国外专家教你如何处理遗留代码 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/291/2021_2022__E5_9B_BD_E5_A4_96_E4_B8_93_E5_c104_291102.htm 遗留代码不可避免在我们的职业生涯中，有很多时候必须忍受遗留代码。或许，你接受一份新的工作，遗留代码是你的第一个任务；或许，你们公司重组，并且有个产品最终在 你这里完成。不论什么缘由，事实就是这样。你想编写一些新的优秀的代码，但是现在负责的是对于你来说全新的完全不熟悉的一段代码。这个代码看起来相当复 杂、陌生，但你却不得不接受这项工作。 事实上，要是我扩展一下这个定义，你就能将任何一段先前编写的代码看作遗留代码。你是否曾经尝试回顾六个月前你编写的代码？忍受你自己所写的代码并非总是易事，更何况对于别人所写的代码。如果你不遵循一些基本的指导原则，这两种情况都是相当有挑战的。 传统的方法是，在你竭尽全力避免非有意的间接损害的同时，适当做些修改。不幸的是，因为对代码的不熟悉，当你改变一个数据结构或者更新一个变量时，你无法确信将要发生什么。 与其在这种充满危险的境况里盲目地徘徊，还不如制定一个处理的策略。不要只是作出改变后就期待万事如意。相反，瞄准目标，用“ BAT ”将它击出棒球场。 处理问题时，你可以三管齐下，构建（build），自动化（automate）并且测试（test）。对遗留代码可以使用这个BAT，并且给自己设置一个安全网。BAT方法将确保代码如你所期的那样继续工作。它可以“捕获 ”非有意的副作用，帮助你“ 消灭 ”它们。 构建 要解决的第一个问题就是构建。除非你能可靠地构建它，不然测试

一个产品很困难。先解决如何在你的电脑上干净地构建该产品，然后再编写构建脚本。 有 时候这是个不成问题的问题，但通常构建不是总能够如期待的一样的干净。构建通常局限于单一机器或者一个特殊的环境。在团队中，当代码在代码所有者间传递 时，很容易积累一些临时的构建需求。每个所有者可以添加他/她自己的特殊需求，并且混合在一起。当你接手这摊子事情时，厨房里可能已经有了很多的厨子。 一个复杂的构建能够引发整个产品一连串的问题。 当一件事情困难时，人们就很少去完成它。当一个构建困难时，人们就很少去构建它。这是人的天性。运行一个干净构建的能力正在成为一种黑色艺术，在你的工作环境中，只有少部分人才能掌握它。 因为你无法测试你未构建的事物，使测试变得不频繁。当人们最终运行他们的测试时，他们发现了更多的缺陷…不频繁的测试使得缺陷有更多的时间积聚。如果你每天都运行测试，那么只需要你报告当天的缺陷。如果你等6个月后去测试，你将会有很多的问题去处理。 因此测试变得繁重。测试者厌烦了测试周期内的所有工作，所以他们试图避免测试工作。没人喜欢记录十几个或者几百个缺陷这种无聊的工作。 开发者开始畏惧测试周期，因为他们感觉自己象被所有的缺陷报告进行轰炸。所以开发者开始怨恨，并且叨扰测试者。这使得测试周期更加痛苦。这真是一个恶性反馈循环。复杂的构建给整个产品生命周期带来一些问题。所以必须要保证构建是干净的。 当任何人都能构建时，任何人也就都能进行测试，于是也就会更加频繁地运行测试，产生出更小范围的缺陷报告。一次少量的工作就意味着更少的琐事。任何人都愿意搬动一桶涂料并且不用三思，但是如果让人去搬五

百桶涂料，看看他们会说什么。 你的目标就是创建一个能够在任何开发机器上易于运行、易于维护的干净的构建。使用一个构建脚本工具或者语言，例如Rake、Ant、Maven或者Nant.这些高层次的构建语言使你能聚焦于构建自己的应用程序，而不是构建、语言或者平台等细节。 当你可以仅用一句命令（如ant all）来构建产品，你就能继续到下一步。确保在不止一台机器上测试这一步。 我想请你试着用BAT方式来处理遗留代码。 看看与日常工作相比的区别是什么，是否需要用不同的手段处理工作。 自动化 现在你已经可以在任何一台机器上自动构建你的产品了，那么让我们来使它自动化吧。 自动化的目标是自动化整个构建，在一台干净的机器上，在测试周期内，使人为干预最小化。使万事自动化并非都有可能。但是我们希望达到的目标是，把那些可以被合理的自动化执行的东西都写入脚本。 有时，安装和配置一个软件要比编写一个脚本来自动安装和配置更为容易。只需一次性安装的应用程序当然是首选。诸如编译器，运行库和已存的数据集都属于这一 类。不要试图花费两个小时去复制已存的数据集。但是，如果你能在30秒内重建一个代表性的数据集，那么你便应该从头开始。以一个干净的已知的数据集开始的好处非常大，但是并非总是符合实际。不要因为重建你的数据，将15分钟的测试变为一个小时。 要确保记录下任何一个手动步骤及所有指令，它们可以为其他想复制运行环境的人提供帮助。 另一方面，为什么你应该整天监视着全部的构建？时间是宝贵的。IDE或多或少都是可以进行增量构建（incremental build）和运行细粒度单元测试（small unit test）。很多时候，这种部分覆盖（partial coverage）已经够用了。

让开发人员针对活动的代码区域（active code area），运行一组冒烟测试（smoke test），就可以覆盖大多数情况。100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com