***ТОП 50 Аннотация Spring-boot***

**@SpringBootApplication:**

* Объединяет аннотации @Configuration, @EnableAutoConfiguration, и @ComponentScan.
* Используется для указания, что класс является основным классом Spring Boot.

**@Controller:**

* Обозначает класс как контроллер в архитектуре MVC.

**@RestController:**

* Объединяет @Controller и @ResponseBody.
* Используется для создания RESTful контроллеров.

**@RequestMapping:**

* Определяет маппинг URL к методу контроллера.

**@GetMapping, @PostMapping, @PutMapping, @DeleteMapping:**

* Аннотации для определения методов контроллера для HTTP-запросов GET, POST, PUT, DELETE соответственно.

**@RequestParam:**

* Используется для извлечения параметра запроса из URL.

**@PathVariable:**

* Используется для извлечения переменной из URI шаблона в маппинге URL.

**@RequestBody:**

* Используется для привязки тела запроса к параметру метода контроллера.

**@ResponseStatus:**

* Устанавливает HTTP-статус ответа для метода контроллера.

**@Autowired:**

* Автоматическое внедрение зависимостей (DI).

**@Qualifier:**

* Используется вместе с @Autowired для указания конкретной реализации бина.

**@Value:**

* Внедрение значения из внешних свойств в поля.

**@Configuration:**

* Объявляет класс как конфигурационный для Spring.

**@ComponentScan:**

* Сканирует пакеты для поиска и регистрации бинов.

**@Bean:**

* Объявляет метод как фабричный метод для создания бина.

**@Profile:**

* Определяет, при каких условиях компонент должен быть включен в контекст.

**@Conditional:**

* Условное включение компонентов в зависимости от выполнения определенного условия.

**@EnableAutoConfiguration:**

* Автоматическая настройка приложения на основе наличия классов в classpath.

**@EnableWebMvc:**

* Включает конфигурацию Spring MVC.

**@EnableTransactionManagement:**

* Включает управление транзакциями в Spring.

**@Transactional:**

* Определяет транзакционное поведение метода или класса.

**@EnableCaching:**

* Включает поддержку кэширования в приложении.

**@Cacheable, @CacheEvict, @CachePut:**

* Аннотации для работы с кэшем.

**@Scheduled:**

* Планирует выполнение метода по расписанию.

**@Async:**

* Определяет метод, который должен выполняться асинхронно.

**@EnableScheduling:**

* Включает поддержку планировщика задач.

**@EnableAsync:**

* Включает поддержку асинхронных методов.

**@ExceptionHandler:**

* Обрабатывает исключения в методах контроллера.

**@ControllerAdvice:**

* Глобальный контроллер для обработки исключений в приложении.

**@ResponseStatus:**

* Устанавливает HTTP-статус для исключения.

**@EnableJpaRepositories:**

* Включает поддержку репозиториев JPA.

**@Entity:**

* Обозначает класс как сущность JPA.

**@Table:**

* Определяет маппинг класса на таблицу в базе данных.

**@Repository:**

* Аннотация для маркировки класса как репозиторий.

**@TransactionalEventListener:**

* Обработчик событий внутри транзакции.

**@ConfigurationProperties:**

* Привязывает свойства из файла конфигурации к полям бина.

**@EnableConfigurationProperties:**

* Включает поддержку свойств конфигурации.

**@EnableJms:**

* Включает поддержку Java Message Service (JMS).

**@JmsListener:**

* Обозначает метод как слушателя сообщений JMS.

**@Retryable:**

* Позволяет методу автоматически повторяться в случае исключения.

**@ConditionalOnProperty:**

* Условие наличия свойства в файле конфигурации.

**@ConditionalOnClass:**

* Условие наличия класса в classpath.

**@ConditionalOnMissingBean:**

* Условие выполнения, если бина с указанным типом отсутствует.

**@ConditionalOnExpression:**

* Условие выполнения, основанное на выражении SpEL.

**@RequestMappingHandlerMapping:**

* Кастомизация маппинга URL для методов контроллера.

**@RequestMappingHandlerAdapter:**

* Кастомизация обработки методов контроллера.

**@Primary:**

* Помечает бин как основной, если есть несколько кандидатов.