**Общий вопрос: 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.

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.

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.

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.

Можно рассматривать JPA как интерфейс, а Hibernate – как реализацию. Без Hibernate, от JPA в 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)?**

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. Нужно расширить класс SpringBootServletInitializer для поддержки развертывания WAR-файла. **Настройка основного класса.** В Spring Boot нам нужно упомянуть основной класс, который должен запускаться в файле сборки. Для Maven добавьте начальный класс в свойствах pom.xml. Для Gradle добавьте имя основного класса в build.gradle. Обновите упаковку JAR в WAR . Мы должны обновить упаковочный JAR-файл в WAR.

Для Maven добавьте упаковку как WAR в pom.xml. Для Gradle добавьте плагин приложения и плагин war в build.gradle. Теперь напишем простую конечную точку отдыха, чтобы вернуть строку «Hello World from Tomcat». Чтобы написать конечную точку отдыха, нам нужно добавить в наш файл сборки зависимость веб-стартера Spring Boot.

## Для Maven добавьте стартовую зависимость Spring Boot в pom.xm. Для Gradle добавьте начальную зависимость Spring Boot в build.gradle. Теперь напишите простой файл конечной точки отдыха в файле класса Spring Boot Application. Упаковка вашего приложения

Теперь создайте файл WAR для развертывания на сервере Tomcat, используя команды Maven и Gradle для упаковки вашего приложения, как указано ниже:

## Для Maven используйте команду mvn package для упаковки вашего приложения. Затем будет создан WAR-файл, и вы сможете найти его в целевом каталоге. Для Gradle используйте команду gradle clean build для упаковки вашего приложения. Затем ваш WAR-файл будет создан, и вы сможете найти его в каталоге build / libs . Развернуть в Tomcat

Теперь запустите сервер Tomcat и разверните файл WAR в каталоге webapps .

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

**Spring MVC:**

Spring широко используется для создания масштабируемых приложений. Для веб-приложений Spring предоставляет платформу Spring MVC, которая является широко используемым модулем Spring, который используется для создания масштабируемых веб-приложений. Фреймворк Spring MVC позволяет разделить модули, а именно Model View, Controller, и легко справляется с интеграцией приложений. Это позволяет разработчику создавать сложные приложения, также используя простые классы Java. Объект модели может передаваться между представлением и контроллером с помощью карт. В этой статье мы увидим, как настроить приложение Spring MVC в Eclipse IDE, и поймем, как создавать приложения.

Фреймворк Spring MVC состоит из следующих компонентов:

* **Модель –** Модель может быть объектом или коллекцией объектов, которые в основном содержат данные приложения.
* **Вид –** Представление используется для отображения информации пользователю в определенном формате. Spring поддерживает различные технологии, такие как **freemarker** , **velocity** и **тимелист** .
* **Контроллер –** Он содержит логическую часть приложения.
* *Аннотация @Controller* используется для обозначения этого класса как контроллера.
* **Фронтальный контроллер –** Он по-прежнему отвечает за управление потоком веб-приложения. Сервлет диспетчера действует как фронт-контроллер в Spring MVC.

**Spring Boot:**  
Spring Boot построен на основе обычной пружинной основы. Таким образом, он обладает всеми функциями пружины и при этом проще в использовании, чем пружина. Spring Boot - это фреймворк на основе микросервисов, позволяющий создавать готовое к работе приложение за очень короткое время. В Spring Boot все настраивается автоматически. Нам просто нужно использовать правильную конфигурацию для использования определенной функции. Spring Boot очень полезен, если мы хотим разработать REST API. Spring Boot предоставляет возможность конвертировать наш проект в файлы war или jar. Кроме того, экземпляр Tomcat также можно запустить в облаке.

В Spring Boot есть четыре основных уровня:

* **Уровень презентации –** Как следует из названия, он состоит из представлений (т.е. части внешнего интерфейса).
* **Уровень доступа к данным –**
* CRUD (создание, получение, обновление, удаление) операций с базой данных подпадает под эту категорию.
* **Уровень обслуживания –**
* Он состоит из классов обслуживания и использует службы, предоставляемые уровнями доступа к данным.
* **Уровень интеграции –**
* Он состоит из различных веб-служб (любая служба, доступная в Интернете и использующая систему обмена сообщениями XML).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. | Spring MVC - это представление модели и веб-фреймворк на основе контроллера, широко используемый для разработки веб-приложений. | | Spring Boot построен на основе обычной инфраструктуры Spring, широко используемой для разработки REST API. | |
| 2. | Если мы используем Spring MVC, нам нужно создать конфигурацию вручную. | | Если мы используем Spring Boot, нет необходимости создавать конфигурацию вручную. | |
| 3. | В Spring MVC требуется дескриптор развертывания. | | В Spring Boot дескриптор развертывания не требуется. | |
| 4. | Spring MVC определяет каждую зависимость отдельно. | Он объединяет зависимости в единый блок. | |
| 5. | Фреймворк Spring MVC состоит из четырех компонентов: модели, представления, контроллера и переднего контроллера. | В Spring Boot есть четыре основных уровня: уровень представления, уровень доступа к данным, уровень обслуживания и уровень интеграции. | |
| 6. | На разработку уходит больше времени. | Это сокращает время разработки и увеличивает производительность. | |

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

## ****Упаковка JAR****

Проще говоря, JAR - или Java Archive - это формат файла пакета. Файлы JAR имеют расширение .jar и могут содержать **библиотеки, ресурсы и файлы метаданных.**

По сути, это заархивированный файл, содержащий сжатые версии файлов .class и ресурсы скомпилированных библиотек и приложений Java.

## ****Упаковка JAR****

Проще говоря, JAR - или Java Archive - это формат файла пакета. Файлы JAR имеют расширение .jar и могут содержать **библиотеки, ресурсы и файлы метаданных.**

По сути, это заархивированный файл, содержащий сжатые версии файлов .class и ресурсы скомпилированных библиотек и приложений Java.

## ****Ключевые различия****

Итак, каковы основные различия между этими двумя типами архивов?

Первое и наиболее очевидное отличие - это **расширение файла** . Файлы JAR имеют расширение .jar , а файл WAR - расширение .war .

Второе главное отличие - **их предназначение и способ работы** . Файлы JAR позволяют нам упаковать несколько файлов, чтобы использовать их как библиотеку, плагин или любое приложение. С другой стороны, файлы WAR используются только для веб-приложений.

**Структура архивов также отличается.** Мы можем создать JAR с любой желаемой структурой. Напротив, WAR имеет предопределенную структуру с каталогами WEB-INF и META-INF .

Наконец, мы можем **запустить JAR из командной строки,** если мы создадим его как исполняемый JAR без использования дополнительного программного обеспечения. Или мы можем использовать его как библиотеку. Напротив, нам **нужен сервер для выполнения WAR** .

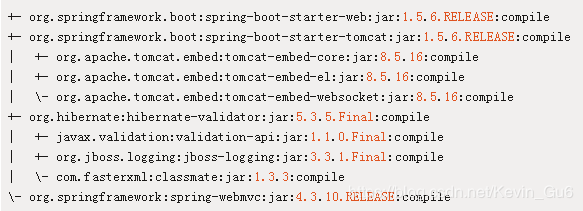
**4. Зачем использовать Spring Boot?**

Spring Boot имеет множество преимуществ, таких как:

* **Работать самостоятельно**

Spring Boot также встроен в различные контейнеры сервлетов, такие как Tomcat, Jetty и т. д. Теперь нет необходимости упаковывать как военный пакет и развертывать его в контейнере. Только Spring Boot должен быть упакован как исполняемый пакет jar. Может работать независимо, все зависимые пакеты находятся в пакете jar.

* **Упрощенная конфигурация**

Spring-boot-starter-web starter автоматически зависит от других компонентов, упрощая настройку maven.  


* **Автоматическая настройка**

Spring Boot может автоматически настраивать bean-компоненты в соответствии с классами и пакетами jar в текущем пути к классам. Например, добавление стартера spring-boot-starter-web может иметь веб-функции без другой конфигурации.

* **Без генерации кода и конфигурации XML**

В процессе настройки Spring Boot код не генерируется, и вся работа по настройке может быть завершена без использования файла конфигурации XML. Все это делается с помощью условных аннотаций, что также является основной функцией Spring 4.x one.

* **Мониторинг приложений**

Spring Boot предоставляет ряд конечных точек для мониторинга служб и приложений и выполнения проверок работоспособности.

**5. Как можно запустить Spring Boot?**

1) Команда для упаковки или запуска в контейнере  
2) Запуск с плагином Maven / Gradle  
3) Выполнить основной метод напрямую

Spring Boot построен поверх spring и содержит все функции spring. И в наши дни становится фаворитом разработчиков из-за его быстрой рабочей среды, которая позволяет разработчикам напрямую сосредоточиться на логике, а не бороться с настройкой и настройкой. Spring Boot - это платформа на основе микросервисов, и создание в ней готового к работе приложения занимает очень мало времени. Ниже приведены некоторые функции Spring Boot:

* Это позволяет избежать тяжелой конфигурации XML, которая присутствует в spring
* Оно обеспечивает простое обслуживание и создание конечных точек REST
* Он включает в себя встроенный Tomcat-сервер
* Развертывание очень простое, файлы war и jar можно легко развернуть на сервере tomcat

Как правило, для разработки приложения Spring Boot мы выбираем **Eclipse, Spring Tool Suite** и **IntelliJ IDEA** IDE.

### Запустите приложение Spring Boot в Eclipse IDE

Среда разработки Eclipse известна интегрированной средой разработки Java (IDE), но у нее есть ряд довольно интересных IDE, включая среду разработки C / C ++, среду разработки JavaScript / TypeScript, среду разработки PHP и многое другое.

**IntelliJ IDEA** - это интегрированная среда разработки (IDE), написанная на Java. Он используется для разработки компьютерного программного обеспечения. Эта среда IDE разработана компанией Jetbrains и доступна в виде лицензионной версии сообщества Apache 2 и коммерческой версии. Это интеллектуальная контекстно-зависимая среда IDE для работы с Java и другими языками JVM, такими как Kotlin, Scala и Groovy, во всех видах приложений. Кроме того, IntelliJ IDEA Ultimate может помочь вам в разработке веб-приложений с полным стеком, благодаря своим мощным интегрированным инструментам, поддержке JavaScript и смежных технологий, а также расширенной поддержке популярных фреймворков, таких как Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, Helidon.

**Spring Tool Suite (STS)** - это Java IDE, предназначенная для разработки корпоративных приложений на базе Spring. Это проще, быстрее и удобнее. И самое главное, оно основано на Eclipse IDE. STS является бесплатным, с открытым исходным кодом и поддерживается VMware. Spring Tools 4 - это следующее поколение инструментов Spring для любимой среды программирования. В значительной степени перестроенное с нуля, оно обеспечивает поддержку мирового уровня для разработки корпоративных приложений на основе Spring, независимо от того, предпочитаете ли вы Eclipse, Visual Studio Code или IDE.

**6. Требуется ли для запуска Spring Boot отдельный контейнер?**

Нет необходимости во встроенных контейнерах, таких как Tomcat / Jetty.

**7. Как настроить приложение Spring Boot с помощью maven?**

Уже имея структуру [проекта](#5-как-можно-запустить-spring-boot), следует установить [\*maven\*](https://maven.apache.org/download.cgi). В переменную окружения `path` необходимо добавить путь к каталогу `bin`. Для проверки следует протестировать maven, узнав ее версию

```

mvn -v

```

Тест пройден при отображении информации о Apache Maven

Для сборки maven необходимо создать файл `pom.xml` с дефолтным содержанием

```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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.springframework</groupId>

<artifactId>gs-maven</artifactId>

<packaging>jar</packaging>

<version>0.1.0</version>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-shade-plugin</artifactId>

<version>2.1</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>shade</goal>

</goals>

<configuration>

<transformers>

<transformer

implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

<mainClass>hello.HelloWorld</mainClass>

</transformer>

</transformers>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

```

Теперь с помощью maven можно собрать проект

```

mvn compile

mvn package

mvn install

```

Приложение настроено и готово к работе, но не стоит забывать о зависимостях, которые следует внедрять в файле `pom.xml`. Без их внедрения maven не сможет разобрать код программы и по этой причине не скомпилирует проект.

**8. Какие встроенные серверы поддерживает Spring Boot?**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Для Spring Boot Application вы можете сгенерировать jar приложения, который содержит Embedded Tomcat. Можно запустить веб-приложение как обычное Java-приложение | | | | | | | | |
| Под встроенным сервером подразумевается, что наш развертываемый модуль содержит двоичные файлы для сервера типа `tomcat.jar`. | | | | | | | | |
| ```xml | | | | | | | |
| <dependency> | | | | | | |
| <groupId>org.springframework.boot</groupId> | | | | | |
| <artifactId>spring-boot-starter-tomcat</artifactId> | | | | |
| <version>2.0.0.RELEASE</version> | | | |
| <scope>compile</scope> | | |
| </dependency> | |
|  |
| Несложно заметить, что по умолчанию Starter Web включает зависимость от tomcat. | | | | | | | | |

Tomcat является встроенным сервером по умолчанию для Spring Boot. Spring Boot также поддерживает Jetty и Undertow.

**9. Для чего используются аннотации @RestController и @Controller?**

Аннотация @RestController в Spring MVC – это не что иное, как сочетание аннотации @Controller и @ResponseBody . Он был добавлен в Spring 4.0 для упрощения разработки веб-служб RESTful в среде Spring. Если вы знакомы с [веб-службами REST,](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://www.java67.com/2017/04/3-great-books-to-learn-java-web-services-soap-and-restful.html&usg=ALkJrhiuYa6hlHB-SKbLhP7vbjmvzTKjUw) вы знаете, что принципиальное различие между веб-приложением и API-интерфейсом REST заключается в том, что ответ веб-приложения представляет собой общий вид HTML + CSS + JavaScript, тогда как API-интерфейс REST просто возвращает данные в форме JSON или XML. Эта разница также очевидна в аннотации @Controller и @RestController . Задача @Controller – создать карту объекта модели и найти представление, но @RestController просто возвращает объект, и данные объекта напрямую записываются в HTTP-ответ в виде JSON или XML.

Это также можно сделать с помощью традиционного @Controller и использования аннотации @ResponseBody но так как это стандартное поведение веб-сервисов RESTful, Spring представил @RestController который сочетал в себе поведение @Controller и @ResponseBody .

Вкратце, следующие два фрагмента кода в Spring MVC равны:

|  |
| --- |
| @Controller  @ResponseBody  public class MVCController {     .. your logic  } |
| @RestController  public class RestFulController {    .... your logic  } |

Очевидно, что все хотели бы объявить только одну аннотацию вместо двух. Кроме того, @RestController более очевиден, чем предыдущие два.

## Что такое @Controller и @RestController в Spring?

В среде Spring Controller – это класс, который отвечает за подготовку карты модели с данными, отображаемыми представлением, а также за выбор правильного представления. Он также может напрямую записывать в поток ответов с помощью аннотации @ResponseBody и завершать запрос.

Поведение записи непосредственно в поток ответов очень полезно для ответа на вызовы веб-сервисов RESTful, потому что там мы просто возвращаем данные вместо того, чтобы возвращать представление, как объяснялось в моем предыдущем посте о [внутренней работе Spring MVC](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=https://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html&usg=ALkJrhgqF55vuuOnoafA-fQR_3rzgWSB5w) .

Если вы разрабатывали веб-сервисы RESTful до Spring 4, например, в Spring 3 или Spring 3.1, вы были бы знакомы с использованием комбинации @Controller и @ResponseBody для создания ответа RESTful. Ребята из Spring @RestController эту @RestController и создали @RestController .

Теперь вам не нужно использовать аннотацию @Controller и @RestponseBody . Вместо этого вы можете использовать @RestController для обеспечения той же функциональности. Короче говоря, это удобный контроллер, который объединяет в себе поведение @Controler и тела @Response .

Вы также можете присоединиться к [REST](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://courses.baeldung.com/p/rest-with-spring-the-master-class%3Faffcode%3D22136_bkwjs9xa&usg=ALkJrhipTJRFnabT_6lptcJq57ZBz1Yp3Q) Eugen Paraschiv [с классом Spring Master,](https://translate.googleusercontent.com/translate_c?depth=1&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://courses.baeldung.com/p/rest-with-spring-the-master-class%3Faffcode%3D22136_bkwjs9xa&usg=ALkJrhipTJRFnabT_6lptcJq57ZBz1Yp3Q) если вам больше интересно изучать передовые методы разработки веб-службы RESTFul в Spring.

**10. Что такое Thymeleaf?**

|  |
| --- |
| Thymeleaf -это движок шаблонов Java XML / XHTML / HTML5 , который может работать как в вебе, так и в не веб-средах. Он лучше подходит для обслуживания XHTML / HTML5 на уровне представления веб-приложений на основе MVC, но он может обрабатывать любой XML-файл даже в автономной среде. Он обеспечивает полную интеграцию с Spring Framework. |
| В веб-приложениях Thymeleaf стремится полностью заменить JavaServer Pages (JSP) и реализует концепцию естественных шаблонов: файлов шаблонов, которые можно открывать непосредственно в браузерах и которые по-прежнему корректно отображаются в виде веб-страниц. |

Thymeleaf - это программное обеспечение с открытым исходным кодом, лицензированное по лицензии Apache License 2.0.