

MySQL and Windows

Abstract

This is the MySQL extract for Microsoft Windows from the MySQL 8.0 Reference Manual.

For legal information, see the [Legal Notices](#).

For help with using MySQL, please visit the [MySQL Forums](#), where you can discuss your issues with other MySQL users.

Document generated on: 2021-10-22 (revision: 71169)

Table of Contents

Preface and Legal Notices	v
1 Installing MySQL on Microsoft Windows	1
1.1 MySQL Installation Layout on Microsoft Windows	4
1.2 Choosing an Installation Package	4
1.3 MySQL Installer for Windows	5
1.3.1 MySQL Installer Initial Setup	7
1.3.2 Setting Alternative Server Paths with MySQL Installer	11
1.3.3 Installation Workflows with MySQL Installer	12
1.3.4 MySQL Installer Product Catalog and Dashboard	20
1.3.5 MySQLInstallerConsole Reference	26
1.4 Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive	31
1.4.1 Extracting the Install Archive	31
1.4.2 Creating an Option File	31
1.4.3 Selecting a MySQL Server Type	32
1.4.4 Initializing the Data Directory	33
1.4.5 Starting the Server for the First Time	33
1.4.6 Starting MySQL from the Windows Command Line	34
1.4.7 Customizing the PATH for MySQL Tools	35
1.4.8 Starting MySQL as a Windows Service	36
1.4.9 Testing The MySQL Installation	39
1.5 Troubleshooting a Microsoft Windows MySQL Server Installation	39
1.6 Windows Postinstallation Procedures	41
1.7 Windows Platform Restrictions	43
2 Upgrading MySQL on Windows	45
3 Connection to MySQL Server Failing on Windows	47
4 Resetting the Root Password: Windows Systems	49
5 MySQL for Visual Studio	51
5.1 General Information	51
5.1.1 New in Version 1.2	52
5.1.2 New in Version 2.0 (Development Release)	53
5.2 Installing MySQL for Visual Studio	58
5.3 Enabling the MySQL Toolbar	60
5.4 Making a Connection	61
5.4.1 Connect Using Server Explorer	63
5.4.2 Connect Using MySQL Connections Manager	65
5.5 Editing	66
5.5.1 MySQL SQL Editor	67
5.5.2 Code Editors	68
5.5.3 Editing Tables	70
5.5.4 Editing Views	76
5.5.5 Editing Indexes	78
5.5.6 Editing Foreign Keys	79
5.5.7 Editing Stored Procedures and Functions	80
5.5.8 Editing Triggers	82
5.6 MySQL Project Items	83
5.6.1 MySQL ASP.NET MVC Items	83
5.6.2 MySQL Windows Forms Items	92
5.7 MySQL Application Configuration Tool	94
5.7.1 Entity Framework	95
5.7.2 Web Providers	97
5.7.3 Using the MySQL Connection String Editor	101

5.8 MySQL Data Export Tool	102
5.9 DDL T4 Template Macro	111
5.10 Debugging Stored Procedures and Functions	112
5.11 MySQL for Visual Studio Frequently Asked Questions	123

Preface and Legal Notices

This is the MySQL extract for Microsoft Windows from the MySQL 8.0 Reference Manual.

Licensing information—MySQL 8.0. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 8.0, see the [MySQL 8.0 Commercial Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 8.0, see the [MySQL 8.0 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL NDB Cluster 8.0. If you are using a *Community* release of MySQL NDB Cluster 8.0, see the [MySQL NDB Cluster 8.0 Community Release License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—MySQL for Visual Studio. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL for Visual Studio, see the [MySQL for Visual Studio Commercial License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL for Visual Studio, see the [MySQL for Visual Studio Community License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 1997, 2021, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab>.

Chapter 1 Installing MySQL on Microsoft Windows

Table of Contents

1.1 MySQL Installation Layout on Microsoft Windows	4
1.2 Choosing an Installation Package	4
1.3 MySQL Installer for Windows	5
1.3.1 MySQL Installer Initial Setup	7
1.3.2 Setting Alternative Server Paths with MySQL Installer	11
1.3.3 Installation Workflows with MySQL Installer	12
1.3.4 MySQL Installer Product Catalog and Dashboard	20
1.3.5 <code>MySQLInstallerConsole</code> Reference	26
1.4 Installing MySQL on Microsoft Windows Using a <code>noinstall</code> ZIP Archive	31
1.4.1 Extracting the Install Archive	31
1.4.2 Creating an Option File	31
1.4.3 Selecting a MySQL Server Type	32
1.4.4 Initializing the Data Directory	33
1.4.5 Starting the Server for the First Time	33
1.4.6 Starting MySQL from the Windows Command Line	34
1.4.7 Customizing the PATH for MySQL Tools	35
1.4.8 Starting MySQL as a Windows Service	36
1.4.9 Testing The MySQL Installation	39
1.5 Troubleshooting a Microsoft Windows MySQL Server Installation	39
1.6 Windows Postinstallation Procedures	41
1.7 Windows Platform Restrictions	43

Important

MySQL 8.0 Server requires the Microsoft Visual C++ 2019 Redistributable Package to run on Windows platforms. Users should make sure the package has been installed on the system before installing the server. The package is available at the [Microsoft Download Center](https://www.microsoft.com/download/details.aspx?id=53349). Additionally, MySQL debug binaries require Visual Studio 2019 to be installed.

MySQL is available for Microsoft Windows 64-bit operating systems only. For supported Windows platform information, see <https://www.mysql.com/support/supportedplatforms/database.html>.

There are different methods to install MySQL on Microsoft Windows.

MySQL Installer Method

The simplest and recommended method is to download MySQL Installer (for Windows) and let it install and configure a specific version of MySQL Server as follows:

1. Download MySQL Installer from <https://dev.mysql.com/downloads/installer/> and execute it.

Note

Unlike the standard MySQL Installer, the smaller `web-community` version does not bundle any MySQL applications, but downloads only the MySQL products you choose to install.

2. Determine the setup type to use for the initial installation of MySQL products. For example:

- **Developer Default:** Provides a setup type that includes the selected version of MySQL Server and other MySQL tools related to MySQL development, such as MySQL Workbench.
 - **Server Only:** Provides a setup for the selected version of MySQL Server without other products.
 - **Custom:** Enables you to select any version of MySQL Server and other MySQL products.
3. Install the server instance (and products) and then begin the server configuration by following the onscreen instructions. For more information about each individual step, see [Section 1.3.3.1, “MySQL Server Configuration with MySQL Installer”](#).

MySQL is now installed. If you configured MySQL as a service, then Windows automatically starts the MySQL server every time you restart the system. Also, this process installs the MySQL Installer application on the local host, which you can use later to upgrade or reconfigure MySQL server.

Note

If you installed MySQL Workbench on your system, consider using it to check your new MySQL server connection. By default, the program automatically start after installing MySQL.

Additional Installation Information

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see [Section 1.4.8, “Starting MySQL as a Windows Service”](#).

To accommodate the `RESTART` statement, the MySQL server forks when run as a service or standalone, to enable a monitor process to supervise the server process. In this case, there are two `mysqld` processes. If `RESTART` capability is not required, the server can be started with the `--no-monitor` option. See [RESTART Statement](#).

Generally, you should install MySQL on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the `PATH` environment variable or accessing the [Service Control Manager](#). When installed, MySQL does not need to be executed using a user with Administrator privileges.

For a list of limitations on the use of MySQL on the Windows platform, see [Section 1.7, “Windows Platform Restrictions”](#).

In addition to the MySQL Server package, you may need or want additional components to use MySQL with your application or development environment. These include, but are not limited to:

- To connect to the MySQL server using ODBC, you must have a Connector/ODBC driver. For more information, including installation and configuration instructions, see [MySQL Connector/ODBC Developer Guide](#).

Note

MySQL Installer installs and configures Connector/ODBC for you.

- To use MySQL server with .NET applications, you must have the Connector/.NET driver. For more information, including installation and configuration instructions, see [MySQL Connector/.NET Developer Guide](#).

Note

MySQL Installer installs and configures MySQL Connector/NET for you.

MySQL distributions for Windows can be downloaded from <https://dev.mysql.com/downloads/>. See [How to Get MySQL](#).

MySQL for Windows is available in several distribution formats, detailed here. Generally speaking, you should use MySQL Installer. It contains more features and MySQL products than the older MSI, is simpler to use than the compressed file, and you need no additional tools to get MySQL up and running. MySQL Installer automatically installs MySQL Server and additional MySQL products, creates an options file, starts the server, and enables you to create default user accounts. For more information on choosing a package, see [Section 1.2, “Choosing an Installation Package”](#).

- A MySQL Installer distribution includes MySQL Server and additional MySQL products including MySQL Workbench, and MySQL for Visual Studio. MySQL Installer can also be used to upgrade these products in the future (see <https://dev.mysql.com/doc/mysql-compat-matrix/en/>).

For instructions on installing MySQL using MySQL Installer, see [Section 1.3, “MySQL Installer for Windows”](#).

- The standard binary distribution (packaged as a compressed file) contains all of the necessary files that you unpack into your chosen location. This package contains all of the files in the full Windows MSI Installer package, but does not include an installation program.

For instructions on installing MySQL using the compressed file, see [Section 1.4, “Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive”](#).

- The source distribution format contains all the code and support files for building the executables using the Visual Studio compiler system.

For instructions on building MySQL from source on Windows, see [Installing MySQL from Source](#).

MySQL on Windows Considerations

- **Large Table Support**

If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use `MAX_ROWS` and `AVG_ROW_LENGTH` when you create tables. See [CREATE TABLE Statement](#).

- **MySQL and Virus Checking Software**

Virus-scanning software such as Norton/Symantec Anti-Virus on directories containing MySQL data and temporary tables can cause issues, both in terms of the performance of MySQL and the virus-scanning software misidentifying the contents of the files as containing spam. This is due to the fingerprinting mechanism used by the virus-scanning software, and the way in which MySQL rapidly updates different files, which may be identified as a potential security risk.

After installing MySQL Server, it is recommended that you disable virus scanning on the main directory (`datadir`) used to store your MySQL table data. There is usually a system built into the virus-scanning software to enable specific directories to be ignored.

In addition, by default, MySQL creates temporary files in the standard Windows temporary directory. To prevent the temporary files also being scanned, configure a separate temporary directory for

MySQL temporary files and add this directory to the virus scanning exclusion list. To do this, add a configuration option for the `tmpdir` parameter to your `my.ini` configuration file. For more information, see [Section 1.4.2, “Creating an Option File”](#).

1.1 MySQL Installation Layout on Microsoft Windows

For MySQL 8.0 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.0` for installations performed with MySQL Installer. If you use the ZIP archive method to install MySQL, you may prefer to install in `C:\mysql`. However, the layout of the subdirectories remains the same.

All of the files are located within this parent directory, using the structure shown in the following table.

Table 1.1 Default MySQL Installation Layout for Microsoft Windows

Directory	Contents of Directory	Notes
<code>bin</code>	<code>mysqld</code> server, client and utility programs	
<code>%PROGRAMDATA%\MySQL\MySQL Server 8.0\</code>	Log files, databases	The Windows system variable <code>%PROGRAMDATA%</code> defaults to <code>C:\ProgramData</code> .
<code>docs</code>	Release documentation	With MySQL Installer, use the Modify operation to select this optional folder.
<code>include</code>	Include (header) files	
<code>lib</code>	Libraries	
<code>share</code>	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

1.2 Choosing an Installation Package

For MySQL 8.0, there are multiple installation package formats to choose from when installing MySQL on Windows. The package formats described in this section are:

- [MySQL Installer](#)
- [MySQL noinstall ZIP Archives](#)
- [MySQL Docker Images](#)

Program Database (PDB) files (with file name extension `pdb`) provide information for debugging your MySQL installation in the event of a problem. These files are included in ZIP Archive distributions (but not MSI distributions) of MySQL.

MySQL Installer

This package has a file name similar to `mysql-installer-community-8.0.27.0.msi` or `mysql-installer-commercial-8.0.27.0.msi`, and utilizes MSIs to install MySQL server and other products

automatically. MySQL Installer downloads and applies updates to itself, and to each of the installed products. It also configures the installed MySQL server (including a sandbox InnoDB cluster test setup) and MySQL Router. MySQL Installer is recommended for most users.

MySQL Installer can install and manage (add, modify, upgrade, and remove) many other MySQL products, including:

- Applications – MySQL Workbench, MySQL for Visual Studio, MySQL Shell, and MySQL Router (see <https://dev.mysql.com/doc/mysql-compat-matrix/en/>)
- Connectors – MySQL Connector/C++, MySQL Connector/NET, Connector/ODBC, MySQL Connector/Python, MySQL Connector/J, MySQL Connector/Node.js
- Documentation – MySQL Manual (PDF format), samples and examples

MySQL Installer operates on all MySQL supported versions of Windows (see <https://www.mysql.com/support/supportedplatforms/database.html>).

Note

Because MySQL Installer is not a native component of Microsoft Windows and depends on .NET, it does not work with minimal installation options like the Server Core version of Windows Server.

For instructions on how to install MySQL using MySQL Installer, see [Section 1.3, “MySQL Installer for Windows”](#).

MySQL noinstall ZIP Archives

These packages contain the files found in the complete MySQL Server installation package, with the exception of the GUI. This format does not include an automated installer, and must be manually installed and configured.

The `noinstall` ZIP archives are split into two separate compressed files. The main package is named `mysql-VERSION-winx64.zip`. This contains the components needed to use MySQL on your system. The optional MySQL test suite, MySQL benchmark suite, and debugging binaries/information components (including PDB files) are in a separate compressed file named `mysql-VERSION-winx64-debug-test.zip`.

If you choose to install a `noinstall` ZIP archive, see [Section 1.4, “Installing MySQL on Microsoft Windows Using a noinstall ZIP Archive”](#).

MySQL Docker Images

For information on using the MySQL Docker images provided by Oracle on Windows platform, see [Deploying MySQL on Windows and Other Non-Linux Platforms with Docker](#).

Warning

The MySQL Docker images provided by Oracle are built specifically for Linux platforms. Other platforms are not supported, and users running the MySQL Docker images from Oracle on them are doing so at their own risk.

1.3 MySQL Installer for Windows

MySQL Installer is a standalone application designed to ease the complexity of installing and configuring MySQL products that run on Microsoft Windows. It supports the following MySQL products:

- MySQL Servers

MySQL Installer can install and manage multiple, separate MySQL server instances on the same host at the same time. For example, MySQL Installer can install, configure, and upgrade a separate instance of MySQL 5.6, MySQL 5.7, and MySQL 8.0 on the same host. MySQL Installer does not permit server upgrades between major and minor version numbers, but does permit upgrades within a release series (such as 8.0.21 to 8.0.22).

Note

MySQL Installer cannot install both *Community* and *Commercial* releases of MySQL server on the same host. If you require both releases on the same host, consider using the [ZIP archive](#) distribution to install one of the releases.

- MySQL Applications

MySQL Workbench, MySQL Shell, MySQL Router, and MySQL for Visual Studio.

- MySQL Connectors

MySQL Connector/NET, MySQL Connector/Python, MySQL Connector/ODBC, MySQL Connector/J, and MySQL Connector/C++. To install MySQL Connector/Node.js, see <https://dev.mysql.com/downloads/connector/nodejs/>.

- Documentation and Samples

MySQL Reference Manuals (by version) in PDF format and MySQL database samples (by version).

Installation Requirements

MySQL Installer requires Microsoft .NET Framework 4.5.2 or later. If this version is not installed on the host computer, you can download it by visiting the [Microsoft website](#).

An internet connection is required to download a manifest containing metadata for the latest MySQL products that are not part of a full bundle. MySQL Installer attempts to download the manifest when you start the application for the first time and then periodically in configurable intervals (see [MySQL Installer options](#)). Alternatively, you can retrieve an updated manifest manually by clicking **Catalog** in the [MySQL Installer dashboard](#).

Note

If the first-time or subsequent manifest download is unsuccessful, an error is logged and you may have limited access to MySQL products during your session. MySQL Installer attempts to download the manifest with each startup until the initial manifest structure is updated. For help finding a product, see [Locating Products to Install](#).

MySQL Installer Community Release

Download software from <https://dev.mysql.com/downloads/installer/> to install the Community release of all MySQL products for Windows. Select one of the following MySQL Installer package options:

- *Web*: Contains MySQL Installer and configuration files only. The web package option downloads only the MySQL products you select to install, but it requires an internet connection for each download.

The size of this file is approximately 2 MB. The file name has the form `mysql-installer-community-web-VERSION.N.msi` in which `VERSION` is the MySQL server version number such as 8.0 and `N` is the package number, which begins at 0.

- *Full or Current Bundle*: Bundles all of the MySQL products for Windows (including the MySQL server). The file size is over 300 MB, and the name has the form `mysql-installer-community-VERSION.N.msi` in which `VERSION` is the MySQL Server version number such as 8.0 and `N` is the package number, which begins at 0.

MySQL Installer Commercial Release

Download software from <https://edelivery.oracle.com/> to install the Commercial release (Standard or Enterprise Edition) of MySQL products for Windows. If you are logged in to your My Oracle Support (MOS) account, the Commercial release includes all of the current and previous GA versions available in the Community release, but it excludes development-milestone versions. When you are not logged in, you see only the list of bundled products that you downloaded already.

The Commercial release also includes the following products:

- Workbench SE/EE
- MySQL Enterprise Backup
- MySQL Enterprise Firewall

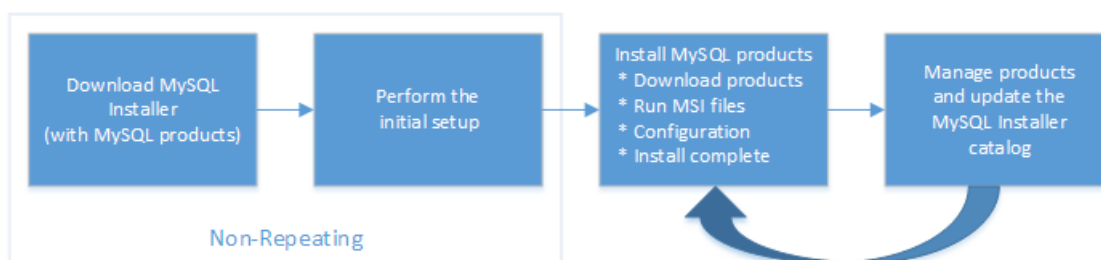
The Commercial release integrates with your MOS account. For knowledge-base content and patches, see [My Oracle Support](#).

1.3.1 MySQL Installer Initial Setup

- [Choosing a Setup Type](#)
- [Path Conflicts](#)
- [Check Requirements](#)
- [MySQL Installer Configuration Files](#)

When you download MySQL Installer for the first time, a setup wizard guides you through the initial installation of MySQL products. As the following figure shows, the initial setup is a one-time activity in the overall process. MySQL Installer detects existing MySQL products installed on the host during its initial setup and adds them to the list of products to be managed.

Figure 1.1 MySQL Installer Process Overview



MySQL Installer extracts configuration files (described later) to the hard drive of the host during the initial setup. Although MySQL Installer is a 32-bit application, it can install both 32-bit and 64-bit binaries.

The initial setup adds a link to the Start menu under the **MySQL** group. Click **Start, All Programs, MySQL, MySQL Installer** to open MySQL Installer.

Choosing a Setup Type

During the initial setup, you are prompted to select the MySQL products to be installed on the host. One alternative is to use a predetermined setup type that matches your setup requirements. By default, both GA and pre-release products are included in the download and installation with the **Developer Default, Client only**, and **Full** setup types. Select the **Only install GA products** option to restrict the product set to include GA products only when using these setup types.

Choosing one of the following setup types determines the initial installation only and does not limit your ability to install or update MySQL products for Windows later:

- **Developer Default:** Install the following products that compliment application development with MySQL:
 - [MySQL Server](#) (Installs the version that you selected when you downloaded MySQL Installer.)
 - [MySQL Shell](#)
 - [MySQL Router](#)
 - [MySQL Workbench](#)
 - [MySQL for Visual Studio](#)
 - [MySQL Connectors](#) (for .NET / Python / ODBC / Java / C++)
 - MySQL Documentation
 - MySQL Samples and Examples
- **Server only:** Only install the MySQL server. This setup type installs the general availability (GA) or development release server that you selected when you downloaded MySQL Installer. It uses the default installation and data paths.
- **Client only:** Only install the most recent MySQL applications and MySQL connectors. This setup type is similar to the [Developer Default](#) type, except that it does not include MySQL server or the client programs typically bundled with the server, such as `mysql` or `mysqladmin`.
- **Full:** Install all available MySQL products.
- **Custom:** The custom setup type enables you to filter and select individual MySQL products from the [MySQL Installer catalog](#).

Note

For MySQL Server versions 8.0.20 (and earlier), 5.7, and 5.6, the account you use to run MySQL Installer may not have adequate permission to install the server data files and this can interrupt the installation because the [ExecSecureObjects](#) MSI action cannot be executed. To proceed, deselect the **Server data files** feature before attempting to install the server again. For help, see [Product Features To Install](#).

The **Server data files** check box was removed from the feature tree for MySQL Server 8.0.21 (and higher).

Use the [Custom](#) setup type to install:

- A product or product version that is not available from the usual download locations. The catalog contains all product releases, including the other releases between pre-release (or development) and GA.
- An instance of MySQL server using an alternative installation path, data path, or both. For instructions on how to adjust the paths, see [Section 1.3.2, “Setting Alternative Server Paths with MySQL Installer”](#).
- Two or more MySQL server versions on the same host at the same time (for example, 5.6, 5.7, and 8.0).
- A specific combination of products and features not offered as a predetermined setup type. For example, you can install a single product, such as MySQL Workbench, instead of installing all client applications for Windows.

Path Conflicts

When the default installation or data folder (required by MySQL server) for a product to be installed already exists on the host, the wizard displays the **Path Conflict** step to identify each conflict and enable you to take action to avoid having files in the existing folder overwritten by the new installation. You see this step in the initial setup only when MySQL Installer detects a conflict.

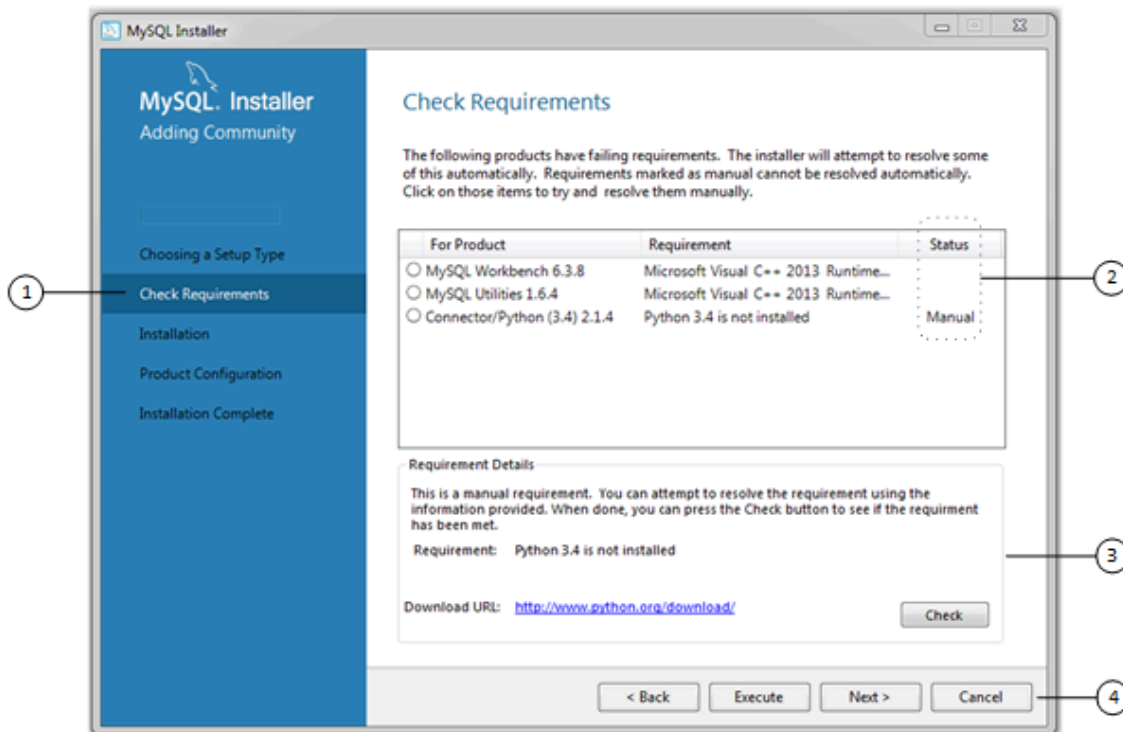
To resolve the path conflict, do one of the following:

- Select a product from the list to display the conflict options. A warning symbol indicates which path is in conflict. Use the browse button to choose a new path and then click **Next**.
- Click **Back** to choose a different setup type or product version, if applicable. The [Custom](#) setup type enables you to select individual product versions.
- Click **Next** to ignore the conflict and overwrite files in the existing folder.
- Delete the existing product. Click **Cancel** to stop the initial setup and close MySQL Installer. Open MySQL Installer again from the Start menu and delete the installed product from the host using the Delete operation from the [MySQL Installer dashboard](#).

Check Requirements

MySQL Installer uses entries in the `package-rules.xml` file to determine whether the prerequisite software for each product is installed on the host. When the requirements check fails, MySQL Installer displays the **Check Requirements** step to help you update the host. Requirements are evaluated each time you download a new product (or version) for installation. The following figure identifies and describes the key areas of this step.

Figure 1.2 Check Requirements



Description of Check Requirements Elements

- Shows the current step in the initial setup. Steps in this list may change slightly depending on the products already installed on the host, the availability of prerequisite software, and the products to be installed on the host.
- Lists all pending installation requirements by product and indicates the status as follows:
 - A blank space in the **Status** column means that MySQL Installer can attempt to download and install the required software for you.
 - The word *Manual* in the **Status** column means that you must satisfy the requirement manually. Select each product in the list to see its requirement details.
- Describes the requirement in detail to assist you with each manual resolution. When possible, a download URL is provided. After you download and install the required software, click **Check** to verify that the requirement has been met.
- Provides the following set operations to proceed:
 - Back** – Return to the previous step. This action enables you to select a different the setup type.
 - Execute** – Have MySQL Installer attempt to download and install the required software for all items without a manual status. Manual requirements are resolved by you and verified by clicking **Check**.
 - Next** – Do not execute the request to apply the requirements automatically and proceed to the installation without including the products that fail the check requirements step.

- **Cancel** – Stop the installation of MySQL products. Because MySQL Installer is already installed, the initial setup begins again when you open MySQL Installer from the Start menu and click **Add** from the dashboard. For a description of the available management operations, see [Product Catalog](#).

MySQL Installer Configuration Files

All MySQL Installer files are located within the `C:\Program Files (x86)` and `C:\ProgramData` folders. The following table describes the files and folders that define MySQL Installer as a standalone application.

Note

Installed MySQL products are neither altered nor removed when you update or uninstall MySQL Installer.

Table 1.2 MySQL Installer Configuration Files

File or Folder	Description	Folder Hierarchy
MySQL Installer for Windows	This folder contains all of the files needed to run MySQL Installer and <code>MySQLInstallerConsole.exe</code> , a command-line program with similar functionality.	<code>C:\Program Files (x86)</code>
Templates	The <code>Templates</code> folder has one file for each version of MySQL server. Template files contain keys and formulas to calculate some values dynamically.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
<code>package-rules.xml</code>	This file contains the prerequisites for every product to be installed.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
<code>products.xml</code>	The <code>products</code> file (or product catalog) contains a list of all products available for download.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows\Manifest</code>
Product Cache	The <code>Product Cache</code> folder contains all standalone <code>.msi</code> files bundled with the full package or downloaded afterward.	<code>C:\ProgramData\MySQL\MySQL Installer for Windows</code>

1.3.2 Setting Alternative Server Paths with MySQL Installer

You can change the default installation path, the data path, or both when you install MySQL server. After you have installed the server, the paths cannot be altered without removing and reinstalling the server instance.

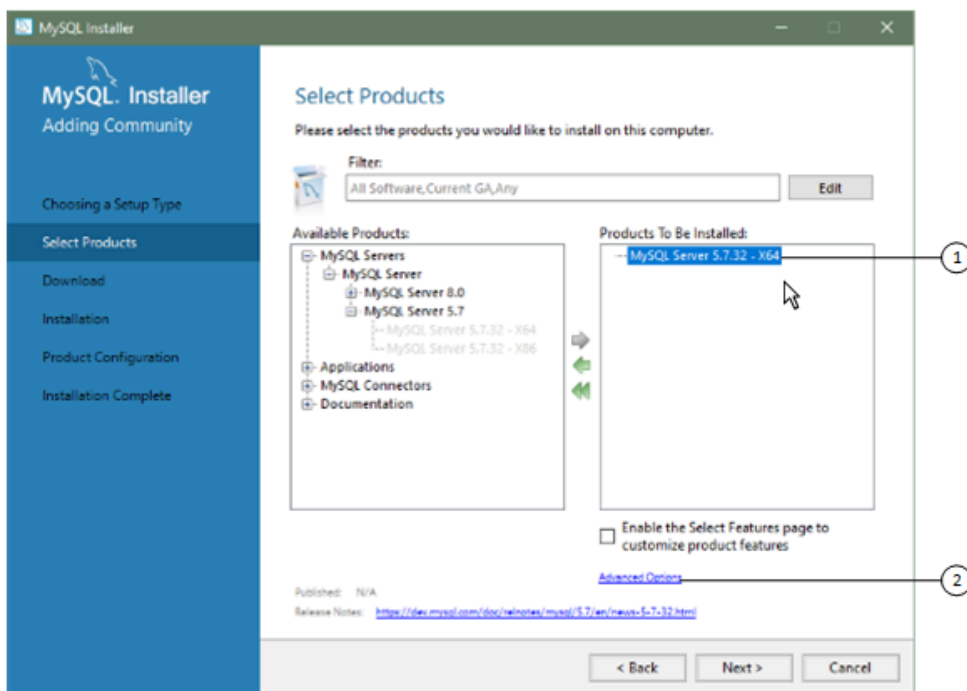
Note

Starting with MySQL Installer 1.4.39, if you move the data directory of an installed server manually, MySQL Installer identifies the change and can process a reconfiguration operation without errors.

To change paths for MySQL server

1. Identify the MySQL server to change and enable the **Advanced Options** link as follows:
 - a. Navigate to the **Select Products** page by doing one of the following:
 - i. If this is an [initial setup](#) of MySQL Installer, select the [Custom](#) setup type and click **Next**.
 - ii. If MySQL Installer is installed on your computer, click **Add** from the dashboard.
 - b. Click **Edit** to apply a filter on the product list shown in **Available Products** (see [Locating Products to Install](#)).
 - c. With the server instance selected, use the arrow to move the selected server to the **Products To Be Installed** list.
 - d. Click the server to select it. When you select the server, the **Advanced Options** link is enabled below the list of products to be installed (see the following figure).
2. Click **Advanced Options** to open a dialog box where you can enter alternative path names. After the path names are validated, click **Next** to continue with the configuration steps.

Figure 1.3 Change MySQL Server Path



1.3.3 Installation Workflows with MySQL Installer

MySQL Installer provides a wizard-like tool to install and configure new MySQL products for Windows. Unlike the initial setup, which runs only once, MySQL Installer invokes the wizard each time you download or install a new product. For first-time installations, the steps of the initial setup proceed directly into the steps of the installation. For assistance with product selection, see [Locating Products to Install](#).

Note

Full permissions are granted to the user executing MySQL Installer to all generated files, such as `my.ini`. This does not apply to files and directories for specific

products, such as the MySQL server data directory in `%ProgramData%` that is owned by `SYSTEM`.

Products installed and configured on a host follow a general pattern that might require your input during the various steps. If you attempt to install a product that is incompatible with the existing MySQL server version (or a version selected for upgrade), you are alerted about the possible mismatch.

MySQL Installer provides the following sequence of actions that apply to different workflows:

- **Select Products.** If you selected the `Custom` setup type during the initial setup or clicked **Add** from the [MySQL Installer dashboard](#), MySQL Installer includes this action in the sidebar. From this page, you can apply a filter to modify the Available Products list and then select one or more products to move (using arrow keys) to the Products To Be Installed list.

Select the check box on this page to activate the Select Features action where you can customize the products features after the product is downloaded.

- **Download.** If you installed the full (not web) MySQL Installer package, all `.msi` files were loaded to the `Product Cache` folder during the initial setup and are not downloaded again. Otherwise, click **Execute** to begin the download. The status of each product changes from `Ready to Download`, to `Downloading`, and then to `Downloaded`.
- **Select Features To Install (disabled by default).** After MySQL Installer downloads a product's `.msi` file, you can customize the features if you enabled the optional check box previously during the Select Products action.

To customize product features after the installation, click **Modify** in the [MySQL Installer dashboard](#).

- **Installation.** The status of each product in the list changes from `Ready to Install`, to `Installing`, and lastly to `Complete`. During the process, click **Show Details** to view the installation actions.

If you cancel the installation at this point, the products are installed, but the server (if installed) is not yet configured. To restart the server configuration, open MySQL Installer from the Start menu and click **Reconfigure** next to the appropriate server in the dashboard.

- **Product configuration.** This step applies to MySQL Server, MySQL Router, and samples only. The status for each item in the list should indicate `Ready to Configure`. Click **Next** to start the configuration wizard for all items in the list. The configuration options presented during this step are specific to the version of database or router that you selected to install.

Click **Execute** to begin applying the configuration options or click **Back** (repeatedly) to return to each configuration page.

- **Installation complete.** This step finalizes the installation for products that do not require configuration. It enables you to copy the log to a clipboard and to start certain applications, such as MySQL Workbench and MySQL Shell. Click **Finish** to open the [MySQL Installer dashboard](#).

1.3.3.1 MySQL Server Configuration with MySQL Installer

MySQL Installer performs the initial configuration of the MySQL server. For example:

- It creates the configuration file (`my.ini`) that is used to configure the MySQL server. The values written to this file are influenced by choices you make during the installation process. Some definitions are host dependent. For example, `query_cache` is enabled if the host has fewer than three cores.

Note

Query cache was deprecated in MySQL 5.7 and removed in MySQL 8.0 (and later).

- By default, a Windows service for the MySQL server is added.
- Provides default installation and data paths for MySQL server. For instructions on how to change the default paths, see [Section 1.3.2, “Setting Alternative Server Paths with MySQL Installer”](#).
- It can optionally create MySQL server user accounts with configurable permissions based on general roles, such as DB Administrator, DB Designer, and Backup Admin. It optionally creates a Windows user named `MySQLSys` with limited privileges, which would then run the MySQL Server.

User accounts may also be added and configured in MySQL Workbench.

- Checking **Show Advanced Options** enables additional **Logging Options** to be set. This includes defining custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log.

During the configuration process, click **Next** to proceed to the next step or **Back** to return to the previous step. Click **Execute** at the final step to apply the server configuration.

The sections that follow describe the server configuration options that apply to MySQL server on Windows. The server version you installed will determine which steps and options you can configure. Configuring MySQL server may include some or all of the steps.

Type and Networking


- Server Configuration Type

Choose the MySQL server configuration type that describes your setup. This setting defines the amount of system resources (memory) to assign to your MySQL server instance.

- **Development:** A computer that hosts many other applications, and typically this is your personal workstation. This setting configures MySQL to use the least amount of memory.
 - **Server:** Several other applications are expected to run on this computer, such as a web server. The Server setting configures MySQL to use a medium amount of memory.
 - **Dedicated:** A computer that is dedicated to running the MySQL server. Because no other major applications run on this server, this setting configures MySQL to use the majority of available memory.
- Connectivity

Connectivity options control how the connection to MySQL is made. Options include:

- **TCP/IP:** This option is selected by default. You may disable TCP/IP Networking to permit local host connections only. With the TCP/IP connection option selected, you can modify the following items:
 - **Port** for classic MySQL protocol connections. The default value is `3306`.
 - **X Protocol Port** shown when configuring MySQL 8.0 server only. The default value is `33060`
 - **Open Windows Firewall port for network access**, which is selected by default for TCP/IP connections.

If a port number is in use already, you will see the information icon () next to the default value and **Next** is disabled until you provide a new port number.

- **Named Pipe:** Enable and define the pipe name, similar to setting the `named_pipe` system variable. The default name is `MySQL`.
- **Shared Memory:** Enable and define the memory name, similar to setting the `shared_memory` system variable. The default name is `MySQL`.
- Advanced Configuration

Check **Show Advanced and Logging Options** to set custom logging and advanced options in later steps. The Logging Options step enables you to define custom file paths for the error log, general log, slow query log (including the configuration of seconds it requires to execute a query), and the binary log. The Advanced Options step enables you to set the unique server ID required when binary logging is enabled in a replication topology.

- MySQL Enterprise Firewall (Enterprise Edition only)

The **Enable MySQL Enterprise Firewall** check box is deselected by default. Select this option to enable a security list that offers protection against certain types of attacks. Additional post-installation configuration is required (see [MySQL Enterprise Firewall](#)).

Important

There is an issue for MySQL 8.0.19 that prevents the server from starting if MySQL Enterprise Firewall is selected during the server configuration steps. If the server startup operation fails, click **Cancel** to end the configuration process and return to the dashboard. You must uninstall the server.

The workaround is to run MySQL Installer without MySQL Enterprise Firewall selected. (That is, do not select the **Enable MySQL Enterprise Firewall** check box.) Then install MySQL Enterprise Firewall afterward using the instructions for manual installation (see [Installing or Uninstalling MySQL Enterprise Firewall](#)).

Authentication Method

The **Authentication Method** step is visible only during the installation or upgrade of MySQL 8.0.4 or higher. It introduces a choice between two server-side authentication options. The MySQL user accounts that you create in the next step will use the authentication method that you select in this step.

MySQL 8.0 connectors and community drivers that use `libmysqlclient` 8.0 now support the `mysql_native_password` default authentication plugin. However, if you are unable to update your clients and applications to support this new authentication method, you can configure the MySQL server to use `mysql_native_password` for legacy authentication. For more information about the implications of this change, see [caching_sha2_password as the Preferred Authentication Plugin](#).

If you are installing or upgrading to MySQL 8.0.4 or higher, select one of the following authentication methods:

- Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8.0 supports a new authentication based on improved, stronger SHA256-based password methods. It is recommended that all new MySQL server installations use this method going forward.

Important

The `caching_sha2_password` authentication plugin on the server requires new versions of connectors and clients, which add support for the new MySQL 8.0 default authentication.

- Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)


Using the old MySQL 5.x legacy authentication method should be considered only in the following cases:

- Applications cannot be updated to use MySQL 8.0 connectors and drivers.
- Recompilation of an existing application is not feasible.
- An updated, language-specific connector or driver is not available yet.

Accounts and Roles

- Root Account Password

Assigning a root password is required and you will be asked for it when performing other MySQL Installer operations. Password strength is evaluated when you repeat the password in the box provided. For descriptive information regarding password requirements or status, move your mouse pointer over

the information icon () when it appears.

- MySQL User Accounts (Optional)

Click **Add User** or **Edit User** to create or modify MySQL user accounts with predefined roles. Next, enter the required account credentials:

- **User Name:** MySQL user names can be up to 32 characters long.
- **Host:** Select `localhost` for local connections only or `<All Hosts (%>` when remote connections to the server are required.
- **Role:** Each predefined role, such as `DB Admin`, is configured with its own set of privileges. For example, the `DB Admin` role has more privileges than the `DB Designer` role. The **Role** drop-down list contains a description of each role.
- **Password:** Password strength assessment is performed while you type the password. Passwords must be confirmed. MySQL permits a blank or empty password (considered to be insecure).

MySQL Installer Commercial Release Only: MySQL Enterprise Edition for Windows, a commercial product, also supports an authentication method that performs external authentication on Windows. Accounts authenticated by the Windows operating system can access the MySQL server without providing an additional password.

To create a new MySQL account that uses Windows authentication, enter the user name and then select a value for **Host** and **Role**. Click **Windows** authentication to enable the `authentication_windows` plugin. In the Windows Security Tokens area, enter a token for each Windows user (or group) who can authenticate with the MySQL user name. MySQL accounts can include security tokens for both local Windows users and Windows users that belong to a domain. Multiple security tokens are separated by the semicolon character (`;`) and use the following format for local and domain accounts:

- Local account

Enter the simple Windows user name as the security token for each local user or group; for example, `finley;jeffrey;admin`.

- Domain account

Use standard Windows syntax (`domain\domainuser`) or MySQL syntax (`domain\domainuser`) to enter Windows domain users and groups.

For domain accounts, you may need to use the credentials of an administrator within the domain if the account running MySQL Installer lacks the permissions to query the Active Directory. If this is the case, select **Validate Active Directory users with** to activate the domain administrator credentials.

Windows authentication permits you to test all of the security tokens each time you add or modify a token. Click **Test Security Tokens** to validate (or revalidate) each token. Invalid tokens generate a descriptive error message along with a red `x` icon and red token text. When all tokens resolve as valid (green text without an `x` icon), you can click **OK** to save the changes.

Windows Service

On the Windows platform, MySQL server can run as a named service managed by the operating system and be configured to start up automatically when Windows starts. Alternatively, you can configure MySQL server to run as an executable program that requires manual configuration.

- **Configure MySQL server as a Windows service** (Selected by default.)

When the default configuration option is selected, you can also select the following:

- **Start the MySQL Server at System Startup**

When selected (default), the service startup type is set to Automatic; otherwise, the startup type is set to Manual.

- **Run Windows Service as**

When **Standard System Account** is selected (default), the service logs on as Network Service.

The **Custom User** option must have privileges to log on to Microsoft Windows as a service. The **Next** button will be disabled until this user is configured with the required privileges.

A custom user account is configured in Windows by searching for "local security policy" in the Start menu. In the Local Security Policy window, select **Local Policies, User Rights Assignment**, and then **Log On As A Service** to open the property dialog. Click **Add User or Group** to add the custom user and then click **OK** in each dialog to save the changes.

- Deselect the Windows Service option

Logging Options

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

Advanced configuration options are related to the following MySQL log files:

- [Error Log](#)
- [General Log](#)

- [Slow Query Log](#)
- [Bin Log](#)

Note

The binary log is enabled by default for MySQL 5.7 and higher.

Advanced Options

This step is available if the **Show Advanced Configuration** check box was selected during the **Type and Networking** step. To enable this step now, click **Back** to return to the **Type and Networking** step and select the check box.

The advanced-configuration options include:

- **Server ID**

Set the unique identifier used in a replication topology. If binary logging is enabled, you must specify a server ID. The default ID value depends on the server version. For more information, see the description of the [server_id](#) system variable.

- **Table Names Case**

You can set the following options during the initial and subsequent configuration the server. For the MySQL 8.0 release series, these options apply only to the initial configuration of the server.

- Lower Case

Sets the [lower_case_table_names](#) option value to 1 (default), in which table names are stored in lowercase on disk and comparisons are not case-sensitive.

- Preserve Given Case

Sets the [lower_case_table_names](#) option value to 2, in which table names are stored as given but compared in lowercase.

Apply Server Configuration

All configuration settings are applied to the MySQL server when you click **Execute**. Use the **Configuration Steps** tab to follow the progress of each action; the icon for each toggles from white to green (with a check mark) on success. Otherwise, the process stops and displays an error message if an individual action times out. Click the **Log** tab to view the log.

When the installation completes successfully and you click **Finish**, MySQL Installer and the installed MySQL products are added to the Microsoft Windows Start menu under the [MySQL](#) group. Opening MySQL Installer loads the [dashboard](#) where installed MySQL products are listed and other MySQL Installer operations are available.

1.3.3.2 MySQL Router Configuration with MySQL Installer

MySQL Installer downloads and installs a suite of tools for developing and managing business-critical applications on Windows. The suite consists of applications, connectors, documentation, and samples.

During the [initial setup](#), choose any predetermined setup type, except [Server only](#), to install the latest GA version of the tools. Use the [Custom](#) setup type to install an individual tool or specific version. If

MySQL Installer is installed on the host already, use the **Add** operation to select and install tools from the MySQL Installer dashboard.

MySQL Router Configuration

MySQL Installer provides a configuration wizard that can bootstrap an installed instance of MySQL Router 8.0 to direct traffic between MySQL applications and an InnoDB Cluster. When configured, MySQL Router runs as a local Windows service.

Note

You are prompted to configure MySQL Router after the initial installation and when you reconfigure an installed router explicitly. In contrast, the upgrade operation does not require or prompt you to configure the upgraded product.

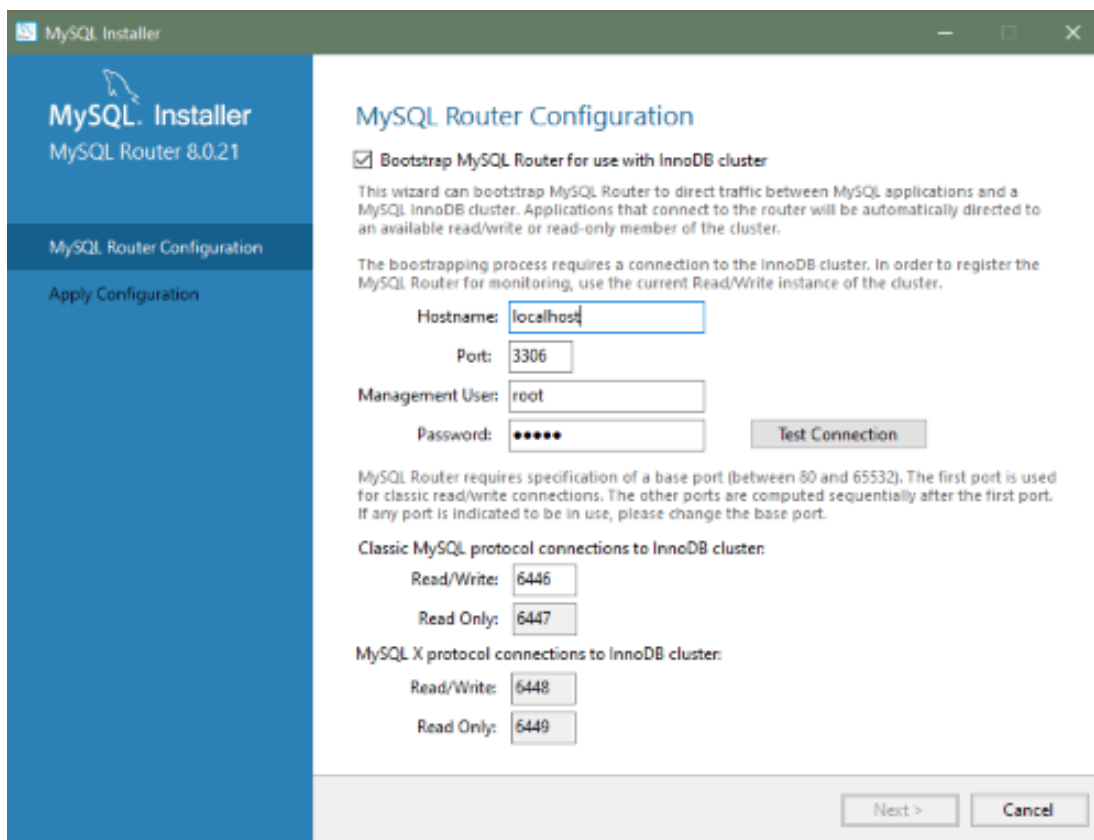
To configure MySQL Router, do the following:

1. Set up InnoDB Cluster.
2. Using MySQL Installer, download and install the MySQL Router application. After the installation finishes, the configuration wizard prompts you for information. Select the **Configure MySQL Router for InnoDB Cluster** check box to begin the configuration and provide the following configuration values:
 - **Hostname:** Host name of the primary (seed) server in the InnoDB Cluster (`localhost` by default).
 - **Port:** The port number of the primary (seed) server in the InnoDB Cluster (`3306` by default).
 - **Management User:** An administrative user with root-level privileges.
 - **Password:** The password for the management user.
 - **Classic MySQL protocol connections to InnoDB Cluster**

Read/Write: Set the first base port number to one that is unused (between 80 and 65532) and the wizard will select the remaining ports for you.

The figure that follows shows an example of the MySQL Router configuration page, with the first base port number specified as 6446 and the remaining ports set by the wizard to 6447, 6448, and 6449.

Figure 1.4 MySQL Router Configuration



3. Click **Next** and then **Execute** to apply the configuration. Click **Finish** to close MySQL Installer or return to the [MySQL Installer dashboard](#).

After configuring MySQL Router, the root account exists in the user table as `root@localhost` (local) only, instead of `root@%` (remote). Regardless of where the router and client are located, even if both are located on the same host as the seed server, any connection that passes through the router is viewed by server as being remote, not local. As a result, a connection made to the server using the local host (see the example that follows), does not authenticate.

```
$> \c root@localhost:6446
```

1.3.4 MySQL Installer Product Catalog and Dashboard

This section describes the MySQL Installer product catalog, the dashboard, and other actions related to product selection and upgrades.

- [Product Catalog](#)
- [MySQL Installer Dashboard](#)
- [Locating Products to Install](#)
- [Upgrading MySQL Server](#)
- [Removing MySQL Server](#)

- [Upgrading MySQL Installer](#)

Product Catalog

The product catalog stores the complete list of released MySQL products for Microsoft Windows that are available to download from [MySQL Downloads](#). By default, and when an Internet connection is present, MySQL Installer attempts to update the catalog at startup every seven days. You can also update the catalog manually from the dashboard (described later).

An up-to-date catalog performs the following actions:

- Populates the **Available Products** pane of the Select Products page. This step appears when you select:
 - The `Custom` setup type during the [initial setup](#).
 - The **Add** operation from the dashboard.
- Identifies when product updates are available for the installed products listed in the dashboard.

The catalog includes all development releases (Pre-Release), general releases (Current GA), and minor releases (Other Releases). Products in the catalog will vary somewhat, depending on the MySQL Installer release that you download.

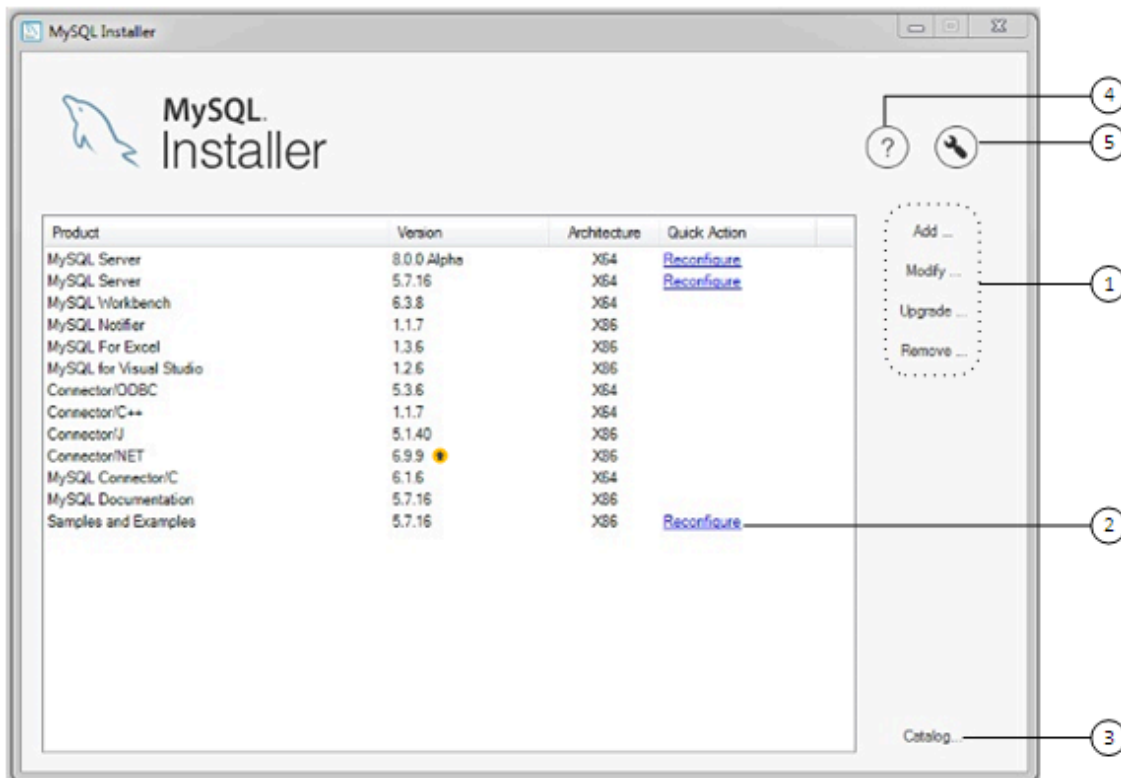
MySQL Installer Dashboard

The MySQL Installer dashboard is the default view that you see when you start MySQL Installer after the [initial setup](#) finishes. If you closed MySQL Installer before the setup was finished, MySQL Installer resumes the initial setup before it displays the dashboard.

Note

Products covered under Oracle Lifetime Sustaining Support, if installed, may appear in the dashboard. These products, such as MySQL for Excel and MySQL Notifier, can be modified or removed only.

Figure 1.5 MySQL Installer Dashboard Elements



Description of MySQL Installer Dashboard Elements

1. MySQL Installer dashboard operations provide a variety of actions that apply to installed products or products listed in the catalog. To initiate the following operations, first click the operation link and then select the product or products to manage:

- **Add:** This operation opens the Select Products page. From there you can adjust the filter, select one or more products to download (as needed), and begin the installation. For hints about using the filter, see [Locating Products to Install](#).

Use the directional arrows to move each product from the **Available Products** column to the **Products To Be Installed** column. To enable the Product Features page where you can customize features, click the related check box (disabled by default).

Note

For MySQL Server versions 8.0.20 (and earlier), 5.7, and 5.6, the account you use to run MySQL Installer may not have adequate permission to install the server data files and this can interrupt the installation because the [ExecSecureObjects](#) MSI action cannot be executed. To proceed, deselect the **Server data files** feature before attempting to install the server again.

The **Server data files** check box was removed from the feature tree for MySQL Server 8.0.21 (or higher).

- **Modify:** Use this operation to add or remove the features associated with installed products. Features that you can modify vary in complexity by product. When the **Program Shortcut** check box is selected, the product appears in the Start menu under the [MySQL](#) group.

- **Upgrade:** This operation loads the Select Products to Upgrade page and populates it with all the upgrade candidates. An installed product can have more than one upgrade version and the operation requires a current product catalog. MySQL Installer upgrades all of the selected products in one action. Click **Show Details** to view the actions performed by MySQL Installer.
- **Remove:** This operation opens the Remove Products page and populates it with the MySQL products installed on the host. Select the MySQL products you want to remove (uninstall) and then click **Execute** to begin the removal process. During the operation, an indicator shows the number of steps that are executed as a percentage of all steps.


To select products to remove, do one of the following:

- Select the check box for one or more products.
 - Select the **Product** check box to select all products.
2. The **Reconfigure** link in the Quick Action column next to each installed server loads the current configuration values for the server and then cycles through all configuration steps enabling you to change the options and values. You must provide credentials with root privileges to reconfigure these items. Click the **Log** tab to show the output of each configuration step performed by MySQL Installer.

On completion, MySQL Installer stops the server, applies the configuration changes, and restarts the server for you. For a description of each configuration option, see [Section 1.3.3.1, “MySQL Server Configuration with MySQL Installer”](#). Installed [Samples and Examples](#) associated with a specific MySQL server version can also be reconfigured to apply new feature settings, if any.


3. The **Catalog** link enables you to download the latest catalog of MySQL products manually and then to integrate those product changes with MySQL Installer. The catalog-download action does not perform an upgrade of the products already installed on the host. Instead, it returns to the dashboard and adds an arrow icon to the Version column for each installed product that has a newer version. Use the **Upgrade** operation to install the newer product version.

You can also use the **Catalog** link to display the current change history of each product without downloading the new catalog. Select the **Do not update at this time** check box to view the change history only.

4. The MySQL Installer About icon () shows the current version of MySQL Installer and general information about MySQL. The version number is located above the **Back** button.

Tip




Always include this version number when reporting a problem with MySQL Installer.

In addition to the About MySQL information () , you can also select the following icons from the side panel:

- License icon () for MySQL Installer.

This product may include third-party software, used under license. If you are using a Commercial release of MySQL Installer, the icon opens the MySQL Installer Commercial License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a Community release of MySQL

Installer, the icon opens the MySQL Installer Community License Information User Manual for licensing information, including licensing information relating to third-party software that may be included in this Community release.

- Resource links icon () to the latest MySQL product documentation, blogs, webinars, and more.
5. The MySQL Installer Options icon () includes the following tabs:
- **Product Catalog:** Manages the automatic catalog updates. By default, MySQL Installer checks for catalog updates at startup every seven days. When new products or product versions are available, MySQL Installer adds them to the catalog and then inserts an arrow icon () next to the version number of installed products listed in the dashboard.

Use the product catalog option to enable or disable automatic updates and to reset the number of days between automatic catalog downloads. At startup, MySQL Installer uses the number of days you set to determine whether a download should be attempted. This action is repeated during next startup if MySQL Installer encounters an error downloading the catalog.
 - **Connectivity Settings:** Several operations performed by MySQL Installer require internet access. This option enables you to use a default value to validate the connection or to use a different URL, one selected from a list or added by you manually. With the **Manual** option selected, new URLs can be added and all URLs in the list can be moved or deleted. When the **Automatic** option is selected, MySQL Installer attempts to connect to each default URL in the list (in order) until a connection is made. If no connection can be made, it raises an error.

Locating Products to Install

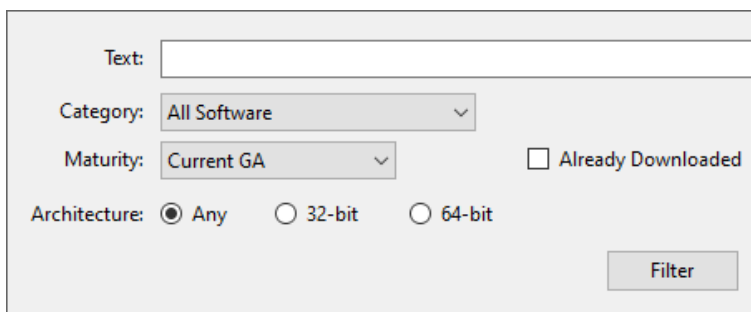
MySQL products in the catalog are listed by category: MySQL Servers, Applications, MySQL Connectors, and Documentation. Only the latest GA versions appear in the **Available Products** pane by default. If you are looking for a pre-release or older version of a product, it may not be visible in the default list.

Note

Keep the product catalog up-to-date. Click **Catalog** on the MySQL Installer dashboard to download the latest manifest.

To change the default product list, click **Add** in the dashboard to open the Select Products page, and then click **Edit** to open the dialog box shown in the figure that follows. Modify the settings and then click **Filter**.

Figure 1.6 Filter Available Products



Text:

Category:

Maturity: Already Downloaded

Architecture: Any 32-bit 64-bit

Reset one or more of the following fields to modify the list of available products:

- Text: Filter by text.

- Category: All Software (default), MySQL Servers, Applications, MySQL Connectors, or Documentation (for samples and documentation).
- Maturity: Current Bundle (appears initially with the full package only), Pre-Release, Current GA, or Other Releases. If you see a warning, confirm that you have the most recent product manifest by clicking **Catalog** on the MySQL Installer dashboard. If MySQL Installer is unable to download the manifest, the range of products you see is limited to bundled products, standalone product MSIs located in the [Product Cache](#) folder already, or both.

Note

The Commercial release of MySQL Installer does not display any MySQL products when you select the Pre-Release maturity filter. Products in development are available from the Community release of MySQL Installer only.

- Already Downloaded (the check box is deselected by default). Permits you to view and manage downloaded products only.
- Architecture: Any (default), 32-bit, or 64-bit.

Upgrading MySQL Server

Important server upgrade conditions:

- MySQL Installer does not permit server upgrades between major release versions or minor release versions, but does permit upgrades within a release series, such as an upgrade from 5.7.18 to 5.7.19.
- Upgrades between milestone releases (or from a milestone release to a GA release) are not supported. Significant development changes take place in milestone releases and you may encounter compatibility issues or problems starting the server.
- For upgrades to MySQL 8.0.16 server and higher, a check box enables you to skip the upgrade check and process for system tables, while checking and processing data dictionary tables normally. MySQL Installer does not prompt you with the check box when the previous server upgrade was skipped or when the server was configured as a sandbox InnoDB Cluster. This behavior represents a change in how MySQL Server performs an upgrade (see [What the MySQL Upgrade Process Upgrades](#)) and it alters the sequence of steps that MySQL Installer applies to the configuration process.

If you select **Skip system tables upgrade check and process. (Not recommended)**, MySQL Installer starts the upgraded server with the `--upgrade=MINIMAL` server option, which upgrades the data dictionary only. If you stop and then restart the server without the `--upgrade=MINIMAL` option, the server upgrades the system tables automatically, if needed.

The following information appears in the **Log** tab and log file after the upgrade configuration (with system tables skipped) is complete:

```
WARNING: The system tables upgrade was skipped after upgrading MySQL Server. The server will be started now with the --upgrade=MINIMAL option, but then each time the server is started it will attempt to upgrade the system tables, unless you modify the Windows service (command line) to add --upgrade=MINIMAL to bypass the upgrade.
```

```
FOR THE BEST RESULTS: Run mysqld.exe --upgrade=FORCE on the command line to upgrade the system tables manually.
```

To choose a new server version:

1. Click **Upgrade**. Confirm that the check box next to product name in the **Upgradeable Products** pane has a check mark. Deselect the products that you do not intend to upgrade at this time.

Note

For server milestone releases in the same release series, MySQL Installer deselects the server upgrade and displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform a logical upgrade manually. You can reselect server upgrade at your own risk. For instructions on how to perform a logical upgrade with a milestone release, see [Logical Upgrade](#).

2. Click a product in the list to highlight it. This action populates the **Upgradeable Versions** pane with the details of each available version for the selected product: version number, published date, and a [Changes](#) link to open the release notes for that version.

Removing MySQL Server

To remove a local MySQL server:

1. Determine whether the local data directory should be removed. If you retain the data directory, another server installation can reuse the data. This option is enabled by default (removes the data directory).
2. Click **Execute** to begin uninstalling the local server. Note that all products that you selected to remove are also uninstalled at this time.
3. (Optional) Click the **Log** tab to display the current actions performed by MySQL Installer.

Upgrading MySQL Installer

MySQL Installer remains installed on your computer, and like other software, MySQL Installer can be upgraded from the previous version. In some cases, other MySQL software may require that you upgrade MySQL Installer for compatibility. This section describes how to identify the current version of MySQL Installer and how to upgrade MySQL Installer manually.

To locate the installed version of MySQL Installer:

1. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
2. Click the MySQL Installer About icon (?). The version number is located above the **Back** button.

To initiate an on-demand upgrade of MySQL Installer:

1. Connect the computer with MySQL Installer installed to the internet.
2. Start MySQL Installer from the search menu. The MySQL Installer dashboard opens.
3. Click **Catalog** on the bottom of the dashboard to open the Update Catalog window.
4. Click **Execute** to begin the process. If the installed version of MySQL Installer can be upgraded, you will be prompted to start the upgrade.
5. Click **Next** to review all changes to the catalog and then click **Finish** to return to the dashboard.
6. Verify the (new) installed version of MySQL Installer (see the previous procedure).

1.3.5 MySQLInstallerConsole Reference

[MySQLInstallerConsole.exe](#) provides command-line functionality that is similar to MySQL Installer. It is installed when MySQL Installer is initially executed and then available within the [MySQL Installer](#)

for `Windows` directory. By default, that is in `C:\Program Files (x86)\MySQL\MySQL Installer for Windows`, and the console must be executed with administrative privileges.

To use, invoke the command prompt with administrative privileges by choosing **Start, Accessories**, then right-click on **Command Prompt** and choose **Run as administrator**. And from the command line, optionally change the directory to where `MySQLInstallerConsole.exe` is located:

```
C:\> cd Program Files (x86)\MySQL\MySQL Installer for Windows
C:\Program Files (x86)\MySQL\MySQL Installer for Windows> MySQLInstallerConsole.exe help
===== Start Initialization =====
MySQL Installer is running in Community mode

Attempting to update manifest.
Initializing product requirements
Loading product catalog
Checking for product catalog snippets
Checking for product packages in the bundle
Categorizing product catalog
Finding all installed packages.
Your product catalog was last updated at 11/1/2016 4:10:38 PM
===== End Initialization =====

The following commands are available:

Configure - Configures one or more of your installed programs.
Help      - Provides list of available commands.
Install   - Install and configure one or more available MySQL programs.
List      - Provides an interactive way to list all products available.
Modify    - Modifies the features of installed products.
Remove    - Removes one or more products from your system.
Status    - Shows the status of all installed products.
Update    - Update the current product catalog.
Upgrade   - Upgrades one or more of your installed programs.
```

MySQL Product Names

Many of the `MySQLInstallerConsole` commands accept one or more keywords that represent a MySQL product (or products) in the catalog. The current set of valid keywords for use with commands is shown in the following table.

Table 1.3 MySQL Product Keywords for MySQLInstallerConsole

Keyword	MySQL Product
<code>server</code>	MySQL Server
<code>workbench</code>	MySQL Workbench
<code>shell</code>	MySQL Shell
<code>visual</code>	MySQL for Visual Studio
<code>router</code>	MySQL Router
<code>backup</code>	MySQL Enterprise Backup
<code>net</code>	MySQL Connector/NET
<code>odbc</code>	MySQL Connector/ODBC
<code>c++</code>	MySQL Connector/C++
<code>python</code>	MySQL Connector/Python
<code>j</code>	MySQL Connector/J
<code>documentation</code>	MySQL Server Documentation

Keyword	MySQL Product
<code>samples</code>	MySQL Samples (sakila and world databases)

MySQLInstallerConsole Command Options

MySQLInstallerConsole.exe supports the following command options:

Note

Configuration block values that contain a colon character (:) must be wrapped in quotation marks. For example, `installdir="C:\MySQL\MySQL Server 8.0"`.

- `configure [product1]:[setting]=[value]; [product2]:[setting]=[value]; [...]`

Configures one or more MySQL products on your system. Multiple setting=value pairs can be configured for each product.

Switches include:

- `-showsettings` Displays the available options for the selected product, by passing in the product name after `-showsettings`.
- `-silent` Disables confirmation prompts.

```
C:\> MySQLInstallerConsole configure -showsettings server
C:\> MySQLInstallerConsole configure server:port=3307
```

- `help [command]`

Displays a help message with usage examples and then exits. Pass in an additional command to receive help specific to that command.

```
C:\> MySQLInstallerConsole help
C:\> MySQLInstallerConsole help install
```

- `install [product]:[features]:[config block]:[config block]:[config block]; [...]`

Installs one or more MySQL products on your system. If pre-release products are available, both GA and pre-release products are installed when the value of the `-type` switch is `Developer`, `Client`, or `Full`. Use the `-only_ga_products` switch to restrict the product set to GA products only when using these setup types.

Switches and syntax options include:

- `-only_ga_products` Restricts the product set to include GA products only.
- `-type=[SetupType]` Installs a predefined set of software. The setup type can be one of the following:
 - `Developer`: Installs a complete development environment.
 - `Server`: Installs a single MySQL server
 - `Client`: Installs client programs and libraries
 - `Full`: Installs everything

- `Custom`: Installs user-selected products. This is the default option.

Note

Non-custom setup types are valid only when no other MySQL products are installed.

<code>-showsettings</code>	Displays the available options for the selected product, by passing in the product name after <code>-showsettings</code> .
<code>-silent</code>	Disable confirmation prompts.
<code>[product]</code>	Each product can be specified by a <code>product keyword</code> with or without a semicolon-separated version qualifier. Passing in a product keyword alone selects the latest version of the product. If multiple architectures are available for that version of the product, the command returns the first one in the manifest list for interactive confirmation. Alternatively, you can pass in the exact version and architecture (<code>x86</code> or <code>x64</code>) after the product keyword using the <code>-silent</code> switch.
<code>[features]</code>	All features associated with a MySQL product are installed by default. The feature block is a semicolon-separated list of features or an asterisk character (*) that selects all features. To remove a feature, use the <code>modify</code> command.
<code>[config block]</code>	One or more configuration blocks can be specified. Each configuration block is a semicolon-separated list of key-value pairs. A block can include either a <code>config</code> or <code>user</code> type key; <code>config</code> is the default type if one is not defined. Configuration block values that contain a colon character (:) must be wrapped in quotation marks. For example, <code>installdir="C:\MySQL\MySQL Server 8.0"</code> . Only one configuration type block can be defined for each product. A user block should be defined for each user to be created during the product installation.

Note

The `user` type key is not supported when a product is being reconfigured.

```
C:\> MySQLInstallerConsole install server;5.6.25:*:port=3307;serverid=2:type=user;username=foo;password=
C:\> MySQLInstallerConsole install server;5.6.25;x64 -silent
```

An example that passes in additional configuration blocks, separated by ^ to fit:

```
C:\> MySQLInstallerConsole install server;5.6.25;x64:*:type=config;openfirewall=true; ^
    generallog=true;binlog=true;serverid=3306;enable_tcpip=true;port=3306;rootpasswd=pass; ^
    installdir="C:\MySQL\MySQL Server 5.6":type=user;datadir="C:\MySQL\data";username=foo;password=
```

- `list`

Lists an interactive console where all of the available MySQL products can be searched. Execute `MySQLInstallerConsole list` to launch the console and enter in a substring to search.

```
C:\> MySQLInstallerConsole list
```

- `modify [product1:-removelist|+addlist] [product2:-removelist|+addlist] [...]`

Modifies or displays features of a previously installed MySQL product. To display the features of a product, append the product keyword to the command, for example:

```
C:\> MySQLInstallerConsole modify server
```

The syntax option for this command:

`-silent` Disable confirmation prompts.

```
C:\> MySQLInstallerConsole modify server:+documentation
C:\> MySQLInstallerConsole modify server:-debug
```

- `remove [product1] [product2] [...]`

Removes one or more products from your system. Switches and syntax options include:

`*` Pass in `*` to remove all of the MySQL products.

`-continue` Continue the operation even if an error occurs.

`-silent` Disable confirmation prompts.

```
C:\> MySQLInstallerConsole remove *
C:\> MySQLInstallerConsole remove server
```

- `status`

Provides a quick overview of the MySQL products that are installed on the system. Information includes product name and version, architecture, date installed, and install location.

```
C:\> MySQLInstallerConsole status
```

- `update`

Downloads the latest MySQL product catalog to your system. On success, the catalog is applied the next time either `MySQLInstaller` or `MySQLInstallerConsole.exe` is executed.

```
C:\> MySQLInstallerConsole update
```

Note

The **Automatic Catalog Update** GUI option executes this command from the Windows Task Scheduler.

- `upgrade [product1:version] [product2:version] [...]`

Upgrades one or more products on your system. Syntax options include:

`*` Pass in `*` to upgrade all products to the latest version, or pass in specific products.

`!` Pass in `!` as a version number to upgrade the MySQL product to its latest version.

`-silent` Disable confirmation prompts.

```
C:\> MySQLInstallerConsole upgrade *
C:\> MySQLInstallerConsole upgrade workbench:8.0.21
```

```
C:\> MySQLInstallerConsole upgrade workbench:!  
C:\> MySQLInstallerConsole upgrade workbench:8.0.21 visual:1.2.9
```

1.4 Installing MySQL on Microsoft Windows Using a `noinstall` ZIP Archive

Users who are installing from the `noinstall` package can use the instructions in this section to manually install MySQL. The process for installing MySQL from a ZIP Archive package is as follows:

1. Extract the main archive to the desired install directory
Optional: also extract the debug-test archive if you plan to execute the MySQL benchmark and test suite
2. Create an option file
3. Choose a MySQL server type
4. Initialize MySQL
5. Start the MySQL server
6. Secure the default user accounts

This process is described in the sections that follow.

1.4.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to [Chapter 2, Upgrading MySQL on Windows](#), before beginning the upgrade process.
2. Make sure that you are logged in as a user with administrator privileges.
3. Choose an installation location. Traditionally, the MySQL server is installed in `C:\mysql`. If you do not install MySQL at `C:\mysql`, you must specify the path to the install directory during startup or in an option file. See [Section 1.4.2, “Creating an Option File”](#).

Note

The MySQL Installer installs MySQL under `C:\Program Files\MySQL`.

4. Extract the install archive to the chosen installation location using your preferred file-compression tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.

1.4.2 Creating an Option File

If you need to specify startup options when you run the server, you can indicate them on the command line or place them in an option file. For options that are used every time the server starts, you may find it most convenient to use an option file to specify your MySQL configuration. This is particularly true under the following circumstances:

- The installation or data directory locations are different from the default locations (`C:\Program Files\MySQL\MySQL Server 8.0` and `C:\Program Files\MySQL\MySQL Server 8.0\data`).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, `C:\`, and the MySQL installation directory (for the full list of locations, see [Using Option Files](#)). The Windows directory typically is named something like `C:\WINDOWS`. You can determine its exact location from the value of the `WINDIR` environment variable using the following command:

```
C:\> echo %WINDIR%
```

MySQL looks for options in each location first in the `my.ini` file, and then in the `my.cnf` file. However, to avoid confusion, it is best if you use only one file. If your PC uses a boot loader where `C:` is not the boot drive, your only option is to use the `my.ini` file. Whichever option file you use, it must be a plain text file.

Note

When using the MySQL Installer to install MySQL Server, it creates the `my.ini` at the default location, and the user executing MySQL Installer is granted full permissions to this new `my.ini` file.

In other words, be sure that the MySQL Server user has permission to read the `my.ini` file.

You can also make use of the example option files included with your MySQL distribution; see [Server Configuration Defaults](#).

An option file can be created and modified with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is in `E:\mydata\data`, you can create an option file containing a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

The rules for use of backslash in option file values are given in [Using Option Files](#).

The ZIP archive does not include a `data` directory. To initialize a MySQL installation by creating the data directory and populating the tables in the mysql system database, initialize MySQL using either `--initialize` or `--initialize-insecure`. For additional information, see [Initializing the Data Directory](#).

If you would like to use a data directory in a different location, you should copy the entire contents of the `data` directory to the new location. For example, if you want to use `E:\mydata` as the data directory instead, you must do two things:

1. Move the entire `data` directory and all of its contents from the default location (for example `C:\Program Files\MySQL\MySQL Server 8.0\data`) to `E:\mydata`.
2. Use a `--datadir` option to specify the new data directory location each time you start the server.

1.4.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 8.0.

Binary	Description
<code>mysqld</code>	Optimized binary with named-pipe support
<code>mysqld-debug</code>	Like <code>mysqld</code> , but compiled with full debugging and automatic memory allocation checking

All of the preceding binaries are optimized for modern Intel processors, but should work on any Intel i386-class or higher processor.

Each of the servers in a distribution support the same set of storage engines. The `SHOW ENGINES` statement displays which engines a given server supports.

All Windows MySQL 8.0 servers have support for symbolic linking of database directories.

MySQL supports TCP/IP on all Windows platforms. MySQL servers on Windows also support named pipes, if you start the server with the `named_pipe` system variable enabled. It is necessary to enable this variable explicitly because some users have experienced problems with shutting down the MySQL server when named pipes were used. The default is to use TCP/IP regardless of platform because named pipes are slower than TCP/IP in many Windows configurations.

1.4.4 Initializing the Data Directory

If you installed MySQL using the `noinstall` package, no data directory is included. To initialize the data directory, use the instructions at [Initializing the Data Directory](#).

1.4.5 Starting the Server for the First Time

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the `noinstall` version, or if you wish to configure and test MySQL manually rather than with the MySQL Installer.

The examples in these sections assume that MySQL is installed under the default location of `C:\Program Files\MySQL\MySQL Server 8.0`. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the `shared_memory` system variable enabled. Clients can connect through shared memory by using the `--protocol=MEMORY` option.

For information about which server binary to run, see [Section 1.4.3, “Selecting a MySQL Server Type”](#).

Testing is best done from a command prompt in a console window (or “DOS window”). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

Note

The database must be initialized before MySQL can be started. For additional information about the initialization process, see [Initializing the Data Directory](#).

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --console
```

For a server that includes [InnoDB](#) support, you should see the messages similar to those following as it starts (the path names and sizes may differ):

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25 InnoDB: Started
```

When the server finishes its startup sequence, you should see something like this, which indicates that the server is ready to service client connections:

```
mysqld: ready for connections
Version: '8.0.27' socket: '' port: 3306
```

The server continues to write to the console any further diagnostic output it produces. You can open a new console window in which to run client programs.

If you omit the `--console` option, the server writes diagnostic output to the error log in the data directory (`C:\Program Files\MySQL\MySQL Server 8.0\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

Note

The initial `root` account in the MySQL grant tables has no password. After starting the server, you should set up a password for it using the instructions in [Securing the Initial MySQL Account](#).

1.4.6 Starting MySQL from the Windows Command Line

The MySQL server can be started manually from the command line. This can be done on any version of Windows.

To start the `mysqld` server from the command line, you should start a console window (or “DOS window”) and enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
```

The path to `mysqld` may vary depending on the install location of MySQL on your system.

You can stop the MySQL server by executing this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" -u root shutdown
```

Note

If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

Note

Users in the MySQL grant system are wholly independent from any operating system users under Microsoft Windows.

If `mysqld` doesn't start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. By default, the error log is located in the `C:\Program Files\MySQL\MySQL Server 8.0\data` directory. It is the file with a suffix of `.err`, or may be specified by passing in the `--log-error` option. Alternatively, you can try to start the server with the `--console` option; in this case, the server may display some useful information on the screen to help solve the problem.

The last option is to start `mysqld` with the `--standalone` and `--debug` options. In this case, `mysqld` writes a log file `C:\mysqld.trace` that should contain the reason why `mysqld` doesn't start. See [The DEBUG Package](#).

Use `mysqld --verbose --help` to display all the options that `mysqld` supports.

1.4.7 Customizing the PATH for MySQL Tools

Warning

You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 8.0\bin`)

Note

There must be a semicolon separating this path from any values present in this field.

Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. The new `PATH` value should now be available to any new command shell you open, allowing you to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

1.4.8 Starting MySQL as a Windows Service

On Windows, the recommended way to run MySQL is to install it as a Windows service, so that MySQL starts and stops automatically when Windows starts and stops. A MySQL server installed as a service can also be controlled from the command line using `NET` commands, or with the graphical `Services` utility. Generally, to install MySQL as a Windows service you should be logged in using an account that has administrator rights.

The `Services` utility (the Windows `Service Control Manager`) can be found in the Windows Control Panel. To avoid conflicts, it is advisable to close the `Services` utility while performing server installation or removal operations from the command line.

Installing the service

Before installing MySQL as a Windows service, you should first stop the current server if it is running by using the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin"  
-u root shutdown
```

Note

If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

Note

Users in the MySQL grant system are wholly independent from any operating system users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install
```

The service-installation command does not start the server. Instructions for that are given later in this section.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 8.0\bin`), and there should be a semicolon separating this path from any values present

in this field. Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

Warning

You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

The following additional arguments can be used when installing the service:

- You can specify a service name immediately following the `--install` option. The default service name is `MySQL`.
- If a service name is given, it can be followed by a single option. By convention, this should be `--defaults-file=file_name` to specify the name of an option file from which the server should read options when it starts.

The use of a single option other than `--defaults-file` is possible but discouraged. `--defaults-file` is more flexible because it enables you to specify multiple startup options for the server by placing them in the named option file.

- You can also specify a `--local-service` option following the service name. This causes the server to run using the `LocalService` Windows account that has limited system privileges. If both `--defaults-file` and `--local-service` are given following the service name, they can be in any order.

For a MySQL server that is installed as a Windows service, the following rules determine the service name and option files that the server uses:

- If the service-installation command specifies no service name or the default service name (`MySQL`) following the `--install` option, the server uses the service name of `MySQL` and reads options from the `[mysqld]` group in the standard option files.
- If the service-installation command specifies a service name other than `MySQL` following the `--install` option, the server uses that service name. It reads options from the `[mysqld]` group and the group that has the same name as the service in the standard option files. This enables you to use the `[mysqld]` group for options that should be used by all MySQL services, and an option group with the service name for use by the server installed with that service name.
- If the service-installation command specifies a `--defaults-file` option after the service name, the server reads options the same way as described in the previous item, except that it reads options only from the named file and ignores the standard option files.

As a more complex example, consider the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
      --install MySQL --defaults-file=C:\my-opts.cnf
```

Here, the default service name (`MySQL`) is given after the `--install` option. If no `--defaults-file` option had been given, this command would have the effect of causing the server to read the `[mysqld]`

group from the standard option files. However, because the `--defaults-file` option is present, the server reads options from the `[mysqld]` option group, and only from the named file.

Note

On Windows, if the server is started with the `--defaults-file` and `--install` options, `--install` must be first. Otherwise, `mysqld.exe` attempts to start the MySQL server.

You can also specify options as Start parameters in the Windows `Services` utility before you start the MySQL service.

Finally, before trying to start the MySQL service, make sure the user variables `%TEMP%` and `%TMP%` (and also `%TMPDIR%`, if it has ever been set) for the operating system user who is to run the service are pointing to a folder to which the user has write access. The default user for running the MySQL service is `LocalSystem`, and the default value for its `%TEMP%` and `%TMP%` is `C:\Windows\Temp`, a directory `LocalSystem` has write access to by default. However, if there are any changes to that default setup (for example, changes to the user who runs the service or to the mentioned user variables, or the `--tmpdir` option has been used to put the temporary directory somewhere else), the MySQL service might fail to run because write access to the temporary directory has not been granted to the proper user.

Starting the service

After a MySQL server instance has been installed as a service, Windows starts the service automatically whenever Windows starts. The service also can be started immediately from the `Services` utility, or by using an `sc start mysql service_name` or `NET START mysql service_name` command. `SC` and `NET` commands are not case-sensitive.

When run as a service, `mysqld` has no access to a console window, so no messages can be seen there. If `mysqld` does not start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. The error log is located in the MySQL data directory (for example, `C:\Program Files\MySQL\MySQL Server 8.0\data`). It is the file with a suffix of `.err`.

When a MySQL server has been installed as a service, and the service is running, Windows stops the service automatically when Windows shuts down. The server also can be stopped manually using the `Services` utility, the `sc stop mysql service_name` command, the `NET STOP mysql service_name` command, or the `mysqladmin shutdown` command.

You also have the choice of installing the server as a manual service if you do not wish for the service to be started automatically during the boot process. To do this, use the `--install-manual` option rather than the `--install` option:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --install-manual
```

Removing the service

To remove a server that is installed as a service, first stop it if it is running by executing `SC STOP mysql service_name` or `NET STOP mysql service_name`. Then use `SC DELETE mysql service_name` to remove it:

```
C:\> SC DELETE mysql
```

Alternatively, use the `mysqld --remove` option to remove the service.

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld" --remove
```

If `mysqld` is not running as a service, you can start it from the command line. For instructions, see [Section 1.4.6, "Starting MySQL from the Windows Command Line"](#).

If you encounter difficulties during installation, see [Section 1.5, “Troubleshooting a Microsoft Windows MySQL Server Installation”](#).

For more information about stopping or removing a Windows service, see [Starting Multiple MySQL Instances as Windows Services](#).

1.4.9 Testing The MySQL Installation

You can test whether the MySQL server is working by executing any of the following commands:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow"  
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqlshow" -u root mysql  
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" version status proc  
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql" test
```

If `mysqld` is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start `mysqld` with the `skip_name_resolve` system variable enabled and use only `localhost` and IP addresses in the `Host` column of the MySQL grant tables. (Be sure that an account exists that specifies an IP address or you may not be able to connect.)

You can force a MySQL client to use a named-pipe connection rather than TCP/IP by specifying the `--pipe` or `--protocol=PIPE` option, or by specifying `.` (period) as the host name. Use the `--socket` option to specify the name of the pipe if you do not want to use the default pipe name.

If you have set a password for the `root` account, deleted the anonymous account, or created a new user account, then to connect to the MySQL server you must use the appropriate `-u` and `-p` options with the commands shown previously. See [Connecting to the MySQL Server Using Command Options](#).

For more information about `mysqlshow`, see [mysqlshow — Display Database, Table, and Column Information](#).

1.5 Troubleshooting a Microsoft Windows MySQL Server Installation

When installing and running MySQL for the first time, you may encounter certain errors that prevent the MySQL server from starting. This section helps you diagnose and correct some of these errors.

Your first resource when troubleshooting server issues is the [error log](#). The MySQL server uses the error log to record information relevant to the error that prevents the server from starting. The error log is located in the [data directory](#) specified in your `my.ini` file. The default data directory location is `C:\Program Files\MySQL\MySQL Server 8.0\data`, or `C:\ProgramData\Mysql` on Windows 7 and Windows Server 2008. The `C:\ProgramData` directory is hidden by default. You need to change your folder options to see the directory and contents. For more information on the error log and understanding the content, see [The Error Log](#).

For information regarding possible errors, also consult the console messages displayed when the MySQL service is starting. Use the `SC START mysqld_service_name` or `NET START mysqld_service_name` command from the command line after installing `mysqld` as a service to see any error messages regarding the starting of the MySQL server as a service. See [Section 1.4.8, “Starting MySQL as a Windows Service”](#).

The following examples show other common error messages you might encounter when installing MySQL and starting the server for the first time:

- If the MySQL server cannot find the `mysql` privileges database or other critical files, it displays these messages:

```
System error 1067 has occurred.  
Fatal error: Can't open and lock privilege tables:  
Table 'mysql.user' doesn't exist
```

These messages often occur when the MySQL base or data directories are installed in different locations than the default locations ([C:\Program Files\MySQL\MySQL Server 8.0](#) and [C:\Program Files\MySQL\MySQL Server 8.0\data](#), respectively).

This situation can occur when MySQL is upgraded and installed to a new location, but the configuration file is not updated to reflect the new location. In addition, old and new configuration files might conflict. Be sure to delete or rename any old configuration files when upgrading MySQL.

If you have installed MySQL to a directory other than [C:\Program Files\MySQL\MySQL Server 8.0](#), ensure that the MySQL server is aware of this through the use of a configuration ([my.ini](#)) file. Put the [my.ini](#) file in your Windows directory, typically [C:\WINDOWS](#). To determine its exact location from the value of the [WINDIR](#) environment variable, issue the following command from the command prompt:

```
C:\> echo %WINDIR%
```

You can create or modify an option file with any text editor, such as Notepad. For example, if MySQL is installed in [E:\mysql](#) and the data directory is [D:\MySQLdata](#), you can create the option file and set up a [\[mysqld\]](#) section to specify values for the [basedir](#) and [datadir](#) options:

```
[mysqld]  
# set basedir to your installation path  
basedir=E:/mysql  
# set datadir to the location of your data directory  
datadir=D:/MySQLdata
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]  
# set basedir to your installation path  
basedir=C:\\Program Files\\MySQL\\MySQL Server 8.0  
# set datadir to the location of your data directory  
datadir=D:\\MySQLdata
```

The rules for use of backslash in option file values are given in [Using Option Files](#).

If you change the [datadir](#) value in your MySQL configuration file, you must move the contents of the existing MySQL data directory before restarting the MySQL server.

See [Section 1.4.2, "Creating an Option File"](#).

- If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service and install MySQL using the MySQL Installer, you might see this error:

```
Error: Cannot create Windows service for MySql. Error: 0
```

This occurs when the Configuration Wizard tries to install the service and finds an existing service with the same name.

One solution to this problem is to choose a service name other than [mysql](#) when using the configuration wizard. This enables the new service to be installed correctly, but leaves the outdated service in place. Although this is harmless, it is best to remove old services that are no longer in use.

To permanently remove the old [mysql](#) service, execute the following command as a user with administrative privileges, on the command line:

```
C:\> SC DELETE mysql
[SC] DeleteService SUCCESS
```

If the `SC` utility is not available for your version of Windows, download the `delsrv` utility from <http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp> and use the `delsrv mysql` syntax.

1.6 Windows Postinstallation Procedures

GUI tools exist that perform most of the tasks described in this section, including:

- **MySQL Installer:** Used to install and upgrade MySQL products.
- **MySQL Workbench:** Manages the MySQL server and edits SQL statements.

If necessary, initialize the data directory and create the MySQL grant tables. Windows installation operations performed by MySQL Installer initialize the data directory automatically. For installation from a ZIP Archive package, initialize the data directory as described at [Initializing the Data Directory](#).

Regarding passwords, if you installed MySQL using the MySQL Installer, you may have already assigned a password to the initial `root` account. (See [Section 1.3, “MySQL Installer for Windows”](#).) Otherwise, use the password-assignment procedure given in [Securing the Initial MySQL Account](#).

Before assigning a password, you might want to try running some client programs to make sure that you can connect to the server and that it is operating properly. Make sure that the server is running (see [Section 1.4.5, “Starting the Server for the First Time”](#)). You can also set up a MySQL service that runs automatically when Windows starts (see [Section 1.4.8, “Starting MySQL as a Windows Service”](#)).

These instructions assume that your current location is the MySQL installation directory and that it has a `bin` subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

If you installed MySQL using MySQL Installer (see [Section 1.3, “MySQL Installer for Windows”](#)), the default installation directory is `C:\Program Files\MySQL\MySQL Server 8.0`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0"
```

A common installation location for installation from a ZIP archive is `C:\mysql`:

```
C:\> cd C:\mysql
```

Alternatively, add the `bin` directory to your `PATH` environment variable setting. That enables your command interpreter to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See [Section 1.4.7, “Customizing the PATH for MySQL Tools”](#).

With the server running, issue the following commands to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use `mysqlshow` to see what databases exist:

```
C:\> bin\mysqlshow
+-----+
| Databases |
+-----+
| information_schema |
| mysql |
```

```
| performance_schema |
| sys                 |
+-----+

```

The list of installed databases may vary, but always includes at least `mysql` and `information_schema`.

The preceding command (and commands for other MySQL programs such as `mysql`) may not work if the correct MySQL account does not exist. For example, the program may fail with an error, or you may not be able to view all databases. If you install MySQL using MySQL Installer, the `root` user is created automatically with the password you supplied. In this case, you should use the `-u root` and `-p` options. (You must use those options if you have already secured the initial MySQL accounts.) With `-p`, the client program prompts for the `root` password. For example:

```
C:\> bin\mysqlshow -u root -p
Enter password: (enter root password here)
+-----+
| Databases |
+-----+
| information_schema |
| mysql             |
| performance_schema |
| sys               |
+-----+

```

If you specify a database name, `mysqlshow` displays a list of the tables within the database:

```
C:\> bin\mysqlshow mysql
Database: mysql
+-----+
| Tables |
+-----+
| columns_priv |
| component    |
| db           |
| default_roles |
| engine_cost  |
| func         |
| general_log  |
| global_grants |
| gtid_executed |
| help_category |
| help_keyword |
| help_relation |
| help_topic   |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| password_history |
| plugin       |
| procs_priv   |
| proxies_priv |
| role_edges   |
| server_cost  |
| servers      |
| slave_master_info |
| slave_relay_log_info |
| slave_worker_info |
| slow_log     |
| tables_priv  |
| time_zone    |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user         |

```


Use the `mysql` program to select information from a table in the `mysql` database:

```
C:\> bin\mysql -e "SELECT User, Host, plugin FROM mysql.user" mysql
+-----+-----+
| User | Host      | plugin                |
+-----+-----+-----+
| root | localhost | caching_sha2_password |
+-----+-----+-----+
```

For more information about `mysql` and `mysqlshow`, see [mysql — The MySQL Command-Line Client](#), and [mysqlshow — Display Database, Table, and Column Information](#).

1.7 Windows Platform Restrictions

The following restrictions apply to use of MySQL on the Windows platform:

- **Process memory**

On Windows 32-bit platforms, it is not possible by default to use more than 2GB of RAM within a single process, including MySQL. This is because the physical address limit on Windows 32-bit is 4GB and the default setting within Windows is to split the virtual address space between kernel (2GB) and user/applications (2GB).

Some versions of Windows have a boot time setting to enable larger applications by reducing the kernel application. Alternatively, to use more than 2GB, use a 64-bit version of Windows.

- **File system aliases**

When using `MyISAM` tables, you cannot use aliases within Windows link to the data files on another volume and then link back to the main MySQL `datadir` location.

This facility is often used to move the data and index files to a RAID or other fast solution.

- **Limited number of ports**

Windows systems have about 4,000 ports available for client connections, and after a connection on a port closes, it takes two to four minutes before the port can be reused. In situations where clients connect to and disconnect from the server at a high rate, it is possible for all available ports to be used up before closed ports become available again. If this happens, the MySQL server appears to be unresponsive even though it is running. Ports may be used by other applications running on the machine as well, in which case the number of ports available to MySQL is lower.

For more information about this problem, see <https://support.microsoft.com/kb/196271>.

- **DATA DIRECTORY and INDEX DIRECTORY**

The `DATA DIRECTORY` clause of the `CREATE TABLE` statement is supported on Windows for `InnoDB` tables only, as described in [Creating Tables Externally](#). For `MyISAM` and other storage engines, the `DATA DIRECTORY` and `INDEX DIRECTORY` clauses for `CREATE TABLE` are ignored on Windows and any other platforms with a nonfunctional `realpath()` call.

- **DROP DATABASE**

You cannot drop a database that is in use by another session.

- **Case-insensitive names**

File names are not case-sensitive on Windows, so MySQL database and table names are also not case-sensitive on Windows. The only restriction is that database and table names must be specified using the same case throughout a given statement. See [Identifier Case Sensitivity](#).

- **Directory and file names**

On Windows, MySQL Server supports only directory and file names that are compatible with the current ANSI code pages. For example, the following Japanese directory name does not work in the Western locale (code page 1252):

```
datadir="C:/私たちのプロジェクトのデータ"
```

The same limitation applies to directory and file names referred to in SQL statements, such as the data file path name in `LOAD DATA`.

- **The \ path name separator character**

Path name components in Windows are separated by the `\` character, which is also the escape character in MySQL. If you are using `LOAD DATA` or `SELECT ... INTO OUTFILE`, use Unix-style file names with `/` characters:

```
mysql> LOAD DATA INFILE 'C:/tmp/skr.txt' INTO TABLE skr;  
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

Alternatively, you must double the `\` character:

```
mysql> LOAD DATA INFILE 'C:\\tmp\\skr.txt' INTO TABLE skr;  
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

- **Problems with pipes**

Pipes do not work reliably from the Windows command-line prompt. If the pipe includes the character `^Z` / `CHAR(24)`, Windows thinks that it has encountered end-of-file and aborts the program.

This is mainly a problem when you try to apply a binary log as follows:

```
C:\> mysqlbinlog binary_log_file | mysql --user=root
```

If you have a problem applying the log and suspect that it is because of a `^Z` / `CHAR(24)` character, you can use the following workaround:

```
C:\> mysqlbinlog binary_log_file --result-file=/tmp/bin.sql  
C:\> mysql --user=root --execute "source /tmp/bin.sql"
```

The latter command also can be used to reliably read any SQL file that may contain binary data.

Chapter 2 Upgrading MySQL on Windows

There are two approaches for upgrading MySQL on Windows:

- [Using MySQL Installer](#)
- [Using the Windows ZIP archive distribution](#)

The approach you select depends on how the existing installation was performed. Before proceeding, review [Upgrading MySQL](#) for additional information on upgrading MySQL that is not specific to Windows.

Note

Whichever approach you choose, always back up your current MySQL installation before performing an upgrade. See [Database Backup Methods](#).

Upgrades between non-GA releases (or from a non-GA release to a GA release) are not supported. Significant development changes take place in non-GA releases and you may encounter compatibility issues or problems starting the server.

Note

MySQL Installer does not support upgrades between *Community* releases and *Commercial* releases. If you require this type of upgrade, perform it using the [ZIP archive](#) approach.

Upgrading MySQL with MySQL Installer

Performing an upgrade with MySQL Installer is the best approach when the current server installation was performed with it and the upgrade is within the current release series. MySQL Installer does not support upgrades between release series, such as from 5.7 to 8.0, and it does not provide an upgrade indicator to prompt you to upgrade. For instructions on upgrading between release series, see [Upgrading MySQL Using the Windows ZIP Distribution](#).

To perform an upgrade using MySQL Installer:

1. Start MySQL Installer.
2. From the dashboard, click **Catalog** to download the latest changes to the catalog. The installed server can be upgraded only if the dashboard displays an arrow next to the version number of the server.
3. Click **Upgrade**. All products that have a newer version now appear in a list.

Note

MySQL Installer deselects the server upgrade option for milestone releases (Pre-Release) in the same release series. In addition, it displays a warning to indicate that the upgrade is not supported, identifies the risks of continuing, and provides a summary of the steps to perform an upgrade manually. You can reselect server upgrade and proceed at your own risk.

4. Deselect all but the MySQL server product, unless you intend to upgrade other products at this time, and click **Next**.
5. Click **Execute** to start the download. When the download finishes, click **Next** to begin the upgrade operation.

Upgrades to MySQL 8.0.16 and higher may show an option to skip the upgrade check and process for system tables. For more information about this option, see [Important server upgrade conditions](#).

6. Configure the server.

Upgrading MySQL Using the Windows ZIP Distribution

To perform an upgrade using the Windows ZIP archive distribution:

1. Download the latest Windows ZIP Archive distribution of MySQL from <https://dev.mysql.com/downloads/>.
2. If the server is running, stop it. If the server is installed as a service, stop the service with the following command from the command prompt:

```
C:\> SC STOP mysqld_service_name
```

Alternatively, use `NET STOP mysqld_service_name` .

If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 5.7 to 8.0, use `mysqladmin` from MySQL 5.7 as follows:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqladmin" -u root shutdown
```

Note

If the MySQL `root` user account has a password, invoke `mysqladmin` with the `-p` option and enter the password when prompted.

3. Extract the ZIP archive. You may either overwrite your existing MySQL installation (usually located at `C:\mysql`), or install it into a different directory, such as `C:\mysql8`. Overwriting the existing installation is recommended.
4. Restart the server. For example, use the `SC START mysqld_service_name` or `NET START mysqld_service_name` command if you run MySQL as a service, or invoke `mysqld` directly otherwise.
5. Prior to MySQL 8.0.16, run `mysql_upgrade` as Administrator to check your tables, attempt to repair them if necessary, and update your grant tables if they have changed so that you can take advantage of any new capabilities. See [mysql_upgrade — Check and Upgrade MySQL Tables](#). As of MySQL 8.0.16, this step is not required, as the server performs all tasks previously handled by `mysql_upgrade`.
6. If you encounter errors, see [Section 1.5, “Troubleshooting a Microsoft Windows MySQL Server Installation”](#).

Chapter 3 Connection to MySQL Server Failing on Windows

When you're running a MySQL server on Windows with many TCP/IP connections to it, and you're experiencing that quite often your clients get a `Can't connect to MySQL server` error, the reason might be that Windows does not allow for enough ephemeral (short-lived) ports to serve those connections.

The purpose of `TIME_WAIT` is to keep a connection accepting packets even after the connection has been closed. This is because Internet routing can cause a packet to take a slow route to its destination and it may arrive after both sides have agreed to close. If the port is in use for a new connection, that packet from the old connection could break the protocol or compromise personal information from the original connection. The `TIME_WAIT` delay prevents this by ensuring that the port cannot be reused until after some time has been permitted for those delayed packets to arrive.

It is safe to reduce `TIME_WAIT` greatly on LAN connections because there is little chance of packets arriving at very long delays, as they could through the Internet with its comparatively large distances and latencies.

Windows permits ephemeral (short-lived) TCP ports to the user. After any port is closed, it remains in a `TIME_WAIT` status for 120 seconds. The port is not available again until this time expires. The default range of port numbers depends on the version of Windows, with a more limited number of ports in older versions:

- Windows through Server 2003: Ports in range 1025–5000
- Windows Vista, Server 2008, and newer: Ports in range 49152–65535

With a small stack of available TCP ports (5000) and a high number of TCP ports being open and closed over a short period of time along with the `TIME_WAIT` status you have a good chance for running out of ports. There are two ways to address this problem:

- Reduce the number of TCP ports consumed quickly by investigating connection pooling or persistent connections where possible
- Tune some settings in the Windows registry (see below)

Important

The following procedure involves modifying the Windows registry. Before you modify the registry, make sure to back it up and make sure that you understand how to restore it if a problem occurs. For information about how to back up, restore, and edit the registry, view the following article in the Microsoft Knowledge Base: <http://support.microsoft.com/kb/256986/EN-US/>.

1. Start Registry Editor (`Regedt32.exe`).
2. Locate the following key in the registry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

3. On the `Edit` menu, click `Add Value`, and then add the following registry value:

```
Value Name: MaxUserPort
Data Type: REG_DWORD
Value: 65534
```

This sets the number of ephemeral ports available to any user. The valid range is between 5000 and 65534 (decimal). The default value is 0x1388 (5000 decimal).

-
4. On the [Edit](#) menu, click [Add Value](#), and then add the following registry value:

```
Value Name: TcpTimedWaitDelay  
Data Type: REG_DWORD  
Value: 30
```

This sets the number of seconds to hold a TCP port connection in [TIME_WAIT](#) state before closing. The valid range is between 30 and 300 decimal, although you may wish to check with Microsoft for the latest permitted values. The default value is 0x78 (120 decimal).

5. Quit Registry Editor.
6. Reboot the machine.

Note: Undoing the above should be as simple as deleting the registry entries you've created.

Chapter 4 Resetting the Root Password: Windows Systems

On Windows, use the following procedure to reset the password for the MySQL `'root'@'localhost'` account. To change the password for a `root` account with a different host name part, modify the instructions to use that host name.

1. Log on to your system as Administrator.
2. Stop the MySQL server if it is running. For a server that is running as a Windows service, go to the Services manager: From the **Start** menu, select **Control Panel**, then **Administrative Tools**, then **Services**. Find the MySQL service in the list and stop it.

If your server is not running as a service, you may need to use the Task Manager to force it to stop.

3. Create a text file containing the password-assignment statement on a single line. Replace the password with the password that you want to use.

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';
```

4. Save the file. This example assumes that you name the file `C:\mysql-init.txt`.
5. Open a console window to get to the command prompt: From the **Start** menu, select **Run**, then enter `cmd` as the command to be run.
6. Start the MySQL server with the `init_file` system variable set to name the file (notice that the backslash in the option value is doubled):

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 8.0\bin"
C:\> mysqld --init-file=C:\\mysql-init.txt
```

If you installed MySQL to a different location, adjust the `cd` command accordingly.

The server executes the contents of the file named by the `init_file` system variable at startup, changing the `'root'@'localhost'` account password.

To have server output to appear in the console window rather than in a log file, add the `--console` option to the `mysqld` command.

If you installed MySQL using the MySQL Installation Wizard, you may need to specify a `--defaults-file` option. For example:

```
C:\> mysqld
      --defaults-file="C:\\ProgramData\\MySQL\\MySQL Server 8.0\\my.ini"
      --init-file=C:\\mysql-init.txt
```

The appropriate `--defaults-file` setting can be found using the Services Manager: From the **Start** menu, select **Control Panel**, then **Administrative Tools**, then **Services**. Find the MySQL service in the list, right-click it, and choose the `Properties` option. The `Path to executable` field contains the `--defaults-file` setting.

7. After the server has started successfully, delete `C:\mysql-init.txt`.

You should now be able to connect to the MySQL server as `root` using the new password. Stop the MySQL server and restart it normally. If you run the server as a service, start it from the Windows Services window. If you start the server manually, use whatever command you normally use.

Chapter 5 MySQL for Visual Studio

Table of Contents

5.1 General Information	51
5.1.1 New in Version 1.2	52
5.1.2 New in Version 2.0 (Development Release)	53
5.2 Installing MySQL for Visual Studio	58
5.3 Enabling the MySQL Toolbar	60
5.4 Making a Connection	61
5.4.1 Connect Using Server Explorer	63
5.4.2 Connect Using MySQL Connections Manager	65
5.5 Editing	66
5.5.1 MySQL SQL Editor	67
5.5.2 Code Editors	68
5.5.3 Editing Tables	70
5.5.4 Editing Views	76
5.5.5 Editing Indexes	78
5.5.6 Editing Foreign Keys	79
5.5.7 Editing Stored Procedures and Functions	80
5.5.8 Editing Triggers	82
5.6 MySQL Project Items	83
5.6.1 MySQL ASP.NET MVC Items	83
5.6.2 MySQL Windows Forms Items	92
5.7 MySQL Application Configuration Tool	94
5.7.1 Entity Framework	95
5.7.2 Web Providers	97
5.7.3 Using the MySQL Connection String Editor	101
5.8 MySQL Data Export Tool	102
5.9 DDL T4 Template Macro	111
5.10 Debugging Stored Procedures and Functions	112
5.11 MySQL for Visual Studio Frequently Asked Questions	123

5.1 General Information

MySQL for Visual Studio provides access to MySQL objects and data from Visual Studio. As a Visual Studio package, MySQL for Visual Studio integrates directly into Server Explorer providing the ability to create new connections and work with MySQL database objects.

Functionality concepts includes:

- **SQL Development:** By integrating directly into Visual Studio, database objects (tables, views, stored routines, triggers, indexes, etc) can be created, altered, or dropped directly inside Server Explorer.

Visual object editors include helpful information to guide you through the editing process. Standard data views are also available to help you view your data.

- **Query Designer:** Visual Studio's query design tool is also directly supported. With this tool, you can query and view data from tables or views while also combining filters, group conditions, and parameters. Stored routines (both with and without parameters) can also be queried.
- **Stored Routine Debugging:** Use the full debugging support for stored routines. Using the standard Visual Studio environment and controls, you can set breakpoints, add watches, and step into, out of, and

over routines and calls. Local variables can be added to the watch window and call stack navigation is also supported.

- **Entity Framework:** The Entity Framework is supported, to allow template based code generation and full support of the model designers and wizards.

For notes detailing the changes in each release, see the [MySQL for Visual Studio Release Notes](#).

5.1.1 New in Version 1.2

This section summarizes many of the new features added to 1.2.x in relation to earlier versions of MySQL for Visual Studio.

- [Support for MySQL 8.0 Features](#)
- [New or Changed Tool Support](#)
- [Version Support for Visual Studio](#)
- [Item Templates versus Project Templates](#)

For notes detailing the changes in each point release, see the [MySQL for Visual Studio Release Notes](#).

Support for MySQL 8.0 Features

- Starting with MySQL for Visual Studio 1.2.9, SSL PEM connections can be made using the classic MySQL protocol. Both PEM and PFX certificates are permitted with Connector/NET 8.0.16 or higher when the server supports SSL connections. For configuration instructions, see [SSL Connections with Server Explorer](#).
- MySQL for Visual Studio 1.2.8 supports the MySQL 8.0 release series (requires MySQL Connector/NET 6.9.12, 6.10.7, or 8.0.11) including:
 - MySQL data dictionary, which uses `INFORMATION_SCHEMA` tables rather than tables in the `mysql` database (see [MySQL Data Dictionary](#)).
 - The `caching_sha2_password` authentication plugin introduced in MySQL 8.0 (see [Caching SHA-2 Pluggable Authentication](#)).

New or Changed Tool Support

- Starting with MySQL for Visual Studio 1.2.9, the plugin detects when the version of Connector/NET has been changed after MySQL for Visual Studio was installed and prompts to update the necessary configuration files using the [Configuration Update Tool](#). Visual Studio must be restarted to activate the updated configuration files.
- The MySQL Website Configuration tool was renamed to MySQL Application Configuration and extended to automate entry updates to the `app.config` file in the MySQL for Visual Studio 1.2.9 release.

Version Support for Visual Studio

- MySQL for Visual Studio 1.2.10:
 - Support for Microsoft Visual Studio 2015 was removed.
- MySQL for Visual Studio 1.2.9:

- Support for Microsoft Visual Studio 2019 was added.
- Support for Microsoft Visual Studio 2012 and 2013 was removed.
- MySQL for Visual Studio 1.2.7:
 - Support for Microsoft Visual Studio 2017 was added.
 - Support for Microsoft Visual Studio 2010 was removed.

Item Templates versus Project Templates

Beginning with MySQL for Visual Studio 1.2.5, the project templates used to create MySQL Windows Forms and MySQL MVC projects are no longer be available, as they were replaced with [MySQL Project Items](#):

- **MySQL MVC Item** replaces *MySQL MVC Project*.
- **MySQL Windows Forms Item** replaces *Windows Form Project*.

These item templates offer the benefit of adding items to existing projects new windows forms or MVC controllers/views connected to MySQL, based on MySQL Entity Framework models, without the need of create an entirely new MySQL project.

In addition, item templates better follow the Visual Studio template standards, which are oriented to create projects regardless of the database connectivity.

For information about using Item Templates, see [Section 5.6, “MySQL Project Items”](#).

5.1.2 New in Version 2.0 (Development Release)

This section summarizes many of the new features added to the 2.0 release series in relation to the MySQL for Visual Studio 1.2 release series. MySQL for Visual Studio 2.0.5 is a development release.

New features are described in the following sections:

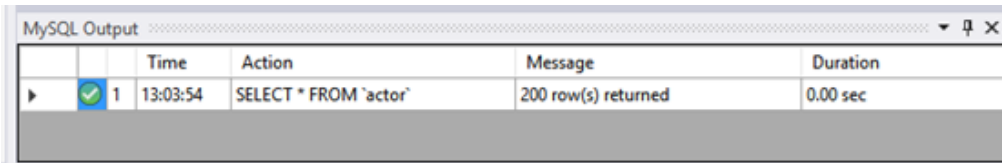
- [Viewing MySQL Query Output](#)
- [Version Support for Visual Studio](#)
- [Switching Connections from Script and Code Editors](#)
- [Making a Connection](#)
- [MySQL Toolbar](#)
- [MySQL JavaScript and Python Code Editors](#)

For notes detailing the changes in each point release, see the [MySQL for Visual Studio Release Notes](#).

Viewing MySQL Query Output

An output pane was added to the MySQL SQL, JavaScript, and Python editors to display information about each executed query. The output pane includes the information that previously appeared in the **Messages** tab.

Figure 5.1 MySQL SQL Editor Output



	Time	Action	Message	Duration
1	13:03:54	SELECT * FROM 'actor'	200 row(s) returned	0.00 sec

Version Support for Visual Studio

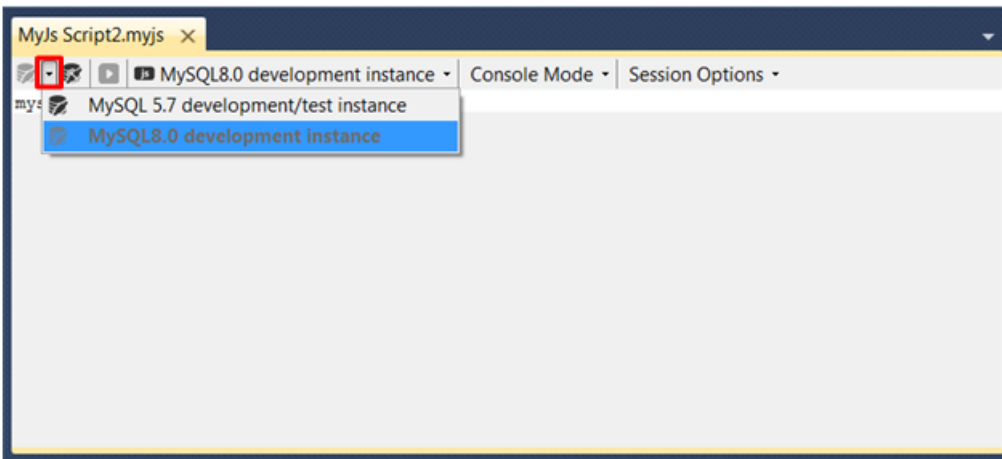
Beginning with MySQL for Visual Studio 2.0.5:

- Support for Microsoft Visual Studio 2017 was added.
- Support for Microsoft Visual Studio 2010 was removed.

Switching Connections from Script and Code Editors

A drop-down list was added to the toolbar of the SQL, JavaScript, and Python editors from which you can select a valid connection. JavaScript and Python editors show only the connections that support the X Protocol.

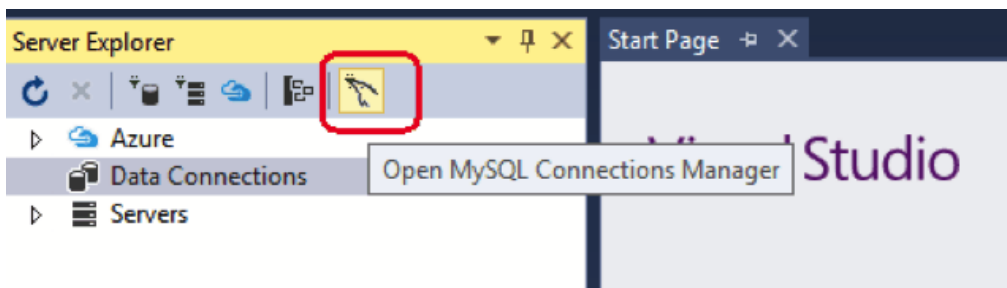
Figure 5.2 Switching Connections



Making a Connection

A new **MySQL Connections Manager** tool was added, and it can create and manage MySQL connections. It is found under the **Server Explorer**.

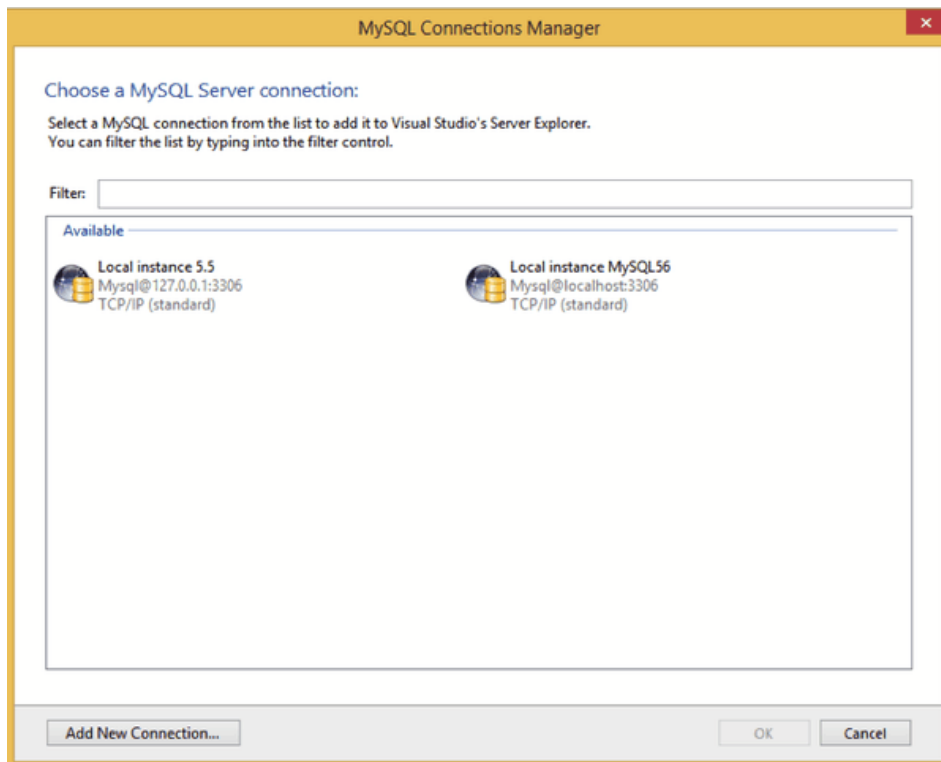
Figure 5.3 Opening the MySQL Connections Manager Dialog



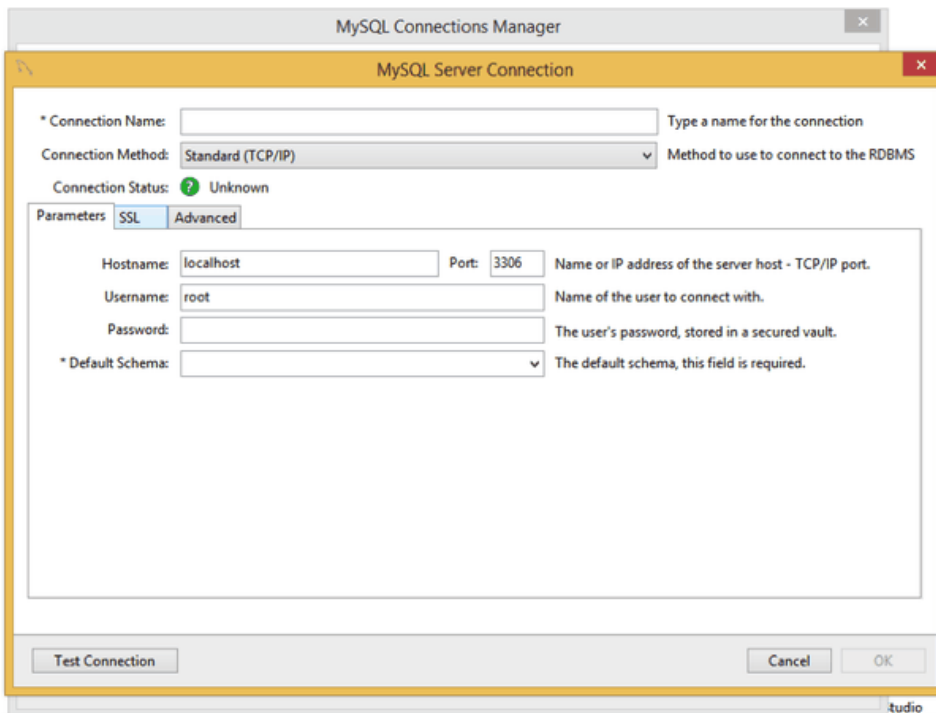
This button opens the **MySQL Connections Manager** dialog that enables the sharing of stored MySQL connections with MySQL Workbench, if it is installed. MySQL connections are displayed in a simpler way and can be created and edited from within this dialog. These connections can be imported to the Visual Studio **Server Explorer** for use with Visual Studio.

After opening the **MySQL Connections Manager**:

Figure 5.4 MySQL Connections Manager Dialog: Choosing a Connection

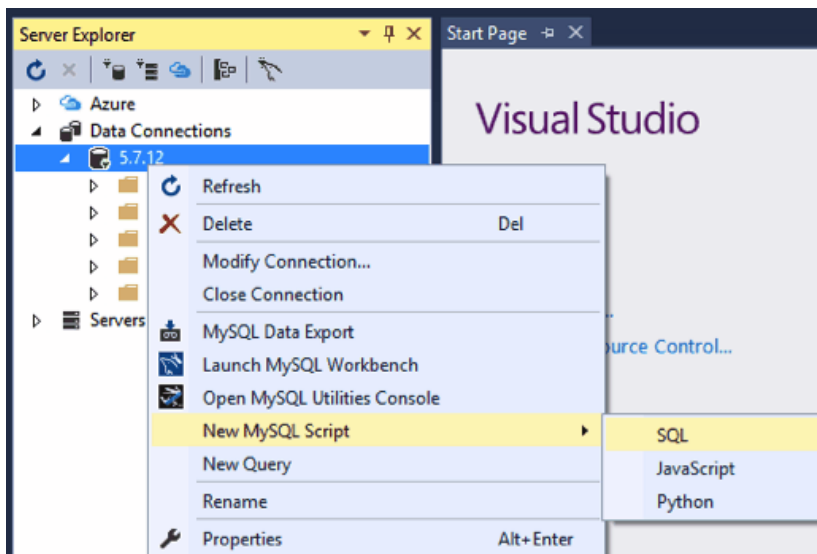


To add a new MySQL connection with the **MySQL Connections Manager**:

Figure 5.5 MySQL Connections Manager Dialog: New Connection

MySQL Toolbar

In the **Server Explorer**, and with MySQL Server 5.7, the MySQL connection context-menu was changed to show the options to create JavaScript or Python scripts, along with the existing SQL script option.

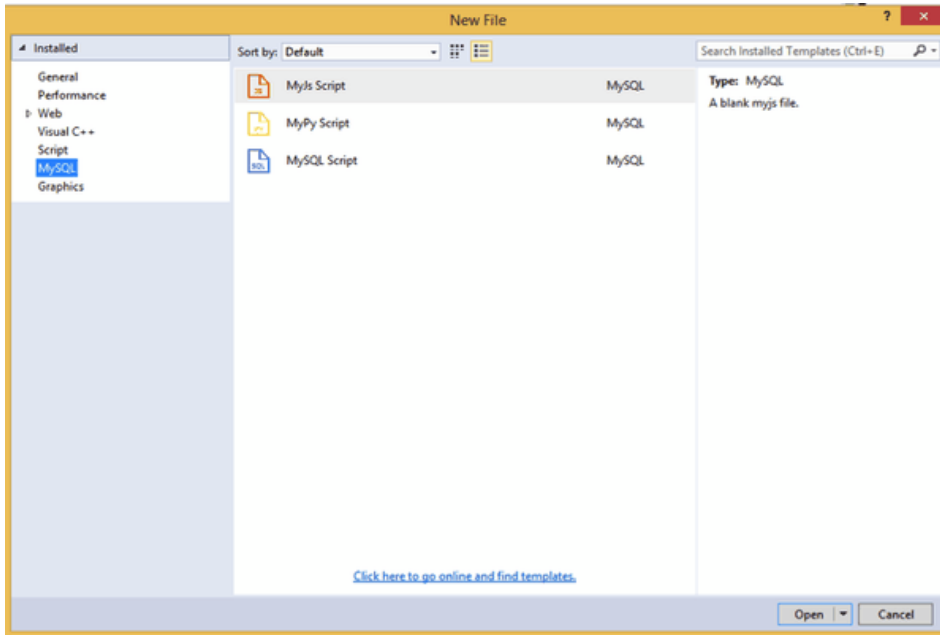
Figure 5.6 MySQL Toolbar: Create New Script

Select **JavaScript** or **Python** to launch the MySQL code editor for the selected language.

MySQL JavaScript and Python Code Editors

Use the code editor to write and execute JavaScript or Python queries with MySQL Server 5.7 and higher, or as before, use SQL queries.

Figure 5.7 MySQL Editor: Script Template



Select **MyJs Script** or **MyPy Script** to launch the MySQL code editor for the selected language.

Figure 5.8 MySQL Editor: JavaScript Code Editor

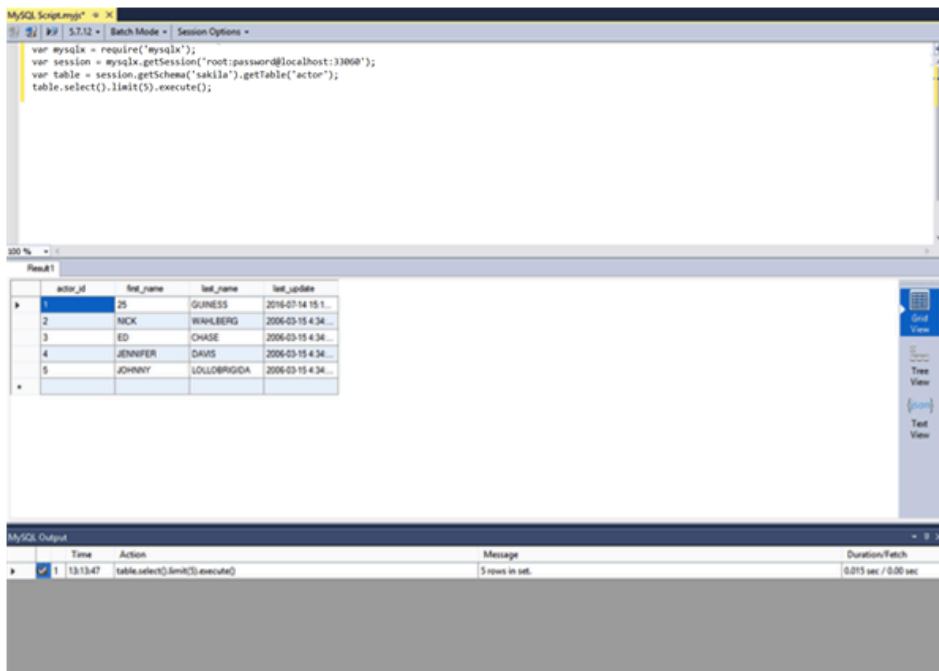
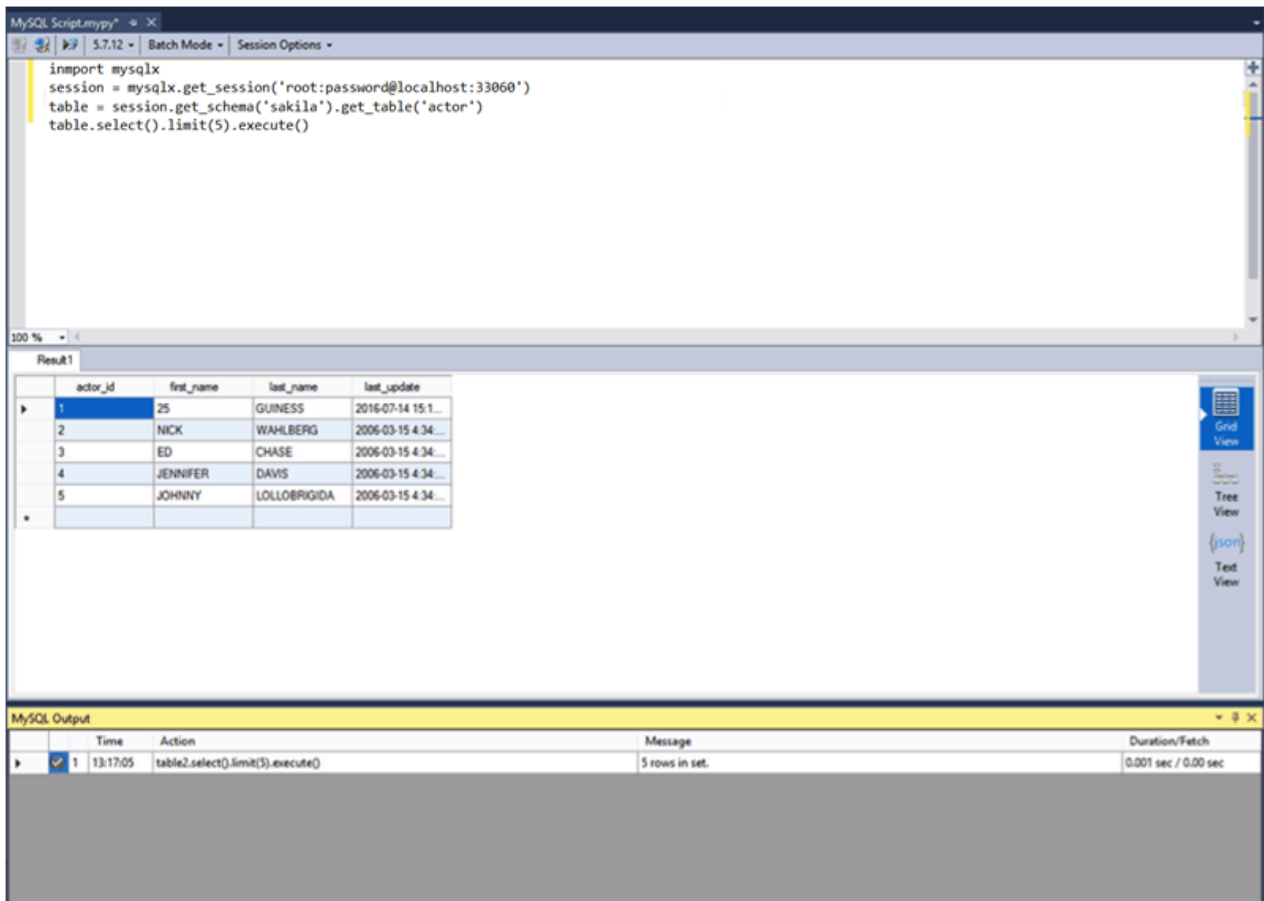


Figure 5.9 MySQL Editor: Python Code Editor



5.2 Installing MySQL for Visual Studio

MySQL for Visual Studio is an add-on for Microsoft Visual Studio that simplifies the development of applications using data stored by the MySQL RDBMS. Many MySQL for Visual Studio features also require that MySQL Connector/NET be installed on the same host where you perform Visual Studio development. Connector/NET is a separate product.

The options for installing MySQL for Visual Studio are:

- Using MySQL Installer (preferred): Download and execute the [MySQL Installer](#).

With this option you can download and install MySQL Server, MySQL for Visual Studio, and Connector/NET together from the same software package, based on the server version. Initially, MySQL Installer assists you by evaluating the software prerequisites needed for the installation. Thereafter, MySQL Installer enables you to keep your installed products updated or to easily add and remove related MySQL products.

For additional information about using MySQL Installer with MySQL products, see [Section 1.3, “MySQL Installer for Windows”](#).

- Using the standalone [Zip or MSI file](#): This option is ideal if you have MySQL Server and Connector/NET installed already. Use the information in this section to determine which version of MySQL for Visual Studio to install.

MySQL for Visual Studio Configuration Update Tool

The MySQL for Visual Studio installation updates its configuration automatically when Connector/NET is installed on your computer. However, if you install Connector/NET after MySQL for Visual Studio, or upgrade Connector/NET after you have installed MySQL for Visual Studio, you are prompted to run the Configuration Update Tool.

A configuration deviation, if present, is detected by MySQL for Visual Studio when you attempt to create an Entity Framework model or the first time you attempt to open, create, or edit a connection to MySQL. Optionally, you can have MySQL for Visual Studio update the configuration files for you when it detects configuration errors and provides a popup window prompting you to invoke the tool. Select **Yes** to run the tool. You must restart Visual Studio after the affected configuration files are updated.

Minimum Requirements

MySQL for Visual Studio operates with multiple versions of Visual Studio, although the extent of support is based on your installed versions of Connector/NET and Visual Studio.

Minimum requirements for the supported versions of Visual Studio are as follows:

- Visual Studio 2019 (Community, Professional, and Enterprise)

MySQL for Visual Studio 1.2.9 with Connector/NET 8.0.14

Tip

Connector/NET 8.0.18 (or later) is recommended.

- Visual Studio 2017 (Community, Professional, and Enterprise):

MySQL for Visual Studio 1.2.7 with Connector/NET 8.0.14 or MySQL for Visual Studio 2.0.5 with Connector/NET 6.9.8

MySQL for Visual Studio does not support Express versions of Microsoft development products, including the Visual Studio and the Microsoft Visual Web Developer.

The following table shows the support information for MySQL for Visual Studio.

Table 5.1 Support Information for Companion Products

MySQL for Visual Studio Version	MySQL Connector/NET Version Supported	Visual Studio Version Supported	MySQL Server Versions Supported	Notes
1.2 (GA)	8.0	2019, 2017, 2015, 2013, 2012	8.0, 5.7, 5.6	Support for MySQL 8.0 features requires MySQL for Visual Studio 1.2.8 or higher.
2.0 (RC)	8.0	2017, 2015, 2013, 2012	5.7, 5.6	Enables MySQL Configurations Manager and code editors (with MySQL 5.7).

MySQL Connector/NET Restrictions

MySQL for Visual Studio is closely tied to Connector/NET, but they are two separate products that can be used without one another. The following restrictions apply:

- MySQL for Visual Studio cannot be installed alongside any version of Connector/NET 6.6 and earlier, which must be removed before installing MySQL for Visual Studio.
- The following MySQL for Visual Studio features require Connector/NET:
 - The Entity Framework Designer
 - The Application Configuration tool
 - Debugging Stored Procedures and Functions
 - The DDL T4 Template Macro (to generate a database from an EF Model)

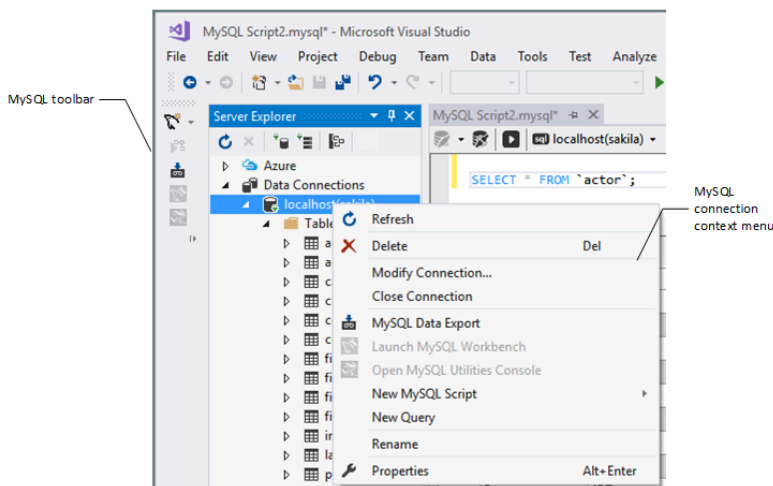
5.3 Enabling the MySQL Toolbar

The optional MySQL toolbar includes MySQL specific functionality and links to external MySQL tools such as MySQL Workbench and MySQL Utilities. Additional actions are available from the context menu for each data connection.

After installing MySQL for Visual Studio, the MySQL toolbar is available by selecting **View, Toolbars, MySQL** from the main menu. To position the MySQL toolbar within Visual Studio, do the following:

1. From the main menu, click **Tools** and then **Customize**.
2. In the **Toolbars** tab, select MySQL to highlight it. The check box should have a check mark to indicate that the toolbar is visible.
3. Select a dock location from **Modify Selection**. For example, the following figure shows the MySQL toolbar in the **Dock location: Left** position. Other dock locations are Top, Right, and Bottom.

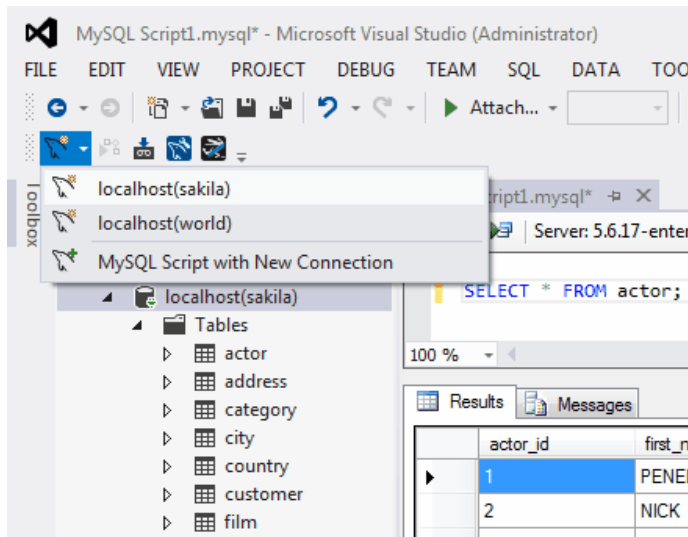
Figure 5.10 MySQL for Visual Studio Toolbar and Context Menu



The MySQL toolbar provides shortcuts to some of the main features of MySQL for Visual Studio:

- **MySQL Script Window:** Opens a new MySQL script window using the selected connection. All available MySQL connections are listed in a submenu, which can be selected on the toolbar:

Figure 5.11 The MySQL for Visual Studio Toolbar: Connections



The MySQL script window supports the IntelliSense feature for easing MySQL script creation inside Visual Studio.

- [Debug MySQL Routine](#): Starts a debugging session on a selected MySQL stored routine inside Visual Studio.
- [MySQL Data Export Tool](#): Opens a new tabbed-window of the Data Export tool.
- **MySQL Workbench SQL Editor**: Opens a new Workbench with an SQL editor window using the current MySQL connection, if [MySQL Workbench](#) has been installed.
- MySQL Utilities Console: Opens a new console window for the MySQL Utilities tool, if it is installed.

5.4 Making a Connection

MySQL for Visual Studio leverages and extends the connection capabilities of Visual Studio to create and store MySQL connections. To ensure access to the full set of connection options, install the most recent version of MySQL Connector/NET on the client computer (see [Minimum Requirements](#)).

MySQL for Visual Studio provides the following two distinct release series:

- **MySQL for Visual Studio 1.2 release series.** Provides General Availability (GA) releases for use with MySQL 5.6, 5.7, and 8.0 servers. Connections can be made using the classic MySQL protocol only. Both PEM and PFX certificates are permitted with Connector/NET 8.0.16 or higher when the server supports SSL connections. Unencrypted connections are supported by the latest versions of MySQL Connector/NET. To add or modify MySQL data connections, use the Server Explorer management console in Visual Studio (see [Section 5.4.1, “Connect Using Server Explorer”](#)).
- **MySQL for Visual Studio 2.0 release series.** Provides a development release series for use with MySQL 5.6 and 5.7 servers (version 2.0.5 is the current milestone). Basic connections are supported for both the classic MySQL protocol and X Protocol (33060 default port number). When the server supports SSL connections, PEM and PFX certificates are permitted with X Protocol; PFX certificates are permitted with the classic MySQL protocol. Use either MySQL Connections Manager or Server Explorer in Visual Studio to add or modify classic MySQL protocol connections. To add or modify X Protocol connections, use MySQL Connections Manager only (see [Section 5.4.2, “Connect Using MySQL Connections Manager”](#)).

MySQL Connections Manager was introduced in the MySQL for Visual Studio 2.0 release series to simplify the creation and management of MySQL server connections within Visual Studio. From MySQL Connections Manager, you have the option to migrate all stored connections to MySQL Workbench (if it is installed), making MySQL Workbench a central repository for MySQL connections.

Basic Connections in Visual Studio

A basic connection is either unencrypted or encrypted (in MySQL 8.0, SSL is enabled by default) and the connection is made using standard TCP/IP, which is the default connection method in MySQL for Visual Studio to connect to the MySQL RDBMS. Basic connections are easy to configure, particularly if the client application and MySQL server are on the same host computer or operate within the same local area network. For instructions on how to create a basic connection to MySQL from within Visual Studio, see [Basic Connections with Server Explorer](#) or [Basic Connections with MySQL Connections Manager](#).

SSL Connections in Visual Studio

MySQL Server uses the PEM format for certificates and private keys. Connector/NET 8.0.17 enables the use of either PEM or PFX certificates with the classic MySQL protocol when Server Explorer in Visual Studio (with MySQL for Visual Studio 1.2.9 or higher) is used to add or modify the data connection.

Both the MySQL server and the client must be configured to enable SSL encryption (see [Using Encrypted Connections](#)). In addition to providing the paths to certificate files, the client can specify the SSL mode to use for connections. When using Server Explorer, the SSL mode value is set with an advanced property. MySQL Connections Manager provides the **Use SSL Encryption** drop-down list with similar values. The following table describes the optional SSL values to select (and the files to specify) with each tool.

Table 5.2 SSL Mode Values

Server Explorer	Connections Manager	Description
None	No	Do not use SSL. No SSL files are required.
Preferred	If Available	Use SSL if the server supports it, but allow connection in all cases. Preferred is the default value with Connector/NET 8.0.11 or higher. No SSL files are required; however, providing the SSL CA file (with either a <code>.pem</code> or <code>.pfx</code> file extension) is the best practice for connections made to MySQL 8.0 servers.
Required	Require	Always use SSL and deny a connection if the server does not support SSL. Do not perform server certificate validation. No SSL files are required.
VerifyCA	Require and Verify CA	Always use SSL. Validate the certificate authorities (CA), but tolerate a name mismatch. Requires the SSL CA file. Use either a <code>.pem</code> or <code>.pfx</code> file extension.

Server Explorer	Connections Manager	Description
VerifyFull	Require and Verify Identity	Always use SSL and fail if the host name is not correct. Requires valid SSL CA, SSL Cert, and SSL Key files for PEM (.pem file extension). Requires the SSL CA file for PFX certificates (.pfx file extension).

5.4.1 Connect Using Server Explorer

This section describes how to create a new connection with or without encryption. After a connection is successfully established, all settings are saved for future use. When you start Visual Studio for the next time, open the connection node in Server Explorer to establish a connection to the MySQL server again. The instructions for setting up connections are provided in these sections.

- [Basic Connections with Server Explorer](#)
- [SSL Connections with Server Explorer](#)

To modify or delete a connection, use the Server Explorer context menu for the corresponding node. You can modify any of the settings by overwriting the existing values with new ones. The connection may be modified or deleted only if no active editor for its objects is opened; otherwise, you may lose your data.

Basic Connections with Server Explorer

To create a connection to an existing MySQL database:

1. Start Visual Studio and open the Server Explorer by clicking **Server Explorer** from the **View** menu.
2. Right-click the Data Connections node and then select **Add Connection**.
3. From the Add Connection window, click **Change** to open the Change Data Source dialog box, then do the following:
 - a. Select `MySQL Database` from the list of data sources. Alternatively, you can select `<other>`, if `MySQL Database` is absent.
 - b. Select `.NET Framework Data Provider for MySQL` as the data provider.
 - c. Click **OK** to return to the Add Connections window.
4. Type a value for each of the following connection settings:
 - **Server name:**
The name or IP address of the computer hosting the MySQL server. For example, `localhost` if the MySQL server is installed on the local computer.
 - **User name:**
The name of a valid MySQL database user account.
 - **Password:**
The password of the user account specified previously. Optionally, click **Save my password** to avoid having to enter the password in the Modify Connections window for each connection session.

- **Database name:**

The database to use as the default schema. You can leave the name blank and select a default schema later from the list of schema on the target server.

You can also set the port to connect with the MySQL server by clicking **Advanced**. To test connection with the MySQL server, set the server host name, the user name and the password, and then click **Test Connection**. If the test succeeds, the success confirmation dialog box opens.

5. Click **OK** to create and store the new connection. The new connection with its tables, views, stored procedures, stored functions, and loadable functions now appears within the **Data Connections** node of Server Explorer.

SSL Connections with Server Explorer

You can enable SSL encryption for a classic MySQL protocol connection from Server Explorer. Both SSL PEM and PFX certificate formats are permitted. In addition, MySQL Connector/NET version 8.0.17 must be installed on the client host.

To create a connection with SSL encryption enabled:

1. Add and test a new basic connection (see [Basic Connections with Server Explorer](#)).

To modify an existing connection, right-click the connection node within **Data Connections** and select **Modify Connection**.

2. In the Add (or Modify) Connection window, click **Advanced** to open the Advanced Properties dialog box. Advanced properties are categorized and presented in a two-column list, showing the property name and value field (or value list). Default values are not shown.
3. In the **Connection** property category, do the following:
 - a. Select **Connection Protocol** and then select [Socket](#) from the value list (use the arrow in the value field to open the list). This property sets the connection protocol to use standard TCP/IP.
 - b. Select **Port** and type [3306](#) in the value field.
4. In the **Authentication** property category, select [Ssl Mode](#) and choose the type of mode that best represents your connection. For a description of each mode and the required files, see [Table 5.2, "SSL Mode Values"](#).
 - For SSL PEM, use the **Ssl CA**, **Ssl Cert**, and **Ssl Key** properties to add the required files (must have a [.pem](#) file extension).
 - For SSL PFX (PKCS#12 format), use the **Certificate File**, **Certificate Password**, **Certificate Store Location**, and **Certificate Thumbprint** properties to add the required information or files (must have a [.pfx](#) file extension).

Click **OK** to close the Advanced Properties dialog box.

5. Click **Test Connection** and adjust the property values if needed.
6. Click **OK** to create and store the new or modified connection.

5.4.2 Connect Using MySQL Connections Manager


This section describes how to create a new connection with or without encryption. After a connection is successfully established, all settings are saved for future use. When you start Visual Studio for the next time, open the connection node in Server Explorer to establish a connection to the MySQL server again. The instructions for setting up connections are provided in these sections.

- [Basic Connections with MySQL Connections Manager](#)
- [SSL Connections with Connections Manager](#)

To modify or delete a connection, start MySQL Connections Manager and select an existing connection. You can modify any of the settings by overwriting the existing values with new ones. The connection may be modified or deleted only if no active editor for its objects is opened; otherwise, you may lose your data.

Basic Connections with MySQL Connections Manager

To create a connection to an existing MySQL database:

1. Click  in the Server Explorer menu bar to open the **MySQL Connections Manager** window.
2. Click **Add New Connection** to create a new connection.
3. Provide a unique name for the new connection in the required **Connection Name** field.
4. Confirm that **TCP/IP (standard)** is selected as the connection method.
5. In the **Parameters** tab, add or modify the following information:
 - **Hostname:** and **Port:**

The name (or IP address) and port number of the computer hosting the MySQL server. For example, `localhost` if the MySQL server is installed on the local computer. The default port value is 3306.
 - **Username:**

The name of a valid MySQL user account.
 - **Password:**

The password of the user account specified previously.
 - **Default Schema:**

A default schema name is required to open the connection. Select a name from the list.
6. Click **Test Connection** to verify the connection information.
7. Click **OK** to create and store the new connection. The new connection now appears in MySQL Connections Manager. Optionally, select the new connection from Connections Manager to add its tables, views, stored procedures, stored functions, and loadable functions to the **Data Connections** node in Server Explorer.

SSL Connections with Connections Manager

X Protocol connections can be configured to use SSL with PEM or PFX files. Connections must be created using the MySQL Connections Manager, which is supported by MySQL for Visual Studio 2.0.5 (or higher).

MySQL Workbench provides similar support to add PEM files, but it does not support certificates in PFX format.

Note

X Plugin must be installed to support connections using X Protocol (see [Setting Up MySQL as a Document Store](#)).

In contrast, classic MySQL protocol connections support SSL PFX files only when you use MySQL Connections Manager to configure the connection.

To create a connection to a MySQL database using SSL encryption:


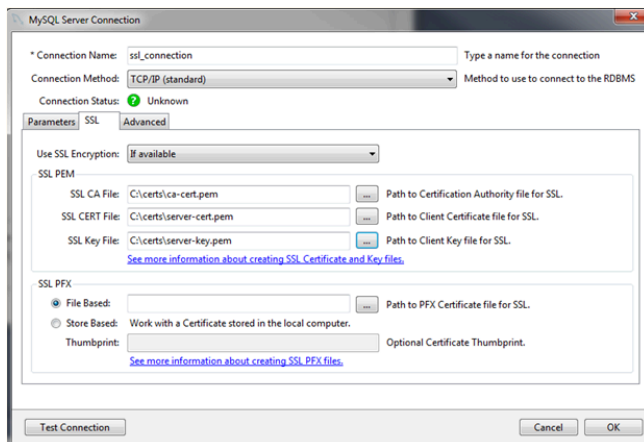
1. Click  in the Server Explorer menu bar to open the **MySQL Connections Manager** window.
2. Add and test a new basic connection (see [Basic Connections with MySQL Connections Manager](#)) or double-click an existing connection to modify it.
3. In the **SSL** tab, add a path to the SSL CA, SSL CERT, and SSL Key files within the SSL PEM area. Click **Test Connection** to verify the connection information. The next figure shows an example of SSL PEM values within this tab.

Figure 5.12 MySQL Server Connection SSL Tab



To configure SSL PFX (PKCS#12 format), select either the file-based or store-based option. Use the `.pfx` file extension to enable the correct certificate format.

4. Click **OK** to save the connection and return to the **MySQL Connections Manager** window.

Note

You must close and then reopen **MySQL Connections Manager** to apply the default schema.

5. Double-click the new SSL connection to add it to Server Explorer (or select the connection and click **OK**). To open the JavaScript or Python code editor, right-click the connection in Server Explorer and then select an editor.

5.5 Editing

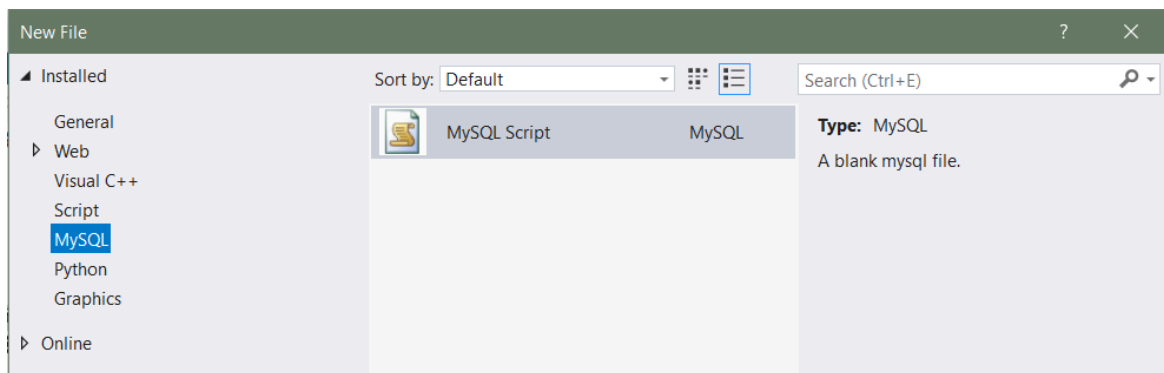
Making edits in MySQL for Visual Studio.

After you have established a connection, for example, using the **Connect to MySQL** toolbar button, you can use auto-completion as you type or by pressing **Control + J**. Depending on the context, the auto-completion dialog can show the list of available tables, table columns, or stored procedures (with the signature of the routine as a tooltip). Typing some characters before pressing **Control + J** filters the choices to those items starting with that prefix.

5.5.1 MySQL SQL Editor

The MySQL SQL Editor can be opened from the [MySQL toolbar](#) or by clicking **File, New**, and **File** from the Visual Studio main menu. This action displays the **New File** dialog.

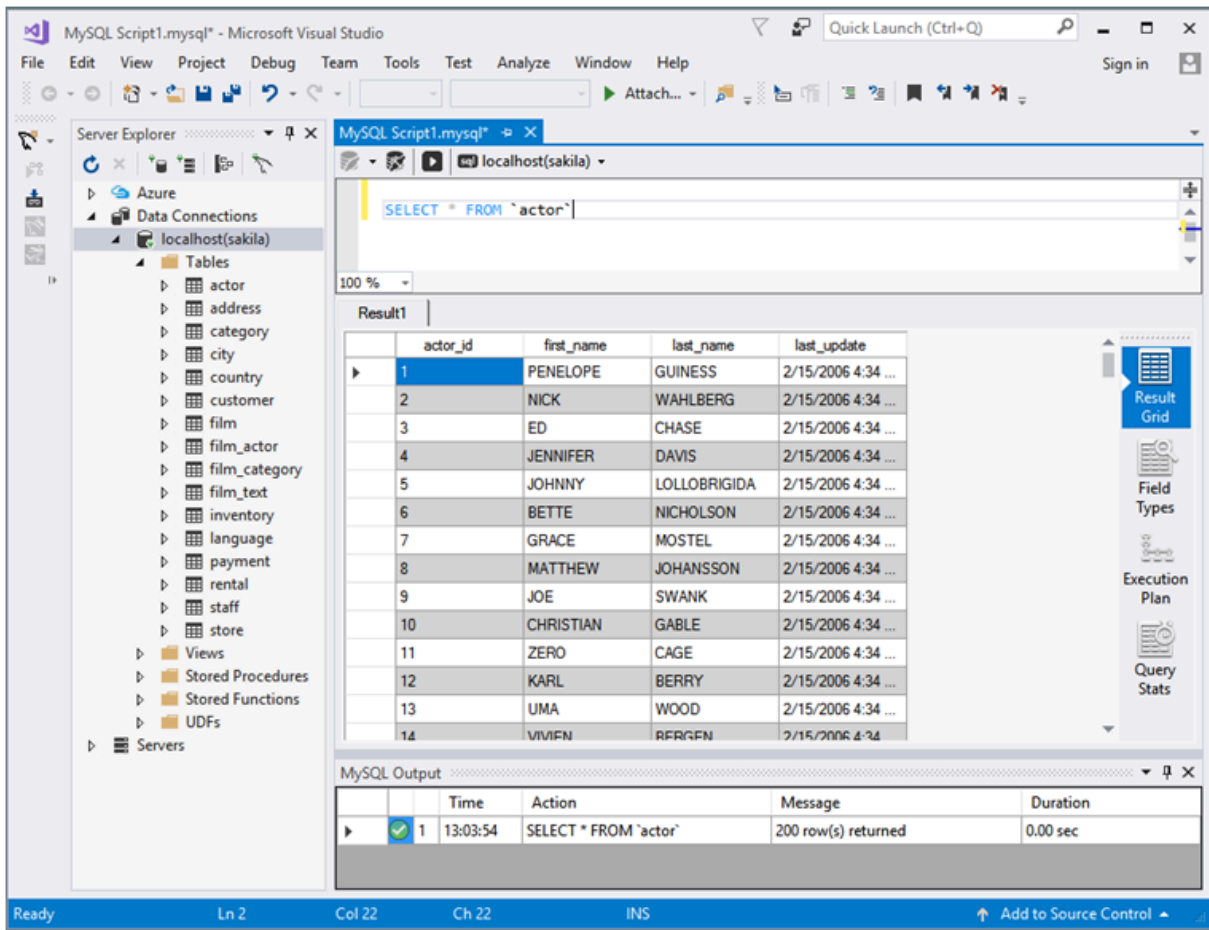
Figure 5.13 MySQL SQL Editor - New File



From the **New File** dialog, select the MySQL template, select the **MySQL Script** document, and then click **Open**.

The MySQL SQL Editor will be displayed. You can now enter SQL code as required, or connect to a MySQL server. Click the **Connect to MySQL** button in the MySQL SQL Editor toolbar. You can enter the connection details into the **Connect to MySQL** dialog that is displayed. You can enter the server name, user ID, password and database to connect to, or click the **Advanced** button to select other connection string options. Click the **Connect** button to connect to the MySQL server. To execute your SQL code against the server, click the **Run SQL** button on the toolbar.

Figure 5.14 MySQL SQL Editor - Query



The results from queries are displayed in the **Results** tab and relevant information appears in the **MySQL Output** pane. The previous example displays the query results within a Result Grid. You can also select the Field Types, Execution Plan, and Query Stats for an executed query.

5.5.2 Code Editors

This section explains how to make use of the code editors in MySQL for Visual Studio.

Introduction

MySQL for Visual Studio provides access to MySQL objects and data without forcing developers to leave Visual Studio. Designed and developed as a Visual Studio package, MySQL for Visual Studio integrates directly into Server Explorer providing a seamless experience for setting up new connections and working with database objects.

The following MySQL for Visual Studio features are available as of version 2.0.5:

- JavaScript and Python code editors, where scripts in those languages can be executed to query data from a MySQL database.
- Better integration with the Server Explorer to open MySQL, JavaScript, and Python editors directly from a connected MySQL instance.

- A newer user interface for displaying query results, where different views are presented from result sets returned by a MySQL server like:
 - Multiple tabs for each result set returned by an executed query.
 - Results view, where the information can be seen in grid, tree, or text representation for JSON results.
 - Field types view, where information about the columns of a result set is shown, such as names, data types, character sets, and more.
 - Query statistics view, displaying information about the executed query such as execution times, processed rows, index and temporary tables usage, and more.
 - Execution plan view, displaying an explanation of the query execution done internally by the MySQL server.

Getting Started

The minimum requirements are:

- MySQL for Visual Studio 2.0.5
- Visual Studio 2012
- MySQL 5.7.12 with X Plugin enabled (Code editors are not supported for use with MySQL 8.0 servers.)

To enable X Plugin for MySQL 5.7:

1. Open a command prompt and navigate to the folder with your MySQL binaries.
2. Invoke the `mysql` command-line client:

```
mysql -u user -p
```

3. Execute the following statement:

```
mysql> INSTALL PLUGIN mysqlx SONAME 'mysqlx.dll';
```

Important

The `mysql.session` user must exist before you can load X Plugin. `mysql.session` was added in MySQL 5.7.19. If your data dictionary was initialized using an earlier version you must run the `mysql_upgrade` procedure. If the upgrade is not run, X Plugin fails to start with the following error message:

```
There was an error when trying to access the server with user: mysql.session@localhost. Make sure the user is present in the server and that mysql_upgrade was ran after a server update.
```

Opening a Code Editor

Before opening a code editor that can run scripts against a MySQL server, a connection needs to be established:

1. Open the Server Explorer pane by clicking **View**.

2. Right-click the Data Connections node and select **Add Connection**.
3. In the Add Connection window, make sure the MySQL Data Provider is being used and fill in all the information.

Note

To enter the port number, click **Advanced** and set the Port among the list of connection properties.

4. Click **Test Connection** to ensure you have a valid connection, then click **OK**. The new connection with its tables, views, stored procedures, and functions now appears within the Data Connections list of Server Explorer.
5. Right-click the connection, select **New MySQL Script**, and then select the language of the editor (JavaScript or Python) to open a new MySQL script tab in Visual Studio.

To create a new editor for existing MySQL connections, you need only to do the last step.

Using the Code Editor

An open editor includes a toolbar with the actions that can be executed. The first two buttons in the toolbar represent a way to connect or disconnect from a MySQL server. If the editor was opened from the Server Explorer, the connection will be already established for the new script tab.

The third button is the **Run** button, the script contained in the editor window is executed by clicking it and results from the script execution are displayed in the lower area of the script tab.

5.5.3 Editing Tables

MySQL for Visual Studio contains a table editor, which enables the visual creation and modification of tables.

The Table Designer can be accessed through a mouse action on table-type node of Server Explorer. To create a new table, right-click the **Tables** node (under the connection node) and choose **Create Table** from the context-menu.

To modify an existing table, double-click the node of the table to modify, or right-click this node and choose the **Design** item from the context menu. Either of the commands opens the Table Designer.

Figure 5.15 Editing New Table

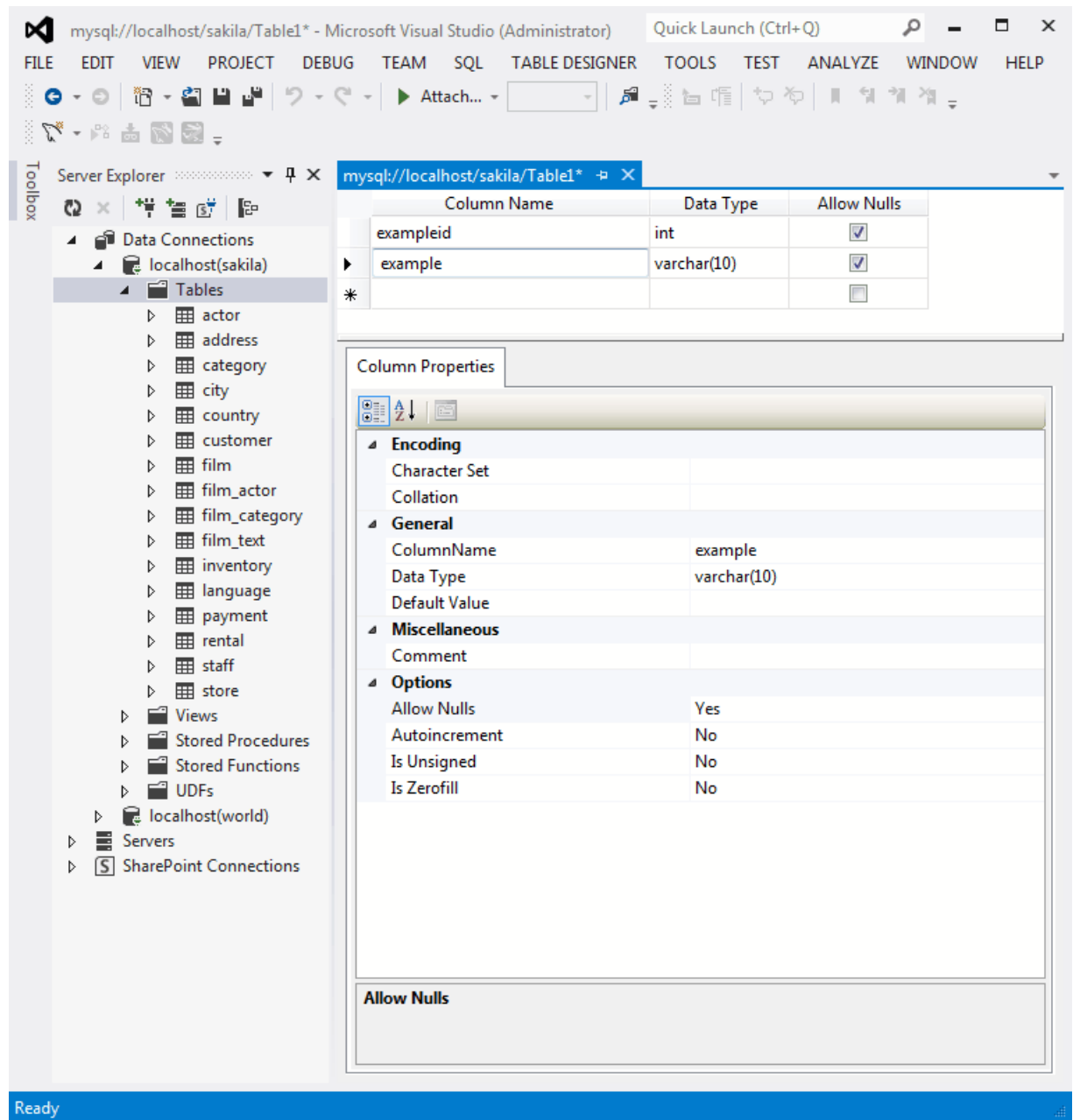


Table Designer consists of the following parts:

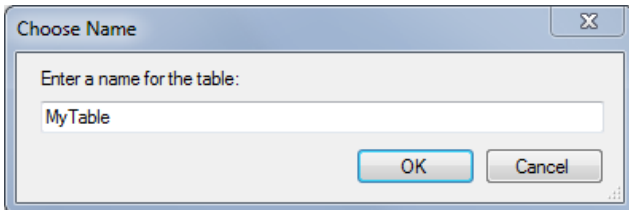
- **Columns Editor** - a data grid on top of the Table Designer. Use the Columns grid for column creation, modification, and deletion. For additional information, see [Section 5.5.3.1, "Column Editor"](#).
- **Indexes/Keys** window - a window opened from the **Table Designer** menu to manage indexes.
- **Relationships** window - a window opened from the **Table Designer** menu to manage foreign keys.
- **Column Properties** panel - a panel near the bottom of the Table Designer. Use the Column Properties panel to set advanced column options.

- **Properties** window - a standard Visual Studio Properties window, where the properties of the edited table are displayed. Use the Properties window to set the table properties. To open, right-click on a table and select the **Properties** context-menu item.

Each of these areas is discussed in more detail in subsequent sections.

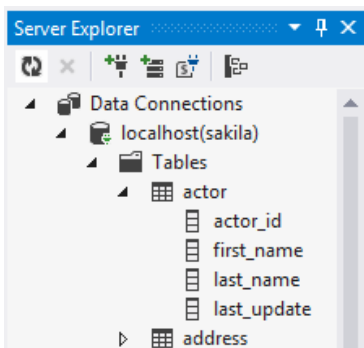
To save changes you have made in the Table Designer, press either **Save** or **Save All** on the Visual Studio main toolbar, or press **Control + S**. If you have not already named the table, you will be prompted to do so.

Figure 5.16 Choose Table Name



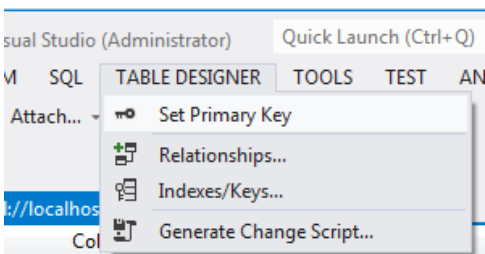
After the table is created, you can view it in the **Server Explorer**.

Figure 5.17 Newly Created Table



The Table Designer main menu lets you set a [primary key](#) column, edit relationships such as [foreign keys](#), and create [indexes](#).

Figure 5.18 Table Designer Main Menu



5.5.3.1 Column Editor

You can use the Column Editor to set or change the name, data type, default value, and other properties of a table column. To set the focus to a needed cell of a grid, use the mouse click. Also you can move through the grid using **Tab** and **Shift + Tab** keys.

To set or change the name, data type, default value and comment of a column, activate the appropriate cell and type the desired value.

To set or unset flag-type column properties (**NOT NULL**, auto incremented, flags), select or deselect the corresponding check boxes. The set of column flags depends on the data type of the column.

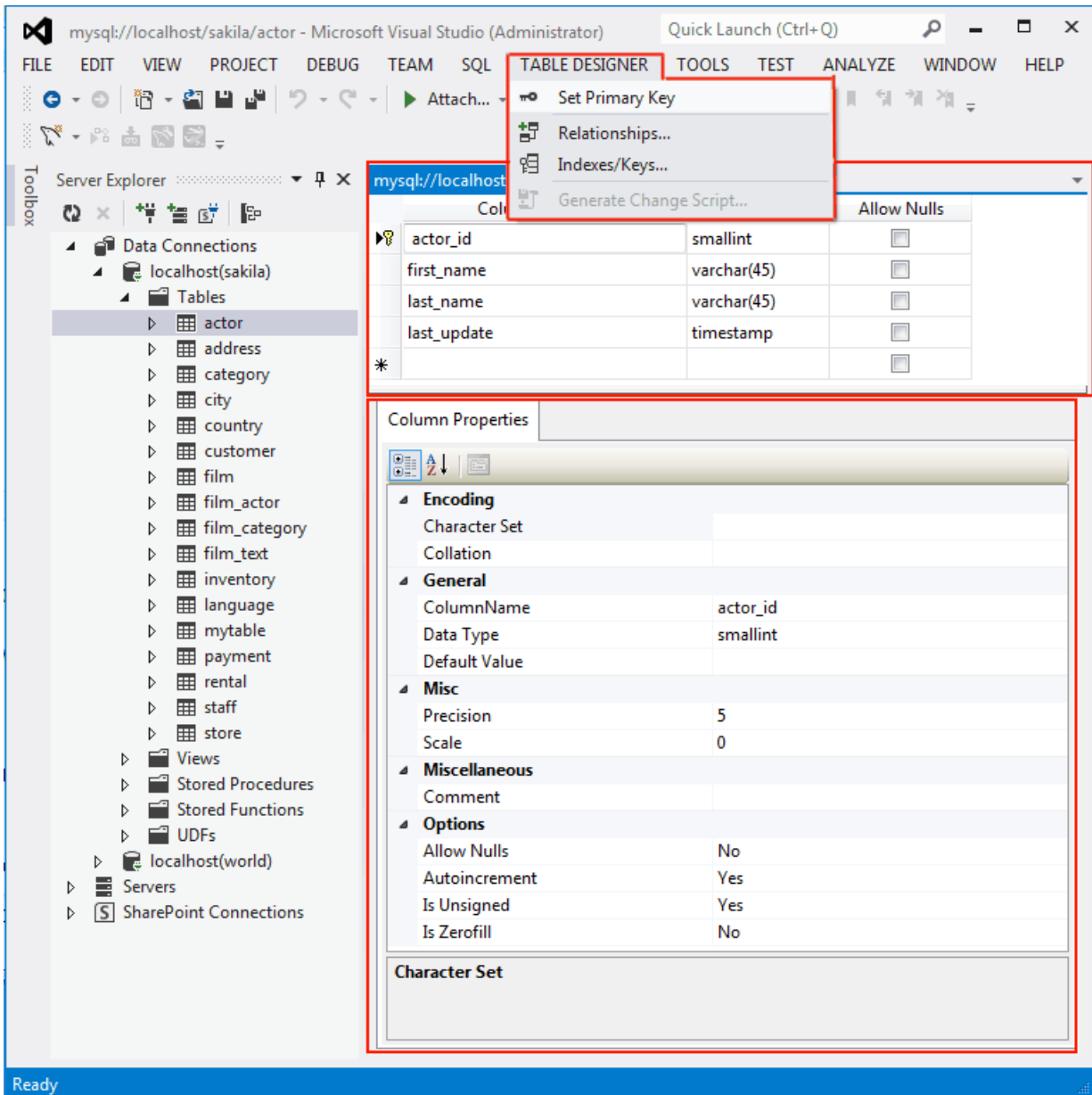
To reorder columns, index columns or foreign key columns in the Column Editor, select the whole column to reorder by clicking the selector column on the left of the column grid. Then move the column by using **Control+Up** (to move the column up) or **Control+Down** (to move the column down) keys.

To delete a column, select it by clicking the selector column on the left of the column grid, then press the **Delete** button on a keyboard.

5.5.3.2 Column Properties

The **Column Properties** tab can be used to set column options. In addition to the general column properties presented in the Column Editor, in the **Column Properties** tab you can set additional properties such as Character Set, Collation and Precision.

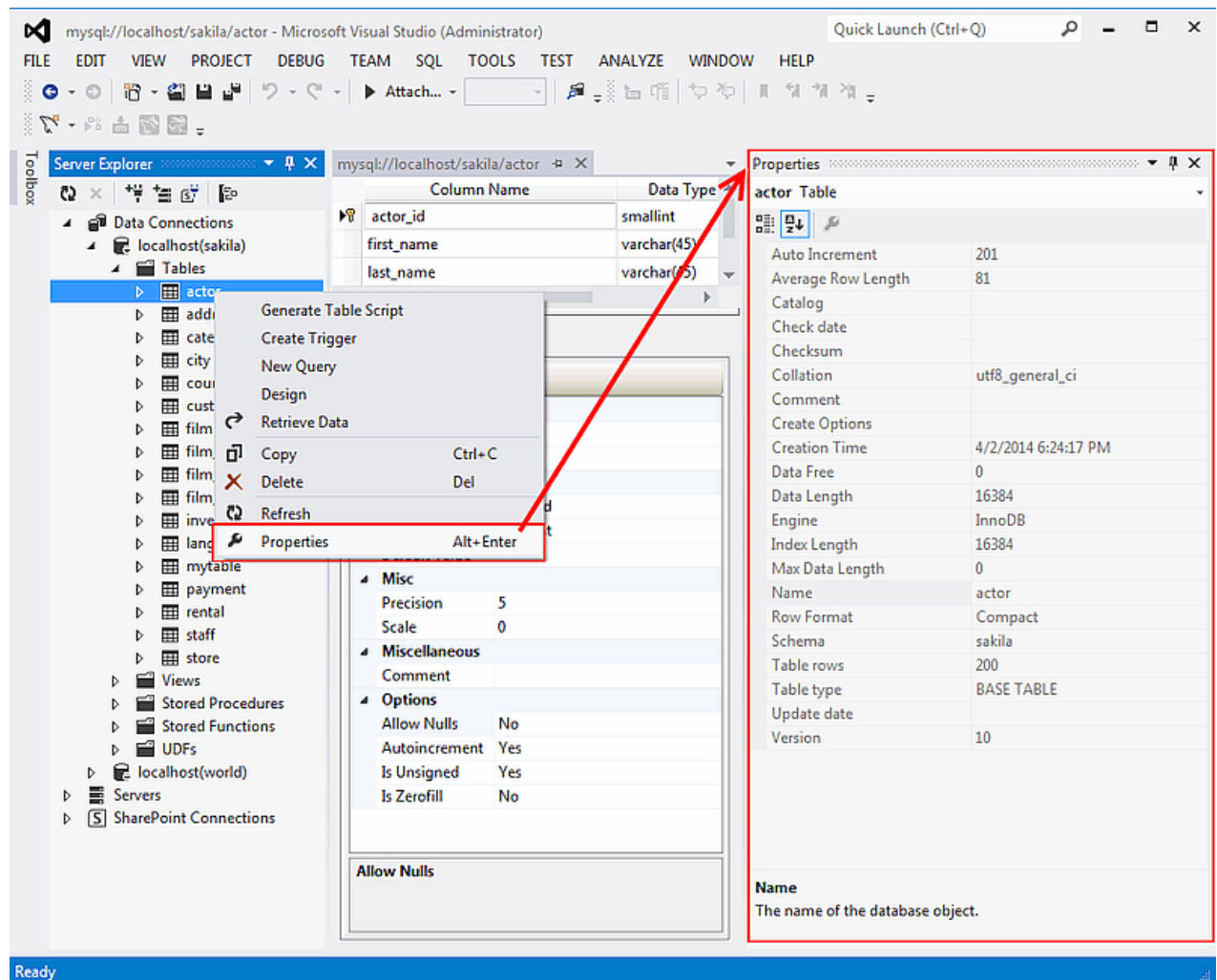
Figure 5.19 Column Properties Panel



5.5.3.3 Table Properties

To bring up Table Properties select the table and right-click to activate the context menu. Select **Properties**. The **Table Properties** dockable window will be displayed.

Figure 5.20 Table Properties Panel



The following table properties are listed under table properties, and many are fully described in the [SHOW TABLE STATUS](#) MySQL documentation.

- **Auto Increment:** The next `AUTO_INCREMENT` value.
- **Average Row Length:** The `AVG_ROW_LENGTH` value.
- **Character Set:** The Charset value.
- **Collation:** The Collation value.
- **Comment:** Table comments.
- **Data Directory:** The directory used to store data files for this table.
- **Index Directory:** The directory used to store index files for this table.
- **Maximum Rows:** Value of the `MAX_ROWS` property.
- **Minimum Rows:** Value of the `MIN_ROWS` property.
- **Name:** Name of the table.

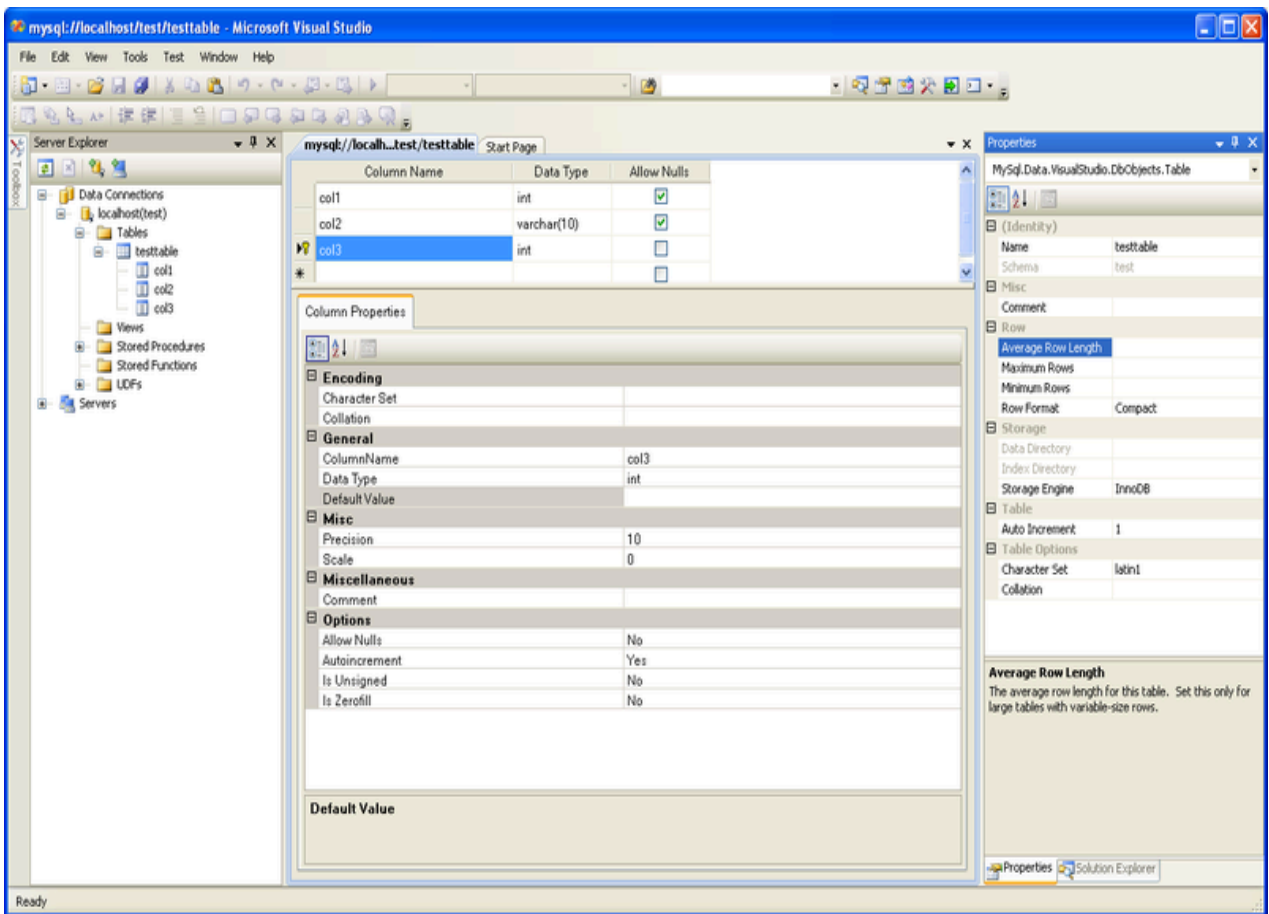
- **Row Format:** The `ROW_FORMAT` value.
- **Schema:** The schema this table belongs to.
- **Storage Engine.**

Note

In MySQL 5.5 and higher, the default storage engine for new tables is [InnoDB](#). See [Introduction to InnoDB](#) for more information about the choice of storage engine, and considerations when converting existing tables to [InnoDB](#).

The property `Schema` is read-only.

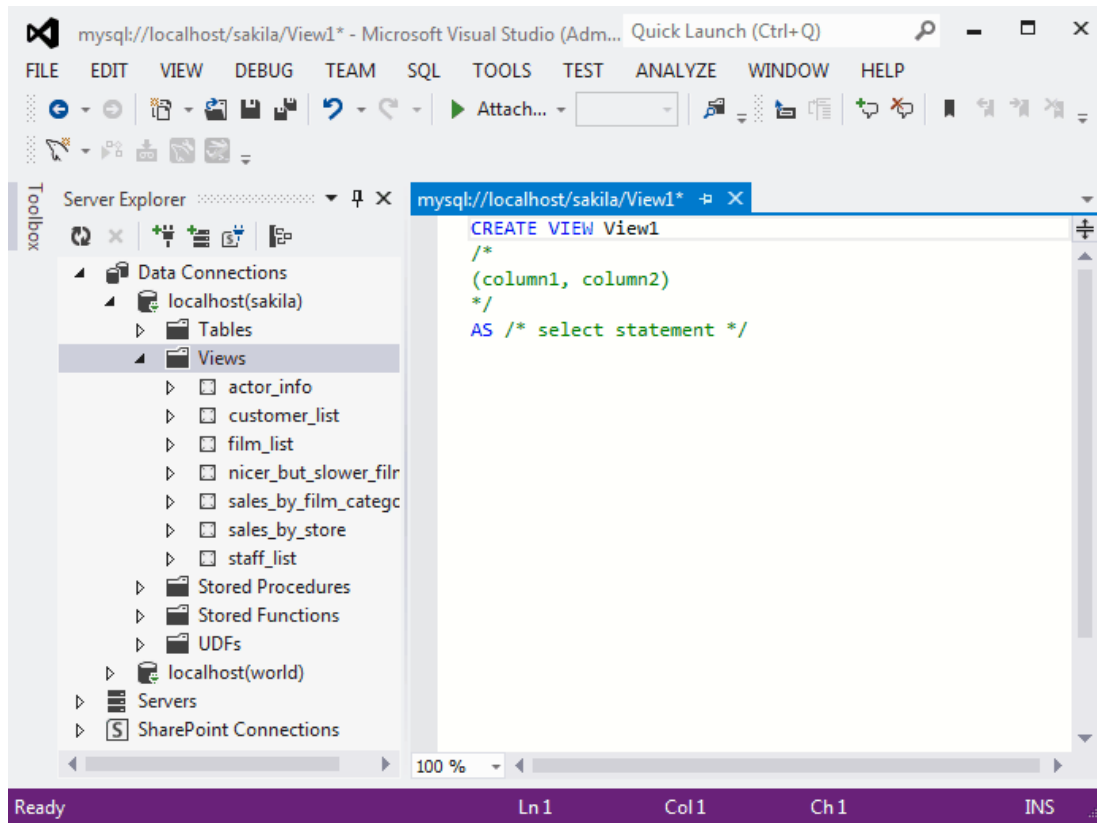
Figure 5.21 Table Properties



5.5.4 Editing Views

To create a new view, right-click the Views node under the connection node in Server Explorer. From the node's context menu, choose the **Create View** command. This command opens the SQL Editor.

Figure 5.22 Editing View SQL



You can then enter the SQL for your view, and then execute the statement.

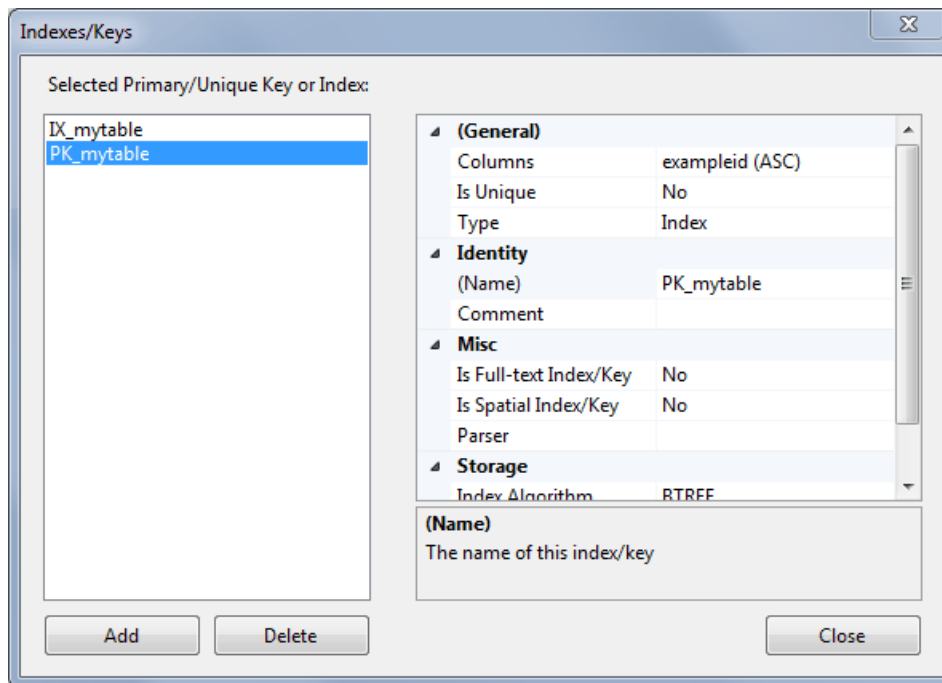
To modify an existing view, double-click a node of the view to modify, or right-click this node and choose the **Alter View** command from a context menu. Either of the commands opens the SQL Editor.

All other view properties can be set in the **Properties** window. These properties are:

- **Catalog:** The [TABLE_CATALOG](#).
- **Check Option:** Whether or not the [WITH CHECK OPTION](#) clause is present. For additional information, see [The View WITH CHECK OPTION Clause](#).
- **Definer:** Creator of the object.
- **Definition:** Definition of the view.
- **Is Updatable:** Whether or not the view is [Updatable](#). For additional information, see [Updatable and Insertable Views](#).
- **Name:** The name of the view.
- **Schema:** The schema which owns the view.
- **Security Type:** The [SQL SECURITY](#) value. For additional information, see [Stored Object Access Control](#).

Some of these properties can have arbitrary text values, others accept values from a predefined set. In the latter case, set the desired value with an embedded combobox.

Figure 5.24 Indexes/Keys Dialog



To remove an index, select it in the list box on the left, and click the **Delete** button.

To change index settings, select the needed index in the list box on the left. The detailed information about the index is displayed in the panel on the right hand side. Change the desired values.

5.5.6 Editing Foreign Keys

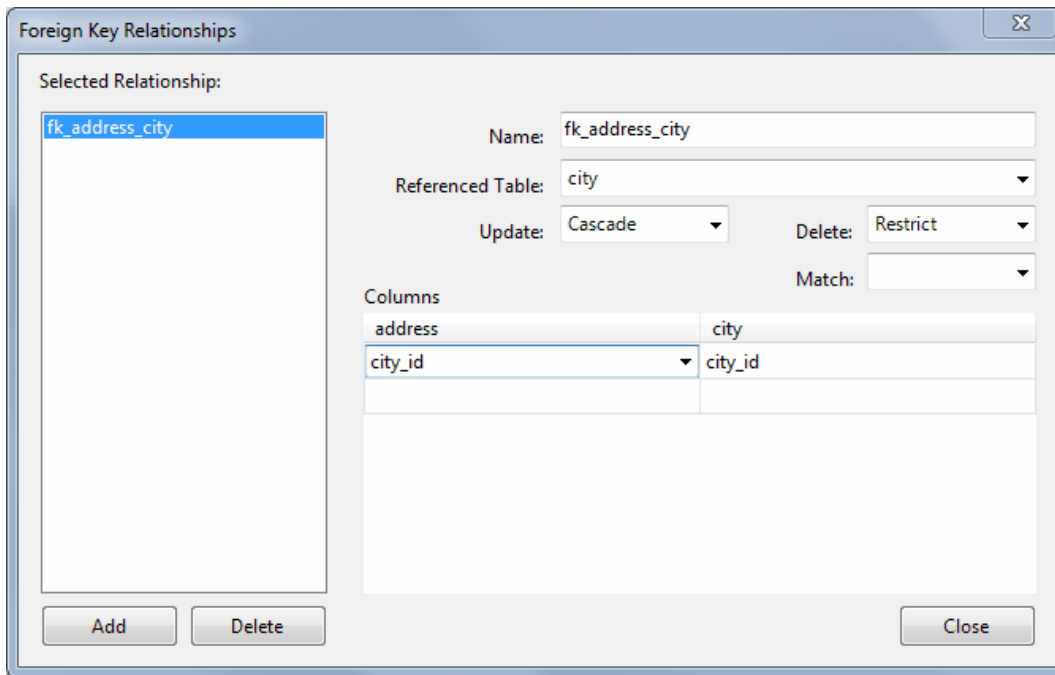
You manage [foreign keys](#) for [InnoDB](#) tables using the **Foreign Key Relationships** dialog.

To add a foreign key, select **Table Designer, Relationships...** from the main menu. This displays the **Foreign Key Relationship** dialog. Click **Add**. You can then set the foreign key name, referenced table name, foreign key columns, and actions upon update and delete.

To remove a foreign key, select it in the list box on the left, and click the **Delete** button.

To change foreign key settings, select the required foreign key in the list box on the left. The detailed information about the foreign key is displayed in the right hand panel. Change the desired values.

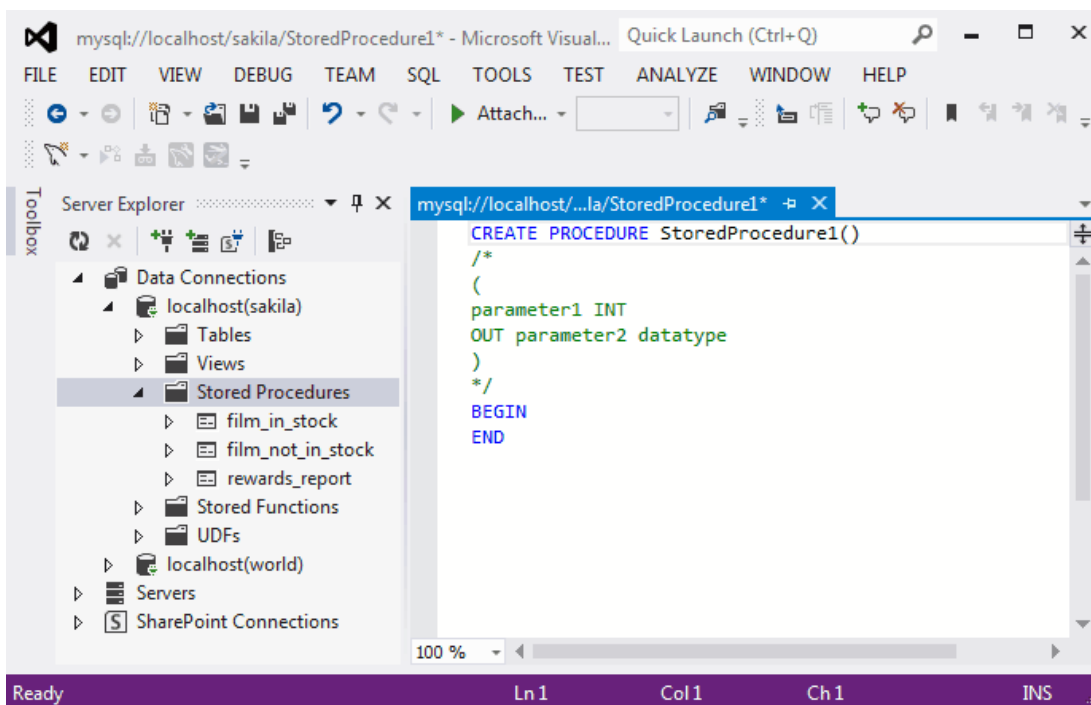
Figure 5.25 Foreign Key Relationships Dialog



5.5.7 Editing Stored Procedures and Functions

To create a new stored procedure, right-click the **Stored Procedures** node under the connection node in Server Explorer. From the node's context menu, choose the **Create Routine** command. This command opens the SQL Editor.

Figure 5.26 Edit Stored Procedure SQL



To create a new stored function, right-click the **Functions** node under the connection node in Server Explorer. From the node's context menu, choose the **Create Routine** command.

To modify an existing stored routine (procedure or function), double-click the node of the routine to modify, or right-click this node and choose the **Alter Routine** command from the context menu. Either of the commands opens the SQL Editor.

Routine properties can be viewed in the **Properties** window. These properties are:

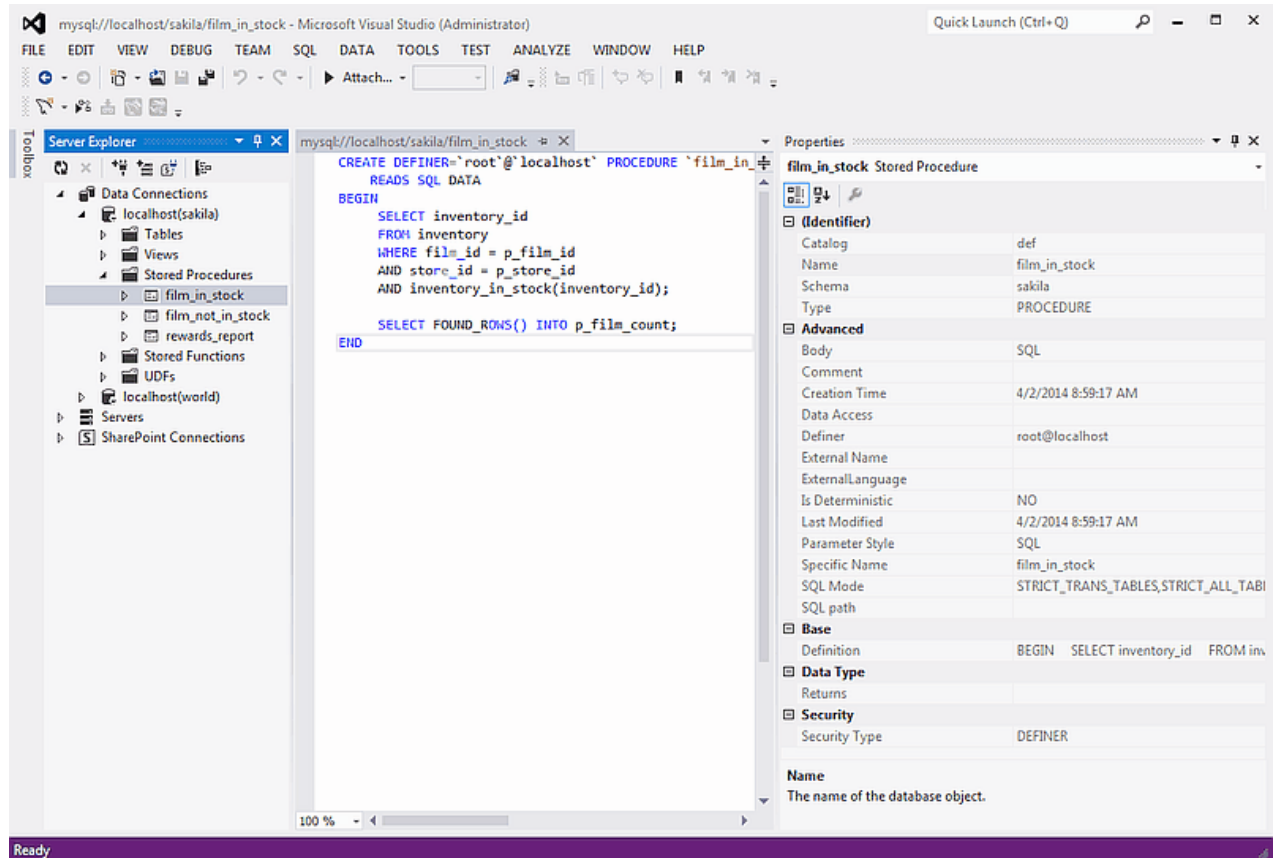
- Body
- Catalog
- Comment
- Creation Time
- Data Access
- Definer
- Definition
- External Name
- External Language
- Is Deterministic
- Last Modified
- Name
- Parameter Style
- Returns
- Schema
- Security Type
- Specific Name
- SQL Mode
- SQL Path
- Type

Some of these properties can have arbitrary text values, others accept values from a predefined set. In both cases, these values cannot be set from the properties panel.

You can also set all the options directly in the SQL Editor, using the standard [CREATE PROCEDURE](#) or [CREATE FUNCTION](#) statement.

To save changes you have made, use either **Save** or **Save All** buttons of the Visual Studio main toolbar, or press **Control + S**.

Figure 5.27 Stored Procedure SQL Saved



To observe the runtime behavior of a stored routine and debug any problems, use the Stored Procedure Debugger. For additional information, see [Section 5.10, “Debugging Stored Procedures and Functions”](#).

5.5.8 Editing Triggers

To create a new trigger, right-click the node of the table in which to add the trigger. From the node's context menu, choose the **Create Trigger** command. This command opens the SQL Editor.

To modify an existing trigger, double-click the node of the trigger to modify, or right-click this node and choose the **Alter Trigger** command from the context menu. Either of the commands opens the SQL Editor.

To create or alter the trigger definition using SQL Editor, type the trigger statement in the SQL Editor using standard SQL.

Note

Enter only the trigger statement, that is, the part of the `CREATE TRIGGER` query that is placed after the `FOR EACH ROW` clause.

All other trigger properties are set in the Properties window. These properties are:

- Definer
- Event Manipulation
- Name

- Timing

Some of these properties can have arbitrary text values, others accept values from a predefined set. In the latter case, set the desired value using the embedded combo box.

The properties [Event Table](#), [Schema](#), and [Server](#) in the Properties window are read-only.

To save changes you have made, use either **Save** or **Save All** buttons of the Visual Studio main toolbar, or press **Control + S**. Before changes are saved, you will be asked to confirm the execution of the corresponding SQL query in a confirmation dialog.

To observe the runtime behavior of a stored routine and debug any problems, use the Stored Procedure Debugger. For additional information, see [Section 5.10, “Debugging Stored Procedures and Functions”](#).

5.6 MySQL Project Items

This two-part tutorial uses MySQL MVC Item templates to set up a MVC web application. In the second part of the tutorial, a Windows Forms Item with MySQL connectivity is created.

Minimum Requirements

- MySQL 5.5 installed on a host that is accessible.
- MySQL for Visual Studio 1.2.5.
- Visual Studio 2012, the professional edition.
- MySQL Connector/NET is required to use web providers in the generated web application.

5.6.1 MySQL ASP.NET MVC Items

To add a MySQL MVC Item to an existing MVC project, first add a MySQL Entity Framework model. Skip this step if you have already done this.

Configure the project to use MySQL with an Entity Framework. There are two ways to do this:

- Manually add the references needed (EntityFramework, MySql.Data & MySql.Data.Entity), and add the required configuration to the [web.config](#) configuration file
- Or (preferred), take advantage of the **MySQL Website Configuration** tool, which allows either Entity Framework 5 or 6 with MySQL. For additional information about this tool, see [Section 5.7, “MySQL Application Configuration Tool”](#).

Once you have configured the project to use MySQL with Entity Framework, proceed to create the model using the standard **ADO.NET Entity Data Model** wizard. For MySQL MVC Item Templates, you need to add the model under the "Models" folder, as illustrated below:

Figure 5.28 ADO.NET Entity Data Model

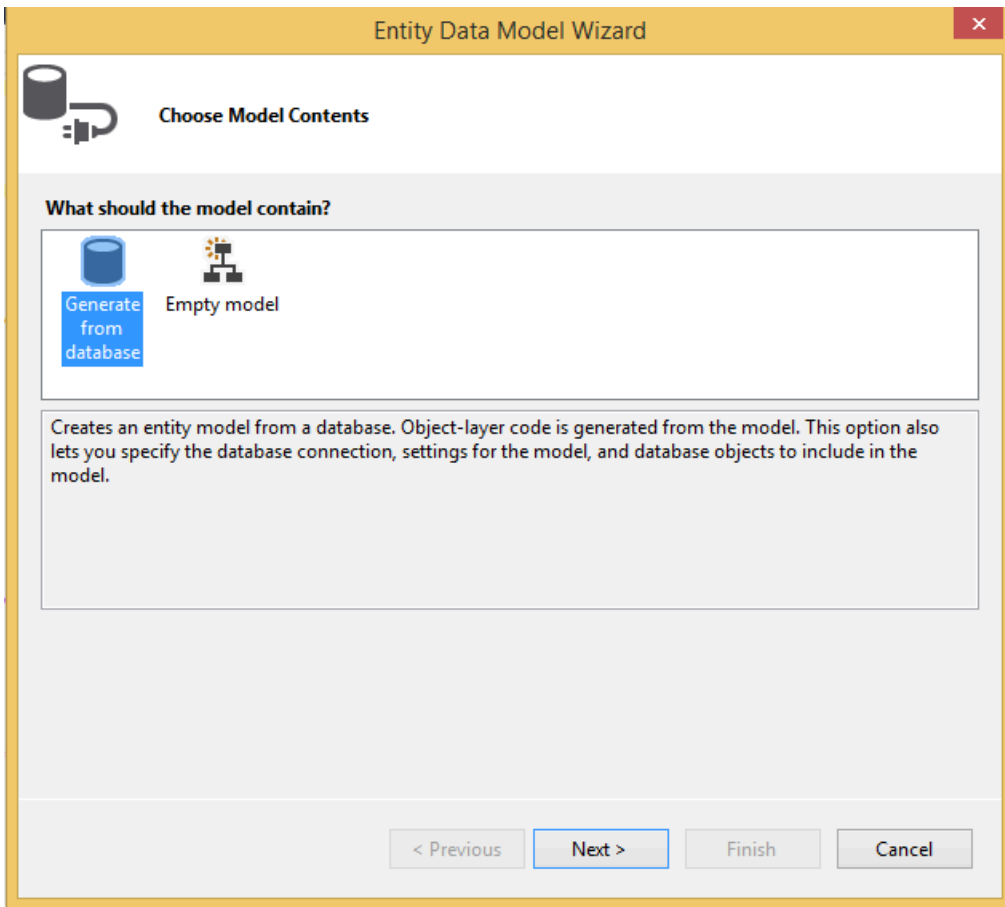
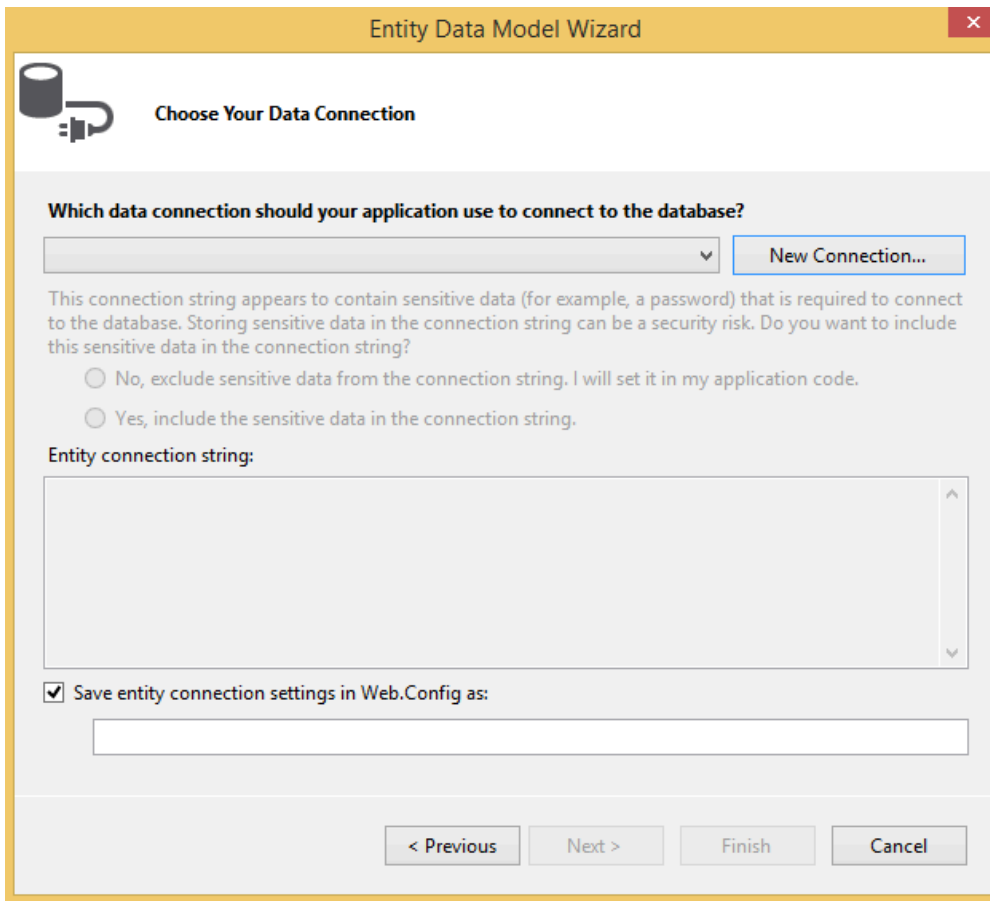
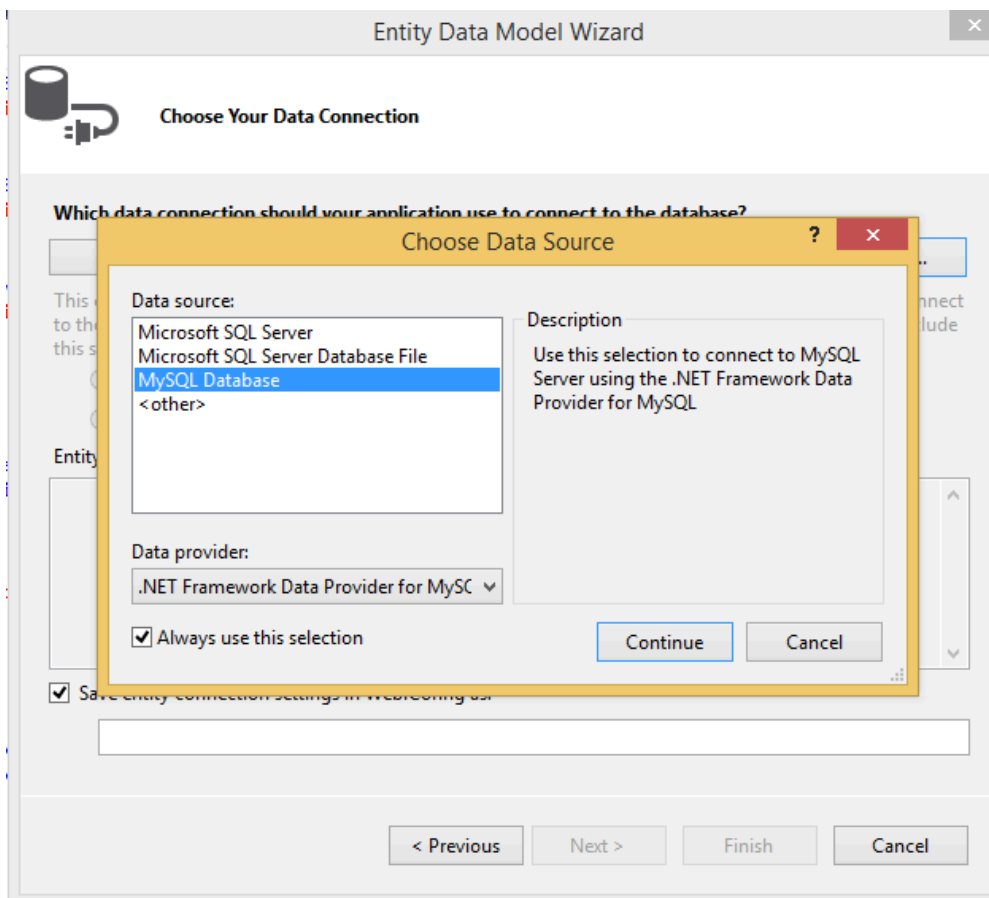


Figure 5.29 Choose or create a new MySQL connection

The screenshot shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The dialog has a title bar with the text 'Entity Data Model Wizard' and a close button. Below the title bar is a header area with a database icon and the text 'Choose Your Data Connection'. The main content area contains the following elements:

- A question: "Which data connection should your application use to connect to the database?"
- A dropdown menu for selecting a connection, with a 'New Connection...' button to its right.
- A warning message: "This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?"
- Two radio button options:
 - No, exclude sensitive data from the connection string. I will set it in my application code.
 - Yes, include the sensitive data in the connection string.
- A label: "Entity connection string:"
- A large text area for entering the connection string.
- A checked checkbox: "Save entity connection settings in Web.Config as:"
- A text box for specifying the configuration key name.
- Navigation buttons at the bottom: "< Previous", "Next >", "Finish", and "Cancel".

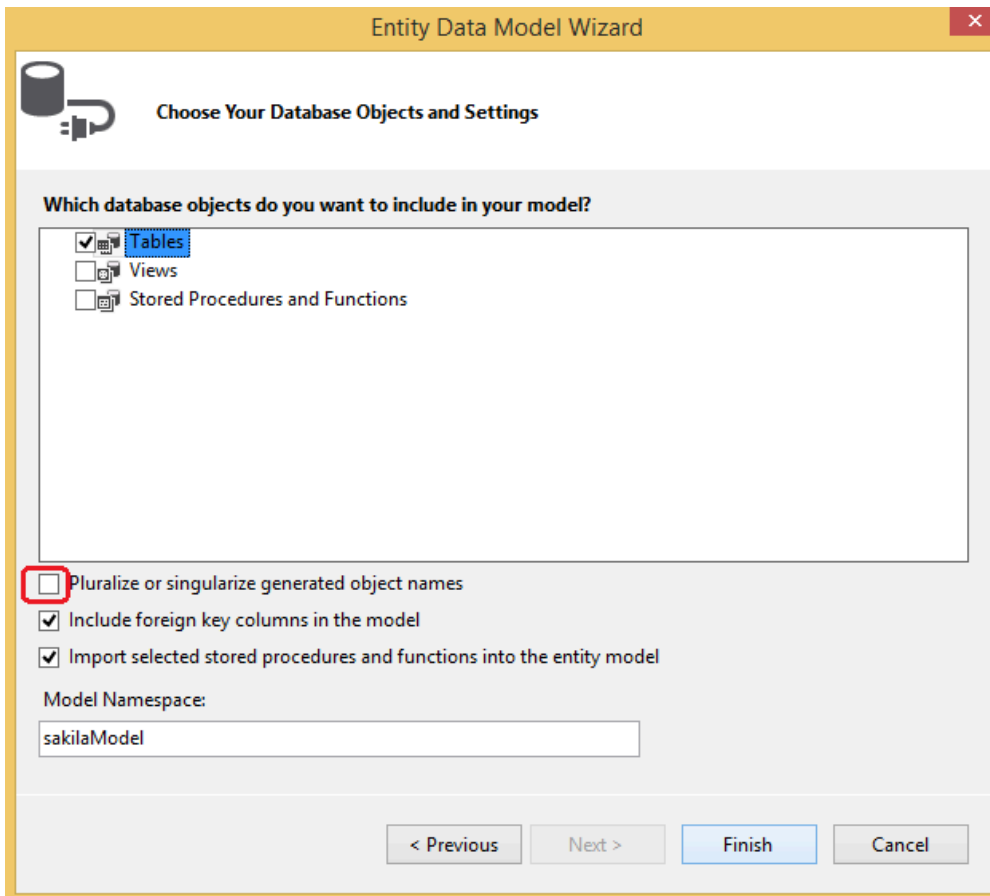
Figure 5.30 Creating a new MySQL connection

After selecting the MySQL connection, you need to select the database objects to include in the model.

Important

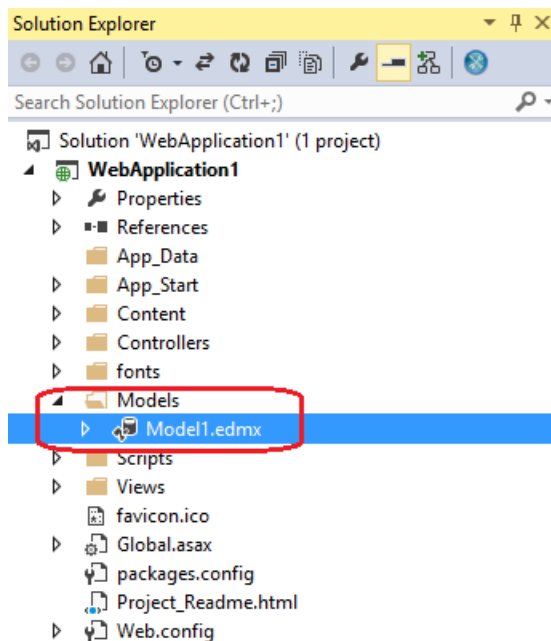
The **Pluralize or singularize generated object names** option must remain unchecked, otherwise the MySQL MVC Item Template will not function properly.

Figure 5.31 Selecting the database object to include in the model



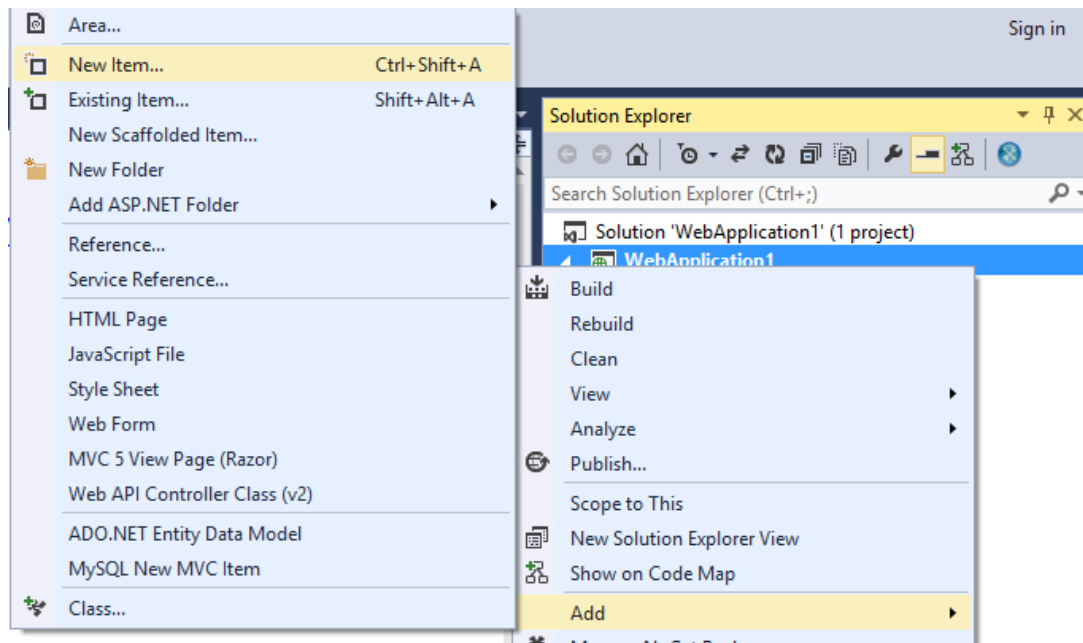
Click **Finish** to generate the model, as demonstrated below:

Figure 5.32 Creating the MySQL Entity Framework model

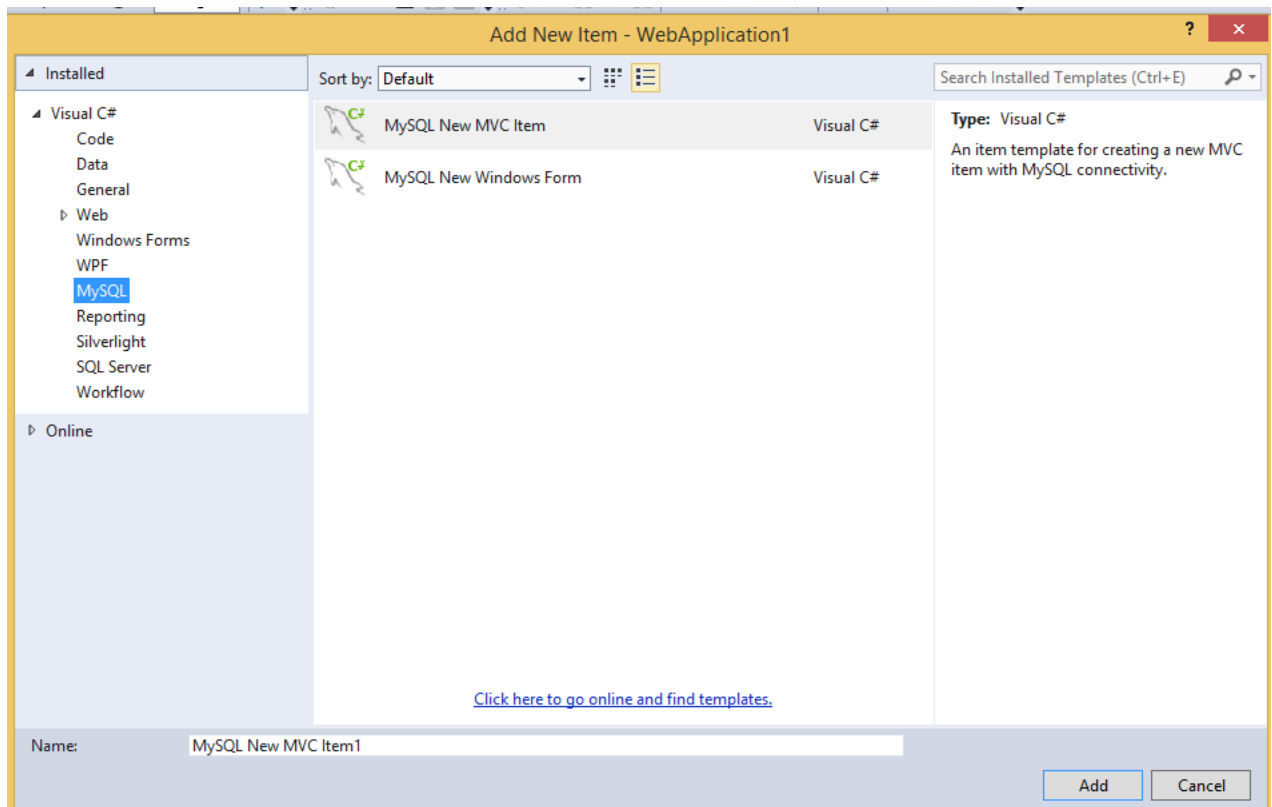


Now, generate a new MySQL MVC Item. Right-click on the project, and select **Add New Item** from the contextual menu.

Figure 5.33 Add New Item



This launches the **Add New Item** wizard. The **MySQL** menu offers two options: **MySQL New MVC Item** and **MySQL New Windows Form**. Select **MySQL New MVC Item**, and then click **Add**.

Figure 5.34 The MySQL menu options

This opens the **MVC Item Template** dialog. Now select the MySQL model and entity that you want to use to create the MVC item. The model dropdown list is populated based on all the MySQL Entity Framework models available in the project, entities dropdown list is populated with entities available for the selected model.

Figure 5.35 MySQL MVC Item Template Dialog

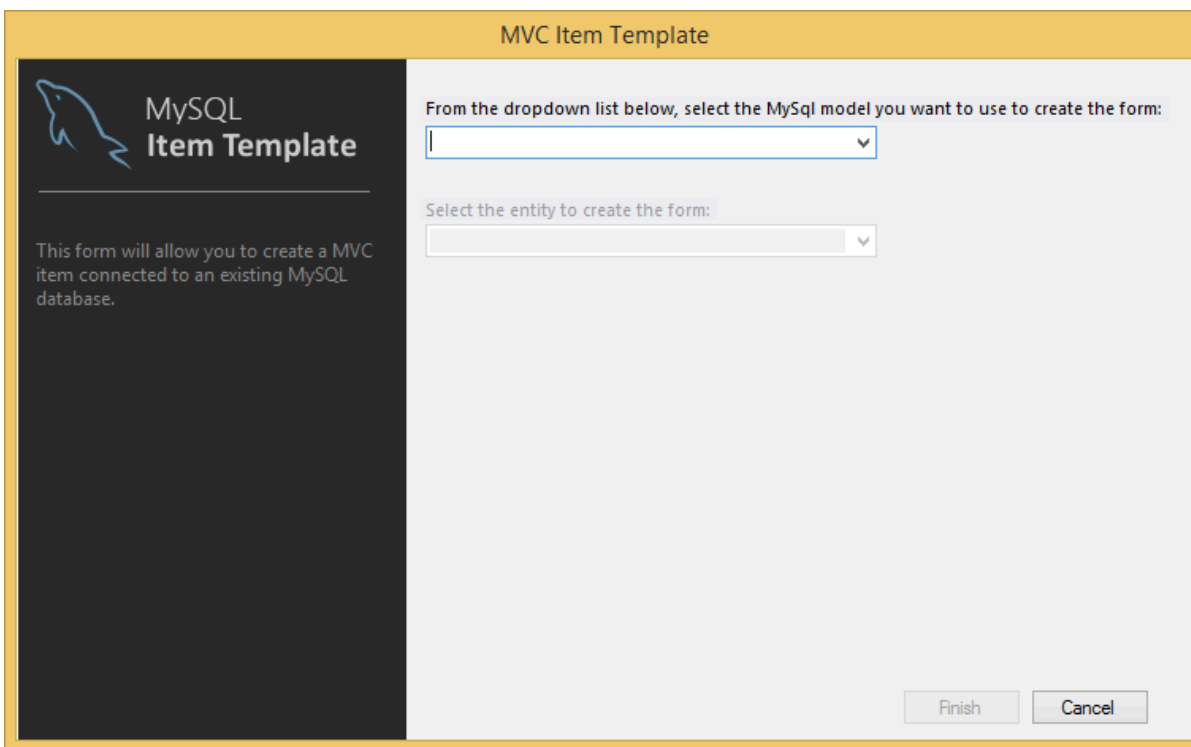
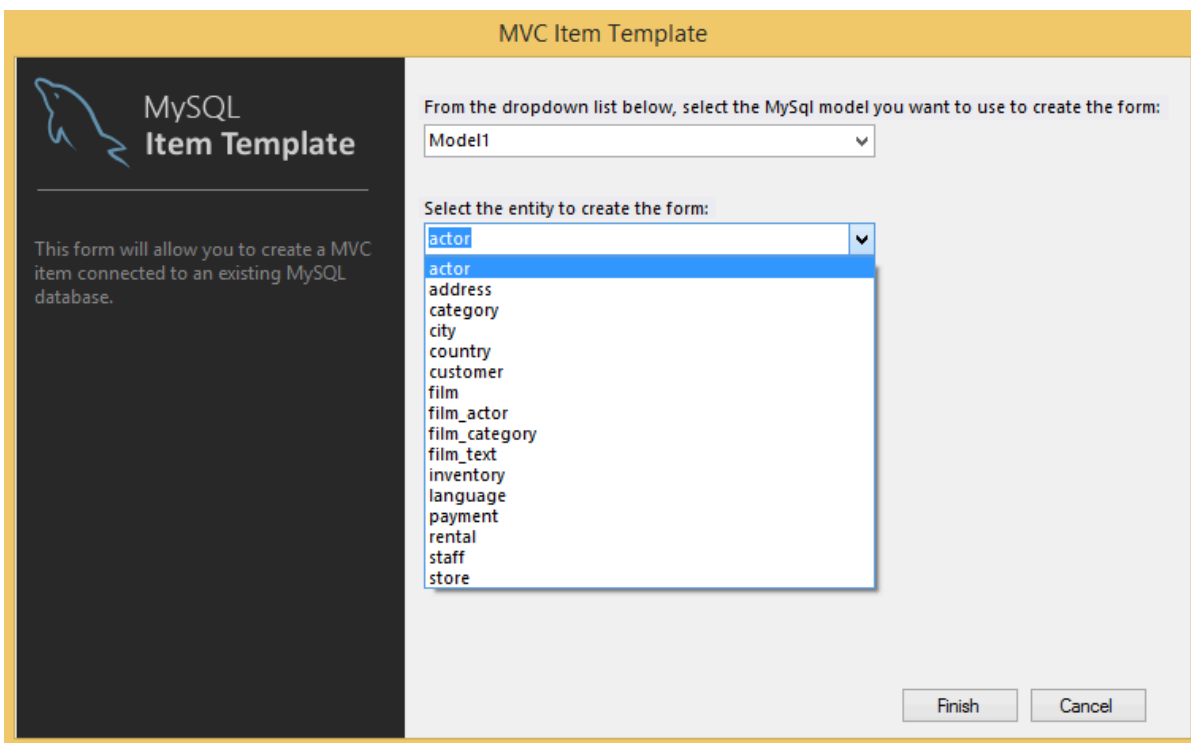
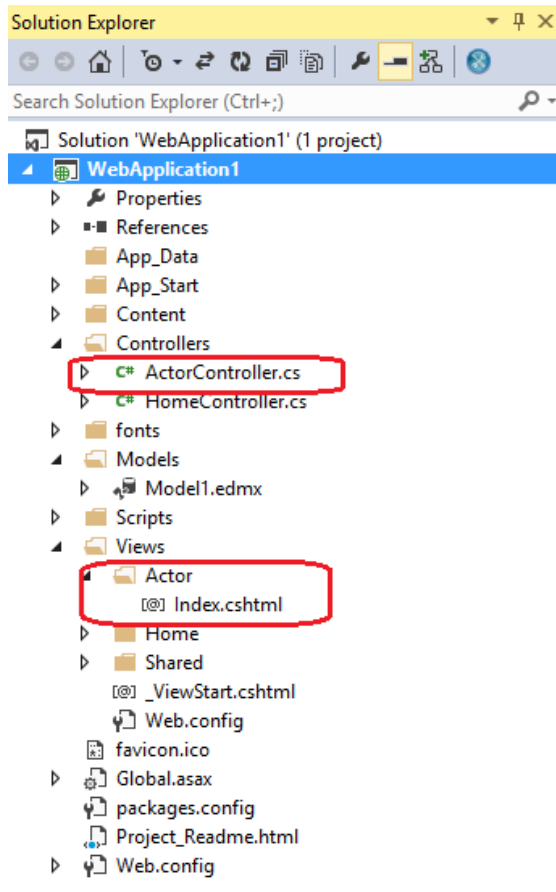


Figure 5.36 MySQL MVC Item Template



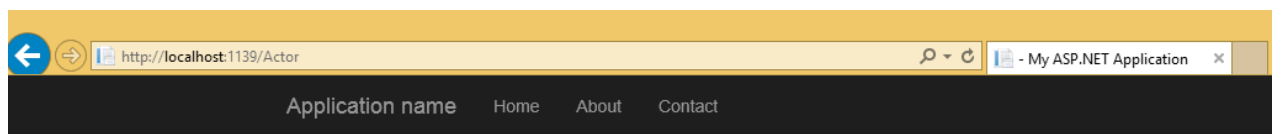
After selecting the model and entity to create the item, click **Finish**, and a new controller and view matching the selected entity will be added to the project. These contain the necessary back end code to render the *entity* data.

Figure 5.37 New controller and view added to the project



You can now execute the application. In our example we used the Sakila database and generated an Actor controller:

Figure 5.38 The Actor View



Actor List

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2/15/2006 4:34:33 AM
2	NICK	WAHLBERG	2/15/2006 4:34:33 AM
3	ED	CHASE	2/15/2006 4:34:33 AM
4	JENNIFER	DAVIS	2/15/2006 4:34:33 AM
5	JOHNNY	LOLLOBRIGIDA	2/15/2006 4:34:33 AM
6	BETTE	NICHOLSON	2/15/2006 4:34:33 AM
7	GRACE	MOSTEL	2/15/2006 4:34:33 AM
8	MATTHEW	JOHANSSON	2/15/2006 4:34:33 AM
9	JOE	SWANK	2/15/2006 4:34:33 AM
10	CHRISTIAN	GABLE	2/15/2006 4:34:33 AM

1 2 3 4 5 >

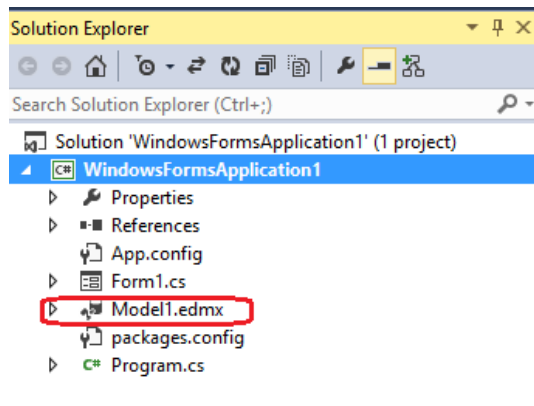
5.6.2 MySQL Windows Forms Items

This portion of the tutorial describes how to create a Windows Form with MySQL connectivity. The item template to use for this operation is named `MySQL New Windows Form`. To open the Add New Item window, right-click your windows form application and select **MySQL** from the list of installed items.

The item template for adding a new MySQL Windows Form is similar to the MySQL MVC item template (see [Section 5.6.1, "MySQL ASP.NET MVC Items"](#)), but with three major differences:

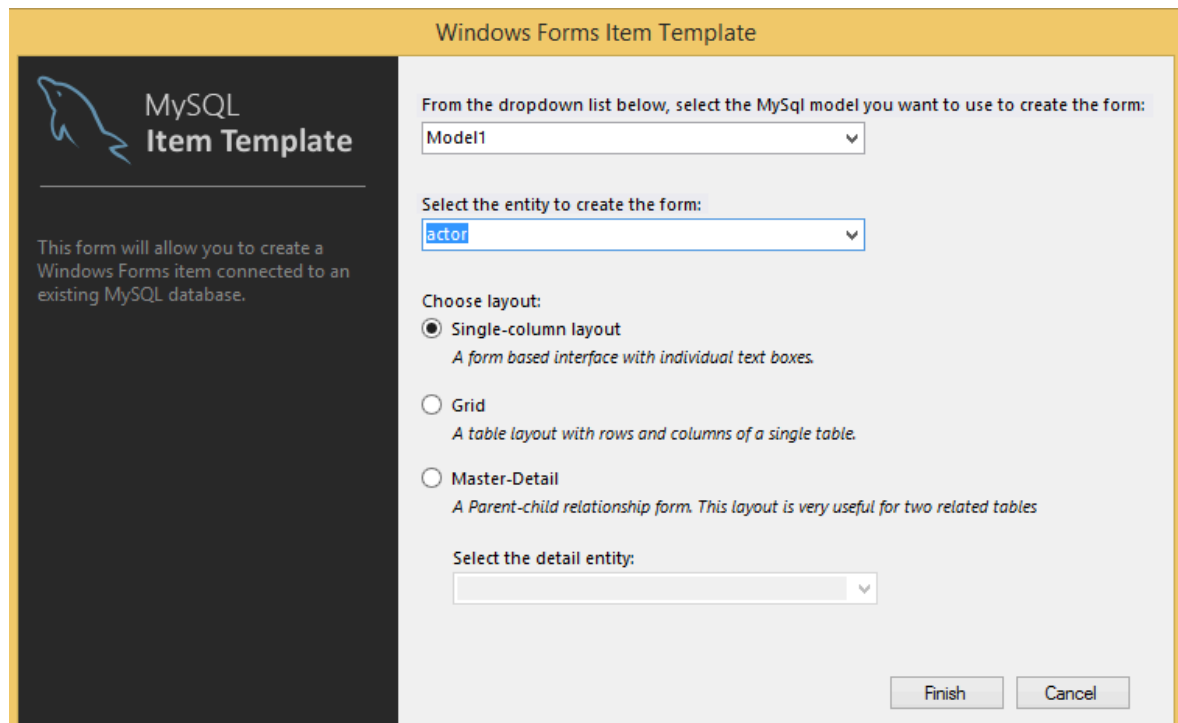
- You can create the MySQL Entity Framework model under the root path of the project.

Figure 5.39 A MySQL Entity Framework model created in a Windows Form Application



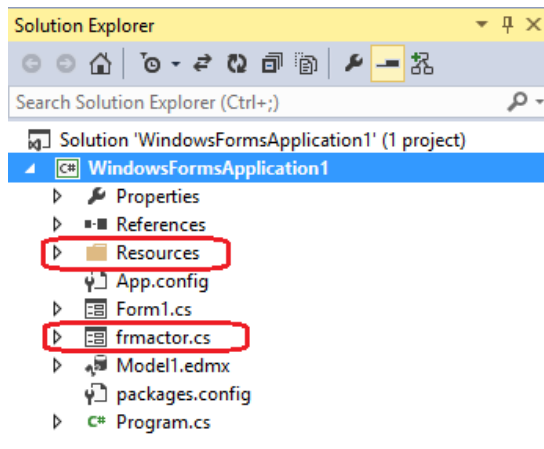
- When selecting the desired entity, you can also select the layout type in which the new form will display the entity data.

Figure 5.40 The "MySQL Windows Form" Item Template dialog, with the layout options



- A **Resources** folder is added to the project that contains images used by the icons for the generated form.

Figure 5.41 The Resources Folder and New Form



The new form will have all the necessary back-end code to display the entity data, with the user interface (UI) based on the previously selected layout.

Figure 5.42 The "frmactor" Form in Design Mode

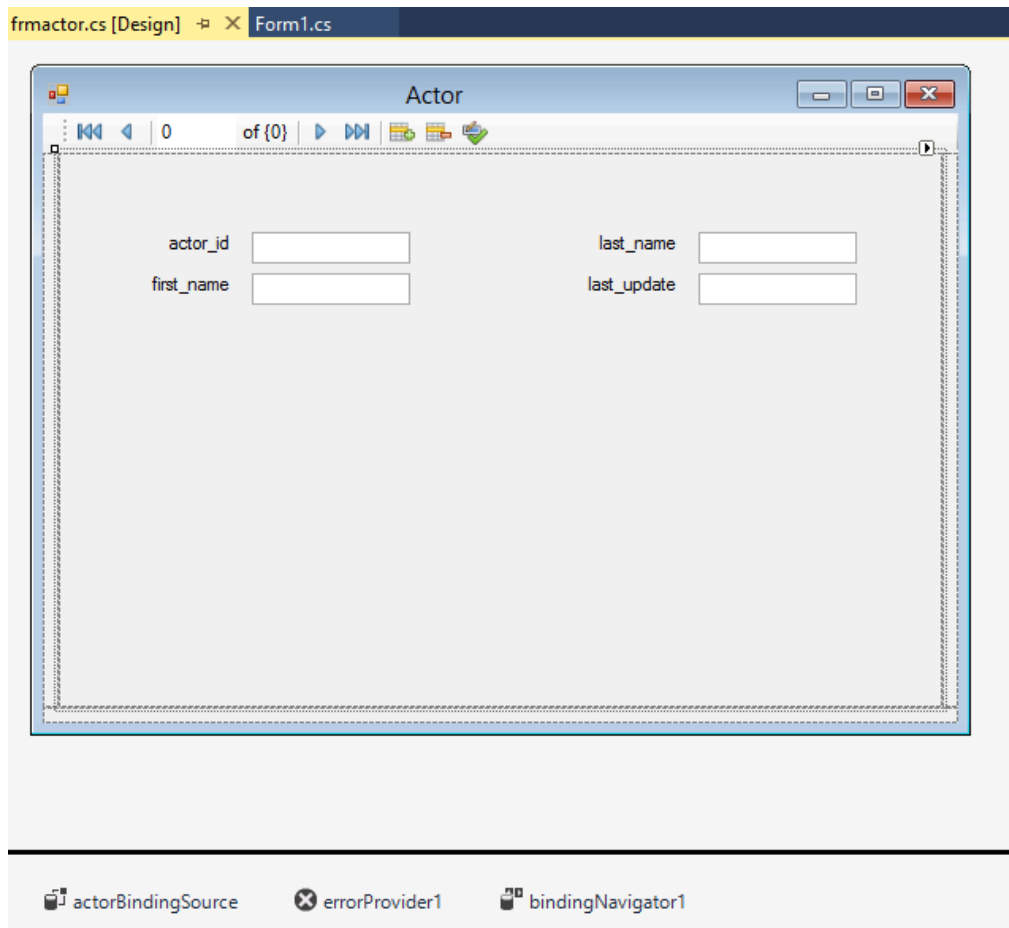
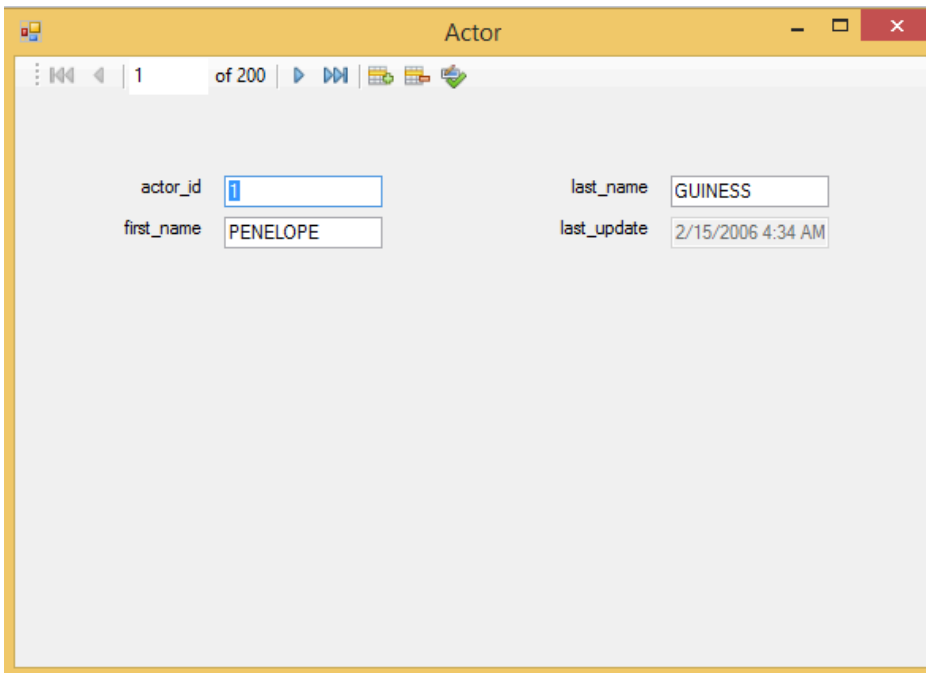


Figure 5.43 The "frmactor" Form to Display Data

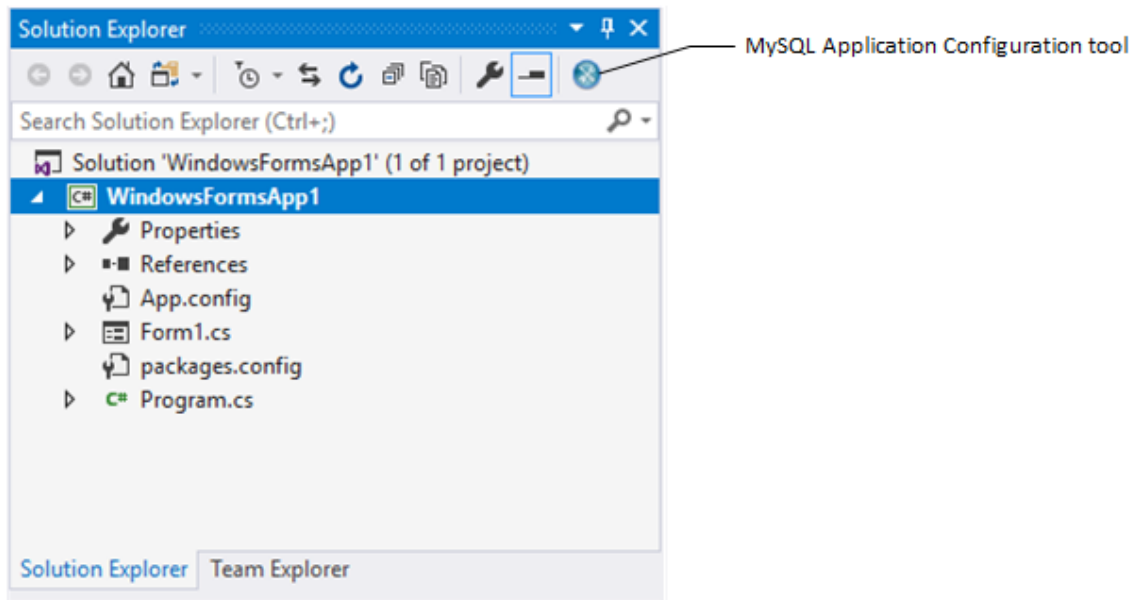
5.7 MySQL Application Configuration Tool

The MySQL Application Configuration tool (previously named MySQL Website Configuration) enables you to configure options for the following categories with MySQL as the database provider: entity framework, membership (advanced or simple), roles, profiles, session state, site map, and web personalization. With this MySQL for Visual Studio feature, you can configure multiple provider pages in sequence and the tool modifies your configuration files accordingly.

Editing configuration files manually can be problematic. The MySQL Application Configuration tool simplifies the task by providing the relevant options for each web (or application) project in a graphical, wizard-like format that permits you to navigate among the provider categories. The tool then adds, modifies, or removes entries from the [App.config](#) file, which applies to settings for non-web projects only, and the [Web.config](#) file for your web-based projects.

The MySQL Application Configuration tool appears as a small icon on the Solution Explorer toolbar in Visual Studio, as shown in the following figure. The icon is visible only when a project is active (with a connection to MySQL), and Connector/NET is installed. Clicking the MySQL Application Configuration icon launches the tool and displays the options for setting up Entity Framework support.

Figure 5.44 MySQL Application Configuration Tool



The remainder of this chapter describes each item that you can configure using the MySQL Application Configuration tool.

5.7.1 Entity Framework

The MySQL Application Configuration tool downloads the latest entity framework and MySQL Connector/NET assemblies from the NuGet gallery to keep the assembly versions synchronized. After the configuration is applied, the tool adds the following references to your project:

- EntityFramework
- EntityFramework.SqlServer
- MySql.Data
- MySql.Data.EntityFramework

Unlike the other web providers supported by this tool, the entity framework options can be applied to either non-web or web applications. The context of your project determines which configuration file the tool updates. The next sections describe the details for setting options with non-web and web projects.

Note

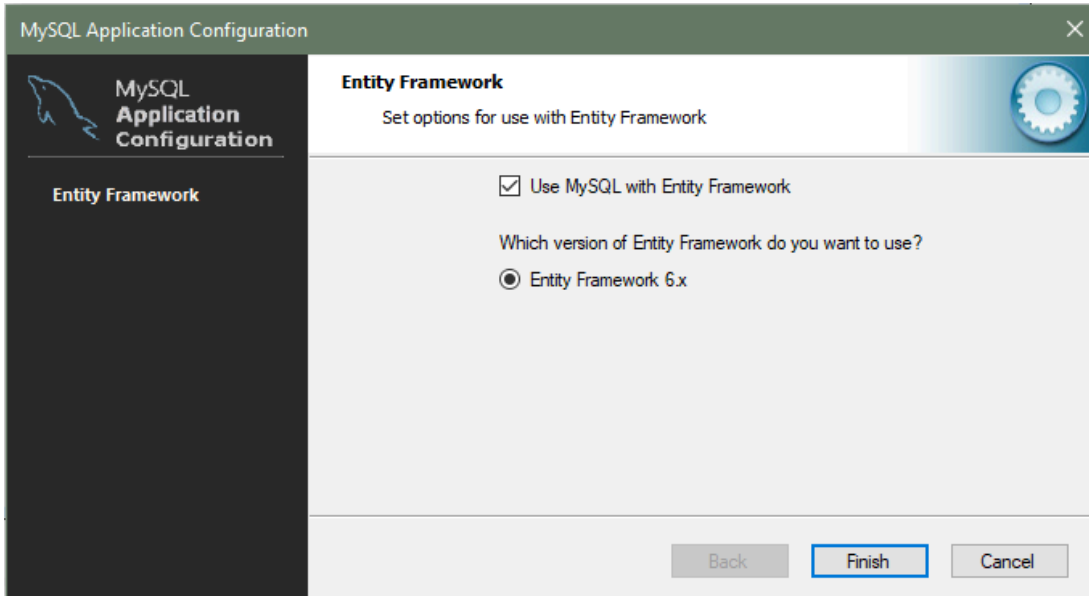
MySQL for Visual Studio does not support Entity Framework Core. For the current support profile, see [Minimum Requirements](#).

Setting Options for Non-Web Applications

The MySQL Application Configuration tool synchronizes the latest version of MySQL Connector/NET and Entity Framework 6 assemblies and then adds the required references to your application's `App.config` file. When started in the context of a non-web application, the tool lists entity framework as the only item and MySQL for Visual Studio 1.2.9 (or higher) is a prerequisite. For an example of when you might use the tool, see [Tutorial: Using an Entity Framework Entity as a Windows Forms Data Source](#).

To enable the tool, select **Use MySQL with Entity Framework**, ensure that **Entity Framework 6.x** is selected, and then click **Finish** (see the figure that follows).

Figure 5.45 MySQL Application Configuration Tool for Entity Framework (non-web)

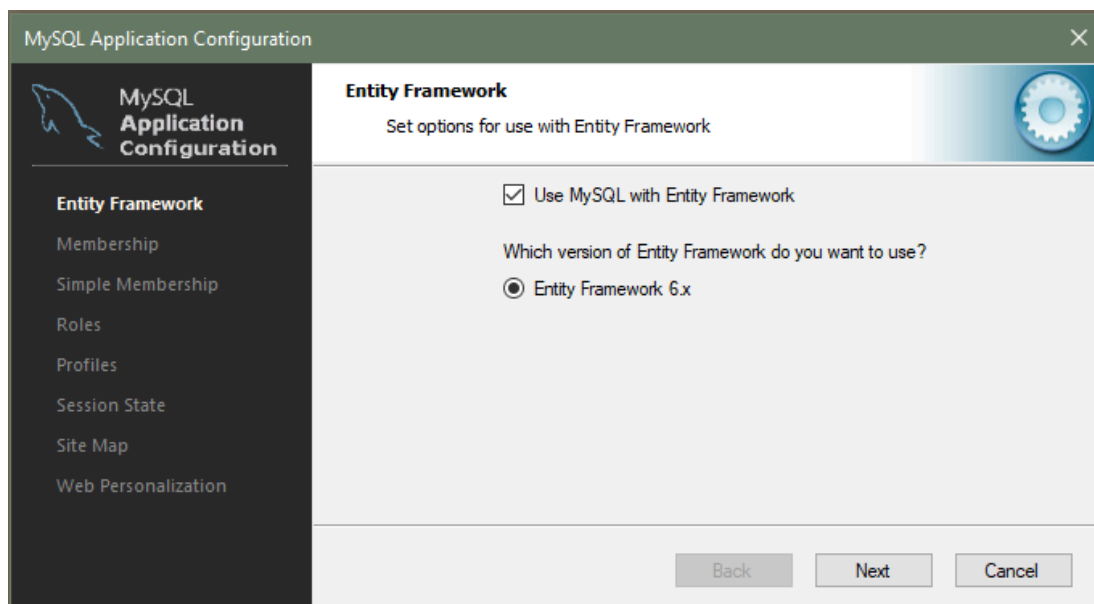


Setting Options for Web Applications

In the context of a web application, entity framework is the first page of options that you can enable. After enabling (or skipping) the entity framework option, the wizard-like tool steps through several additional web provider pages (see [Section 5.7.2, “Web Providers”](#)). You can enable entity framework alone and skip the other web providers, or selectively enable other providers in the same session.

With MySQL for Visual Studio 1.2.9 (or higher) installed, Entity Framework 6 is the only version to select, as the following figure shows. Previous versions of MySQL for Visual Studio permit you to configure your application to use Entity Framework 5 or 6. (Prior to version 1.2.9, this tool was named MySQL Website Configuration.) The current version of MySQL Connector/NET no longer supports Entity Framework 5.

To include entity framework options, select **Use MySQL with Entity Framework**, ensure that an **Entity Framework** version is selected, and then click **Next** to advance to the membership provider options. To skip configuring entity framework options, deselect the **Use MySQL with Entity Framework** check box.

Figure 5.46 MySQL Application Configuration Tool for Entity Framework (web)

For information about the options on the membership (or simple membership), roles, profiles, session state, site map, and web personalization pages, see [Section 5.7.2, “Web Providers”](#).

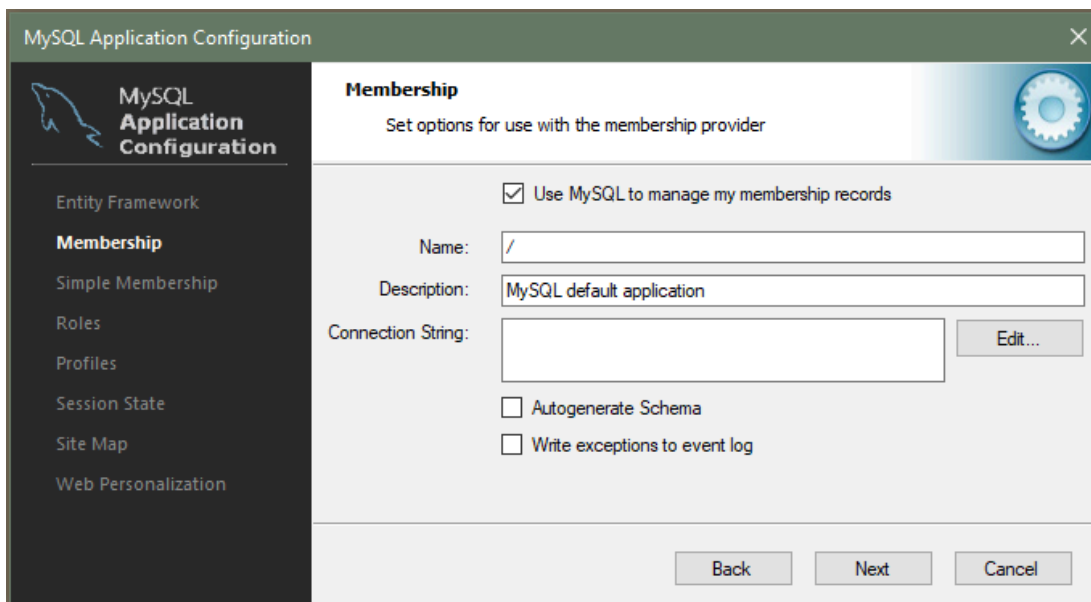
5.7.2 Web Providers

The MySQL Application Configuration tool establishes MySQL as the database provider for one or more web providers by making changes to your application's `web.config` file. You can choose only one of the two membership providers: membership or simple membership. All of the other web providers may be enabled independently or together. The configuration entries for each web provider managed by the MySQL Application Configuration tool are included in the sections that follow.

- [Membership Provider](#)
- [Simple Membership Provider](#)
- [Roles Provider](#)
- [Profiles Provider](#)
- [Session State Provider](#)
- [Site Map Provider](#)
- [Web Personalization Provider](#)

Web-provider pages share several common elements, as the membership provider page shows.

Figure 5.47 MySQL Application Configuration Tool - Membership



Check box to enable the provider. Web providers are enabled when you select the check box on the page. When selected, the page enforces requirement-checking specific to the provider. For example, if you enable any provider, the configuration file for your application must include a connection string. If the connection string is missing, the tool returns a warning when you attempt to move to the next provider. To skip (or disable) an individual web provider, deselect the check box.

Name. Each enabled web provider requires a value to specify the application name. If you do not provide a value, then the default value creates an application name for you. The value is associated with the `applicationName` property in the `Web.config` file.

Connection String. The `Web.config` file stores a single connection string for all of the MySQL web-providers. Each web-provider page includes an area for entering a connection string, however only the last entry is saved. For example, if you set it in the first web provider and also set it on the third web provider, the connection string from the third provider page is saved. You can enter a connection string directly in the text box provided or click **Edit** to use a visual editor to help you sort and select the various connection string options (see [Section 5.7.3, "Using the MySQL Connection String Editor"](#)).

Autogenerate Schema. Select the **Autogenerate Schema** option check box to ensure that the necessary schemas are created automatically for the web provider being configured. These schemas are used to store MySQL web provider information. The database used for storage is the one specified in the connection string.

Write exceptions to event log. Select the **Write exceptions to event log** option check box to ensure that exceptions generated by the application are written to the Windows event log.

Membership Provider

In addition to the standard membership provider, there is also a simple membership provider. You can only choose one of these two membership providers. To use the membership provider, select **Use MySQL to manage my membership records** to enable the page. You can now enter the name of the application that you are creating the configuration for. You can also enter a description for the application. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a membership provider, a new section is added to the web configuration file.


```
<membership defaultProvider="MySQLMembershipProvider">
  <providers>
    <remove name="MySQLMembershipProvider" />
    <add name="MySQLMembershipProvider" type="MySql.Web.Security.MySQLMembershipProvider,
      MySql.Web, Version=8.0.18.0, Culture=neutral, PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="MySQL default application" connectionStringName="LocalMySqlServer"
      writeExceptionsToEventLog="False" autogenerateschema="False" enableExpireCallback="False"
      enablePasswordRetrieval="False" enablePasswordReset="True" requiresQuestionAndAnswer="True"
      requiresUniqueEmail="False" passwordFormat="Clear" maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="7" minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
      passwordStrengthRegularExpression="/" />
  </providers>
</membership>
```

With one of the membership providers configured, click **Next** to advance to the roles provider page.

Simple Membership Provider

The simple membership provider options are similar to those of the membership provider, but with fewer properties in the configuration file. To enable, check the **Use MySQL to manage my simple membership records**.

Note

The simple membership provider is not supported by MySQL Connector/NET 8.0 and cannot be enabled if you have the 8.0 version of the connector installed.

The MySQL simple membership provider handles the website membership tasks with ASP.NET. This provider is a simpler version of the ASP.NET Membership provider, and it can also work with OAuth Authentication. For additional information about using OAuth authentication, see [Adding OAuth Authentication to a Project](#).

The required configuration options for the simple membership provider are: a name for the connection string and a connection string that contains a valid database with a local or remote MySQL server instance, a user table to store the credentials, and column names for the `User ID` and `User Name` columns.

Select the **Auto Create Tables** option to create the required tables when adding the first user to the table. After setting up a membership provider, a new section is added to the web configuration file.

```
<membership defaultProvider="MySQLSimpleMembershipProvider">
  <providers>
    <clear />
    <remove name="MySQLSimpleMembershipProvider" />
    <add name="MySQLSimpleMembershipProvider"
      type="MySql.Web.Security.MySQLSimpleMembershipProvider, MySql.Web, Version=6.10.8.0, Culture=neutral, PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="MySQL default application"
      connectionStringName="LocalMySqlServer"
      writeExceptionsToEventLog="False"
      autogenerateschema="False"
      enableExpireCallback="False"
      userTableName="Users"
      userIdColumn="UserId" userNameColumn="UserName" autoGenerateTables="True" />
  </providers>
</membership>
```

After setting up one of the membership providers, click **Next** to configure the roles provider page.

Roles Provider

Click **Use MySQL to manage my roles** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a roles provider, a new section is added to the web configuration file.

```
<roleManager defaultProvider="MySQLRoleProvider">
  <providers>
    <remove name="MySQLRoleProvider" />
    <add name="MySQLRoleProvider" type="MySql.Web.Security.MySQLRoleProvider, MySql.Web, Version=8.0.18.0,
      PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="" connectionStringName="LocalMySqlServer"
      writeExceptionsToEventLog="False" autogenerateschema="False" enableExpireCallback="False" />
  </providers>
</roleManager>
```

Click **Next** to configure the profiles provider page.

Profiles Provider

Click **Use MySQL to manage my profiles** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema**, **Write exceptions to event log**, and **Callback for session end event** options can be selected for this web provider.

After setting up a profiles provider, a new section is added to the web configuration file.

```
<profile defaultProvider="MySQLProfileProvider">
  <providers>
    <remove name="MySQLProfileProvider" />
    <add name="MySQLProfileProvider" type="MySql.Web.Profile.MySQLProfileProvider, MySql.Web, Version=8.0.
      Culture=neutral, PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="" connectionStringName="LocalMySqlServer"
      writeExceptionsToEventLog="False" autogenerateschema="False" enableExpireCallback="False" />
  </providers>
</profile>
```

Click **Next** to configure the session state provider page.

Session State Provider

Click **Use MySQL to manage my ASP.Net session state** to enable this provider page. The page includes the following options to edit: the connection string, the application name, and a description of the provider. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up a session provider, a new section is added to the web configuration file.

```
<sessionState mode="Custom" cookieless="true" regenerateExpiredSessionId="true" customProvider="MySqlSessionSt
  <providers>
    <add name="MySqlSessionStateProvider" type="MySql.Web.SessionState.MySqlSessionStateStore, MySql.Web,
      Version=8.0.18.0, Culture=neutral, PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="" connectionStringName="LocalMySqlServer"
      writeExceptionsToEventLog="False" autogenerateschema="False" enableExpireCallback="False" />
  </providers>
</sessionState>
```

Click **Next** to configure the site map provider page.

Site Map Provider

The site map provider builds a site map from a MySQL database, and builds a complete tree of the [SitemapNode](#) objects. It also provides methods so that the generated nodes can be read from the site map. Click **Use MySQL to manage my ASP.NET site map** to enable this provider page.

The required configuration options: A name for the application and a connection string that contains a valid database with a local or remote MySQL server instance. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up the site map provider, a new section is added to the web configuration file.

```
<siteMap defaultProvider="MySQLSiteMapProvider">
  <providers>
    <remove name="MySQLSiteMapProvider" />
    <add name="MySQLSiteMapProvider" type="MySQL.Web.SiteMap.MySQLSiteMapProvider, MySQL.Web, Version=
      Culture=neutral, PublicKeyToken=c2222fc2222c44d"
      applicationName="/" description="" connectionStringName="LocalMySQLServer" writeExceptionsToEvent
      autogenerateschema="False" enableExpireCallback="False" />
    </providers>
  </siteMap>
```

Click **Next** to proceed to the web personalization configuration page:

Web Personalization Provider

The web personalization provider is used when a website application needs to store persistent information for the content and layout of the Web Parts pages that are generated by a Web Parts personalization service. This provider should be used along with the membership, roles, and profiles providers. Click **Use MySQL to manage my ASP.NET personalization provider** to enable this provider page.

The required configuration options: A name for the application and a connection string that contains a valid database with a local or remote MySQL server instance. The **Autogenerate Schema** and **Write exceptions to event log** options can be selected for this web provider.

After setting up the web personalization provider, a new section is added to the web configuration file.

```
<webParts>
  <personalization defaultProvider="MySQLPersonalizationProvider">
    <providers>
      <remove name="MySQLPersonalizationProvider" />
      <add name="MySQLPersonalizationProvider" type="MySQL.Web.Personalization.MySQLPersonalizationPro
        MySQL.Web, Version=8.0.18.0, Culture=neutral, PublicKeyToken=c2222fc2222c44d"
        applicationName="/" description="" connectionStringName="LocalMySQLServer" writeExceptionsToEvent
        autogenerateschema="False" enableExpireCallback="False" />
      </providers>
    </personalization>
  </webParts>
```

When you have selected the web personalization options, click **Finish** to write the changes for all web providers to the [Web.config](#) file and close the tool.

5.7.3 Using the MySQL Connection String Editor

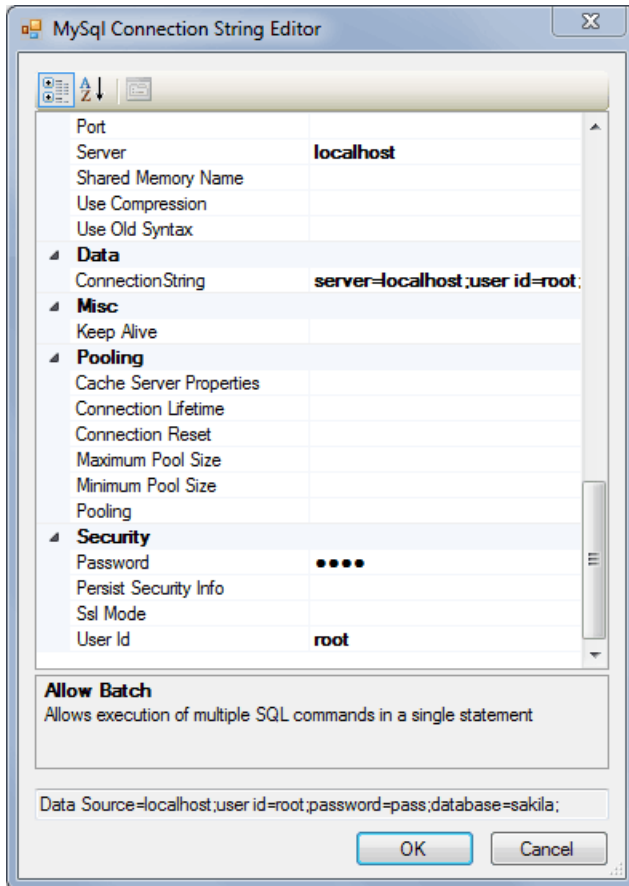
The MySQL Connection String Editor lists and describes all supported connection-string options for the version of MySQL Connector/NET that you have installed. To open the editor from any web-provider page (see the **Connection String** entry box), click **Edit**.

By default, the editor lists connection-string options by category (see the figure that follows). Alternatively, you can sort the list alphabetically using the sorting buttons provided. The main list consists of options on the left and values on the right of a split area. To add a text value, such as the database name, select the option in the list (*Database*, in this example), and then type the name of the database (*sakila*) after the cursor. Some connection-string options have predefined values and show a down arrow when selected.

When you select a connection-string option, a brief description is shown in an information box below the list. After you add an option value and then select another option, the editor appends the new option/value

pair to the existing connection string. Click **OK** to return to the web-provider page, which now includes the modified connection string.

Figure 5.48 MySQL Application Configuration Tool - Connection String Editor



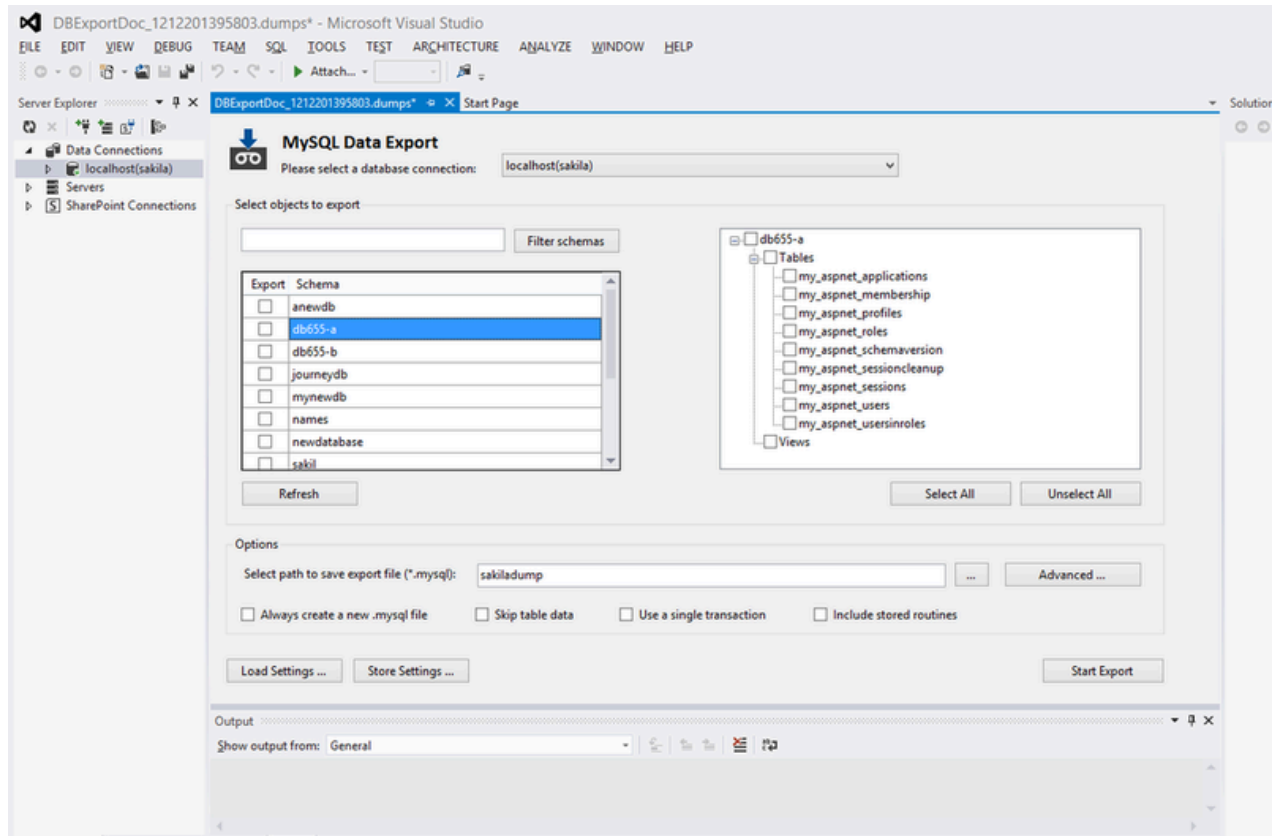
Note

Defined connection strings are automatically loaded and available in editor, whether they were created manually in `web.config` for each application or previously using this tool.

5.8 MySQL Data Export Tool

MySQL for Visual Studio has a data export tool that creates a dump file for a MySQL database.

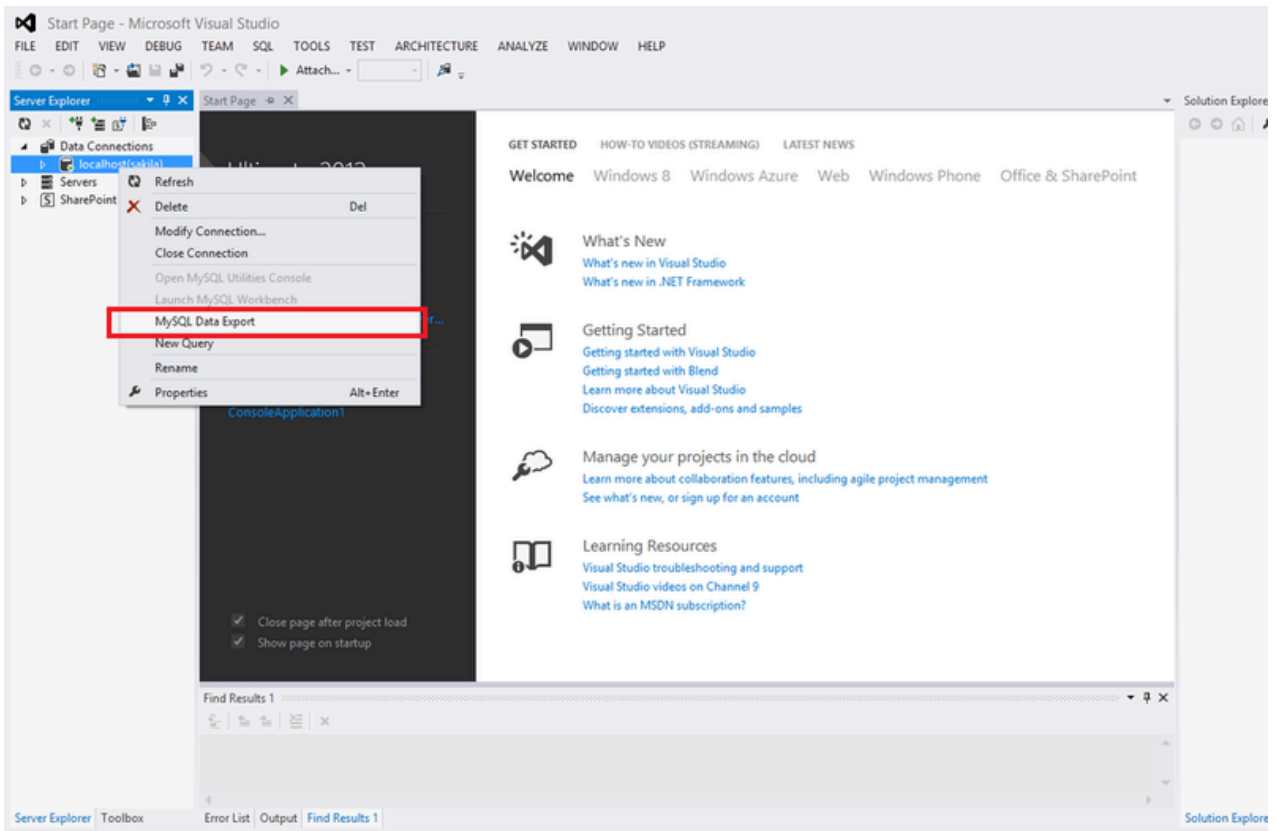
Figure 5.49 MySQL for Visual Studio Data Export Tool: Main Window



Creating a Dump of an existing MySQL Database

To open a new window for the MySQL Data Export tool, you must first create a new connection using the **Server Explorer** window inside Visual Studio. After the connection is established, a contextual menu can be opened by right clicking on the connection node. From the menu, choose the **MySQL Data Export** option. A new tab opens for the current connection. You can then select one or more databases to include in the dump.

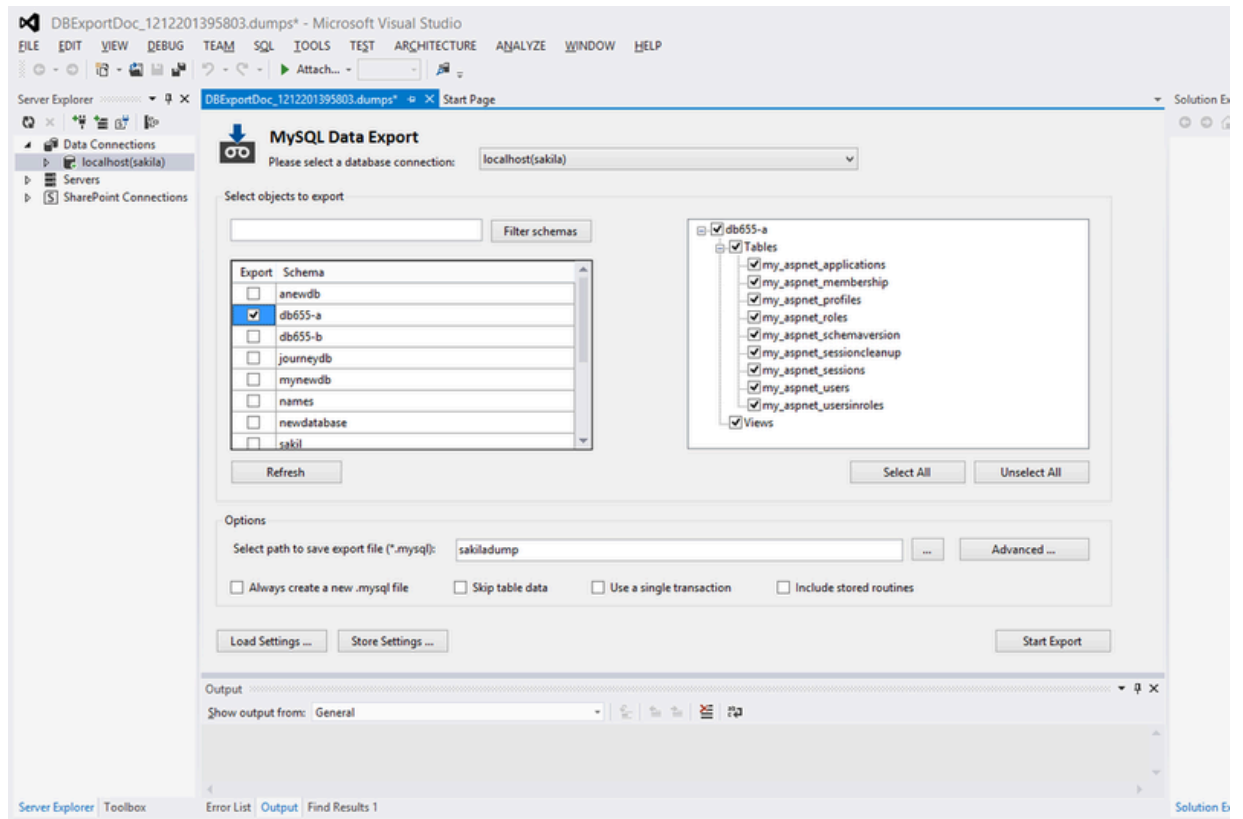
Figure 5.50 MySQL for Visual Studio Data Export Tool: Connection Context Menu



Follow these steps to create a dump for the MySQL Databases:

1. Select all the databases and their objects to be included in the dump.

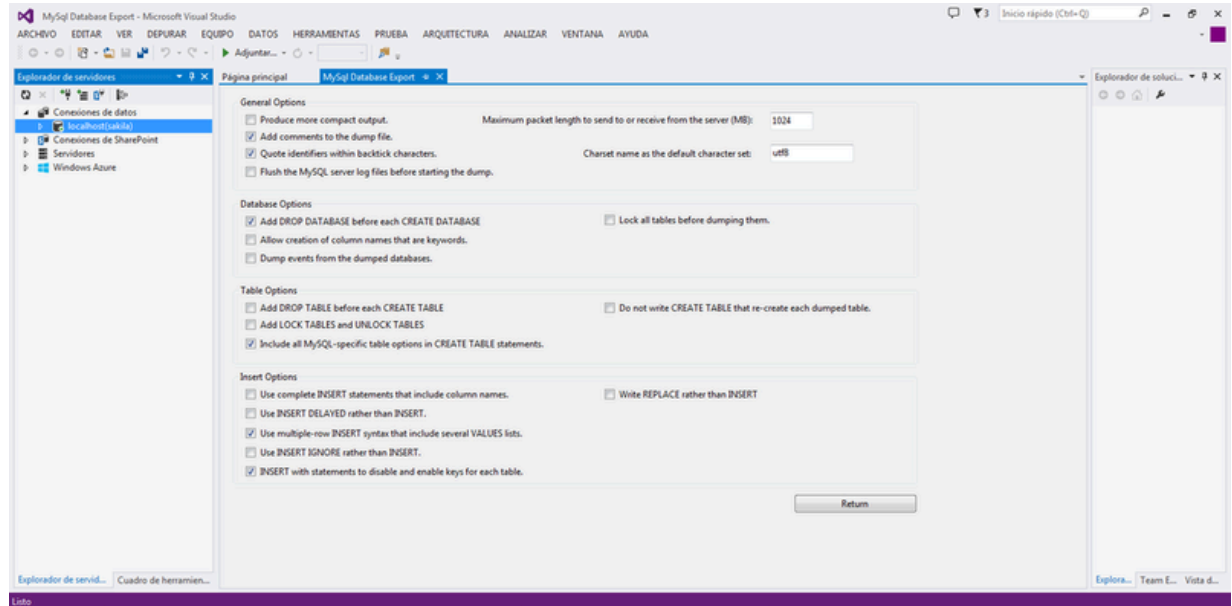
Figure 5.51 MySQL for Visual Studio Data Export Tool: Selecting a Database



- It is very important to select the desired settings for the dump: whether the dump will include the data, whether the insert operations will be logged in extended mode, and so on. In the main window of the

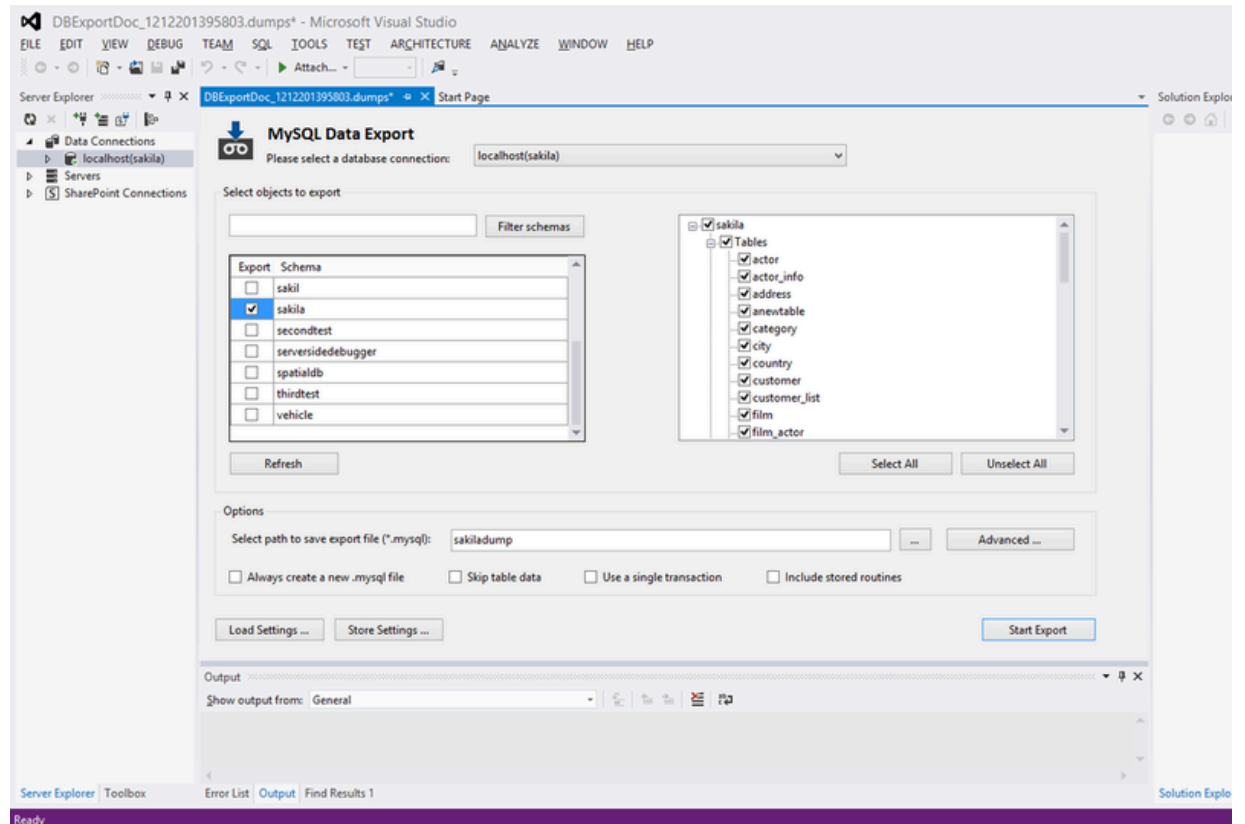
MySQL Database Export tool are shown the basic options for the dump. Also, by clicking on the **Advanced** button, more specific options can also be selected.

Figure 5.52 MySQL for Visual Studio Data Export Tool: Advanced Options



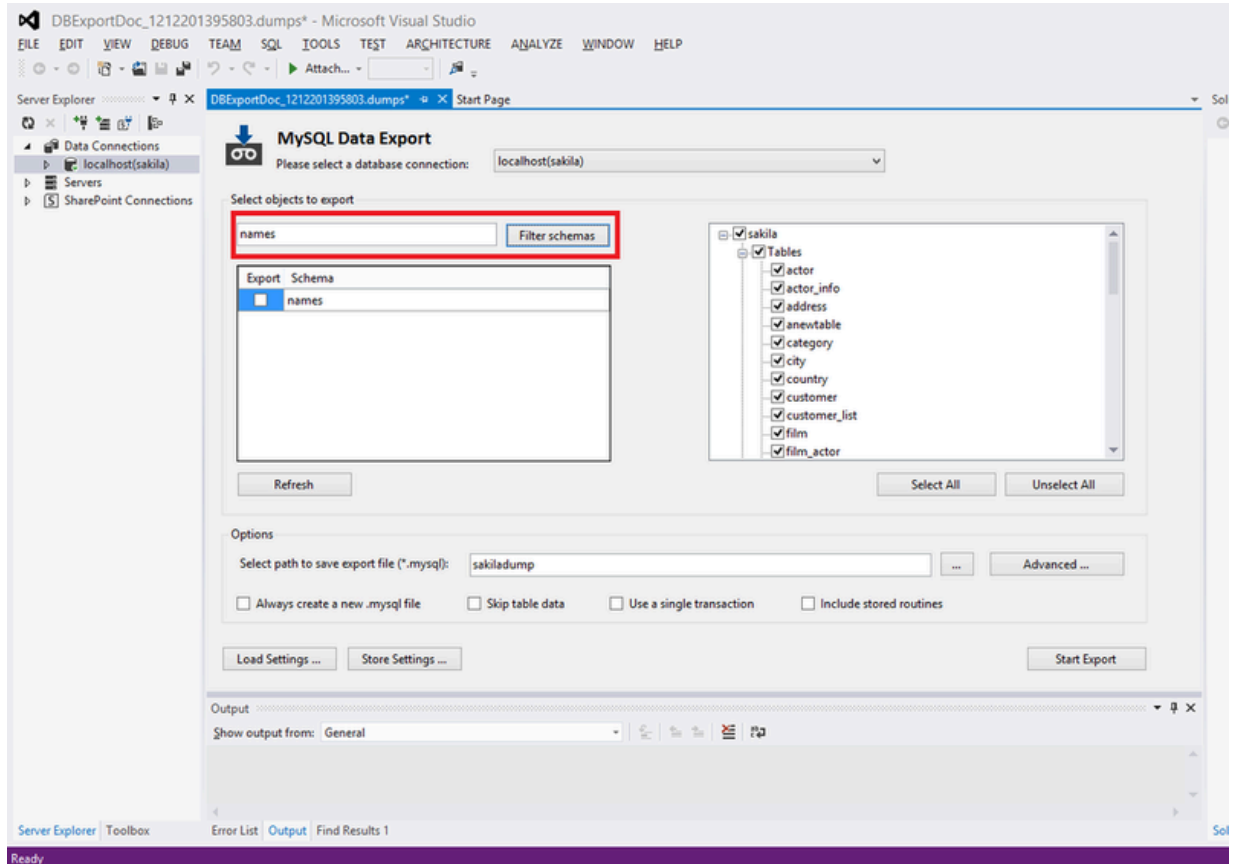
- When the selection of the options is done, give a name to the result file that will be created. If no path is given for the result file, the default path to be used is *My Documents* under the user's folder.

Figure 5.53 MySQL for Visual Studio Data Export Tool: Generating the Dump File



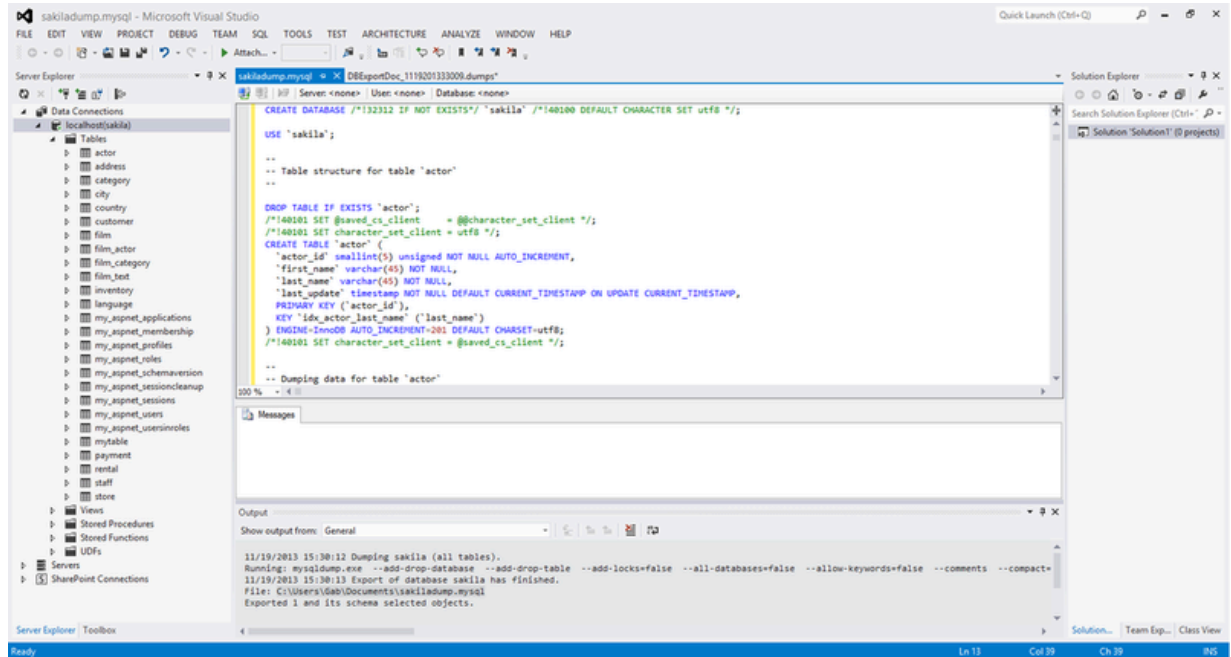
4. A filter can be applied on the list of schemas for the selected connection. With it, the user can easily locate the databases to be included in the dump.

Figure 5.54 MySQL for Visual Studio Data Export Tool: Filtering the Schemas



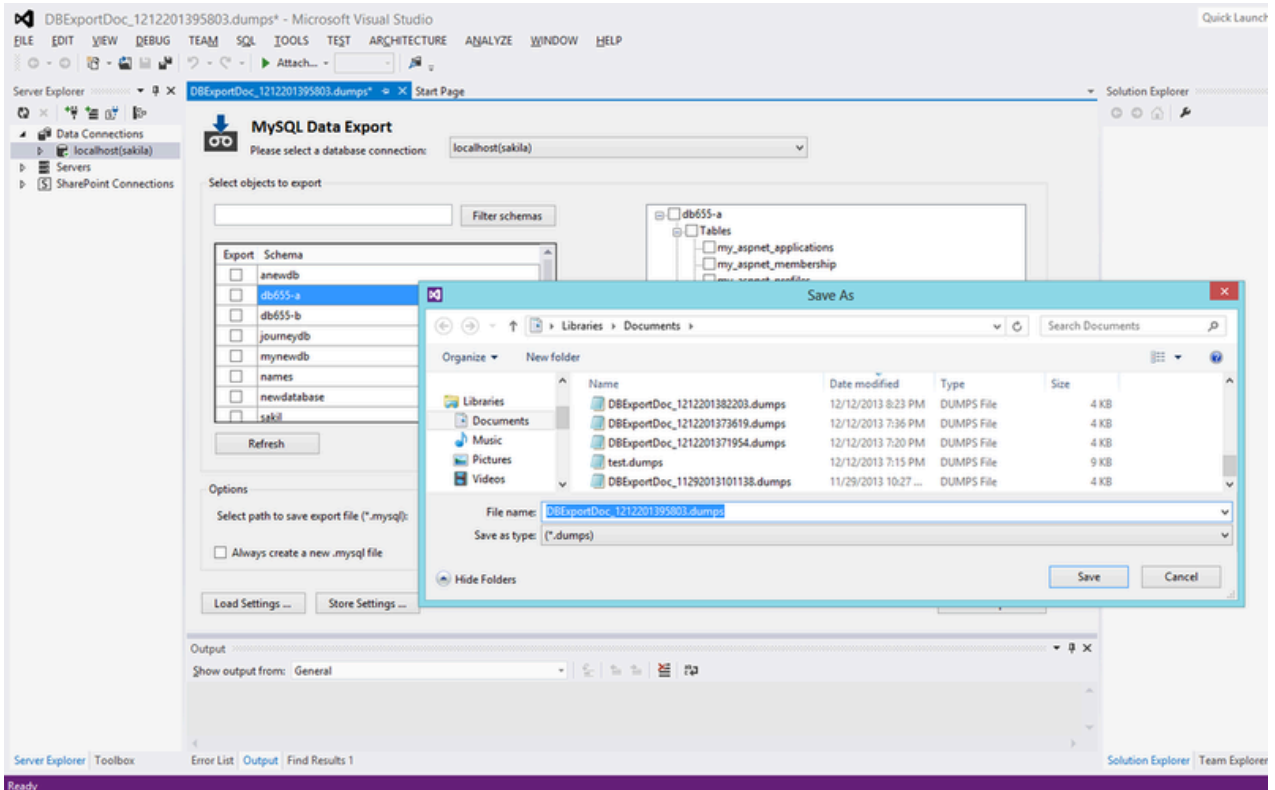
5. After selecting the options and the name for the dump file, the user can click the **Export** button, which generates the dump.

Figure 5.55 MySQL for Visual Studio Data Export Tool: Viewing the Generated Script



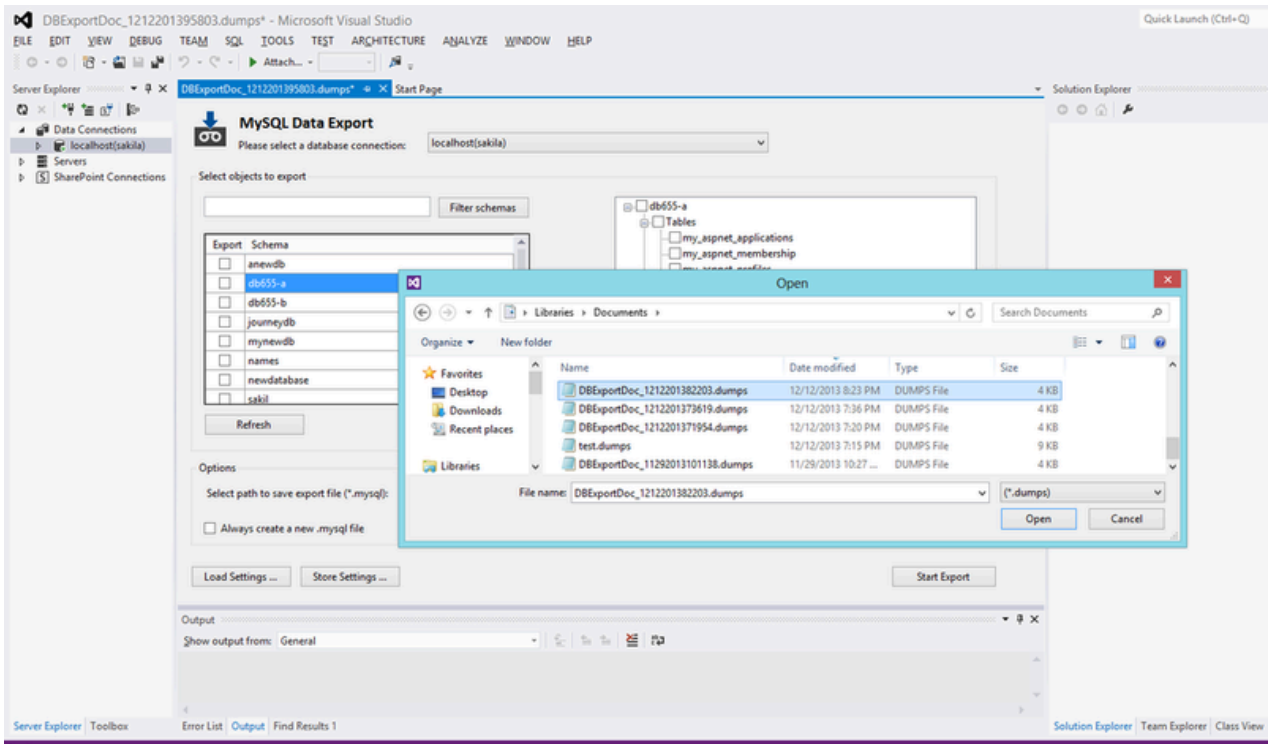
Each dump can have different settings. After configuring the dump operation, the settings can be saved into a setting file for later use. This file includes: the connection selected, the name of the file for the dump, and the database or databases and the objects selected for dumping. The file extension for the setting file is `.dumps`.

Figure 5.56 MySQL for Visual Studio Data Export Tool: Saving a Setting File



A saved setting file can be loaded into the **MySQL Data Export** tool by clicking the **Load Settings** button.

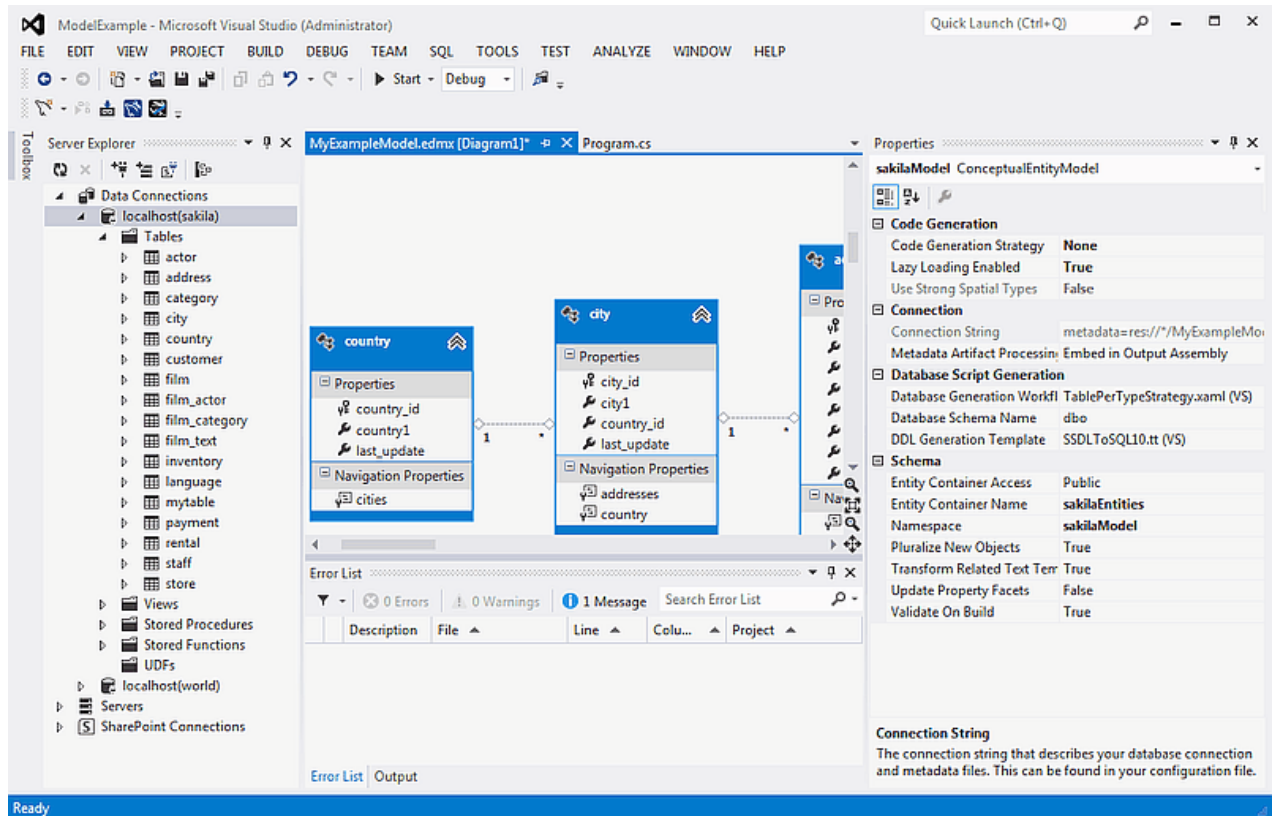
Figure 5.57 MySQL for Visual Studio Data Export Tool: Opening a Setting File



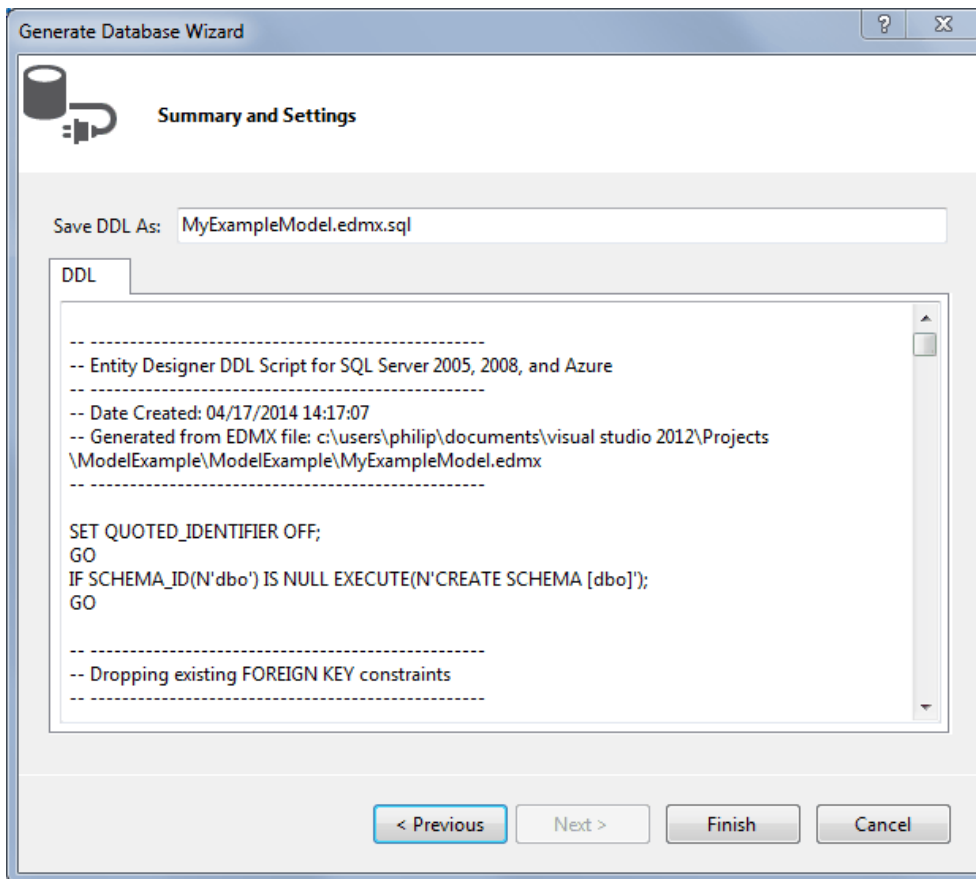
5.9 DDL T4 Template Macro

Convert an Entity Framework model to MySQL DDL code. Starting with a blank model, you can develop an entity model in Visual Studio's designer. Once the model is created, you can select the model's properties, and in the Database Script Generation category of the model's properties, the property **DDL Generation** can be found. Select the value **SSDLToMySQL.tt(VS)** from the drop-down list.

Figure 5.58 DDL T4 Template Macro - Model Properties



Right-clicking the model design area displays a context-sensitive menu. Selecting **Generate Database from Model** from the menu displays the **Generate Database Wizard**. The wizard can then be used to generate MySQL DDL code.

Figure 5.59 DDL T4 Template Macro - Generate Database Wizard

5.10 Debugging Stored Procedures and Functions

The stored procedure debugger provides facilities for setting breakpoints, stepping into individual statements (Step Into, Step Out, Step Over), evaluating and changing local variable values, evaluating breakpoints, and other debugging tasks.

Privileges

The debugger recreates at the start of each debug session a [serversidedebugger](#) database in your server. This database helps to track the instrumented code and implement observability logic in the debugged routine. Your current connection needs to have privileges to create that database, and its associated stored routines, functions, and tables.

The debugger makes changes behind the scenes to temporarily add instrumentation code to the stored routines that you debug. You must have the [ALTER ROUTINE](#) privilege for each stored procedure, function, or trigger that you debug. (Including procedures and functions that are called, and triggers that are fired, by a procedure that you are debugging.)

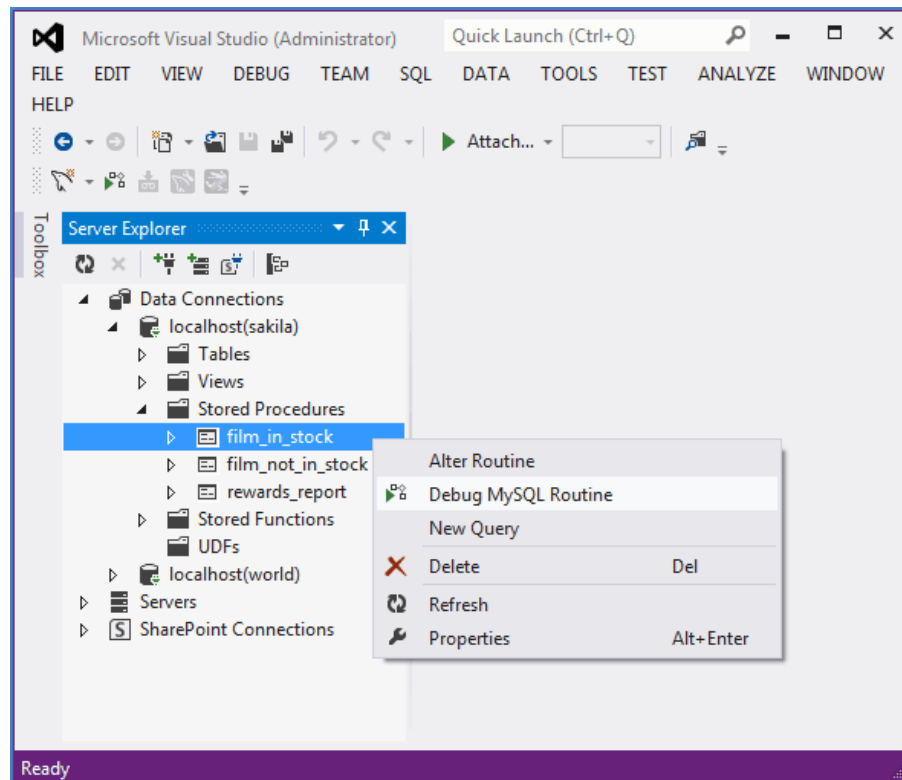
Starting the Debugger

To start the debugger, follow these steps:

1. Choose a connection in the Visual Studio Server Explorer.

2. Expand the **Stored Procedures** folder. Only stored procedures can be debugged directly. To debug a user-defined function, create a stored procedure that calls the function.
3. Click on a stored procedure node, then right-click and from the context menu choose **Debug Routine**.

Figure 5.60 Choose a Stored Routine to Debug



Usage

At this point, Visual Studio switches to debug mode, opening the source code of the routine being debugged in step mode, positioned on the first statement.

If the initial routine you debug has one or more arguments, a pop-up will show up with a grid (a row per each argument and three columns: one for the argument, one for the argument value (this is editable) and one for nullifying that argument value (a check box)). After setting up all the argument values, you can press **OK** to start the debug session, or **Cancel** to cancel the debug session.

Figure 5.61 Setting Arguments (1 of 2)

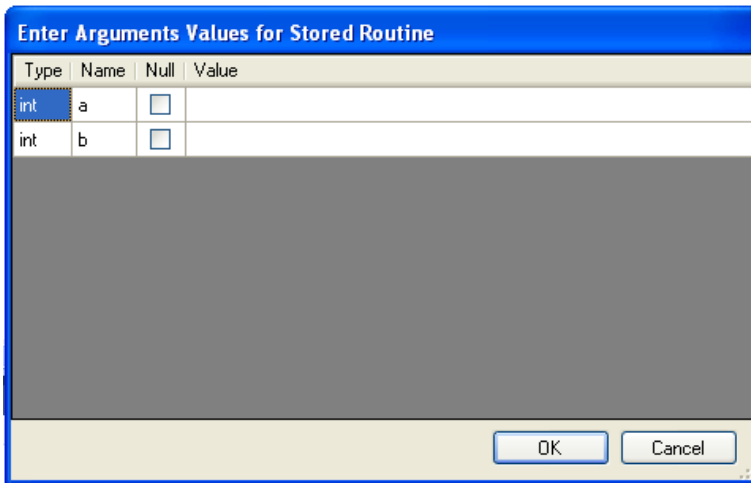
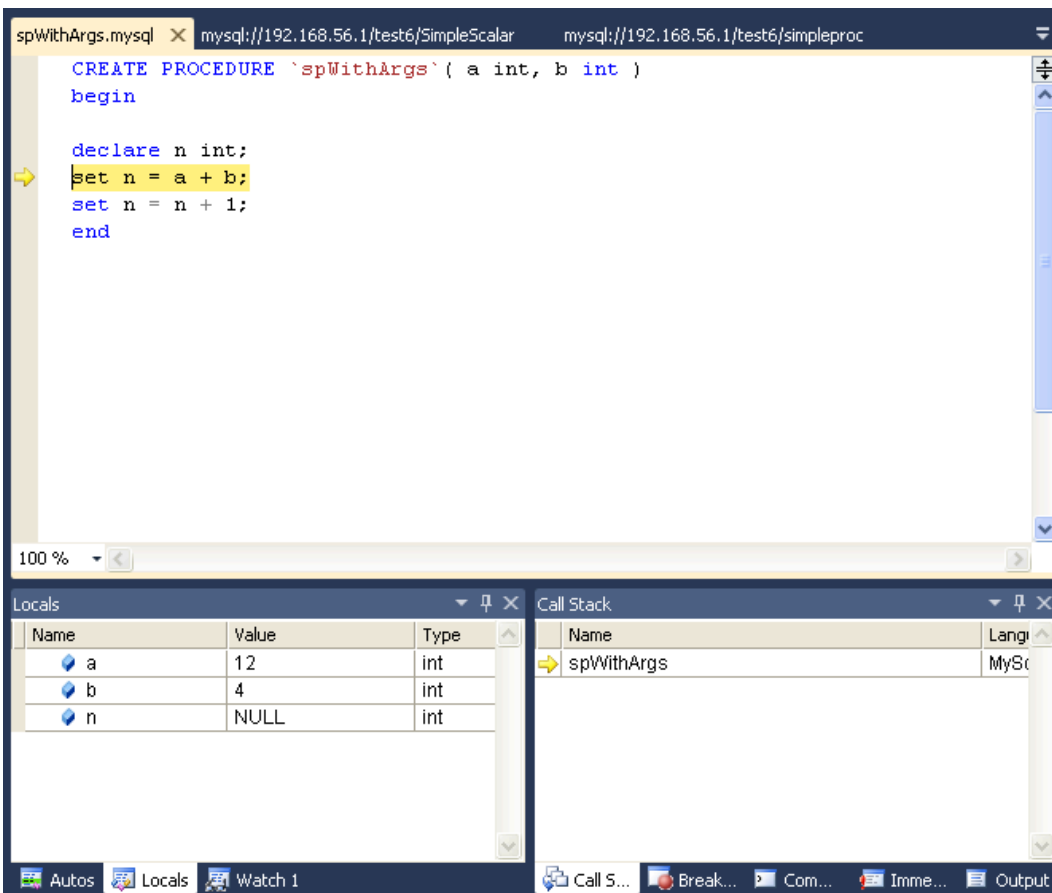


Figure 5.62 Setting Arguments (2 of 2)



How the Debugger Functions

To have visibility into the internal workings of a stored routine, the debugger prepares a special version of the procedure, function, or trigger being debugged, instrumented with extra code to keep track of the

current line being stepped into and the values of all the local variables. Any other stored procedures, functions, or triggers called from the routine being debugged are instrumented the same way. The debug versions of the routines are prepared for you automatically, and when the debug session ends (by either pressing **F5** or **Shift + F5**), the original versions of the routines are automatically restored.

A copy of the original version of each instrumented routine (the version without instrumentation) is stored in the `AppData\Roaming\MySqlDebuggerCache` folder for the current Windows user (the path returned by calling `System.Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)` in .NET, plus appending `MySqlDebuggerCache`). There is one file for each instrumented routine, named `routine_name.mysql`. For example, in Windows 7, for a user named `fergs`, the path is `C:\Users\fergs\AppData\Roaming\MySqlDebuggerCache`.

Two threads are used, one for the debugger and one for the routine being debugged. The threads run in strict alternation, switching between the debugger and the routine as each statement is executed in the stored routine.

Basic Debugging Operations

The debugger has the same look and feel as the standard Visual Studio debuggers for C#, VB.NET or C+. In particular, the following are true:

Locals and Watches

- To show the **Locals** tab, choose the menu item **Debug, Windows, Locals**.

The **Locals** tab lists all the variables available in the current scope: variables defined with `DECLARE` at any point in the routine, argument parameters, and session variables that are referenced.

- If the last step operation changes the value of a local, its value will be highlighted in red (until another statement is executed or stepped).
- You can change the value of any local.
- To show the **Watch** tab, choose the menu item **Debug, Windows, Watch**.

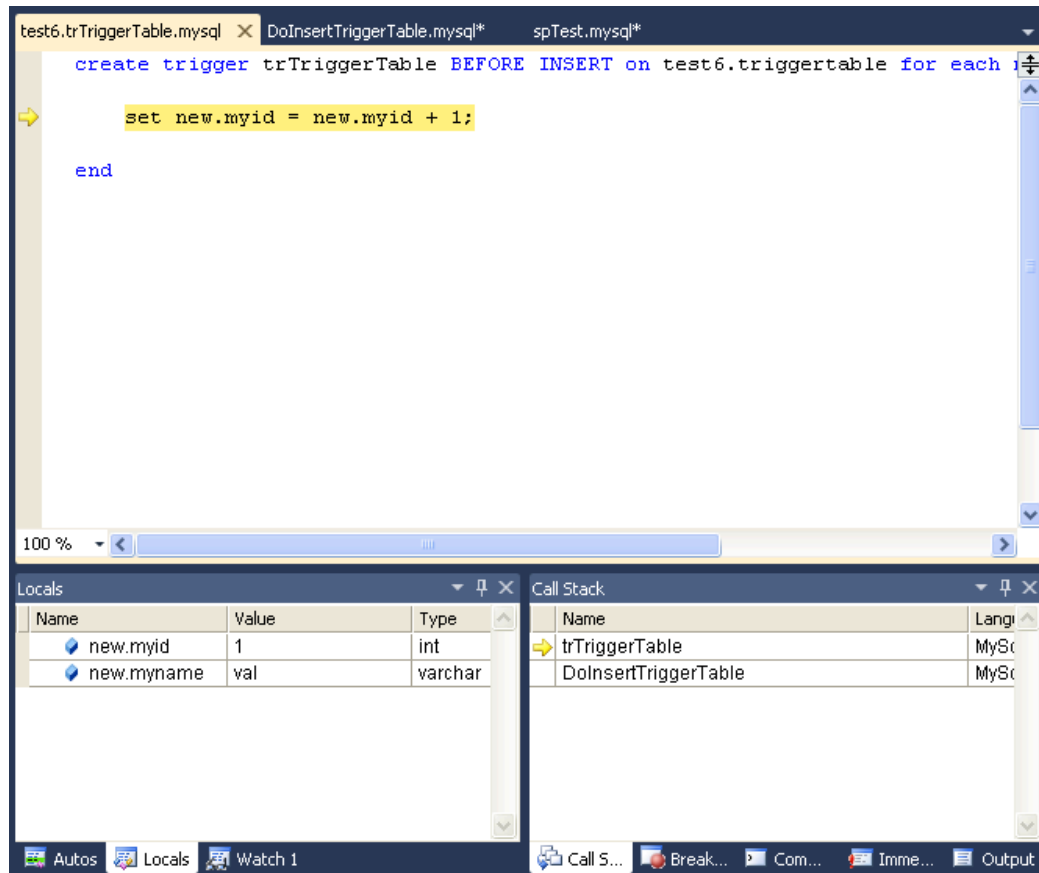
To define a watch, type any valid MySQL expression, optionally including function calls. If the watch evaluation makes sense in the current context (current stack frame), it will show its value, otherwise it will show an error message in the same row the watch was defined.

- When debugging a trigger, in addition to any locals declared or session variables referenced, the new and old object (when applicable) will be listed. For example in a trigger for `INSERT`, for a table defined like:

```
create table t1( id int, myname varchar( 50 ));
```

the locals will list the extra variables `new.id` and `new.myname`. For an `UPDATE` trigger, you will also get the extra variables `old.id` and `old.myname`. These variables from the new and old objects can be manipulated the same way as any ordinary local variable.

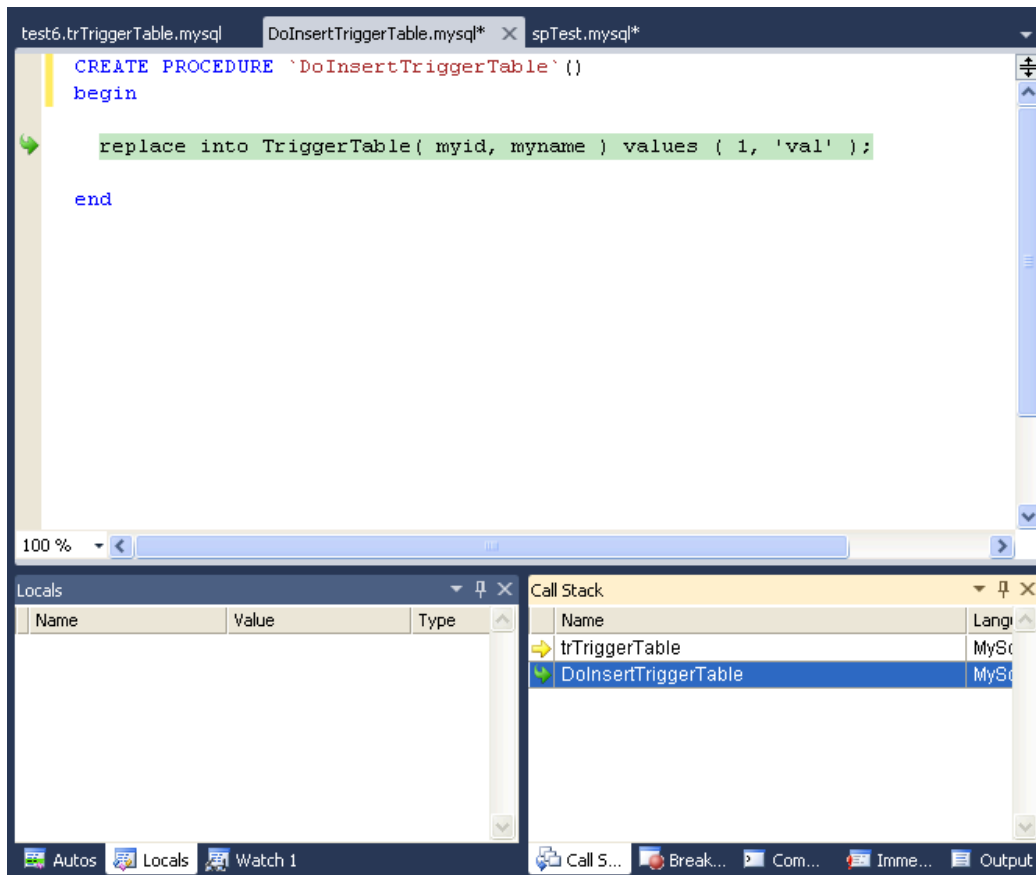
Figure 5.63 Debugging a Trigger



Call Stack

- To show the **Call Stack** tab, choose the menu item **Debug, Windows, Call Stack**.
- The stack trace (in the **Call Stack** tab) will list all the stack traces, one for each routine invocation. The one with a yellow mark is the current stepping point. Clicking in another will activate in the editor the tab for that routine source, highlighting in green the last statement stepped.

Figure 5.64 Call Stack



Stepping

- Stepping of a new routine starts in the first executable instruction (excluding declares, handlers, cursor declarations, and so on).

Figure 5.65 Debug Stepping

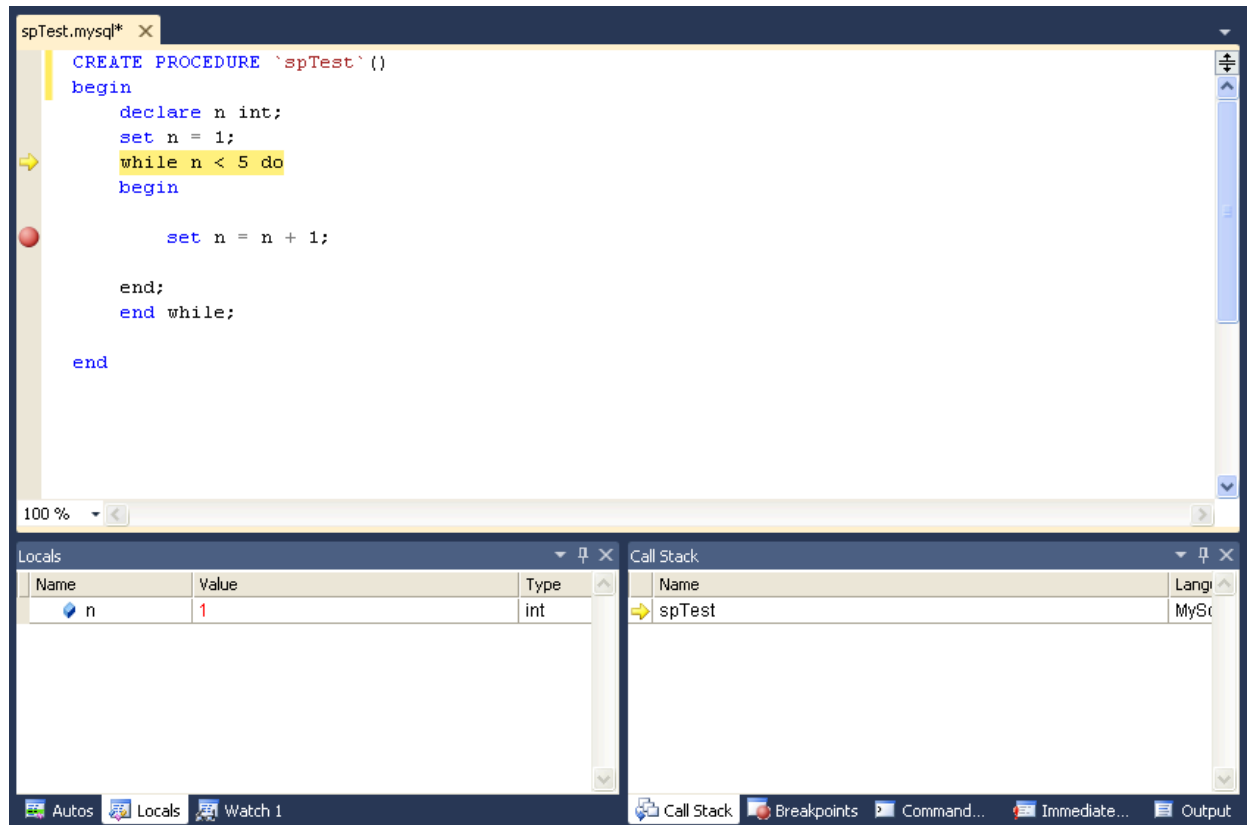


Figure 5.66 Function Stepping (1 of 2)

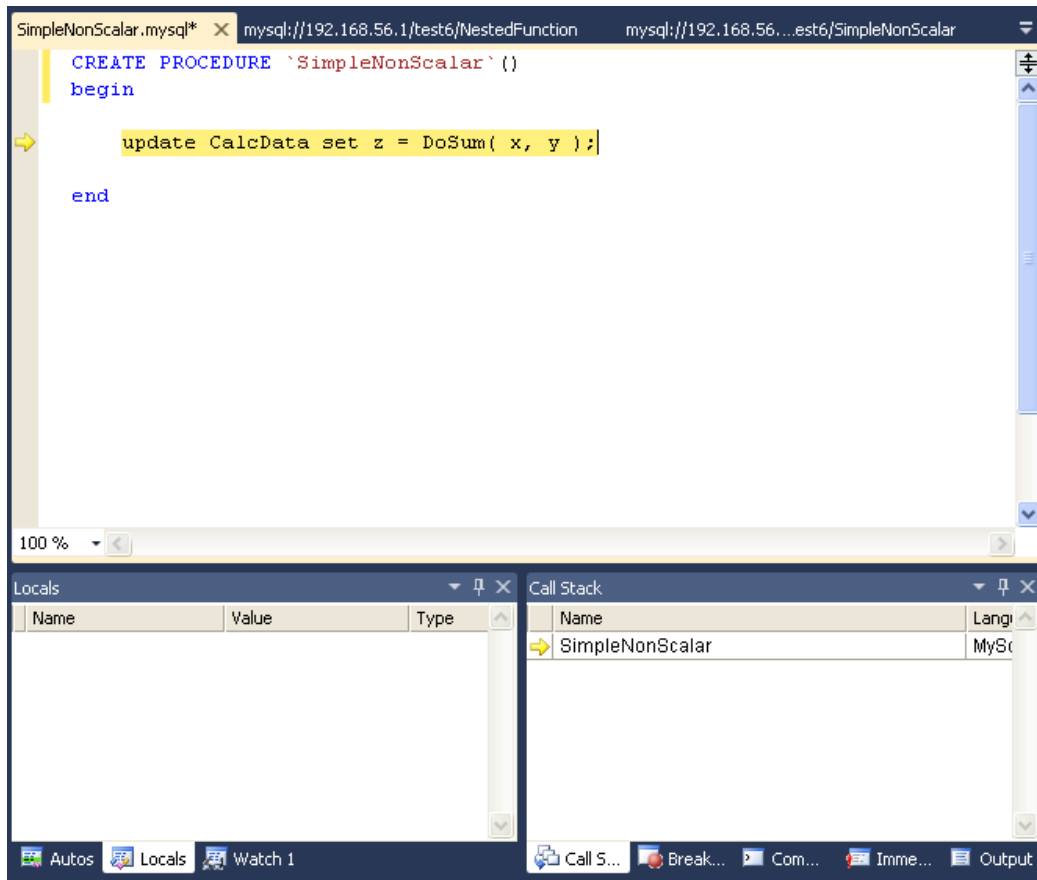
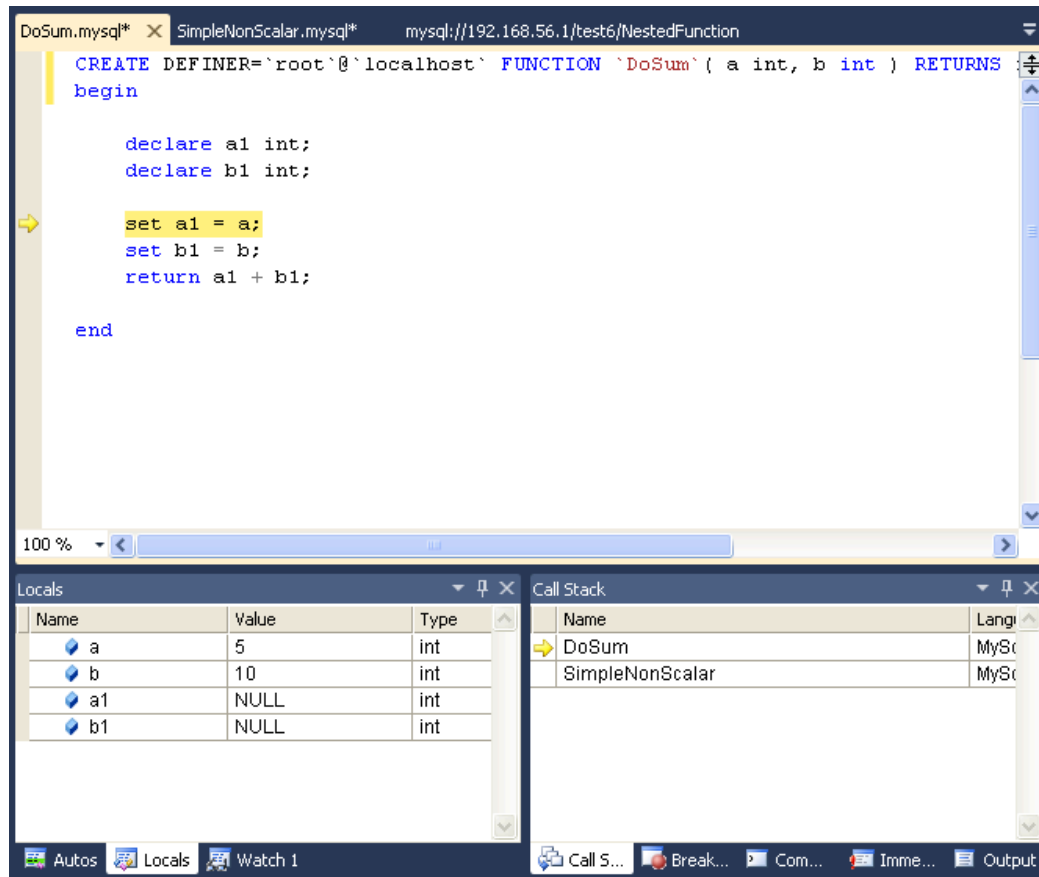


Figure 5.67 Function Stepping (2 of 2)



- To step into the code of a condition handler, the condition must be triggered in the rest of the MySQL routine.
- The next statement to be executed is highlighted in yellow.
- To continue stepping, you can choose between **Step Into** (by pressing **F11**), **Step Out** (by pressing **F10**) or **Step Over** (by pressing **Shift + F11**).
- You can step out of any of functions, triggers or stored procedures. If you step from the main routine, it will run that routine to completion and finish the debug session.
- You can step over stored procedure calls, stored functions, and triggers. (To step over a trigger, step over the statement that would cause the trigger to fire.)
- When stepping into a single statement, the debugger will step into each individual function invoked by that statement and each trigger fired by that statement. The order in which they are debugged is the same order in which the MySQL server executes them.
- You can step into triggers triggered from **INSERT**, **DELETE**, **UPDATE**, and **REPLACE** statements.
- Also, the number of times you enter into a stored function or trigger depends on how many rows are evaluated by the function or affected by the trigger. For example, if you press **F11 (Step Into)** into an **UPDATE** statement that modifies three rows (calling a function for a column in the **SET** clause, thus invoking the function for each of the three rows), you will step into that function three times in succession, once for each of the rows. You can accelerate this debug session by disabling any

breakpoints defined in the given stored function and pressing **Shift + F11** to step out. In this example, the order in which the different instances of the stored function are debugged is server-specific: the same order used by the current MySQL server instance to evaluate the three function invocations.

Breakpoints

- To show the **Breakpoints** tab, choose the menu item **Debug, Windows, Breakpoints**.
- The **Breakpoints** tab will show all the breakpoints defined. From here, you can enable and disable breakpoints one by one or all at once (using the toolbar on top of the **Breakpoints** tab).
- You can define new breakpoints only in the middle of a debug session. Click in the left gray border of any MySQL editor, or click anywhere in a MySQL editor and press **F9**. In the familiar Visual Studio way, you press **F9** once to create a breakpoint in that line, and press it again to remove that breakpoint.
- Once a breakpoint is defined, it will appear enabled (as filled red circle left to the current row if that line is a valid statement to put a breakpoint) or disabled (as a non-filled red circle left to the current row if that row is not valid to put a breakpoint).
- To define conditional breakpoints, after creating the breakpoint, right click in the red dot and choose **Condition....** There you can put any valid MySQL expression and state if the condition is **Is True** or **Has changed**. The former will trigger the breakpoint every time the condition is true, the latter every time the condition value has changed. (If you define a conditional breakpoint, it is not enough to step into the line with the breakpoint defined to trigger such a breakpoint.)

Figure 5.68 Conditional Breakpoints

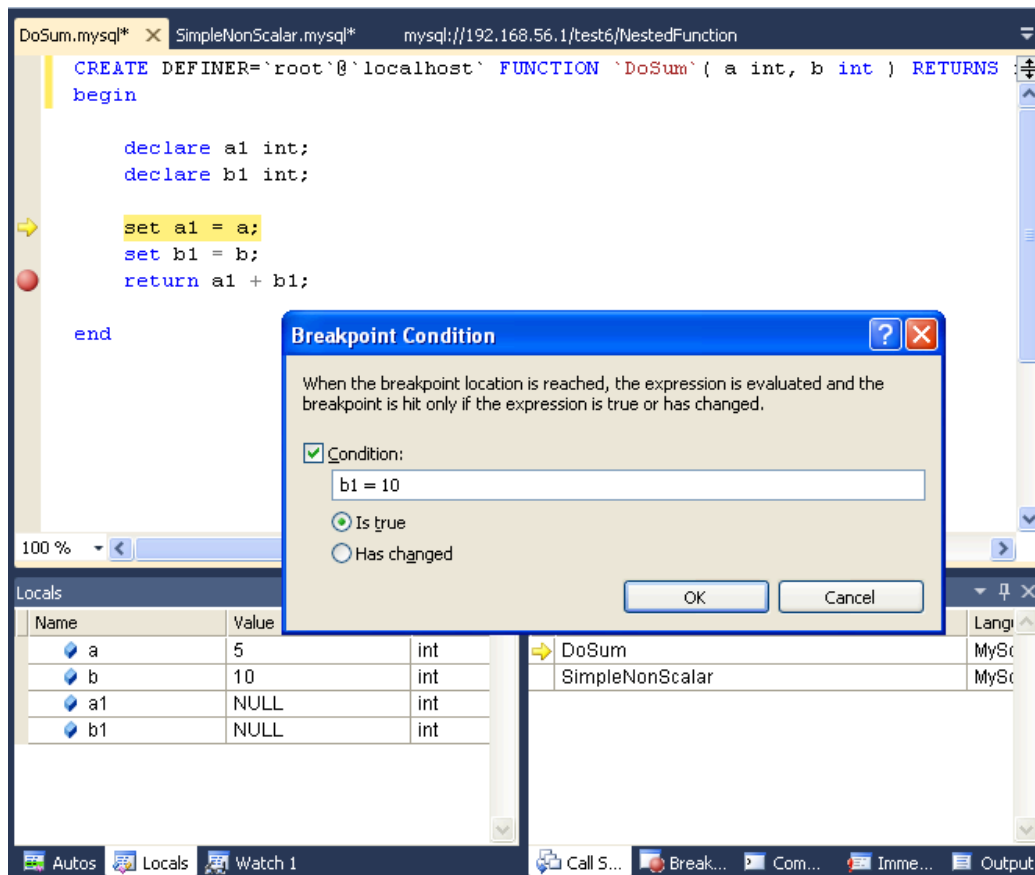
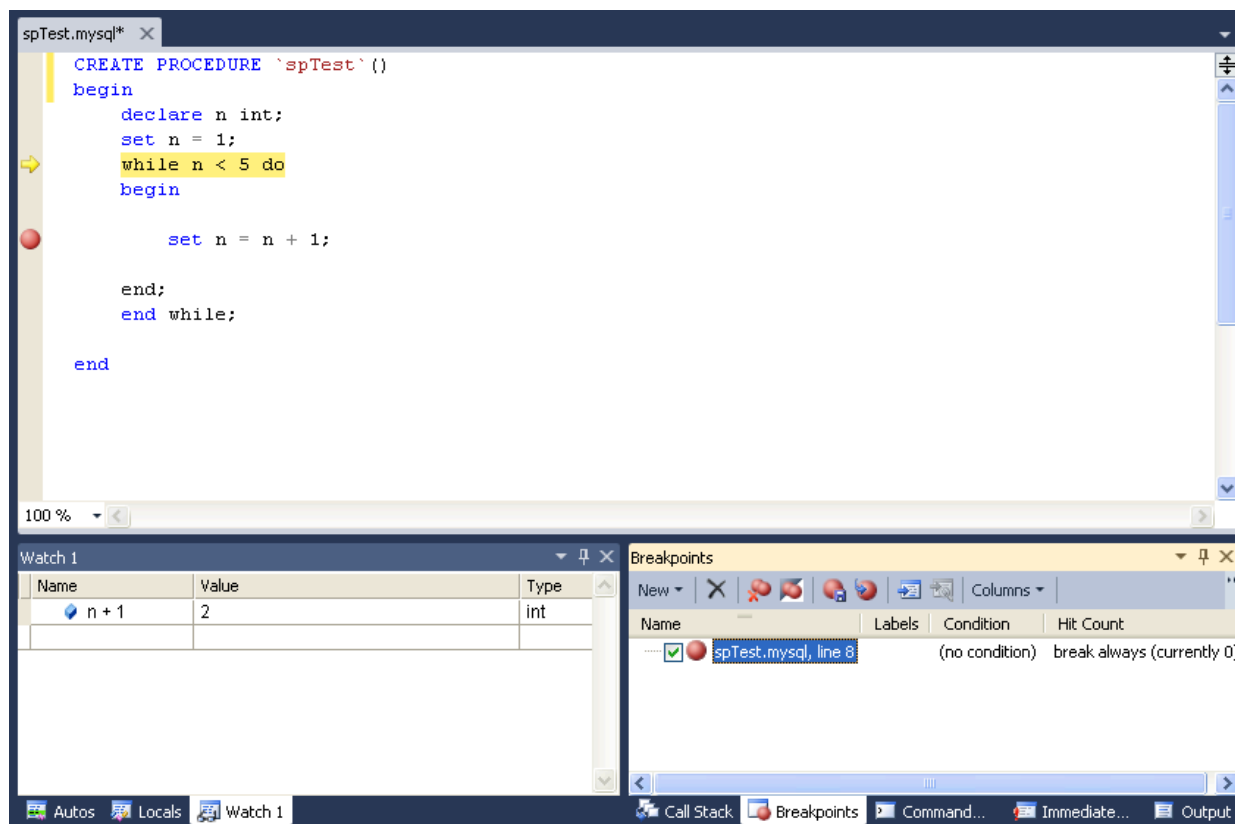


Figure 5.69 Expressions and Breakpoints



- To define pass count breakpoints, after creating the breakpoint, right click in the red dot and choose **Hit Count...** In the pop-up dialog, define the specific condition to set. For example, **break when the hit count is equal to** and a value 3 will trigger the breakpoint the third time it is hit.

Other Features

- To abort the debug session (and the execution of the current call stack of routines), press **Shift + F5**.
- To run the routine to completion (or until next breakpoint hit), press **F5**.
- For all functionality you can use (in addition to the shortcuts documented), see the options in the **Debug** menu of Visual Studio.

Limitations

- Code being debugged must not use `get_lock` or `release_lock` MySQL functions, since they are used internally by the debugger infrastructure to synchronize the debugger and the debugged routine.
- Code being debugged must avoid using any transaction code (`START TRANSACTION`, `COMMIT`, `ROLLBACK`) due to the possibility of wiping out the contents of the debugger tables. (This limitation may be removed in the future).
- You cannot debug the routines in the `serversidedebugger` database.
- The MySQL server running the routine being debugged can be any MySQL server version after 5.0, and running on any supported platform.

- Always run debug sessions on test and development servers, rather than against a MySQL production server, because debugging can cause temporary performance issues or even deadlocks. The instrumented versions of the routines being debugged use locks that might not pertain to the rest of the production code.

Keyboard Shortcuts

The following list summarizes the keyboard shortcuts for debugging:

- **F9** Toggles breakpoints
- **F11**: Step into once
- **F10**: Step over once
- **Shift + F11**: Step out once
- **F5**: Run
- **Shift + F5**: Abort current debug session

5.11 MySQL for Visual Studio Frequently Asked Questions

Questions

- [5.11.1](#): How do I know if MySQL for Visual Studio is installed?

Questions and Answers

5.11.1: How do I know if MySQL for Visual Studio is installed?

Open Visual Studio and go to **View, Toolbars**, and look for (and enable) the **MySQL** toolbar. Or, open [MySQL Installer](#) and look for the MySQL for Visual Studio product.

