

# **MySQL for Visual Studio**

---

## Abstract

This is the MySQL™ for Visual Studio Reference Manual. It documents the MySQL for Visual Studio through 1.2.8.

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

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.

**Licensing information.** 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.

Document generated on: 2019-06-07 (revision: 62307)

---

---

# Table of Contents

Preface and Legal Notices .....	v
1 General Information .....	1
1.1 New in Version 2.0 .....	1
1.2 New in Version 1.2 .....	5
2 Installing MySQL for Visual Studio .....	7
3 The MySQL Toolbar .....	9
4 Making a Connection .....	11
5 Editing .....	15
5.1 MySQL SQL Editor .....	15
5.2 Code Editors .....	16
5.3 Editing Tables .....	18
5.3.1 Column Editor .....	20
5.3.2 Column Properties .....	21
5.3.3 Table Properties .....	21
5.4 Editing Views .....	23
5.5 Editing Indexes .....	25
5.6 Editing Foreign Keys .....	26
5.7 Editing Stored Procedures and Functions .....	27
5.8 Editing Triggers .....	29
6 MySQL Website Configuration Tool .....	31
7 MySQL Project Items .....	41
7.1 Minimum Requirements .....	41
7.2 MySQL ASP.NET MVC Items .....	41
7.3 MySQL Windows Forms Items .....	48
8 MySQL Data Export Tool .....	53
9 Using the ADO.NET Entity Framework .....	59
10 DDL T4 Template Macro .....	61
11 Debugging Stored Procedures and Functions .....	63
A MySQL for Visual Studio Frequently Asked Questions .....	73



---

# Preface and Legal Notices

This is the User Manual for the MySQL for Visual Studio.

**Licensing information.** 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 © 2004, 2019, Oracle and/or its affiliates. All rights reserved.

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 installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. 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 Xeon 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, Opteron, the AMD logo, and the AMD Opteron 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

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

---

# Chapter 1 General Information

## Table of Contents

1.1 New in Version 2.0 .....	1
1.2 New in Version 1.2 .....	5

This chapter provides general information about MySQL for Visual Studio and how it has changed.

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](#).

## 1.1 New in Version 2.0

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:

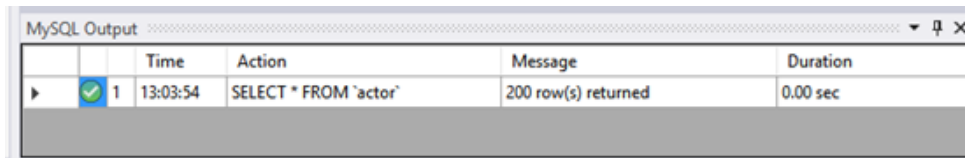
- [Viewing MySQL Query Output](#)
- [Version Support for Visual Studio](#)
- [Switching Connections from Script and Code Editors](#)
- [Making a Connection](#)
- [MySQL Toolbar](#)
- [MySQL JavaScript and Python Code Editors](#)

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 1.1 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

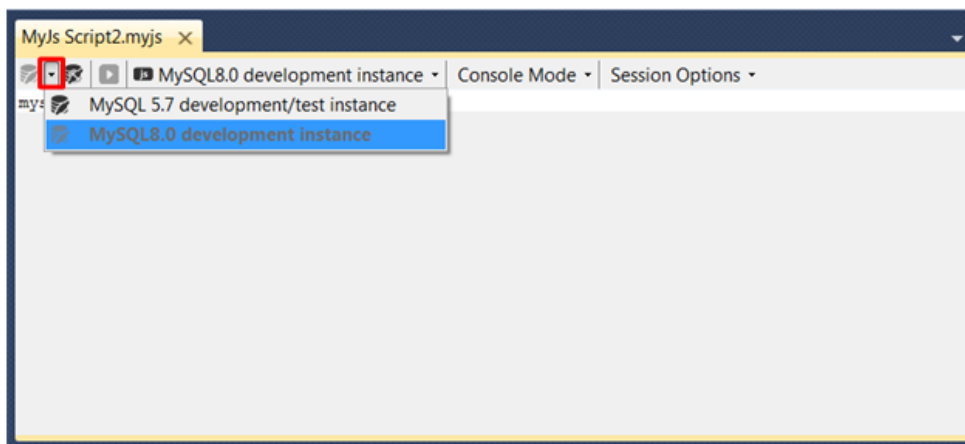
Beginning with MySQL for Visual Studio 2.0.5:

- 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.

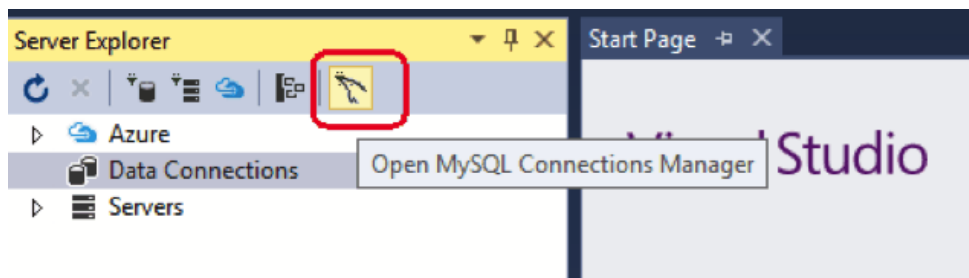
Figure 1.2 Switching Connections



## 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 1.3 Opening the MySQL Connections Manager Dialog

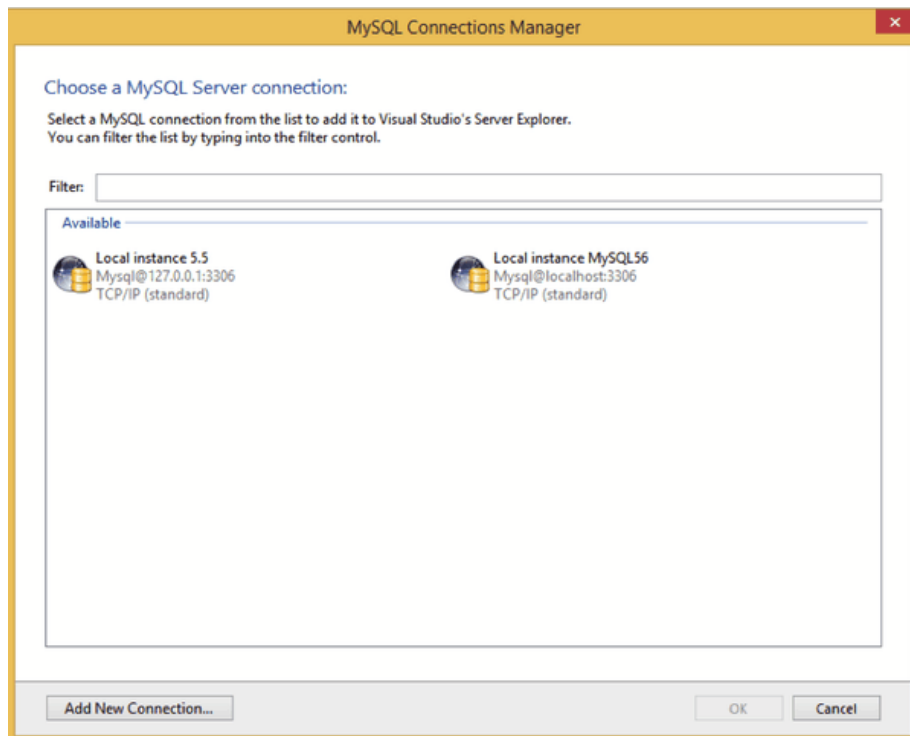


This button opens the **MySQL Connections Manager** dialog that enables the sharing of stored MySQL connections with MySQL Workbench, if it is installed. MySQL connections are displayed in a simpler way and can be created and edited from within this dialog. These connections can be imported to the Visual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



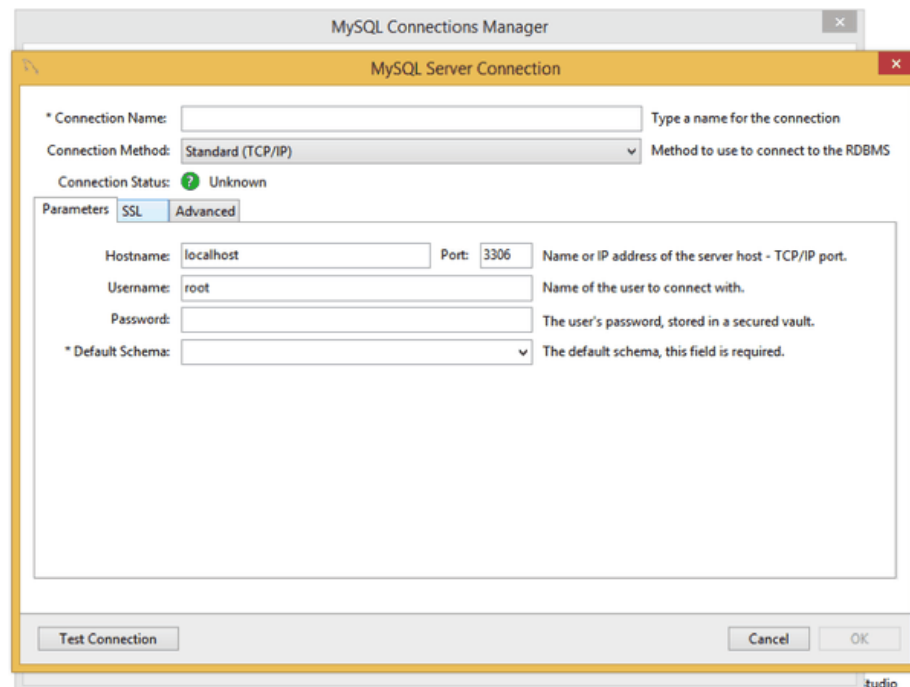
After opening the **MySQL Connections Manager**:

**Figure 1.4 MySQL Connections Manager Dialog: Choosing a Connection**



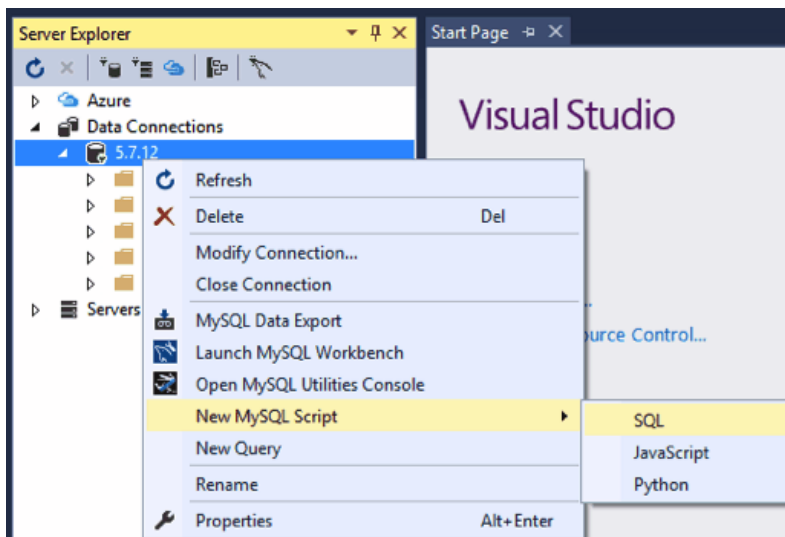
To add a new MySQL connection with the **MySQL Connections Manager**:

**Figure 1.5 MySQL Connections Manager Dialog: New Connection**



## MySQL Toolbar

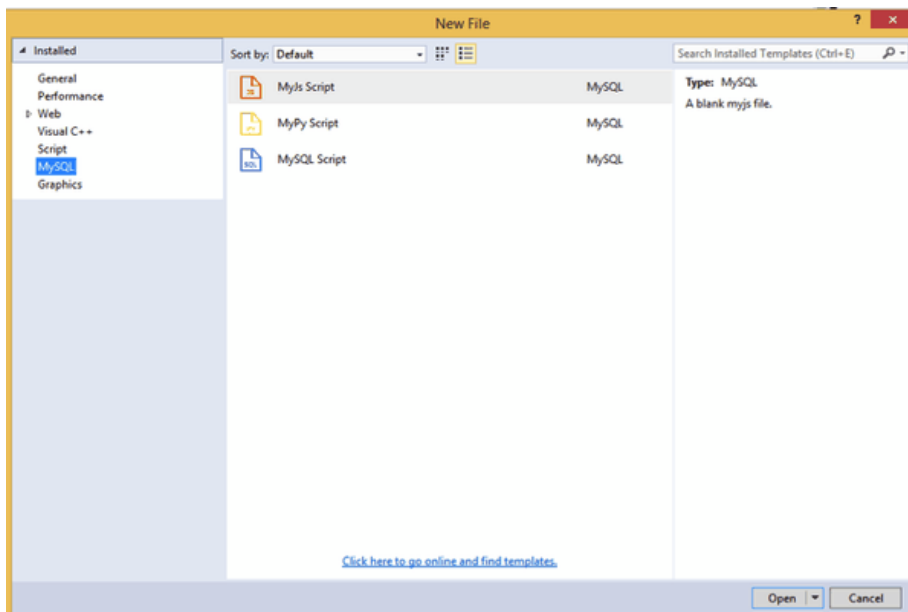
In the **Server Explorer**, and with MySQL Server 5.7, the MySQL connection context-menu was changed to show the options to create JavaScript or Python scripts, along with the existing SQL script option.

**Figure 1.6 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 1.7 MySQL Editor: Script Template**

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

Figure 1.8 MySQL Editor: JavaScript Code Editor

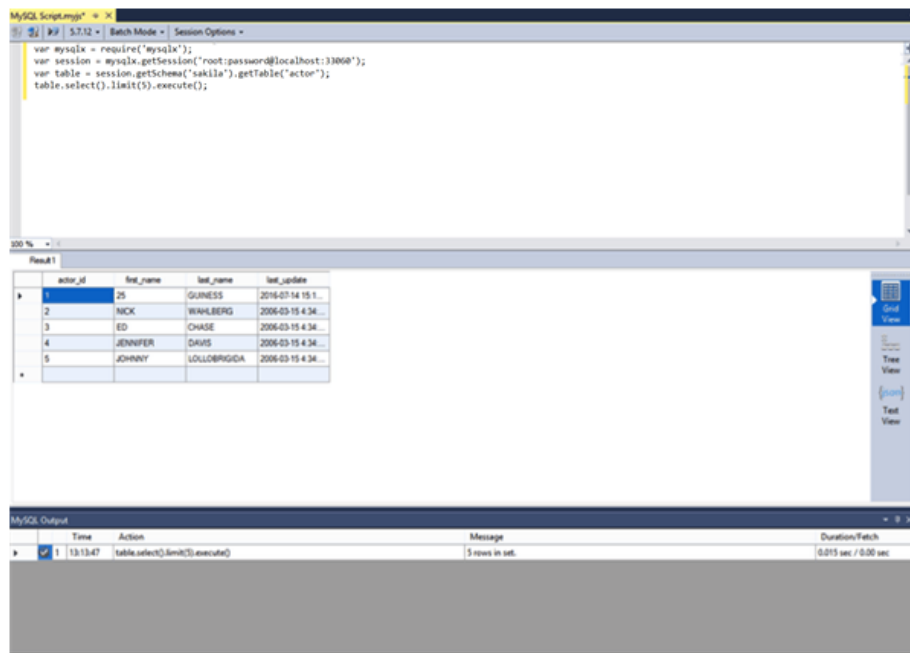
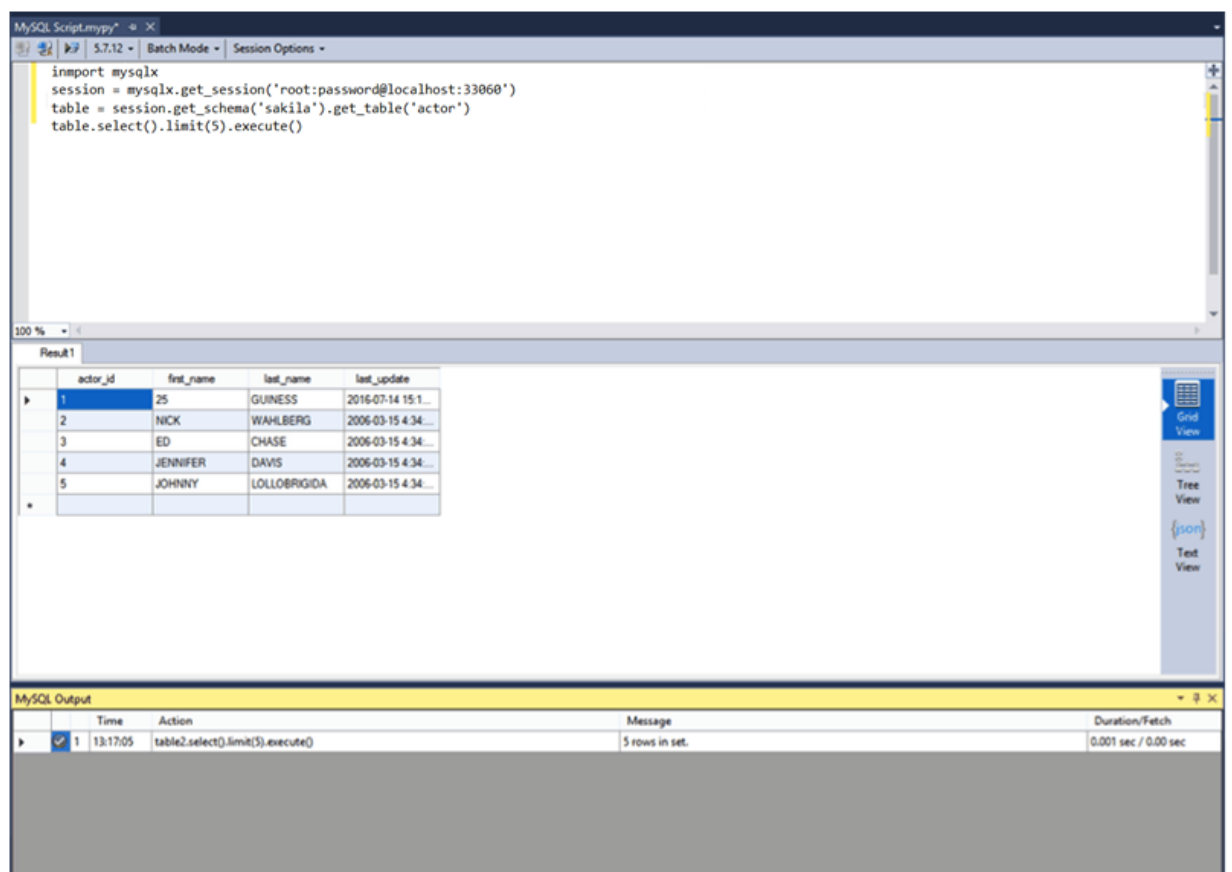


Figure 1.9 MySQL Editor: Python Code Editor



## 1.2 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.

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

## Support for MySQL 8.0 Features

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](#)).

## Version Support for Visual Studio

Beginning with 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 [Chapter 7, MySQL Project Items](#).

---

## Chapter 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 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 with several versions.

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 [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.

### Minimum Requirements

MySQL for Visual Studio 1.2.8 is compatible with Connector/NET 6.9.12, 6.10.7 and 8.0.11. Previous Connector/NET versions are not supported by this release.

MySQL for Visual Studio operates with several versions of Visual Studio, although the extent of support is based on your installed versions of Connector/NET and Visual Studio. MySQL for Visual Studio no longer supports Visual Studio 2010 or 2008. Minimum requirements for the supported versions of Visual Studio are as follows:

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

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

- Visual Studio 2015 (Community, Professional, and Enterprise):

MySQL for Visual Studio 1.2.7 or 2.0.2 with Connector/NET 6.9.8

- Visual Studio 2013 (Professional, Premium, Ultimate):

- .NET Framework 4.5.2 (install first).
- MySQL for Visual Studio 1.2.1 or 2.0.0 with Connector/NET 6.9.8

- Visual Studio 2012 (Professional, Test Professional, Premium, Ultimate):

- .NET Framework 4.5.2 (install first).
- MySQL for Visual Studio 1.2.1 or 2.0.0 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. To use Connector/NET with Express versions of Microsoft development products, use Connector/NET 6.9 or later, without installing the MySQL for Visual Studio.

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

**Table 2.1 Support Information for Companion Products**

MySQL for Visual Studio Version	MySQL Connector/NET Version Supported	Visual Studio Version Supported	MySQL Server Versions Supported	Notes
2.0	8.0, 6.10, 6.9	2017, 2015, 2013, 2012	5.7, 5.6, 5.5	Enables MySQL Configurations Manager and code editors (with MySQL 5.7).
1.2	8.0, 6.10, 6.9	2017, 2015, 2013, 2012	8.0, 5.7, 5.6, 5.5	Support for MySQL 8.0 features requires MySQL for Visual Studio 1.2.8 or higher.

## 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 before, 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 Website Configuration Tool
  - Debugging Stored Procedures and Functions
  - The DDL T4 Template Macro (to generate a database from an EF Model)

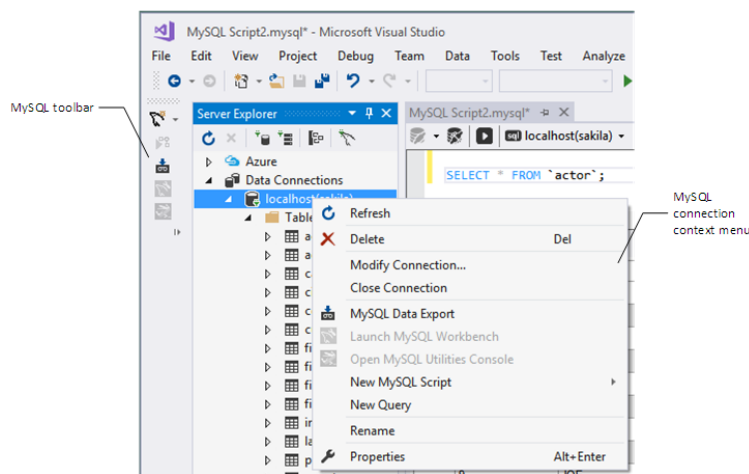
## Chapter 3 The MySQL Toolbar

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.

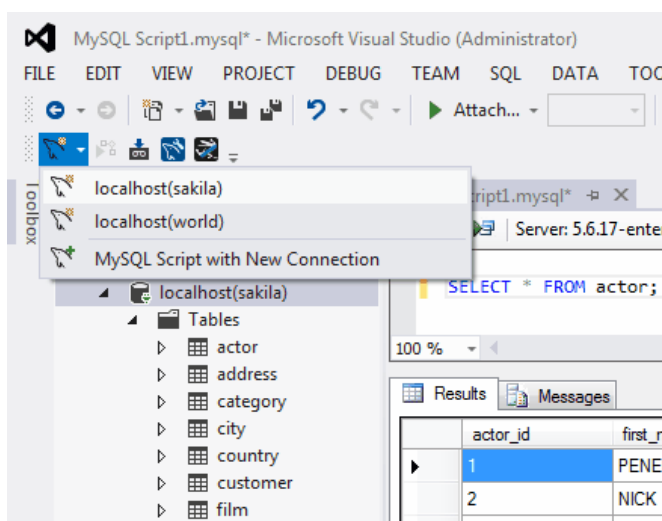
**Figure 3.1 MySQL for Visual Studio Toolbar and Context Menu**



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 3.2 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.



---

## Chapter 4 Making a Connection

- [Connect Using Server Explorer](#)
- [Connect with SSL and the X Protocol](#)

MySQL for Visual Studio enables you to create, modify, and delete connections to MySQL databases.

### Connect Using Server Explorer

To create a connection to an existing MySQL database, perform the following steps:

1. Start Visual Studio and open the Server Explorer by clicking **View** and then **Server Explorer** from the main menu.
2. Right-click the Data Connections node and then click **Add Connection**.
3. From the Add Connection window, click **Change** to open the Change Data Source dialog, 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 dialog.
4. Type a value for each of the following connection settings:

- **Server name:**

For example, [localhost](#) if the MySQL server is installed on the local computer.

- **User name:**

The name of a valid MySQL database user account, such as [root](#).

- **Password:**

The password of the user account specified previously. Optionally, click **Save my password** to avoid having to enter the password for each connection session.

- **Database name:**

A default schema name is required to open the connection. Select a name from the list.

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 opens.

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

After the 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.

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. Note that the

connection may be modified or deleted only if no active editor for its objects is opened: otherwise, you may lose your data.

## Connect with SSL and the X Protocol


Connections that use the X Protocol can be configured to use SSL with PEM files. To use SSL encryption, connections must be created using the MySQL Connections Manager included with MySQL for Visual Studio 2.0.3 (or later) or with MySQL Workbench.



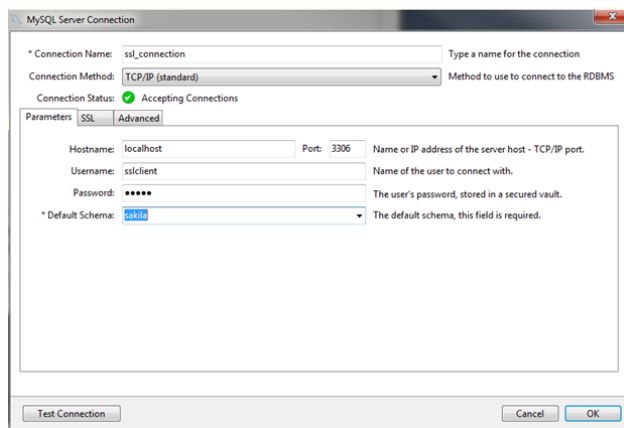
### Note

SSL must be enabled in the target MySQL server instance and the X Plugin must be installed to support connections using SSL encryption and the X Protocol.

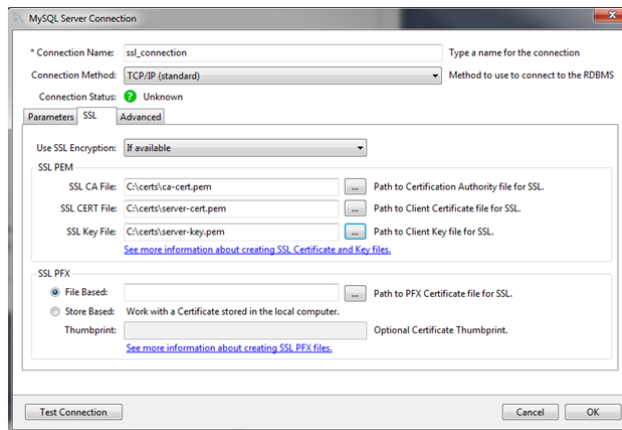
To create a connection to a MySQL database using SSL encryption and the X Protocol, perform the following steps:

1. Click the MySQL button (  ) in Visual Studio Server Explorer to open the **MySQL Connections Manager** window.
2. Click **Add New Connection** to create a new connection.
3. In the **Parameters** tab, add the host name, port, user name and password, and a default schema. Click **Test Connection** to verify the connection information. The following figure shows an example of parameter values within this tab.

**Figure 4.1 MySQL Server Connection Parameter Tab**



4. 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 4.2 MySQL Server Connection SSL Tab**

5. 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.

6. 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.



# Chapter 5 Editing

## Table of Contents

5.1 MySQL SQL Editor .....	15
5.2 Code Editors .....	16
5.3 Editing Tables .....	18
5.3.1 Column Editor .....	20
5.3.2 Column Properties .....	21
5.3.3 Table Properties .....	21
5.4 Editing Views .....	23
5.5 Editing Indexes .....	25
5.6 Editing Foreign Keys .....	26
5.7 Editing Stored Procedures and Functions .....	27
5.8 Editing Triggers .....	29

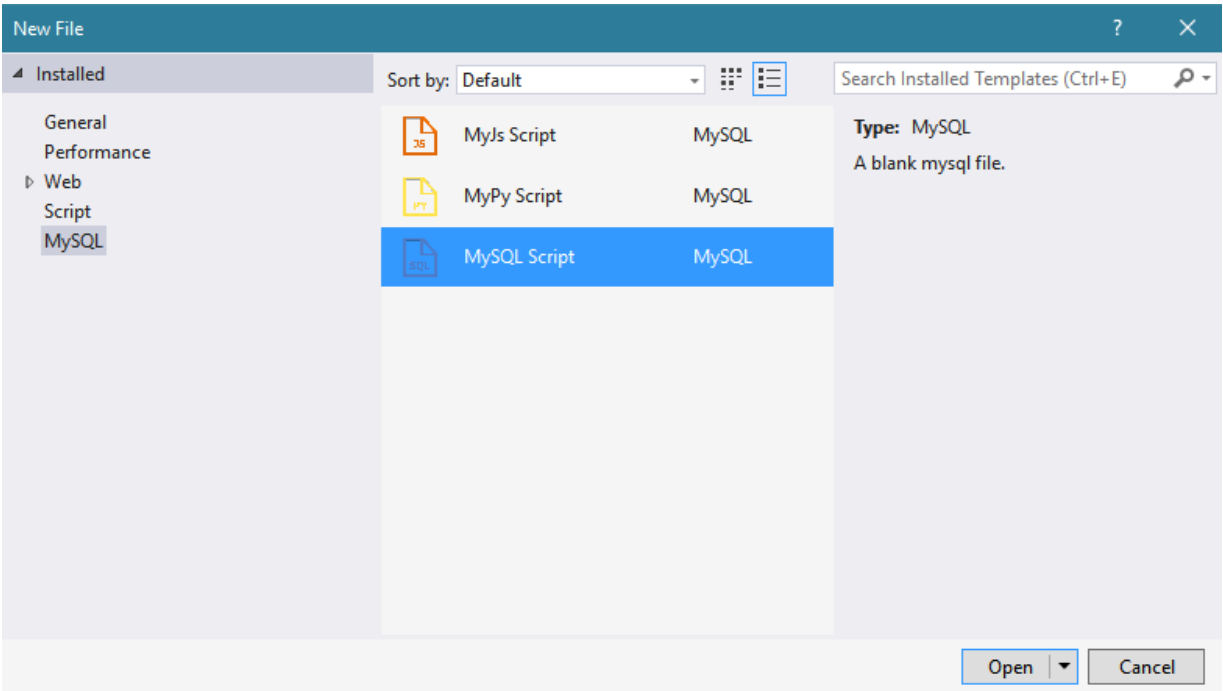
Making edits in MySQL for Visual Studio.

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.

## 5.1 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.

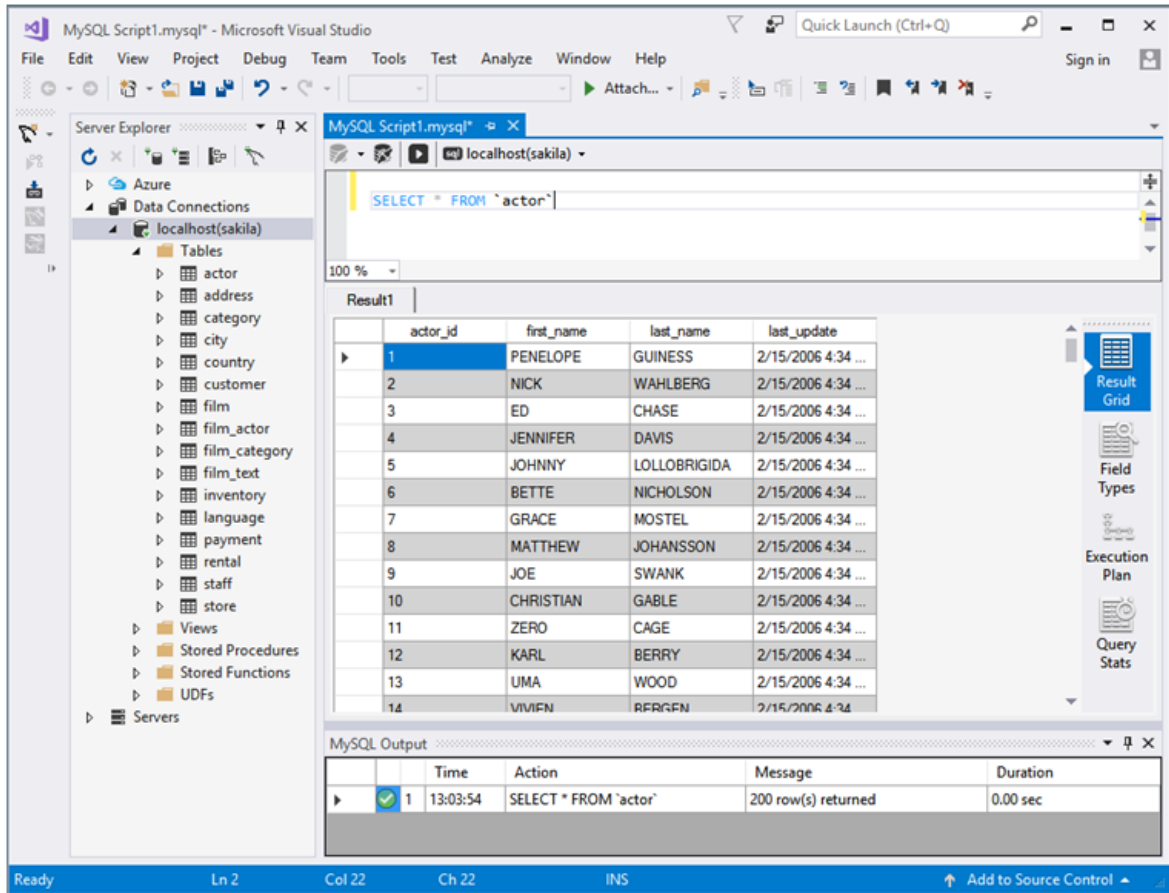
Figure 5.1 MySQL SQL Editor - New File



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 5.2 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.

## 5.2 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 2.0.2:

- 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.

## 5.3 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 5.3 Editing New Table

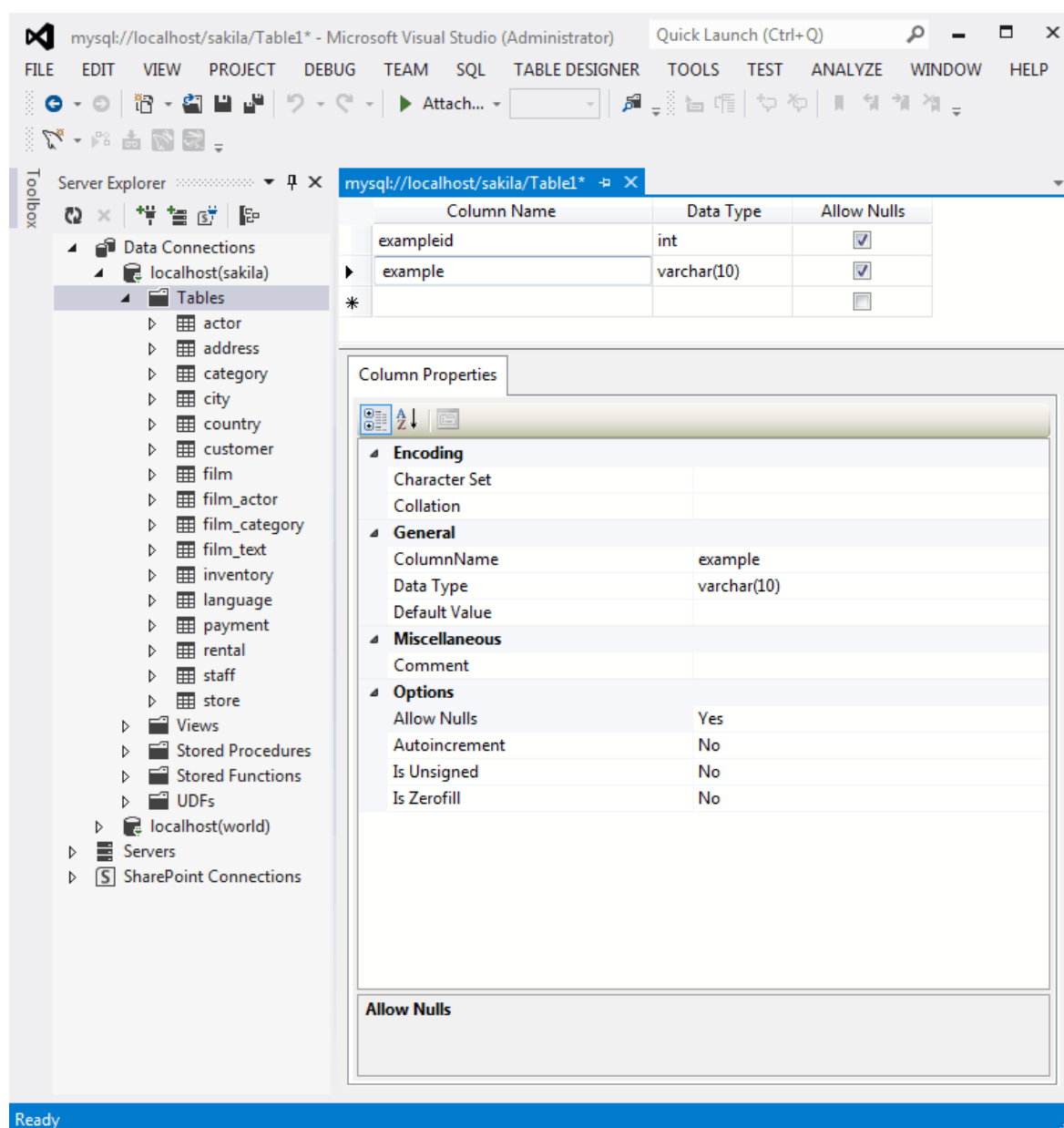


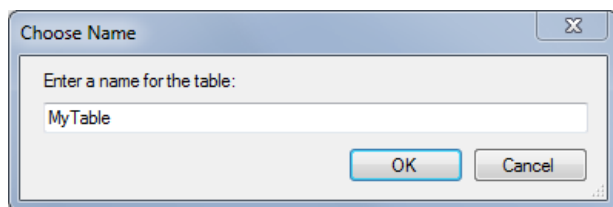
Table Designer consists of the following parts:

- **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 5.3.1, “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.

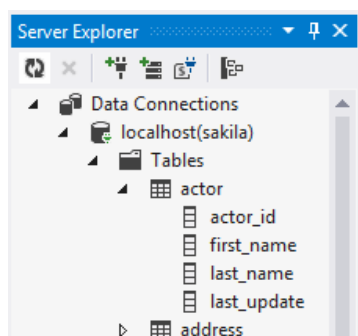
To save changes you have made in the Table Designer, press either **Save** or **Save All** on the Visual Studio main toolbar, or press **Control + S**. If you have not already named the table, you will be prompted to do so.

**Figure 5.4 Choose Table Name**



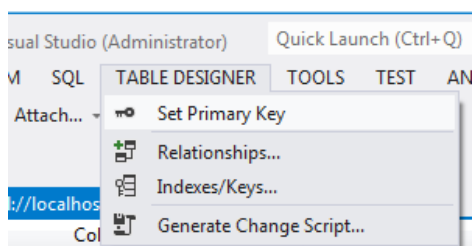
After the table is created, you can view it in the **Server Explorer**.

**Figure 5.5 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 5.6 Table Designer Main Menu**



### 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. Note that the set of column flags depends on its data type.

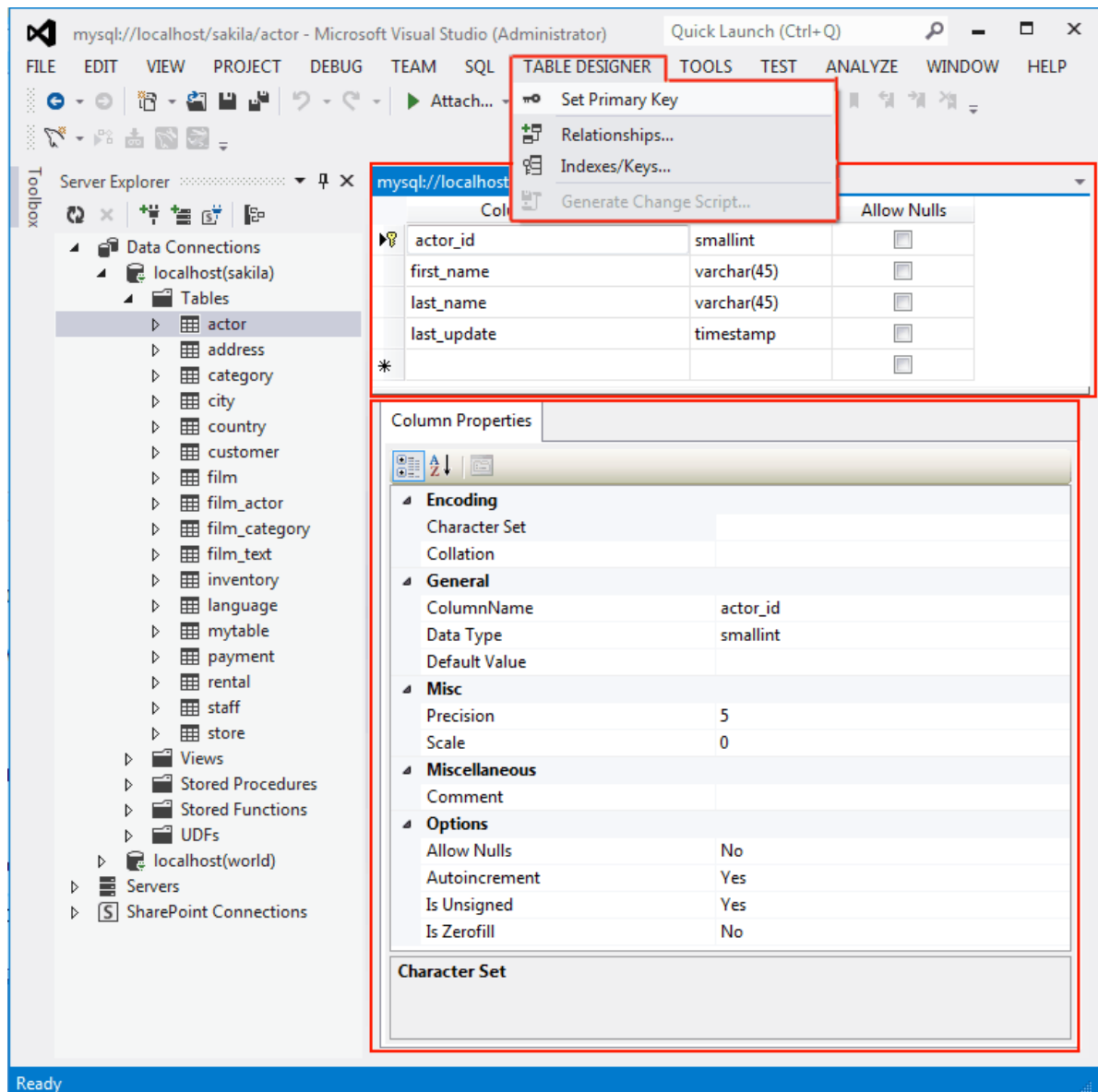
To reorder columns, index columns or foreign key columns in the Column Editor, select the whole column to reorder by clicking the selector column on the left of the column grid. Then move the column by using **Control+Up** (to move the column up) or **Control+Down** (to move the column down) keys.

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.

### 5.3.2 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.

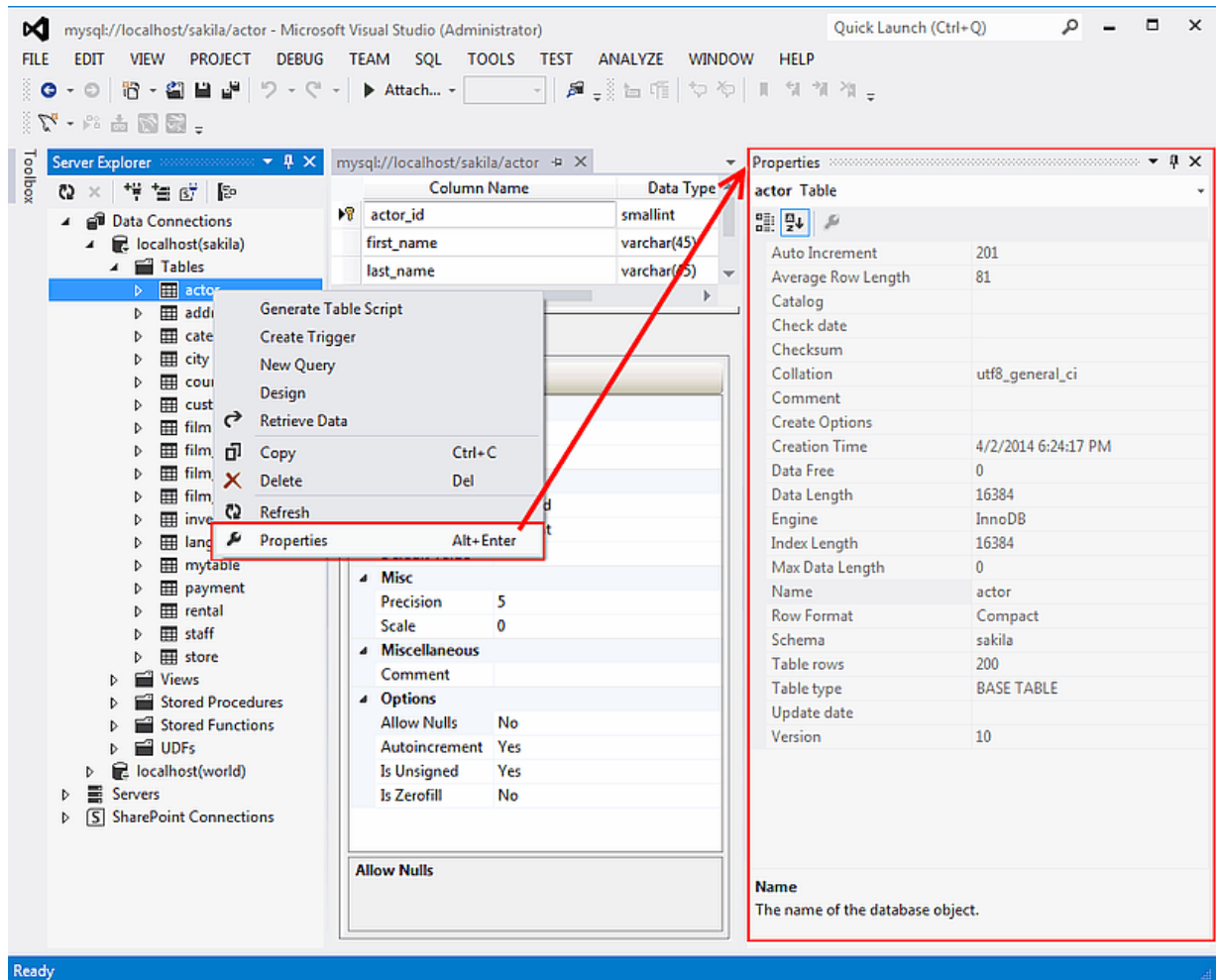
**Figure 5.7 Column Properties Panel**



### 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 5.8 Table Properties Panel



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

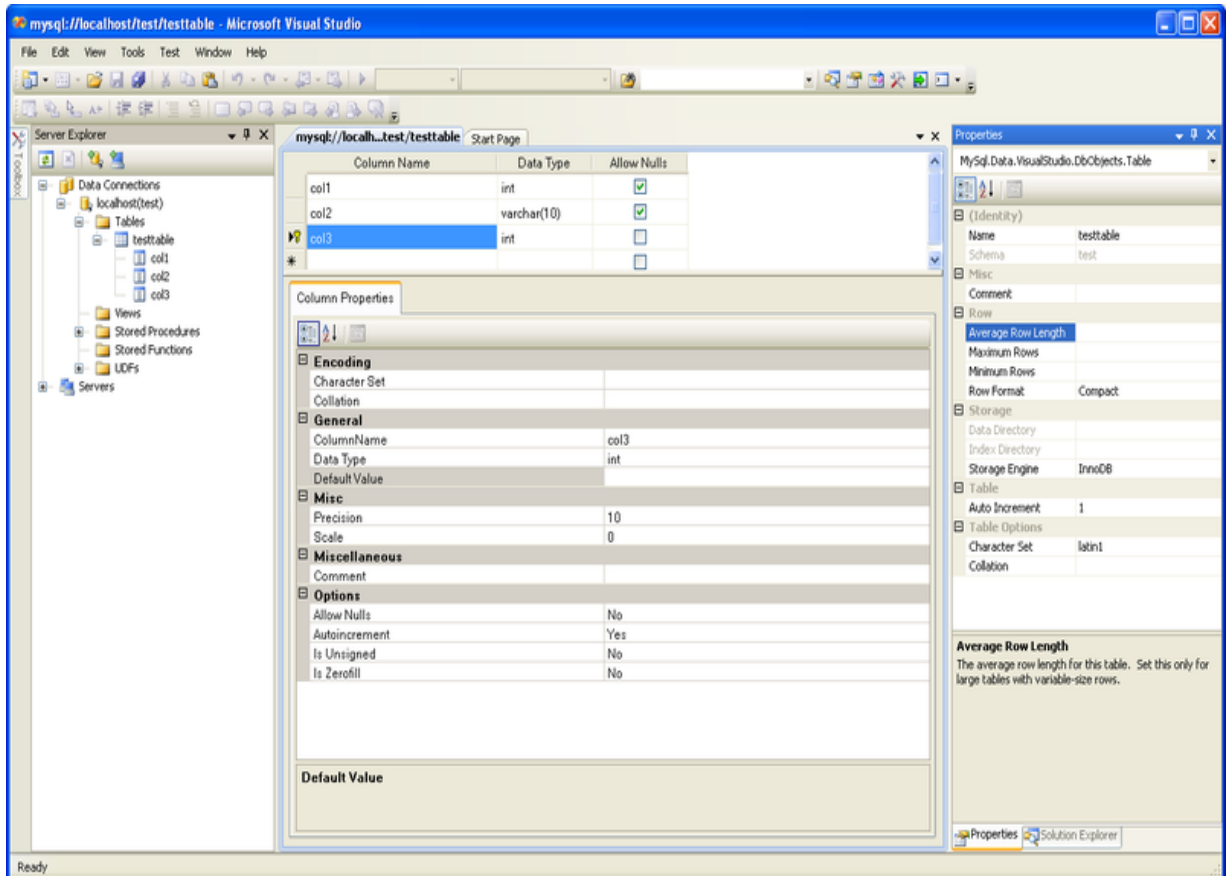
- **Auto Increment:** The next [AUTO\\_INCREMENT](#) value.
- **Average Row Length:** The [AVG\\_ROW\\_LENGTH](#) value.
- **Character Set:** The Charset value.
- **Collation:** The Collation value.
- **Comment:** Table comments.
- **Data Directory:** The directory used to store data files for this table.
- **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.

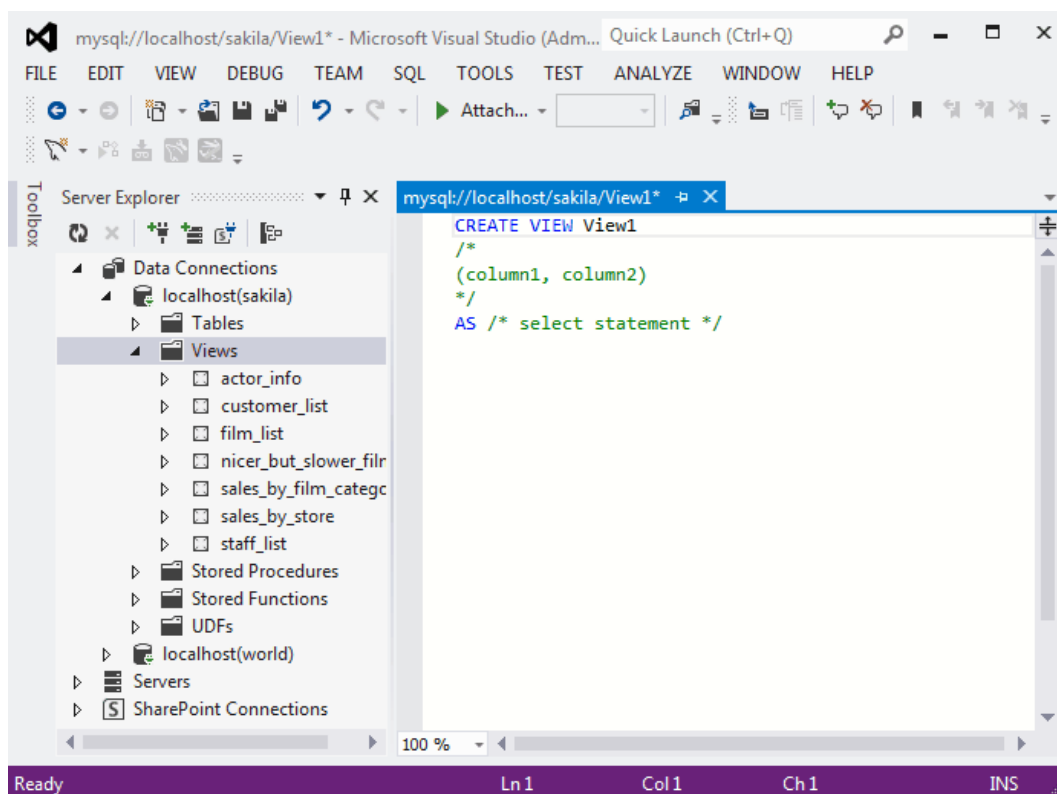
**Figure 5.9 Table Properties**



## 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 5.10 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:

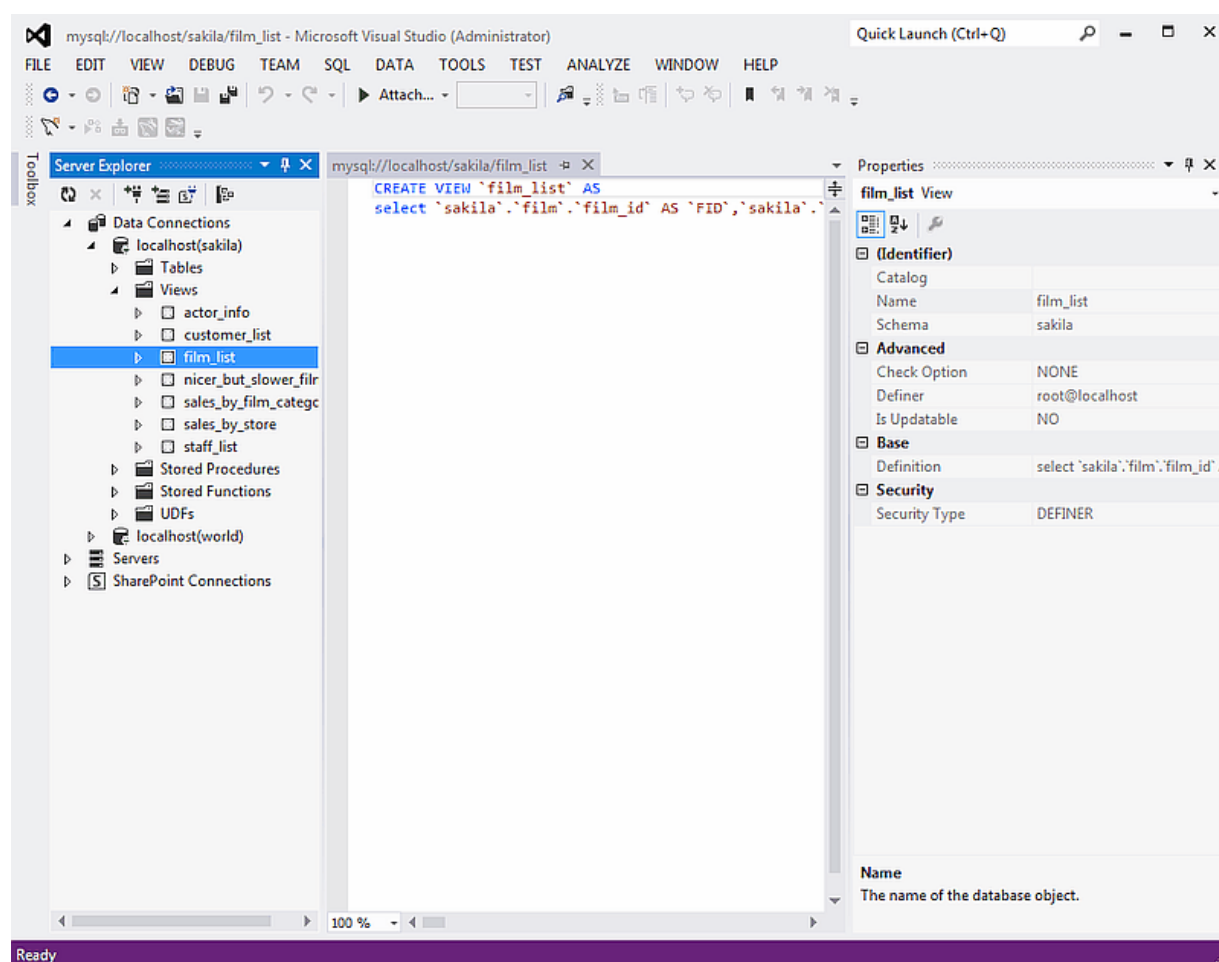
- **Catalog:** The `TABLE_CATALOG`.
- **Check Option:** Whether or not the `WITH CHECK OPTION` clause is present. For additional information, see [The View WITH CHECK OPTION Clause](#).
- **Definer:** Creator of the object.
- **Definition:** Definition of the view.
- **Is Updatable:** Whether or not 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.

The properties `Is Updatable` and `Schema` 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**.

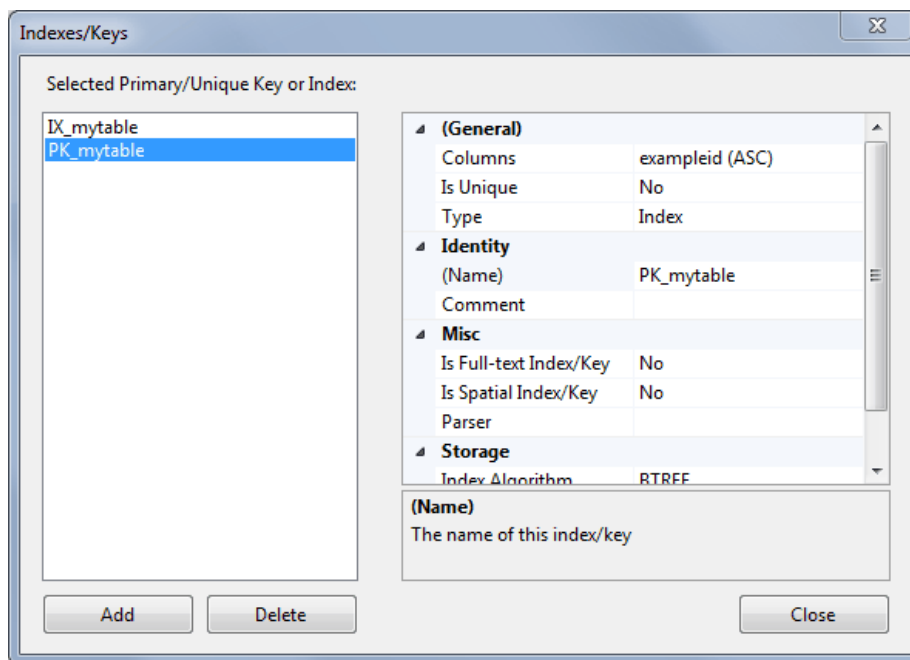
### Figure 5.11 View SQL Saved



## 5.5 Editing Indexes

Indexes management is performed using the **Indexes/Keys** dialog.

To add an index, select **Table Designer, Indexes/Keys...** from the main menu, and click **Add** to add a new index. You can then set the index name, index kind, index type, and a set of index columns.

**Figure 5.12 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.

## 5.6 Editing Foreign Keys

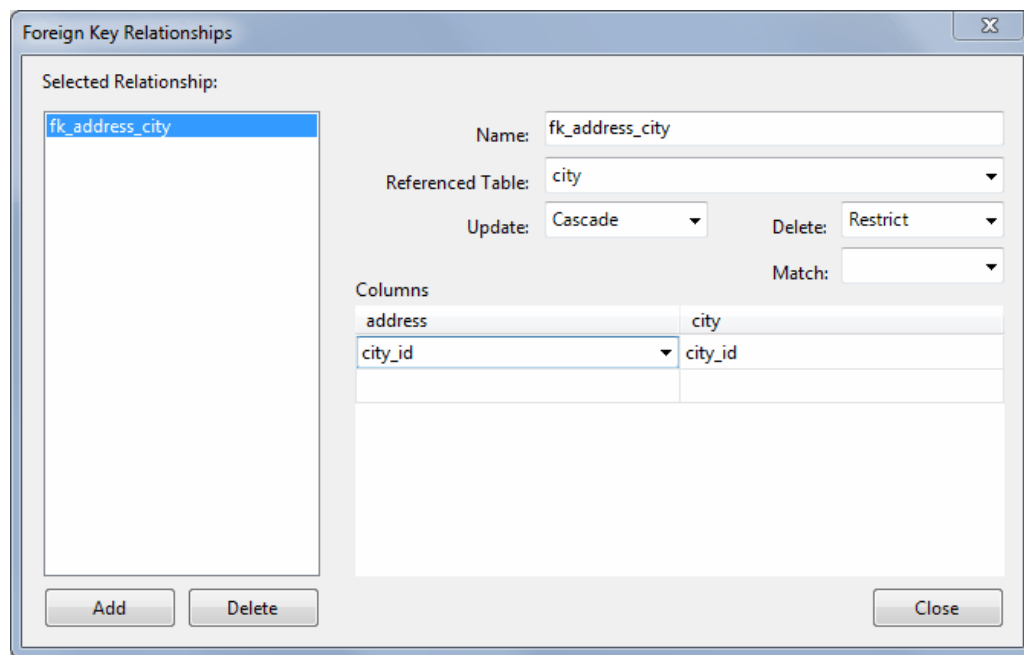
You manage [foreign keys](#) for [InnoDB](#) tables using the **Foreign Key Relationships** dialog.

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.

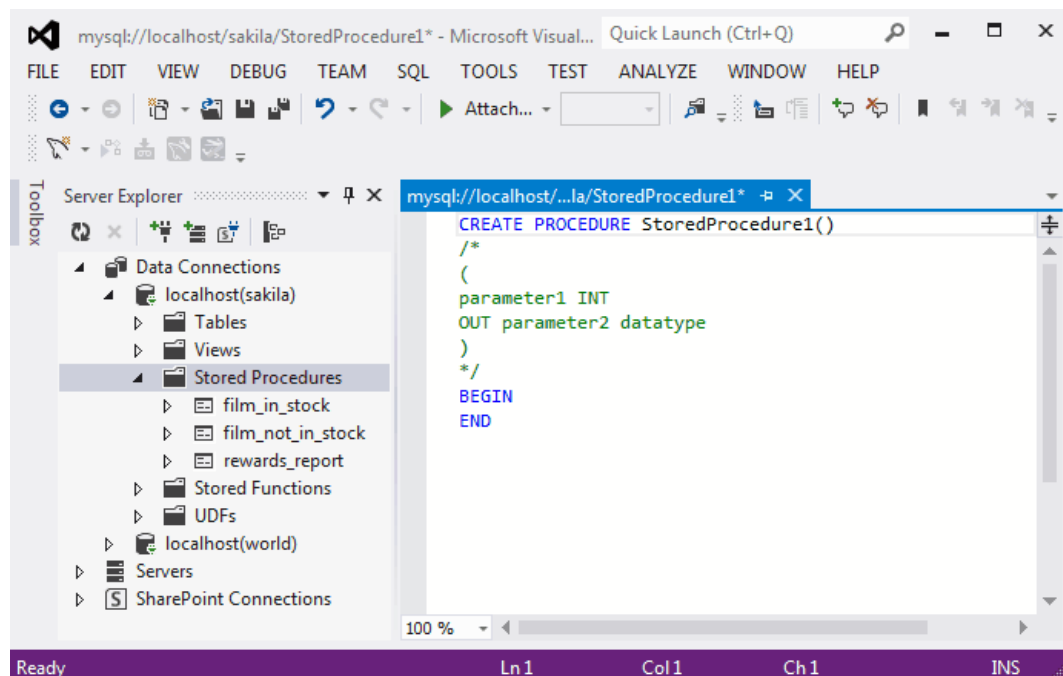
To change foreign key settings, select the required foreign key in the list box on the left. The detailed information about the foreign key is displayed in the right hand panel. Change the desired values.



**Figure 5.13 Foreign Key Relationships Dialog**

## 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 5.14 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:

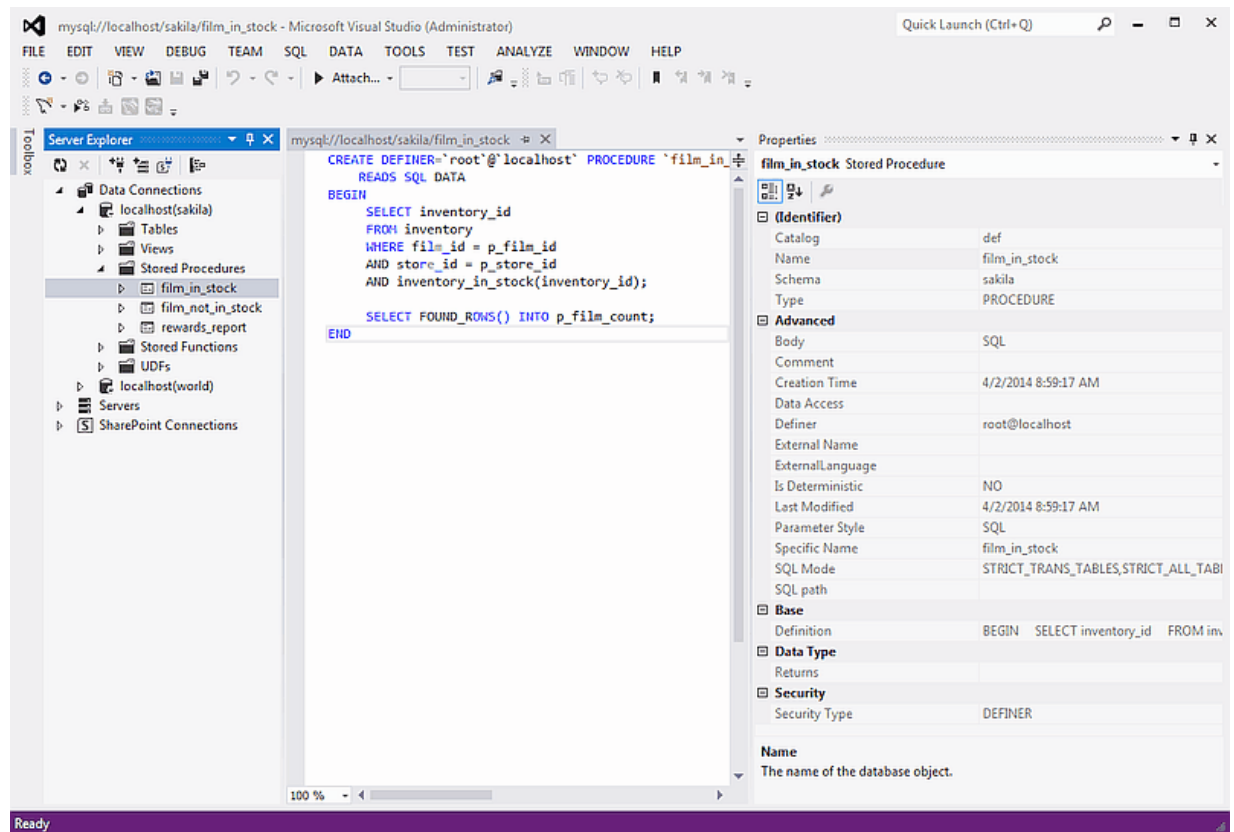
- Body
- Catalog
- Comment
- Creation Time
- Data Access
- Definer
- Definition
- External Name
- External Language
- Is Deterministic
- Last Modified
- Name
- Parameter Style
- Returns
- Schema
- Security Type
- Specific Name
- SQL Mode
- SQL Path
- Type

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 5.15 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 [Chapter 11, Debugging Stored Procedures and Functions](#).

## 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 [Chapter 11, \*Debugging Stored Procedures and Functions\*](#).

## Chapter 6 MySQL Website Configuration Tool

This MySQL for Visual Studio feature enables you to configure the Membership, Role, feature enables you to configure the Entity Framework, Membership, Role, Site Map, Personalization, Session State, and Profile Provider options without editing the configuration files. You set your configuration options within the tool, and the tool modifies your `web.config` file accordingly.

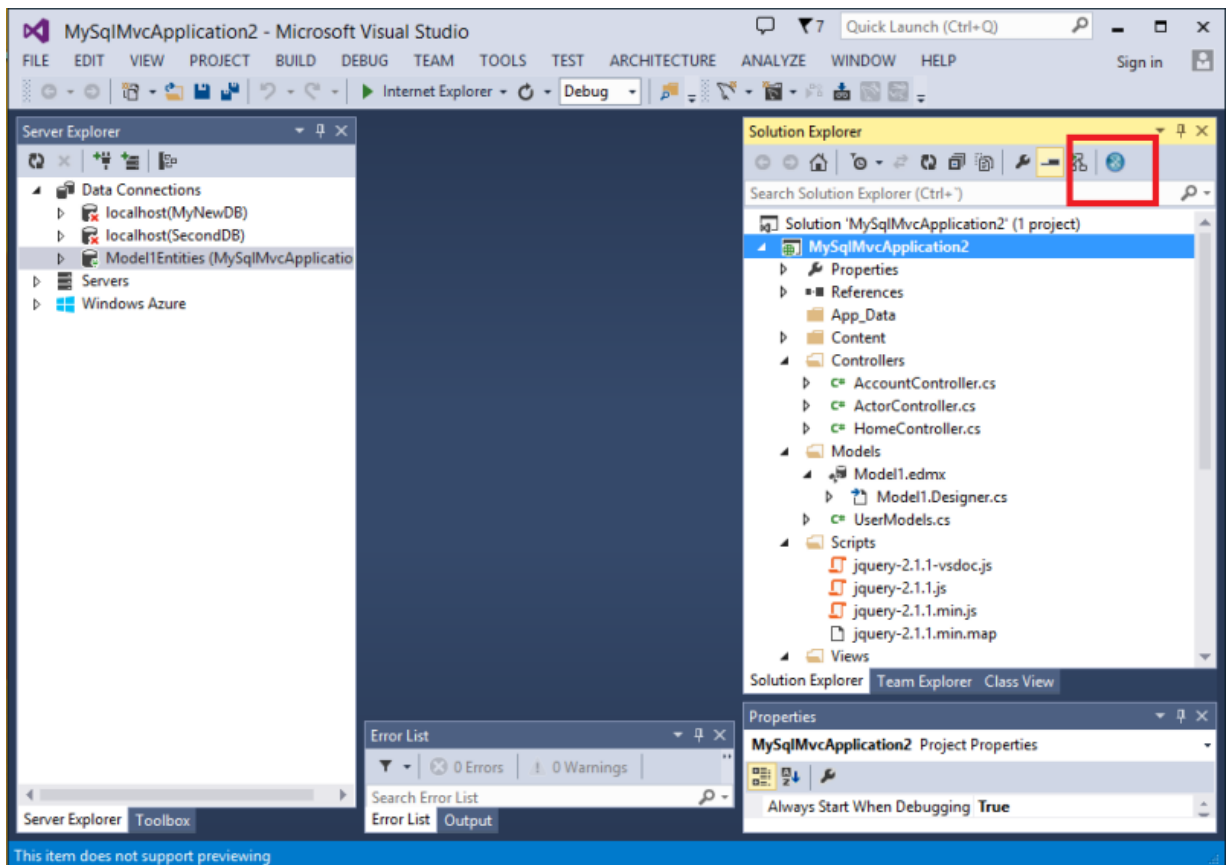


### Note

Site Map and Personalization provider support requires MySQL Connector/NET 6.9.2 or higher and MySQL for Visual Studio 1.2.1 or higher.

The MySQL Website Configuration Tool appears as a small icon on the Solution Explorer toolbar in Visual Studio, as shown in the following figure.

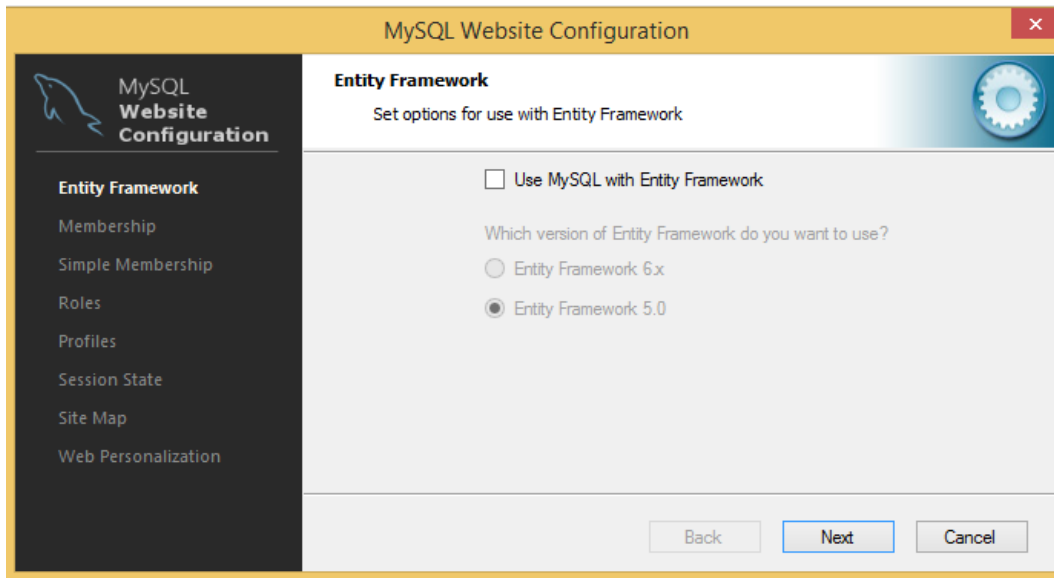
**Figure 6.1 MySQL Website Configuration Tool**



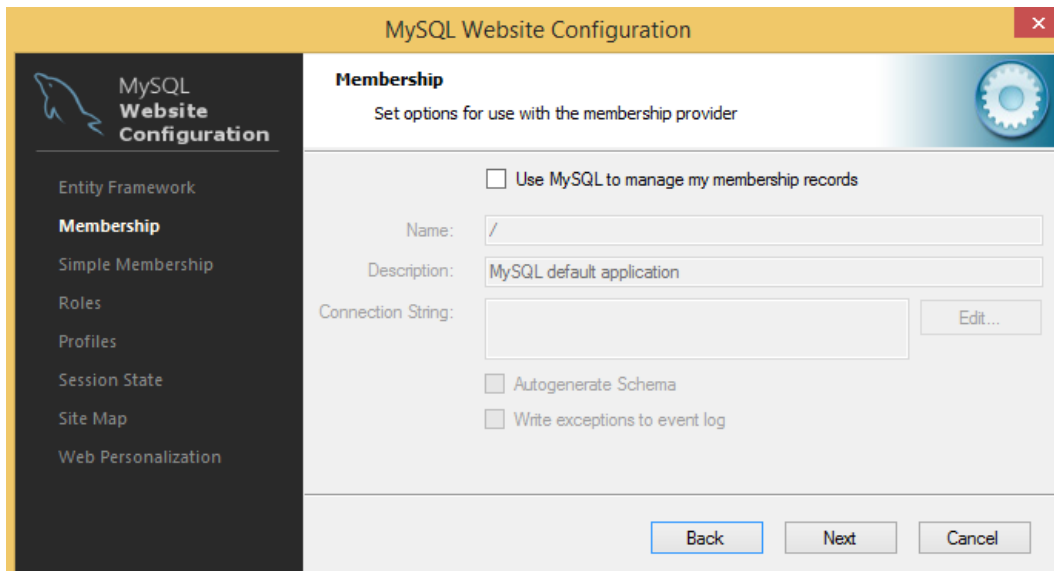
### Note

The MySQL Website Configuration Tool icon is only visible if a MySQL project is active and if Connector/NET is installed.

Clicking the Website Configuration Tool icon launches the wizard and displays the first step (Entity Framework), as the figure that follows shows.

**Figure 6.2 MySQL Website Configuration Tool - Entity Framework**

This allows you to configure your application to use Entity Framework 5 or 6 with MySQL as the database provider, adding the required references to the project and updating the configuration file accordingly.

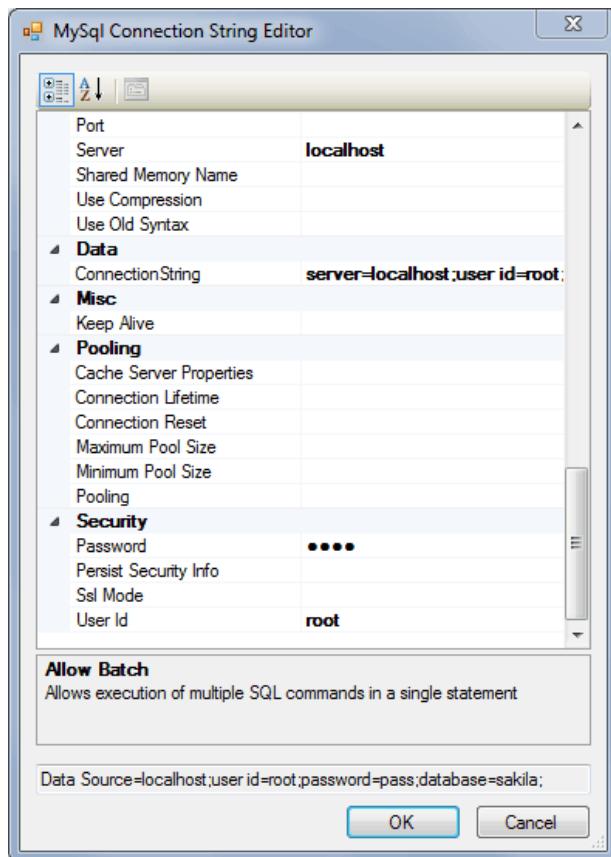
**Figure 6.3 MySQL Website Configuration Tool - Membership**

By clicking next, the next screen will allow you to enable a MySQL Membership Provider. In addition to the standard (advanced) "Membership" provider, there is also a "Simple Membership" provider. You can only choose one of these two membership providers.

## Advanced Membership Provider

To use the more advanced "Membership" provider, select the "*Use MySQL to manage my membership records*" check box to enable this. 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.

You can then click the **Edit...** button to launch the Connection String Editor:

**Figure 6.4 MySQL Website Configuration Tool - Connection String Editor****Note**

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

You can also ensure that the necessary schemas are created automatically for you by selecting the **Autogenerate Schema** check box. These schemas are used to store membership information. The database used to storage is the one specified in the connection string.

You can also ensure that exceptions generated by the application will be written to the Windows event log by selecting the **Write exceptions to event log** check box.

Clicking the **Advanced** button launches a dialog that enables you to set Membership Options. These options dictate such variables as password length required when a user signs up, whether the password is encrypted and whether the user can reset their password or not.

**Figure 6.5 MySQL Website Configuration Tool - Advanced Options**

The screenshot shows the 'Membership Options' dialog box with three sections: Password Options, User Options, and Access Options. The Password Options section includes 'Minimum Required Password Length' (6), 'Min Required Non-alphanumeric Characters' (1), 'Password strength regular expression' (empty), and 'Password Format' (Hashed). The User Options section includes 'Require Question/Answer' (unchecked), 'Enable Password Reset' (checked), 'Requires Unique Email' (checked), and 'Enable Password Retrieval' (unchecked). The Access Options section includes 'Maximum Invalid Password Attempts' (5) and 'Password Attempt Window' (10). At the bottom are 'OK' and 'Cancel' buttons.

## Simple Membership Provider

The "Simple Membership" provider is similar to the advanced version, but it includes less options. To enable, check the *"Use MySQL to manage my simple membership records"* check box.



### Note

The "Simple Membership" option was added in MySQL for Visual Studio version 1.2.3.

**Figure 6.6 MySQL Website Configuration Tool - Simple Membership**

The screenshot shows the 'MySQL Website Configuration' dialog box with the 'Simple Membership' tab selected. The left sidebar lists 'Entity Framework', 'Membership', 'Simple Membership' (selected), 'Roles', 'Profiles', 'Session State', 'Site Map', and 'Web Personalization'. The main area is titled 'Simple Membership' and 'Set options for use with the simple membership provider'. It includes a checked checkbox 'Use MySQL to manage my simple membership records', a 'Connection Name' field (LocalMySQLServer), a 'Connection String' field (port=3306;server=localhost;user id=root;database=MvcDatabase) with an 'Edit...' button, 'User Table Name' (Users), 'User Id Column' (UserId), 'User Name Column' (UserName), and a checked checkbox 'Auto Create Tables'. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

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



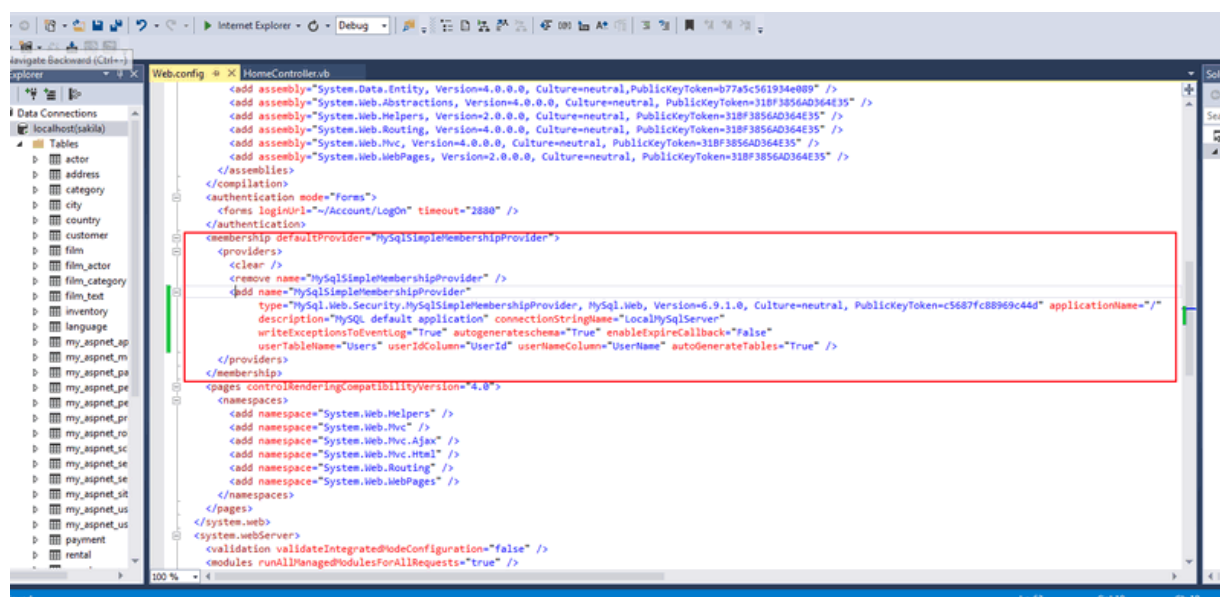
OAuth Authentication. For additional information about using OAuth authentication, see [Adding OAuth Authentication to a Project](#).

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.

Check 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.

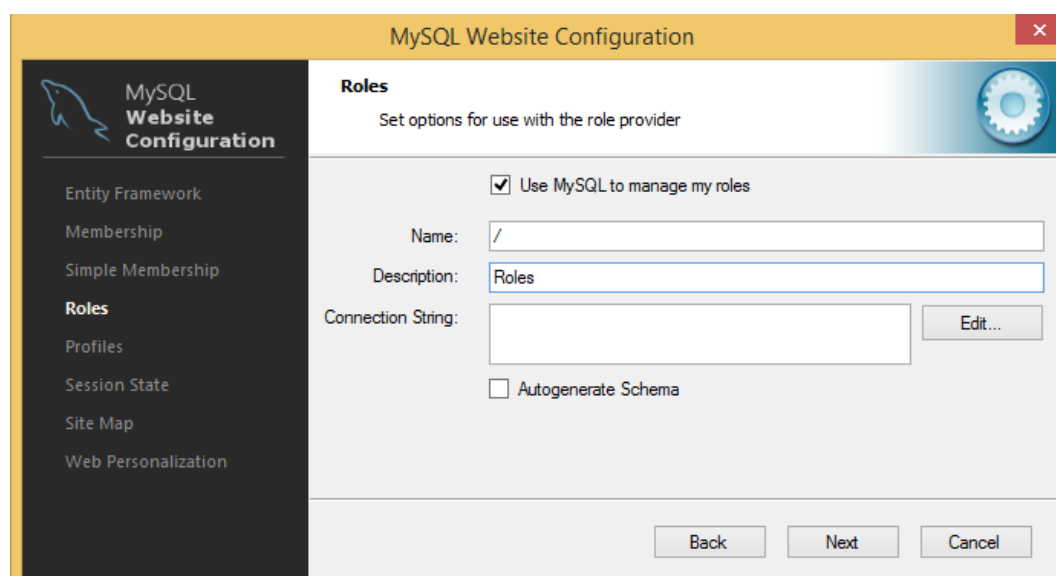
After setting up a membership provider, a new section is added to the web configuration file.

**Figure 6.7 MySQL Website Configuration Tool - Simple Membership Entry**



After setting up one of the membership providers, click **Next** to configure the Roles Provider:

**Figure 6.8 MySQL Website Configuration Tool - Roles**



Again the connection string can be edited, a description added and Autogenerate Schema can be enabled before clicking **Next** to go to the Profiles Provider screen:

**Figure 6.9 MySQL Website Configuration Tool - Profiles**

The screenshot shows the 'MySQL Website Configuration' window with the 'Profiles' tab selected. The left sidebar lists various configuration options: Entity Framework, Membership, Simple Membership, Roles, Profiles (highlighted), Session State, Site Map, and Web Personalization. The main area is titled 'Profiles' and contains the instruction 'Set options for use with the profile provider'. It features a checkbox 'Use MySQL to manage my profiles' which is checked. Below this are input fields for 'Name' (containing '/') and 'Description' (containing 'Profiles'). A 'Connection String' field is present with an 'Edit...' button. At the bottom, there are three checkboxes: 'Autogenerate Schema', 'Write exceptions to event log', and 'Callback for session end event', all of which are unchecked. Navigation buttons 'Back', 'Next', and 'Cancel' are at the bottom right.

This screen display similar options to the previous screens.

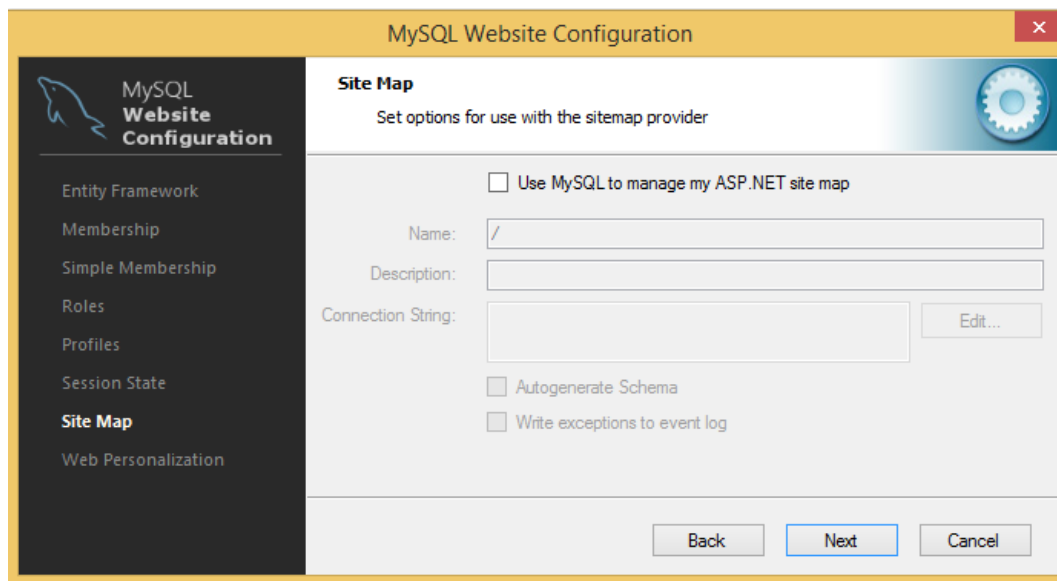
Click **Next** to proceed to the Session State configuration page:

**Figure 6.10 MySQL Website Configuration Tool - Session State**

The screenshot shows the 'MySQL Website Configuration' window with the 'Session State' tab selected. The left sidebar lists various configuration options: Entity Framework, Membership, Simple Membership, Roles, Profiles, Session State (highlighted), Site Map, and Web Personalization. The main area is titled 'Session State' and contains the instruction 'Set options for use with the session state provider'. It features a checkbox 'Use MySQL to manage my ASP.Net session state' which is checked. Below this are input fields for 'Name' (containing '/') and 'Description' (containing 'Sessions'). A 'Connection String' field is present with an 'Edit...' button. At the bottom, there are two checkboxes: 'Autogenerate Schema' and 'Write exceptions to event log', both of which are unchecked. Navigation buttons 'Back', 'Next', and 'Cancel' are at the bottom right.

Click **Next** to proceed to the Site Map configuration page:

Figure 6.11 MySQL Website Configuration Tool - Site Map

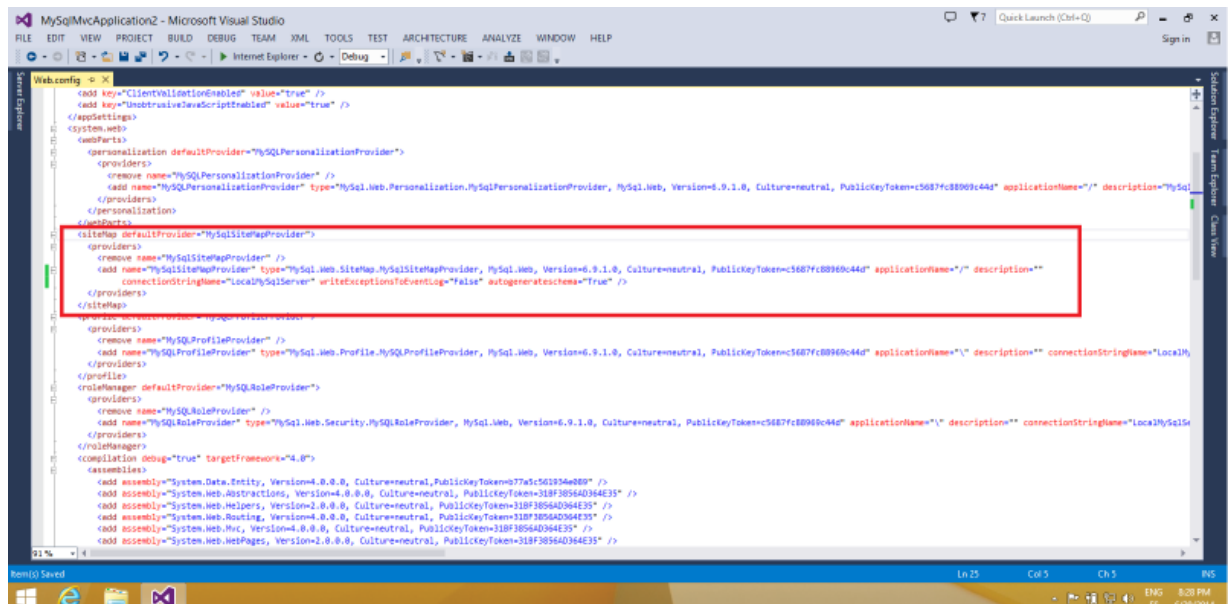


The Site Map provider builds a site map from a MySQL database, and builds a complete tree of the [SitemapNode](#) objects. It also provides methods so that the generated nodes can be read from the site map.

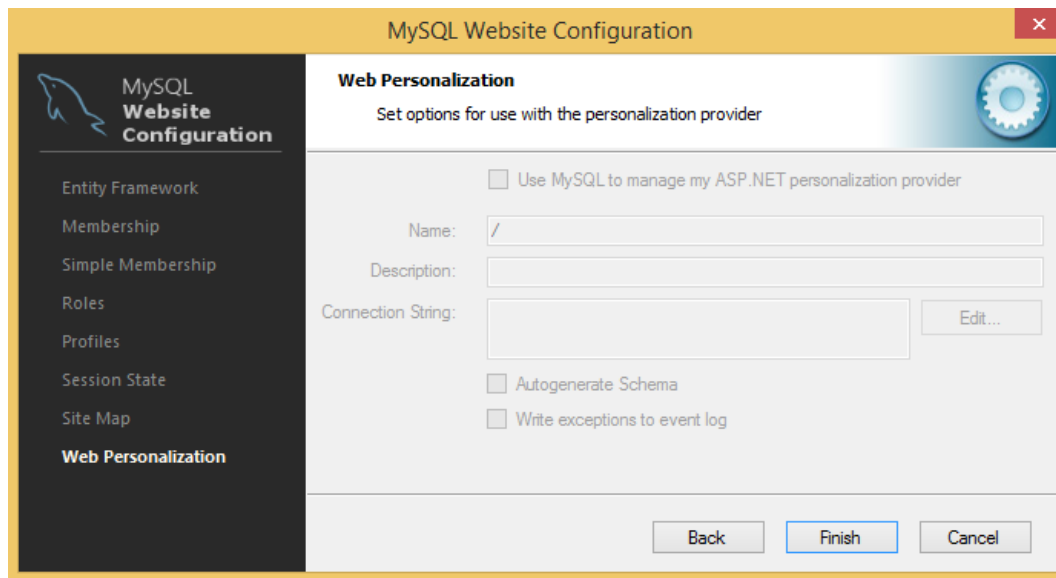
The required configuration options: A name for the connection string, and a connection string that contains a valid database with a local or remote MySQL server instance.

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

Figure 6.12 MySQL Website Configuration Tool - Site Map, Configuration Entry



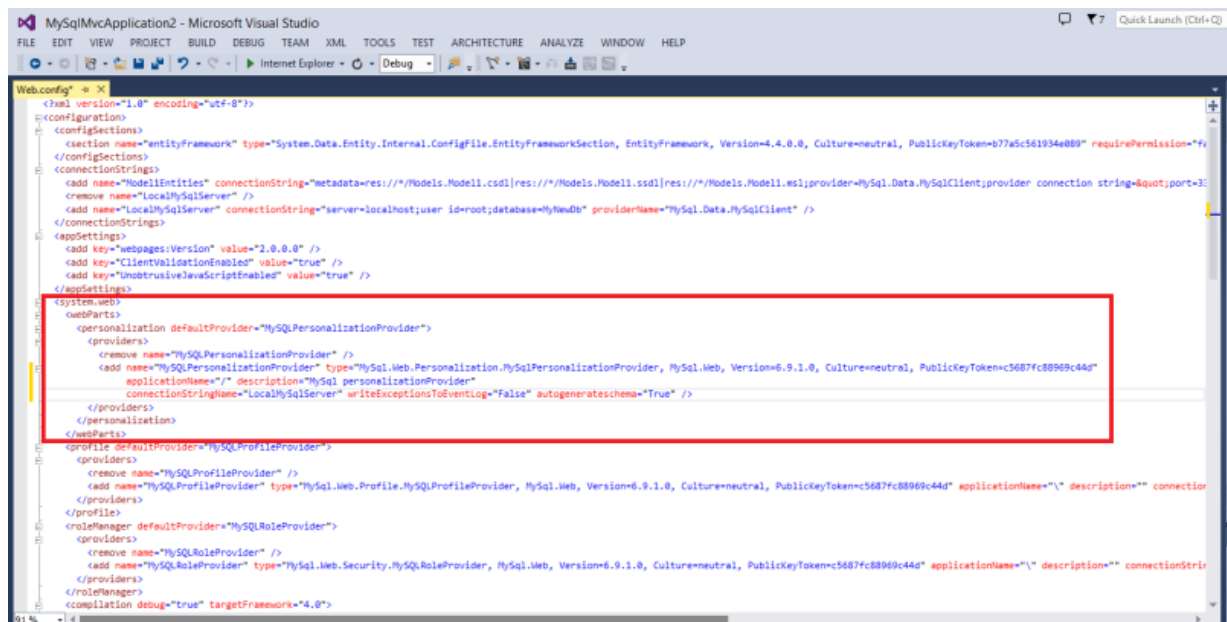
Click **Next** to proceed to the Web Personalization configuration page:

**Figure 6.13 MySQL Website Configuration Tool - Web Personalization**

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.

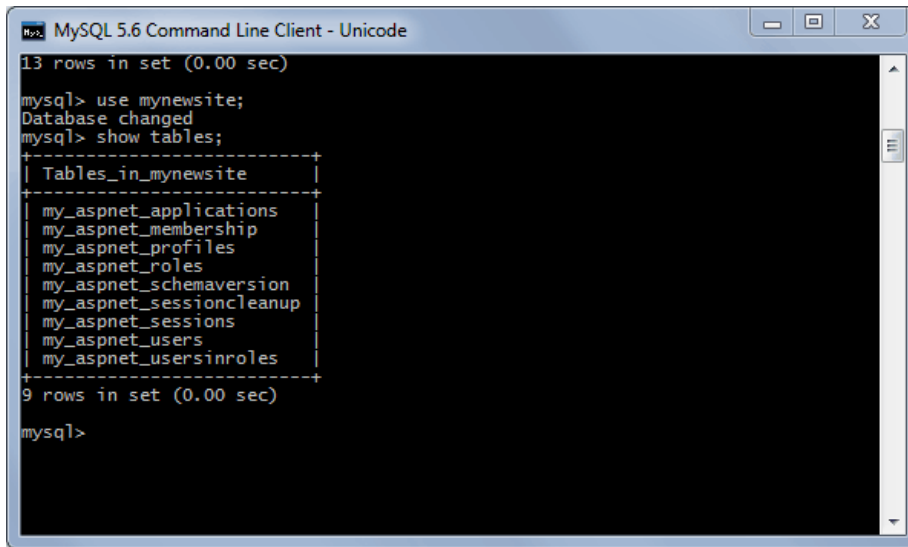
The required configuration options: A name for the connection string, and a connection string that contains a valid database with a local or remote MySQL server instance.

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

**Figure 6.14 MySQL Website Configuration Tool - Web Personalization, Configuration Entry**

Once you have set up the optional Web Personalization options, click **Finish** to exit the wizard.

At this point, select the **Authentication Type** to **From Internet**. Launch the **WEBSITE, ASP.NET Configuration** tool and select the **Security** tab. Click the **Select authentication type** link and ensure that the **From the internet** radio button is selected. You can now examine the database you created to store membership information. All the necessary tables will have been created for you:

**Figure 6.15 MySQL Website Configuration Tool - Tables**A screenshot of a MySQL 5.6 Command Line Client window titled "MySQL 5.6 Command Line Client - Unicode". The window has a black background with white text. The command prompt shows the following sequence of commands and output:  
13 rows in set (0.00 sec)  
mysql> use mynewsite;  
Database changed  
mysql> show tables;  
+-----+  
| Tables\_in\_mynewsite |  
+-----+  
| my\_aspnet\_applications |  
| my\_aspnet\_membership |  
| my\_aspnet\_profiles |  
| my\_aspnet\_roles |  
| my\_aspnet\_schemaversion |  
| my\_aspnet\_sessioncleanup |  
| my\_aspnet\_sessions |  
| my\_aspnet\_users |  
| my\_aspnet\_usersinroles |  
+-----+  
9 rows in set (0.00 sec)  
mysql>  
The window includes standard Windows-style window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side of the text area.

```
MySQL 5.6 Command Line Client - Unicode
13 rows in set (0.00 sec)
mysql> use mynewsite;
Database changed
mysql> show tables;
+-----+
| Tables_in_mynewsite |
+-----+
| my_aspnet_applications |
| my_aspnet_membership |
| my_aspnet_profiles |
| my_aspnet_roles |
| my_aspnet_schemaversion |
| my_aspnet_sessioncleanup |
| my_aspnet_sessions |
| my_aspnet_users |
| my_aspnet_usersinroles |
+-----+
9 rows in set (0.00 sec)
mysql>
```



---

## Chapter 7 MySQL Project Items

### Table of Contents

7.1 Minimum Requirements .....	41
7.2 MySQL ASP.NET MVC Items .....	41
7.3 MySQL Windows Forms Items .....	48

This tutorial uses MySQL MVC Item templates to set up a MVC web application. At the end of the tutorial a Windows Forms Item with MySQL connectivity will also be created.

### 7.1 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.

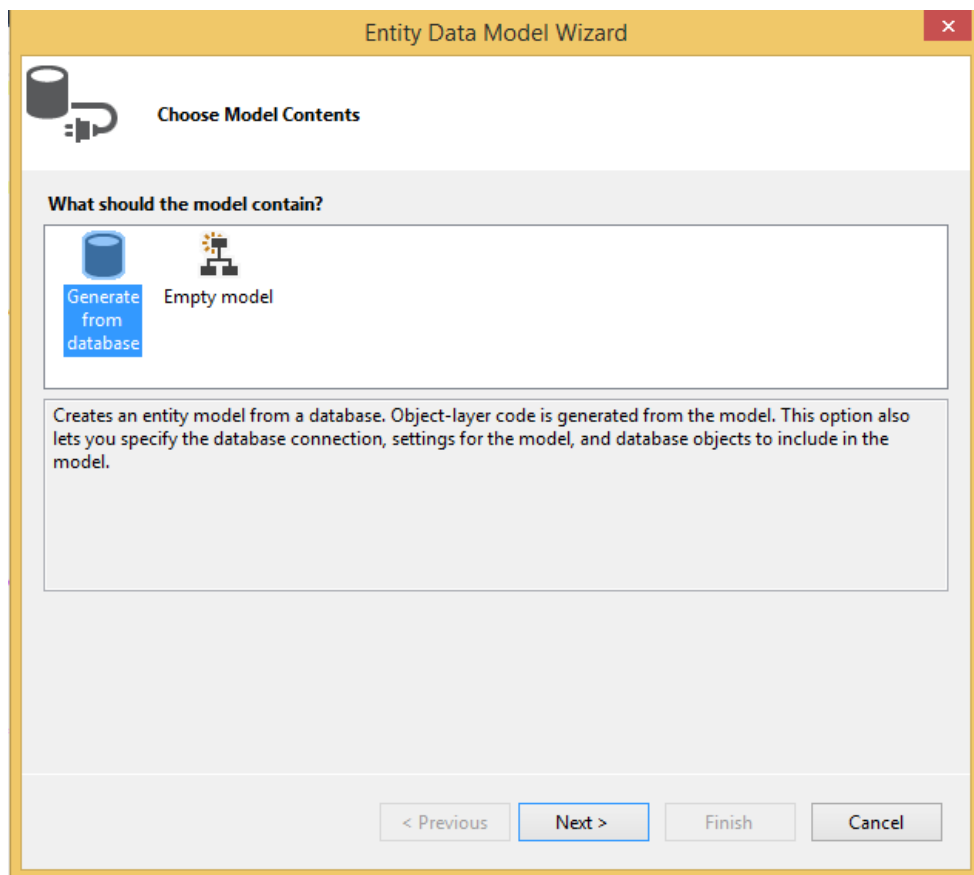
### 7.2 MySQL ASP.NET MVC Items

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.

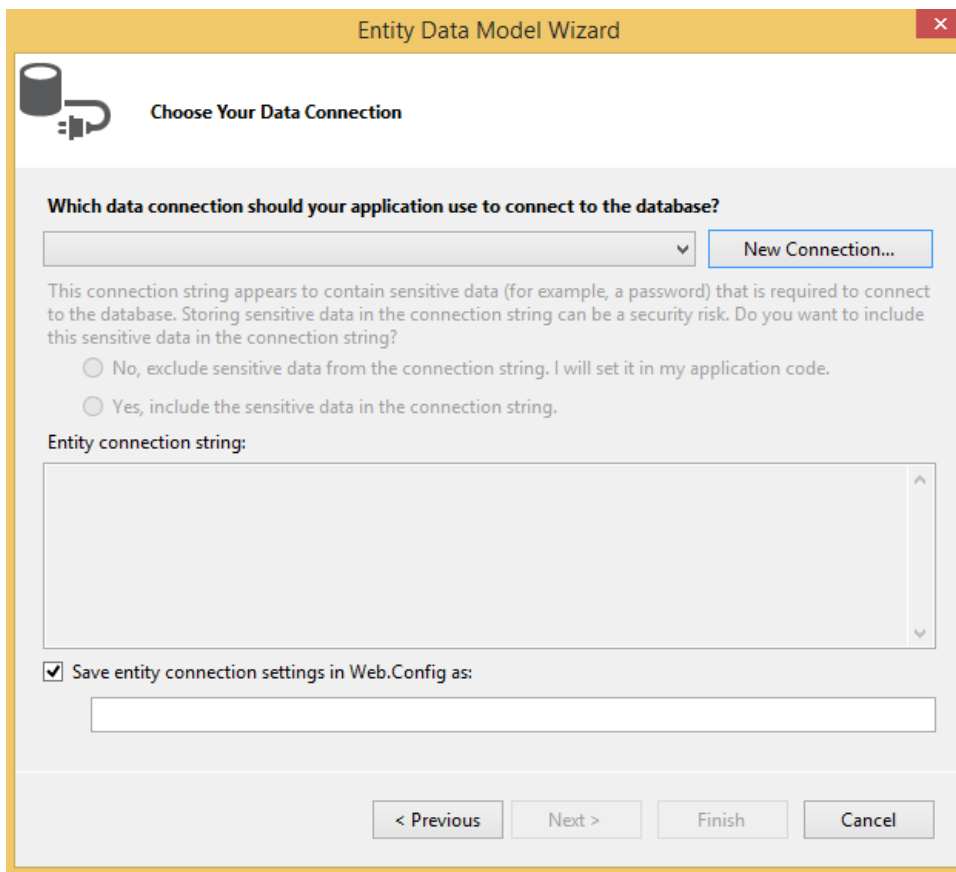
Configure the project to use MySQL with an Entity Framework. There are two ways to do this:

- 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 [Chapter 6, MySQL Website Configuration Tool](#).

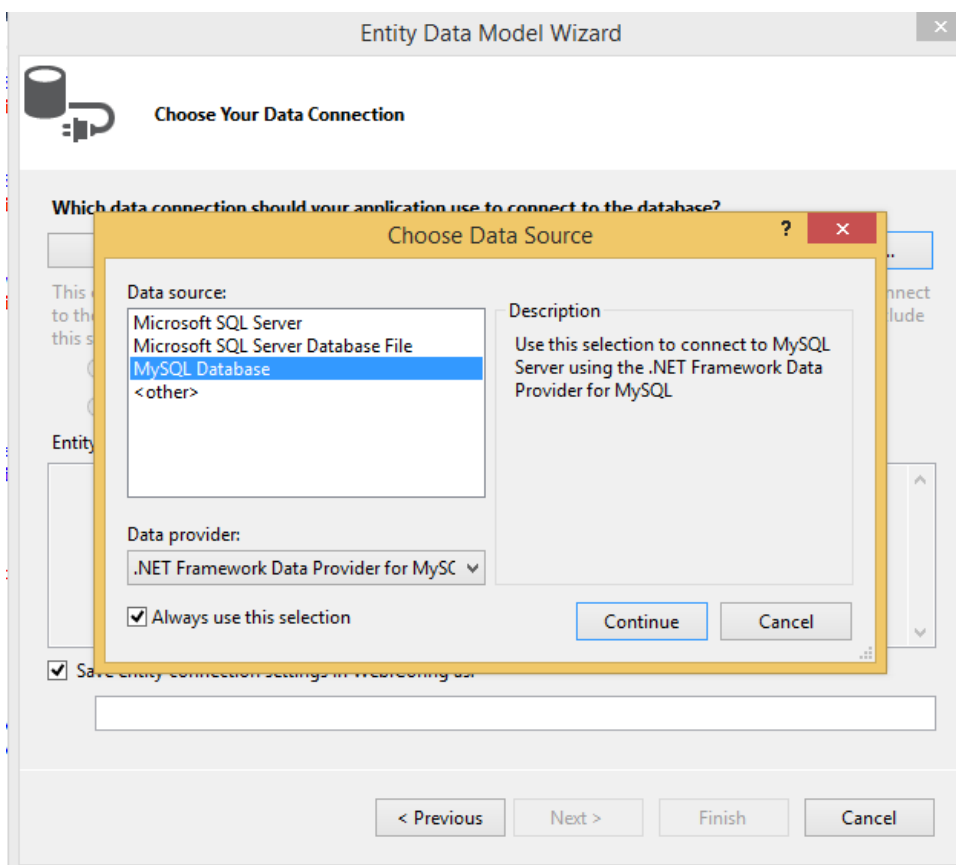
Once you have configured the project to use MySQL with Entity Framework, proceed to create the model using the standard **ADO.NET Entity Data Model** wizard. For MySQL MVC Item Templates, you need to add the model under the "Models" folder, as illustrated below:

**Figure 7.1 ADO.NET Entity Data Model**



**Figure 7.2 Choose or create a new MySQL connection**

The dialog box is titled "Entity Data Model Wizard" and "Choose Your Data Connection". It features a database icon and a question: "Which data connection should your application use to connect to the database?". Below this is a dropdown menu and a "New Connection..." button. A text block explains that connection strings may contain sensitive data like passwords and asks if the user wants to include it. Two radio buttons are provided: "No, exclude sensitive data from the connection string. I will set it in my application code." and "Yes, include the sensitive data in the connection string.". A large text area labeled "Entity connection string:" is present. At the bottom, there is a checkbox "Save entity connection settings in Web.Config as:" followed by a text input field. Navigation buttons "< Previous", "Next >", "Finish", and "Cancel" are at the bottom right.

**Figure 7.3 Creating a new MySQL connection**

This figure shows the same "Entity Data Model Wizard" dialog box as Figure 7.2, but with a "Choose Data Source" sub-dialog box open. The sub-dialog has a title bar with a question mark and a close button. It contains a "Data source:" list with options: "Microsoft SQL Server", "Microsoft SQL Server Database File", "MySQL Database" (which is selected), and "<other>". To the right of this list is a "Description" text area that reads: "Use this selection to connect to MySQL Server using the .NET Framework Data Provider for MySQL". Below the list is a "Data provider:" dropdown menu showing ".NET Framework Data Provider for MySC". At the bottom of the sub-dialog are checkboxes "Always use this selection" and buttons "Continue" and "Cancel". The background dialog box is partially visible and dimmed.

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 7.4 Selecting the database object to include in the model

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- ☒ Tables
- ☐ Views
- ☐ Stored Procedures and Functions

☐ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

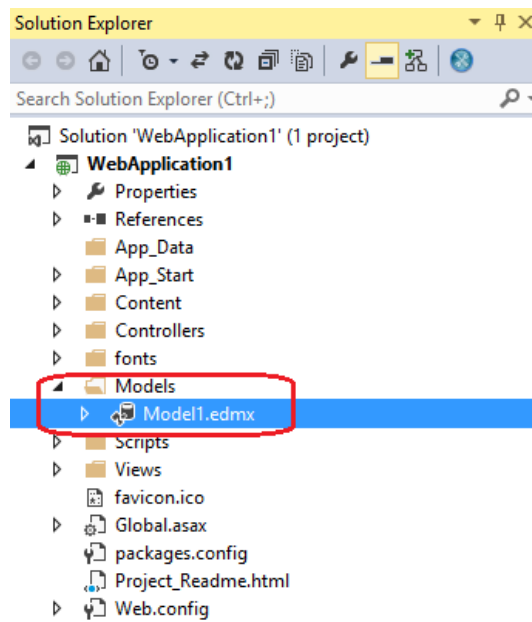
☒ Import selected stored procedures and functions into the entity model

Model Namespace:

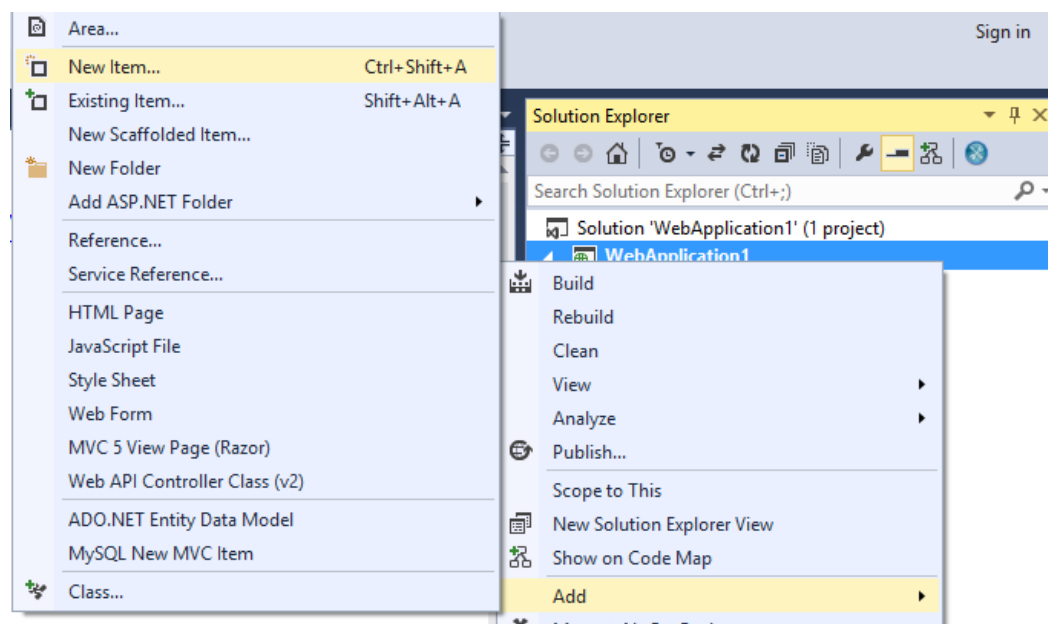
sakilaModel

< Previous   Next >   Finish   Cancel

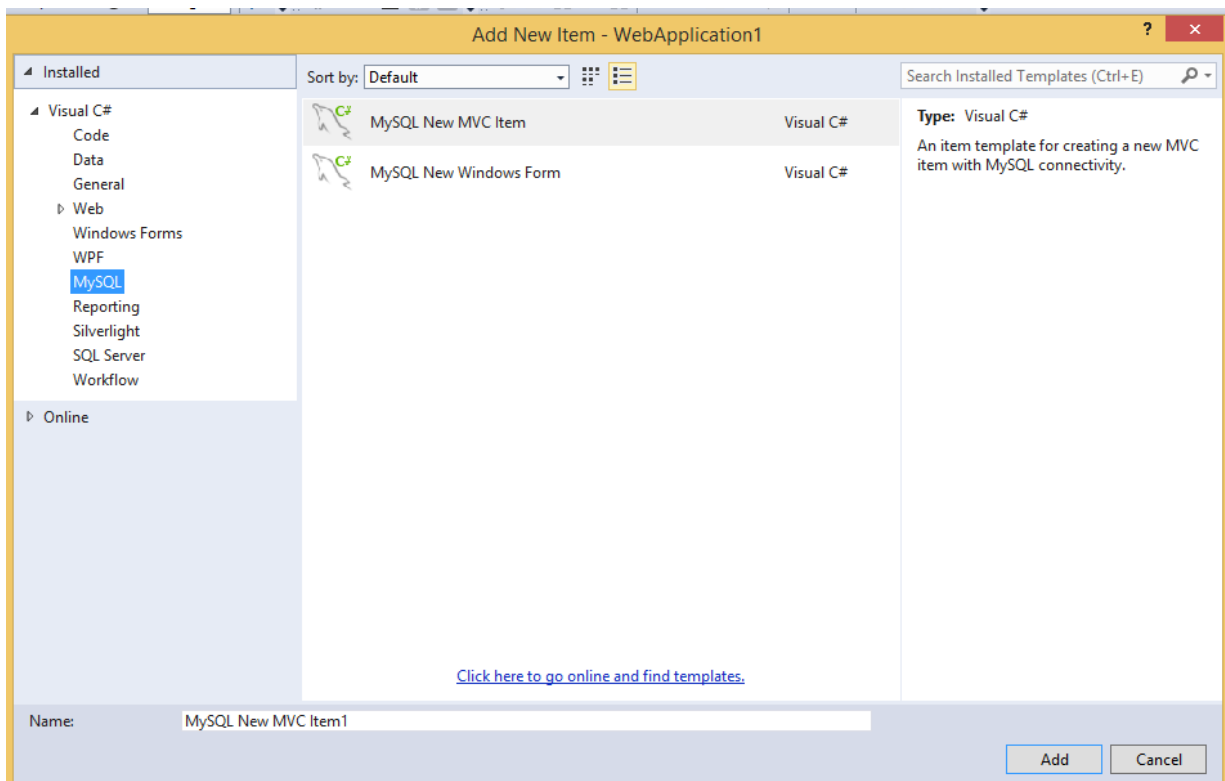
Click **Finish** to generate the model, as demonstrated below:

**Figure 7.5 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 7.6 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 7.7 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.

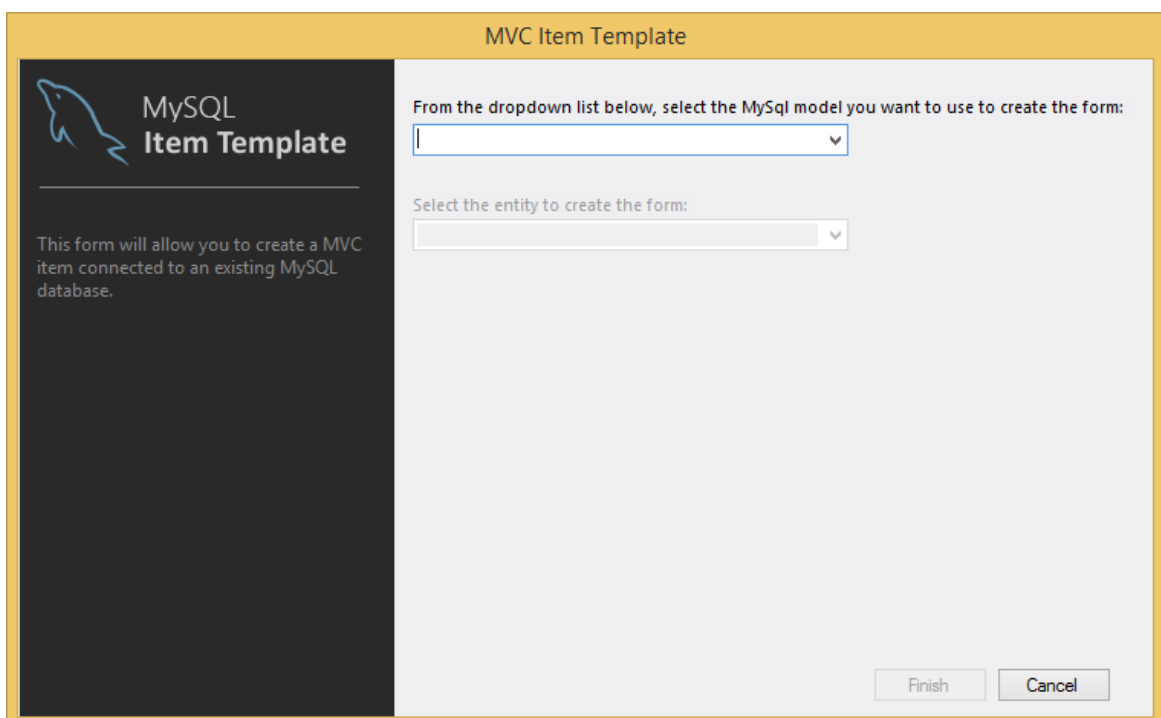
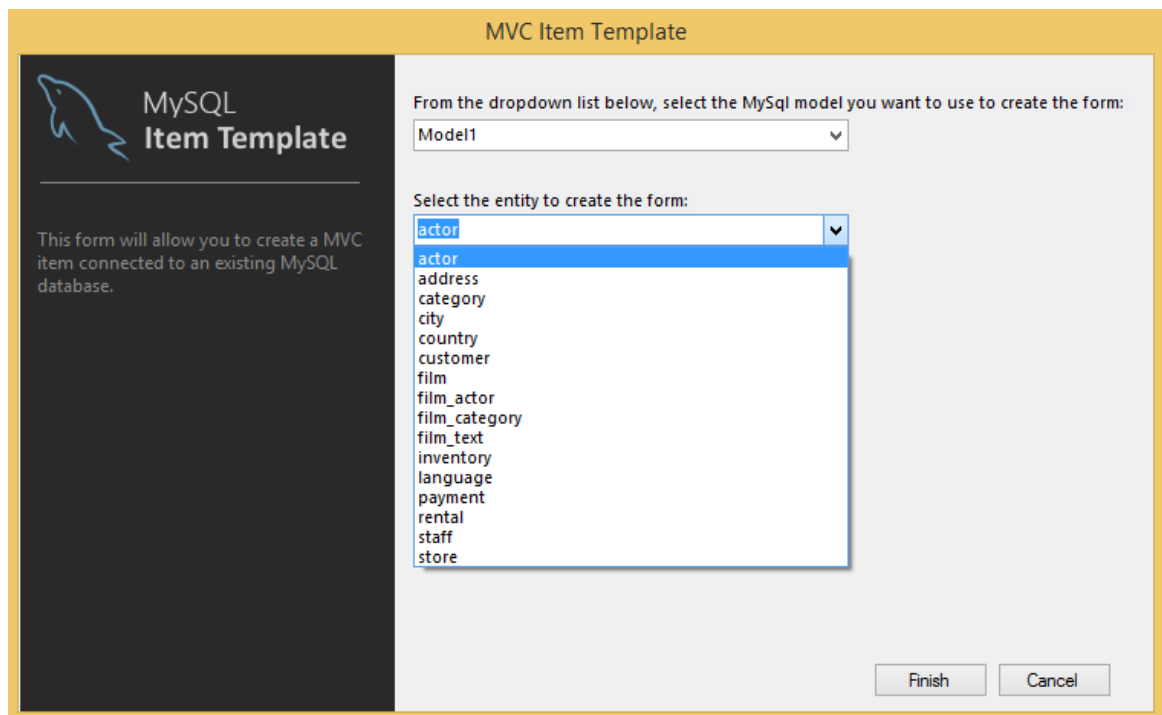
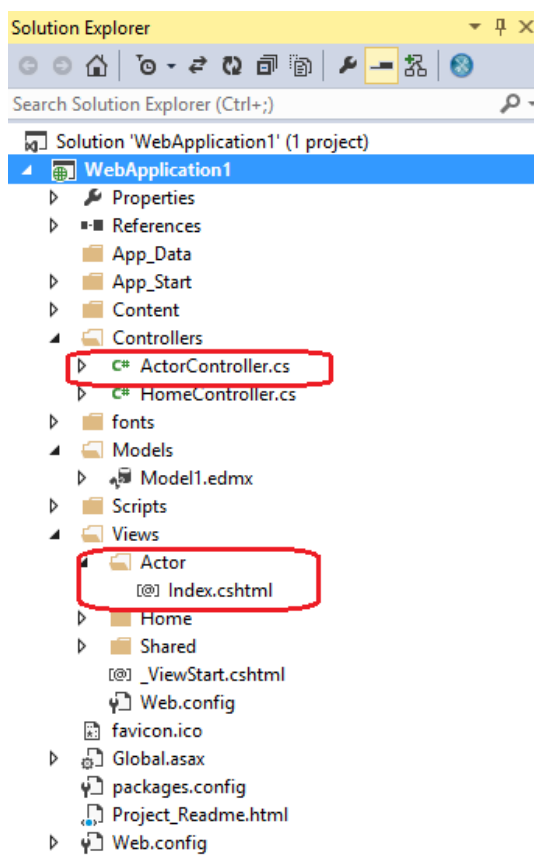
**Figure 7.8 MySQL MVC Item Template Dialog**

Figure 7.9 MySQL MVC Item Template



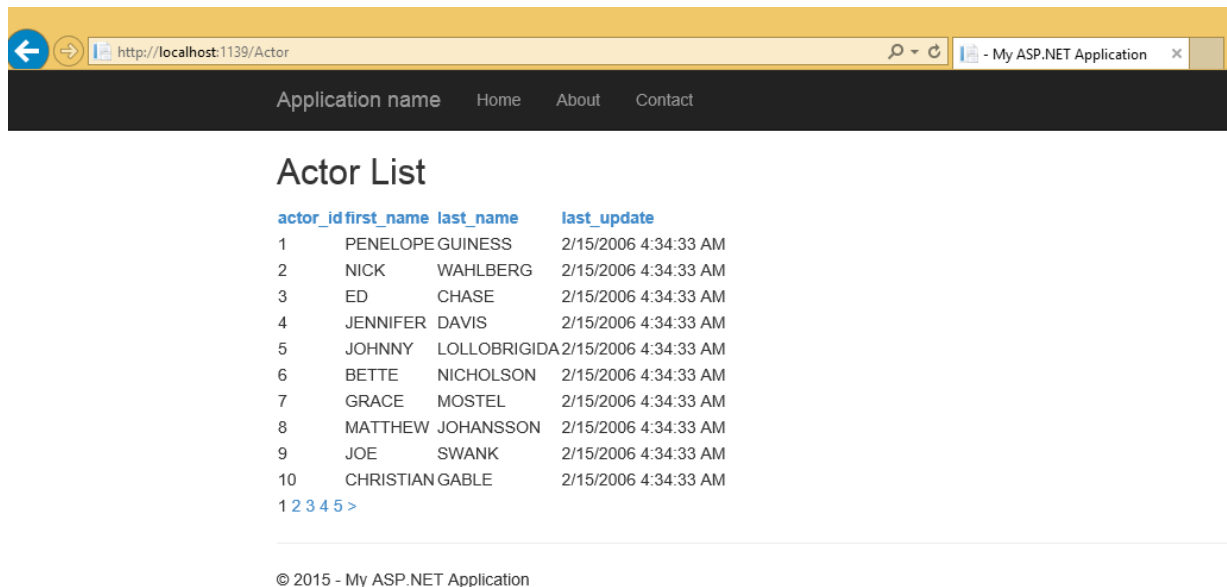
After selecting the model and entity to create the item, click **Finish**, and a new controller and view matching the selected entity will be added to the project. These contain the necessary back end code to render the *entity* data.

Figure 7.10 New controller and view added to the project



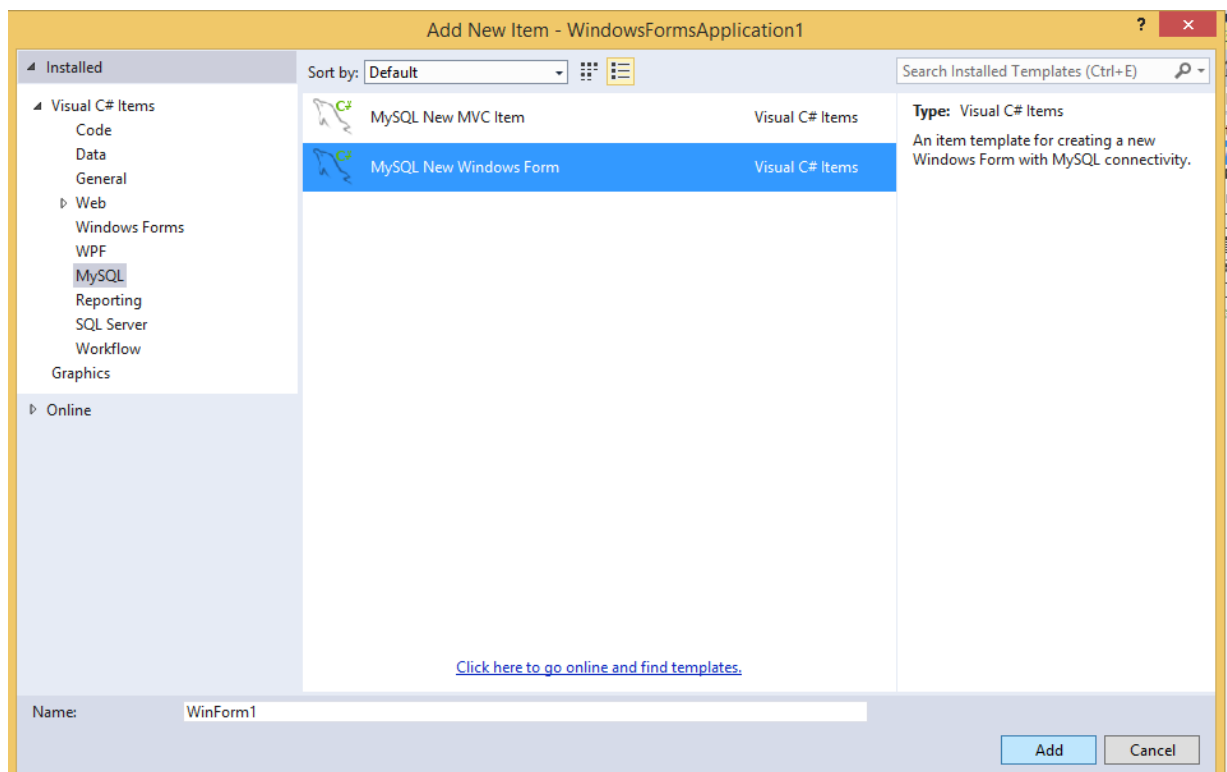
You can now execute the application. In our example we used the Sakila database and generated an Actor controller:

**Figure 7.11 The Actor View**



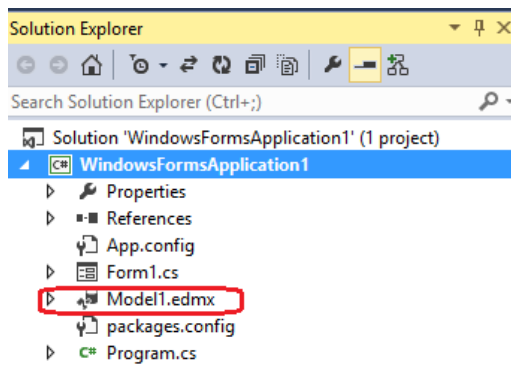
## 7.3 MySQL Windows Forms Items

**Figure 7.12 MySQL New Windows Form**

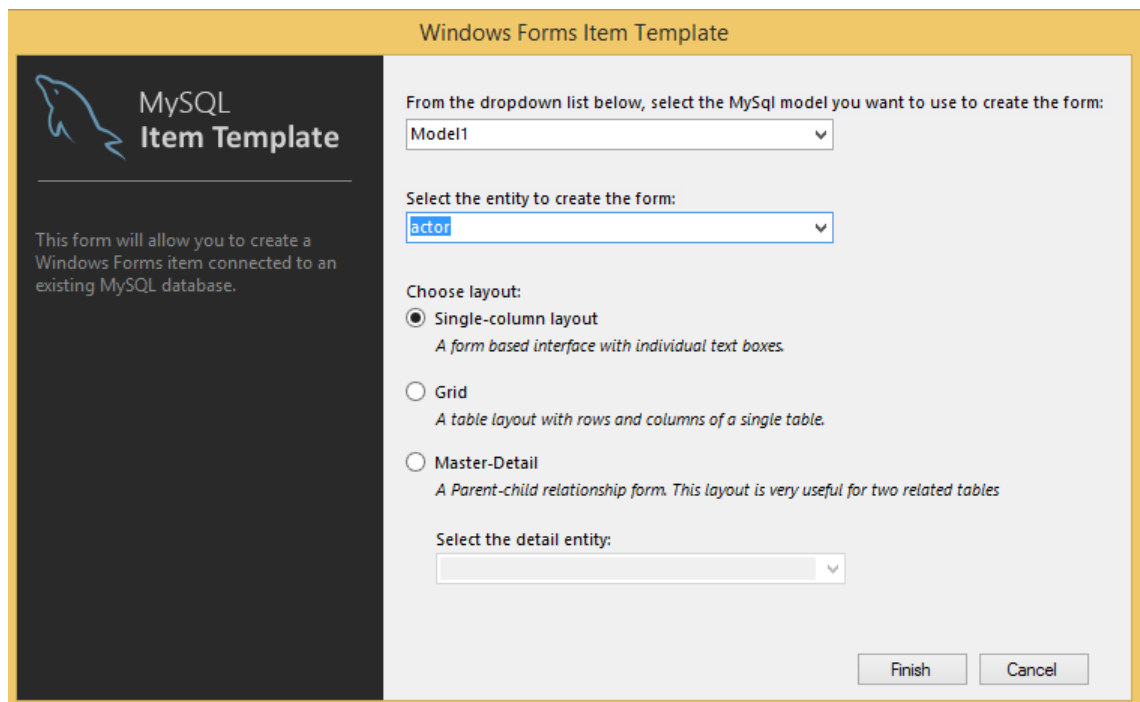


The procedure to add a MySQL Windows Forms Item is basically similar to adding a MySQL MVC Item, with three major differences:

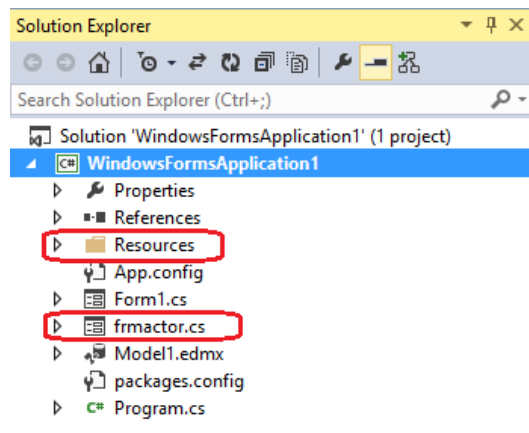
- You can create the MySQL Entity Framework model under the root path of the project.

**Figure 7.13 A MySQL Entity Framework model created in a Windows Form Application**

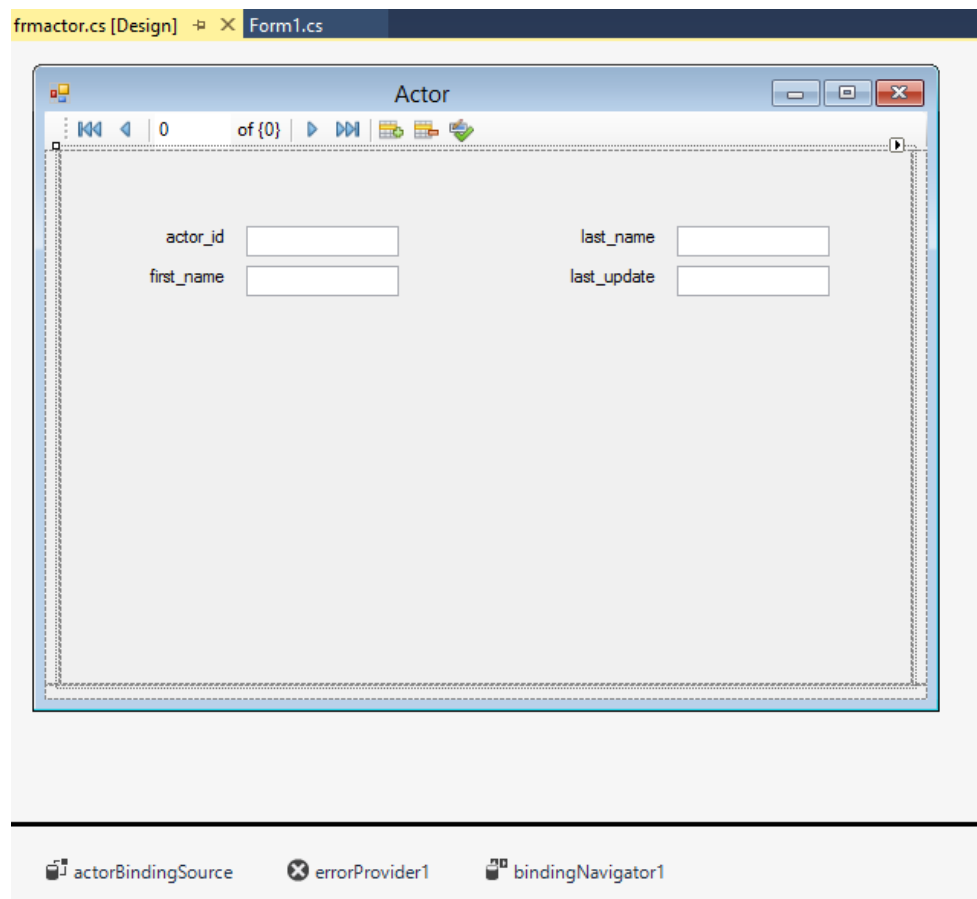
- When selecting the desired entity, you can also select the layout type in which the new form will display the entity data.

**Figure 7.14 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 7.15 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 7.16 The "frmactor" Form in Design Mode**



**Figure 7.17 The "frmactor" Form to Display Data**

The screenshot shows a Windows Forms application window titled "Actor". The window has a yellow title bar with standard minimize, maximize, and close buttons. Below the title bar is a toolbar with navigation icons (back, forward, search, etc.) and a status bar showing "1 of 200". The main area of the form contains four text boxes arranged in a 2x2 grid. The first row contains "actor\_id" with the value "1" and "last\_name" with the value "GUINNESS". The second row contains "first\_name" with the value "PENELOPE" and "last\_update" with the value "2/15/2006 4:34 AM".

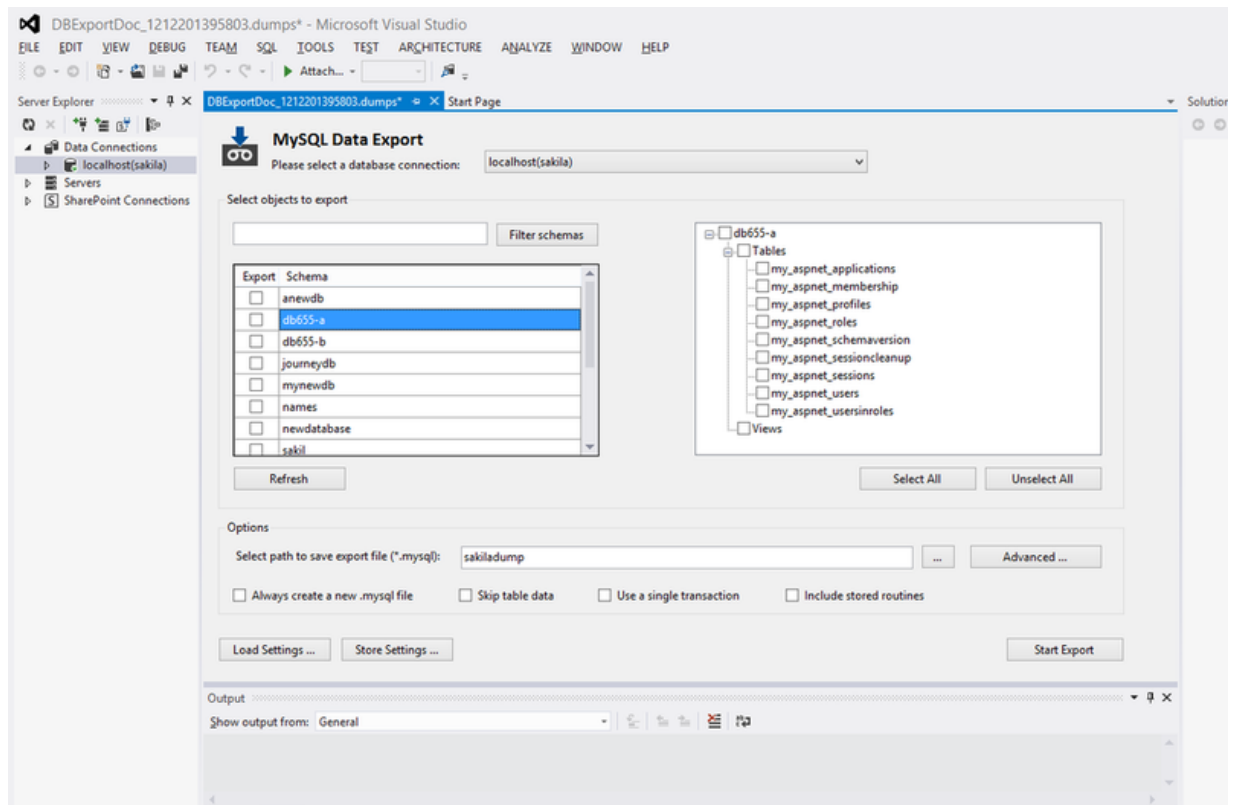
Field	Value
actor_id	1
last_name	GUINNESS
first_name	PENELOPE
last_update	2/15/2006 4:34 AM



## Chapter 8 MySQL Data Export Tool

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

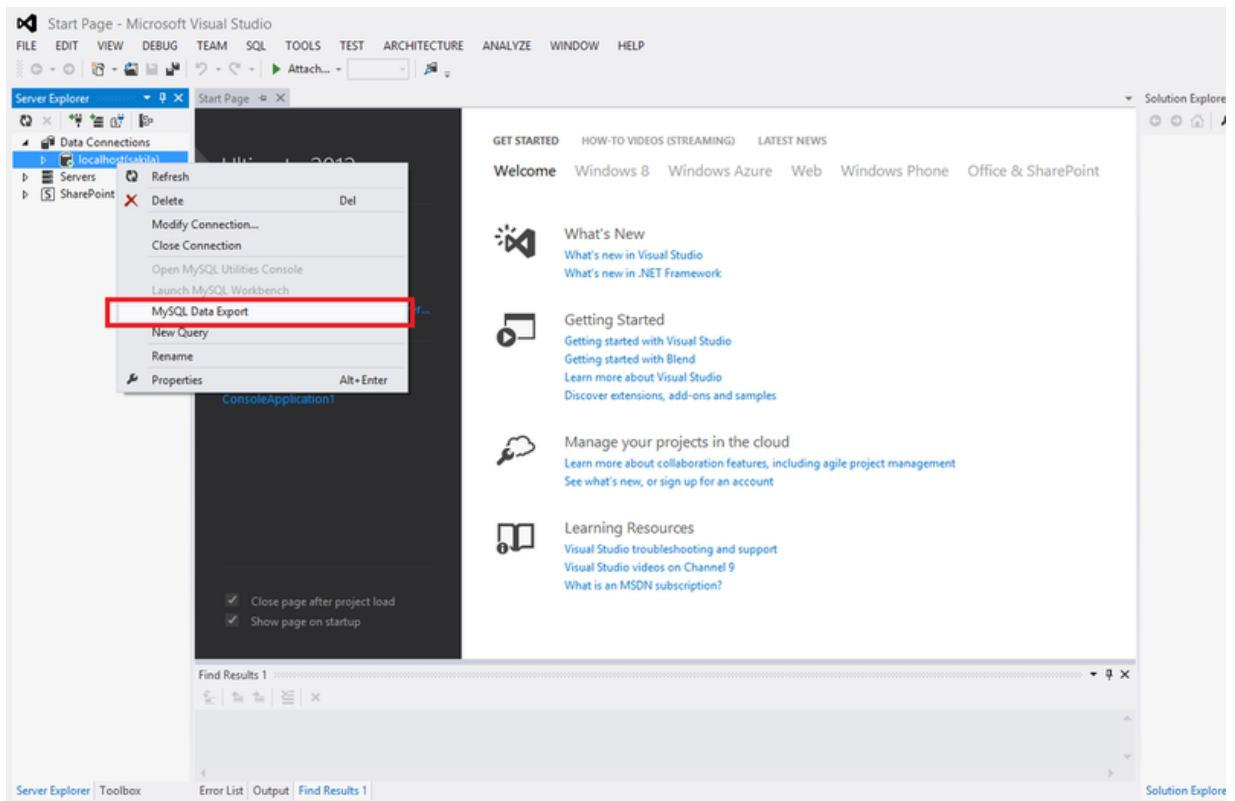
**Figure 8.1 MySQL for Visual Studio Data Export Tool: Main Window**



### Creating a Dump of an existing MySQL Database

In order to open a new window for the MySQL Data Export tool, the user must create a new connection using the **Server Explorer Window** inside Visual Studio. Once 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 tabbed-window opens for the current connection. The user can select one or more databases to include in the dump.

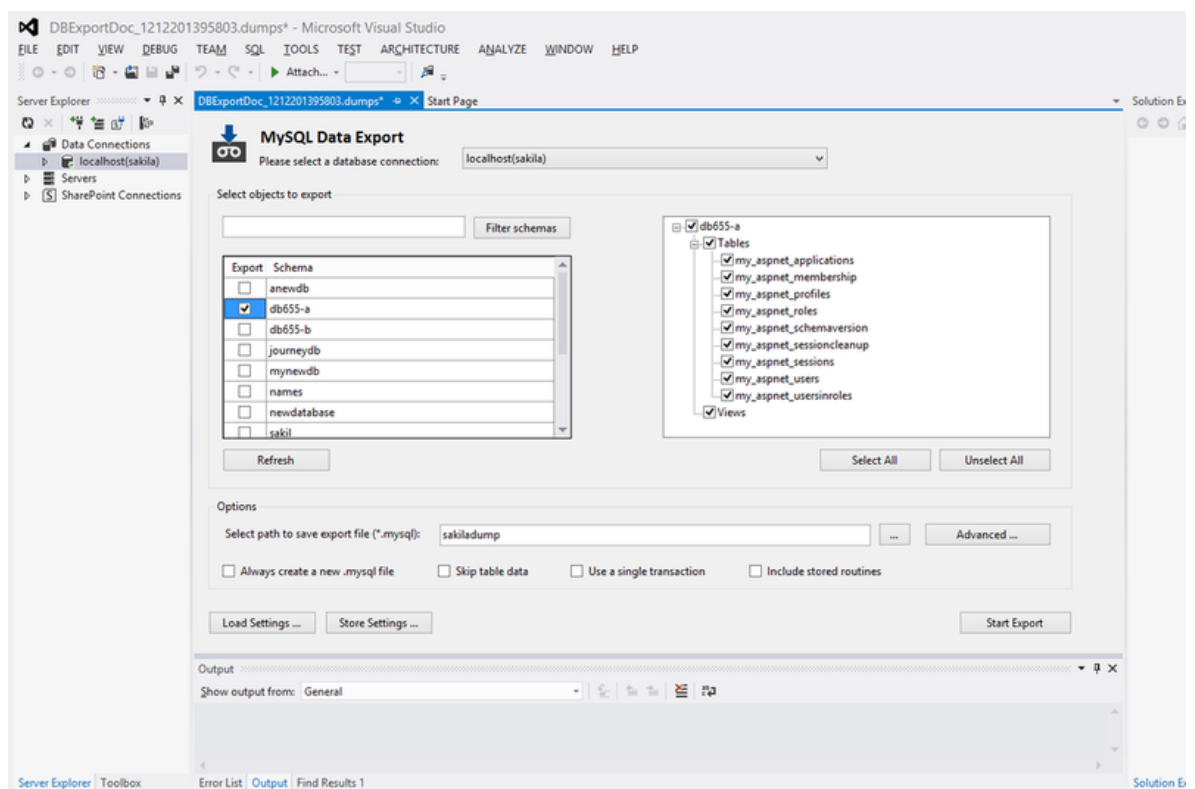
**Figure 8.2 MySQL for Visual Studio Data Export Tool: Connection Context Menu**



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

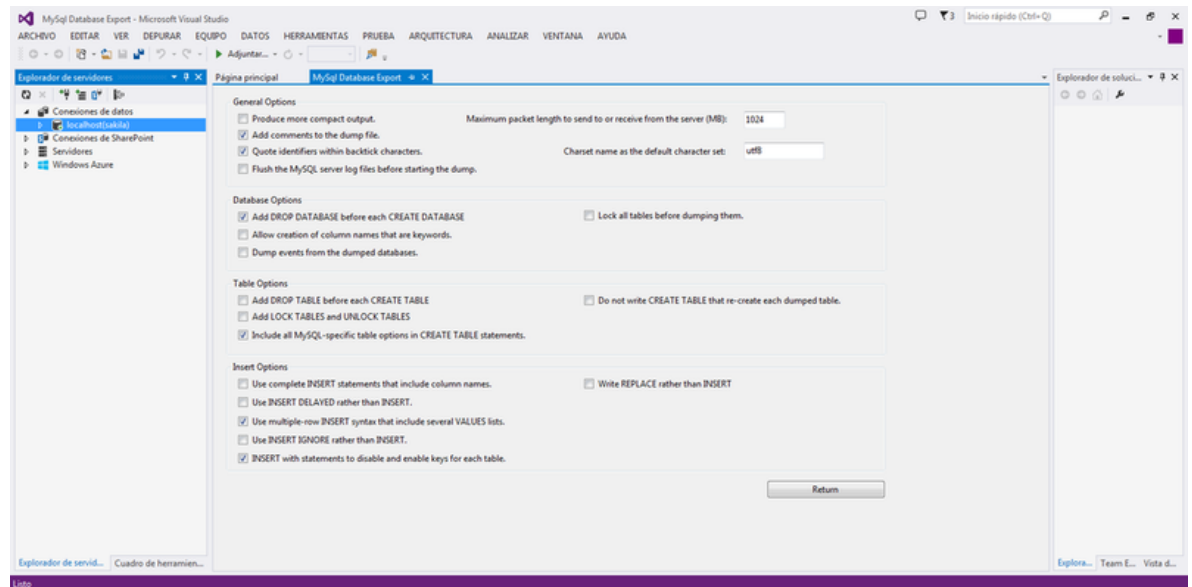
1. Select all the databases and their objects to be included in the dump.

**Figure 8.3 MySQL for Visual Studio Data Export Tool: Selecting a Database**



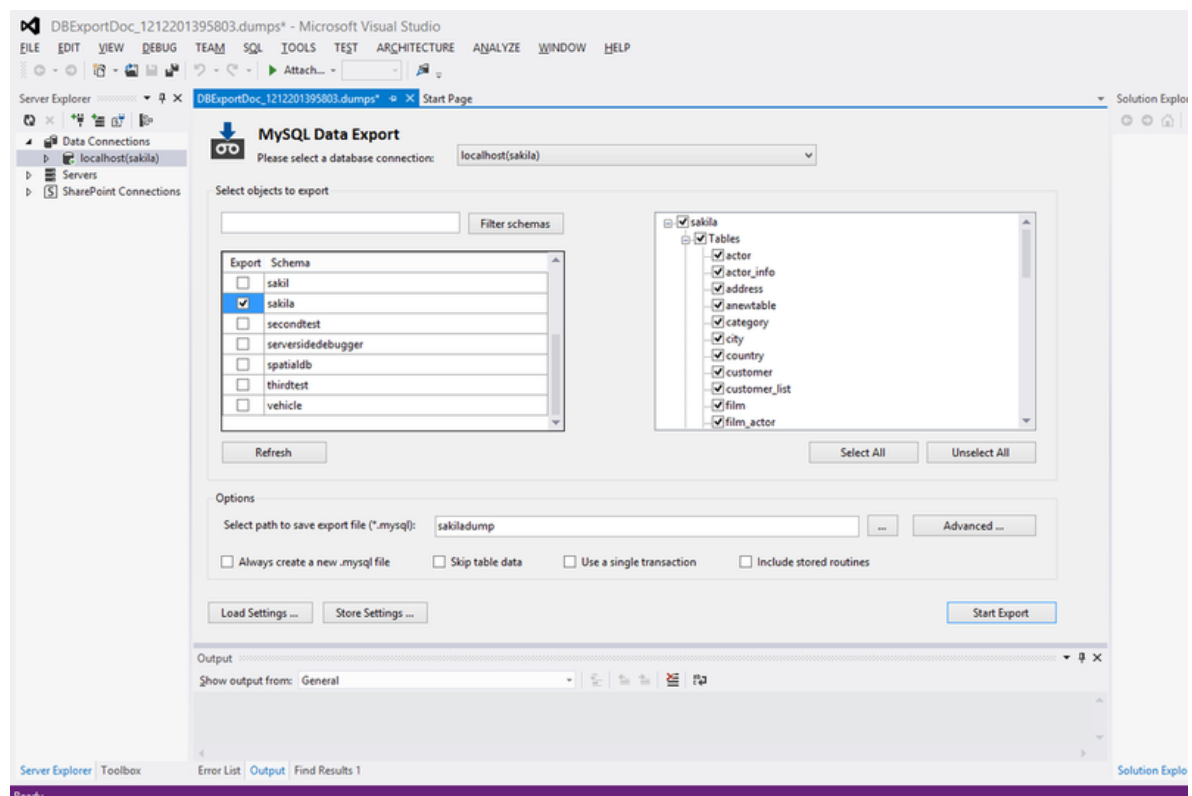
- It is very important to select the desired settings for the dump: whether the dump will include the data, whether the insert operations will be logged in extended mode, and so on. In the main window of the **MySQL Database Export** tool are shown the basic options for the dump. Also, by clicking on the **Advanced** button, more specific options can also be selected.

**Figure 8.4 MySQL for Visual Studio Data Export Tool: Advanced Options**



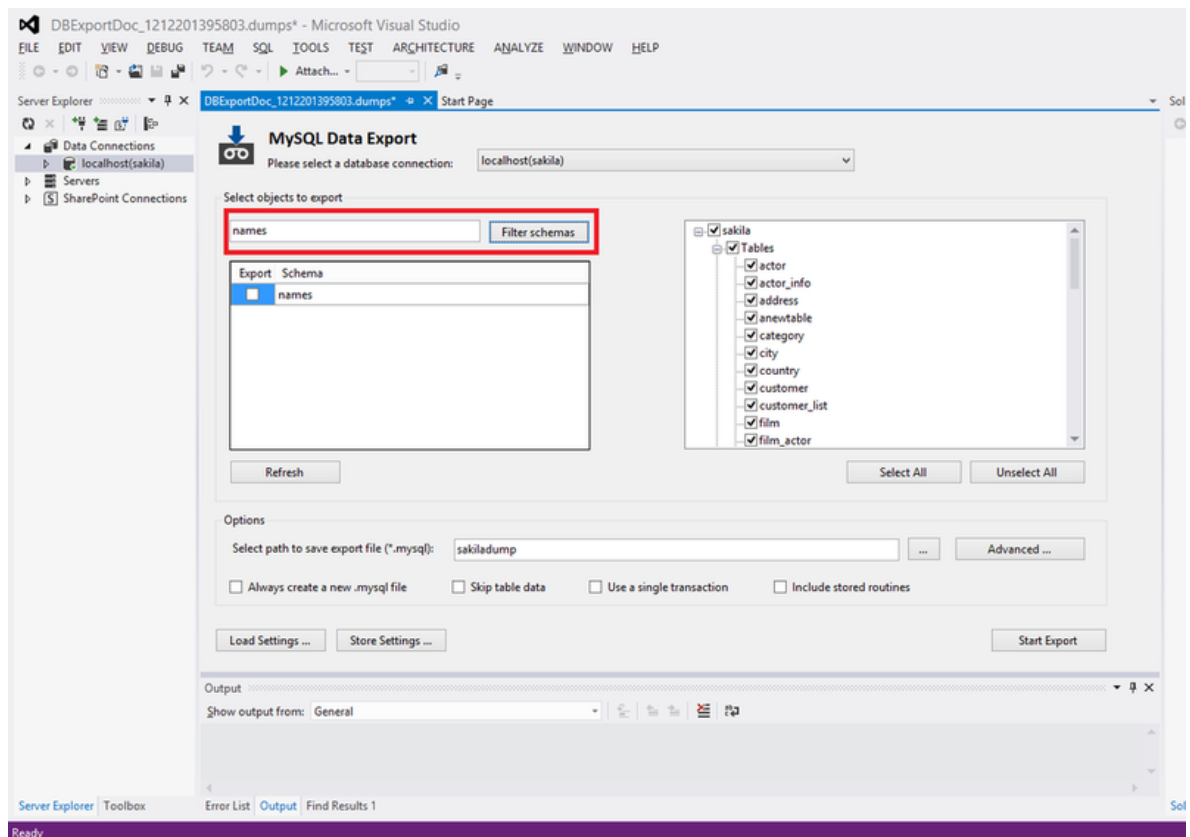
- When the selection of the options is done, give a name to the result file that will be created. If no path is given for the result file, the default path to be used is **My Documents** under the user's folder.

**Figure 8.5 MySQL for Visual Studio Data Export Tool: Generating the Dump File**



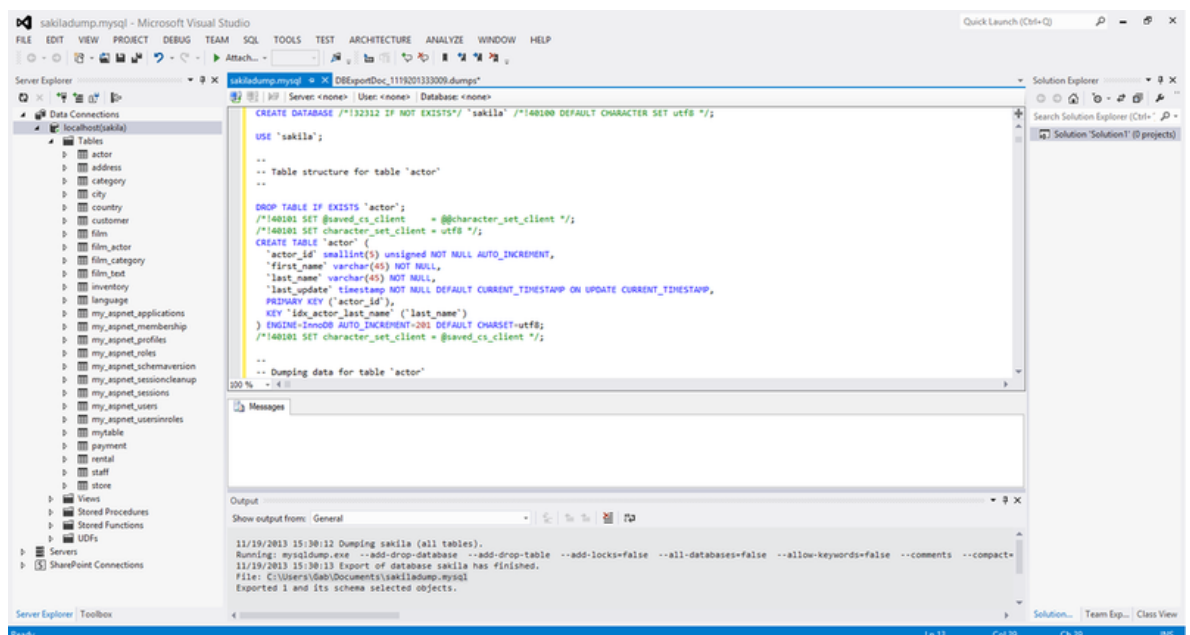
4. A filter can be applied on the list of schemas for the selected connection. With it, the user can easily locate the databases to be included in the dump.

**Figure 8.6 MySQL for Visual Studio Data Export Tool: Filtering the Schemas**



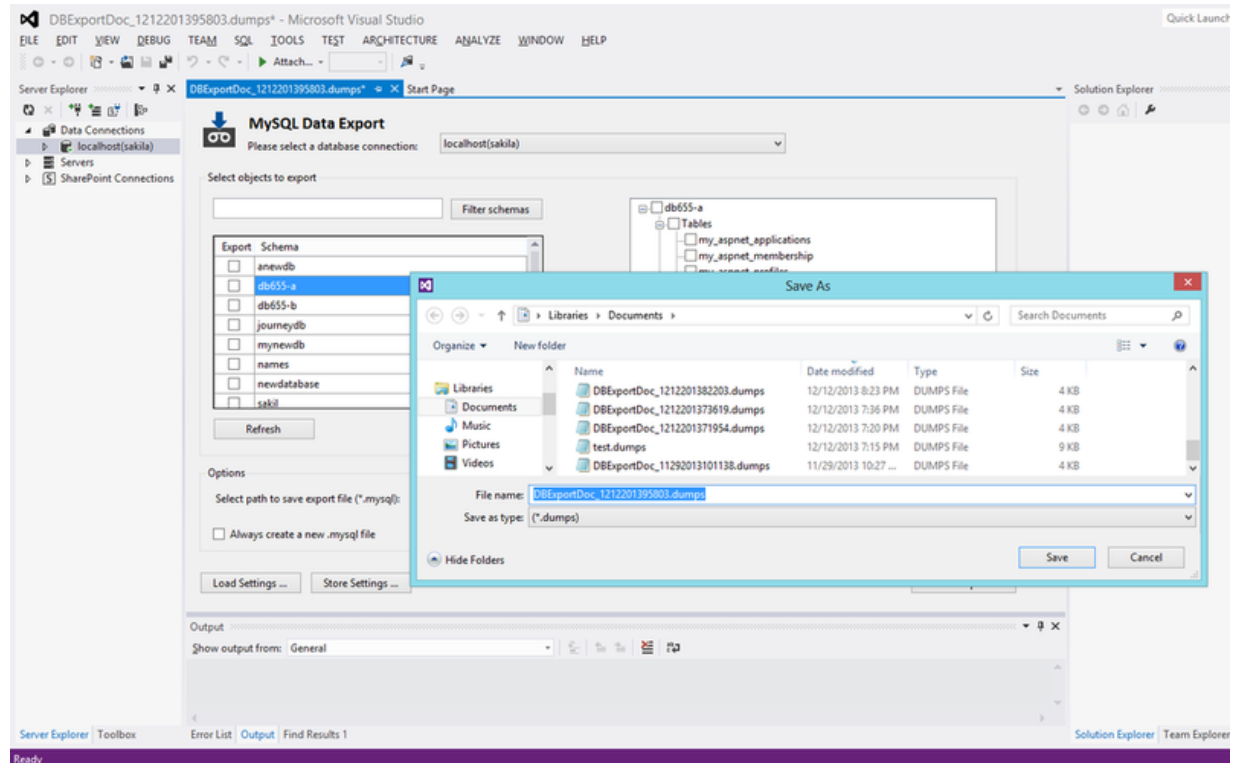
5. After selecting the options and the name for the dump file, the user can click the **Export** button, which generates the dump.

**Figure 8.7 MySQL for Visual Studio Data Export Tool: Viewing the Generated Script**



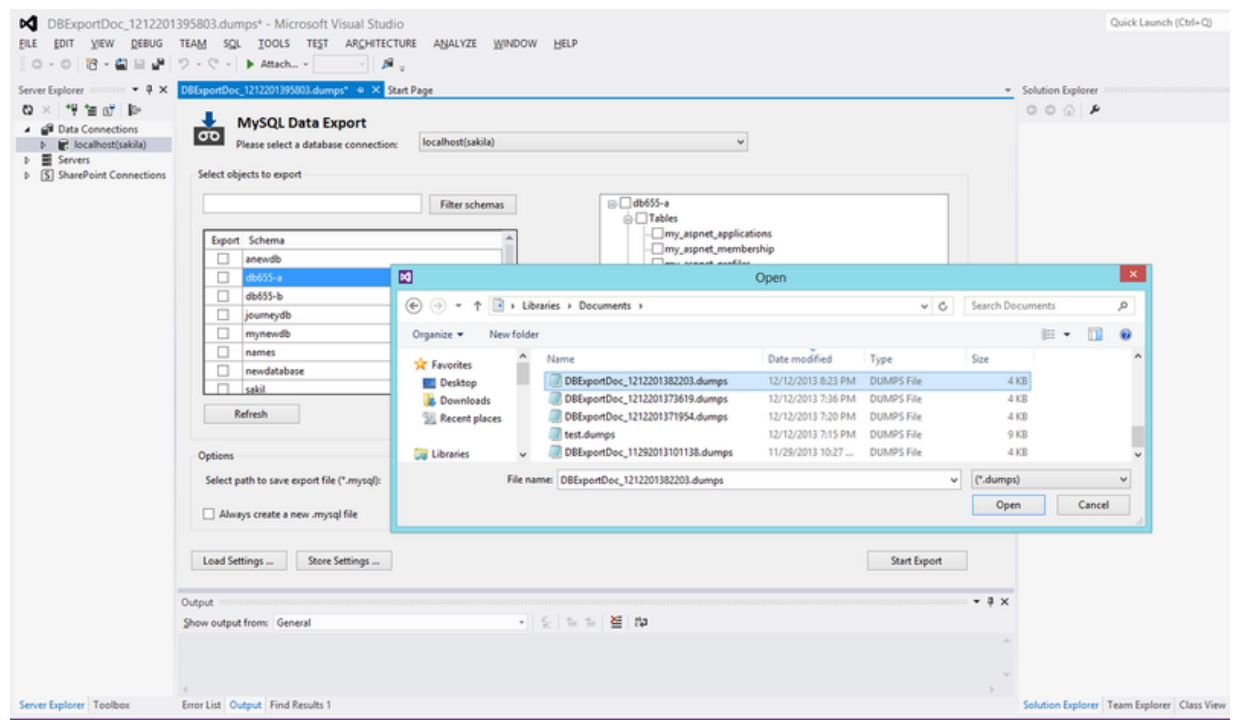
Each dump can have different settings. After configuring the dump operation, the settings can be saved into a setting file for later use. This file includes: the connection selected, the name of the file for the dump, and the database or databases and the objects selected for dumping. The file extension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

**Figure 8.8 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 8.9 MySQL for Visual Studio Data Export Tool: Opening a Setting File**







---

## Chapter 9 Using the ADO.NET Entity Framework

ADO.NET Entity Framework provides an Object Relational Mapping (ORM) service, mapping the relational database schema to objects. The ADO.NET Entity Framework defines several layers, these can be summarized as:

- **Logical** - this layer defines the relational data and is defined by the Store Schema Definition Language (SSDL).
- **Conceptual** - this layer defines the .NET classes and is defined by the Conceptual Schema Definition Language (CSDL)
- **Mapping** - this layer defines the mapping from .NET classes to relational tables and associations, and is defined by Mapping Specification Language (MSL).

MySQL Connector/NET integrates with Visual Studio to provide a range of helpful tools to assist development.

A full treatment of ADO.NET Entity Framework is beyond the scope of this manual. If you are unfamiliar with ADO.NET, review the [Microsoft ADO.NET Entity Framework documentation](#).

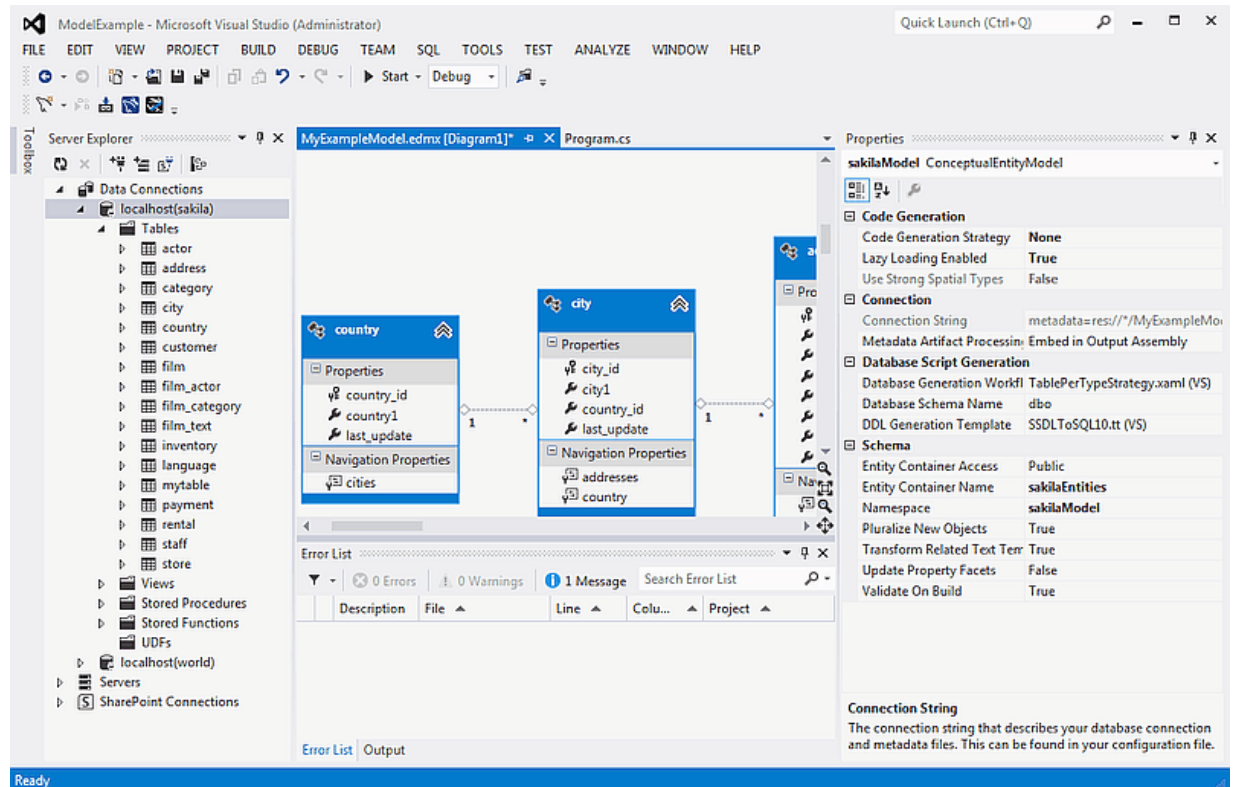
Tutorials on getting started with ADO.NET Entity Framework are available. See [Tutorial: Using an Entity Framework Entity as a Windows Forms Data Source](#) and [Tutorial: Data Binding in ASP.NET Using LINQ on Entities](#).



## Chapter 10 DDL T4 Template Macro

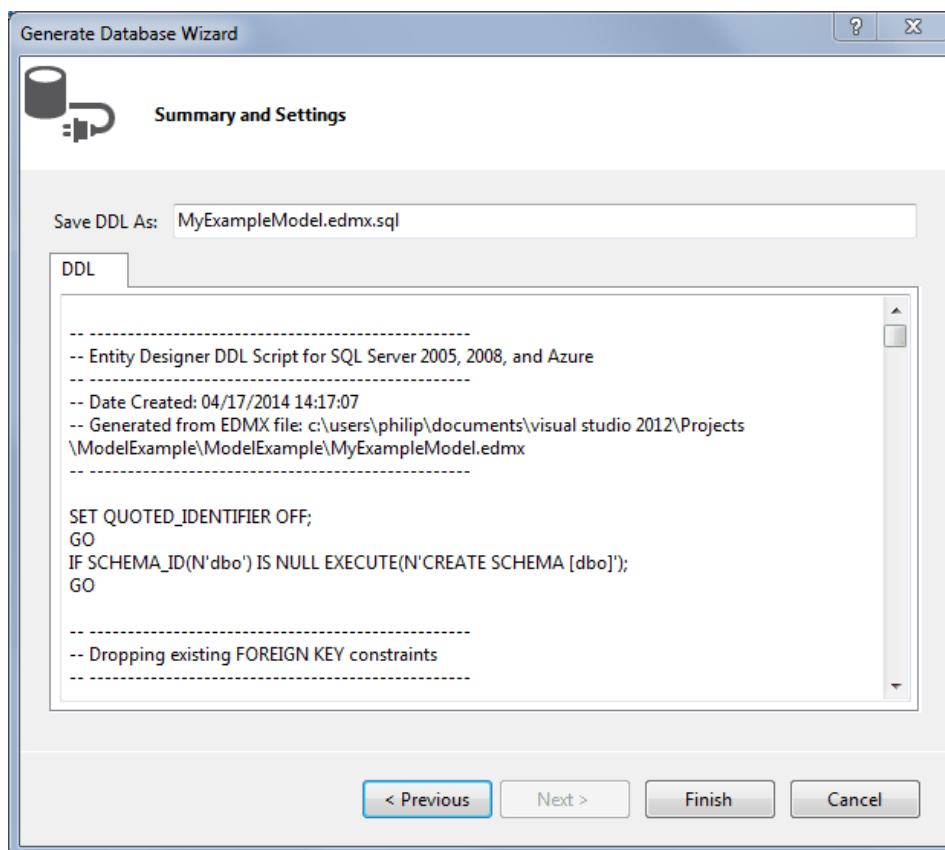
Convert an Entity Framework model to MySQL [DDL](#) code. Starting with a blank model, you can develop an entity model in Visual Studio's designer. Once the model is created, you can select the model's properties, and in the Database Script Generation category of the model's properties, the property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

Figure 10.1 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 10.2 DDL T4 Template Macro - Generate Database Wizard**



# Chapter 11 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.

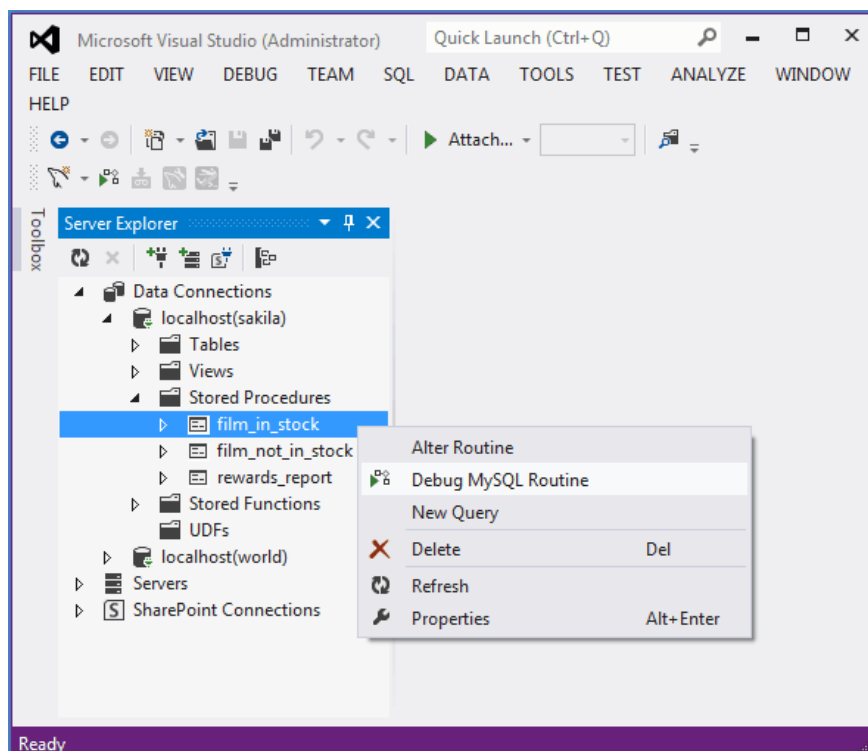
The debugger makes changes behind the scenes to temporarily add instrumentation code to the stored routines that you debug. You must have the `ALTER ROUTINE` privilege for each stored procedure, function, or trigger that you debug. (Including procedures and functions that are called, and triggers that 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 11.1 Choose a Stored Routine to Debug**

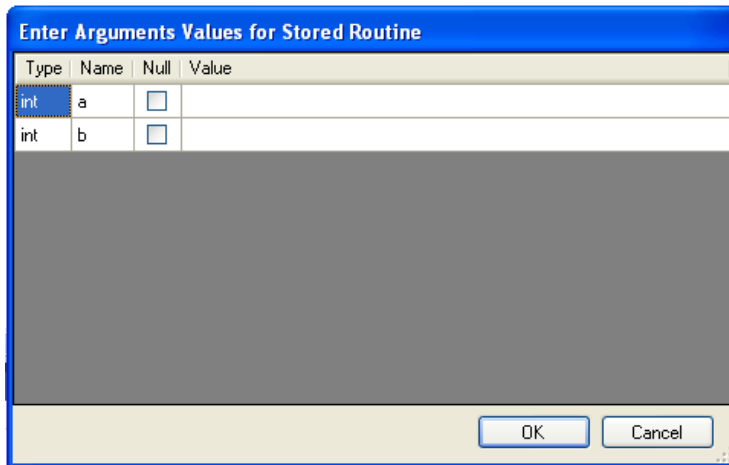


## Usage

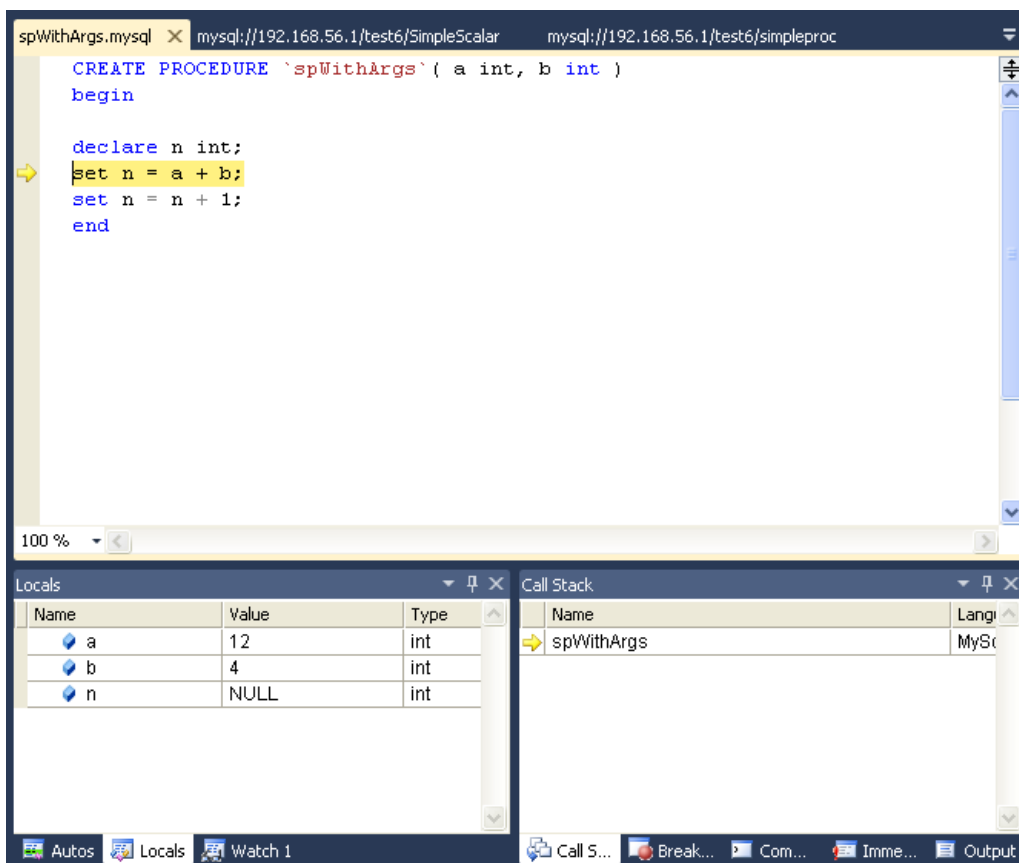
At this point, Visual Studio switches to debug mode, opening the source code of the routine being debugged in step mode, positioned on the first statement.

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 checkbox). After setting up all the argument values, you can press **OK** to start the debug session, or **Cancel** to cancel the debug session.

**Figure 11.2 Setting Arguments (1 of 2)**



**Figure 11.3 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**.

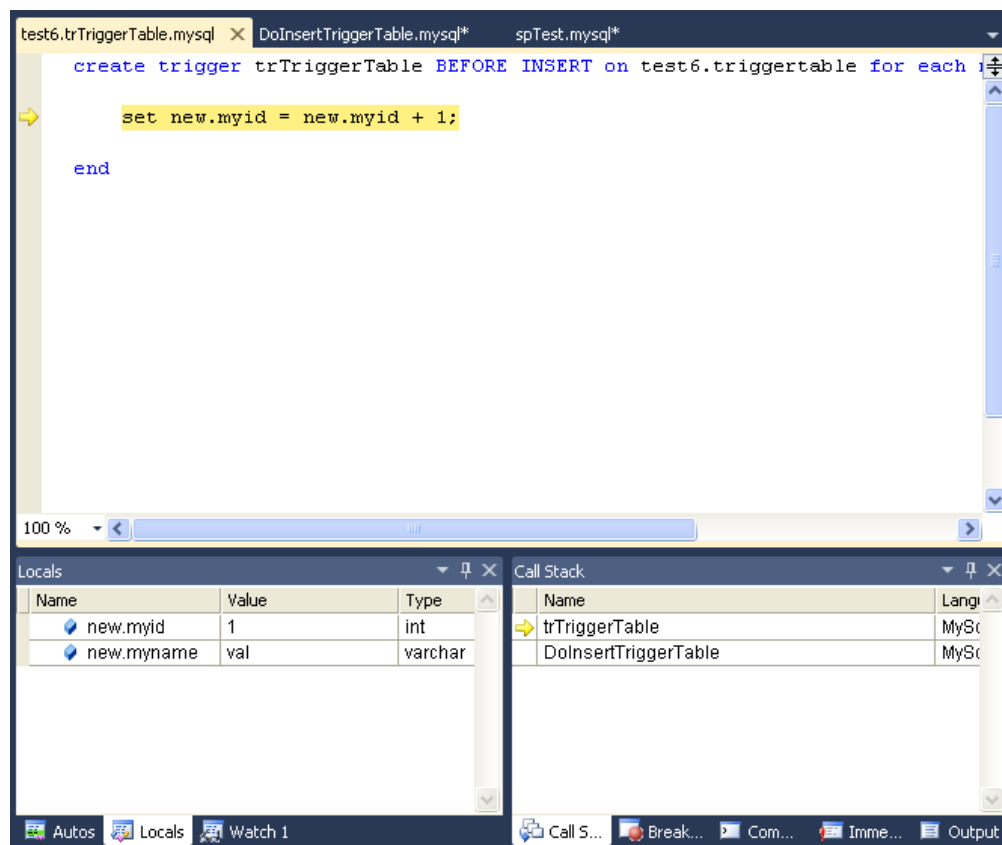
To define a watch, type any valid MySQL expression, optionally including function calls. If the watch evaluation makes sense in the current context (current stack frame), it will show its value, otherwise it will show an error message in the same row the watch was defined.

- 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.

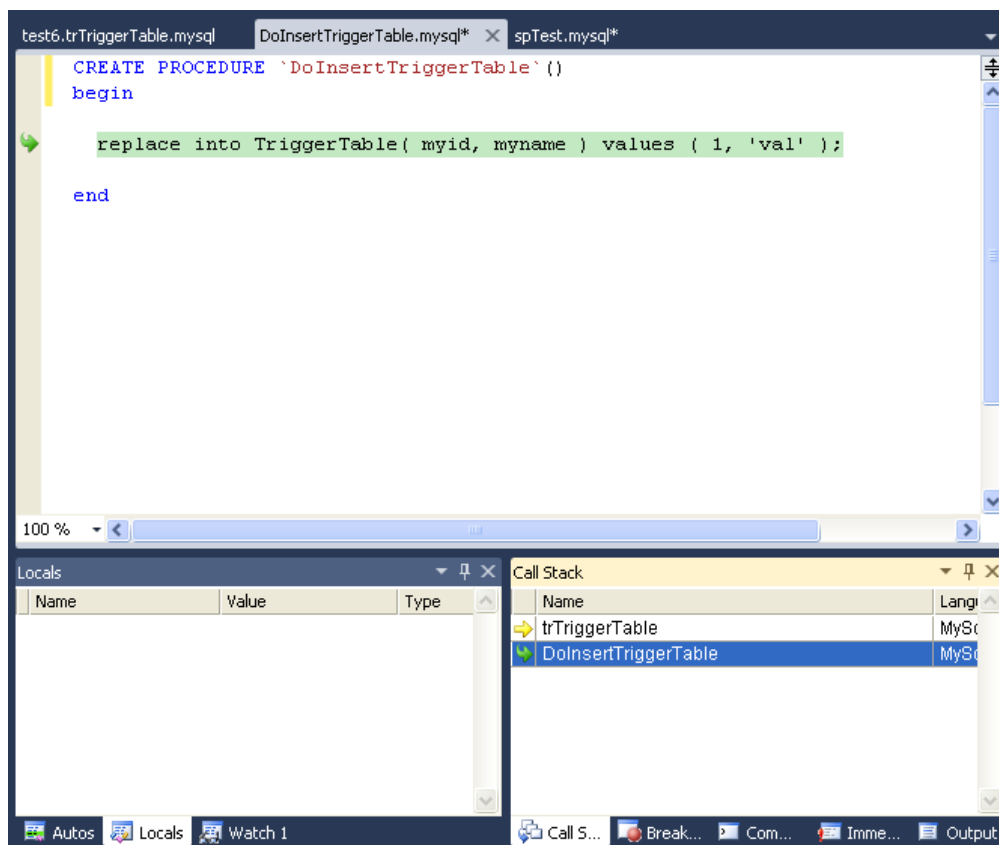
Figure 11.4 Debugging a Trigger



### 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 11.5 Call Stack**

### Stepping

- Stepping of a new routine starts in the first executable instruction (excluding declares, handlers, cursor declarations, and so on).

Figure 11.6 Debug Stepping

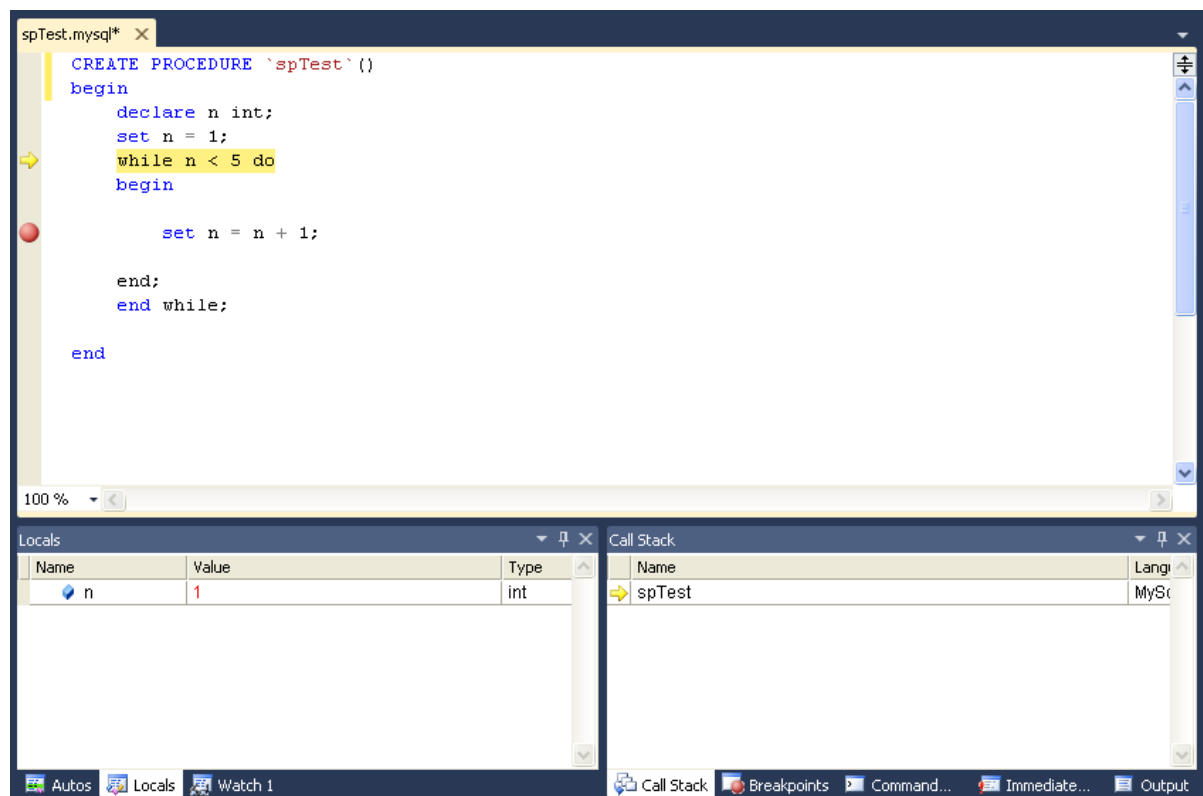


Figure 11.7 Function Stepping (1 of 2)

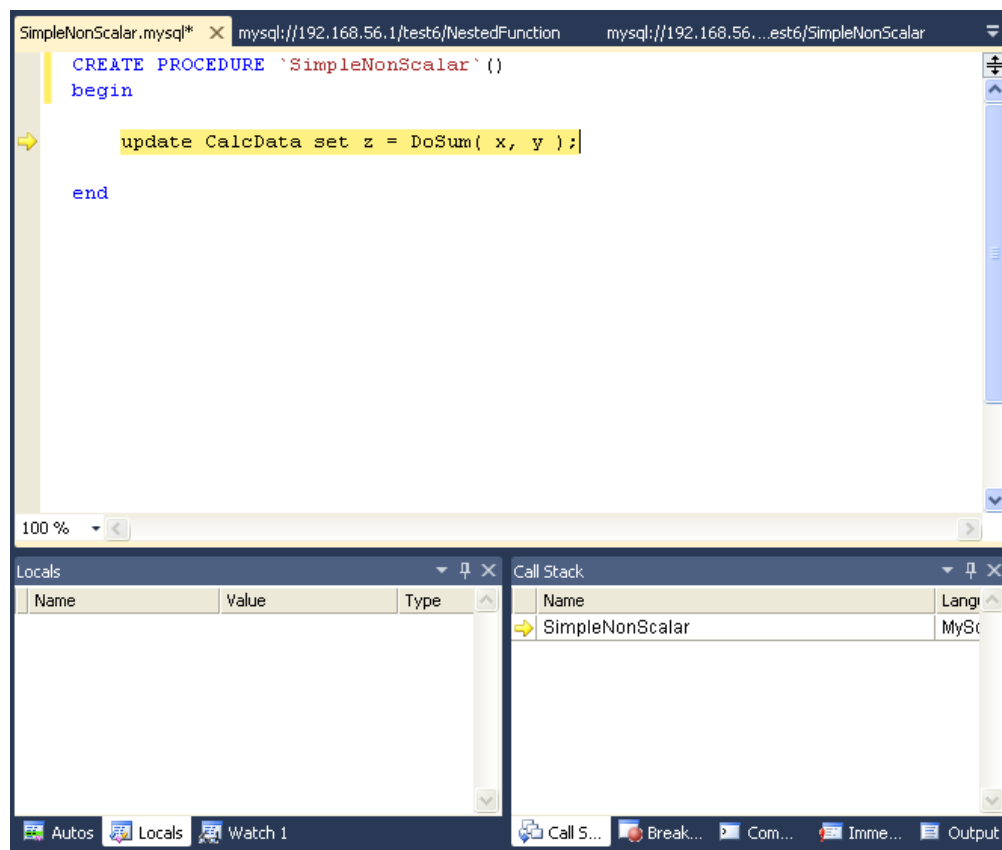
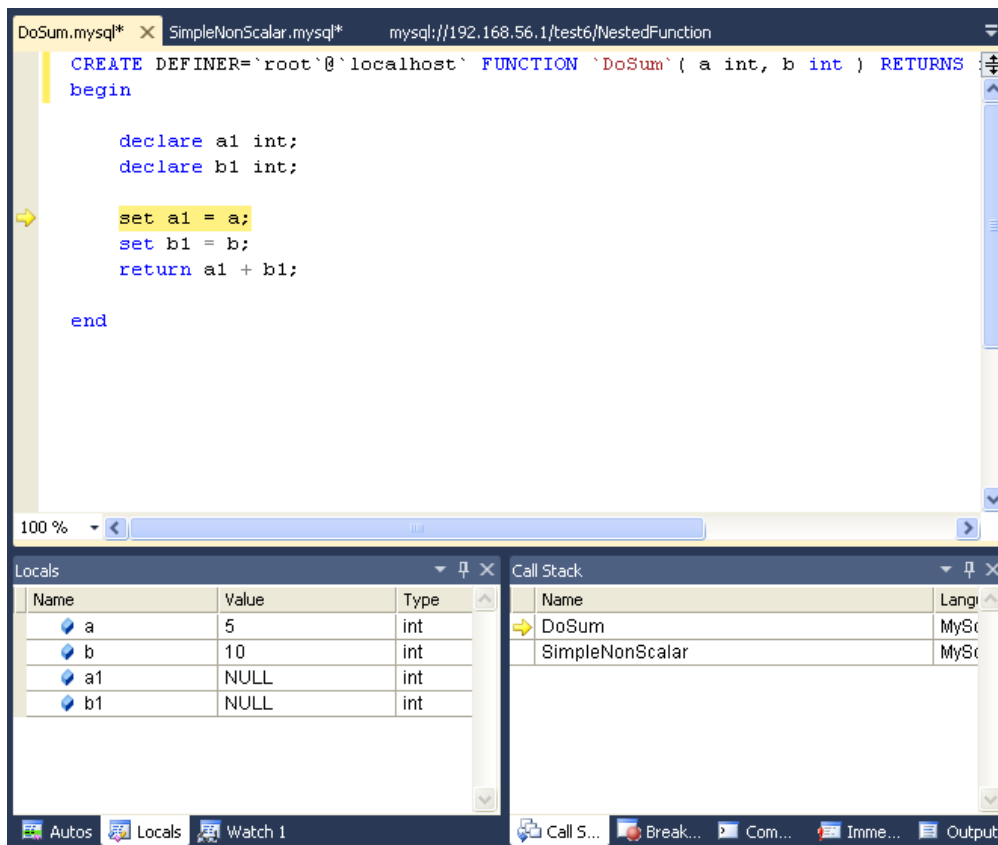


Figure 11.8 Function Stepping (2 of 2)



- 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 11.9 Conditional Breakpoints

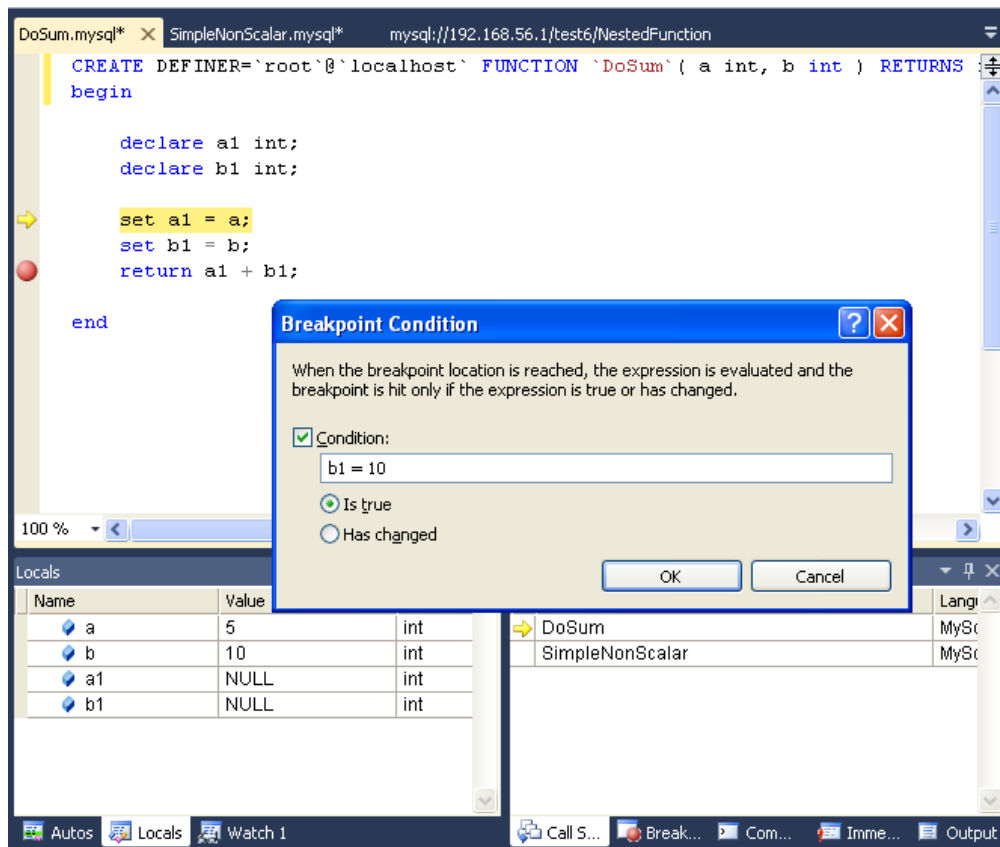
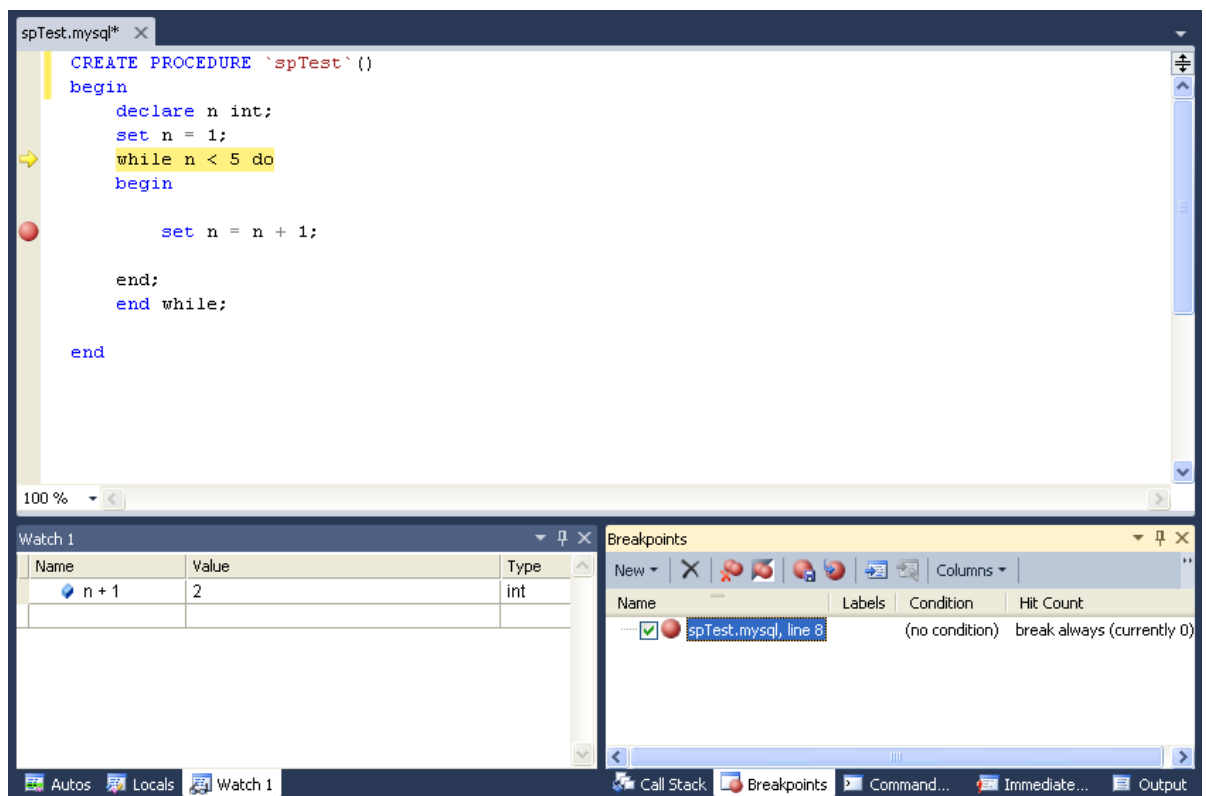


Figure 11.10 Expressions and Breakpoints



- 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

---

# Appendix A MySQL for Visual Studio Frequently Asked Questions

## Questions

- [A.1](#): How do I know if MySQL for Visual Studio is installed?

## Questions and Answers

### A.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.

