MySQL APIs

A main strength of MySQL is its Application Program Interface (API) system. MySQL has published open source APIs called *Connectors* for Java, .NET, ODBC, Python, C and C++. Connector/J is the Java API, Connector/C++ is the C++ API; Connector/NET opens MySQL to .NET languages including Visual Basic and C#, Connector/ODBC to Microsoft Access and any other language that can talk to ODBC.The C API can be the backbone of an API for any language; third parties have published APIs for Perl, PHP, Ruby, Delphi, Kylix, Eiffel and Arduino.

MySQL 5.5 also introduced a *MySQL Excel plugin*, which you obtain with the MySQL Installer (*Chapter 3*). It requires .NET Framework 4.0 client or full, Excel 2007, Visual Studio Tools for Office 4.0, and a MySQL server connection. Click on the Excel Data menu tab, then the MySQL for Excel icon, to open an Excel sidebar with MySQL options, which include appending, editing, importing and exporting MySQL data.

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.

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

- connect,
- issue queries to retrieve and store data,
- update user data views, interfaces and reports accordingly,
- disconnect.

The more you partition presentation from data management, and the more you design to *Use Cases*, the more useful the MySQL API system becomes, and the less crucial becomes your programming language. Development environments like Java and .NET encourage, even require 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 functional modules or objects.

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?

Database application language trends don't always follow *general trends*. Judging by the rate at which new questions appear in MySQL *help fora*, MySQL developers use the PHP API about twice as much as .NET, Java or ODBC, four times as much as C and C++, 15 times as much as Perl, 30 times as much as Python, and 60 times as much as Ruby. We take the PHP, .NET, ODBC, Java, Perl and C APIs to be essential. Python and Ruby are good for prototyping, but have performance issues.

The ODBC, Perl and PHP APIs have small function sets, so productivity comes quickly. 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 book*shelf*. 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

Write once and run everywhere or debug everywhere? Both, but whether an organisation builds around Windows or Linux, Java is there. Despite memory and complexity and security issues, Java is still a major web services 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, Connector/ODBC gives you access to databases trapped in SQL Server, Access, Approach or QuickBooks, and to deal with their data.

.NET

The .NET framework is impressive. So is Microsoft marketing. 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 to make this possible for Java too.

This is a powerful model, but with big drawbacks—it's Windows-centric, it's complex, and the verbosity of C# and C++ slow down development.

Perl and PHP

Most MySQL installations are part of a LAMP (Linux-Apache-MySQL-PHP/Perl) or WAMP (W for Windows) stack. As scripting languages, Perl and PHP execute more slowly than native code, but they are cross-platform, they have robust MySQL APIs, and they easily generate HTML. A big difference is that Perl syntax is obscure, whereas PHP syntax is seat-of-the-pants-user-friendly. For that reason and others, purists have long predicted that PHP would fade into legacy. It hasn't happened.

Given high-powered tools like Java and .NET, why not? Perhaps for the same reason Ali could knock out bigger opponents: LAMP is light enough on its feet to float like a butterfly and sting like a bee. With PHP, a useful website with database smarts and user validation can go from idea to deployment in *hours* rather than weeks, months or years.

C and C++

Truth be told, not many clients are brave, patient or rich enough to underwrite development of a multi-user database application written in C or C++, but it often happens that an application *module* needs the speed of C or C++. MySQL offers the *C API* and *Connector/C++*. If you don't need them 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 force your choice, or your client or employer 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? 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 write ports to each environment.

How do Java, PHP and .NET compare on platform versatility, development speed, availability of development tools, scaling, and maintainability? If your project needs to reach *Nix and MAC, .NET is not a leading option. GUI and RAD development tools for Java and .NET are much more sophisiticated and powerful than for PHP, and their O-O structure greatly improves maintainability. On the other hand, PHP shines for quick development, and many argue it actually scales better than Java.

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*** *Last updated 26 Sep 2016*