1. Перечислите Spring модули и их назначение.

**Spring Core Container**: Предоставляет основные функции IoC и DI, включая управление бинами и их зависимостями.

**Spring AOP (Aspect-Oriented Programming)**: Позволяет реализовывать аспектно-ориентированное программирование, что позволяет разделять логику, касающуюся поперечных аспектов (например, логирование, транзакции) от бизнес-логики.

**Spring Data Access/Integration**: Предоставляет абстракции для работы с базами данных, JMS (Java Messaging Service), JMX (Java Management Extensions) и т.д.

**Spring Web**: Обеспечивает интеграцию с веб-технологиями и предоставляет функции для разработки веб-приложений.

**Spring Test**: Предоставляет инструменты для тестирования Spring компонентов.

1. Расскажите о составе Spring Framework.

Состав Spring Framework включает в себя:

* **Spring Core Container**: Основной модуль, включающий **beans**, **core**, **context** и **expression language** (SpEL).
* **Spring AOP**: Поддержка аспектно-ориентированного программирования.
* **Spring Data Access/Integration**: Модули **jdbc**, **tx**, **orm**, **oxm**, **jms**, **web**, **aop**, **aspects**.
* **Spring Web**: Модули **web**, **webmvc**, **websocket**.
* **Spring Test**: Поддержка для тестирования с использованием JUnit или TestNG.

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

Spring Boot - это проект в экосистеме Spring, который облегчает создание, развертывание и поддержку приложений на основе Spring. Он предоставляет автоматизированные настройки и дефолтные значения для множества компонентов Spring, что позволяет быстро создавать рабочие приложения без необходимости вручную настраивать множество параметров.

Преимущества Spring Boot включают:

* Упрощенная конфигурация.
* Автоматическое управление зависимостями.
* Встроенные серверы приложений (например, Tomcat, Jetty).
* Простота создания и запуска приложения.
* Удобное управление внешними настройками.

Spring Boot используется для разработки самостоятельных, готовых к работе приложений с минимальными усилиями по конфигурации.

1. Для чего используется аннотация @SpringBootApplication?

Здесь @SpringBootApplication - составная аннотация, которая объединяет три другие аннотации Spring:

@SpringBootConfiguration - обозначает класс как класс конфигурации.

Эта аннотация фактически является специализированной формой аннотации @Configuration.

@ EnableAutoConfiguration - включает автоматическую настройку Spring Boot. Эта аннотация говорит Spring Boot автоматически настраивать любые компоненты, которые, по ее мнению, вам понадобятся.

@ ComponentScan - включает сканирование компонентов. Это позволяет объявлять другие классы с аннотациями, такими как @Component, @Controller, @Service и другие, чтобы Spring автоматически обнаруживал их и регистрировал как компоненты в контексте приложения Spring.

Она упрощает настройку и запуск spring-приложений.

1. Объясните принцип IoC (Inversion of Control)? Какие формы используются в Spring для внедрения?

Принцип IoC (Inversion of Control) в Spring означает, что контроль над созданием объектов и управлением их зависимостями переходит от приложения к контейнеру Spring. Вместо того, чтобы явно создавать и связывать объекты, приложение определяет свои зависимости, и контейнер Spring автоматически внедряет эти зависимости при создании объектов.

Формы внедрения зависимостей, используемые в Spring:

1. Конструктор (Constructor Injection): Зависимости внедряются через конструктор класса. Spring создает объекты, передавая им необходимые зависимости через конструктор.
2. Сеттер (Setter Injection): Зависимости внедряются через сеттеры или методы-инициализаторы. Spring вызывает соответствующие сеттеры и передает зависимости в объект.
3. Поле (Field Injection): Зависимости внедряются напрямую в поля класса. Spring использует рефлексию для прямого доступа к полям и внедрения зависимостей.
4. Аннотации (Annotation-based Injection): Зависимости внедряются на основе аннотаций, таких как @Autowired или @Resource. Аннотации указывают на зависимости, которые нужно внедрить, и Spring автоматически обеспечивает внедрение при создании объектов.
5. В чем суть понятия lnversion of Control (IoC)?

Ответ выше

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

Внедрение зависимостей (Dependency Injection):

В DI зависимости передаются объекту извне, обычно через конструктор, сеттеры или аннотации. Объект не самостоятельно создает или ищет свои зависимости, а полагается на контейнер или фреймворк, который обеспечивает инъекцию зависимостей.

Поиск зависимостей (Dependency Lookup):

В DL объект самостоятельно запрашивает зависимости у контейнера или фабрики, используя какой-то механизм поиска, такой как вызов метода или использование контекста приложения. Объект знает, где искать свои зависимости и самостоятельно получает ссылки на них.

1. Что такое JavaBean? Какие есть правила описания и использования?

JavaBean - это стандартный способ создания переносимых и повторно используемых компонентов в языке программирования Java. JavaBean представляет собой класс, который следует определенным правилам описания и использования.Основные правила описания и использования JavaBean:

1. Класс должен иметь открытый (public) конструктор без параметров.
2. Класс должен иметь приватные (private) поля, для которых предоставлены открытые (public) методы доступа (геттеры и сеттеры).
3. Класс должен реализовывать интерфейс Serializable, если объекты JavaBean могут быть сериализованы (например, для передачи по сети или сохранения в файл).
4. Имена геттеров и сеттеров должны соответствовать стандартным соглашениям JavaBeans, например, для поля "name" геттер будет называться "getName", а сеттер - "setName".
5. Класс должен быть открытым (public) и доступным для использования другими компонентами.
6. Перечислите области видимости bean.

Области видимости bean в Spring включают:  
**singleton** (один экземпляр для всего контейнера),

**prototype** (новый экземпляр при каждом запросе),

**request** (один экземпляр для каждого HTTP запроса),

**session** (один экземпляр для каждой HTTP сессии),

**application** (один экземпляр для всего приложения), и

**websocket** (один экземпляр для каждого WebSocket соединении).

1. Опишите ЖЦ бина.

Жизненный цикл бина в Spring включает в себя создание (instantiation), настройку (configuration), и уничтожение (destruction) объекта. Эти этапы зависят от области видимости бина и способа его создания.

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

**@Configuration**: Помечает класс как конфигурационный и позволяет определить бины с помощью методов, помеченных **@Bean**.

**@Bean**: Помечает метод как фабричный метод для создания и конфигурирования бина.

**@Component**: Общая аннотация для любого управляемого компонента в Spring.

**@Service**: Аннотация для сервисных компонентов, обычно используется в слое бизнес-логики.

**@Repository**: Аннотация для компонентов доступа к данным (DAO).

**@Controller**: Аннотация для компонентов, обрабатывающих HTTP-запросы в Spring MVC.

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

Spring Expression Language (SpEL) - это выразительный язык, который позволяет осуществлять манипуляции с данными внутри Spring приложения. Он используется для описания и обращения к свойствам объектов, выполнения операций и многого другого. SpEL может использоваться в аннотациях, XML-конфигурациях, а также во многих других местах

1. Охарактеризуйте основные Core Container Spring.

Основные Core Container Spring включают **beans**, **core**, **context** и **expression language** (SpEL). Эти модули предоставляют основу для создания и управления бинами в Spring-приложениях, управление компонентами, ресурсами и классами, внедрение зависимостей и выразительный язык SpEL.

1. Как в Spring происходит разрешение зависимостей?

Разрешение зависимостей в Spring происходит благодаря контейнеру, который анализирует зависимости бина и внедряет необходимые объекты или значения в соответствующие поля или методы

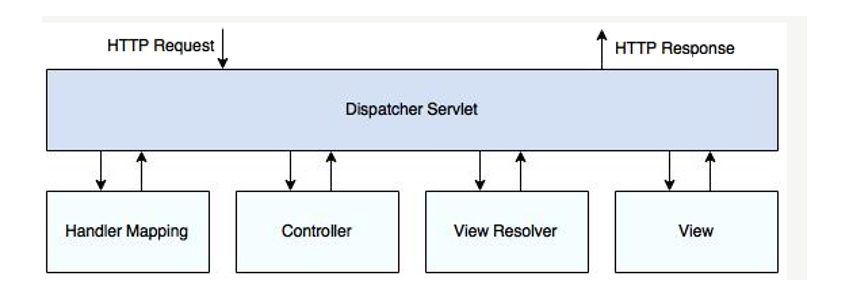
1. Поясните как работает DispatcherServlet, HandlerMapping, ViewResolver? Как происходит обработка запроса?

**DispatcherServlet**: Это фронт-контроллер Spring MVC, который принимает все запросы от клиента и направляет их к соответствующим обработчикам.

**HandlerMapping**: Этот компонент отображает запросы на соответствующие контроллеры (обработчики).

**ViewResolver**: Отвечает за поиск и разрешение представлений (шаблонов) для возвращаемых данных.

Обработка запроса начинается с DispatcherServlet, который использует HandlerMapping для определения соответствующего контроллера. Контроллер обрабатывает запрос и возвращает ModelAndView, который затем обрабатывается ViewResolver для генерации ответа.



1. Как используется паттерн  «Front Controller» в Spring?

В Spring паттерн "Front Controller" реализуется с помощью **DispatcherServlet**, который является центральной точкой входа для всех запросов. Он обрабатывает все запросы и направляет их к соответствующим обработчикам.

1. Как происходит адресация в контроллере?

В контроллере адресация происходит с помощью аннотаций, путем указания URL-шаблонов для методов контроллера с помощью аннотаций **@RequestMapping**, **@GetMapping**, **@PostMapping**, и т.д

1. Расскажите про Spring MVC архитектуру.

Spring MVC (Model-View-Controller) - это архитектурный шаблон, используемый для разработки веб-приложений. В нем модель представляет данные и бизнес-логику, вид (view) отвечает за отображение данных пользователю, а контроллер обрабатывает запросы пользователя и управляет взаимодействием между моделью и видом.

1. За что отвечает WebApplicationContext?

WebApplicationContext представляет собой специальный контекст приложения для веб-приложений. Он расширяет функциональность ApplicationContext и предоставляет дополнительные возможности, связанные с веб-приложениями, такие как доступ к сервлетам, слушателям и фильтрам