1、JAVA中，池化是什么意思，为什么要用池化这个技术？有什么好处？

在 Java 中，**池化（Pooling）** 是一种通过**预先创建并管理一组可重用资源（对象、线程、数据库连接等）来优化性能和资源利用的技术。其核心思想是避免频繁创建和销毁资源**，而是通过复用已有资源来减少系统开销。

为什么要用池化技术？**（资源统一管理，有序且减少了创建销毁的开销）**

1、**资源创建和销毁成本高**：某些资源（如线程、数据库连接、网络连接）的创建和销毁需要消耗大量时间和系统资源。例如：线程创建涉及操作系统层面的资源分配。数据库连接需要经历网络握手、身份验证等过程。对象频繁创建会触发垃圾回收（GC），影响性能。

2、**资源管理的复杂性**：直接管理资源的生命周期容易导致资源泄漏（如忘记关闭连接）或竞争问题。池化技术通过统一管理资源，简化了开发流程。

3、**应对高并发场景：**在高并发下，频繁创建资源可能导致系统过载（如线程过多引发内存耗尽）。池化通过限制资源数量，确保系统稳定。（以及线程安全）

池化的好处？

1、**性能提升：减少重复开销**：复用资源避免重复初始化（如数据库连接的 TCP 握手）。**降低 GC 压力**：对象池（如 ObjectPool）减少短生命周期对象的创建，从而减少垃圾回收频率

**2、资源可控性**：通过设置池的最大容量（如线程池的 maxPoolSize），防止资源耗尽导致系统崩溃。支持动态调整资源数量，适应负载变化。

**3、响应速度优化**：池中的资源处于“就绪状态”，可直接使用，避免了资源初始化的等待时间（如直接从连接池获取连接）。

**4、统一管理和监控：**池化框架（如 HikariCP、Apache Commons Pool）提供监控接口，可跟踪资源使用情况。支持超时、重试等策略，增强系统健壮性。

潜在问题

1、**配置复杂性：**需合理设置池的大小（如线程池的 corePoolSize 和 maxPoolSize），否则可能导致资源浪费或饥饿。

**2、资源泄漏：**若未正确释放资源（如忘记关闭数据库连接），会导致池中资源耗尽。

**3、状态一致性：**对象池中的对象需重置状态后再复用，避免脏数据（如缓存未清理的临时数据）

2、什么是JAVA中的单元测试？都有哪些？都有什么相关的注解？

在 Java 中，**单元测试（Unit Testing）** 是针对代码中最小可测试单元（如类、方法）的验证过程，目的是确保每个独立单元的行为符合预期。单元测试通常由开发者编写，并借助测试框架（如 JUnit、TestNG）实现自动化和可重复性

Java 中常见的单元测试框架

**1、Junit：**最主流的单元测试框架，广泛支持 Java 项目，与构建工具（Maven、Gradle）和 IDE（IntelliJ、Eclipse）深度集成。最新版本为 JUnit 5（包含 JUnit Jupiter、JUnit Vintage 等模块），旧版本为 JUnit 4。

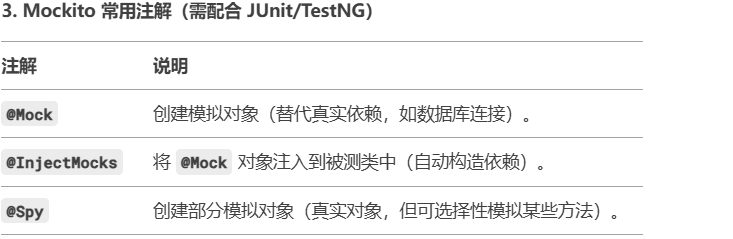
**2、TestNG：**功能更丰富的测试框架，支持依赖测试、分组测试、参数化测试等高级特性，常用于复杂场景。语法类似 JUnit，但设计更灵活。

**3、Mockito：**用于模拟依赖对象（如数据库、外部服务）的框架，常与 JUnit/TestNG 结合，隔离被测代码的依赖。

**核心注解与用途**

****

****

****

**单元测试的核心原则**

**1、隔离性：**每个测试应独立运行，不依赖外部状态（如数据库、文件）。

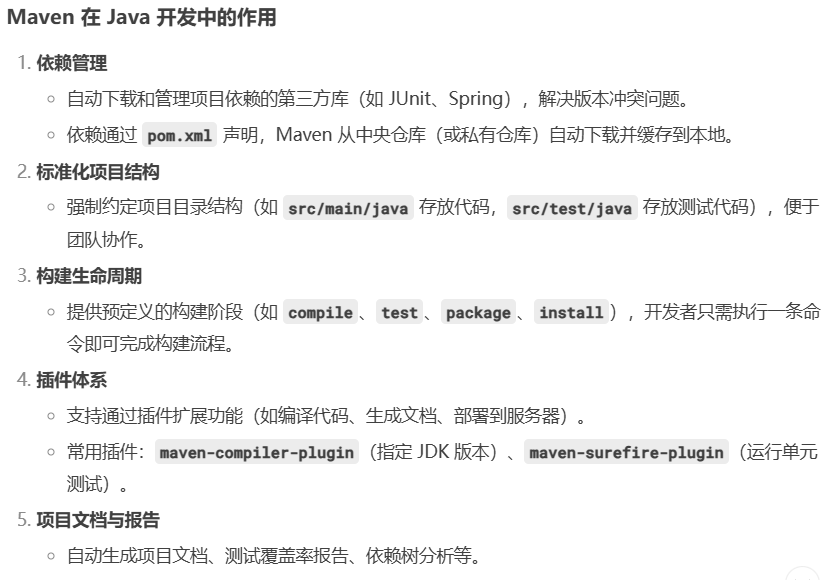
**2、可重复性：**测试结果应一致，不受环境或执行顺序影响。

**3、快速反馈：**测试应快速执行，避免阻塞开发流程。

**4、高覆盖率：**尽可能覆盖所有代码分支和边界条件。

3、Maven是什么？他在JAVA开发中起到什么作用？应该如何使用？

**Maven** 是 Apache 基金会维护的一款 **项目管理和构建自动化工具**，广泛用于 Java 开发。它通过标准化项目结构、依赖管理和构建流程，显著简化了开发、测试、打包和部署的复杂性.





2. 坐标（Coordinates）

在 pom.xml 的 <dependencies> 中添加依赖：通过 groupId（组织标识）、artifactId（项目标识）、version（版本）唯一标识一个依赖或项目。

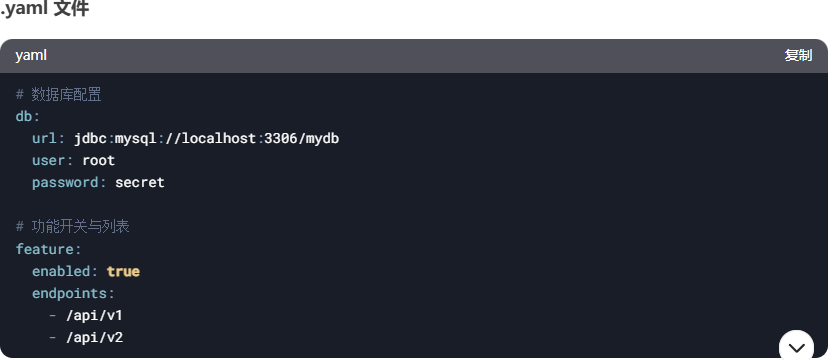


4、JAVA中，.properties文件和.yaml文件都是什么意思？二者有什么区别和共同点？

在Java开发中，.properties文件和.yaml（或.yml）文件都是常用的配置文件格式，用于存储应用程序的配置参数。

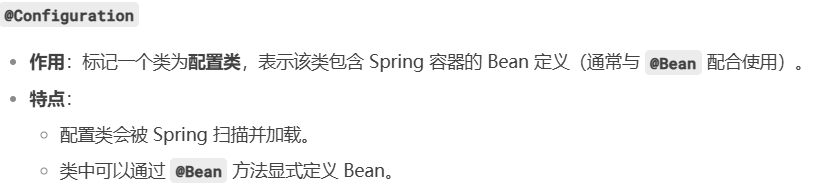


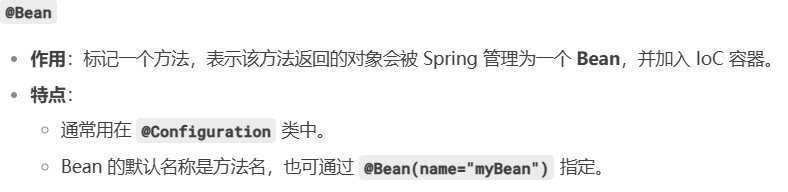


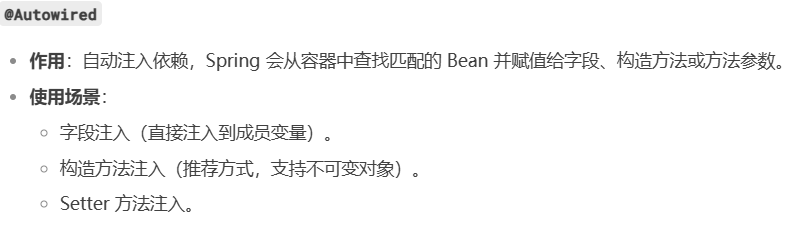


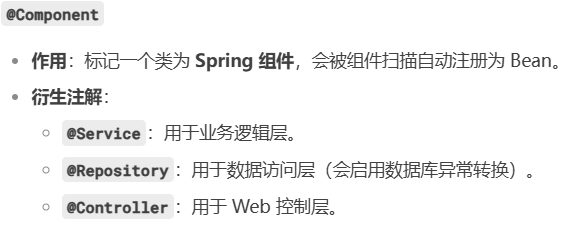
5、在SpringBoot中，@Bean、@Configuration 和 @Autowired这些注解都是干嘛用的？在介绍一下与之相关的常用注解

在 Spring Boot 中，@Bean、@Configuration 和 @Autowired 是核心注解，用于管理对象的创建、配置和依赖注入。









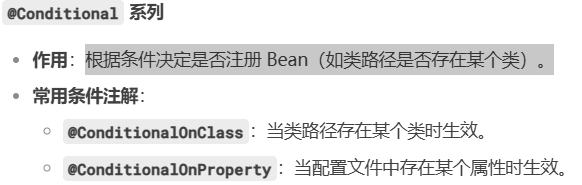
**@Qualifier:** 当存在多个同类型 Bean 时，指定具体注入哪一个（通过 Bean 名称）。

**@Primary:** 标记某个 Bean 为**首选候选者**，当存在多个同类型 Bean 时，优先注入此 Bean。

**@Value:** 注入配置文件（如 application.properties）中的值到字段或方法参数

**@Profile:** 根据当前激活的环境（Profile）决定是否注册 Bean。

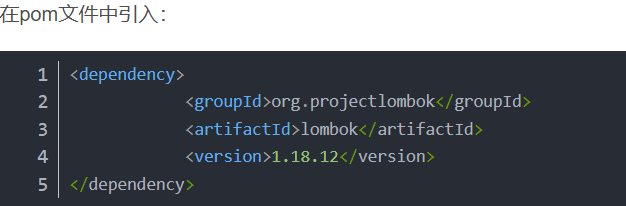
**@Scope:** 定义 Bean 的作用域（如单例、原型、请求、会话等）  
**@Lazy:** 延迟初始化 Bean，直到第一次使用时才创建。

****

****

6、什么是lombok？简单介绍其中常用的注解？

Lombok是一个java库，可以自动插入到你的编辑器和构建工具中，让你的java变得更加简单。再也不用写其getter、setter、[equals](https://so.csdn.net/so/search?q=equals&spm=1001.2101.3001.7020" \t "_blank)等方法，只需要一个注解，你的类就可以拥有一个功能齐全的javaBean。（通过注解快速让一个类拥有关键方法）

****

@Data：自动生成getter、setter、toString、equals等方法，实体类中就不用写了节约代码

@Getter、@Setter、@ToString都是单生成get或者set方法

@EqualsAndHashCode 自动生成全属比较 equals和hashcode方法

构造器:NoArgsConstructor 无参数构造器、AllArgsConstructor 全参数构造器、RequiredArgsConstructor 必须的参数构造器

@Value注解是不变注解，在@Data基础上加了final

@Slf4j 日志注解

@Accesser(chain=true)生成函数风格的set方法：

7、在JAVA中，@Resource 和 @Autowired有什么关联和区别

在Java中，@Resource 和 @Autowired 都是用于依赖注入的注解，但它们在来源、注入机制和使用场景上有明显区别。以下是它们的关联和主要区别，两者都用于自动装配 Bean，减少手动配置依赖的代码。@Autowired 支持构造器注入，而 @Resource 不支持。

