Вопросы и ответы на собеседование по Spring

(46-60)

46). Для чего используется аннотация @Bean?

В классах конфигурации Spring, @Bean используется для определения компонентов с кастомной логикой.

47). В чём разница между @Bean и @Component?

@Bean используется в конфигурационных классах Spring. Он используется для непосредственного создания бина.

@Component используется со всеми классами, которыми должен управлять Spring. Когда Spring видит класс с @Component, Spring определяет этот класс как кандидата для создания bean.

48). Можем ли мы использовать @Component вместо @Service для бизнес логики?

Да. конечно.

Если **@Component** является универсальным стереотипом для любого Spring компонента, то **@Service** в настоящее время является его псевдонимом. Однако, в официальной документации Spring рекомендуется использовать именно @Service для бизнес логики.
Вполне возможно, что в будущих версиях фреймворка, для данного стереотипа добавится дополнительная семантика, и его бины станут обладать дополнительной логикой.

49). В чем различие между web.xml и servlet.xml?

web.xml — Метаданные и конфигурация любого веб-приложения, совместимого с Java EE. Java EE стандарт для веб-приложений.

servlet.xml — файл конфигурации, специфичный для Spring Framework.

50). Что предпочитаете использовать для конфигурации Spring – xml или аннотирование?

Предпочитаю аннотации, если кодовая база хорошо описывается такими элементами, как @Service, @Component, @Autowired.

Однако когда дело доходит до конфигурации, у меня нет каких-либо предпочтений. Я бы оставил этот вопрос команде с которой я буду работать.

51). Можем ли мы применить @Autowired с не сеттерами и не конструкторами методами?

Да, конечно.

@Autowired может использоваться вместе с конструкторами, сеттерами или любыми другими методами. Когда Spring находит @Autowired на методе, Spring автоматически вызовет этот метод, после создания экземпляра бина. В качестве аргументов, будут подобраны подходящие объекты из контекста Spring.

52). В чем разница между Сквозной Функциональностью (Cross Cutting Concerns) и АОП (аспектно оринтированное программирование)?

Сквозная Функциональность — функциональность, которая может потребоваться вам на нескольких различных уровнях — логирование, управление производительностью, безопасность и т.д.
АОП — один из подходов к реализации данной проблемы.

53). В чем разница между IOC (Inversion of Control) и Application Context?

IOC — инверсия управления. Вместо ручного внедрения зависимостей, фреймворк забирает ответственность за это.
ApplicationContext — реализация IOC спрингом.

Bean Factory — это базовая версия IOC контейнера

Application Context также включает дополнительные функции, которые обычно нужны для разработки корпоративных приложений.

54). В чем разница между classPathXmlApplicationContext и annotationConfigApplicationContext?

classPathXmlApplicationContext — если вы хотите инициализировать контекст Spring при помощи xml
annotationConfigApplicationContext — если вы хотите инициализировать контекст Spring при помощи конфигурационного класса java.

55). Почему возвращаемое значение при применении аспекта @Around может потеряться? Назовите причины.

Метод, помеченный аннотацией @Around, должен возвращать значение, которое он (метод) получил из joinpoint.proceed()

```
@Around("trackTimeAnnotation()")
public Object around(ProceedingJoinPoint joinPoint) throwsThrowable{
    long startTime = System.currentTimeMillis();
    Object retVal = joinPoint.proceed();
    long timeTaken = System.currentTimeMillis() - startTime;
    logger.info("Time taken by {} is equal to {}",joinPoint, timeTaken);
    return retVal;
}
```

56). Как вы решаете какой бин инжектить, если у вас несколько подходящих бинов. Расскажите о @Primary и @Qualifier?

Если есть бин, который вы предпочитаете большую часть времени по сравнению с другими, то используйте **@Primary**, и используйте **@Qualifier** для нестандартных сценариев. Когда Spring обнаруживает неоднозначность при инжекте бина, он выбирает тот бин, который помечен аннотацией @Primary.

Если все бины имеют одинаковый приоритет, мы всегда будем использовать @Qualifier.
@Qualifier - это аннотация, которую вы можете использовать для явного указания имени или значения квалификатора для бина, который должен быть инжектирован.

Если бин надо выбрать во время исполнения программы, то эти аннотации вам не подойдут. Вам надо в конфигурационном классе создать метод, пометить его аннотацией **@Bean**, и вернуть им требуемый бин.

57). В чём разница между @Controller и @RestController?

@RestController = @Controller + @ResponseBody

@RestController превращает помеченный класс в Spring-бин. Этот бин для конвертации входящих/исходящих данных использует **Jackson message converter**. Как правило целевые данные представлены в json или xml.

58). Почему иногда мы используем @ResponseBody, а иногда ResponseEntity?

ResponseEntity необходим, только если мы хотим кастомизировать ответ, добавив к нему **статус ответа**. Во всех остальных случаях будем использовать @ResponseBody.

```
@GetMapping(value=”/resource”)
@ResponseBody
public Resource sayHello() { return resource; }

@PostMapping(value=”/resource”)
public ResponseEntity createResource() {
    ....
return ResponseEntity.created(resource).build();
}
```

Стандартные HTTP коды статусов ответов, которые можно использовать.

200 — SUCCESS
201 — CREATED
404 — RESOURCE NOT FOUND
400 — BAD REQUEST
401 — UNAUTHORIZED
500 — SERVER ERROR

Для @ResponseBody единственные состояния статуса это SUCCESS(200), если всё ок и SERVER ERROR(500), если произошла какая-либо ошибка.

Допустим мы что-то создали и хотим отправить статус CREATED(201). В этом случае мы используем ResponseEntity.

59). В чем разница между Filters, Listeners и Interceptors?

Концептуально всё просто, фильтры сервлетов могут перехватывать только **HTTPServlets**. Listeners могут перехватывать специфические **события**. Как перехватить события которые относятся ни к тем не другим?

Фильтры и перехватчики делают по сути одно и тоже: они перехватывают какое-то событие, и делают что-то до или после.

Java EE использует термин Filter, Spring называет их Interceptors.

Именно здесь AOP используется в **полную силу,** благодаря чему возможно перехватывание вызовов любых объектов.

60). Как работает аннотация @RequestMapping ?

Аннотация @**RequestMapping** используется для сопоставления веб-запросов с методами Spring Controller. Помимо простых вариантов использования, мы можем использовать его для сопоставления заголовков HTTP, привязки частей URI с помощью @**PathVariable** и работы с параметрами URI и аннотацией @**RequestParam** .