EntityClient Provider for the Entity Framework

.NET Framework (current version)

The EntityClient provider is a data provider used by Entity Framework applications to access data described in a conceptual model. For information about conceptual models, see Modeling and Mapping. EntityClient uses other .NET Framework data providers to access the data source. For example, EntityClient uses the .NET Framework Data Provider for SQL Server (SqlClient) when accessing a SQL Server database. For information about the SqlClient provider, see SqlClient for the Entity Framework. The EntityClient provider is implemented in the System.Data.EntityClient namespace.

Managing Connections

The Entity Framework builds on top of storage-specific ADO.NET data providers by providing an EntityConnection to an underlying data provider and relational database. To construct an EntityConnection object, you have to reference a set of metadata that contains the necessary models and mapping, and also a storage-specific data provider name and connection string. After the EntityConnection is in place, entities can be accessed through the classes generated from the conceptual model.

You can specify a connection string in app.config file.

The System.Data.EntityClient also includes the EntityConnectionStringBuilder class. This class enables developers to programmatically create syntactically correct connection strings, and parse and rebuild existing connection strings, by using properties and methods of the class. For more information, see How to: Build an EntityConnection Connection String.

Creating Queries

The Entity SQL language is a storage-independent dialect of SQL that works directly with conceptual entity schemas and supports Entity Data Model concepts such as inheritance and relationships. The EntityCommand class is used to execute an Entity SQL command against an entity model. When you construct EntityCommand objects, you can specify a stored procedure name or a query text. The Entity Framework works with storage-specific data providers to translate generic Entity SQL into storage-specific queries. For more information about writing Entity SQL queries, see Entity SQL Language.

The following example creates an EntityCommand object and assigns an Entity SQL query text to its EntityCommand.CommandText property. This Entity SQL query requests products ordered by the list price from the conceptual model. The following code has no knowledge of the storage model at all.

```
EntityCommand cmd = conn.CreateCommand();

cmd.CommandText = @" SELECT VALUE p

FROM AdventureWorksEntities.Product AS p

ORDER BY p.ListPrice ";
```

Executing Queries

When a query is executed, it is parsed and converted into a canonical command tree. All subsequent processing is performed on the command tree. The command tree is the means of communication between the System.Data.EntityClient and the underlying .NET Framework data provider, such as System.Data.SqlClient.

The EntityDataReader exposes the results of executing a EntityCommand against a conceptual model. To execute the command that returns the EntityDataReader, call ExecuteReader. The EntityDataReader implements IExtendedDataRecord to describe rich structured results.

Managing Transactions

In the Entity Framework, there are two ways to use transactions: automatic and explicit. Automatic transactions use the System.Transactions namespace, and explicit transactions use the EntityTransaction class.

To update data that is exposed through a conceptual model; see How to: Manage Transactions in the Entity Framework.

In This Section

How to: Navigate Relationships with the Navigate Operator

See Also

Managing Connections and Transactions
ADO.NET Entity Framework
Language Reference