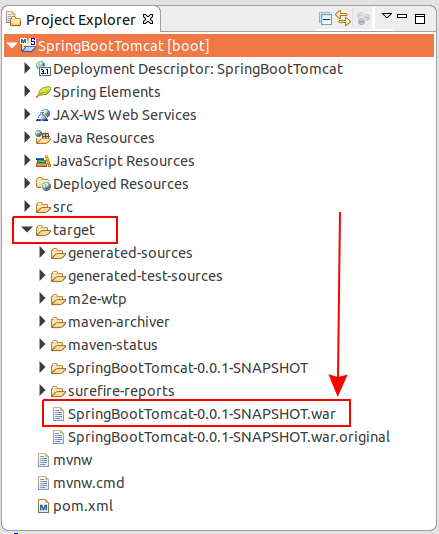
Следующий шаг это вам нужно использовать **Maven** чтобы создать файл **WAR**. Нужно удостовериться ваш **Eclipse** использует **JDK** вместо **JRE**, если нет то произойдет ошибка в данном процессе.

Нажмите на правую кнопку мыши в Project и выберите:

* **Run As/Maven Install**



Теперь у вас будет файл **WAR** расположенный в папке **target** проекта, вы можете использовать данный файл для развертывания на **Tomcat Server**.



1. В чем разница между Spring Boot и Spring MVC?

Spring — один из наиболее широко используемых фреймворков для разработки интерпрайз-приложений, обеспечивающий продуманную модель программирования и конфигурации. Целью создания данного фреймворка способствовало желание упростить разработку приложений на популярном в то время (да и сейчас) Java EE стеке технологий от компании Oracle, который на тот момент был очень сложен и не всегда удобен в использовании.

В отличие от других платформ, Spring фокусируется на нескольких областях функционирования приложений и предоставляет для них широкий спектр дополнительных функций.

Одной из основных особенностей Spring Framework является использование паттерна Dependency Injection. DI помогает намного проще реализовывать необходимую приложениям функциональность, а также позволяет разрабатывать слабо связанные классы, делая их более универсальными.

Если Spring Framework фокусируется на предоставлении гибкости, то Spring Boot стремится сократить длину кода и упростить разработку web-приложения. Используя конфигурацию при помощи аннотаций и стандартного кода, Spring Boot сокращает время, затрачиваемое на разработку приложений. Данная возможность помогает создать автономные приложения с меньшими или почти нулевыми затратами на их конфигурацию.

Автоконфигурация — это особенность Spring Boot. При помощи аннотаций он автоматически настраивает специальные конфигурационные классы.

Теперь рассмотрим некоторые особенности и преимущества упомянутых фреймворков.

Преимущества Spring Framework

* Spring Framework может быть задействован на всех архитектурных слоях, применяемых при при разработке web-приложений   
    
  • Использует модель POJO при написании классов, а это очень легкая структура   
    
  • Позволяет свободно связывать модули и легко их тестировать   
    
  • Поддерживает декларативное программирование   
    
  • Избавляет от самостоятельного создания фабричных и синглтон-классов   
    
  • Поддерживает различные способы конфигурации   
    
  • Предоставляет сервис уровня middleware

Несмотря на наличие стольких преимуществ, которыми обладает Spring, длительная процедура подготовки, связанная с его настройкой, способствовала появлению Spring Boot.

Преимущества Spring Boot

Spring Boot упрощает использование Spring Framework. Spring обеспечивает слабо связанное приложение — это отличная функция. Однако, когда есть несколько слабо связанных блоков, отслеживание их становится утомительной и неблагодарной задачей. Тут-то Spring Boot и помогает упростить все, не предлагая никакой функции конфигурации.

* Spring Boot не требует развертывания war-файлов   
    
  • Создает автономные приложения   
    
  • Помогает напрямую встроить в приложение Tomcat, Jetty или Undertow   
    
  • Не требует XML-конфигурации   
    
  • Направлен на уменьшение объема исходного кода   
    
  • Имеет дополнительную функциональность «из коробки»   
    
  • Простота запуска   
    
  • Простая настройка и управление

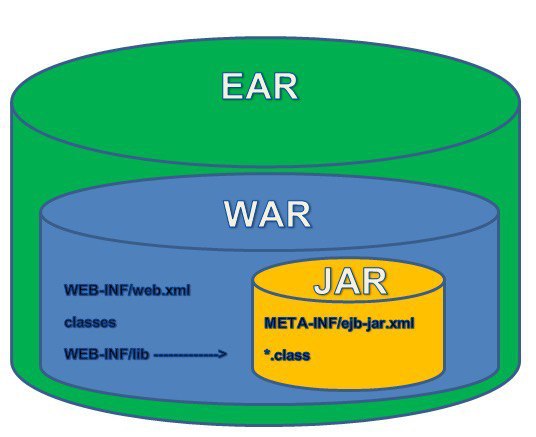
Благодаря таким функциям, как автоконфигурация, Spring Boot избавляет вас от написания лишнего кода и помогает избежать ненужной настройки.

Spring Framework не только предлагает вам такие функции, как внедрение зависимостей или обработка транзакций, но также выступает в качестве основы для других фреймворков Spring. Лучшим примером для этого является Spring Boot. Spring Boot использует Spring Framework в качестве своей основы. Он упрощает зависимости Spring и запускает приложения прямо из командной строки. Он также не требует наличия внешнего контейнера приложений. Spring Boot помогает контролировать компоненты приложения и настраивает их извне.

В заключение отметим, что Spring Framework внес значительный вклад в развитие разработки Java EE приложений и продолжает это делать. Благодаря многим функциям, упомянутым выше, Spring Framework является отличным выбором для разработчиков. Но все достоинства этого фреймворка особенно проявляются при его использовании вместе с Spring Boot. Дополнительные преимущества, которые предоставляет Spring Boot, имеют большое значение для разработчиков, поскольку они позволяют завершать проекты с меньшими усилиями. Для задач, возникающих при использовании Spring Framework, Spring Boot является решением, которое ускорит их разработку.

1. Что такое War-файл и отличие от Jar-файла?

**WAR – Web Archive.** **Технически имеет ту же структуру, но другую роль – архив JavaEE web-компонента**. Обычно содержит jar-ы с реализацией, JSP, статические файлы фронт-энда, и мета-информацию для сервлет-контейнера ( web. xml ).

jar и war – расширения платформо-независимых файлов-архивов Java-приложения. Кроме них есть еще третий формат, ear. Все эти форматы – на самом деле просто zip-архивы с классами и другими частями приложения. Вы можете переименовать такой файл в .zip, разархивировать, и посмотреть, что внутри.  
  
JAR – **J**ava **Ar**chive. Содержит файлы классов, ресурсы, зависимые библиотеки, и другие необходимые для приложения файлы. Может содержать точку входа, и использоваться как цель для исполнения команды java.  
  
WAR – **W**eb **Ar**chive. Технически имеет ту же структуру, но другую роль – архив JavaEE web-компонента. Обычно содержит jar-ы с реализацией, JSP, статические файлы фронт-энда, и мета-информацию для сервлет-контейнера (web.xml). В основном используется как деплоймент web-приложения в сервлет-контейнер. С приходом Servlet API 3.0 и embedded-контейнеров, всё больше становится принято запаковывать и web-компоненты в самодостаточные jar (Лозунг Spring Boot: Make jar, not war).  
  
EAR – **E**nterprise **Ar**chive. Для разработчиков, начавших карьеру уже во времена Spring Framework, может показаться совсем экзотичным. Это собранное воедино монолитное JavaEE приложение. Содержит дескрипторы деплоймента и JavaEE модули (веб-модули, EJB, клиентские модули, адаптеры ресурсов). Деплоится в JavaEE Application Server.  
  
Все три типа файлов собираются утилитой jar из JDK. [](https://itsobes.ru/assets/JavaSobes/196.jpg)

1. Зачем использовать 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 основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

Spring Boot позволяет вам легко создавать полноценные, производственного класса Spring-приложения, про которые можно сказать - "просто запусти". Мы включили Spring-платформу и сторонние библиотеки, чтобы вы могли запустить с минимум усилиями. Большинству Spring Boot приложениям требуется совсем маленькая Spring-конфигурация.

**Возможности**

* Создание полноценных Spring приложений
* Встроенный Tomcat или Jetty (не требуется установки WAR файлов)
* Обеспечивает 'начальные' POMs для упрощения вашей Maven конфигурации
* Автоматическая конфигурирация Spring когда это возможно
* Обеспечивает такими возможностями, как метрики, мониторинг состояниями и расширенная конфигурация
* Абсолютно **без генерации кода** и **без написания XML** конфигурации

1. Как можно запустить Spring Boot?

Запуск из IDE

Вы можете запустить приложение Spring Boot из IDE как простое Java приложение. Однако сначала вам нужно импортировать ваш проект. Этапы импорта зависят от вашей IDE и системы сборки. Большинство IDE могут импортировать проекты Maven напрямую. Например, пользователи Eclipse могут выбрать «Импорт… → Существующие проекты Maven» из меню «Файл».

Если вы не можете напрямую импортировать свой проект в IDE, вы можете создать метаданные IDE с помощью подключаемого модуля сборки. Maven включает плагины для Eclipse и IDEA. Gradle предлагает плагины для различных IDE.

Если вы случайно запустили веб-приложение дважды, вы увидите ошибку “Port already in use” («Порт уже используется»). Пользователи STS могут использовать кнопку «Перезапустить» вместо кнопки «Выполнить», чтобы убедиться, что любой существующий экземпляр закрыт.

Запуск в виде упакованного приложения

Если вы используете Spring Boot Maven или Gradle плагины для создания исполняемого файла jar, вы можете запустить свое приложение, используя java -jar, как показано в следующем примере:

$ java -jar target/myapplication-0.0.1-SNAPSHOT.jar

Также возможно запустить упакованное приложение с включенной поддержкой удаленной отладки. Это позволяет подключить отладчик к упакованному приложению, как показано в следующем примере:

$ java -Xdebug -Xrunjdwp:server=y,transport=dt\_socket,address=8000,suspend=n \

-jar target/myapplication-0.0.1-SNAPSHOT.jar

Использование Maven плагина

Spring Boot Maven плагин включает цель run, которую можно использовать для быстрой компиляции и запуска приложения. Приложения запускаются в разобранном виде, как в вашей IDE. В следующем примере показана типичная команда Maven для запуска приложения Spring Boot:

$ mvn spring-boot:run

Вы также можете использовать переменную среды операционной системы MAVEN\_OPTS, как показано в следующем примере:

$ export MAVEN\_OPTS=-Xmx1024m

Использование Gradle плагина

Spring Boot Gradle плагин также включает в себя задачу bootRun, которую можно использовать для запуска приложения в разобранном виде. Задача bootRun добавляется всякий раз, когда вы применяете плагины org.springframework.boot и java, и отображается в следующем примере:

$ gradle bootRun

Вы также можете использовать переменную среды операционной системы JAVA\_OPTS, как показано в следующем примере:

$ export JAVA\_OPTS=-Xmx1024m

Горячая замена (Hot Swapping)

Поскольку приложения Spring Boot представляют собой простые Java-приложения, горячая замена JVM должна работать «из коробки». Горячая замена JVM несколько ограничена байт-кодом, который она может заменить. Для более полного решения можно использовать JRebel.

Модуль spring-boot-devtools также включает поддержку быстрого перезапуска приложения.

## Различные способы запуска метода после запуска Spring Boot

У каждого способа есть свои преимущества. Давайте рассмотрим подробнее, чтобы решить, что нам следует использовать:

1. интерфейс CommandLineRunner
2. интерфейс ApplicationRunner
3. события Spring Boot Application
4. аннотацию @Postconstruct для метода
5. интерфейс InitializingBean
6. атрибут инициализации аннотации @bean
7. Требуется ли для запуска Spring Boot отдельный контейнер?

Каждое Spring Boot web-приложение включает встроенный web-сервер. 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.

1. Как настроить приложение Spring Boot с помощью maven?

#### Maven

Пользователи Maven могут наследовать от проекта spring-boot-starter-parent для получения разумных значений по умолчанию. parent проект предоставляет следующие возможности:

* Java 1.8 как уровень компилятора по умолчанию.
* Кодировка UTF-8 источника.
* Раздел управления зависимостями, унаследованный от spring-boot-dependencies pom, который управляет версиями общих зависимостей. Это управление зависимостями позволяет вам опустить теги <version> для этих зависимостей при использовании в вашем pom.
* Выполнение цели repackage с идентификатором выполнения repackage.
* Разумная фильтрация ресурсов.
* Разумная конфигурация плагина (exec plugin, Git commit ID и shade).
* Разумная фильтрация ресурсов для application.properties и application.yml, включая специфичные для профиля файлы (например, application-dev.properties и application-dev.yml)

Обратите внимание, что, поскольку файлы application.properties и application.yml принимают заполнители в стиле Spring (${…}), фильтрация Maven изменяется на использование @..@ заполнителей. (Вы можете переопределить это, установив свойство Maven с именем resource.delimiter.)

#### Наследование parent стартера

Чтобы настроить ваш проект для наследования от spring-boot-starter-parent, установите parent следующим образом:

<!-- Inherit defaults from Spring Boot -->

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.4.RELEASE</version>

</parent>

Вы должны указать только номер версии Spring Boot для этой зависимости. Если вы импортируете дополнительные стартеры, вы можете спокойно пропустить номер версии.

С помощью этой настройки вы также можете переопределить отдельные зависимости, переопределив свойство в вашем собственном проекте. Например, чтобы перейти на другой релиз Spring Data, вы должны добавить следующее в ваш pom.xml:

<properties>

<spring-data-releasetrain.version>Fowler-SR2</spring-data-releasetrain.version>

</properties>

#### Использование Spring Boot без parent POM

Не всем нравится наследование от POM spring-boot-starter-parent. У вас может быть собственный корпоративный стандартный parent, который вам нужно использовать, или вы можете явно объявить всю свою конфигурацию Maven.

Если вы не хотите использовать spring-boot-starter-parent, вы все равно можете сохранить преимущества управления зависимостями (но не управления плагинами), используя зависимость scope=import следующим образом:

<dependencyManagement>

<dependencies>

<dependency>

<!-- Import dependency management from Spring Boot -->

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>2.2.4.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

Предыдущий пример настройки не позволяет переопределять отдельные зависимости с помощью свойства, как описано выше. Чтобы достичь того же результата, вам нужно добавить запись в dependencyManagement вашего проекта перед записью spring-boot-dependencies. Например, чтобы перейти на другой релиз Spring Data, вы можете добавить следующий элемент в ваш файл pom.xml:

<dependencyManagement>

<dependencies>

<!-- Override Spring Data release train provided by Spring Boot -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-releasetrain</artifactId>

<version>Fowler-SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>2.2.4.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

#### Использование плагина Spring Boot Maven

Spring Boot включает в себя плагин Maven, который может упаковать проект в виде исполняемого файла jar. Добавьте плагин в раздел <plugins>, если вы хотите его использовать, как показано в следующем примере:

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

Если вы используете Spring Boot parent pom starter, вам нужно добавить только плагин. Нет необходимости настраивать его, если вы не хотите изменять параметры, определенные в parent элементе

1. Какие встроенные серверы поддерживает Spring Boot?

Spring Boot включает поддержку следующих встроенных реактивных веб-серверов: **Reactor Netty, Tomcat, Jetty и Undertow**. Большинство разработчиков используют соответствующий «Starter» для получения полностью настроенного экземпляра.

1. Для чего используются аннотации @RestController и @Controller?

Классы контроллеров в Spring аннотируются аннотацией @Controller или @RestController . Они помечают классы контроллеров как обработчики запросов, чтобы Spring мог распознать их как службу RESTful во время выполнения.

@Controller - это специализация универсальной стереотипной @Component , которая позволяет распознавать класс как управляемый Spring компонент.

@Controller расширяет возможности использования @Component и отмечает аннотированный класс как бизнес-уровень или уровень представления. Когда запрос сделан, это проинформирует DispatcherServlet о включении класса контроллера в сканирование методов, отображаемых аннотацией @RequestMapping

Интерфейс RestController аннотируется @Controller и @ResponseBody вместо того, чтобы напрямую аннотировать его с помощью @Component .

Если мы заменим аннотацию нашего контроллера на @RestController , нам не нужно будет изменять домен и уровень сохраняемости, поскольку они все равно будут совместимы с этой аннотацией.

1. Что такое Thymeleaf?

**Thymeleaf** — современный серверный механизм Java-шаблонов для веб- и автономных сред, способный обрабатывать HTML, XML, JavaScript, CSS и даже простой текст.  
  
Основной целью Thymeleaf является создание элегантного и удобного способа шаблонизации. Чтобы достичь этого, Thymeleaf основывается на концепции Natural Templates, чтобы внедрить свою логику в файлы шаблонов таким образом, чтобы этот шаблон не влиял на отображение прототипа дизайна. Это улучшает коммуникацию в команде и уменьшает разрыв между дизайнерско-программистскими группами.  
  
Thymeleaf также был разработан с самого начала с учетом стандартов Web, особенно **HTML5**, что позволяет вам создавать полностью соответствующие стандарту шаблоны.