**ЛАБОРАТОРНАЯ РАБОТА No6  
JAVA SPRING BOOT**

**Общий вопрос: что такое JDBC, JPA и Hibernate? В чем их разница?**

Это три разных метода работы с БД, которые можно использовать в приложениях на Java, каждый из которых предоставляет больший уровень абстракции и автоматизации, чем предыдущий. (Т. е. JDBC – самый «низкий» уровень и, соответственно, требует больше кода, тогда как Hibernate – «высший» и предоставляет большую степень абстракции).

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. Спецификация Java Persistence API – это технология, которая позволяет удобно мапить объект Java и таблицу базы данных. В JPA (которая использует JDBC “под капотом”) также указываются эти данные, но уже только один раз, «навешиванием» аннотаций на Java класс. JPA – это спецификация, поэтому нужен инструмент для ее реализации. Этим инструментом может быть Hibernate. Hibernate − самый популярный ORM (Object Relational Mapping) фреймворк для работы с базой данных.

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

Одна из особенностей Spring Boot ‒ это встроенный сервер. Необходимо создать исполняемый jar-файл с помощью Maven и запустить его, как обычное автономное приложение. Для этого необходимо выполнить следующие шаги:

* Войдите в режим командной строки (команда cmd), перейдите в папку с pom.xml и введите команду «mvn clean package»;
* Maven cгенерирует исполняемый jar-файл;
* Перейдите в папку cd target;
* Затем запустите jar-файл: java -jar [имя файла];
* Перейдите в браузере по адресу на соответствующий адрес.

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

|  |  |
| --- | --- |
| **Spring MVC** | **Spring Boot** |
| Spring MVC — это веб-фреймворк на основе Model View и Controller, широко используемый для разработки веб-приложений. | Spring Boot построен на основе обычной среды Spring, широко используемой для разработки REST API. |
| Если мы используем Spring MVC, нам нужно создать конфигурацию вручную. | Если мы используем Spring Boot, нет необходимости создавать конфигурацию вручную. |
| В Spring MVC требуется дескриптор развертывания. | В Spring Boot нет необходимости в дескрипторе развертывания. |
| Spring MVC определяет каждую зависимость отдельно. | Он объединяет зависимости в единое целое. |
| Фреймворк Spring MVC состоит из четырех компонентов: Model, View, Controller и Front Controller. | В Spring Boot есть четыре основных уровня: уровень представления, уровень доступа к данным, уровень обслуживания и уровень интеграции. |
| Требуется больше времени на разработку. | Сокращает время разработки и повышает производительность. |
| Spring MVC не обеспечивает мощной пакетной обработки. | Spring Boot обеспечивает мощную пакетную обработку. |
| Готовая к использованию функция предоставляется им для создания веб-приложений. | Конфигурации по умолчанию предоставляются им для создания среды на базе Spring. |

**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 – **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.

**4. Зачем использовать Spring Boot?**

Spring Boot создан, чтобы помочь программистам ускорить процесс разработки. Он позволяет избавиться от трудоемкой первоначальной установки и настройки среды развертывания. Основные преимущества Spring Boot:

* Быстрая и легкая разработка приложений на основе Spring.
* Автоконфигурация всех компонентов для приложения Spring производственного уровня.
* Готовые встроенные серверы (Tomcat, Jetty и Undertow), обеспечивающие ускоренное и более продуктивное развертывание приложений.
* HTTP end-points, позволяющие вводить внутренние функции приложения, такие как показатели, состояние здоровья и другие.
* Отсутствие конфигурации XML.
* Огромный выбор плагинов, облегчающих работу разработчиков со встроенными базами данных и базами данных в памяти.
* Легкий доступ к базам данных и службам очередей, таким как MySQL, Oracle, MongoDB, Redis, ActiveMQ и другим.
* Плавная интеграция с экосистемой Spring.
* Большое сообщество и множество обучающих программ, облегчающих ознакомительный период.

**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?**

Разработчикам не надо беспокоиться о настройке контейнера сервлетов и развертывании приложения на нем. Приложение может запускаться само, как исполняемый jar-файл с использованием встроенного сервера.

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

[**См. вопрос 5**](#_Hlk119408850)

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)

**Наследование parent стартера**

Чтобы настроить проект для наследования от spring-boot-starter-parent, необходимо установить parent следующим образом (пример):

<parent>

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

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

<version>2.2.4.RELEASE</version>

</parent>

С помощью этой настройки также можно переопределить отдельные зависимости, переопределив свойство в собственном проекте. Например, чтобы перейти на другой релиз Spring Data, можно добавить следующее в 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. Может быть и собственный корпоративный стандартный parent, который нужно использовать, или можно 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 (пример):

<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, нужно добавить запись в dependencyManagement проекта перед записью spring-boot-dependencies. Например, чтобы перейти на другой релиз Spring Data, можно добавить следующий элемент в 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. Необходимо добавить 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.

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

Spring Boot поддерживает следующие встроенные контейнеры сервлетов:

|  |  |
| --- | --- |
| **Name** | **Servlet Version** |
| Tomcat 9.0 | 4.0 |
| Jetty 9.4 | 3.1 |
| Jetty 10.0 | 4.0 |
| Undertow 2.0 | 4.0 |

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

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?**

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.

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.

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

* HTML
* XML
* TEXT
* JAVASCRIPT
* CSS
* RAW

Итого: два markup фармата (HTML and XML), три текстовых формата (TEXT, JAVASCRIPT и CSS) и простой формат (RAW).