**1.利斯科夫替换原则（Liskov Substitution Principle, LSP）**

子类对象必须能够替换父类对象，而不会破坏程序的正确性。即：程序中任何使用父类对象的地方，都可以透明地替换为子类对象，且程序的行为不应发生异常或错误。

多态安全性：

例如，遍历用户列表执行创建或删除操作时，无需检查具体子类型。

降低耦合：

客户端代码依赖抽象的 User 或 Article，而非具体子类。

可扩展性：

新增普通用户或管理员用户时、新增文章的创建等操作，无需修改现有逻辑

2. 单一职责原则 (Single Responsibility Principle - SRP)

单一职责原则规定，一个类或模块应仅有一个引起其变化的原因，即其应仅承担一项职责。此原则旨在提高内聚性，降低复杂性，增强代码的可读性和可维护性。

在我们的博客系统中，功能被明确划分为独立的模块，这体现了SRP的应用：

用户模块：其职责限定于用户账户管理，包括支持用户注册、登录、个人主页管理等基础功能。

文章管理模块：专注于文章的生命周期管理，支持文章的发布、编辑、删除及多媒体分享。

评论管理模块：负责处理所有与用户评论、回复以及用户对自身评论进行管理相关的功能。

管理员后台模块：承担系统级管理与监控职责，提供管理员进行用户管理、内容审核、数据统计与系统设置的功能。

这种划分确保了每个模块的职责单一，例如，文章管理模块不处理用户认证或评论的具体交互逻辑，这些由其他专门模块负责。

3. 开闭原则 (Open/Closed Principle - OCP)

开闭原则的核心思想是软件实体（如类、模块、函数等）应对扩展开放，对修改关闭。这意味着新增功能应通过添加新代码实现，而非修改现有已稳定运行的代码。

我们的博客系统的内容管理与互动模块在设计上采用了松耦合、统一接口架构，以提升内容处理的实时性与扩展性。项目支持Markdown及富文本格式，用户可进行文本输入、图像嵌入、视频链接等多媒体处理。这为应用OCP奠定了基础。

例如，对于文章编辑器未来可能需要支持的新内容类型（如代码高亮块或投票功能），依据OCP，系统的设计应避免直接修改编辑器核心类。一种实现方式是定义一个通用的“内容块”接口或抽象类，每种内容类型（文本、图片、代码块等）都作为此接口的具体实现。编辑器核心逻辑仅与此抽象接口交互。因此，增加对新内容类型的支持，仅需添加新的“内容块”实现类，而无需改动原有核心代码。项目中关于如何兼容多媒体嵌入（如视频或代码高亮）的需求，适宜通过OCP来解决。

4. 里氏替换原则 (Liskov Substitution Principle - LSP)

里氏替换原则要求所有引用基类（父类）的地方必须能透明地使用其子类的对象，即子类对象应能够替换其父类对象而不影响程序的正确性。LSP是确保继承被正确应用的关键。

在我们的博客系统中，存在如RegisteredUser（注册用户）这样的用户类型，该用户是发布与编辑文章、评论与回复等用例的执行者。若RegisteredUser为某个通用User基类的子类，则LSP的应用至关重要。系统中可能存在的其他用户类型（如管理员Administrator）若也继承自User，那么任何期望接收User类型参数的函数或方法，在传递RegisteredUser或Administrator实例时，其行为应符合User基类的契约，不应产生意外或错误。子类在重写父类方法时，必须保持方法原有语义的完整性。

5. 迪米特法则 (Law of Demeter - LoD) / 最少知识原则

迪米特法则建议一个对象应尽可能少地了解其他对象，仅与其直接朋友（成员变量、方法参数、方法中创建的对象）通信。此原则旨在降低类之间的耦合。

我们的博客系统定义了清晰的模块间交互关系。遵循LoD意味着这些交互应通过明确定义的接口进行，而非深入对方内部实现。例如，当评论管理模块需要获取评论者信息时，它应调用用户模块提供的服务接口，而不是直接访问用户数据存储或通过一长串的间接对象引用来获取数据。这减少了模块间的依赖，提高了系统的可维护性。

6. 依赖倒置原则 (Dependency Inversion Principle - DIP)

依赖倒置原则指出：高层模块不应依赖于低层模块，两者都应依赖于抽象；抽象不应依赖于细节，细节应依赖于抽象。此原则有助于解耦，提高系统的灵活性。

我们的博客系统后端基于FAST API，数据库采用MySQL，并且系统通过数据库接口完成文章内容的结构化存储。这里的“数据库接口”即是DIP的体现。

以文章服务（高层业务逻辑）为例，它不应直接依赖于具体的MySQL文章存储库（低层数据访问实现）。而是，文章服务应依赖于一个抽象的文章存储库接口。具体的MySQL文章存储库则实现此接口。如此，若未来需更换数据库技术（例如迁移至PostgreSQL），只需提供一个新的PostgreSQL文章存储库实现，并将其注入文章服务即可，文章服务本身的代码无需修改。

7. 合成复用原则 (Composite Reuse Principle - CRP)

合成复用原则提倡优先使用对象组合/聚合，而非继承，来实现代码复用。组合/聚合（“has-a”关系）通常比继承（“is-a”关系）具有更低的耦合度和更高的灵活性。

我们博客系统的模块化设计本身即鼓励了组合。在实体层面，如一篇文章（Article），其字段列表包含文章ID、用户ID、标题、内容、标签列表等，这表明文章实体包含了作者信息（通过用户ID关联至用户模块的数据）和标签信息。Article实体通过拥有一个用户标识和一组标签来实现这些关联，而不是通过继承某个User类。这种组合关系使得Article的结构更为灵活，例如，未来若一篇文章可以有多个共同作者，通过修改组合关系（如将单个作者ID改为作者ID列表）比通过多重继承更容易实现。