1. **Spring как семейство проектов**: Spring Framework - это мощный и гибкий фреймворк, ориентированный на создание приложений на Java. Он включает в себя несколько модулей, таких как Spring MVC для веб-приложений, Spring Data для доступа к данным и Spring Security для аутентификации и авторизации.
2. **Жизненный цикл запроса в Spring MVC**: Запрос в Spring MVC проходит несколько этапов, включая прием, диспетчеризацию, обработку и, наконец, генерацию ответа. DispatcherServlet отвечает за маршрутизацию запроса к соответствующему контроллеру.
3. **Архитектура Spring MVC**: Spring MVC использует шаблон проектирования Front Controller, где DispatcherServlet выступает в роли Front Controller. Создание контроллера в Spring MVC обычно включает определение класса с аннотацией @Controller и методов, обрабатывающих запросы.
4. **Конфигурация Spring**: Spring предоставляет различные способы конфигурации, включая XML, Java Config и аннотации. WebMvcConfigurer используется для настройки Spring MVC. Аннотации @Controller, @Repository и @Service используются для определения компонентов в различных слоях приложения.
5. **Адресация в контроллере**: Аннотации @RequestMapping и @GetMapping используются для маппинга HTTP-запросов на методы контроллера.
6. **Понятие Inversion of Control (IoC) и Dependency Injection (DI)**: IoC - это принцип, при котором контроль над выполнением программы передается фреймворку. DI - это форма IoC, при которой зависимости передаются в объекты.
7. **JavaBean**: JavaBean - это класс в Java, который следует определенным соглашениям, включая наличие конструктора без аргументов и возможность получения и установки значений свойств. В Spring аннотации @Autowired, @Primary, @Qualifier и @Inject используются для автоматического связывания зависимостей.
8. **Жизненный цикл Bean в Spring**: Spring управляет жизненным циклом бинов, включая их создание, инициализацию, использование и уничтожение. @ComponentScan используется для автоматического обнаружения компонентов.
9. **Spring Expression Language (SpEL)**: SpEL - это язык выражений Spring, который предоставляет поддержку для запросов и манипуляций с объектами во время выполнения.
10. **Валидация в Spring Framework**: Spring предоставляет интерфейс Validator для валидации объектов.
11. **Правила валидации и ограничения**: Валидация в Spring может включать в себя проверку различных ограничений, таких как обязательные поля, максимальная и минимальная длина, форматы и т.д.
12. **Создание пользовательского валидатора**: Spring позволяет создавать пользовательские валидаторы для выполнения сложной валидации.
13. **Понятие ORM**: ORM (Object-Relational Mapping) - это техника, которая позволяет манипулировать данными из базы данных с использованием объектно-ориентированного подхода. JPA (Java Persistence API) - это стандарт Java для ORM. Он включает в себя EntityManager, Persistence и другие компоненты.
14. **Требования к Entity**: Entity - это класс, который представляет таблицу в базе данных. У него есть жизненный цикл и он может участвовать в различных типах связей, таких как один-ко-многим, многие-ко-многим и т.д.
15. **Аннотации Spring Data**: Spring Data использует аннотации для упрощения работы с базами данных. Это включает в себя аннотации для определения репозиториев, запросов, транзакций и т.д.
16. **JPA callback mechanism (@Pre… @Post…)**: JPA предоставляет механизм обратных вызовов, который позволяет выполнять определенные действия перед или после определенных событий, таких как сохранение, обновление или удаление Entity.
17. **Pattern Service, Repository, Controller**: Это общий шаблон проектирования в Spring, где Service обрабатывает бизнес-логику, Repository обеспечивает доступ к данным, а Controller обрабатывает HTTP-запросы.
18. **Аспектно-ориентированное программирование**: Аспектно-ориентированное программирование (AOP) - это подход, который позволяет модульно организовывать код, связанный с перекрестными заботами. В Spring AOP используются аспекты, советы, срезы и точки соединения.
19. **Архитектура AOP в Spring**: Spring AOP использует ProxyFactory для создания прокси-объектов, которые обрабатывают аспекты. Spring поддерживает различные фреймворки AOP.
20. **Конфигурации Spring AOP**: Spring AOP использует аннотации для определения аспектов и советов. Это включает в себя аннотации @Pointcut, @Before, @AfterReturning, @Around и другие.
21. **Понятие SPA и MPA приложений**: SPA (Single Page Application) - это веб-приложение, которое загружает всю необходимую HTML, CSS и JavaScript при первой загрузке страницы. MPA (Multi Page Application) - это традиционное веб-приложение, которое загружает новую страницу при каждом действии пользователя.
22. **Entity – DTO конвертация**: Entity и DTO (Data Transfer Object) - это два разных объекта, которые используются для разных целей. Entity используется для взаимодействия с базой данных, в то время как DTO используется для передачи данных между слоями и/или через сеть. Модель Mapper используется для конвертации между Entity и DTO.
23. **Понятие REST**: REST (Representational State Transfer) - это архитектурный стиль для создания веб-служб. Он использует стандартные HTTP-методы, такие как GET, POST, PUT, DELETE и т.д.
24. **HTTP-методы REST**: REST использует стандартные HTTP-методы для взаимодействия с ресурсами. Это включает в себя GET для получения данных, POST для отправки данных, PUT для обновления данных, DELETE для удаления данных и т.д.
25. **REST контроллер**: REST контроллер в Spring обрабатывает HTTP-запросы и генерирует HTTP-ответы. Он может использовать различные аннотации для маппинга запросов, обработки параметров запроса и формирования ответа.
26. **Отображение кодов ответа HTTP**: Spring позволяет настраивать коды ответа HTTP для REST-контроллеров. Это может включать в себя стандартные коды ответа, такие как 200 (OK), 404 (Not Found), 500 (Internal Server Error) и т.д.
27. **Настраиваемые исключения при ошибках запроса REST**: Spring предоставляет механизмы для обработки исключений в REST-контроллерах. Это может включать в себя определение глобального обработчика исключений или аннотацию @ExceptionHandler для обработки конкретных исключений.
28. **Тестирование REST**: Для тестирования REST-сервисов можно использовать различные инструменты, такие как Postman, которые позволяют отправлять HTTP-запросы и просматривать ответы.
29. **Понятие HATEOAS REST сервиса**: HATEOAS (Hypermedia as the Engine of Application State) - это принцип, согласно которому клиент взаимодействует с сервером исключительно через гипермедиа, предоставляемую сервером.
30. **Документирование REST на основе Open API**: Open API - это спецификация для документирования REST-сервисов. Она позволяет автоматически генерировать документацию на основе кода и аннотаций.
31. **Spring Security Framework**: Spring Security - это мощный фреймворк для обеспечения безопасности в приложениях Spring. Он предоставляет поддержку для аутентификации, авторизации, защиты от атак CSRF и многого другого.
32. **Аутентификация и авторизация**: Аутентификация - это процесс проверки идентификации пользователя, в то время как авторизация - это процесс проверки прав доступа пользователя.
33. **Перехват запросов**: Spring Security предоставляет механизмы для перехвата и обработки запросов на основе различных критериев, таких как роли пользователей, IP-адреса и т.д.
34. **Поддержка безопасности в Spring Security на уровне метода**: Spring Security предоставляет возможность защищать доступ к методам на основе аннотаций.
35. **Настройка Spring Security для OAuth 2.0 Login и Resource Server**: Spring Security поддерживает OAuth 2.0, что позволяет легко интегрировать внешние системы аутентификации и авторизации.
36. **Spring Cloud**: Spring Cloud - это набор инструментов и библиотек для построения облачных приложений и микросервисов. Он предоставляет поддержку для централизованной конфигурации, обнаружения сервисов, маршрутизации, обработки сбоев и многого другого.