Software Testing

软件测试丛书

.NET软件测试实战技术大全

——测试基础 流行工具 典型案例

.NET Testing Guide

Fundamentals, Tools and Practice

陈能技 编著

- 依照.NET平台软件测试的流程，全面阐述了.NET平台专门的测试技术，从单元测试到自动化功能测试、性能测试，从基本理论到测试的实用技术，从测试工具的使用到测试工具的开发
- 详细讲述了主流测试工具的实战应用技术，如 LoadRunner、QTP、TestComplete、NUnit 等
- 来自.NET平台测试实践中的典型案例，如界面测试、网络测试、Web服务测试、安全性测试、浏览器兼容性测试、智能设备测试、数据库测试等

人民邮电出版社
POSTS & TELECOM PRESS

.NET 测试基础篇

Chapter 1

第1章 软件测试基础介绍

本章介绍软件测试的基础知识，包括软件质量与软件测试的关系、软件开发与软件测试的关系、测试工具与软件测试的关系，最后还介绍了微软公司的 MSF 项目管理方法，以及其中的质量管理方法和测试方法。

1.1 软件质量与软件测试

软件测试的目的是保证软件产品的质量，但是仅仅靠软件测试是没办法完全确保软件产品的质量的，软件企业应该考虑更加全面的软件质量保证手段和方法。

1.1.1 软件产品与硬件产品的区别

软件产品与硬件产品存在很多明显的区别，其中一个区别是软件产品的出错模式没有硬件产品的出错模式那么明显。硬件产品通常在设计完毕后，就可以基本估计其可能出错的模式和原因，例如由于磨损造成的缺陷。而软件产品在设计完成后，其制作过程非常依赖于开发人员的个人能力和质量意识，在编码过程中可能引入大量的缺陷。硬件产品则一般具备比较成熟的制造工艺和制造设备，在制造过程中引入的缺陷会非常少。

说明

硬件产品的检测可以使用各种成熟的设备和工具，可以精确地测量出产品的合格程度。而软件产品虽然也有很多测试工具可以使用，但是不可能精确地判断产品的正确与否，仍然比较依赖于人工的判断或通过测试脚本的编程来实现判断逻辑的应用。

1.1.2 软件测试能保证软件质量吗

开发人员的编码能力、编码习惯、质量控制能力都极大地影响了软件产品的质量。另外，由于软件的复杂度会比普通硬件的复杂度更高，因为其状态空间、组合的可能性会更多，因此也造成了软件质量控制的复杂性。由于软件可能涉及复杂的业务逻辑，造成了对其正确性的判断比较依赖于人工的判断，以及需求定义的清晰程度。

如果仅仅依赖软件测试来保证软件的质量是注定要失败的，因为软件测试活动本身会存在错误和缺陷，测试人员本身会存在误测、错误判断、误用工具等现象。而且软件测试活动处于软件开发的“末端”，软件测试始终会略为滞后，因此往往容易受到开发进度的影响；如果缺乏测试时间或测试资源的话，测试人员是根本没有办法保证软件测试的充分性的，因此也无法保证软件产品的质量。

说明

从很多大型项目的失败经验看来，如果整体的项目管理和进度没有控制好的话，将直接影响软件测试的质量，从而影响软件产品的质量。如果软件的配置管理没有做好的话，即使软件测试质量得以控制和充分地保证，仍然无法确保最终发布产品的质量，因为可能实施人员所获取的是一个混乱的、未经测试的版本。

1.1.3　全面质量管理

软件产品的质量不能仅仅依赖测试人员来保证，必须从错误的源头出发，争取把错误“扼杀”在源头上。由于提倡全面的质量管理方法，项目组中的所有角色都应该尽到自己的质量职责。

软件错误越早被发现，其修改成本越低；越晚发现，其修改成本越高。应该首先确保在需求分析和设计阶段的质量，因为编码和测试都是依据这些阶段的成果来进行的。如果这些阶段的质量没有得到有效的保证的话，则后续的工作都是基于错误的、隐含风险的工作成果开展的，最终这些错误和缺陷会暴露出来，要么在编码阶段、要么在测试阶段、要么在实施维护阶段。越往后，则修改这些错误所要付出的代价越高。

说明

应该提倡全面的质量管理方法，开发人员不能认为质量保证、测试仅仅是测试人员的事情，而是要把软件产品的质量控制作为产品开发工作中的一部分来考虑。在开发过程中进行充分的调试和自测，确保提交给测试人员检测的程序不是一个充满低级错误的版本。

1.2　软件开发与软件测试

软件开发与软件测试看起来是一个对立面，因为开发做的是“建设性”的事情，而测试做的是“破坏性”的事情。但是，如果换一个角度来看的话，软件开发与软件测试实际上是一个相辅相成的过程。

1.2.1　缺乏软件测试的开发过程

软件开发不能脱离软件测试来进行。如果缺乏了软件测试，软件开发将失去度量的标准，仅仅依赖编码完成、编译通过是无法保证产品质量的，也无法衡量产品的质量，只能等待用户在使用过程中发现产品的质量缺陷。

目前比较流行的极限编程、测试驱动的软件开发方式是软件开发与软件测试紧密结合的最佳例子。软件测试是软件开发的“车头灯”，软件测试确保软件开发的方向是正确的，同时也是衡量软件开发进度的最佳度量方式，只有软件质量达到一定的程度，软件开发才能宣告停止。

1.2.2　软件测试中的开发技术

反过来，软件测试其实也离不开软件开发，软件测试也需要掌握开发和编程技术。

（1）软件测试人员需要了解产品设计的知识。对产品设计知识了解得越多，测试就越能深入到产品的核心位置。如果不了解软件架构方面的知识，则很难有效地帮助开发人员定位性能瓶颈，很难有效地协助开发人员解决性能问题。

（2）软件测试人员需要了解 UML 的知识。现在大部分软件开发组织都在使用 UML（统一建模语言）来指导设计和开发。其实，UML 对于测试人员也是非常有指导意义的，测试人

员也非常有必要学习一下 UML 的相关知识。

UML 中的用例图可以指导测试人员进行功能测试，而类图则可用于指导单元测试，状态图、协作图和活动图可用于指导测试用例的设计，顺序图则可用于系统测试、流程测试，构件图可用于指导单元测试和回归测试，配置图则可以指导性能测试、环境测试、兼容性测试等。

（3）测试人员一般不做编码工作，但是至少要懂得如何使用开发工具的基本编译功能。有时候，一些 Bug 是比较难重现的，甚至是只有在测试人员的机器上才会出现。这时候，测试人员应该使用开发工具运行程序，当程序出现异常时，会自动定位到出现异常的代码行，此时应马上请开发人员过来调试，寻找出错的原因。

大部分的自动化测试工具，需要测试人员具备一定的编码能力和语言知识。对于黑盒测试、手工测试者而言，具备一定的编程能力也会有好处。至少在与开发人员沟通一个 Bug 的时候，能理解开发人员的话，开发人员也会感觉测试人员是明白和理解其代码的人，而不会被认为是生硬的、不可理喻的、专门挑刺的人。

注意

具备良好的编程知识，可以让测试人员做更多层面的测试，例如单元测试、白盒测试、性能测试。还可以自己动手编写测试小程序或测试工具，帮助自己进行某些特殊的测试。

1.3 测试工具与软件测试

测试工具可以说是测试人员的“武器”，合理地使用测试工具将会对软件测试起到很好地帮助作用，对于提高测试效率、提高测试覆盖面都有很好的作用。

1.3.1 测试工具的分类

软件测试工具按照其用途，可大致分成以下几大类：

- 测试管理工具；
- 自动化功能测试工具；
- 性能测试工具；
- 单元测试工具；
- 白盒测试工具；
- 测试用例设计工具。

如果按测试工具的收费方式，又可分为以下几类：

- 商业测试工具；
- 开源测试工具；
- 免费测试工具；

技巧

合理选用各类测试工具，应用在合适的测试项目和测试类型中，将会使测试人员“如虎添翼”，可以让测试更加规范、使某些测试更加精确、高效率。

1.3.2　正确地使用测试工具

测试工具的作用很明显，就像所有的工具一样，都是起到辅助工作的作用。像项目组为用户开发的软件系统一样，很多时候，用户没有这些软件系统也可以正常地工作，但是效率可能会比较低。测试工具能极大地减轻测试人员的工作量，提高测试工作效率。例如，某些数据生成工具，可以在短时间内产生大量的数据，而不需要测试人员进行一条条数据的添加。

有些测试工具在提高工作效率方面的作用并不明显，但是却起到让测试更加规范的作用。例如，某些测试管理工具或缺陷跟踪工具，可以让测试的流程更加规范，测试有条不紊地进行，测试报告更加规范，数据记录更加完整，统计分析更加准确。

对待测试工具的辨证态度应该是：测试工具对于测试人员来说是必不可少的，但是不能迷恋工具。必不可少是因为很多测试如果缺少了工具是不可能完成的。实用主义测试者不会浪费时间去进行一条条数据的手工录入，造出 100 万条的数据表。

有些测试人员非常向往工具的使用，认为如果不使用工具，好像只是停留在手工的测试，感觉处于测试的很低级的阶段。这些测试人员会很迫切地寻找、试用各种新版本的测试工具，以为这样就可以让自己跟上测试的潮流了。

说明

真正的实用主义测试者不会这样，实用主义测试人员对待测试工具的态度是：“按需索取，信手拈来”。测试人员应该掌握和精通常用的测试工具，平时应注意积累各种测试工具的使用方法，在适当的时候、适当的测试项目中去应用这些方法。

1.4　MSF 中的软件测试

MSF 最早是由微软公司提出的针对软件开发和应用服务的项目管理模式，而今天它已经成为被全世界各软件研发企业所采用的一种项目管理理念。

1.4.1　MSF 项目管理模式

20 世纪 70 年代的软件项目在规模上比较小，一两个人基本上就可以胜任一个软件的开发，这样的人被称为 Hero，这个时代也被认为是英雄主导着软件项目的时代。但随着企业对软件依赖性逐步加强，软件在规模、复杂度上都有了较大的提高，对软件的质量要求也在不断提高，一两个人已经无法胜任软件开发的需要，开发人员一旦离开公司，那么整个项目甚至整个公司可能会陷入瘫痪的状态。

所以，在 20 世纪 80 年代初，软件企业开始重视软件开发的项目管理，开始把其他行业成功的项目管理经验引入软件开发领域。CMM 在部分软件企业得到了推崇，但是并不是所有的软件企业都采用 CMM，微软本身就没有采用。尽管如此，微软本身的管理方法与 CMM

也有异曲同工的地方。

MSF（Microsoft Solutions Framework）是微软所倡导的软件项目管理模式，是基于传统模式的基础上发展起来的，属于比较正式的模式。最新版本包含了灵活性的模板，加入了使用者角色（Personals）的概念，包含以下特点：

（1）推行一个从角色到使用方案的设计流程；

（2）开发过程采用循环型的 3 星期的周期；

（3）要求单元测试的程序与开发程序的源代码一起提交；

（4）要求 100%的源代码执行测试（Code coverage）。

说明

MSF 为成功地规划、设计、开发和部署 IT 解决方案提供了一套成熟的方法论。与具有固定框架的方法相反，MSF 提供了一个可以伸缩的灵活框架，以满足任何规模的组织或者项目开发团队的需要。MSF 指导由原理、模型和用来管理人员、项目和技术元素的准则（大多数项目都会碰到）组成。MSF 模型本身来源于微软公司在大规模软件开发和服务操作项目上的宝贵经验积累，来自于微软公司的顾问在为企业客户实施项目时所获得的经验，以及融合了来自于全球 IT 行业的先进理念，最终形成的一套方法论。

1.4.2 MSF 中的质量管理

在国内一些规模不是很大的开发团队中，质量管理投入往往是没有被重视起来。原因可能是多方面的，如项目时间紧张，人员紧张，调配不出更多的人员来进行专门的质量保证工作，而其中最重要的一条是，团队的领导者对质量管理投入的重视程度不够。必要的质量投入会为项目的实施与正式上线之后节省很多成本，而且质量投入是随着项目的进展一直进行的。

MSF 团队模型要求团队里的每一个人都要对质量负起职责，同时承担起测试过程管理的角色。测试角色会鼓励团队在项目期间进行必要的投入，以确保最终交付的软件产品或解决方案质量水平能够满足期望。在 MSF 过程模型里，由于项目交付内容是逐步生产和审查的，所以测试就成了质量的一部分。该模型定义了关键里程碑，并提出了中间里程碑，供测试角色和相关角色使用团队建立的质量标准对解决方案进行量化的测试。在软件项目进行的过程中，不断的对这些里程碑进行检查可以确保对质量的不断关注，并为在必要的时候进行中途的修正提供机会，避免风险，提高项目最终成功率。

注意

在 MSF 的框架中，并不只是测试人员才进行测试和质量保证的工作。例如“Usability Engineer”（可用性工程师）也会担任一部分的测试和产品检查工作。Usability Engineer 负责将设计的界面原型与客户沟通，进行易用性、用户体验等方面的检查、调查和测试，并根据所得到的结果对 UI 设计提出修改意见。也就是说，Usability Engineer 的工作是检验 UI 设计的合理性，这与测试人员的工作存在一定的交集，但是会更加专注于 UI 设计的检查，对界面设计人员的工作进行检查。

1.4.3　MSF 中的测试角色

MSF 中的质量保证团队被称为“Quality Assurance”或者“Test Team”，其组队模型中一般包括以下角色：

- 测试团队领导（QA Manager）：负责管理质量保证团队。
- 测试组长（Test Lead）：负责管理测试工程师，制定测试计划等。
- 测试工程师（Tester 或者 Test Engineer）：负责具体的测试工作。
- 测试开发工程师（Developer in Test 或者 STED）：负责测试工具、测试框架、测试脚本的开发。

测试人员的目标是要确保应用程序达到客户期望的水平。对于测试人员，一个主要的目标就是要在一个产品发布之前确定并提交产品中的错误，确保缺陷得以修补，并得以完整地记录。在开发过程中发现并修复错误要比在已经发布的产品中发现并修复错误的代价小。

注意

测试人员要参与项目的全部阶段和每一个迭代周期，测试人员需要和整个项目组就软件的质量标准达成一致的共识。

1.5　小结

本章主要介绍了以下内容：

（1）软件质量与软件测试的关系。

软件产品与硬件产品存在很多差异，因此软件产品的检测与硬件产品的检测也必然存在很多差异。软件测试是保证软件质量的手段和方法之一，但是绝对不是唯一的方法。软件产品的质量要靠软件开发团队的全体人员共同保证，要贯彻全面质量管理的思想，从软件研发的各个阶段、各个环节、各个角色和岗位出发，共同保证软件产品的质量。

（2）软件开发与软件测试的关系。

软件开发需要软件测试作为质量控制的方法，而软件测试也离不开软件开发，因为软件测试中同样有开发的内容，例如测试脚本的开发、测试工具或测试小程序的开发等都需要用到开发的知识。另外，对软件开发方面的内容了解得越多，越有机会做深入的测试，从而发现更多的 Bug。

（3）测试工具在软件测试中的作用。

掌握测试工具是测试人员进行专业性测试的必备技能，但是不能因为要使用工具而使用工具，而是要从实际的测试活动出发，选择合适的工具。既不能轻视测试工具的作用，又不能“迷恋”测试工具。

（4）MSF 中的软件测试活动。

MSF 是微软的项目管理方法，其中的质量管理方法和测试管理方法非常值得大部分软件企业学习和借鉴。

.NET 测试基础篇

Chapter 1

第2章

.NET 软件基础介绍

作为测试员，对被测试的对象了解得越多，则能考虑得更加全面、测试得更加充分。本章介绍.NET 软件的基本概念和特征，同时介绍.NET 软件的常用测试工具和测试方法。

2.1　.NET 平台简介

.NET 平台的软件是指基于.NET Framework 构建和运行的应用程序。.NET 平台的基本特征是代码托管、自动垃圾回收，基于.NET 平台来构建应用程序，可以免去很多与硬件、底层 API 打交道的麻烦，是构建面向对象的应用程序的基础框架。

2.1.1　托管代码介绍

.NET Framework 提供了一个称为公共语言运行库的运行时环境，它运行代码并提供使开发过程更轻松的服务。使用基于公共语言运行库的语言编译器开发的代码称为托管代码。托管代码具有以下优点：

- 跨语言集成、跨语言异常处理；
- 更强的安全性；
- 简化的组件交互模型、调试和分析服务。

托管的执行过程包括以下步骤。

（1）选择编译器。为了获得公共语言运行库提供的优点，必须使用一个或多个针对运行库的语言编译器。

（2）将代码编译为 Microsoft 中间语言（MSIL）。编译器将源代码翻译为 MSIL 并生成所需的元数据。

（3）将 MSIL 编译为本机代码。在执行时，实时（JIT）编译器将 MSIL 翻译为本机代码。在编译过程中，代码必须通过验证过程，该过程检查 MSIL 和元数据以查看是否可以将代码确定为类型安全。

（4）运行代码。公共语言运行库提供了可以托管执行的基础框架，并提供了执行期间可以使用的各种服务的结构。

2.1.2　.NET 基本概念

.NET Framework 是用于生成、部署和运行 XML Web Services 与各种应用程序的多语言环境，如图 2.1 所示。

.NET Framework 主要由三部分组成。

（1）公共语言运行库。

在组件运行时，运行库除了负责满足此组件在其他组件上可能具有的依赖项外，还负责管理内存分配、启动和停止线程与进程，以及强制执行安全策略。在开发时，由于做了大量的自动处理工作（如内存管理），运行库使开发人员的操作非常简单，尤其是与 COM 相比。特别是反射等功能显著减少了开发人员为将业务逻辑转变为可重用组件而必须编写的代码量。

（2）统一编程类框架。

框架为开发人员提供了统一的、面向对象的、分层的和可扩展的类库集（API）。目前，

C++开发人员使用 Microsoft 基础类，而 Java 开发人员使用 Windows 基础类。框架统一了这些完全不同的模型，还为 Visual Basic 和 JScript 程序员提供了对类库的访问。通过创建跨所有编程语言的公共 API 集，公共语言运行库使得跨语言继承、错误处理和调试成为可能。从 JScript 到 C++的所有编程语言具有对框架的相似访问，开发人员可以自由地选择他们要使用的语言。

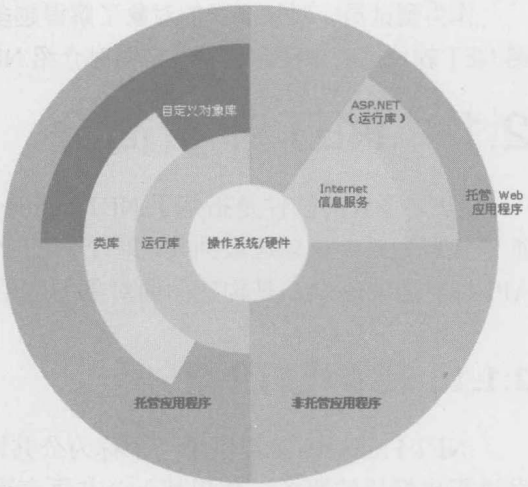

图 2.1 .NET Framework 环境

（3）ASP.NET。

ASP.NET 建立在.NET Framework 的编程类的基础上，为 Web 应用程序模型提供了一组可简化 Web 应用程序生成的控件和基础结构。ASP.NET 包括可用于封装通用 HTML 用户界面元素（如文本框、按钮和列表框）的一组控件。但这些控件在 Web 服务器上运行，并以 HTML 的形式将其用户界面呈现在浏览器中。在服务器上，这些控件公布面向对象的编程模型，该模型为 Web 开发人员提供面向对象编程的丰富功能。ASP.NET 还提供基础结构服务，如状态管理和进程回收，从而可以进一步减少开发人员必须编写的代码数量，并提高应用程序的可靠性。

另外，ASP.NET 使用这些同样的概念使开发人员能够以服务的形式交付软件。使用 XML Web Services 功能，ASP.NET 开发人员可以编写业务逻辑，并使用 ASP.NET 基础结构通过 SOAP 交付该服务。

2.2 自动垃圾回收

.NET Framework 提供了垃圾回收的自动内存管理机制，节省了代码编写中的内存申请和处理的麻烦。

2.2.1 .NET 的内存管理

.NET Framework 的垃圾回收器管理应用程序的内存分配和释放。每次使用 New 运算符创建对象时，运行库都从托管堆为该对象分配内存。只要托管堆中有地址空间可用，运行库就会继续为新对象分配空间。

说明

垃圾回收器根据正在进行的分配情况确定执行回收的最佳时间。当垃圾回收器执行回收时，它检查托管堆中不再被应用程序使用的对象并执行必要的操作来回收它们占用的内存。

2.2.2 .NET 程序的内存测试

造成内存泄漏的原因有很多，最常见的有以下几种。

- 分配完内存之后忘了回收；
- 程序写法有问题，造成没办法回收；
- 某些 API 函数的使用不正确，造成内存泄漏；
- 没有及时释放。

.NET 平台虽然有自动垃圾回收和内存托管机制，但是同样存在内存泄漏的问题。这种内存泄漏包括堆栈内存泄漏、资源内存泄漏等类型。可以使用一些专门的测试工具来找到这些错误，例如 CLRProfiler、GdiUsage、AQTime 等。

2.3 反射机制

通过 System.Reflection 命名空间中的类以及 System.Type，可以获取有关程序集的类、接口、属性等信息，可以使用反射在运行时创建类型实例，访问和调用这些实例。

2.3.1 反射机制简介

公共语言运行库加载器管理应用程序域，这些域在拥有相同应用程序范围的对象周围形成了确定的边界。这种管理包括将每个程序集加载到相应的应用程序域以及控制每个程序集中类型层次结构的内存布局。

说明

程序集包含模块，模块包含类型，类型包含成员。反射则提供了封装程序集、模块和类型的对象。可以使用反射动态地创建类型的实例，将类型绑定到现有对象，或从现有对象中获取类型。然后，就可以调用类型的方法或访问其字段和属性。

2.3.2 反射机制在测试中的应用

基于 GUI 对象识别和控制的自动化测试工具中，在过去一直依赖于 Windows API 函数的调用。而随着新的编程语言和平台的出现，涌现了很多新的语言特性，这些语言特性可用于自动化测试工具的设计中，例如，反射机制就是其中一项技术。

反射机制可被用在测试中，通过反射来加载被测试程序，获取被测试程序的各种属性，触发被测试程序的各种事件，从而达到自动化测试的目的。例如下面的 C#代码通过反射机制读取程序中 textBox1 控件的 Text 属性。

```
// 获取控件属性
static object GetControlPropertyValue(string controlName, string propertyName)
{
    if(AUTForm.InvokeRequired)
    {
        Thread.Sleep(1000);
        return AUTForm.Invoke(new GetControlPropertyValueHandler(GetControlProperty Value),
        new object[]{controlName, propertyName });
    }
    // 获取类型
    Type t1 = AUTForm.GetType();
```

```
        // 获取类型中的成员
        FieldInfo fi = t1.GetField(controlName, BindingFlags.Public | BindingFlags.NonPublic
        | BindingFlags.Static | BindingFlags.Instance);
        object ctrl = fi.GetValue(AUTForm);
        Type t2 = ctrl.GetType();
        // 获取成员中的属性
        PropertyInfo pi = t2.GetProperty(propertyName, BindingFlags.Public | BindingFlags.
        NonPublic | BindingFlags.Static | BindingFlags.Instance);
        // 返回控件属性值
        return pi.GetValue(ctrl, new object[0]);
    }
    delegate object GetControlPropertyValueHandler(string controlName, string propertyName);
```

而下面的 C#代码则通过反射机制调用控件的方法，模拟用户点击按钮的过程。

```
    // 模拟用户点击按钮
    InvokeMethod("button1_Click", new object[] { null, new EventArgs() });
    // 调用控件方法
    static void InvokeMethod(string methodName, params object[] parms)
    {
        if (AUTForm.InvokeRequired)
        {
            Thread.Sleep(1000);
            AUTForm.Invoke(new InvokeMethodHandler(InvokeMethod), new object[] { methodName, parms});
            return;
        }
        // 获取类型
        Type t = AUTForm.GetType();
        // 获取类型中的指定方法
        MethodInfo mi = t.GetMethod(methodName, BindingFlags.Public | BindingFlags.NonPublic
        | BindingFlags.Static | BindingFlags.Instance;
        // 调用方法
        mi.Invoke(AUTForm, parms);
    }
    delegate void InvokeMethodHandler(string methodName, params object[] parms);
```

另外，像 QTP、TestCompelte 这些测试工具也充分利用了.NET 的反射机制来测试.NET 的应用程序。例如，TestCompelte 通过在如图 2.2 所示的界面中加载程序集后，就可以在测试脚本中通过 CLR Bridge 来访问程序集中的对象属性和方法。

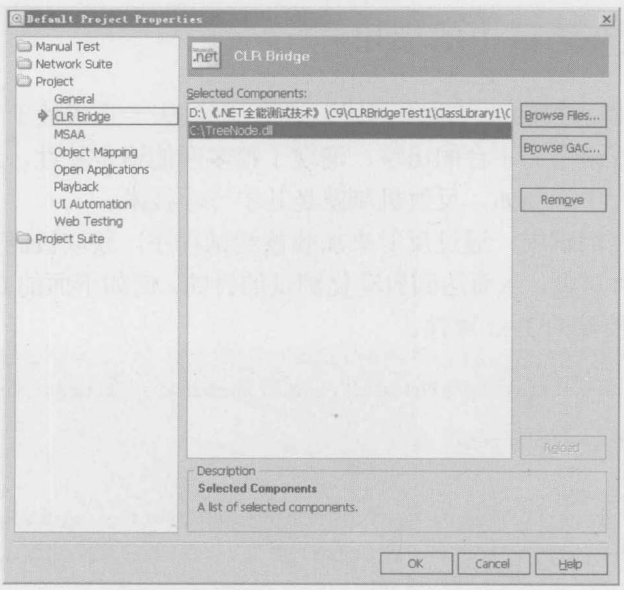

图 2.2 在 TestComplete 中设置 CLR Bridge

2.4 .NET 开发工具与测试工具

.NET 程序的开发可以采用各种开发工具，例如，Visual Studio.NET 2005、SharpDevelop 等，相应地也有各种各样的测试工具，用于测试.NET 开发的各种类型的软件。

2.4.1 .NET 开发工具简介

Visual Studio 是一套完整的开发工具集，用于生成 Windows 应用程序、ASP.NET 程序、Web 服务、移动设备应用程序等。Visual Basic、Visual C++、Visual C#和 Visual J#都是用相同的集成开发环境（IDE），利用此 IDE 可以共享工具且有助于创建混合语言解决方案。

Visual Studio Team System 是一个高效、集成且可扩展的软件开发生命周期工具平台，可以帮助软件团队提高整个软件开发过程中的沟通和协作能力。它由以下部分组成。

- Team Foundation：可扩展的团队协作服务器，可提供工作项跟踪、源代码管理、报告和流程指南。
- Team Edition for Architects：是一组集成的、用于面向服务的开发的应用程序设计工具。
- Team Edition for Developers：提供代码质量和性能工具，使团队可以构建可靠的、高质量的服务和应用程序。
- Team Edition for Tester：提供高级负载测试工具，使团队可以在产品发布之前验证应用程序的性能。

说明

如果安装了 Visual Studio 2005 Team Edition for Database Professionals，还可在 Visual Studio.NET 2005 中对数据库进行单元测试、检查数据库差异、生成大批量的数据。

2.4.2 .NET 平台的测试工具

除了 Visual Studio.NET 2005 自带的测试工具外，还有很多为测试.NET 平台软件而生产的测试工具。这些测试工具主要分成以下几类。

（1）单元测试工具：用于对.NET 平台编写的代码进行单元测试，例如 NUnit、NMock、NUnitForms 等。另外，还有一些用于辅助单元测试的工具，例如 NCover，可用于统计代码的测试覆盖率。

（2）代码分析和检查工具：用于检查.NET 代码存在的隐藏缺陷，找出不满足规范要求的代码。例如 DevPartner、StyleCop 等。

（3）自动化功能测试工具：用于验证.NET 软件的功能正确性，自动化地进行回归测试。例如 QTP、TestComplete 等。

（4）性能测试工具：用于检查.NET 软件的性能是否满足要求，验证软件的压力承受能力。例如 LoadRunner、NTime、AQTime 等。

2.5 .NET 软件类型及其测试

使用.NET 平台可以构建广泛类型的应用程序，对于每一类应用程序，其测试方法和测试的重点有所不同。

2.5.1 .NET 软件的分类

基于.NET 平台构建的应用程序主要分成以下几种类型。

（1）C/S 结构的应用程序。

这类程序的特点是客户端拥有丰富的界面，在用户界面、易用性、易操作性等方面要求比较高，并且后台一般拥有一个或多个的数据库，用于存储各种业务数据。

（2）B/S 结构的应用程序。

这类程序的特点是客户端使用浏览器访问各种应用，业务逻辑处理一般集中在后台服务器进行。由于处理相对集中，因此对应用系统的压力承载能力要求比较高。

（3）Web 服务。

这类程序的特点是没有 GUI 界面，提供统一的服务调用接口，可以使用各种编程语言访问和调用，可以作为组件整合到各种应用系统中。

（4）智能设备应用程序。

这类程序的特点是运行在各种智能设备上，内存和处理能力都会受到一定的限制。

2.5.2 .NET 软件的测试

对于前面所列的几种类型的.NET 软件，分别有不同的测试重点和特点。

（1）C/S 结构的应用程序。

这类程序的测试特点在于界面操作，需要结合用户界面规范来检查其规范性。并且由于 C/S 结构的客户端程序拥有比较丰富的业务逻辑处理代码，因此，对于这些代码的单元测试、功能的集成测试是测试的重点。可结合一些单元测试工具、基于 GUI 的自动化测试工具来编写自动化的回归测试脚本。

大部分 C/S 结构的应用程序用于解决某些企业的内部业务处理问题，因此会涉及比较多的基础数据、业务数据的问题。对后台数据库设计的检查会显得非常必要，可结合一些数据库测试和检查工具来进行数据库的单元测试、数据结构的差异性检查、数据的正确性检查。

（2）B/S 结构的应用程序。

对于这类程序的测试，需要注意浏览器的兼容性测试，因为客户端的用户可能会采用各种平台下的各种类型和版本的浏览器来访问应用，为了支持这些访问，则必须进行浏览器的兼容性测试。可以结合一些自动化的测试工具来进行这项测试，例如 QTP、Selenium RC 等。

一般采用 ASP.NET 来构建 B/S 结构的程序，对于 ASP.NET 程序的测试，与其他 Web 程序的测试一样需要注意链接的检查，Web 页面的功能测试，以及压力测试。可结合 LoadRunner 等性能测试工具来对这类程序进行压力测试。

（3）Web 服务。

Web 服务的测试有点类似于单元测试，因为 Web 服务不提供界面，只能通过访问 WSDL 来调用 Web 服务操作，检查返回的 SOAP 消息来验证 Web 服务的功能正确性。另外，由于 Web 服务可能要支持多个使用者的并发调用，因此也需要进行压力测试，可结合 soapUI、LoadRunner 等测试工具来进行 Web 服务的压力测试。

（4）智能设备应用程序。

智能设备的应用程序和普通 PC 平台的应用程序一样需要进行功能测试、界面规范性检查、性能测试。但是由于其运行平台的限制，未必能非常方便地进行这些测试，因此需要注意利用一些小工具来辅助进行这类程序的测试。

2.6　小结

本章主要介绍了以下内容。

（1）.NET 平台的基本概念。

.NET 平台的软件是指基于.NET Framework 构建和运行的应用程序。.NET 平台的基本特征是代码托管、自动垃圾回收。基于.NET 平台来构建应用程序，可以免去很多与硬件、底层 API 打交道的麻烦，是构建面向对象的应用程序的基础框架。

.NET Framework 提供了一个称为公共语言运行库的运行时环境，它运行代码并提供使开发过程更轻松的服务。使用基于公共语言运行库的语言编译器开发的代码称为托管代码。

（2）.NET 的内存管理及其测试方法。

.NET 平台虽然有自动垃圾回收和内存托管机制，但是同样存在内存泄漏的问题。这种内存泄漏包括堆栈内存泄漏、资源内存泄漏等类型。

（3）.NET 的反射机制及其在测试中的应用。

反射机制可被用在测试中，通过反射来加载被测试程序，获取被测试程序的各种属性，触发被测试程序的各种事件，从而达到自动化测试的目的。

（4）.NET 平台的开发工具和测试工具。

Visual Studio 是一套完整的开发工具集，用于生成 Windows 应用程序、ASP.NET 程序、Web 服务、移动设备应用程序等。这套工具也附带了一些可用于测试工作的工具，例如单元测试工具、性能测试工具等。

除了 Visual Studio.NET 2005 自带的测试工具外，还有很多为测试.NET 平台软件而生产的测试工具。这些测试工具主要包括单元测试工具、代码分析和检查工具、自动化功能测试工具、性能测试工具。

（5）.NET 软件的类型及其测试。

基于.NET 平台构建的应用程序主要分成 C/S 结构的应用程序、B/S 结构的应用程序、Web 服务、智能设备应用程序等，每一类应用程序都有自己的特点，相应地应该采取不同的测试方法和技术。

.NET 测试基础篇

Chapter 1

第3章

.NET 软件调试