# ЛАБОРАТОРНАЯ РАБОТА №6 JAVA SPRING BOOT

## 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.

Hibernate — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения, самая популярная реализация спецификации JPA.

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

В JDBC *(*Java Database Connectivity) при написании каждого запроса необходимо указать в коде все детали, необходимые для операций CRUD, такие как имена таблиц, имена столбцов.

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

Можно рассматривать JPA как интерфейс, а Hibernate – как реализацию.

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

Перед работой требуется установить сам Tomcat. После создать проект на сайте spring Initializer со всеми необходимыми зависимостями.

После собрать проект в IDE. Описать структуру вашего проекта. Запустить сервер Tomcatи и проект, выбрав запуск через сервер Tomcat.

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

|  |  |
| --- | --- |
| Spring Boot | Spring MVC |
| **Spring Boot - это модуль Spring для упаковки приложения на основе Spring с разумными значениями по умолчанию.** | **Spring MVC - это веб-фреймворк на основе контроллера представления модели в рамках Spring framework.** |
| Он предоставляет конфигурации по умолчанию для создания фреймворка на базе Spring. | Он предоставляет готовые к использованию функции для создания веб-приложения. |
| Нет необходимости создавать конфигурацию вручную. Для этого требуется настройка сборки вручную. | Нет никаких требований к дескриптору развертывания. Требуется дескриптор развертывания. |
| Нет никаких требований к дескриптору развертывания. | Требуется дескриптор развертывания. |
| Это позволяет избежать шаблонного кода и объединяет зависимости в единое целое. | В нем указывается каждая зависимость отдельно. |
| Это сокращает время разработки и повышает производительность. | Для достижения того же требуется больше времени. |

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

jar и war – расширения платформо-независимых файлов-архивов Java-приложения. Все эти форматы – на самом деле просто zip-архивы с классами и другими частями приложения.

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. 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).

Они собираются утилитой jar из JDK.

**4. Зачем использовать Spring Boot?**

Spring Boot — это фреймворк на основе Java с открытым исходным кодом, разработанный компанией Pivotal Software. Благодаря быстродействию и простоте работы он стал популярным решением для создания развертываний в виде архива веб-приложений (WAR) и автономных Java-приложений.

Spring Boot выделяется среди других фреймворков, поскольку он предоставляет разработчикам программного обеспечения гибкую настройку, надежную пакетную обработку, эффективный рабочий процесс и большое количество инструментов, помогая разрабатывать надежные и масштабируемые приложения на базе Spring.

## 5. Как можно запустить Spring Boot?

Для настройки и запуска Spring Boot приложений требуется:

- Java 8+;

- Apache Maven 3.

При создании web-проекта с использованием Maven необходимо настроить Spring Boot в файле pom.xml.

Все приложения Spring Boot конфигурируются от spring-boot-starter-parent, поэтому перед дальнейшим определением зависимостей, в pom.xml нужно добавить starter-parent следующим образом (пример):

<parent>

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

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

<version>2.1.1.RELEASE</version>

</parent>

Для создания REST API 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:

<dependency>

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

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

</dependency>

После чего необходимые jar-библиотеки автоматически импортируются в ваш проект.

Следующий шаг ‒ добавление Spring Boot плагина:

<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>.

После создания ресурсов REST, необходимо создать класс конфигурации и запустить приложение. Spring Boot поддерживает новую аннотацию @SpringBootApplication, которая эквивалентна использованию @Configuration, @EnableAutoConfiguration и @ComponentScan с их атрибутами по умолчанию.

Таким образом, необходимо просто создать класс, аннотированный с помощью @SpringBootApplication, а Spring Boot включит автоматическую настройку и отсканирует ресурсы в текущем пакете.

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

Приложение может запускаться само, как исполняемый jar-файл с использованием встроенного сервера.

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

Уже имея структуру проекта, описанную в вопросе 5, следует установить maven. 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?

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 и @Controller?

В Spring Boot класс контроллера отвечает за обработку входящих запросов REST API, подготовку модели и возврат представления для отображения в качестве ответа.

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.

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 нужно просто возвращать данные вместо того, чтобы возвращать представление.

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. Теперь нет необходимости использовать аннотации @Controller и @RestponseBody одновременно. Вместо этого можно использовать @RestController для обеспечения той же функциональности.

## 10. Что такое Thymeleaf?

это механизм шаблонов Java XML/XHTML/HTML5, который может работать как в веб-среде, так и вне ее. Он лучше подходит для обслуживания XHTML/HTML5 на уровне представления веб-приложений на основе MVC, но может обрабатывать любой файл XML даже в автономной среде.

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.

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.