**1. Перечислите Spring модули и их назначение.**

**Spring Boot** — комплексный фреймворк для создания и запуска приложений с минимальными усилиями и настройками. Этот модуль делится на два стека: основанный на API сервлетов Spring MVC и реактивный Spring WebFlux.

**Spring Data**

Модуль обеспечивает приложениям доступ к данным через реляционные и нереляционные базы данных (БД), map-reduce фреймворки и облачные сервисы. Spring Data содержит множество подпроектов, предназначенных для определённых СУБД. Среди них есть, например, MySQL, MongoDB, Redis и многие другие.

**Spring Cloud**

Со Spring Cloud вы сможете легко и быстро создавать шаблоны в распределённых системах. Из примеров таких шаблонов: управление конфигурацией, обнаружение сервисов, интеллектуальная маршрутизация, микропрокси, одноразовые токены и многое другое.

**Spring Cloud Data Flow**

Spring Cloud Data Flow нужен приложениям, в которых используется потоковая передача и пакетная обработка данных.

**Spring Security**

Spring Security — среда аутентификации, авторизации и контроля доступа. Это стандартный фреймворк, который используется для защиты приложений на основе Spring.

**Spring Integration**

Spring Integration позволяет облегчить обмен сообщениями в приложениях на основе Spring, поддерживает интеграцию с внешними системами и даёт инструменты для обработки данных из разных источников.

**2. Расскажите о составе Spring Framework.**

Цель данного фреймворка – чтобы программист сосредоточился на решении бизнес-задачи вместо того, чтобы тратить время на настройку кода.

Сам по себе фреймворк состоит из модулей, которые можно подключать и настраивать тогда, когда они нужны. Напр. Spring Boot берет на себя вся работу с настройками и позволяет запускать веб приложения фактически в пару кликов.

Spring состоит из большого числа модулей. Среди них есть модули-стартеры, без которых Spring-приложение просто не запустится, а есть также вспомогательные проекты, добавляющие в приложение определённую функциональность

**3. Что такое Spring Boot? В чем его преимущества и для чего он используется?**

(Ответ выше)

**4. Для чего используется аннотация @SpringBootApplication?**

Этот класс аннотирован через @**SpringBootApplication.** Он выполняет автоматическую конфигурацию Spring, и автоматически сканирует (scan) весь проект, чтобы найти компоненты Spring (Controller, Bean, Service, ...)

**5-6. Объясните принцип IoC (Inversion of Control)? Какие формы используются в Spring для внедрения?**

Существует такое понятие как Inversion of Control, по-русски – Инверсия управления, сокращенно – IoC. IoC — один из принципов, приближающий наш код к слабосвязанности (каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов). IoC — это делегирование части наших обязанностей внешнему компоненту.

**7. В чем различие внедрение зависимостей (Dependency lnjection) и поиска зависимостей (Dependency Lookup)?**

Реализуется инверсия управления несколькими способами, среди которых есть внедрение зависимостей (Dependency Injection, DI) и поиск зависимостей (Dependency Lookup, DL).

Внедрение зависимостей реализуется несколькими способами, среди которых можно выделить:

Внедрение через конструктор

Внедрение через set-метод

Внедрение через интерфейс

В случае с поиском зависимостей класс должен самостоятельно реализовывать логику получения зависимостей извне. Для этого он должен иметь доступ к некоему источнику зависимостей.

**8. Что такое JavaBean? Какие есть правила описания и использования?**

**JavaBean** – **это** одноуровневые объекты, использующиеся для того, чтобы инкапсулировать в одном объекте код, данные или и то и другое. Компонент **JavaBean** может иметь свойства, методы и события, открытые для удаленного доступа.

**9. Перечислите области видимости bean.**

**Singleton** определяет один единственный бин для каждого контейнера Spring IoC (используется по умолчанию).  
**Prototype** позволяет иметь любое количество экземпляров бина.  
**request** создаётся один экземпляр бина на каждый HTTP запрос. Касается исключительно ApplicationContext.  
**session** создаётся один экземпляр бина на каждую HTTP сессию. Касается исключительно ApplicationContext.  
**global-session** создаётся один экземпляр бина на каждую глобальную HTTP сессию. Касается исключительно ApplicationContext.

**10. Опишите ЖЦ бина.**

Через следующие этапы проходит каждый отдельно взятый бин:

1. Инстанцирование объекта. Техническое начало жизни бина, работа конструктора его класса;

2. Установка свойств из конфигурации бина, внедрение зависимостей;

3. Нотификация aware-интерфейсов. BeanNameAware, BeanFactoryAware и другие. Мы уже писали о таких интерфейсах ранее. Технически, выполняется системными подтипами BeanPostProcessor, и совпадает с шагом 4;

4. Пре-инициализация – метод postProcessBeforeInitialization() интерфейса BeanPostProcessor;

5. Инициализация. Разные способы применяются в таком порядке:

• Метод бина с аннотацией @PostConstruct из стандарта JSR-250 (рекомендуемый способ);

• Метод afterPropertiesSet() бина под интерфейсом InitializingBean;

• Init-метод. Для отдельного бина его имя устанавливается в параметре определения initMethod. В xml-конфигурации можно установить для всех бинов сразу, с помощью default-init-method;

6. Пост-инициализация – метод postProcessAfterInitialization() интерфейса BeanPostProcessor.

**11. Поясните значения аннотаций: @Configuration, @Bean, @Component, @Service, @Repository, @Controller.**

@Configuration сообщает Spring что данный класс является конфигурационным, содержит определения и зависимости bean-компонентов.

Для определения бина используется аннотация @Bean.

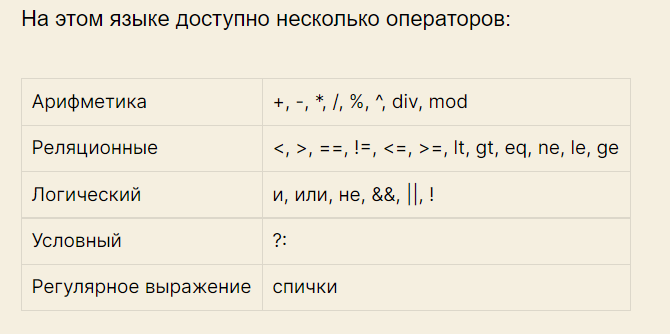
@Component – это аннотация, которая позволяет Spring автоматически определять ваши пользовательские компоненты.@service – сервис (тоже самое что компонент, просто для вида)

@repository

@controller – в mvc

**12. Spring Expression Language (SpEL): расскажите об особенности и области использования.**

Spring  [Expression Language](https://translate.googleusercontent.com/translate_c?depth=1&pto=aue&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://en.wikipedia.org/wiki/Unified_Expression_Language&usg=ALkJrhgJT-q4G7ZGjSza3o5L9xhOmpOycg" \o "Унифицированный язык выражений" \t "_blank)  (сокращенно SpEL) — это мощный язык выражений, который поддерживает запросы и манипулирование  [графом объектов](https://translate.googleusercontent.com/translate_c?depth=1&pto=aue&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://en.wikipedia.org/wiki/Object_graph&usg=ALkJrhhzWvJ_c45W1SsFGPLlIQK1u5klXQ)  во время выполнения, а также оценку логических и  [математических выражений](https://translate.googleusercontent.com/translate_c?depth=1&pto=aue&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://en.wikipedia.org/wiki/Expression_%2528mathematics%2529&usg=ALkJrhjyS-8seza46qcNy3lrFGmSQfNaIQ)  и многие другие функции. SpEL может использоваться независимо, независимо от того, использует ли ваша [среда приложения](https://translate.googleusercontent.com/translate_c?depth=1&pto=aue&rurl=translate.google.ru&sl=en&sp=nmt4&tl=ru&u=http://en.wikipedia.org/wiki/Application_framework&usg=ALkJrhinSHhtRS5F1DgrDJGgu4gI7-SSCw) Spring   или нет.

\

**13. Охарактеризуйте основные Core Container Spring.**

Core содержит несколько базовых классов и инструментов. Вся Spring Framework основан на Core контейнере.

Модуль Core содержит базовые классы Spring Framework, включая внедрение зависимостей (DI) и инверсию управления (IOC).

А так же бины, контекст и spel

**14. Как в Spring происходит разрешение зависимостей?**

порядке:

1) Создается и инициализируется ApplicationContext с метаданными конфигурации, которые описывают все бины. Эти метаданные могут быть описаны через XML, Java-код или аннотации

2) Для каждого бина и его зависимостей вычисляются свойства, аргументы конструктора. Эти зависимости предоставляются бину.

3) Каждому свойству или аргументу конструктора устанавливается значение или ссылка на другой бин в контейнере

4) Для каждого свойства или аргумента конструктора подставляемое значение конвертируется в тот формат, который указан для свойства или аргумента. По умолчанию Spring может конвертировать значения из строкового формата во все встроенные типы, такие как int, long, String, boolean и др.

**15. Поясните как работает DispatcherServlet, HandlerMapping, ViewResolver?**

Когда мы пишем в строке браузера запрос, его принимает ***Dispatcher Servlet***, далее он находит для обработки этого запроса подходящий контроллер с помощью ***HandlerMapping*** , затем диспетчер находит нужное представления по имени при помощи ***ViewResolver***'а, после чего на это представление передаются данные модели и на выход мы получаем нашу страничку.

**16. Как используется паттерн «Front Controller» в Spring?**

Паттерн Front Controller используется для обеспечения централизованного механизма обработки запросов, так что все запросы обрабатываются одним обработчиком. Этот обработчик может выполнить аутентификацию, авторизацию, регистрацию или отслеживание запроса, а затем передать запрос соответствующему контроллеру.

**17. Как происходит адресация в контроллере?**

В RequestMapping передаем тип метода и ссылку.

**18. Расскажите про Spring MVC архитектуру.**

Фреймворк **Spring MVC** обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними

* **Model** (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).
* **View** (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.
* **Controller** (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

**SpringBook application. Аннотирован через @SpringBootApplication. Тут автоматическая конфигурация spring.(+ищет все компоненты)**

**Application.properties строки приветствия и ошибки**

**Класс book – модель с двумя полями.**

**Класс bookForm представляет данные когда создастся новый бук**

**Контроллер – класс, обрабатывающий запросы.**

**@value – самый легкий способ добавления значений из конфигурации в код**