1. Вопрос: Перечислите Spring модули и их назначение.

Ответ: Spring Framework состоит из множества модулей, каждый из которых выполняет определенные задачи. Некоторые из основных модулей и их назначение:

- \*\*Spring Core Container\*\*: Обеспечивает основные функции IoC (Inversion of Control) и DI (Dependency Injection). Включает модули `beans`, `core`, `context`, и `spel`.

- \*\*Spring AOP (Aspect-Oriented Programming)\*\*: Предоставляет поддержку аспектно-ориентированного программирования, позволяя создавать аспекты для управления поперечными заботами, такими как логирование и транзакции.

- \*\*Spring Data Access/Integration\*\*: Обеспечивает поддержку для работы с базами данных и интеграцию с различными технологиями, такими как JDBC, ORM (Hibernate, JPA), JMS и другими.

- \*\*Spring Web\*\*: Предоставляет инфраструктуру для создания веб-приложений, включая модули для Spring MVC и WebSocket.

- \*\*Spring Security\*\*: Обеспечивает функциональность безопасности, включая аутентификацию, авторизацию и защиту от атак.

- \*\*Spring Test\*\*: Предоставляет инструменты для тестирования приложений на основе Spring.

2. Вопрос: Расскажите о составе Spring Framework.

Ответ: Состав Spring Framework включает в себя:

- \*\*Spring Core Container\*\*: Этот модуль включает в себя основные компоненты, такие как `BeanFactory`, `ApplicationContext`, и модуль `beans`, который предоставляет функциональность для создания и управления бинами (компонентами) приложения.

- \*\*Spring AOP\*\*: Этот модуль обеспечивает аспектно-ориентированное программирование, позволяя создавать аспекты для обработки поперечных забот, таких как логирование, безопасность и транзакции.

- \*\*Spring Data Access/Integration\*\*: Включает модули для работы с базами данных (JDBC, ORM) и интеграции с различными технологиями, такими как JMS и JCA.

- \*\*Spring Web\*\*: Предоставляет инфраструктуру для создания веб-приложений, включая Spring MVC для создания веб-слоя и WebSocket для обработки двунаправленной связи.

- \*\*Spring Security\*\*: Этот модуль предоставляет средства для обеспечения безопасности приложения, включая аутентификацию, авторизацию и другие механизмы безопасности.

- \*\*Spring Test\*\*: Предоставляет инструменты для тестирования Spring-приложений.

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

Ответ: Spring Boot - это проект, который упрощает разработку и настройку приложений на основе Spring. Его преимущества:

- Автоматическая настройка: Spring Boot автоматически конфигурирует множество аспектов приложения, что уменьшает необходимость в явной настройке.

- Упрощенное управление зависимостями: Spring Boot позволяет определить зависимости в файле `pom.xml` и автоматически управляет зависимостями.

- Встроенные серверы приложений: Spring Boot включает в себя встроенные серверы приложений, такие как Tomcat или Jetty, что упрощает развертывание приложений.

- Актуальные версии библиотек: Spring Boot предоставляет актуальные версии библиотек, что снижает риск несовместимости зависимостей.

- Удобство в создании микросервисов: Spring Boot отлично подходит для создания микросервисов и упрощает их конфигурирование и развертывание.

4. Вопрос: Для чего используется аннотация @SpringBootApplication?

Ответ: Аннотация `@SpringBootApplication` используется для указания главного класса (entry point) приложения Spring Boot. Она объединяет три аннотации в одной: `@Configuration`, `@EnableAutoConfiguration` и `@ComponentScan`. Это позволяет Spring Boot автоматически сканировать компоненты приложения, автоматически конфигурировать бины и включать настройки, и, таким образом, упрощает конфигурацию и запуск Spring Boot-приложения.

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

Ответ: Принцип IoC (Inversion of Control) представляет собой инверсию управления созданием и управлением объектами. Вместо того, чтобы приложение создавало и управляло своими зависимостями, IoC делегирует эти задачи контейнеру. Spring использует следующие формы внедрения (DI - Dependency Injection) для реализации IoC:

- Конструктор внедрения (Constructor Injection): Зависимости внедряются через конструктор класса.

- Setter-метод внедрения (Setter Injection): Зависимости внедряются через сеттеры (setter methods) класса.

- Внедрение через поля (Field Injection): Зависимости внедряются напрямую в поля класса.

- Внедрение через методы (Method Injection): Зависимости внедриваются через специальные методы класса.

- Внедрение через интерфейсы (Interface Injection): Зависимости внедриваются через методы интерфейсов.

6. Вопрос: В чем суть понятия lnversion of Control (IoC)?

Ответ: Суть понятия IoC (Inversion of Control) заключается в том, что контроль над жизненным циклом и управлением зависимостями объектов перекладывается с самого объекта на контейнер (или фреймворк). Вместо того, чтобы объекты создавать и настраивать в коде приложения, IoC делегирует эти задачи контейнеру. Это позволяет создавать более гибкие, расширяемые и тестируемые приложения, так как зависимости между компонентами становятся слабосвязанными и контейнер может управлять их жизненным циклом.

7. Вопрос: В чем различие внедрение зависимостей (Dependency Injection) и поиска зависимостей (Dependency Lookup)?

Ответ: Внедрение зависимостей (Dependency Injection, DI) и поиск зависимостей (Dependency Lookup, DL) - это два разных подхода к управлению зависимостями объектов.

- DI: При использовании DI зависимости внедряются в объект извне (например, через конструктор или сеттеры), и объект не знает, как и где создавать свои зависимости. Этот подход делает объект более независимым и уменьшает его связанность с другими объектами.

- DL: При использовании DL объект самостоятельно запрашивает (или "ищет") свои зависимости через некий механизм (например, сервисный локатор или контейнер служб). В этом случае объект знает, как создавать или получать свои зависимости.

Spring обычно использует DI для управления зависимостями.

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

Ответ: JavaBean - это простой Java-класс, который следует определенным соглашениям (JavaBeans Convention) для создания объектов, которые могут быть легко использованы и настроены в различных средах. Основные правила JavaBeans:

- Должен иметь публичный конструктор без аргументов (безаргументный конструктор).

- Должен предоставлять методы-геттеры и методы-сеттеры для доступа к своим полям (getters и setters).

- Должен быть сериализуемым, если предполагается его сохранение или передача по сети.

- Имена геттеров и сеттеров должны соответствовать соглашению JavaBeans (например, `getPropertyName` и `setPropertyName` для поля `propertyName`).

- Должен предоставлять стандартные методы `equals`, `hashCode` и `toString`.

9. Вопрос: Перечислите области видимости bean.

Ответ: В Spring Framework существуют четыре области видимости (scope) для bean-компонентов:

- Singleton (по умолчанию): Создается один экземпляр bean для всего контейнера Spring.

- Prototype: Создается новый экземпляр bean при каждом запросе.

- Request: Создается один экземпляр bean для каждого HTTP-запроса (только для веб-приложений).

- Session: Создается один экземпляр bean для каждой сессии пользователя (только для веб-приложений).

10. Вопрос: Опишите ЖЦ бина.

Ответ: Жизненный цикл (Lifecycle) бина в Spring охватывает несколько этапов, включая:

- \*\*Создание (Instantiation)\*\*: Бин создается, и ему могут быть переданы необходимые зависимости.

- \*\*Инициализация (Initialization)\*\*: Вызывается метод `@PostConstruct` (если он есть) и другие методы инициализации, если они указаны.

- \*\*Использование (In Use)\*\*: Бин находится в активном состоянии и может использоваться в приложении.

- \*\*Деактивация (Destruction)\*\*: Вызывается метод `@PreDestroy` (если он есть) и другие методы очистки, когда бин больше не нужен.

- \*\*Уничтожение (Destruction)\*\*: Бин уничтожается и освобождаются ресурсы.

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

Ответ: Эти аннотации используются для описания компонентов и их конфигураций в Spring:

- `@Configuration`: Обозначает класс как конфигурационный и позволяет определить бины и их настройки внутри класса. Класс с этой аннотацией может содержать методы, помеченные `@Bean`, для создания бинов.

- `@Bean`: Указывает метод для создания бина в конфигурационном классе. Метод должен возвращать объект, который станет бином в контейнере Spring.

- `@Component`: Обозначает класс как компонент Spring, который будет автоматически сканироваться и создаваться контейнером Spring. Общая аннотация для всех компонентов.

- `@Service`: Специализированная версия `@Component`, используется для обозначения классов, предоставляющих сервисы или бизнес-логику.

- `@Repository`: Специализированная версия `@Component`, используется для обозначения классов, работающих с базами данных или источниками данных.

- `@Controller`: Используется в Spring MVC для обозначения классов контроллеров, которые обрабатывают HTTP-запросы и управляют представлениями.

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

Ответ: Spring Expression Language (SpEL) - это язык выражений, предоставляемый Spring, для доступа к свойствам объектов и выполнения различных операций в конфигурационных файлах Spring и аннотациях. Особенности SpEL:

- Позволяет ссылаться на свойства бинов, окружения и другие объекты в конфигурационных файлах.

- Поддерживает арифметические, логические и реляционные операции.

- Позволяет вызывать методы на объектах.

- Может использоваться для фильтрации и преобразования коллекций данных.

- Поддерживается в аннотациях, таких как `@Value`, `@Conditional`, и других.

13. Вопрос: Охарактеризуйте основные Core Container Spring.

Ответ: Основные Core Container модули Spring включают:

- `beans`: Обеспечивает основные функции для создания и управления бинами (компонентами) приложения, включая поддержку DI и различные области видимости.

- `core`: Предоставляет основные классы и интерфейсы, такие как `ApplicationContext` и `BeanFactory`, для управления жизненным циклом бинов.

- `context`: Предоставляет возможности для доступа к объектам Spring, их конфигурации и взаимодействия между ними.

- `spel`: Предоставляет Spring Expression Language (SpEL) для оценки выражений в конфигурационных файлах.

14. Вопрос: Как в Spring происходит разрешение зависимостей?

Ответ: Разрешение зависимостей в Spring происходит путем инъекции (внедрения) зависимостей в бины (компоненты) во время создания контейнером Spring. Это может быть достигнуто через конструкторы (Constructor Injection), сеттеры (Setter Injection), поля (Field Injection) или методы (Method Injection). Spring осуществляет внедрение зависимостей на основе типов и имен, а также использует конфигурационные аннотации и XML-файлы для определения того, какие бины должны быть связаны между собой.

15. Вопрос: Поясните как работает DispatcherServlet, HandlerMapping, ViewResolver? Как происходит обработка запроса?

Ответ: В Spring MVC:

- `DispatcherServlet`: Это центральный сервлет, который принимает все входящие HTTP-запросы и управляет их обработкой. Он определяет, какой контроллер должен обработать запрос.

- `HandlerMapping`: Определяет, какой контроллер (handler) должен быть вызван для обработки запроса. HandlerMapping связывает URL-путь с соответствующим контроллером.

- `ViewResolver`: Определяет, какая представление (view) должна быть отображена для результата выполнения контроллера. ViewResolver находит имена представлений и преобразует их в конкретные представления (например, JSP или Thymeleaf шаблоны).

Обработка запроса происходит следующим образом:

1. `DispatcherServlet` получает HTTP-запрос.

2. `HandlerMapping` определяет, какой контроллер должен быть вызван для обработки запроса.

3. Вызывается соответствующий метод контроллера для обработки запроса.

4. Метод контроллера выполняет необходимую бизнес-логику и возвращает модель данных.

5. `DispatcherServlet` использует `ViewResolver` для нахождения представления, связанного с результатом выполнения контроллера.

6. Представление отображается, и модель данных передается для отображения в представлении.

7. Сформированный ответ отправляется клиенту.

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

Ответ: Паттерн "Front Controller" используется в Spring через `DispatcherServlet`. `DispatcherServlet` является центральным контроллером (front controller) в архитектуре Spring MVC. Он принимает все входящие HTTP-запросы и координирует их обработку.

17. Вопрос: Как происходит адресация в контроллере?

Ответ: В контроллерах Spring MVC адресация запросов обычно осуществляется через методы контроллера, которые ассоциированы с определенными URL-шаблонами (маршрутами). Методы контроллера могут быть аннотированы, чтобы указать, какие URL-шаблоны они обслуживают. Например:

```java

@Controller

public class MyController {

@RequestMapping("/home")

public String homePage() {

return "index"; // Возвращает имя представления "index

"

}

}

```

В этом примере метод `homePage()` будет обслуживать запросы, поступающие по адресу "/home".

18. Вопрос: Расскажите про Spring MVC архитектуру.

Ответ: Архитектура Spring MVC (Model-View-Controller) представляет собой способ организации веб-приложений в Spring Framework. Она включает следующие компоненты:

- \*\*Model (Модель)\*\*: Это компонент, который представляет бизнес-логику и данные приложения. Обычно представлен как Java-объекты или сервисы.

- \*\*View (Представление)\*\*: Это компонент, который отвечает за отображение данных пользователю. В Spring MVC, представления часто являются шаблонами (например, JSP, Thymeleaf) или фрагментами интерфейса пользователя.

- \*\*Controller (Контроллер)\*\*: Это компонент, который принимает HTTP-запросы от клиента, обрабатывает их, взаимодействует с моделью и выбирает представление для отображения результата. Контроллеры в Spring MVC ассоциируются с определенными URL-шаблонами.

Процесс обработки запроса в Spring MVC следующий:

1. HTTP-запрос поступает в `DispatcherServlet`.

2. `DispatcherServlet` определяет, какой контроллер (handler) должен обработать запрос, используя `HandlerMapping`.

3. Контроллер выполняет бизнес-логику и взаимодействует с моделью.

4. Контроллер выбирает представление (view), используя `ViewResolver`.

5. Данные модели передаются в представление, которое отображается клиенту.

19. Вопрос: За что отвечает WebApplicationContext?

Ответ: `WebApplicationContext` - это специализированный подкласс `ApplicationContext`, предназначенный для веб-приложений. Он отвечает за управление бинами (компонентами) и их жизненным циклом в контексте веб-приложения. `WebApplicationContext` предоставляет дополнительные функции, связанные с веб-приложениями, такие как доступ к объектам `HttpServletRequest` и `HttpSession`, а также обработка файловых ресурсов и представлений.

Каждое веб-приложение, созданное с использованием Spring, имеет свой собственный `WebApplicationContext`, который управляется `DispatcherServlet`. Это позволяет изолировать бины и их конфигурации между разными веб-приложениями на одном сервере.