**Контрольные вопросы**

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

*JDBC* — платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE. JDBC основан на концепции так называемых драйверов, позволяющих получать соединение с базой данных по специально описанному URL.

*Java Persistence API* — спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных. Существует несколько реализаций этого интерфейса, одна из самых популярных использует для этого Hibernate. JPA реализует концепцию ORM.

*Hibernate* — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения, самая популярная реализация спецификации JPA. Распространяется свободно на условиях GNU Lesser General Public License.

Различия между JDBC и Hibernate:

1. Hibernate и JDBC Главное отличие заключается в том, что Hibernate сначала извлекает отображение объектов в кэше (т.е. операции Hibernate – это объекты), а JDBC является прямой операционной базой данных.

2. Hibernate – это легкий объектный пакет JDBC, который является отдельным объектом постоянным слоем рамы. Hibernate можно использовать в любом JDBC.

3. Hibernate – это рамка, тесно связанная с JDBC, так что Hibernate совместимость и драйвер совместимости JDBC, и база данных имеет определенные отношения, но используют свою программу Java, нет никаких отношений с сервером App, и нет проблем со совместимости.

4. Если технология JDBC используется правильно, эффективность его выполнения должна быть лучше, чем Hibernate, потому что Hibernate – это технология на основе JDBC.

5. JDBC использует оператор SQL, Hibernate использует оператор HQL, но оператор HQL в конечном итоге преобразуется в операторы SQL.

**1. Как развернуть приложение Spring Boot и Hibernate? В чем их разница?**

Традиционный способ развертывания – заставить класс Spring Boot Application @SpringBootApplication расширить класс SpringBootServletInitializer . Файл класса Spring Boot Servlet Initializer позволяет настроить приложение при его запуске с помощью контейнера сервлетов. Нам нужно расширить класс SpringBootServletInitializer для поддержки развертывания WAR-файла. Для Maven добавьте начальный класс в свойствах pom.xml. Мы должны обновить упаковочный JAR-файл в WAR, используя следующие фрагменты кода:

<packaging>war</packaging>

Для Maven добавьте стартовую зависимость Spring Boot в pom.xml, используя код:

<dependency>

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

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

</dependency>

Теперь создайте файл WAR для развертывания на сервере Tomcat. Теперь запустите сервер Tomcat и разверните файл WAR в каталоге webapps. После успешного развертывания, нажмите на URL в вашем веб-браузере http: // localhost: 8080 / …

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

Spring MVC – это фреймворк MVC, основанный на сервлетах. Он решает проблему разработки WEB с помощью Dispatcher Servlet и ModelAndView. Но его конфигурация громоздка, с большим количеством файлов .xml и .properties. И при использовании maven для управления проектами легко возникают конфликты пакетов jar.

Spring Boot – это пакет для быстрой разработки и интеграции на основе Spring, который включает не только Spring mvc, но также Spring JPA и Spring Security реализует автоматическую настройку и снижает сложность построения проекта. А самодостаточный tomcat.pom.xml может быть настроен как стартовый, что упрощает настройку и решает проблему конфликтов пакетов jar.

Для веб-приложения на Java необходим фреймворк (набор библиотек, архитектур, шаблонов проектирования для минимизации повторяющегося кода), и один из популярных веб фреймворков на Java — это Spring MVC, который крутится под капотом у Spring Boot. Под экосистемой Spring лежит очень много библиотек, но в проектах не все они нужны. И они спроектированы так, чтобы можно было их подключать гибко, не мешая друг другу. Но со временем, когда в проекте становится много библиотек и кода, то сконфигурировать их становится нетривиальной задачей. И хочется взять готовый шаблон, который работает вместе хорошо. Вот Spring Boot предоставляет возможность склеивать между собой много Spring библиотек легко и просто, в пару строчек кода или конфигурации.

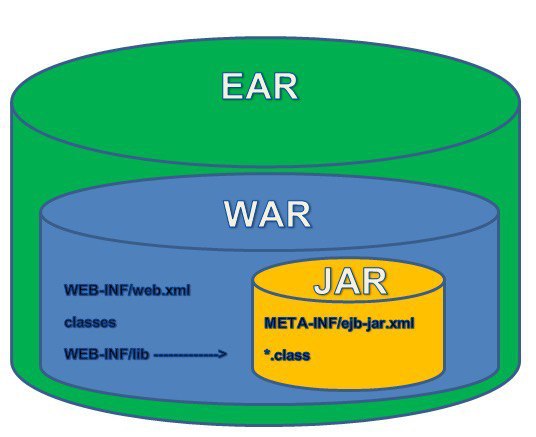
А под капотом у Spring MVC лежит Servlet API. Можно написать веб-приложение, используя только Servlet API, но объем кода, особенно повторяющегося, будет очень большим.

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

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 – Java Archive. 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 – 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). 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.



4. Зачем использовать 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** обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов.

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.).

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.

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.

5. Как можно запустить Spring Boot?

*Запуск из IDE*

Можно запустить приложение Spring Boot из IDE как простое Java приложение. Однако сначала 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 не можете напрямую импортировать свой проект в IDE, можно 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.

6. Требуется ли для запуска Spring Boot отдельный контейнер?

Да, требуется отдельный контейнер. Проект Spring Boot будет иметь встроенный tomcat, проект можно запустить только с помощью простой команды, но в производственной среде мы по-прежнему используем публикацию проекта в третьем контейнере сервлета. Изменение способа упаковки проекта на war:

<packaging>jar</packaging>

7. Как настроить приложение Spring Boot с помощью maven?

Spring Boot можно использовать с "классическими" инструментами разработки Java или установить в качестве инструмента командной строки. В любом случае вам нужен Java SDK v1.8 или выше. Прежде чем начать, вы должны проверить текущую установку Java с помощью следующей команды:

$ java -version

Вы можете использовать Spring Boot так же, как и любую стандартную библиотеку Java. Для этого включите соответствующие файлы spring-boot-\*.jar в ваш classpath. Spring Boot не требует каких-либо специальных инструментов интеграции, поэтому вы можете использовать любую IDE или текстовый редактор. Кроме того, в приложении Spring Boot нет ничего особенного, поэтому вы можете запускать и отлаживать приложение Spring Boot, как и любую другую программу Java.

Хотя вы можете копировать jar-файлы Spring Boot, обычно рекомендуется использовать инструмент сборки, который поддерживает управление зависимостями (например, Maven или Gradle).

Maven установка

Spring Boot совместим с Apache Maven 3.3 или выше. Если у вас еще не установлен Maven, вы можете следовать инструкциям на maven.apache.org.

Во многих операционных системах Maven может быть установлен с менеджером пакетов. Если вы используете OSX Homebrew, попробуйте установить brew maven. Пользователи Ubuntu могут запустить sudo apt-get install maven. Пользователи Windows с Chocolatey могут запустить команду choco install maven из привилегированного администратора.

Зависимости Spring Boot используют идентификатор группы org.springframework.boot. Как правило, ваш POM-файл Maven наследуется от проекта spring-boot-starter-parent и объявляет зависимости для одного или нескольких "Starters". Spring Boot также предоставляет дополнительный плагин Maven для создания исполняемых jar-файлов.

Следующий листинг показывает типичный 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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>myproject</artifactId>

<version>0.0.1-SNAPSHOT</version>

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

<parent>

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

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

<version>2.2.4.RELEASE</version>

</parent>

<!-- Override inherited settings -->

<description/>

<developers>

<developer/>

</developers>

<licenses>

<license/>

</licenses>

<scm>

<url/>

</scm>

<url/>

<!-- Add typical dependencies for a web application -->

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

<!-- Package as an executable jar -->

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

Spring-boot-starter-parent - отличный способ использовать Spring Boot, но он может подходить не всегда. Иногда вам может потребоваться наследование от другого родительского POM, или вам могут потребоваться настройки, отличающиеся от настроек по умолчанию.

8. 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. По умолчанию встроенный сервер прослушивает HTTP-запросы через порт 8080. При автоматической настройке сервера Reactor Netty или Jetty Spring Boot создаст определенные bean-компоненты, которые будут предоставлять HTTP-ресурсы экземпляру сервера: ReactorResourceFactory или JettyResourceFactory.

По умолчанию эти ресурсы также будут совместно использоваться клиентами Reactor Netty и Jetty для оптимальной производительности, учитывая:

* та же технология используется для сервера и клиента
* экземпляр клиента создается с помощью bean-компонента WebClient.Builder, автоматически настраиваемого Spring Boot

Разработчики могут переопределить конфигурацию ресурсов для Jetty и Reactor Netty, предоставив пользовательский компонент ReactorResourceFactory или JettyResourceFactory – это будет применяться как к клиентам, так и к серверам.

9. Для чего используются аннотации @RestController, @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. @RestController в Spring, по сути, представляют собой просто комбинацию @Controller и @ResponseBody. Эта аннотация была добавлена в Spring 4.0, чтобы удалить избыточность объявления @ResponseBody в вашем контроллере. 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.

10. 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.

Из коробки, Thymeleaf позволяет работать с шестью типами шаблонов, каждый из которых называется «Формат шаблона/Template Mode»:

HTML

XML

TEXT

JAVASCRIPT

CSS

RAW

**Thymeleaf** — чрезвычайно расширяемый механизм (на самом деле его можно назвать платформой шаблонов), который позволяет вам определять и настраивать способ обработки ваших шаблонов до тонкого уровня детализации.