Spring

Spring 简介

Spring 自诞生以来一直备受青睐，它包括许多框架，例如 Spring framework、SpringMVC、SpringBoot、Spring Cloud、Spring Data、Spring Security 等，所以有人将它们亲切的称之为：Spring 全家桶。

Spring framework:

Spring framework 是目前主流的 Java Web 开发框架，是 Java 世界最为成功的框架。该框架是一个轻量级的开源框架，具有很高的凝聚力和吸引力。

Spring 是分层的 Java SE/EE 一站式轻量级开源框架，以 IoC（Inverse of Control，控制反转）和 AOP（Aspect Oriented Programming，面向切面编程）为内核。

发展历史:

Spring 由 Rod Johnson 创立，2004 年发布了 Spring 框架的第一版，其目的是用于简化企业级应用程序开发的难度和周期。

什么是 IoC:

IoC 指的是将对象的创建权交给 Spring 去创建。使用 Spring 之前，对象的创建都是由我们使用 new 创建，而使用 Spring 之后，对象的创建都交给了 Spring 框架。

什么是 AOP:

AOP 用来封装多个类的公共行为，将那些与业务无关，却为业务模块所共同调用的逻辑封装起来，减少系统的重复代码，降低模块间的耦合度。另外，AOP 还解决一些系统层面上的问题，比如日志、事务、权限等。

Spring 框架的特点:

1）方便解耦，简化开发

Spring 就是一个大工厂，可以将所有对象的创建和依赖关系的维护交给 Spring 管理。

2）方便集成各种优秀框架

Spring 不排斥各种优秀的开源框架，其内部提供了对各种优秀框架（如 Struts2、Hibernate、MyBatis 等）的直接支持。

3）降低 Java EE API 的使用难度

Spring 对 Java EE 开发中非常难用的一些 API（JDBC、JavaMail、远程调用等）都提供了封装，使这些 API 应用的难度大大降低。

4）方便程序的测试

Spring 支持 JUnit4，可以通过注解方便地测试 Spring 程序。

5）AOP 编程的支持

Spring 提供面向切面编程，可以方便地实现对程序进行权限拦截和运行监控等功能。

6）声明式事务的支持

只需要通过配置就可以完成对事务的管理，而无须手动编程。

Servlet 模块：提供了一个 Spring MVC Web 框架实现。Spring MVC 框架提供了基于注解的请求资源注入、更简单的数据绑定、数据验证等及一套非常易用的 JSP 标签，完全无缝与 Spring 其他技术协作。

Spring IoC 容器

IoC 容器是 Spring 的核心，也可以称为 Spring 容器。Spring 通过 IoC 容器来管理对象的实例化和初始化，以及对象从创建到销毁的整个生命周期。

Spring 提供 2 种不同类型的 IoC 容器，即 BeanFactory 和 ApplicationContext 容器。

二者的主要区别在于，如果 Bean 的某一个属性没有注入，使用 BeanFacotry 加载后，第一次调用 getBean() 方法时会抛出异常，而 ApplicationContext 则会在初始化时自检，这样有利于检查所依赖的属性是否注入。

Spring Bean 定义：

由 Spring IoC 容器管理的对象称为 Bean，Bean 根据 Spring 配置文件中的信息创建。

Spring 基于注解装配 Bean：

Spring 默认不使用注解装配 Bean，因此需要在配置文件中添加 <context:annotation-config/>，启用注解。

Spring 中常用的注解如下。

1）@Component

可以使用此注解描述 Spring 中的 Bean，但它是一个泛化的概念，仅仅表示一个组件（Bean），并且可以作用在任何层次。使用时只需将该注解标注在相应类上即可。

2）@Repository

用于将数据访问层（DAO 层）的类标识为 Spring 中的 Bean，其功能与 @Component 相同。

3）@Service

通常作用在业务层（Service 层），用于将业务层的类标识为 Spring 中的 Bean，其功能与 @Component 相同。

4）@Controller

通常作用在控制层（如 Struts2 的 Action、SpringMVC 的 Controller），用于将控制层的类标识为 Spring 中的 Bean，其功能与 @Component 相同。

5）@Autowired

可以应用到 Bean 的属性变量、属性的 setter 方法、非 setter 方法及构造函数等，配合对应的注解处理器完成 Bean 的自动配置工作。默认按照 Bean 的类型进行装配。

6）@Resource

作用与 Autowired 相同，区别在于 @Autowired 默认按照 Bean 类型装配，而 @Resource 默认按照 Bean 实例名称进行装配。

@Resource 中有两个重要属性：name 和 type。

Spring 将 name 属性解析为 Bean 的实例名称，type 属性解析为 Bean 的实例类型。如果指定 name 属性，则按实例名称进行装配；如果指定 type 属性，则按 Bean 类型进行装配。如果都不指定，则先按 Bean 实例名称装配，如果不能匹配，则再按照 Bean 类型进行装配；如果都无法匹配，则抛出 NoSuchBeanDefinitionException 异常。

7）@Qualifier

与 @Autowired 注解配合使用，会将默认的按 Bean 类型装配修改为按 Bean 的实例名称装配，Bean 的实例名称由 @Qualifier 注解的参数指定。

Spring AOP（面向切面编程）

AOP 采取横向抽取机制（动态代理），取代了传统纵向继承机制的重复性代码，其应用主要体现在事务处理、日志管理、权限控制、异常处理等方面。主要作用是分离功能性需求和非功能性需求，使开发人员可以集中处理某一个关注点或者横切逻辑，减少对业务代码的侵入，增强代码的可读性和可维护性。

简单的说，AOP 的作用就是保证开发者在不修改源代码的前提下，为系统中的业务组件添加某种通用功能。AOP 就是代理模式的典型应用。

Spring AOP 是基于 AOP 编程模式的一个框架，它能够有效的减少系统间的重复代码，达到松耦合的目的。Spring AOP 使用纯 Java 实现，不需要专门的编译过程和类加载器，在运行期间通过代理方式向目标类植入增强的代码。有两种实现方式：基于接口的 JDK 动态代理和基于继承的 CGLIB 动态代理。

Spring JDK 动态代理：

Spring JDK 动态代理需要实现 InvocationHandler 接口，重写 invoke 方法，客户端使用 Java.lang.reflect.Proxy 类产生动态代理类的对象。

JDK 动态代理只能代理实现了接口的类

Spring CGLIB 动态代理：

JDK 动态代理使用起来非常简单，但是 JDK 动态代理的目标类必须要实现一个或多个接口，具有一定的局限性。如果不希望实现接口，可以使用 CGLIB 代理。

CGLIB（Code Generation Library）是一个高性能开源的代码生成包，它被许多 AOP 框架所使用，其底层是通过使用一个小而快的字节码处理框架 ASM（Java 字节码操控框架）转换字节码并生成新的类。

JDK 代理和 CGLIB 代理的区别：

JDK 动态代理是利用反射机制生成一个实现代理接口的匿名类，在调用具体方法前调用 InvokeHandler 来处理。而 CGLIB 动态代理是利用 ASM 开源包，加载代理对象类的 class 文件，通过修改其字节码生成子类来处理。

JDK 动态代理只能对实现了接口的类生成代理，而不能针对类。

CGLIB 是针对类实现代理，主要是对指定的类生成一个子类，覆盖其中的方法。因为是继承，所以该类或方法不能声明成 final 类型。

JDK 与 CGLIB 动态代理的性能比较

生成代理实例性能：JDK > CGLIB

代理实例运行性能：JDK > CGLIB

Servlet

Servlet 是 Server Applet 的简称，译作“服务器端小程序”。它是一种基于 Java 技术的 Web 组件，运行在服务器端，由 Servlet 容器管理，用来生成动态的 Web 内容。

Servlet 程序其实就是一个按照 Servlet 规范编写的 Java 类。它具有平台独立性，可以被编译成字节码，移植到任何支持 Java 技术的服务器中运行。

发展历史：

Servlet 是 CGI 技术的替代品，直接使用 Servlet 开发依旧十分繁琐，因此 SUN 公司又推出了 JSP 技术。JSP 对 Servlet 再次进行了封装，JSP 经过编译后依然是 Servlet。

JSP

JSP（Java Server Pages）是一种动态网页开发技术。JSP 文件就是在传统的 HTML 文件中插入 Java 代码和 JSP 标签，后缀名为.jsp。

SP 使用 JSP 标签在 HTML 网页中插入 Java 代码，标签通常以<%开头，以%>结束。JSP 标签有多种功能，比如访问数据库和 JavaBean 组件等，还可以在不同的网页之间传递和共享信息。

JSP 是 Servlet 的扩展，我们可以在 JSP 中使用 Servlet 的所有功能。另外，JSP 还提供了一些其他功能，例如 EL 表达式、自定义标签等。

JSP 依赖于 Servlet，用户访问 JSP 页面时，JSP 代码会被翻译成 Servlet 代码，最终，以字符串的形式向外输出 HTML 代码。所以，JSP 只是在 Servlet 的基础上做了进一步封装。

JSP 通过表单获取用户输入的数据、访问数据库或其它数据源生成动态的 Web 内容。

JSP 具有以下特点:

- JSP 具有 Servlet 的所有优点，例如 JSP 文件跨平台，即一次编写，处处运行。
- JSP 比 CGI 性能更加优越，因为 JSP 可以直接在 HTML 中嵌入标签，而 CGI 需要单独引用 CGI 文件。
- JSP 比 Servlet 更易于维护，JSP 将业务逻辑与网页设计分离，使其更加灵活。
- 使用 JSP，Web 开发人员可以更注重于网页设计，Java 开发人员可以更注重于逻辑处理。

Servlet 与 JSP 异同点:

相同点：与 Servlet 一样，JSP 也用于生成动态网页。
不同点如下:

序号	Servlet	JSP
1	Servlet 在 Java 内添加 HTML 代码	JSP 在 HTML 内添加 Java 代码
2	Servlet 是一个 Java 程序，支持 HTML 标签	JSP 是一种 HTML 代码，支持 Java 语句
3	Servlet 一般用于开发程序的业务层（做一些复杂的逻辑处理）	JSP 一般用于开发程序的表示层（显示数据）
4	Servlet 由 Java 开发人员创建和维护	JSP 常用于页面设计，由 Web 开发人员使用

JSP 相对于 Servlet 的优点:

1）易于维护

相对于 Servlet 来说，JSP 更易于管理。在 JSP 中，我们可以轻松地将业务逻辑与网页设计分开，而在 Servlet 技术中，它们是混合在一起的。

2）快速开发：无需重新编译和部署

JSP 页面被修改后，不需要重新编译和部署项目。而 Servlet 被修改后，需要重新编译和部署。

3）代码简洁

在 JSP 中，我们可以使用 EL、JSTL、自定义标签、隐式对象等，能够有效的减少代码。