

MySQL APIs

A major strength of MySQL is its Application Program Interface (API) system. In principle it makes MySQL databases available to virtually any programming language. MySQL publishes *connectors* for PHP, Java, .NET and ODBC, and a C/C++ API. Connector/NET opens MySQL to any .NET language including Visual Basic and C#. Connector/ODBC opens MySQL to Microsoft Access and any other language that can talk to ODBC. Third parties have published connectors for Perl, Python, Ruby, Delphi, Kylix, Eiffel and more.

A MySQL API is a collection of recipes for executing MySQL commands in a particular programming language or environment. In most APIs, you will find all the DDL and DML functionality you need—functions to create, drop and use databases, tables and so on, plus functions to select, insert, update and delete.

We return to the importance of the [specification and its Use Cases](#). The more you design the solution at the Use Case level, the less crucial the particular programming language you are using. Much of the database programming part of the job becomes syntax lookup.

A MySQL-enabled application tends to do four basic things again and again:

- connect,
- issue DML or occasionally DDL to retrieve and/or store data,
- update the user interface (UI) accordingly,
- disconnect.

The application should encapsulate Use Cases, and partition the presentation layer from the data management layer. Development environments like Java and .NET encourage such structure. Scripting languages like Perl and PHP do not; you are solely responsible for encapsulating code in routines for each Use Case, then breaking out smaller units of work into connect, query, UI and disconnect submodules.

Why these APIs?

No book has enough pages to cover all language interfaces to MySQL. Which will be most interesting and necessary in a year, in three years?

Judging by the fairly steady rate at which new questions are posted to MySQL [help forums](#), the PHP API is used three times as much as .NET or ODBC, four times as much as Java, six times as much as the C API, 16 times as much as Perl, 40 times as much as

Python, and 80 times as much as Ruby. We take the PHP, .NET, ODBC, Java, Perl and C APIs to be essential.

The ODBC, Perl and PHP APIs have small function sets, so productivity comes soon. Even for Perl and PHP, though, a thorough account fills a large book. In the case of Java and .NET, a thorough account fills a bookshelf. Our API chapters do not even remotely approach thoroughness. We offer them as user-friendly entry points, pathways to early productivity, and above all as invitations to further study, experiment and practice.

Java

Java is everywhere. The fact that IBM and Sun champion Java ensures Java's continuing penetration into enterprise computing, most of which currently runs on Windows. Part of IBM's pitch is the move from Windows to Linux, but from our perspective that is almost irrelevant. Whether or not an organisation converts to Linux, Java will be there. In the emerging world of web services, Java is a major player. The MySQL bridge to this world is *Connector/J*.

ODBC

Connector/ODBC (formerly MyODBC) enables any ODBC-compliant language to connect to MySQL. A single API works for Access, Visual Basic and C++, Delphi, PowerBuilder – virtually any Windows development environment.

From the Linux perspective, understanding Connector/ODBC will enable you to address databases trapped in SQL Server or Access or Approach or QuickBooks, and to deal with their data.

.NET

The .NET framework is impressive, and so is Microsoft's marketing team. A formidable marketing team becomes much more so when it is selling a quality product, and .NET is a quality product. The MySQL bridge to that platform is *Connector/.NET*.

The .NET framework offers its own variety of language-abstraction. As Java has a runtime that interprets javacode instructions, so .NET has a Common Language Runtime (CLR) that interprets CLR instructions. Languages that support .NET speak to the CLR, which in turn passes instructions to the operating system, printers, other programs in memory and so on. If you write module *u* in Visual Basic, *v* in C#, *x* in C++, *y* in Eiffel and *z* in Java, you can already get the first four of these to run against the CLR, and there is a project underway to make this possible for Java too.

This is a very powerful model. But it's confined to Windows environments, you say. As it turns out, .NET is *not* confined to Windows. There is a project called *Mono* which implements the CLR in Linux! You *can* have your cake and eat it too.

Perl and PHP

The majority of MySQL installations are part of a LAMP (Linux-Apache-MySQL-Perl/PHP) setup. For years, software futurists have been predicting that this combination would soon fade into legacy. It has not happened.

Given competing high-powered tools like Java and .NET, why not? For the same reason, perhaps, that Ali could knock out heavier opponents: LAMP is light enough on its feet to float like a butterfly and sting like a bee. A website with database smarts and user validation can go from idea to deployment in *hours* rather than weeks, months or years. There is a database interface plugin (*DBI*), and a MySQL driver plugin (*DBD-mysql*). *Connector/PHP* connects PHP with MySQL.

C and C++

Truth be told, not many clients are brave, patient or rich enough to underwrite development of an entire multi-user database application written in C or C++. Nevertheless, it often happens that a module needs the speed of one of these languages. If this is not true of you, at least now, you don't lose much by skipping over this chapter. On the other hand, you don't know when you'll need the information, do you?

Choosing an API

You may have the freedom to choose an API, or your current application may choose it for you, or your client may. When the choice is yours, you must consider several factors:

- *In-house developer skills* – what language do you and the in-house talent know?
- *Development schedule* – how quickly must you deliver the application? The higher the language level, the quicker the delivery. If hurry is the leading consideration, you can't beat LAMP.
- *Performance and scalability* – do you expect 10 users per day, or 10 million?
- *Supported platforms* – how many platforms must your application run on? Even if your answer is just an intranet or the web, you have to allow for significant browser differences. If you are deploying to a Windows-only shop, you can take liberties. If you have to anticipate Windows and *nix and Macintosh, you must ensure that whatever you do ports to each environment.

Summary

For every important programming language, the MySQL API is out there. With luck, our choices match your situation. In that event, focus on the chapter of interest and read the others when the need arises.

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