

# Using .NET with MySQL

[.NET architecture](#) [.NET and MySQL](#) [Connecting from Visual Studio](#) [ODBC.NET Starter C# ODBC application](#) [Master-detail window](#) [Native .NET drivers](#)  
[ASP.NET Model](#) [ASP.NET development](#) [Web applications with MySQL](#)

## Why .NET?

.NET is a large, evolving collection of old and new Microsoft technologies, bundled into development and server packages for Windows 2000, XP Pro and 2003 Server. It includes:

- a kernel, the *.NET Framework*, that implements HTTP (Hypertext Transfer Protocol), XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), and UDDI (Universal Description Discovery and Integration). There are already three versions of .NET Framework: 1.0, 1.1 and 2.0;
- data handling classes bundled as *ADO.NET* (ActiveX Data Objects);
- *Internet Information Services* (IIS);
- a *Common Language Runtime* (CLR), a virtual machine for executing code compiled from various languages (analogous, and not accidentally, to the Java Virtual Machine);
- *ODBC.NET*, a wrapper for ODBC drivers;
- Microsoft's *SQL Server*;
- *ASP.NET* (Active Server Pages .NET), an environment for developing web applications using VBScript, JavaScript, Perlscript, Python, Visual Basic, C#, C, Cobol, and even Smalltalk or Lisp; now in two versions, 1.0 and 2.0;
- *Visual Basic .NET*;
- *Visual Studio*: an elegant, expensive application development environment incorporating all the above, like .NET Framework in three versions—VS 2002, 2003 and [2005](#);
- *Web Matrix*: a web application development environment for NET Framework versions 1 and 1.1, simpler than Visual Studio and free;
- *Visual Basic, C#, C++, J++ and WebDeveloper 2005 Express Editions*, which are basically free language-specific follow-ons from Web Matrix for .NET Framework 2.0.

.NET supports true O-O programming with inheritance, polymorphism, and encapsulation. There is tremendous language flexibility and tremendous debugging support. There is good support for multi-level authentication—a must for multi-tiered web applications. If Visual Studio .NET is too rich for your budget or your PC, you can use Web Matrix with .NET Framework 1, or Visual Web Developer with .NET Framework 2.0, or you can use a Borland tool like Delphi or C++ Builder. Since .NET is available for virtually any device (PC, workstation, phone, PDA, whatever) that can reach the internet, it is an appealing platform for developing software which can be leased as on-demand service rather than requiring purchase (to compare various editions browse <http://msdn.microsoft.com/vstudio/products/compare/>). All that power, and convenience, and flexibility, and robustness, are strong reasons to consider .NET.

## But ...

All that flexibility, robustness and marketing, cost you money, computing power, efficiency, freedom, and security.

*About the money:* Some .NET elements—including .NET Framework, ODBC.NET, Web Matrix and Visual Web Developer—are free, so you can develop real web applications without making a big purchase. Middleware that connects .NET to MySQL databases (Connector/NET from MySQL) is free too. Visual Basic .NET is not expensive. But if you want the full-fledged .NET development environment, you will find that Visual Studio .NET costs somewhere between \$500 and \$2,500 depending on the package, and the total cost of ownership, including add-ins and necessary updates, runs beyond the affordable range of many a small organisation.

*About the computing power:* If the machine where you run Visual Studio is limping along with a quarter of a gigabyte of RAM or less, or a slow processor, be prepared for endless frustration till you upgrade. You need half a gigabyte of RAM at least, and for acceptable performance, double that. The more computing power you throw at Visual Studio, the happier you will be.

*About efficiency:* The magic of .NET database interoperability is an XML layer between code and data source. That mapping costs you CPU cycles, memory and time. So does the huge .NET class hierarchy. ASP and ODBC impose additional length on code paths, are memory-hungry, and execute slowly—serious problems for web applications and their servers.

And if, to keep costs down, you are considering ODBC.NET, there is the double-edged question of database independence. If application code is free of specifics tying it to a particular RDBMS, the database backend can be swapped out like a battery in a car. Isn't that a terrific advantage? Yes and no. Yes to the extent that the application's SQL scripts are limited to entirely portable, vanilla code. No to a similar extent, since the degree to which the SQL code is generic is the degree to which the code fails to take advantage of optimisations available in a specific RDBMS. Database independence is good for portability. Database *dependence* is good for performance.

*About freedom and security:* Microsoft products are famous for features which "encourage" use of other Microsoft software. ASP.NET officially requires that you deploy on Microsoft's IIS, which is infamous enough for security vulnerabilities to inspire a steady march away from it, especially to PHP and Apache. It doesn't matter whether IIS vulnerabilities are due to bad code, or to hacker preferences. The problems are serious, they continue, and they cost.

You may be able to use .NET and sidestep IIS. Covalent Technologies discontinued its Apache web server that supported ASP.NET, but you can run a website on Apache and relay requests to IIS running internally on another machine. Or you can switch to Linux/UNIX and develop ASP.NET applications using Novell's Mono (<http://www.mono-project.com/about/index.html>).

The choice between PHP/Apache and .NET is a matrix (Table 15-1). .NET is extremely powerful and versatile. It is rigorously O-O. It is designed with the enterprise in mind. It is supported by the biggest software company on the planet. It is here to stay. The Visual Studio IDE is spectacularly good. Total cost of PHP/Apache ownership is much lower, though, because full-featured downloads are free, upgrades are free, many add-ins are free or nearly so, and there is a huge and helpfully responsive user community. No open-source development company will charge you money to report a bug, and the PHP and Apache open-source development groups are very good at correcting bugs quickly. PHP and Apache run on every platform you are likely to work on or

port to. Which line items in the matrix weigh most heavily in your situation will determine whether .NET should be your choice.

**Table 15-1: Apache/PHP or .NET?**

	<i>Apache/PHP</i>	<i>.NET</i>
Price of core elements	zero	zero
Price of development tools	zero	high
OOP	weak	strong
Open-source	yes	no
Development GUI	no	excellent
Rich UI data controls	none	many
Support, bug fixes	free, quick	expensive, slow
Speed	good	fair
Efficiency	good	poor
Security	good	good
Vulnerability	small	large
Exception management	since PHP 5	yes

To read the rest of this and other chapters, [buy a copy of the book](#)

---

[TOC](#) [Previous](#) [Next](#)

---