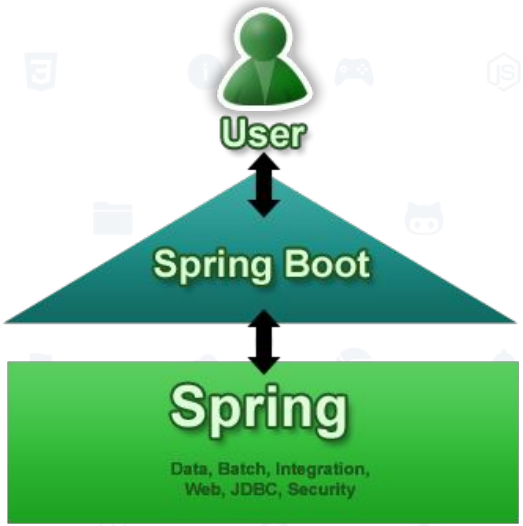
Lesson 34

23.02.2023

Что такое Spring Boot

Spring Boot — это дополнение к Spring, которое облегчает и ускоряет работу с ним. Сам Spring Boot представляет собой набор утилит, автоматизирующих настройки фреймворка. Вот что он берёт на себя:

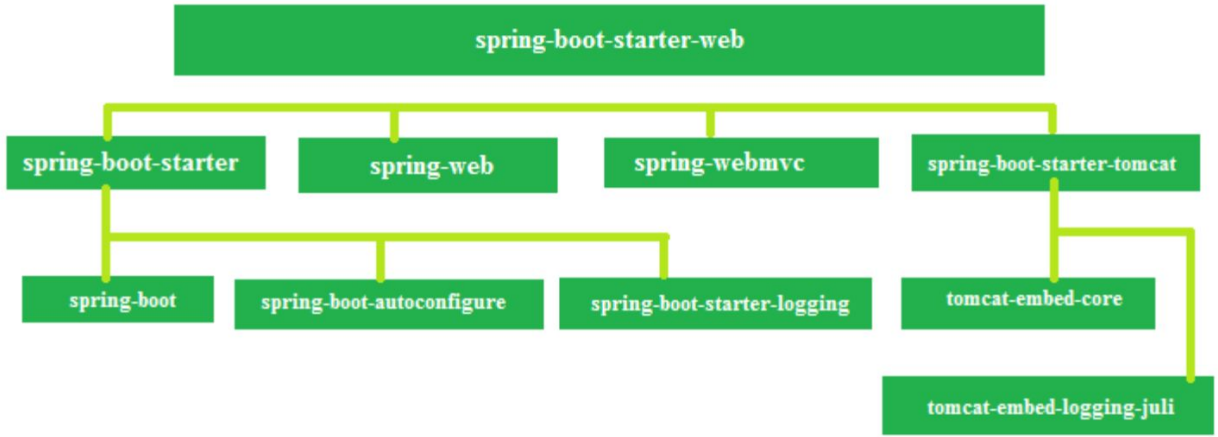
- упаковывает зависимости в starter-пакеты;
- автоматически конфигурирует приложения с помощью jar-зависимостей;
- создает веб-сервер, что позволяет локально запускать на нём приложения.



Простота управления зависимостями

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.).

Starter-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволит получить универсальное решение для всех, связанных со Spring технологий, избавляя программиста от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей (пример таких дескрипторов и стартовых пакетов будет показан ниже).



- `spring-boot-starter-web-services`: For building applications exposing SOAP web services
- `spring-boot-starter-web`: Build web applications and RESTful applications
- `spring-boot-starter-test`: Write great unit and integration tests
- `spring-boot-starter-jdbc`: Traditional JDBC applications
- `spring-boot-starter-hateoas`: Make your services more RESTful by adding HATEOAS features
- `spring-boot-starter-security`: Authentication and authorization using Spring Security
- `spring-boot-starter-data-jpa`: Spring Data JPA with Hibernate
- `spring-boot-starter-cache`: Enabling the Spring Framework's caching support
- `spring-boot-starter-data-rest`: Expose simple REST services using Spring Data REST

Автоматическая конфигурация

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.

Например, если вы добавите **Spring-boot-starter-web**, Spring Boot автоматически конфигурирует такие зарегистрированные бины, как **DispatcherServlet**, **ResourceHandlers**, **MessageSource.**

Если вы используете **spring-boot-starter-jdbc**, **Spring Boot** автоматически регистрирует бины **DataSource**, **EntityManagerFactory**, **TransactionManager** и считывает информацию для подключения к базе данных из файла **application.properties.**

Автоматическая конфигурация может быть полностью переопределена в любой момент с помощью пользовательских настроек.

Встроенная поддержка сервера приложений — контейнера сервлетов

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.

@SpringBootApplication : Содержит аннотации @ComponentScan, @Configuration и @EnableAutoConfiguration. Среди них @ComponentScan позволяет Spring Boot сканировать класс Configuration и добавлять его в контекст программы.

@Configuration Эквивалентно XML-файлу конфигурации Spring; используйте код Java для проверки безопасности типов.

@EnableAutoConfiguration Автоматическая настройка.

@ComponentScan Сканирование компонентов может автоматически обнаруживать и собирать некоторые компоненты.

@Component Его можно использовать с CommandLineRunner для выполнения некоторых основных задач после запуска программы.

@RestController Аннотация представляет собой набор @Controller и @ResponseBody, что означает, что это bean-компонент контроллера, а возвращаемое значение функции напрямую заполняется в теле ответа HTTP, который является контроллером в стиле REST.

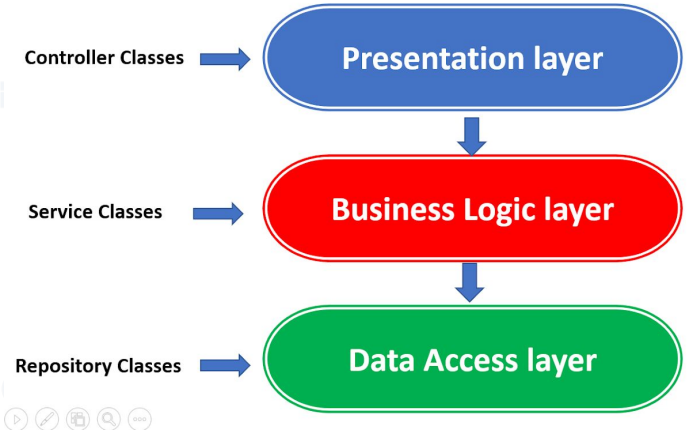
@Autowired Импортировать автоматически.

@PathVariable Получить параметры.

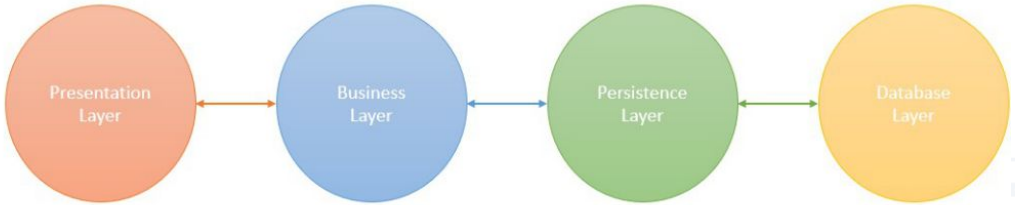
@RepositoryRestResourcepublic Используйте с spring-boot-starter-data-rest.

@ControllerAdvice : Содержат @Component. Можно сканировать. Единообразно обрабатывайте исключения..

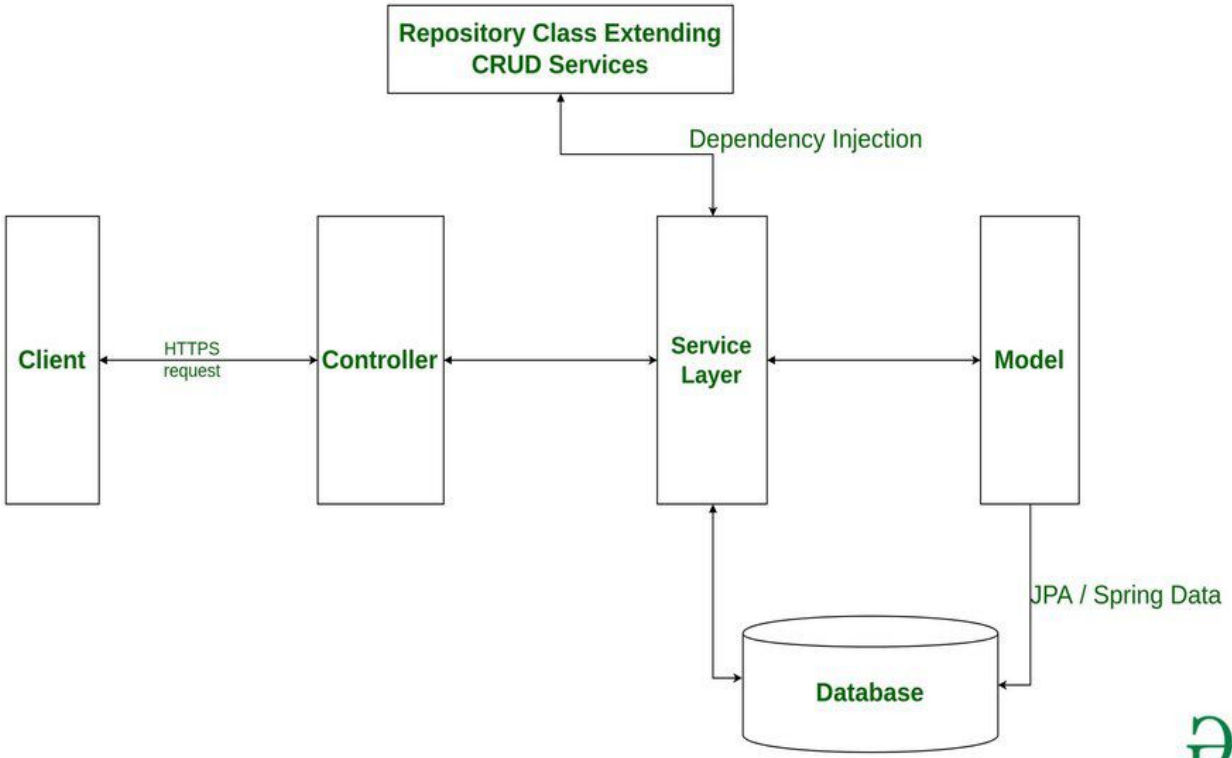
Three-Tier (or Three-Layer) Architecture in Spring MVC



1. **Presentation Layer** – Authentication & Json Translation
2. **Business Layer** – Business Logic, Validation & Authorization
3. **Persistence Layer** – Storage Logic
4. **Database Layer** – Actual Database



Spring Boot flow architecture



Explanation:

- The Client makes an **HTTP** request(GET, PUT, POST, etc.)
- The HTTP request is forwarded to the **Controller**. The controller maps the request. It processes the handles and calls the server logic.
- The business logic is performed in the **Service layer**. The spring boot performs all the logic over the data of the database which is mapped to the spring boot model class through **JPA** .
- The JSP page is returned as Response from the controller.