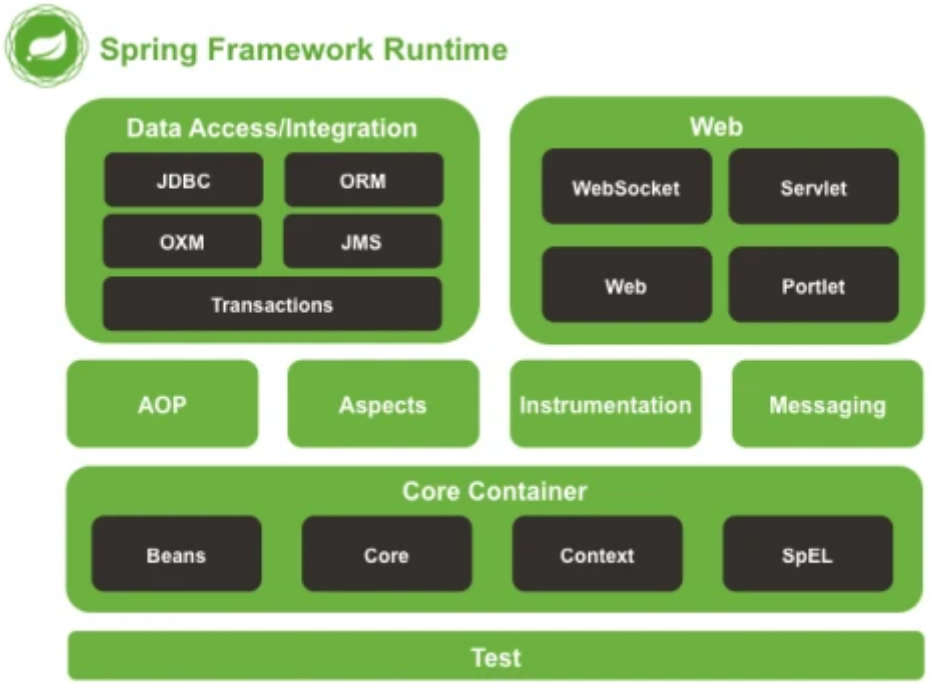
Шо це такое Spring?
Spring - это то, что делает Java разработчика не задротом, а умелым создатель веб-сервисов за 5 минут. Это самый популярный модульный фреймворк.




Dependency Injection, IoC (inversion of Control)

## DI, IoC

Dependency Injection - внедрение зависимостей.
IoC - инверсия контроля. Это такой своего рода философский паттерн, который означает, что мы передаём управление многими аспектами - фреймворку. Мы не храним бины, мы не создаём эти бины, мы не создаём контекст. ЭТО ДЕЛАЕТ ЦЕЛИКОМ И ПОЛНОСТЬЮ СПРИНГ. Единственное, мы можем чуть повлиять на эту историю.

## Bean, Context

Основные термины, связанные со Spring boot:
Bean - объекты класса. Вот этим и занимается спринг. Он создаётся объекты классов помеченных аннотациях @Component и иже с ним (Repository, Controller, Service)
Context - это хранение всех бинов из термина выше.

## Component

@Repository - указывает, что класс используется для работы с поиском, получением и хранением данных. Аннотация может использоваться для реализации шаблона DAO.
@Service - указывает, что класс является сервисом для реализации бизнес-логики.
@Repository, @Service, @Controller и @Configuration являются алиасами @Component, их также называют **стереотипными** аннотациями.
Единственная аннотация, которая отличная от @Component и не является алиасом - это @RestController (там есть какая-то доп. логика, но я никогда её не объяснял особо).

## Controller vs RestController

@Controller - специальный тип класса, обрабатывает HTTP-запросы и часто используется с аннотацией @RequestMapping.
@RestController ставится на класс-контроллер вместо @Controller. Она указывает, что этот класс оперирует не моделями, а данными. Она состоит из аннотаций @Controller и @ResponseBody. Была введена в Spring 4.0 для упрощения создания RESTful веб-сервисов.

@ResponseBody сообщает контроллеру, что возвращаемый объект автоматически сериализуется (используя Jackson message converter) в json или xml и передается обратно в объект HttpResponse.

ResponseEntity используется для формирования кастомизированного HTTP-ответа с пользовательскими параметрами (заголовки, код статуса и тело ответа). Во всех остальных случаях достаточно использовать @ResponseBody.
Если мы хотим использовать ResponseEntity, то просто должны вернуть его из метода, Spring позаботится обо всем остальном.
return ResponseEntity.status(213);

## BeanFactory vs ApplicationContext

ApplicationContext является наследником BeanFactory и полностью реализует его функционал, добавляя больше специфических enterprise-функций. Может работать с бинами всех скоупов.
BeanFactory - это фактический контейнер, который создает, настраивает и управляет рядом bean-компонентов. Эти бины обычно взаимодействуют друг с другом и, таким образом, имеют зависимости между собой. Эти зависимости отражены в данных конфигурации, используемых BeanFactory. Может работать с бинами singleton и prototype.

## Scope

Скоупы - это возможность управлять тем, как будут созданы бины. В основном используются 2 вида Singleton (по-умолчанию используется он) и Prototype. Но ещё

есть скоупы для HTTP-сессий (скажите на собесе про это - заебись, request, session, globalSession)
Назначаются скоупы вместе с аннотацией Component (или другими стереотипных аннотациями).
@Component
@Scope(“prototype”)
В записи вида Scope(“singleton”) - нет смысла, оно и так будет таковым.

| Scopes | Definition |
|---|---|
| singleton | This creates a single bean instance per Spring IoC container |
| prototype | This creates a new bean instance each time when requested |
| request | This creates a single bean instance per HTTP request. It is valid only in the context of a web application |
| session | This creates a single bean instance per HTTP session. It is valid only in the context of a web application |
| globalSession | This creates a single bean instance per global HTTP session. It is valid only in the context of a portal application |

МОЖНО СОЗДАВАТЬ КАСТОМНЫЕ СКОУПЫ, ЧТОБЫ ДОСТИГАТЬ КАКОГО-ТО НУЖНОГО ЭФФЕКТА. НО Я НИКОГДА НЕ СЛЫШАЛ, ЧТОБЫ ТАК ДЕЛАЛИ. В ТЕОРИИ ВОЗМОЖНО И ЕСЛИ ЭТО УПОМЯНУТЬ - ВАМ ПАЛЬЧИКИ НА НОЖКАХ НАЛИЖУТ СЛАДКО.

В этой статье подробно расписаны ключевые вещи экосистемы Spring`а (просто какие модули существуют и чё делают: https://habr.com/ru/articles/674858/)

## Из чего состаит спринг буте

Spring Boot — это инструмент, который упрощает разработку приложений на Java. Он состоит из нескольких компонентов:

- **Auto-configuratio**n. Автоматическая конфигурация позволяет Spring Boot автоматически настраивать компоненты приложения на основе зависимостей в classpath. Это избавляет разработчика от необходимости вручную настраивать приложение. Через application.yaml например
- **Starters**. Стартеры — это готовые наборы зависимостей, которые упрощают процесс начальной настройки приложения. Они включают в себя необходимые библиотеки и конфигурации для работы с различными технологиями. В pom.xml указывается спринговский парент, который будет тянуть все  нужные зависимости по-умолчанию (не абсолютно все, а те, что в экосистеме спринга)
- **Embedded servers**. Встроенный сервер позволяет запускать приложение без необходимости устанавливать и настраивать отдельный сервер. Spring Boot поддерживает встроенные версии популярных серверов, таких как Tomcat, Jetty

и другие. (Благодаря этому просто нажимаем кнопку стартовать, и если наша приложуха на спринг буте - она просто запускается на localhost:8080 или какой укажем и мы сразу можем слать туда запросы)

**Actuator**. Компонент Actuator предоставляет дополнительные возможности для мониторинга и управления приложением во время выполнения. Он включает в себя такие функции, как проверка состояния приложения, управление конечными точками и т. д. (Предоставляет клёвые метрики и отслеживает всякие хелсчеки
**Security**. Spring Boot предоставляет поддержку для настройки безопасности приложения, включая аутентификацию и авторизацию пользователей. (Не путай со Spring Security, в Spring Boot ВОЗМОЖНОСТИ БАЗОВОЙ НАСТРОЙКИ БЕЗОПАСНОСТИ, ОБЫЧНО ГОЛЫЙ СПРИНГ БУТОВСКИЙ НИКТО НЕ ЮЗАЕТ)
**Test support**. Поддержка тестирования позволяет разработчикам легко писать и запускать тесты для своего приложения. Spring Boot предоставляет инструменты для создания и запуска тестов, а также интеграции с популярными тестовыми фреймворками.