**理解这八大优势，才算精通单元测试**

**什么是单元测试**

　　在计算机编程中，单元测试是一种软件测试方法，通过该方法可以测试源代码的各个单元以确定它们是否适合使用。 单元是最小的可测试软件组件， 它通常执行单个内聚功能。单元测试就是是指对这个最小可测试组件——即单元进行检查和验证。

　　单元体量小，因此比大块代码更容易设计、执行、记录和分析测试结果。 通过单元测试发现的缺陷很容易定位，并且相对容易修复。单元测试的目标是将程序分离成各自独立的部分，并测试各个部分是否正常工作。它将可测试软件的最小部分与代码的其余部分隔离开来，并确定其行为是否与预期的完全一致。单元测试能在使用过程中发现很多缺陷，在这种过程中证明自身价值。它实现了测试过程的自动化，减少了发现应用程序中更复杂部分中包含的错误的困难，并且由于可以关注到每一个单元而提高测试覆盖率。

　**单元测试工具**

　　常见单元测试框架有JUnit, TestNG, PHPUnit, PyTest, Jest, CppUnit, GTest, QTest 等八种，目前国产主流项目管理软件禅道全面集成这八种单元测试框架，打通持续集成闭环，将测试用例细分了单元测试用例和功能测试用例，可以直接在禅道页面上导入各种各样的单元测试框架的执行结果。这八种单元测试框架通过禅道ZTF与Jenkins持续集成功能打通。用户发起任务后，通过ZTF自动执行测试脚本，把单元测试的结果回传给禅道，二者合作打通了持续集成闭环，打通了项目管理工具和持续集成工具之间的沟壑。

　**为何单元测试是敏捷方法论**

　　在Apiumhub，我们采用敏捷方法，并且大量应用单元测试。单元测试是极限编程（Extreme Programming，XP）的一个特征，极限编程是敏捷软件开发方法之一，它能带来快速的测试驱动开发。我们坚信敏捷就要做持续集成和测试驱动开发。通过测试驱动开发，开发人员在开发代码时会创建单元测试，以便每个单元测试通常在编写代码之前就测试一小段软件代码。

　　**单元测试的优势**

　　单元测试提供了许多好处，包括及早发现软件错误、促进变化、简化集成、提供文档来源以及许多其他优点，接下来将对其进行详细介绍。

　　1、使流程更灵活

　　单元测试的主要好处之一是它使编码过程更加灵活，更遵循敏捷开发方法论。 当向软件中添加越来越多的功能时，一般需要更改旧的设计和代码。 但是，更改已经测试过的代码既冒险又高成本。 如果此时采用单元测试，那么就可以放心地进行重构。

　　单元测试实际上与各种类型的敏捷编程紧密结合，因为测试被内置在其中，让程序员可以更轻松地进行更改。 换句话说，单元测试有助于安全重构。

　　2、保证代码质量

　　单元测试可以提高代码的质量。 它能够确定在进一步发送代码进行集成测试之前可能出现的每个缺陷，在实际编码之前编写测试让人更难以考虑到这种问题。 而单元测试可以暴露出极端情况，让人编写出质量更高的代码。

　　3、尽早发现软件Bug

　　使用单元测试会让问题在早期就被识别发现。由于单元测试是由在集成之前测试单个代码的开发人员执行的，这样可以很早地发现问题，并在不影响其他代码片段的情况下解决问题。这既包括实施中的Bug，也包括单元规范中的缺陷或缺失部分。

　　4、促进变化并简化集成

　　单元测试允许在将来重构代码或升级系统库，并确保该模块仍然正常工作。单元测试能监测到可能违反设计合同的变化，有助于维护和更改代码。单元测试还可以减少新开发功能中的缺陷，减少现有功能更改时出现的错误。

　　然后通过单元测试对应用程序的各个部分进行测试，验证每个单元的准确性，再将单元集成到应用程序中。由于已经对各个单元进行了验证，在之后的集成过程中对应用程序进行测试就变得更容易。

　　5、提供文档

　　单元测试提供系统的文档。希望了解单元提供了哪些功能以及如何使用这些功能的开发人员可以查看单元测试，以获得对单元接口（API）的基本理解。

　　6、简化调试过程

　　单元测试有助于简化调试过程。 如果测试失败，则仅需要调试代码中最新的更改，这样以往的冗长的调试过程将被大大缩减。

　　7、设计

　　率先编写测试会迫使程序员在编写代码之前就仔细考虑设计和其他必须完成的工作。 这不仅可以让人专注，还可以创建更好的设计。 测试一段代码会迫使程序员定义该代码的责任。如果可以轻松做到这一点，则意味着代码的职责是被明确定义的，因此将具有很高的凝聚力。

　　8、降低成本

　　单元测试会更早地发现错误，有助于降低错误修复的成本。想象一下在开发的后期阶段（比如在系统测试或验收测试中）才发现Bug的成本将有多高。当然，前期检测到的错误也更容易修复，因为后期检测到的错误通常是许多更改的结果，测试人员可能就不会真正知道是哪一个导致了错误。

　　单元测试是针对代码单元的独立测试，核心是“独立”，优势来源也是这种独立性，而所面临的不足也正是因为其独立性：既然是“独立”，就难以测试与其他代码和依赖环境的相互关系。 单元测试与系统测试是互补而非代替关系。单元测试的优势，正是系统测试的不足，单元测试的不足，又恰是系统测试的优势。不能将单元测试当做解决所有问题的万金油，而需理解其优势与不足，扬长避短，与系统测试相辅相成，实现测试的最大效益。