[JUNIT]通过测试分类实现敏捷构建（3） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__5BJUNIT_5D_E9_80_9A_c104_144676.htm 理想情况下，还需要有调用单元测试和系统测试的任务。最后，在想要运行整个测试套件时，应该创建一个依赖于所有三种测试种类的第四项任务，如清单 3 中如示： 清单 3. 用于所有测试的 Ant 任务 创建定制 TestSuite 是实现测试分类的一个快速解决方案。这个方法的缺点是：一旦创建新测试，就必须通过编程将它们添加到适当的 TestSuite 中，这很痛苦。为每种测试创建定制目录更具扩展性，且允许不 经过重新编译就添加新的经过分类的测试。创建定制目录 我发现，用 JUnit 实现测试分类最简单的方法是将测试在逻辑上划分为与其测试类型相应的特定目录。使用这项技术，所有的单元测试将驻留在一个 unit 目录中，所有的组件测试将驻留在一个 component 目录中，依此类推。 例如，在一个保存所有未分类测试的 test 目录中，可以创建三个新的子目录，如清单 4 所示：清单 4. 实现测试分类的目录结构acme-proj/ test/ unit/ component/ system/ onf/ 为运行这些测试，必需至少定义四个 Ant 任务：为单元测试定义一个，为组件测试定义一个，依此类推。第 4 项任务是一个方便的任务，它运行所有三种测试类型（如 清单 3 所示）。 该 JUnit 任务和 清单 2 中定义的任务非常相似。所不同的是该任务 batchtest 方面的一个细节。此时，fileset 指向一个具体的目录。在清单 5 的例子中，它指向 unit 目录。 清单 5. 用于运行所有单元测试的 JUnit 任务的批量测试方面 请注意，这个测试只运行 test/unit 目录下的所有测试。当创建了新的单元测试

（或针对此问题的任何其他测试），只需要将它们放到该目录下，一切就准备妥当了！比起需要将一行新代码添加到 TestSuite 文件并进行重新编译，这样还是多少简单了一点。问题解决了！ 回到最初的场景中，假设您和您的团队认为使用特定目录是针对构建时间问题的最具扩展性的解决方案。该任务最困难的地方是检查及分配测试类型。您重构了 Ant 构建文件并创建了 4 项新任务（为单个的测试类型创建了三项，为运行所有这些测试类型创建了一项）。不仅如此，您还修改了 CruiseControl，从而只在（代码）签入时运行真正的单元测试，并以小时为基础运行组件测试。在进一步检查之后，发现系统测试也可以按小时运行，所以您创建了一个将组件测试和系统测试一起运行的额外任务。 最终结果是，测试每天都运行很多次，您的团队能够更快地发现集成错误通常在几个小时之内。 当然，创建敏捷性构建并未解决全部问题，但它在确保代码质量方面确实扮演了至关重要的角色。测试运行得更加频繁了，针对开发人员测试价值的顾虑成为一段遥远的记忆。另外，更重要的是，现在 2006 年您的公司获得了极大的成功！ 