单一职责原则（SRP：Single responsibility principle）又称单一功能原则，面向对象五个基本原则（SOLID： **SRP** 单一责任原则、**OCP** 开放封闭原则、**LSP** 里氏替换原则、**DIP** 依赖倒置原则、**ISP** 接口分离原则）之一。它规定一个类应该只有一个发生变化的原因。该原则由罗伯特·C·马丁（Robert C. Martin）于《敏捷软件开发：原则、模式和实践》一书中给出的。马丁表示此原则是基于汤姆·狄马克(Tom DeMarco)和 Meilir Page-Jones 的著作中的内聚性原则发展出的。

所谓职责是指类变化的原因。如果一个类有多于一个的动机被改变，那么这个类就具有多于一个的职责。而单一职责原则就是指一个类或者模块应该有且只有一个改变的原因。

单一职责原则告诉我们：一个类不能太“累”！在软件系统中，一个类（大到模块，小到方法）承担的职责越多，它被复用的可能性就越小，而且一个类承担的职责过多，就相当于将这些职责耦合在一起，当其中一个职责变化时，可能会影响其他职责的运作，因此要将这些职责进行分离，将不同的职责封装在不同的类中，即将不同的变化原因封装在不同的类中，如果多个职责总是同时发生改变则可将它们封装在同一类中。

之所以会出现单一职责原则就是因为在软件设计时会出现以下类似场景：T 负责两个不同的职责：职责 P1，职责 P2。当由于职责 P1 需求发生改变而需要修改类 T 时，有可能会导致原本运行正常的职责 P2 功能发生故障。也就是说职责 P1 和 P2 被耦合在了一起。

解决办法：遵守单一职责原则，将不同的职责封装到不同的类或模块中。分别建立两个类 T1、T2，使 T1 完成职责 P1 功能，T2 完成职责 P2 功能。这

样，当修改类 T1 时，不会使职责 P2 发生故障风险；同理，当修改 T2 时，也不会使职责 P1 发生故障风险。