外文文献译文

关于 Microsoft .NET 和 Sun 的 J2EE 的比较

简介

关于.NET 技术与 Sun 公司的 Java2 企业版（J2EETM）相比较，许多客户都想了解 Microsoft 公司的观点。由于以下的几个原因，.NET 和 JEE 的比较有点棘手：

1）一般来说，Windows .NET Framework 是 Microsoft 的 Windows 系统中经过精心定义的技术部分，而 J2EE 则是一个书面的协议。如果不局限于学术方面的讨论，换句话说，就是在几个应用平台上讨论这个话题的商业价值，那么仅仅比较 J2EE 和一个实际应用的工具是没有意义的。

这样实际应用的工具如：IBM 公司的 WebSphere 应用服务，BEA 的 WebLogic 服务或是其它类似的应用服务。

要想得到令人满意的分析，只有进行产品之间的比较，例如比较开发效率。使用 J2EE，开发者需要创建 4 个组件来建立一个单一的 EJB。表面上看来，这只不过是为开发效率付出的一点代价而已。但是 Java 的一些开发工具隐藏了一些开发技巧，降低了效率。另一个例子，J2EE 的部署体系十分复杂难解，类嵌入 JAR，而 JAR 嵌入 WAR，WAR 又嵌入 EAR。但是在一定程度上，有些工具能自动完成部署进程。上述情况导致决定一个应用服务商业价值的关键因素开发效率因不同的销售商而有差异，这主要取决于开发工具的效率。

2）关于“J2EE 全明星队伍”的问题。当比较.NET 和 J2EE 所有组件的集合时，这个问题就产生了。例如，分析者考虑开发效率时可能碰到下列问题，A 公司的产品， B 公司的应用服务程序， C 公司的安全规则， D 公司简便安装， E 公司决定价格。所有这些都可能和 J2EE 有关。集合上述这些特征属性，J2EE 工具看起来还行：价格便宜，安装简便，速度快，安全性高，有超高速缓存，并且有好的开发工具，等等。但这些都无关痛痒—因为不可能同时获得所有的这些特性。事实上，一次只能得到一个准确的特性。因为这些产品来自不同的公司，它们不可能合作无间。例如，IBM 公司的工具不能和 BEA 公司的 WebLogic 服务同时工作，因为后者是用的 Oracle 公司的缓存引擎，而 Oracle 的引擎不能用 Iona 的价格获得，等等诸如此类。人们有时候会误将“J2EE 的所有特性集合”作为比较的基础；但这是不合理的。客户需要的是知道一对一，产品对产品的比较。

3）比较.NET 和 J2EE 而忽视其它应用平台是十分重要的。J2EE 是仅关注应用程序服务器的规范。但是绝大多数客户对下列这些感兴趣：应用程序服务器，端口，商业服务器和分析工具，数据库，分离数据和流动性，信息代理，应用程序集合，容量管理，智能客户端等等。作为对客户要求的回应，这些因素应该统一工作，所有的主要销售商应该推行整合的平台。例如 Microsoft 的平台（包括 Windows 系统的客户端和服务器，Windows .NET Framework，Visual Studio.NET Framework，和 Microsoft 企业服务器）；BEA 的 WebLogic 平台；IBM 公司的 WebSphere 平台；Oracle 的平台；还有 Sun 公司的一个平台。将精力集中在这些平台的一个难题（应用服务器）上将会导致一个类似“树林和森林”关系的问题。这样的一个比方是合适的，但是它应该被考虑到一个更广阔平台的一部分。

从 Microsoft 的角度来看，和那些不常见的警告相比，这些是 Windows .NET Framework 和基于 J2EE 的产品的关键性的异同点。

相似点

1） Windows .NET Framework 和 Java 都有一个受控的运行时环境，它不但将源代码转换成中间语言，而且将这些中间语言编译成本地的可执行代码。两个环境都支持碎片整理、动态类加载和异常处理等。

2） .NET 和 Java 都倡导和支持基于组件的设计、多态性、继承和接口等，也提供基础类库来执行 I/O、XML 处理、带有连接池的数据库接入、文本操作与网页脚本编写等。

3） 两者都经过特有的销售商的产品进行发布。J2EE 规范自己是“销售中立”的，但实际上那些遵从规范的产品都必须实现规范外的特性，例如管理特性或是展开特性。因此，这些产品必须是对应特定的销售商。例如 Microsoft 公司的 Windows 和.NET 系统。

4） Windows .NET Framework 和基于 J2EE 的产品都和第三方的产品一起工作。例如，在后端数据库领域，.NET 和基于 J2EE 的应用程序能访问储存在 Microsoft 的 SQL 服务器、IBM 的 DB2、Oracle，Informix、Sybase 等服务器里面的数据。再举一个例子，.NET 和基于 J2EE 的系统能访问流行的信息中间设备，如 Microsoft 的 MSMQ 或是 IBM 的 MQSeries。同样，也包括文件系统，第三方开发工具，代码版本系统，防火墙等。

不同点

1） 原理

J2EE 是一个单一语言的平台，关注跨平台的可移植性。这就意味着，要利用 J2EE，设计方案能使用多个操作系统其中的一个，但开发者必须接受关于 Java 的培训。Microsoft 提供的.NET 构架作为 Windows 系统的一部分。开发者能使用多种语言，并且效率很高而不用进行一种新语言的重新训练。但.NET Framework 是 Windows 系统的一部分。

2） 宽度和广度

a. .NET 包括代码、产品、工具和构架，来利用网络上全部的计算资源，包括设备、个人电脑和服务器等。.NET 使所有的这些设备能经过标准通讯协议全部连接在一起，即所谓的“XML WEB 服务”。（.NET 应用程序可以和任何一个系统连接，无论系统用什么语言和平台，甚至是 J2EE。只要目标系统遵照 XML WEB 服务标准。）.NET 模型是广泛的分布式计算，它和许多代码互相通讯并交换信息。

b. J2EE 是面向服务器的模型，它并不开发网络上的智能和计算功能。总的来说，基于 J2EE 的产品只支持服务器端的应用程序。J2EE 一般把 PC 只看作是一个 HTML 的浏览器，而将这些设备认为是哑终端。至于 XML WEB 服务，现有的协议标准支持分布式的计算，现有版本的 J2EE 规范并没有提到 XML WEB 服务的问题，但是基于 J2EE 的产品在添加了附加装置后也可以支持 XML Web 服务。然而，添加附加装置也就意味着有严格的限制。例如，还不清楚现有的规范是否允许 EJB 调用 Web 服务，虽然 Web 服务的组件能调用一些 EJB 程序。

3） 编程模型的一致性

Windows .NET Framework 提供了一个跨服务器、PC 和其它设备的一致的、面向组件的模型。而 J2EE 提供 EJB 作为服务器端的组件模型；为客户端或是本地组件建立开放的完全用 Java 编写的 API；为用户界面提供 servlet；也为移动设备提供另一种不同的模型。甚至在 EJB 内部也有至少 3 种明显不同的子模型，每一种子模型都有不同的语言定义。

Microsoft 的.NET 编程模型与 Java 平台相比较，在各种服务器和客户端上有更好的一致性。J2SE 是

基于开放的完全用 Java 编写的 API，而 J2EE 是基于 Java servlet 和 EJB。
DH Brown，2002 年 7 月

4） 功能
a．Windows .NET Framework 提供一个能识别版本的类加载器，这就意味着应用程序的开发者能确保他们开发的应用程序在一部分代码已经更新的情况下仍能运行。而 Java 和 J2EE（现有的）没有版本识别的类加载器，这就意味着开发者和管理员不能保证代码被执行时是正确的。或是说，开发者只能靠运气来保证这一点。

b．Windows .NET Framework 显示了语言层面上的类属性—这就使得编程更加简单。例如，在源代码中只用一个简单的属性就能把.NET 组件标志为处理模式。或者说，一个.NET 组件和 XML 的串行化可以在一个属性中被定义。这个机制大大简化了许多编程任务。而 Java 不显示语言层上的类属性，虽然 Sun 公司考虑到要修改 Java 语言来改变现状。这种变化估计在两三年内才能第一次实现。
c．.NET 还支持分离数据访问，这主要用于在移动设备或是偶尔联网的场合里运行的应用程序。数据能被脱机操作，接着再和起始数据重新同步。而不论是 J2EE 还是 J2SE 现阶段都不支持分离数据访问，需要这项功能的 J2EE 开发者必须自己写“plumbing code”。

d．为建立基于网络的用户界面，Windows .NET Framework 提供基于事件的模型，这些模型类似于流行的 Visual Basic 中的智能客户端模型。ASP .NET 模型使得建立、发布和维护一个基于网络的用户界面变得更加容易。与之形成对比的是，J2EE 在 JSP 中不支持这样的模型。有一些第三方的扩展程序部分弥补了这些功能，但是它们的实用性和简便性不能和 ASP .NET 相比。作为一个推荐的 J2EE 附加程序，Java Server Faces 可能做到这一点。但这个附加程序并没有包括在 J2EE 的 1.4 版本以前。而要获得销售商的支持，则又需至少一年的时间。
e．J2EE 支持一个对象相关的数据映像模型，它被称作 EJB Entity Beans。这样是为了允许开发者更容易地从一个相关的数据库建立对象模型。然而，实际上把这个想法编程实现却要面对下列问题：

Ⅰ.易用性：当那些熟知的、正规定义的、被广泛支持的结构化查询语言（SQL）和开发者的数据相互作用时，开发者不得不放弃它们，而使用一个被称为 EJBQL 的弱定义查询语言。和 SQL 相类似，EJBQL 的功能并不强大（例如，在现有的规范中，它没有 ORDER BY 的语句，这样开发者就没法使用特定数据库的 SQL 扩展），而它的语义也没有被正规定义。还有，在对象间建立联系和附属关系十分困难，而且在对象间和 XML 以及后端之间的数据翻译是手动控制的。

Ⅱ.性能：基于 EJB 系统的性能仍是一个未知数。没有提供公开的基准。客户反映，得到的性能远远偏离了 Entity Beans，并且转向一个更直接的数据访问策略。这是 EJB Entity Beans 没有被广泛使用的一个关键因素。
在 Windows .NET Framework 中，数据访问是基于数据集比较的。数据集保存了相关数据的一个子集，它由一个或多个 SQL 查询语句描述。数据集中的数据可能保存关键的联系，并且开发者能直接对数据进行操作，能将数据转换成 XML 格式和上次操作的类型，能使用标准的 SQL 过滤数据等。总而言之，相对于 EJB Entity Beans，.NET 的数据集模型提供一个更丰富而且简单熟悉的途径。

5）简便性
a． 配置：对于 J2EE，配置是由部署描述信息获得的 XML 格式的文件，它们和实际执行的商用逻辑代码有明显区别。这种方法有很多问题。第一，考虑到特定类的元数据，有些代码中的改变和元数据中的改变是相互依赖的。两个独立文件的同步性要求有可能产生错误。第二，考虑到应用程序层的元

数据，在J2EE中，没有可以从一个程序继承元数据到另一个程序的途径。与J2EE不同，Windows的.NET构架包括了这个功能，使得可以在源代码中直接向类添加属性，这样就不会产生第一个问题。Windows .NET中的元数据模型允许客户自己添加扩展程序，这样开发者就可以编写和使用自己的属性。为了在Windows的.NET构架中配置外部元数据，这个功能被包括在配置文件的分级系统中，它能从父系统中继承属性，这样每个文件会很小，它只记录改变的设定。这就避免了J2EE模型的第二个问题

b. 数据库连接池Windows .NET Framework中，是根据需要自动建立和管理这些池的。而在J2EE模型中，连接池必须被明确配置和管理。

c. XML Web服务：在.NET中建立一个XML Web服务就像在类中添加一个属性那样简单。有些基于J2EE的产品也想在Java中模拟这个功能，.NET提供更简单的方法来建立和使用可由双方共同操作的XML Web服务。

d. 部署：在.NET中，要部署一个应用程序，管理员只需要拷贝文件。而在J2EE中，管理员必须将很多编译文件和JAR、WAR以及EAR绑定，然后在一个特定的服务器部署工具中解开并运行它们，接着拷贝结果档案。这个多步部署过程意味着典型的编辑/编译/调试循环被大大延长了。此外，由于动态加载类过程中的一些变化，更新一个简单的类常常需要重新启动基于J2EE的服务器。

虽然许多公司选择Java作为企业发展的策略平台，但它们的使用却由于J2EE的复杂性而受到阻碍。Meta Group，8月

6）成本

a. 为了部署，运行在Windows .NET Framework之外编写的服务器端的应用程序需要一个Windows Server的许可，这比三个遵从J2EE的商业服务器中的任何一个许可都便宜很多。包括四个网络服务器的系统部署费用的差别可达到数十万美元。例如，Microsoft Windows Server 2003（企业版）的一个四机器系统（每个有四个pc）的许可费用不超过16，000美元（这考虑了零售因素）。而WebSphere Application Server 5.0在同样的系统中每台pc的许可费用达12，000美元，这共要192，000美元。这个比率是12比1。大多数基于J2EE的商业应用程序服务器的价格都和这类似。（这假定了性能相等。然而实际上Middleware公司2002年10月的报告显示，一个建立在Windows .NET Framework上的应用程序的效率是建立在同样流行的基于J2EE的服务器上的程序的2-4倍。所以实际上价格的优势远高于12比1）有很多免费的，基于J2EE的开放源应用服务器，但是它们并没有J2EE-compliant的商标。还有关于文件和产品的问题：需要产品之间的比较来讨论采许可费用。

b. 为Windows .NET Framework开发工具的费用也更加低廉。Visual Studio .NET是.NET的整合开发工具，它的许可费用大大低于商业化的J2EE销售商制定的开发工具的费用。并且在业界，Visual Studio .NET作为最佳开发工具赢得了一系列的大奖。评估过Visual Studio .NET和其竞争对手的客户都说，相对于最好的Java工具，Visual Studio .NET开发效率更高（See Giga，2002年6月）。

c. 使用Windows .NET Framework的开发和维护费用更低。专家认为许可费用并不是一个项目的最大开支。典型的软件开发和维护占项目总费用的50-80%（Glass，2002；Kemerer，1995；Gartner，2001）。Middleware公司2002年10月的研究表明，在Windows .NET Framework上一个给定的应用程序开发相对于J2EE，只需要1/3的代码。代码越少就意味着维护更加简单。

总结

显而易见：正确的产品选择决策不可能不评估实际的产品。对比Microsoft Windows Server及

Windows .NET Framework 和 J2EE（Sun 公司的规范）是有价值的，但是这样的努力缺少实际产品的评估。然而，还是可以从中得出一些结论：

1） J2EE 展现了一个以服务器为中心的原则，并将重心放在 EJB 和解决“相关对象的映像问题”上。J2EE 在支持 XML 和 Web 服务上已经落后了。Windows .NET Framework 的原则则是通过协议标准和 XML、充分利用服务器、接口和设备的的大规模分布式计算。

2） 相对于编写在 Windows .NET Framework 上的程序，J2EE 应用程序需要更多的代码来执行相同的任务。

3） J2EE 的管理和部署模型更像是一个主机模型，它关注保护和限制稀有的计算资源，按比率使用。而 Windows .NET Framework 展现出的原则是计算资源是廉价的，而且将更加廉价，但是部署能力将保持大部分昂贵的资源。

总之，如果一个项目要求必须从几个操作系统中选择一个作为部署平台，而不考虑开发成本；强制（并且重新培训训练）开发者使用单一的编程语言来执行这个项目，从而代码的版本问题就不再重要；重要的是配给和限制相对便宜的计算资源；这样使用昂贵复杂的开发和维护工具就显得顺理成章；而编写更多的代码也有其优越性-- J2EE 也许是一个不错的选择。

然而，如果商业目标显示最优化的开发效率是重要的；低廉的性价比更符合要求；通过通讯协议的标准获得的可相互操作性有较高价值；大量支持基于界面的应用程序和移动的应用程序是重要的；更感兴趣的是易扩展性—这样的话，建立一个 Windows .NET Framework 上的 Windows Server 应用程序是正确的选择。