Contents

- 개요
- 자습서
  - ADO.NET을 사용하여 간단한 데이터 앱 만들기
  - WPF 및 Entity Framework를 사용하여 간단한 데이터 앱 만들기
  - C++ 앱에서 데이터에 연결
  - O/R 디자이너를 사용하여 LINQ to SQL 클래스 만들기
  - n 계층 데이터 애플리케이션 만들기
- 개념
  - 호환되는 데이터베이스 시스템
  - .NET용 Visual Studio 데이터 도구
  - Entity Framework 도구
  - DataSets
    - 데이터 세트 도구
    - 형식화된 데이터 세트 및 형식화되지 않은 데이터 세트
    - TableAdapters 개요
    - n 계층 애플리케이션에서 데이터 세트 작업
  - 데이터베이스 프로젝트 및 데이터 계층 애플리케이션(DAC)
  - n 계층 데이터 애플리케이션 개요
- 방법 가이드
  - 데이터베이스 만들기
  - Connections
    - 연결 추가
    - 연결 문자열 저장 및 편집
    - Access 데이터베이스의 데이터에 연결
  - .NET 데이터 원본 추가
  - LINQ to SQL 도구
    - 개요
    - DataContext 메서드의 반환 형식 변경
    - 저장 프로시저에 매핑된 DataContext 메서드 만들기

- O/R 디자이너를 사용하여 상속 구성
- 테이블 및 뷰에 매핑된 LINQ to SQL 클래스 만들기
- O/R 디자이너에서 생성한 코드 확장
- LINQ to SQL 클래스 간 연결 만들기
- 엔터티 클래스에 유효성 검사 추가
- 엔터티 클래스의 삽입, 업데이트 및 삭제 동작 사용자 지정
- 저장 프로시저를 할당하여 업데이트, 삽입 및 삭제 수행
- 복수 적용 설정 및 해제

데이터 세트 도구 사용

- 데이터 세트 만들기
  - 데이터 세트 만들기 및 구성
  - 데이터 세트 간 관계 만들기
  - 데이터 세트 디자이너를 사용하여 데이터 세트 만들기
  - 데이터 세트 디자이너를 사용하여 DataTable 만들기
- TableAdapter 구성
  - TableAdapter 만들기 및 구성
  - 매개 변수가 있는 TableAdapter 쿼리 만들기
  - TableAdapter를 사용하여 데이터베이스에 직접 액세스
  - 데이터 세트를 채우는 동안 제약 조건 끄기
  - TableAdapter의 기능 확장
- XML 데이터를 데이터 세트에 읽어오기
- 데이터 세트의 데이터 편집
- 데이터 세트의 데이터 유효성 검사
- 데이터를 다시 데이터베이스에 저장
  - 개요
  - 데이터베이스에 새 레코드 삽입
  - TableAdapter를 사용하여 데이터 업데이트
  - 계층적 업데이트 실행
  - 동시성 예외 처리
  - 트랜잭션
    - 트랜잭션을 사용하여 데이터 저장
    - 연습: 트랜잭션에 데이터 저장

데이터베이스에 데이터 저장(여러 테이블)
개체에서 데이터베이스로 데이터 저장
TableAdapter DBDirect 메서드를 사용하여 데이터 저장
데이터 세트를 XML로 저장
데이터 세트 쿼리
n 계층 애플리케이션
n 계층 데이터 세트에 유효성 검사 추가
n 계층 애플리케이션에서 데이터 세트에 코드 추가
n 계층 애플리케이션에서 TableAdapter에 코드 추가
데이터 세트 및 TableAdapter를 다른 프로젝트로 분리
컨트롤
데이터 원본에 컨트롤 바인딩
데이터 소스 창에서 끌어올 때 만들 컨트롤 설정
데이터 소스 창에 사용자 지정 컨트롤 추가
WPF 컨트롤
데이터에 WPF 컨트롤 바인딩
데이터 세트로 WPF 컨트롤 바인딩
WCF 데이터 서비스에 WPF 컨트롤 바인딩
조회 테이블 만들기
관련 데이터 표시
데이터베이스의 그림에 컨트롤 바인딩
Windows Forms 컨트롤
데이터에 Windows Forms 컨트롤 바인딩
Windows Forms 애플리케이션에서 데이터 필터링 및 정렬
데이터 바인딩된 컨트롤에서 데이터를 저장하기 전에 In-Process 편집 커밋
조회 테이블 만들기
데이터 검색을 위한 Windows Form 만들기
단순 데이터 바인딩을 지원하는 사용자 정의 컨트롤 만들기
복합 데이터 바인딩을 지원하는 사용자 정의 컨트롤 만들기
조회 데이터 바인딩을 지원하는 사용자 정의 컨트롤 만들기
폼 간에 데이터 전달
사용자 지정 개체 바인딩

Visual Studio에서 데이터 바인딩된 컨트롤에 대한 캡션을 만드는 방식 사용자 지정
Windows Communication Foundation 서비스 및 WCF Data Services
Visual Studio의 WCF 개요
개념적 모델 작업
서비스의 데이터에 연결
Windows Forms에서 WCF 서비스 만들기
WPF 및 Entity Framework를 사용하여 WCF 데이터 서비스 만들기
서비스 참조 문제 해결
서비스 참조 구성 대화 상자
WCF 데이터 서비스 참조 추가, 업데이트 또는 제거
.mdf 파일 업그레이드
데이터 액세스 오류 문제 해결
참조
O/R 디자이너(Linq to SQL)
DataContext 메서드
데이터 클래스 상속
O/R 디자이너 메시지
선택한 클래스가 하나 이상의 DataContext 메서드의 반환 형식으로 사용되므로 해당 클래스를 삭제할 수 없습니다.
연결 문자열에 일반 텍스트 암호가 있는 자격 증명이 포함되어 있으며 통합 보안을 사용하지 않습니다.
<property name> 속성이 <association name> 연결에 참여하고 있으므로 이 속성을 삭제할 수 없습니다.
<property name> 속성을 삭제할 수 없습니다.
애플리케이션 설정 파일에서 연결 속성이 없거나 잘못되었습니다.
<association name> 연결을 만들 수 없습니다. 속성 형식이 일치하지 않습니다.
경고. 동작 구성 대화 상자에 적용되지 않은 변경 내용이 있습니다.
지원되지 않는 데이터베이스 공급자에서 데이터베이스 공급자를 선택했습니다.
이 관련 메서드는 다음과 같은 기본 삽입, 업데이트 및 삭제 메서드를 지원하는 메서드입니다.
선택한 항목 중 하나 이상이 디자이너에서 지원되지 않는 데이터 형식을 포함하고 있습니다.
DataContext 메서드의 반환 형식 변경은 취소할 수 없습니다.
디버깅하는 동안에는 디자이너를 수정할 수 없습니다.

선택한 연결에서 지원되지 않는 데이터베이스 공급자를 사용합니다.

디자이너에 추가하려는 개체가 현재 디자이너가 사용 중인 것과 다른 데이터 연결을 사용합니다.

<association name> 연결을 만들 수 없습니다. 속성이 두 번 나열되었습니다.

데이터베이스 개체의 스키마 정보를 검색할 수 없습니다.

선택한 데이터베이스 개체 중 하나 이상이 대상 클래스의 스키마와 일치하지 않는 스키마를 반환합니다.

Visual Studio에서 데이터 작업

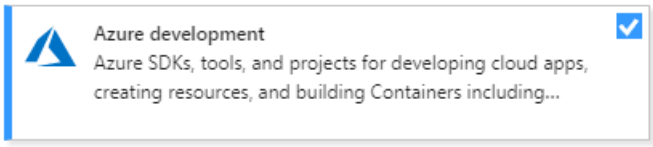
2020-02-18 • 25 minutes to read • Edit Online

Visual Studio에서는 거의 모든 데이터베이스 제품 또는 서비스의 데이터에 연결 하는 응용 프로그램을 로컬 컴퓨터, 로컬 영역 네트워크 또는 공용, 사설 또는 하이브리드 클라우드에서 모든 형식으로 만들 수 있습니다.

JavaScript, Python, PHP, Ruby 또는 C++의 응용 프로그램의 경우 라이브러리를 가져오고 코드를 작성 하 여 다른 작업을 수행 하는 것 처럼 데이터에 연결 합니다. .NET 응용 프로그램의 경우 Visual Studio는 데이터 원본을 탐색 하 고, 개체 모델을 만들어 메모리에 데이터를 저장 및 조작 하 고, 데이터를 사용자 인터페이스에 바인딩하는 데 사용할 수 있는 도구를 제공 합니다. Microsoft Azure는 .NET, Java, node.js, PHP, Python, Ruby 및 모바일 앱에 대 한 Sdk 및 Visual Studio에서 Azure Storage에 연결 하는 데 사용할 수 있는 도구를 제공 합니다.

다음 목록에서는 Visual Studio에서 사용할 수 있는 몇 가지 데이터베이스 및 저장소 시스템을 보여 줍니다. Microsoft Azure 제품은 기본 데이터 저장소의 모든 프로 비전 및 관리를 포함 하는 데이터 서비스입니다. Visual studio 2017 의 **azure 개발** 워크 로드를 통해 Visual studio에서 직접 azure 데이터 저장소를 사용할 수 있습니다.

다음 목록에서는 Visual Studio에서 사용할 수 있는 몇 가지 데이터베이스 및 저장소 시스템을 보여 줍니다. Microsoft Azure 제품은 기본 데이터 저장소의 모든 프로 비전 및 관리를 포함 하는 데이터 서비스입니다. Visual studio 2019 의 **azure 개발** 워크 로드를 통해 Visual studio에서 직접 azure 데이터 저장소를 사용할 수 있습니다.



여기에 나열 된 다른 SQL 및 NoSQL 데이터베이스 제품은 대부분 로컬 컴퓨터, 로컬 네트워크 또는 가상 컴퓨터의 Microsoft Azure에서 호스팅될 수 있습니다. Microsoft Azure 가상 머신에서 데이터베이스를 호스트 하는 경우 데이터베이스 자체를 관리 하는 일을 담당 합니다.

Microsoft Azure

- SQL Database
- Azure Cosmos DB
- 저장소 (blob, 테이블, 큐, 파일)
- SQL Data Warehouse
- SQL Server Stretch Database
- StorSimple
- 이 외에도 다양한 혜택이 제공됩니다!

SQL

- SQL Server 2005-2016 (Express 및 LocalDB 포함)
- Firebird
- MariaDB
- MySQL
- Oracle
- PostgreSQL
- SQLite
- 이 외에도 다양한 혜택이 제공됩니다!

NoSQL

- Apache Cassandra
- CouchDB
- MongoDB
- NDatabase
- OrientDB|
- RavenDB
- VelocityDB
- 이 외에도 다양한 혜택이 제공됩니다!

많은 데이터베이스 공급 업체와 제 3 자가 NuGet 패키지를 통해 Visual Studio 통합을 지원 합니다. Nuget.org에서 또는 Visual Studio의 NuGet 패키지 관리자를 통해 제공 되는 기능을 탐색할 수 있습니다 (**도구 > nuget 패키지 관리자 > 솔루션에 대 한 nuget 패키지 관리**). 다른 데이터베이스 제품은 Visual Studio와 확장으로 통합 됩니다. Visual Studio Marketplace 에서 이러한 제품을 찾아보거나, **도구 > 확장 및 업데이트** 로 이동한 다음 대화 상자의 왼쪽 창에서 **온라인** 을 선택 합니다. 자세한 내용은 Visual Studio에 대 한 호환 데이터베이스 시스템을 참조 하세요.

많은 데이터베이스 공급 업체와 제 3 자가 NuGet 패키지를 통해 Visual Studio 통합을 지원 합니다. Nuget.org에서 또는 Visual Studio의 NuGet 패키지 관리자를 통해 제공 되는 기능을 탐색할 수 있습니다 (**도구 > nuget 패키지 관리자 > 솔루션에 대 한 nuget 패키지 관리**). 다른 데이터베이스 제품은 Visual Studio와 확장으로 통합 됩니다. Visual Studio Marketplace 에서 이러한 **제품을 찾아보거나 확장 > 확장** 으로 이동한 다음 대화 상자의 왼쪽 창에서 **온라인 을 선택** 하면 됩니다. 자세한 내용은 Visual Studio에 대 한 호환 데이터베이스 시스템을 참조 하세요.

> **NOTE**
> SQL Server 2005에 대해 연장된 지원은 2016년 4월 12일에 종료되었습니다. Visual Studio 2015 이상에서 데이터 도구가 SQL Server 2005에서 계속 작동할 수 있는 것은 아닙니다. 자세한 내용은 SQL Server 2005에 대 한 지원 종료 알림을 참조 하세요.

.NET 언어

.NET Core에 포함 된 모든 .NET 데이터 액세스는 모든 종류의 데이터 원본에 액세스 하기 위한 인터페이스를 정의 하는 클래스 집합인 ADO.NET을 기반으로 합니다. Visual Studio에는 데이터베이스에 연결 하 고, 데이터를 조작 하 고, 사용자에 게 데이터를 표시 하는 데 도움이 되는 ADO.NET와 함께 작동 하는 몇 가지 도구와 디자이너가 있습니다. 이 섹션의 설명서에서는 이러한 도구를 사용 하는 방법을 설명 합니다. ADO.NET 명령 개체에 대해 직접 프로그래밍할 수도 있습니다. ADO.NET Api를 직접 호출 하는 방법에 대 한 자세한 내용은 ADO.NET를 참조 하세요.

ASP.NET 관련 된 데이터 액세스 설명서는 ASP.NET 사이트에서 데이터 작업 을 참조 하세요. ASP.NET MVC와 함께 Entity Framework를 사용 하는 방법에 대 한 자습서는 mvc 5를 사용 하 여 Entity Framework 6 Code First 시작을 참조 하세요.

또는 Visual Basic의 C# UWP (유니버설 Windows 플랫폼) 앱은 .net 용 Microsoft Azure SDK를 사용 하 여 Azure Storage 및 기타 Azure 서비스에 액세스할 수 있습니다. Windows. HttpClient 클래스를 사용 하면 모든 RESTful 서비스와 통신할 수 있습니다. 자세한 내용은 Windows를 사용 하 여 http 서버에 연결 하는 방법을 참조 하세요.

로컬 컴퓨터에 데이터를 저장 하는 경우 권장 되는 방법은 앱과 동일한 프로세스에서 실행 되는 SQLite를 사용 하는 것입니다. ORM (개체-관계형 매핑) 계층이 필요한 경우 Entity Framework를 사용할 수 있습니다. 자세한 내용은 Windows 개발자 센터에서 데이터 액세스 를 참조 하세요.

Azure 서비스에 연결 하는 경우 최신 AZURE SDK 도구를 다운로드 해야 합니다.

데이터 공급자

ADO.NET에서 데이터베이스를 사용할 수 있도록 하려면 사용자 지정 *ADO.NET 데이터 공급자* 가 있어야 합니다.

그렇지 않으면 ODBC 또는 OLE DB 인터페이스를 노출 해야 합니다. Microsoft는 SQL Server 제품 뿐만 아니라 ODBC 및 OLE DB 공급자에 대 한 ADO.NET 데이터 공급자의 목록을 제공 합니다.

데이터 모델링

.NET에서는 데이터 원본에서 데이터를 검색 한 후 메모리에서 데이터를 모델링 하 고 조작할 수 있는 세 가지 옵션이 있습니다.

Entity Framework 기본 설정 된 Microsoft ORM 기술입니다. 이를 사용 하 여 관계형 데이터를 최고 수준의 .NET 개체로 프로그래밍할 수 있습니다. 새 응용 프로그램의 경우 모델이 필요할 때 첫 번째로 선택 해야 합니다. 기본 ADO.NET 공급자의 사용자 지정 지원이 필요 합니다.

LINQ to SQL 이전 세대 개체-관계형 매퍼입니다. 더 복잡 한 시나리오에 적합 하지만 더 이상 개발에 더 이상 필요 하지 않습니다.

데이터 집합 세 가지 모델링 기술 중 가장 오래 된 기술입니다. 대량의 데이터를 처리 하거나 복잡 한 쿼리 또는 변환을 수행 하지 않는 "데이터 폼" 응용 프로그램을 신속 하 게 개발 하는 데 주로 설계 되었습니다. DataSet 개체는 SQL 데이터베이스 개체와 논리적으로 동일한 SQL 데이터베이스 개체와 유사한 DataTable 및 DataRow 개체로 구성 됩니다. SQL 데이터 원본을 기반으로 하는 비교적 간단한 응용 프로그램의 경우에도 데이터 집합을 선택 하는 것이 좋습니다.

이러한 기술을 사용할 필요는 없습니다. 특히 성능이 중요 한 일부 시나리오에서는 DataReader 개체를 사용 하 여 데이터베이스에서 읽고 필요한 값을<T > 목록 등의 컬렉션 개체에 복사할 수 있습니다.

네이티브 C++

C++SQL Server에 연결 하는 응용 프로그램은 대부분의 경우 SQL Server 용 Microsoft® ODBC 드라이버 13.1 를 사용 해야 합니다. 서버가 연결 된 경우 OLE DB 필요 하며 SQL Server Native Client를 사용 합니다. ODBC 또는 OLE DB 드라이버를 직접 사용 하 여 다른 데이터베이스에 액세스할 수 있습니다. ODBC는 현재 표준 데이터베이스 인터페이스 이지만 대부분의 데이터베이스 시스템은 ODBC 인터페이스를 통해 액세스할 수 없는 사용자 지정 기능을 제공 합니다. OLE DB은 계속 지원 되지만 새 응용 프로그램에는 권장 되지 않는 레거시 COM 데이터 액세스 기술입니다. 자세한 내용은 시각적 개체 C++의 데이터 액세스 를 참조 하세요.

C++rest 서비스를 사용 하는 프로그램은 C++ rest SDK를 사용할 수 있습니다.

C++Microsoft Azure Storage와 함께 작동 하는 프로그램은 Microsoft Azure Storage 클라이언트를 사용할 수 있습니다.

Visual Studio—데이터 모델링은에 대 한 C++ORM 계층을 제공 하지 않습니다. ODB 는의 C++인기 있는 오픈 소스 ORM입니다.

응용 프로그램에서 C++ 데이터베이스에 연결 하는 방법에 대 한 자세한 내용은 용 C++Visual Studio data tools 를 참조 하세요. 레거시 시각적 C++ 데이터 액세스 기술에 대 한 자세한 내용은 데이터 액세스를 참조 하세요.

JavaScript

Visual Studio의 JavaScript 는 플랫폼 간 앱, UWP 앱, 클라우드 서비스, 웹 사이트 및 웹 앱을 빌드하기 위한 최고 수준의 언어입니다. Visual Studio 내에서 Bower, Grunt, Gulp, npm 및 NuGet을 사용 하 여 즐겨 사용 하는 JavaScript 라이브러리 및 데이터베이스 제품을 설치할 수 있습니다. Azure 웹 사이트에서 sdk를 다운로드 하 여 azure storage 및 서비스에 연결 합니다. Node.js는 서버 쪽 JavaScript (node.js)를 ADO.NET 데이터 원본에 연결 하는 라이브러리입니다.

Python

Python 응용 프로그램을 만들기 위해 Visual Studio에서 python 지원을 설치 합니다. Azure 설명서에는 다음을 포함 하 여 데이터에 연결 하는 몇 가지 자습서가 있습니다.

- Azure의 Django 및 SQL Database

- Azure의 Django 및 MySQL
- Blob, 파일, 큐및 테이블 (cosmo DB)작업

관련 항목

MICROSOFT AI platform—는 Cortana Analytics Suite 및 사물 인터넷에 대 한 지원을 포함 하 여 microsoft intelligent cloud에 대 한 소개를 제공 합니다.

Microsoft Azure Storage—에서는 Azure Storage에 대해 설명 하 고, Azure blob, 테이블, 큐 및 파일을 사용 하 여 응용 프로그램을 만드는 방법을 설명 합니다.

Azure SQL Database—관계형 Database as a Service Azure SQL Database에 연결 하는 방법을 설명 합니다.

SQL Server Data Tools—에서는 데이터 연결 응용 프로그램 및 데이터베이스의 디자인, 탐색, 테스트 및 배포를 간소화 하는 도구에 대해 설명 합니다.

ADO.NET—는 ADO.NET 아키텍처와 ADO.NET 클래스를 사용 하 여 응용 프로그램 데이터를 관리 하 고 데이터 원본 및 XML과 상호 작용 하는 방법을 설명 합니다.

ADO.NET Entity Framework—는 개발자가 관계형 데이터베이스에 대해 직접 프로그래밍 하는 대신 개념적 모델을 기반으로 프로그래밍할 수 있도록 하는 데이터 응용 프로그램을 만드는 방법을 설명 합니다.

WCF Data Services 4.5—WCF Data Services를 사용 하 여 웹 또는 OData (Open Data Protocol)를 구현 하는 인트라넷에 데이터 서비스를 배포 하는 방법을 설명 합니다.

Office 솔루션의 데이터—에는 office 솔루션에서 데이터가 작동 하는 방식을 설명 하는 항목에 대 한 링크가 포함 되어 있습니다. 여기에는 스키마 지향 프로그래밍, 데이터 캐싱 및 서버 쪽 데이터 액세스에 대 한 정보가 포함 됩니다.

LINQ (통합 언어 쿼리)—에 C# 기본 제공 되 고 Visual Basic 되는 쿼리 기능과 관계형 데이터베이스, XML 문서, 데이터 집합 및 메모리 내 컬렉션을 쿼리 하기 위한 일반 모델을 설명 합니다.

Visual Studio의 Xml 도구—xml 데이터 작업, XSLT 디버깅, .net xml 기능 및 xml 쿼리 아키텍처에 대해 설명 합니다.

Xml 문서 및 데이터—는 .NET에서 xml 문서 및 데이터를 사용 하는 포괄적이 고 통합 된 클래스 집합에 대 한 개요를 제공 합니다.

ADO.NET을 사용하여 간단한 데이터 애플리케이션 만들기

2020-01-06 • 33 minutes to read • Edit Online

데이터베이스의 데이터를 조작하는 애플리케이션을 만들면 연결 문자열 정의, 데이터 삽입 및 저장 프로시저 실행과 같은 기본 작업을 수행합니다. 이 항목을 참조 하 여 Visual C# 또는 Visual Basic 및 ADO.NET를 사용 하 여 간단한 Windows Forms "데이터 폼" 응용 프로그램 내에서 데이터베이스와 상호 작용 하는 방법을 알아볼 수 있습니다. 데이터 집합, LINQ to SQL 및 Entity Framework를 비롯 한 모든 .NET 데이터 기술은 궁극적으로이 문서에 표시 된 것과 매우 유사한 단계를 수행 합니다.

이 문서에서는 빠른 방법으로 데이터베이스에서 데이터를 가져오는 간단한 방법을 보여 줍니다. 응용 프로그램에서 중요 하지 않은 방법으로 데이터를 수정 하 고 데이터베이스를 업데이트 해야 하는 경우 Entity Framework 사용을 고려 하 고 데이터 바인딩을 사용 하 여 사용자 인터페이스 컨트롤을 기본 데이터의 변경 내용에 자동으로 동기화 해야 합니다.

> **IMPORTANT**
> 코드를 간단히 유지하기 위해 프로덕션에 사용하는 예외 처리는 포함되어 있지 않습니다.

전제 조건

애플리케이션을 만들려면 다음이 필요 합니다.

- 보여 줍니다.
- SQL Server Express LocalDB. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 설치할 수 있습니다.

이 항목에서는 사용자가 Visual Studio IDE의 기본 기능에 대해 잘 알고 있는 것으로 가정 하 고, 프로젝트에 폼을 추가 하 고, 폼에 단추 및 기타 컨트롤을 추가 하 고, 컨트롤의 속성을 설정 하 고, 간단한 이벤트를 코딩 하 여 Windows Forms 응용 프로그램을 만듭니다. 이러한 작업에 익숙하지 않은 경우이 연습을 시작 하기 전에 시각적 개체 C# 및 Visual Basic 시작 항목을 완료 하는 것이 좋습니다.

샘플 데이터베이스 설정

다음 단계를 수행 하 여 예제 데이터베이스를 만듭니다.

1. Visual Studio에서 **서버 탐색기** 창을 엽니다.
2. **데이터 연결** 을 마우스 오른쪽 단추로 클릭 하 고 **새 SQL Server 데이터베이스 만들기**를 선택 합니다.
3. **서버 이름** 텍스트 상자에 **(localdb) \mssqllocaldb**를 입력 합니다.
4. **새 데이터베이스 이름** 텍스트 상자에 **Sales**를 입력 한 다음 **확인**을 선택 합니다.

   빈 **Sales** 데이터베이스가 만들어지고 서버 탐색기의 데이터 연결 노드에 추가 됩니다.
5. **Sales** 데이터 연결을 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.
6. Sales transact-sql 스크립트 를 클립보드에 복사 합니다.

7. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 데이터베이스 개체가 만들어집니다. 데이터베이스에는 Customer와 Orders 라는 두 개의 테이블이 있습니다. 이러한 테이블은 처음에는 데이터를 포함 하지 않지만 만들 응용 프로그램을 실행할 때 데이터를 추가할 수 있습니다. 데이터베이스에는 네 개의 간단한 저장 프로시저도 있습니다.

폼 만들기 및 컨트롤 추가

1. Windows Forms 애플리케이션의 프로젝트를 만든 다음, 이름을 **SimpleDataApp**으로 지정합니다.

   Visual Studio에서는 프로젝트와 **Form1**이라는 빈 Windows 양식을 포함한 여러 파일을 만듭니다.

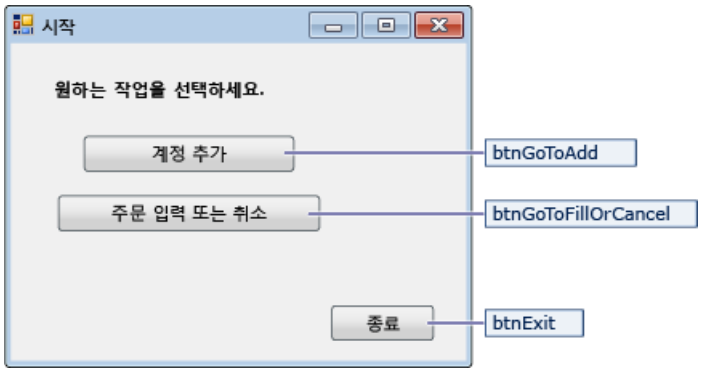
2. 프로젝트에 두 개의 Windows 양식을 추가하여 총 세 개의 양식을 만든 다음, 다음 이름을 지정합니다.

   - **탐색**
   - **NewCustomer**
   - **FillOrCancel**

3. 각 폼에 대해 다음 그림에 나오는 텍스트 상자, 단추 및 기타 컨트롤을 추가합니다. 각 컨트롤에 대해 테이블이 설명하는 속성을 설정합니다.

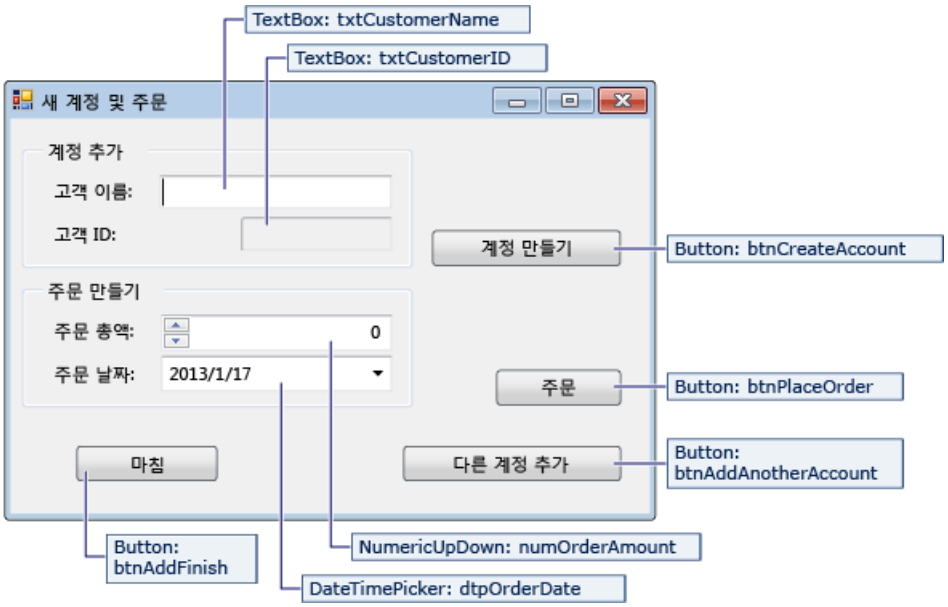
   > **NOTE**
   > 그룹 상자 및 레이블 컨트롤도 선명성을 더해 주지만 코드에서는 사용하지 않습니다.

   탐색 양식



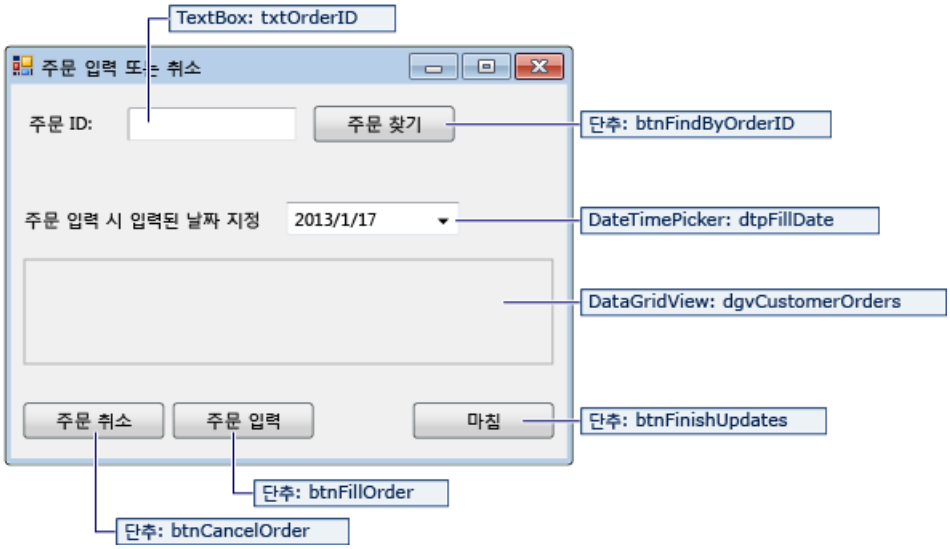
NAVIGATION 폼 컨트롤	속성
단추	Name = btnGoToAdd
단추	Name = btnGoToFillOrCancel
단추	Name = btnExit

NewCustomer 양식



NEWCUSTOMER 폼 컨트롤	속성
TextBox	Name = txtCustomerName
TextBox	Name = txtCustomerID Readonly = True
단추	Name = btnCreateAccount
NumericUpdown	DecimalPlaces = 0 Maximum = 5000 Name = numOrderAmount
DateTimePicker	Format = Short Name = dtpOrderDate
단추	Name = btnPlaceOrder
단추	Name = btnAddAnotherAccount
단추	Name = btnAddFinish

FillOrCancel 양식



FILLORCANCEL 폼 컨트롤	속성
TextBox	Name = txtOrderID
단추	Name = btnFindByOrderID
DateTimePicker	Format = Short Name = dtpFillDate
DataGridView	Name = dgvCustomerOrders Readonly = True RowHeadersVisible = False
단추	Name = btnCancelOrder
단추	Name = btnFillOrder
단추	Name = btnFinishUpdates

연결 문자열 저장

애플리케이션이 데이터베이스에 대한 연결을 열려면 애플리케이션에는 연결 문자열에 액세스할 수 있어야 합니다. 각 폼에 문자열을 수동으로 입력 하지 않도록 하려면 프로젝트의 *app.config 파일에* 문자열을 저장 하 고, 응용 프로그램의 모든 폼에서 메서드가 호출 될 때 문자열을 반환 하는 메서드를 만듭니다.

서버 탐색기 에서 **판매** 데이터 연결을 마우스 오른쪽 단추로 클릭 하 고 **속성**을 선택 하 여 연결 문자열을 찾을 수 있습니다. **ConnectionString** 속성을 찾은 다음 **ctrl**+**A**, **ctrl**+**C** 를 사용 하 여 문자열을 선택 하 고 클립보드에 복사 합니다.

1. 를 C#사용 하는 경우 **솔루션 탐색기**에서 프로젝트의 **속성** 노드를 확장 한 다음 **설정** 파일을 엽니다. Visual Basic를 사용 하는 경우 **솔루션 탐색기**에서 **모든 파일 표시**를 클릭 하 고 **내 프로젝트** 노드를 확장 한 다음 **설정** 파일을 엽니다.
2. **이름** 열에 `connString`을 입력 합니다.
3. **유형** 목록에서 **(연결 문자열)** 을 선택 합니다.
4. **범위** 목록에서 **응용 프로그램**을 선택 합니다.

5. **값** 열에 외부 따옴표 없이 연결 문자열을 입력 한 다음 변경 내용을 저장 합니다.

> **NOTE**
> 실제 응용 프로그램에서는 연결 문자열 및 구성 파일에 설명 된 대로 연결 문자열을 안전 하 게 저장 해야 합니다.

폼에 대한 코드 작성

이 섹션에는 각 양식이 수행 하는 작업에 대 한 간략 한 개요가 포함 되어 있습니다. 또한 폼의 단추를 클릭할 때 기본 논리를 정의 하는 코드도 제공 합니다.

Navigation 폼

애플리케이션을 실행하면 Navigation 폼이 열립니다. **계정 추가** 단추는 NewCustomer 양식을 엽니다. **주문 이행 또는 취소** 단추를 누르면 FillOrCancel 양식이 열립니다. **끝내기** 단추를 클릭하면 애플리케이션이 닫힙니다.

Navigation 폼을 시작 폼으로 만들기

C#을 사용하는 경우 **솔루션 탐색기**에서 **Program.cs**를 연 다음, `Application.Run` 줄을 `Application.Run(new Navigation());`으로 변경합니다.

Visual Basic를 사용 하는 경우 **솔루션 탐색기**에서 **속성** 창을 열고 **응용 프로그램** 탭을 선택한 후 **시작 폼** 목록에서 **simpledataapp.navigation** 를 선택 합니다.

자동 생성 된 이벤트 처리기 만들기

탐색 폼의 세 단추를 두 번 클릭 하 여 빈 이벤트 처리기 메서드를 만듭니다. 단추를 두 번 클릭 하면 단추 클릭으로 이벤트를 발생 시킬 수 있는 자동 생성 코드도 디자이너 코드 파일에 추가 됩니다.

탐색 폼 논리에 대 한 코드 추가

탐색 폼의 코드 페이지에서 다음 코드와 같이 세 개의 단추 클릭 이벤트 처리기에 대 한 메서드 본문을 완성 합니다.

```
/// <summary>
/// Opens the NewCustomer form as a dialog box,
/// which returns focus to the calling form when it is closed.
/// </summary>
private void btnGoToAdd_Click(object sender, EventArgs e)
{
    Form frm = new NewCustomer();
    frm.Show();
}

/// <summary>
/// Opens the FillorCancel form as a dialog box.
/// </summary>
private void btnGoToFillOrCancel_Click(object sender, EventArgs e)
{
    Form frm = new FillOrCancel();
    frm.ShowDialog();
}

/// <summary>
/// Closes the application (not just the Navigation form).
/// </summary>
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

```
''' <summary>
''' Opens the NewCustomer form as a dialog box, which returns focus to the calling form when it closes.
''' </summary>
Private Sub btnGoToAdd_Click(sender As Object, e As EventArgs) Handles btnGoToAdd.Click
    Dim frm As Form = New NewCustomer()
    frm.Show()
End Sub

''' <summary>
''' Opens the FillOrCancel form as a dialog box.
''' </summary>
Private Sub btnGoToFillOrCancel_Click(sender As Object, e As EventArgs) Handles btnGoToFillOrCancel.Click
    Dim frm As Form = New FillOrCancel()
    frm.ShowDialog()
End Sub

''' <summary>
''' Closes the application (not just the navigation form).
''' </summary>
Private Sub btnExit_Click(sender As Object, e As EventArgs) Handles btnExit.Click
    Me.Close()
End Sub
```

NewCustomer 폼

고객 이름을 입력 한 후 **계정 만들기** 단추를 선택 하면 newcustomer 폼은 고객 계정을 만들고 SQL SERVER는 id 값을 새 고객 id로 반환 합니다. 그런 다음 금액 및 주문 날짜를 지정 하 고 **주문** 단추를 선택 하 여 새 계정에 대 한 주문을 수행할 수 있습니다.

자동 생성 된 이벤트 처리기 만들기

각각의 네 단추를 두 번 클릭 하 여 NewCustomer 폼의 각 단추에 대해 빈 Click 이벤트 처리기를 만듭니다. 단추를 두 번 클릭 하면 단추 클릭으로 이벤트를 발생 시킬 수 있는 자동 생성 코드도 디자이너 코드 파일에 추가 됩니다.

NewCustomer 양식 논리에 대 한 코드 추가

NewCustomer 양식 논리를 완료 하려면 다음 단계를 수행 합니다.

1. 해당 멤버의 이름을 정규화 할 필요가 없도록 `System.Data.SqlClient` 네임 스페이스를 범위로 가져옵니다.

   ```
   using System.Data.SqlClient;
   ```

   ```
   Imports System.Data.SqlClient
   ```

2. 다음 코드와 같이 일부 변수와 도우미 메서드를 클래스에 추가 합니다.

```
// Storage for IDENTITY values returned from database.
private int parsedCustomerID;
private int orderID;

/// <summary>
/// Verifies that the customer name text box is not empty.
/// </summary>
private bool IsCustomerNameValid()
{
    if (txtCustomerName.Text == "")
    {
        MessageBox.Show("Please enter a name.");
        return false;
    }
    else
    {
        return true;
    }
}

/// <summary>
/// Verifies that a customer ID and order amount have been provided.
/// </summary>
private bool IsOrderDataValid()
{
    // Verify that CustomerID is present.
    if (txtCustomerID.Text == "")
    {
        MessageBox.Show("Please create customer account before placing order.");
        return false;
    }
    // Verify that Amount isn't 0.
    else if ((numOrderAmount.Value < 1))
    {
        MessageBox.Show("Please specify an order amount.");
        return false;
    }
    else
    {
        // Order can be submitted.
        return true;
    }
}

/// <summary>
/// Clears the form data.
/// </summary>
private void ClearForm()
{
    txtCustomerName.Clear();
    txtCustomerID.Clear();
    dtpOrderDate.Value = DateTime.Now;
    numOrderAmount.Value = 0;
    this.parsedCustomerID = 0;
}
```

```
' Storage for ID values returned from the database.
Private parsedCustomerID As Integer
Private orderID As Integer

''' <summary>
''' Verifies that the customer name text box is not empty.
''' </summary>
Private ReadOnly Property IsCustomerNameValid As Boolean
    Get
        If txtCustomerName.Text = "" Then
            MessageBox.Show("Please enter a name.")
            Return False
        Else
            Return True
        End If
    End Get
End Property

''' <summary>
''' Verifies the order data is valid.
''' </summary>
Private Function IsOrderDataValid() As Boolean

    ' Verify that CustomerID is present.
    If txtCustomerID.Text = "" Then
        MessageBox.Show("Please create a customer account before placing order.")
        Return False

        ' Verify that order amount isn't 0.
    ElseIf (numOrderAmount.Value < 1) Then
        MessageBox.Show("Please specify an order amount.")
        Return False
    Else
        ' Order can be submitted.
        Return True
    End If
End Function

''' <summary>
''' Clears values from controls.
''' </summary>
Private Sub ClearForm()
    txtCustomerName.Clear()
    txtCustomerID.Clear()
    dtpOrderDate.Value = DateTime.Now
    numOrderAmount.Value = 0
    Me.parsedCustomerID = 0
End Sub
```

3. 다음 코드와 같이 4 개의 단추 클릭 이벤트 처리기에 대 한 메서드 본문을 완성 합니다.

```
/// <summary>
/// Creates a new customer by calling the Sales.uspNewCustomer stored procedure.
/// </summary>
private void btnCreateAccount_Click(object sender, EventArgs e)
{
    if (IsCustomerNameValid())
    {
        // Create the connection.
        using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
        {
            // Create a SqlCommand, and identify it as a stored procedure.
            using (SqlCommand sqlCommand = new SqlCommand("Sales.uspNewCustomer", connection))
            {
                sqlCommand.CommandType = CommandType.StoredProcedure;
```

```
                // Add input parameter for the stored procedure and specify what to use as its value.
                sqlCommand.Parameters.Add(new SqlParameter("@CustomerName", SqlDbType.NVarChar, 40));
                sqlCommand.Parameters["@CustomerName"].Value = txtCustomerName.Text;

                // Add the output parameter.
                sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
                sqlCommand.Parameters["@CustomerID"].Direction = ParameterDirection.Output;

                try
                {
                    connection.Open();

                    // Run the stored procedure.
                    sqlCommand.ExecuteNonQuery();

                    // Customer ID is an IDENTITY value from the database.
                    this.parsedCustomerID = (int)sqlCommand.Parameters["@CustomerID"].Value;

                    // Put the Customer ID value into the read-only text box.
                    this.txtCustomerID.Text = Convert.ToString(parsedCustomerID);
                }
                catch
                {
                    MessageBox.Show("Customer ID was not returned. Account could not be created.");
                }
                finally
                {
                    connection.Close();
                }
            }
        }
    }
}

/// <summary>
/// Calls the Sales.uspPlaceNewOrder stored procedure to place an order.
/// </summary>
private void btnPlaceOrder_Click(object sender, EventArgs e)
{
    // Ensure the required input is present.
    if (IsOrderDataValid())
    {
        // Create the connection.
        using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
        {
            // Create SqlCommand and identify it as a stored procedure.
            using (SqlCommand sqlCommand = new SqlCommand("Sales.uspPlaceNewOrder", connection))
            {
                sqlCommand.CommandType = CommandType.StoredProcedure;

                // Add the @CustomerID input parameter, which was obtained from uspNewCustomer.
                sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
                sqlCommand.Parameters["@CustomerID"].Value = this.parsedCustomerID;

                // Add the @OrderDate input parameter.
                sqlCommand.Parameters.Add(new SqlParameter("@OrderDate", SqlDbType.DateTime, 8));
                sqlCommand.Parameters["@OrderDate"].Value = dtpOrderDate.Value;

                // Add the @Amount order amount input parameter.
                sqlCommand.Parameters.Add(new SqlParameter("@Amount", SqlDbType.Int));
                sqlCommand.Parameters["@Amount"].Value = numOrderAmount.Value;

                // Add the @Status order status input parameter.
                // For a new order, the status is always O (open).
                sqlCommand.Parameters.Add(new SqlParameter("@Status", SqlDbType.Char, 1));
                sqlCommand.Parameters["@Status"].Value = "O";

                // Add the return value for the stored procedure, which is  the order ID.
                sqlCommand.Parameters.Add(new SqlParameter("@RC", SqlDbType.Int));
```

```
                sqlCommand.Parameters["@RC"].Direction = ParameterDirection.ReturnValue;

                try
                {
                    //Open connection.
                    connection.Open();

                    // Run the stored procedure.
                    sqlCommand.ExecuteNonQuery();

                    // Display the order number.
                    this.orderID = (int)sqlCommand.Parameters["@RC"].Value;
                    MessageBox.Show("Order number " + this.orderID + " has been submitted.");
                }
                catch
                {
                    MessageBox.Show("Order could not be placed.");
                }
                finally
                {
                    connection.Close();
                }
            }
        }
    }
}

/// <summary>
/// Clears the form data so another new account can be created.
/// </summary>
private void btnAddAnotherAccount_Click(object sender, EventArgs e)
{
    this.ClearForm();
}

/// <summary>
/// Closes the form/dialog box.
/// </summary>
private void btnAddFinish_Click(object sender, EventArgs e)
{
    this.Close();
}
```

```
''' <summary>
''' Creates a new account by executing the Sales.uspNewCustomer
''' stored procedure on the database.
''' </summary>
Private Sub btnCreateAccount_Click(sender As Object, e As EventArgs) Handles btnCreateAccount.Click

    ' Ensure a customer name has been entered.
    If IsCustomerNameValid Then

        ' Create the SqlConnection object.
        Using connection As New SqlConnection(My.Settings.connString)

            ' Create a SqlCommand, and identify the command type as a stored procedure.
            Using sqlCommand As New SqlCommand("Sales.uspNewCustomer", connection)
                sqlCommand.CommandType = CommandType.StoredProcedure

                ' Add the customer name input parameter for the stored procedure.
                sqlCommand.Parameters.Add(New SqlParameter("@CustomerName", SqlDbType.NVarChar, 40))
                sqlCommand.Parameters("@CustomerName").Value = txtCustomerName.Text

                ' Add the customer ID output parameter.
                sqlCommand.Parameters.Add(New SqlParameter("@CustomerID", SqlDbType.Int))
                sqlCommand.Parameters("@CustomerID").Direction = ParameterDirection.Output
```

```
                Try
                    ' Open the connection.
                    connection.Open()

                    ' Run the stored procedure.
                    sqlCommand.ExecuteNonQuery()

                    ' Convert the Customer ID value to an Integer.
                    Me.parsedCustomerID = CInt(sqlCommand.Parameters("@CustomerID").Value)

                    ' Insert the customer ID into the corresponding text box.
                    Me.txtCustomerID.Text = Convert.ToString(parsedCustomerID)
                Catch
                    MessageBox.Show("Customer ID was not returned. Account could not be created.")
                Finally
                    ' Close the connection.
                    connection.Close()
                End Try
            End Using
        End Using
    End If
End Sub

''' <summary>
''' Places the order by executing the Sales.uspPlaceNewOrder
''' stored procedure on the database.
''' </summary>
Private Sub btnPlaceOrder_Click(sender As Object, e As EventArgs) Handles btnPlaceOrder.Click

    If IsOrderDataValid() Then

        ' Create the connection.
        Using connection As New SqlConnection(My.Settings.connString)

            ' Create SqlCommand and identify it as a stored procedure.
            Using sqlCommand As New SqlCommand("Sales.uspPlaceNewOrder", connection)
                sqlCommand.CommandType = CommandType.StoredProcedure

                ' Add the @CustomerID parameter, which was an output parameter from uspNewCustomer.
                sqlCommand.Parameters.Add(New SqlParameter("@CustomerID", SqlDbType.Int))
                sqlCommand.Parameters("@CustomerID").Value = Me.parsedCustomerID

                ' Add the @OrderDate parameter.
                sqlCommand.Parameters.Add(New SqlParameter("@OrderDate", SqlDbType.DateTime, 8))
                sqlCommand.Parameters("@OrderDate").Value = dtpOrderDate.Value

                ' Add the @Amount parameter.
                sqlCommand.Parameters.Add(New SqlParameter("@Amount", SqlDbType.Int))
                sqlCommand.Parameters("@Amount").Value = numOrderAmount.Value

                ' Add the @Status parameter. For a new order, the status is always O (open).
                sqlCommand.Parameters.Add(New SqlParameter("@Status", SqlDbType.[Char], 1))
                sqlCommand.Parameters("@Status").Value = "O"

                ' Add a return value parameter for the stored procedure, which is the orderID.
                sqlCommand.Parameters.Add(New SqlParameter("@RC", SqlDbType.Int))
                sqlCommand.Parameters("@RC").Direction = ParameterDirection.ReturnValue

                Try
                    ' Open connection.
                    connection.Open()

                    ' Run the stored procedure.
                    sqlCommand.ExecuteNonQuery()

                    ' Display the order number.
                    Me.orderID = CInt(sqlCommand.Parameters("@RC").Value)
                    MessageBox.Show("Order number " & (Me.orderID).ToString & " has been submitted.")
                Catch
```

```
                    ' A simple catch.
                    MessageBox.Show("Order could  not be placed.")
                Finally
                    ' Always close a connection after you finish using it,
                    ' so that it can be released to the connection pool.
                    connection.Close()
                End Try
            End Using
        End Using
    End If
End Sub

''' <summary>
''' Resets the form for another new account.
''' </summary>
Private Sub btnAddAnotherAccount_Click(sender As Object, e As EventArgs) Handles
btnAddAnotherAccount.Click
    Me.ClearForm()
End Sub

''' <summary>
''' Closes the NewCustomer form and returns focus to the Navigation form.
''' </summary>
Private Sub btnAddFinish_Click(sender As Object, e As EventArgs) Handles btnAddFinish.Click
    Me.Close()
End Sub
```

FillOrCancel 폼

FillOrCancel 폼은 주문 ID를 입력 하 고 주문 **찾기** 단추를 클릭할 때 주문을 반환 하는 쿼리를 실행 합니다. 반환되는 행은 읽기 전용 데이터 표에 표시됩니다. 주문 **취소** 단추를 선택 하는 경우 주문을 취소 됨 (X)으로 표시할 수 있습니다. 또는 **주문 입력** 단추를 선택 하는 경우 순서를 채워진 (F)으로 표시할 수 있습니다. **주문 찾기** 단추를 다시 선택 하면 업데이트 된 행이 나타납니다.

자동 생성 된 이벤트 처리기 만들기

단추를 두 번 클릭 하 여 FillOrCancel 폼의 네 단추에 대해 빈 Click 이벤트 처리기를 만듭니다. 단추를 두 번 클릭 하면 단추 클릭으로 이벤트를 발생 시킬 수 있는 자동 생성 코드도 디자이너 코드 파일에 추가 됩니다.

FillOrCancel 폼 논리에 대 한 코드 추가

FillOrCancel 폼 논리를 완료 하려면 다음 단계를 수행 합니다.

1. 멤버의 이름을 정규화 할 필요가 없도록 다음 두 가지 네임 스페이스를 범위로 가져옵니다.

```
using System.Data.SqlClient;
using System.Text.RegularExpressions;
```

```
Imports System.Data.SqlClient
Imports System.Text.RegularExpressions
```

2. 다음 코드와 같이 변수와 도우미 메서드를 클래스에 추가 합니다.

```
// Storage for the order ID value.
private int parsedOrderID;

/// <summary>
/// Verifies that an order ID is present and contains valid characters.
/// </summary>
private bool IsOrderIDValid()
{
    // Check for input in the Order ID text box.
    if (txtOrderID.Text == "")
    {
        MessageBox.Show("Please specify the Order ID.");
        return false;
    }

    // Check for characters other than integers.
    else if (Regex.IsMatch(txtOrderID.Text, @"^\D*$"))
    {
        // Show message and clear input.
        MessageBox.Show("Customer ID must contain only numbers.");
        txtOrderID.Clear();
        return false;
    }
    else
    {
        // Convert the text in the text box to an integer to send to the database.
        parsedOrderID = Int32.Parse(txtOrderID.Text);
        return true;
    }
}
```

```
' Storage for OrderID.
Private parsedOrderID As Integer

''' <summary>
''' Verifies that OrderID is valid.
''' </summary>
Private Function IsOrderIDValid() As Boolean

    ' Check for input in the Order ID text box.
    If txtOrderID.Text = "" Then
        MessageBox.Show("Please specify the Order ID.")
        Return False

        ' Check for characters other than integers.
    ElseIf Regex.IsMatch(txtOrderID.Text, "^\D*$") Then
        ' Show message and clear input.
        MessageBox.Show("Please specify integers only.")
        txtOrderID.Clear()
        Return False
    Else
        ' Convert the text in the text box to an integer to send to the database.
        parsedOrderID = Int32.Parse(txtOrderID.Text)
        Return True
    End If
End Function
```

3. 다음 코드와 같이 4 개의 단추 클릭 이벤트 처리기에 대 한 메서드 본문을 완성 합니다.

```
/// <summary>
/// Executes a t-SQL SELECT statement to obtain order data for a specified
/// order ID, then displays it in the DataGridView on the form.
/// </summary>
private void btnFindByOrderID_Click(object sender, EventArgs e)
```

```
{
    if (IsOrderIDValid())
    {
        using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
        {
            // Define a t-SQL query string that has a parameter for orderID.
            const string sql = "SELECT * FROM Sales.Orders WHERE orderID = @orderID";

            // Create a SqlCommand object.
            using (SqlCommand sqlCommand = new SqlCommand(sql, connection))
            {
                // Define the @orderID parameter and set its value.
                sqlCommand.Parameters.Add(new SqlParameter("@orderID", SqlDbType.Int));
                sqlCommand.Parameters["@orderID"].Value = parsedOrderID;

                try
                {
                    connection.Open();

                    // Run the query by calling ExecuteReader().
                    using (SqlDataReader dataReader = sqlCommand.ExecuteReader())
                    {
                        // Create a data table to hold the retrieved data.
                        DataTable dataTable = new DataTable();

                        // Load the data from SqlDataReader into the data table.
                        dataTable.Load(dataReader);

                        // Display the data from the data table in the data grid view.
                        this.dgvCustomerOrders.DataSource = dataTable;

                        // Close the SqlDataReader.
                        dataReader.Close();
                    }
                }
                catch
                {
                    MessageBox.Show("The requested order could not be loaded into the form.");
                }
                finally
                {
                    // Close the connection.
                    connection.Close();
                }
            }
        }
    }
}

/// <summary>
/// Cancels an order by calling the Sales.uspCancelOrder
/// stored procedure on the database.
/// </summary>
private void btnCancelOrder_Click(object sender, EventArgs e)
{
    if (IsOrderIDValid())
    {
        // Create the connection.
        using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
        {
            // Create the SqlCommand object and identify it as a stored procedure.
            using (SqlCommand sqlCommand = new SqlCommand("Sales.uspCancelOrder", connection))
            {
                sqlCommand.CommandType = CommandType.StoredProcedure;

                // Add the order ID input parameter for the stored procedure.
                sqlCommand.Parameters.Add(new SqlParameter("@orderID", SqlDbType.Int));
                sqlCommand.Parameters["@orderID"].Value = parsedOrderID;
```

```
                try
                {
                    // Open the connection.
                    connection.Open();

                    // Run the command to execute the stored procedure.
                    sqlCommand.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("The cancel operation was not completed.");
                }
                finally
                {
                    // Close connection.
                    connection.Close();
                }
            }
        }
    }
}

/// <summary>
/// Fills an order by calling the Sales.uspFillOrder stored
/// procedure on the database.
/// </summary>
private void btnFillOrder_Click(object sender, EventArgs e)
{
    if (IsOrderIDValid())
    {
        // Create the connection.
        using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
        {
            // Create command and identify it as a stored procedure.
            using (SqlCommand sqlCommand = new SqlCommand("Sales.uspFillOrder", connection))
            {
                sqlCommand.CommandType = CommandType.StoredProcedure;

                // Add the order ID input parameter for the stored procedure.
                sqlCommand.Parameters.Add(new SqlParameter("@orderID", SqlDbType.Int));
                sqlCommand.Parameters["@orderID"].Value = parsedOrderID;

                // Add the filled date input parameter for the stored procedure.
                sqlCommand.Parameters.Add(new SqlParameter("@FilledDate", SqlDbType.DateTime, 8));
                sqlCommand.Parameters["@FilledDate"].Value = dtpFillDate.Value;

                try
                {
                    connection.Open();

                    // Execute the stored procedure.
                    sqlCommand.ExecuteNonQuery();
                }
                catch
                {
                    MessageBox.Show("The fill operation was not completed.");
                }
                finally
                {
                    // Close the connection.
                    connection.Close();
                }
            }
        }
    }
}

/// <summary>
/// Closes the form.
```

```
/// </summary>
private void btnFinishUpdates_Click(object sender, EventArgs e)
{
    this.Close();
}
```

```
''' <summary>
''' Executes a t-SQL SELECT query on the database to
''' obtain order data for a specified order ID.
''' </summary>
Private Sub btnFindByOrderID_Click(sender As Object, e As EventArgs) Handles btnFindByOrderID.Click

    ' Prepare the connection and the command.
    If IsOrderIDValid() Then

        ' Create the connection.
        Using connection As New SqlConnection(My.Settings.connString)

            ' Define the query string that has a parameter for orderID.
            Const sql As String = "select * from Sales.Orders where orderID = @orderID"

            ' Create a SqlCommand object.
            Using sqlCommand As New SqlCommand(sql, connection)

                ' Define the @orderID parameter and its value.
                sqlCommand.Parameters.Add(New SqlParameter("@orderID", SqlDbType.Int))
                sqlCommand.Parameters("@orderID").Value = parsedOrderID

                Try
                    ' Open connection.
                    connection.Open()

                    ' Execute the query.
                    Dim dataReader As SqlDataReader = sqlCommand.ExecuteReader()

                    ' Create a data table to hold the retrieved data.
                    Dim dataTable As New DataTable()

                    ' Load the data from SqlDataReader into the data table.
                    dataTable.Load(dataReader)

                    ' Display the data from the data table in the data grid view.
                    Me.dgvCustomerOrders.DataSource = dataTable

                    ' Close the SqlDataReader.
                    dataReader.Close()
                Catch
                    MessageBox.Show("The requested order could not be loaded into the form.")
                Finally
                    ' Close the connection.
                    connection.Close()
                End Try
            End Using
        End Using
    End If
End Sub

''' <summary>
''' Fills an order by running the Sales.uspFillOrder stored procedure on the database.
''' </summary>
Private Sub btnFillOrder_Click(sender As Object, e As EventArgs) Handles btnFillOrder.Click

    ' Set up and run stored procedure only if OrderID is valid.
    If IsOrderIDValid() Then

        ' Create the connection.
        Using connection As New SqlConnection(My.Settings.connString)
```

```
            ' Create command and identify it as a stored procedure.
            Using sqlCommand As New SqlCommand("Sales.uspFillOrder", connection)

                sqlCommand.CommandType = CommandType.StoredProcedure

                ' Add input parameter for the stored procedure.
                sqlCommand.Parameters.Add(New SqlParameter("@orderID", SqlDbType.Int))
                sqlCommand.Parameters("@orderID").Value = parsedOrderID

                ' Add second input parameter.
                sqlCommand.Parameters.Add(New SqlParameter("@FilledDate", SqlDbType.DateTime, 8))
                sqlCommand.Parameters("@FilledDate").Value = dtpFillDate.Value

                Try
                    ' Open the connection.
                    connection.Open()

                    ' Run the SqlCommand.
                    sqlCommand.ExecuteNonQuery()
                Catch
                    ' A simple catch.
                    MessageBox.Show("The fill operation was not completed.")
                Finally
                    ' Close the connection.
                    connection.Close()
                End Try
            End Using
        End Using
    End If
End Sub

''' <summary>
''' Cancels an order by running the Sales.uspCancelOrder stored procedure on the database.
''' </summary>
Private Sub btnCancelOrder_Click(sender As Object, e As EventArgs) Handles btnCancelOrder.Click

    ' Set up and run the stored procedure only if OrderID is ready.
    If IsOrderIDValid() Then

        ' Create the connection.
        Using connection As New SqlConnection(My.Settings.connString)

            ' Create the command and identify it as a stored procedure.
            Using sqlCommand As New SqlCommand("Sales.uspCancelOrder", connection)
                sqlCommand.CommandType = CommandType.StoredProcedure

                ' Add input parameter for the stored procedure.
                sqlCommand.Parameters.Add(New SqlParameter("@orderID", SqlDbType.Int))
                sqlCommand.Parameters("@orderID").Value = parsedOrderID

                Try
                    ' Open the connection.
                    connection.Open()

                    ' Run the SqlCommand.
                    sqlCommand.ExecuteNonQuery()
                Catch
                    ' A simple catch.
                    MessageBox.Show("The cancel operation was not completed.")
                Finally
                    ' Close connection.
                    connection.Close()
                End Try
            End Using
        End Using
    End If
End Sub
```

```
''' <summary>
''' Closes the form and returns focus to the Navigation form.
''' </summary>
Private Sub btnFinishUpdates_Click(sender As Object, e As EventArgs) Handles btnFinishUpdates.Click
    Me.Close()
End Sub
```

응용 프로그램 테스트

각 Click 이벤트 처리기를 코딩하고 코딩을 마친 후에 **F5** 키를 선택하여 애플리케이션을 빌드하고 테스트합니다.

참조

- .NET용 Visual Studio 데이터 도구

WPF 및 Entity Framework 6을 사용하여 간단한 데이터 애플리케이션 만들기

2020-06-08 • 38 minutes to read • Edit Online

이 연습에서는 Visual Studio에서 기본 "데이터 폼" 응용 프로그램을 만드는 방법을 보여 줍니다. 앱은 SQL Server LocalDB, Northwind 데이터베이스, Entity Framework 6 및 Windows Presentation Foundation를 사용 합니다. 마스터-세부 보기를 사용 하 여 기본 데이터 바인딩을 수행 하는 방법을 보여 줍니다. 또한 **다음으로 이동**, **이전**으로 이동, **처음으로**이동, **끝으로 이동**, **업데이트** 및 **삭제** 단추가 포함 된 사용자 지정 바인딩 탐색기가 있습니다.

이 문서에서는 Visual Studio에서 데이터 도구를 사용 하는 방법을 집중적으로 설명 하며, 기본 기술에 대해 설명 하지 않습니다. XAML, Entity Framework 및 SQL에 대 한 기본적인 지식이 있다고 가정 합니다. 또한이 예제에서는 WPF 응용 프로그램의 표준인 MVVM (모델-뷰-ViewModel) 아키텍처를 보여 주지 않습니다. 그러나이 코드를 수정 하지 않은 사용자 고유의 MVVM 응용 프로그램으로 복사할 수 있습니다.

Northwind에 설치 및 연결

이 예에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다. 해당 제품에 대 한 ADO.NET 데이터 공급자가 Entity Framework를 지 원하는 경우 다른 SQL 데이터베이스 제품과 함께 작동 해야 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **.NET 데스크톱 개발** 워크로드의 일부로 또는 개별 구성 요소로서 SQL Server Express LocalDB를 설치할 수 있습니다.

2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. **SQL Server 개체 탐색기** 는 **데이터 저장소 및 처리** 워크 로드의 일부로 **Visual Studio 설치 관리자**에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

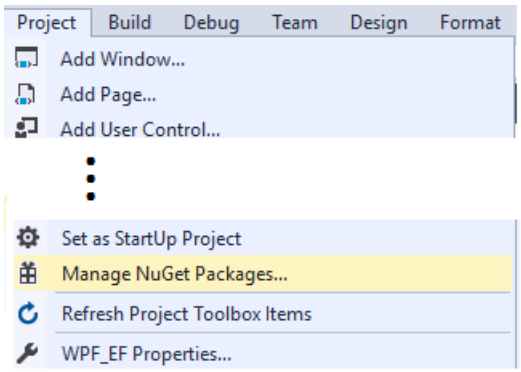
      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

3. Northwind에 대 한 새 연결을 추가 합니다.

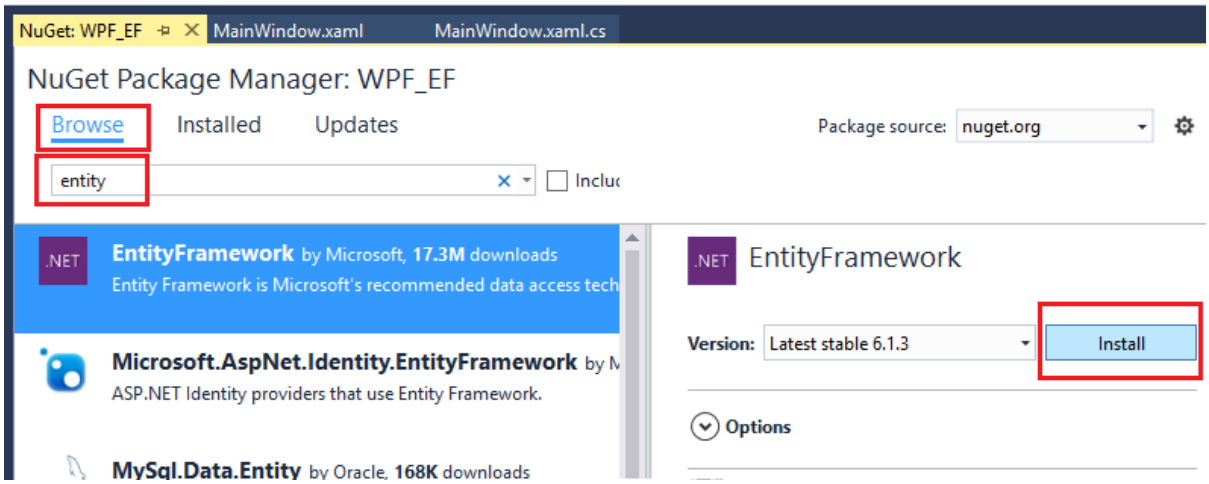
프로젝트 구성

1. Visual Studio에서 새 c # **WPF 앱** 프로젝트를 만듭니다.

2. Entity Framework 6 용 NuGet 패키지를 추가 합니다. **솔루션 탐색기**에서 프로젝트 노드를 선택 합니다. 주 메뉴에서 **프로젝트** > **NuGet 패키지 관리**를 선택 합니다.



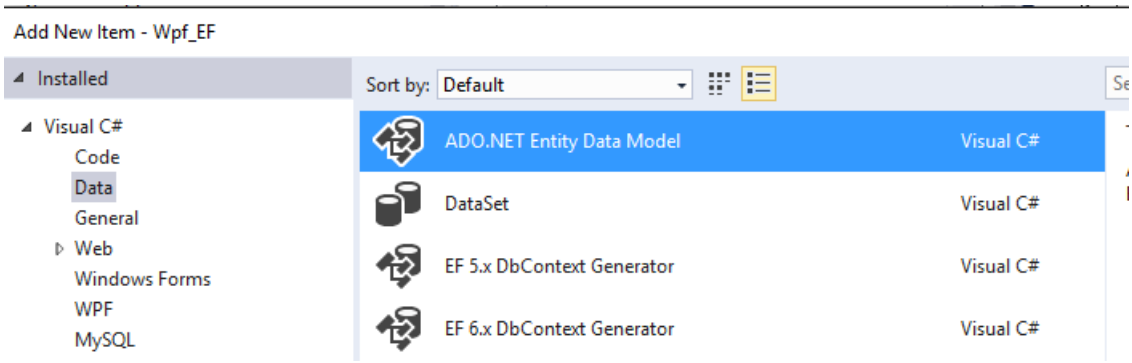
3. **NuGet 패키지 관리자**에서 **찾아보기** 링크를 클릭 합니다. Entity Framework은 목록의 맨 위 패키지인 것입니다. 오른쪽 창에서 **설치** 를 클릭 하 고 프롬프트를 따릅니다. 설치가 완료 되 면 출력 창에 사용자에 게 알려 줍니다.



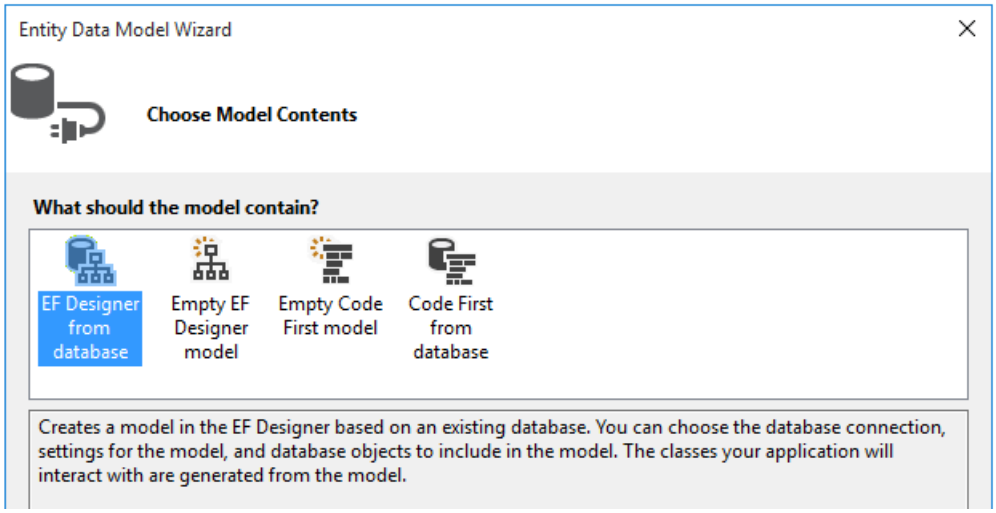
4. 이제 Visual Studio를 사용 하 여 Northwind 데이터베이스를 기반으로 모델을 만들 수 있습니다.

모델 만들기

1. **솔루션 탐색기** 에서 프로젝트 노드를 마우스 오른쪽 단추로 클릭 하 고 **Add** > **새 항목**추가를 선택 합니다. 왼쪽 창의 c # 노드 아래에서 **데이터** 를 선택 하 고 가운데 창에서 **ADO.NET 엔터티 데이터 모델**를 선택 합니다.

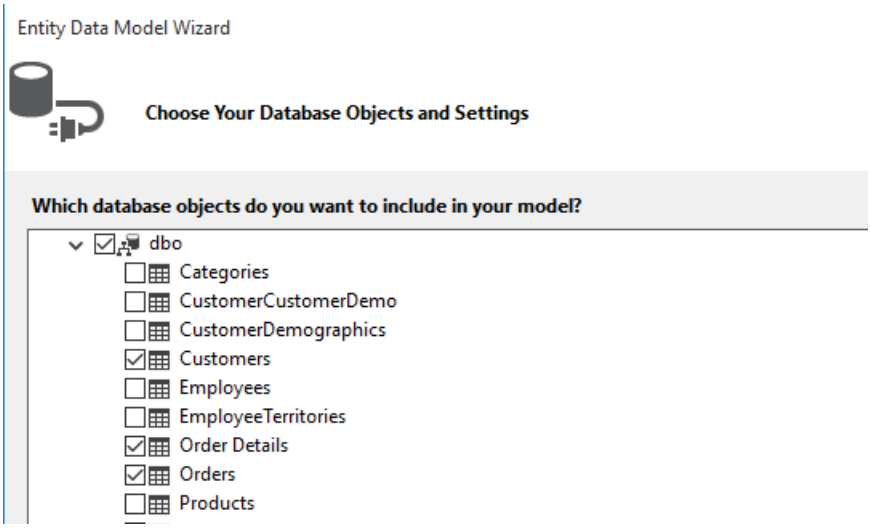


2. 모델을 호출 `Northwind_model` 하 고 **확인**을선택 합니다. **엔터티 데이터 모델 마법사** 가 열립니다. **데이터베이스에서 EF Designer** 를 선택 하 고 **다음**을 클릭 합니다.

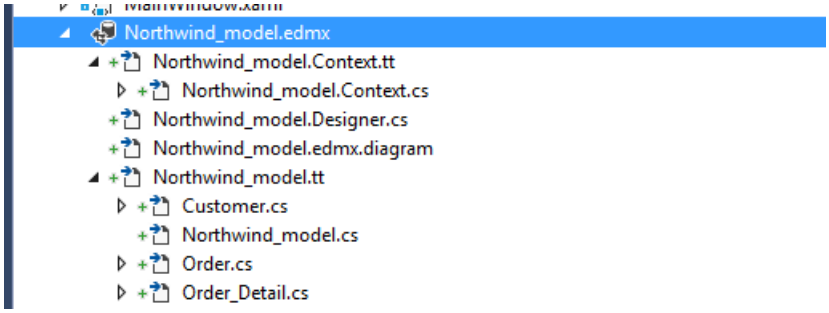


3. 다음 화면에서 LocalDB Northwind 연결 (예: (localdb) \MSSQLLocalDB)을 입력 하거나 선택 하 고 Northwind 데이터베이스를 지정한 후 **다음**을 클릭 합니다.

4. 마법사의 다음 페이지에서 Entity Framework 모델에 포함할 테이블, 저장 프로시저 및 기타 데이터베이스 개체를 선택 합니다. 트리 뷰에서 dbo 노드를 확장 하 고 **고객**, **주문** 및 **주문 세부 정보**를 선택 합니다. 기본값을 선택 된 채로 두고 **마침**을 클릭 합니다.



5. 마법사는 Entity Framework 모델을 나타내는 c # 클래스를 생성 합니다. 클래스는 일반 c # 클래스 이며 WPF 사용자 인터페이스에 바인딩하는 것입니다. *.Edmx* 파일은 클래스를 데이터베이스의 개체와 연결 하는 관계 및 기타 메타 데이터를 설명 합니다. *.Tt* 파일은 모델에서 작동 하는 코드를 생성 하 고 변경 내용을 데이터베이스에 저장 하는 T4 템플릿입니다. Northwind_model 노드 아래 **솔루션 탐색기** 에서 이러한 모든 파일을 볼 수 있습니다.



.Edmx 파일의 디자이너 화면에서는 모델의 일부 속성 및 관계를 수정할 수 있습니다. 이 연습에서는 디자이너를 사용 하지 않겠습니다.

6. *.Tt* 파일은 일반적인 용도로, 이러한 파일 중 하나를 조정 하 여 WPF 데이터 바인딩 작업을 수행 해야 합니다. 이 경우에는 ObservableCollections가 필요 합니다. **솔루션 탐색기**에서 *Northwind_model*를 찾을 때까지

Northwind_model 노드를 확장 합니다. (에 있지 않은지 확인 *합니다. Context.tt* 파일은 *.edmx* 파일 바로 아래에 있습니다.

- 의 두 항목을 ICollection 로 바꿉니다 ObservableCollection<T> .
- 첫 번째 항목을 HashSet<T> ObservableCollection<T> 51 줄로 바꿉니다. HashSet의 두 번째 항목을 바꾸지 마십시오.
- (431 줄)의만 발생 하는를 System.Collections.Generic 로 바꿉니다 System.Collections.ObjectModel .

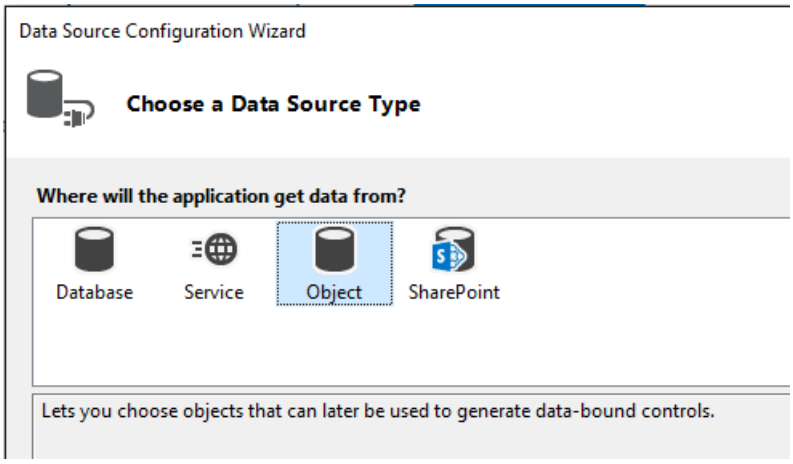
7. **Ctrl** + **Shift** + **B** 를 눌러 프로젝트를 빌드합니다. 빌드가 완료 되 면 모델 클래스는 데이터 소스 마법사에 표시 됩니다.

이제이 모델을 XAML 페이지에 연결 하 여 데이터를 보고, 탐색 하 고, 수정할 수 있습니다.

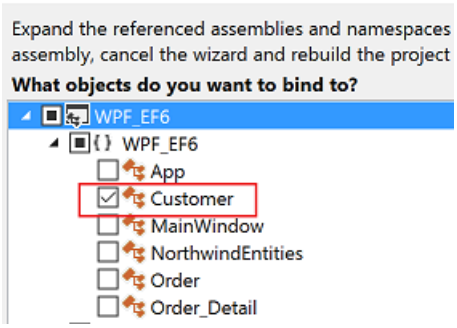
모델을 XAML 페이지에 Databind

사용자 고유의 데이터 바인딩 코드를 작성할 수 있지만 Visual Studio에서 사용자에 게 더 쉽게 수행할 수 있습니다.

1. 주 메뉴에서 **프로젝트** > **새 데이터 소스 추가** 를 선택 하 여 **데이터 소스 구성 마법사**를 엽니다. 데이터베이스가 아니라 모델 클래스에 바인딩되어 있기 때문에 **개체** 를 선택 합니다.



2. 프로젝트에 대 한 노드를 확장 하 고 **Customer**를 선택 합니다. 주문의 원본은 Customer의 Orders 탐색 속성에서 자동으로 생성 됩니다.

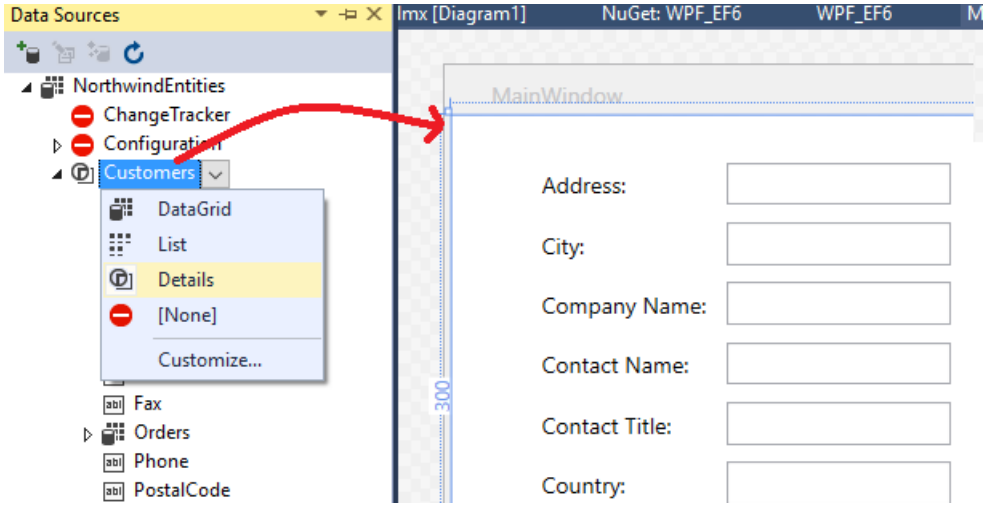


3. **마침**을 클릭합니다.

4. 코드 보기에서 *mainwindow.xaml* 로 이동 합니다. 이 예제의 목적을 위해 XAML을 간단 하 게 유지 합니다. Mainwindow.xaml의 제목을 좀 더 설명적인 이름으로 변경 하 고, 현재의 높이와 너비를 600 x 800로 늘립니다. 나중에 언제 든 지 변경할 수 있습니다. 이제 다음 세 개의 행 정의를 주 그리드에 추가 하 고, 탐색 단추를 위한 행 하나, 고객 세부 정보에 대 한 행, 주문을 표시 하는 그리드에 대 한 행을 추가 합니다.

```
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
```

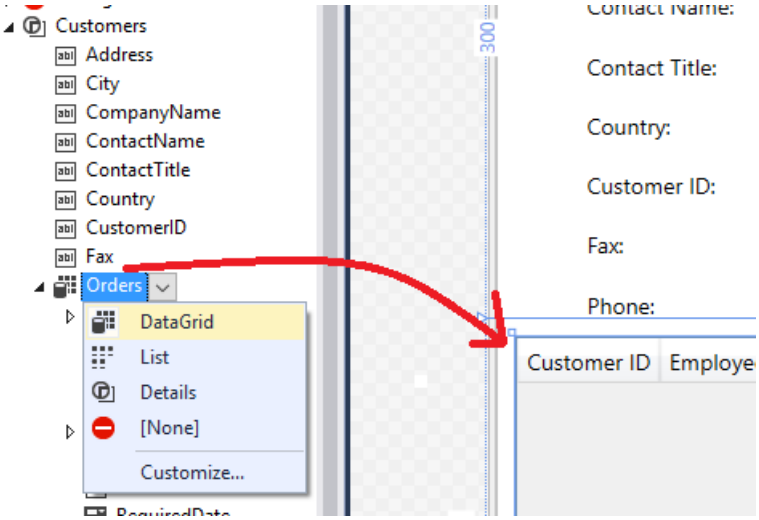
5. 이제 디자이너에서 볼 수 있도록 *mainwindow.xaml* 을 엽니다. 이렇게 하면 **데이터 소스** 창이 **도구 상자**옆의 Visual Studio 창 여백에 옵션으로 표시 됩니다. 탭을 클릭 하 여 창을 열거나 **Shift** + **Alt** + **D** 를 누르거나 **View** > **다른 Windows** > **데이터 원본**보기를 선택 합니다. Customers 클래스의 각 속성을 자체의 개별 텍스트 상자에 표시할 예정입니다. 먼저 **Customers** 콤보 상자의 화살표를 클릭 하 고 **세부 정보**를 선택 합니다. 그런 다음 디자이너에서 가운데 행으로 이동할 것임을 알 수 있도록 노드를 디자인 화면의 가운데 부분으로 끕니다. 잃어버리지 경우 나중에 XAML에서 수동으로 행을 지정할 수 있습니다. 기본적으로 컨트롤은 grid 요소에 세로로 배치 되지만이 시점에서 폼에서 원하는 대로 정렬할 수 있습니다. 예를 들어 **이름** 텍스트 상자를 주소 위의 위쪽에 배치 하는 것이 적합할 수 있습니다. 이 문서의 샘플 응용 프로그램은 필드의 순서를 다시 정렬 하 고 두 개의 열로 다시 정렬 합니다.



이제 코드 보기에서 `Grid` 부모 표의 행 1 (중간 행)에 새 요소를 볼 수 있습니다. 부모 표에는 `DataContext` 요소에 추가 된 CollectionViewSource를 참조 하는 특성이 있습니다 `Windows.Resources` . 해당 데이터 컨텍스트가 지정 된 경우 첫 번째 텍스트 상자를 **Address**에 바인딩하면 해당 이름이 `Address` `Customer` collectionviewsource의 현재 개체의 속성에 매핑됩니다.

```
<Grid DataContext="{StaticResource customerViewSource}">
```

6. 고객이 창의 위쪽 절반에 표시 되 면 해당 주문을 아래쪽 절반에 표시 하려고 합니다. 단일 그리드 뷰 컨트롤에 주문을 표시 합니다. 마스터-세부 데이터 바인딩이 예상 대로 작동 하려면 별도의 Orders 노드가 아닌 Customers 클래스의 Orders 속성에 바인딩해야 합니다. Customers 클래스의 Orders 속성을 폼의 아래쪽으로 끌어 놓으면 디자이너는 2 행에 배치 됩니다.



7. Visual Studio는 UI 컨트롤을 모델의 이벤트에 연결 하는 모든 바인딩 코드를 생성 했습니다. 일부 데이터를 확인 하려면 모델을 채우는 코드를 작성 해야 합니다. 먼저 *MainWindow.xaml.cs* 로 이동 하 여 데이터 컨텍스트의 mainwindow.xaml 클래스에 데이터 멤버를 추가 합니다. 사용자를 위해 생성 된이 개체는 모델의 변경 내용 및 이벤트를 추적 하는 컨트롤 처럼 동작 합니다. 또한 고객과 주문의 CollectionViewSource 데이터 멤버와 관련 생성자 초기화 논리를 추가 합니다. 클래스의 맨 위는 다음과 같습니다.

```
public partial class MainWindow : Window
{
    NorthwindEntities context = new NorthwindEntities();
    CollectionViewSource custViewSource;
    CollectionViewSource ordViewSource;

    public MainWindow()
    {
        InitializeComponent();
        custViewSource = ((CollectionViewSource)(FindResource("customerViewSource")));
        ordViewSource = ((CollectionViewSource)(FindResource("customerOrdersViewSource")));
        DataContext = this;
    }
```

`using` Load 확장 메서드를 범위로 가져오기 위해 system.object에 대 한 지시문을 추가 합니다.

```
using System.Data.Entity;
```

이제 아래로 스크롤하고 이벤트 처리기를 찾습니다 `Window_Loaded` . Visual Studio에서 CollectionViewSource 개체를 추가 했습니다. 모델을 만들 때 선택한 창의 northwindentities 개체를 나타냅니다. 이를 이미 추가 했으므로 여기에 필요 하지 않습니다. 이제 메서드가 다음과 같이 표시 되도록에서 코드를 바꿉니다 `Window_Loaded` .

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    // Load is an extension method on IQueryable,
    // defined in the System.Data.Entity namespace.
    // This method enumerates the results of the query,
    // similar to ToList but without creating a list.
    // When used with Linq to Entities, this method
    // creates entity objects and adds them to the context.
    context.Customers.Load();

    // After the data is loaded, call the DbSet<T>.Local property
    // to use the DbSet<T> as a binding source.
    custViewSource.Source = context.Customers.Local;
}
```

8. **F5**키를 누릅니다. CollectionViewSource로 검색 된 첫 번째 고객에 대 한 세부 정보가 표시 됩니다. 또한 데이터 표에 해당 주문이 표시 되어야 합니다. 서식 지정이 유용 하지 않으므로이를 해결 해 보겠습니다. 다른 레코드를 보고 기본 CRUD 작업을 수행 하는 방법도 만들 수 있습니다.

새 고객과 주문에 대해 페이지 디자인 조정 및 그리드 추가

Visual Studio에서 생성 되는 기본 정렬은 응용 프로그램에 적합 하지 않으므로 코드에 복사할 최종 XAML을 여기에 제공 합니다. 사용자가 새 고객 또는 주문을 추가할 수 있도록 일부 "양식" (실제로는 그리드)도 필요 합니다. 새 고객과 주문을 추가할 수 있으려면에 데이터 바인딩되지 않은 별도의 텍스트 상자 집합이 필요 `CollectionViewSource` 합니다. 처리기 메서드에서 Visible 속성을 설정 하 여 언제 든 지 사용자가 볼 수 있는 그리드를 제어 합니다. 마지막으로 주문 표에 있는 각 행에 삭제 단추를 추가 하 여 사용자가 개별 주문을 삭제할 수 있게 합니다.

먼저 Mainwindow.xaml의 요소에 이러한 스타일을 추가 `Windows.Resources` *합니다*.

```
<Style x:Key="Label" TargetType="{x:Type Label}" BasedOn="{x:Null}">
    <Setter Property="HorizontalAlignment" Value="Left"/>
    <Setter Property="VerticalAlignment" Value="Center"/>
    <Setter Property="Margin" Value="3"/>
    <Setter Property="Height" Value="23"/>
</Style>
<Style x:Key="CustTextBox" TargetType="{x:Type TextBox}" BasedOn="{x:Null}">
    <Setter Property="HorizontalAlignment" Value="Right"/>
    <Setter Property="VerticalAlignment" Value="Center"/>
    <Setter Property="Margin" Value="3"/>
    <Setter Property="Height" Value="26"/>
    <Setter Property="Width" Value="120"/>
</Style>
```

다음으로 전체 외부 그리드를이 태그로 바꿉니다.

```
<Grid>
     <Grid.RowDefinitions>
         <RowDefinition Height="auto"/>
         <RowDefinition Height="auto"/>
         <RowDefinition Height="*"/>
     </Grid.RowDefinitions>
     <Grid x:Name="existingCustomerGrid" Grid.Row="1" HorizontalAlignment="Left" Margin="5"
Visibility="Visible" VerticalAlignment="Top" Background="AntiqueWhite" DataContext="{StaticResource
customerViewSource}">
         <Grid.ColumnDefinitions>
             <ColumnDefinition Width="Auto" MinWidth="233"/>
             <ColumnDefinition Width="Auto" MinWidth="397"/>
         </Grid.ColumnDefinitions>
         <Grid.RowDefinitions>
             <RowDefinition Height="Auto"/>
             <RowDefinition Height="Auto"/>
```

```
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Customer ID:" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="customerIDTextBox" Grid.Row="0" Style="{StaticResource CustTextBox}"
                 Text="{Binding CustomerID, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Company Name:" Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="companyNameTextBox" Grid.Row="1" Style="{StaticResource CustTextBox}"
                 Text="{Binding CompanyName, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Contact Name:" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="contactNameTextBox" Grid.Row="2" Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactName, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Contact title:" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="contactTitleTextBox" Grid.Row="3" Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactTitle, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Address:" Grid.Row="4" Style="{StaticResource Label}"/>
        <TextBox x:Name="addressTextBox" Grid.Row="4" Style="{StaticResource CustTextBox}"
                 Text="{Binding Address, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="City:" Grid.Column="1" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="cityTextBox" Grid.Column="1" Grid.Row="0" Style="{StaticResource CustTextBox}"
                 Text="{Binding City, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Country:" Grid.Column="1" Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="countryTextBox" Grid.Column="1" Grid.Row="1" Style="{StaticResource CustTextBox}"
                 Text="{Binding Country, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Fax:" Grid.Column="1" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="faxTextBox" Grid.Column="1" Grid.Row="2" Style="{StaticResource CustTextBox}"
                 Text="{Binding Fax, Mode=TwoWay, NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
        <Label Content="Phone:" Grid.Column="1" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="phoneTextBox" Grid.Column="1" Grid.Row="3" Style="{StaticResource CustTextBox}"
                 Text="{Binding Phone, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Postal Code:" Grid.Column="1" Grid.Row="4" VerticalAlignment="Center" Style="
{StaticResource Label}"/>
        <TextBox x:Name="postalCodeTextBox" Grid.Column="1" Grid.Row="4" Style="{StaticResource CustTextBox}"
                 Text="{Binding PostalCode, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Region:" Grid.Column="1" Grid.Row="5" Style="{StaticResource Label}"/>
        <TextBox x:Name="regionTextBox" Grid.Column="1" Grid.Row="5" Style="{StaticResource CustTextBox}"
                 Text="{Binding Region, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <Grid x:Name="newCustomerGrid" Grid.Row="1" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="5"
DataContext="{Binding RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Window}},
Path=newCustomer, UpdateSourceTrigger=Explicit}" Visibility="Collapsed" Background="CornflowerBlue">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" MinWidth="233"/>
            <ColumnDefinition Width="Auto" MinWidth="397"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="Customer ID:" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_customerIDTextBox" Grid.Row="0" Style="{StaticResource CustTextBox}"
                 Text="{Binding CustomerID, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
```

```
        <Label Content="Company Name:" Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_companyNameTextBox" Grid.Row="1" Style="{StaticResource CustTextBox}"
                 Text="{Binding CompanyName, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true }"/>
        <Label Content="Contact Name:" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_contactNameTextBox" Grid.Row="2" Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactName, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Contact title:" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_contactTitleTextBox" Grid.Row="3" Style="{StaticResource CustTextBox}"
                 Text="{Binding ContactTitle, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Address:" Grid.Row="4" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_addressTextBox" Grid.Row="4" Style="{StaticResource CustTextBox}"
                 Text="{Binding Address, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="City:" Grid.Column="1" Grid.Row="0" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_cityTextBox" Grid.Column="1" Grid.Row="0" Style="{StaticResource CustTextBox}"
                 Text="{Binding City, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Country:" Grid.Column="1" Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_countryTextBox" Grid.Column="1" Grid.Row="1" Style="{StaticResource CustTextBox}"
                 Text="{Binding Country, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Fax:" Grid.Column="1" Grid.Row="2" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_faxTextBox" Grid.Column="1" Grid.Row="2" Style="{StaticResource CustTextBox}"
                 Text="{Binding Fax, Mode=TwoWay, NotifyOnValidationError=true, ValidatesOnExceptions=true}"/>
        <Label Content="Phone:" Grid.Column="1" Grid.Row="3" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_phoneTextBox" Grid.Column="1" Grid.Row="3" Style="{StaticResource CustTextBox}"
                 Text="{Binding Phone, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Postal Code:" Grid.Column="1" Grid.Row="4" VerticalAlignment="Center" Style="
{StaticResource Label}"/>
        <TextBox x:Name="add_postalCodeTextBox" Grid.Column="1" Grid.Row="4" Style="{StaticResource
CustTextBox}"
                 Text="{Binding PostalCode, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Region:" Grid.Column="1" Grid.Row="5" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_regionTextBox" Grid.Column="1" Grid.Row="5" Style="{StaticResource CustTextBox}"
                 Text="{Binding Region, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <Grid x:Name="newOrderGrid" Grid.Row="1" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="5"
DataContext="{Binding Path=newOrder, Mode=TwoWay}" Visibility="Collapsed" Background="LightGreen">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" MinWidth="233"/>
            <ColumnDefinition Width="Auto" MinWidth="397"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Label Content="New Order Form" FontWeight="Bold"/>
        <Label Content="Employee ID:"  Grid.Row="1" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_employeeIDTextBox" Grid.Row="1" Style="{StaticResource CustTextBox}"
                 Text="{Binding EmployeeID, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Order Date:"  Grid.Row="2" Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_orderDatePicker" Grid.Row="2"  HorizontalAlignment="Right" Width="120"
                SelectedDate="{Binding OrderDate, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Required Date:" Grid.Row="3" Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_requiredDatePicker" Grid.Row="3" HorizontalAlignment="Right" Width="120"
                 SelectedDate="{Binding RequiredDate, Mode=TwoWay, NotifyOnValidationError=true,
```

```
ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Shipped Date:"  Grid.Row="4"  Style="{StaticResource Label}"/>
        <DatePicker x:Name="add_shippedDatePicker"  Grid.Row="4"  HorizontalAlignment="Right" Width="120"
                SelectedDate="{Binding ShippedDate, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
        <Label Content="Ship Via:"  Grid.Row="5" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_ShipViaTextBox"  Grid.Row="5" Style="{StaticResource CustTextBox}"
                 Text="{Binding ShipVia, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
        <Label Content="Freight"  Grid.Row="6" Style="{StaticResource Label}"/>
        <TextBox x:Name="add_freightTextBox" Grid.Row="6" Style="{StaticResource CustTextBox}"
                 Text="{Binding Freight, Mode=TwoWay, NotifyOnValidationError=true,
ValidatesOnExceptions=true}"/>
    </Grid>
    <DataGrid x:Name="ordersDataGrid" SelectionUnit="Cell" SelectionMode="Single" AutoGenerateColumns="False"
CanUserAddRows="false" IsEnabled="True" EnableRowVirtualization="True" Width="auto" ItemsSource="{Binding
Source={StaticResource customerOrdersViewSource}}" Margin="10,10,10,10" Grid.Row="2"
RowDetailsVisibilityMode="VisibleWhenSelected">
        <DataGrid.Columns>
            <DataGridTemplateColumn>
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <Button Content="Delete" Command="{StaticResource DeleteOrderCommand}"
CommandParameter="{Binding}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="customerIDColumn" Binding="{Binding CustomerID}" Header="Customer ID"
Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="employeeIDColumn" Binding="{Binding EmployeeID}" Header="Employee ID"
Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="freightColumn" Binding="{Binding Freight}" Header="Freight"
Width="SizeToHeader"/>
            <DataGridTemplateColumn x:Name="orderDateColumn" Header="Order Date" Width="SizeToHeader">
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <DatePicker SelectedDate="{Binding OrderDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="orderIDColumn" Binding="{Binding OrderID}" Header="Order ID"
Width="SizeToHeader"/>
            <DataGridTemplateColumn x:Name="requiredDateColumn" Header="Required Date" Width="SizeToHeader">
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <DatePicker SelectedDate="{Binding RequiredDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="shipAddressColumn" Binding="{Binding ShipAddress}" Header="Ship
Address" Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="shipCityColumn" Binding="{Binding ShipCity}" Header="Ship City"
Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="shipCountryColumn" Binding="{Binding ShipCountry}" Header="Ship
Country" Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="shipNameColumn" Binding="{Binding ShipName}" Header="Ship Name"
Width="SizeToHeader"/>
            <DataGridTemplateColumn x:Name="shippedDateColumn" Header="Shipped Date" Width="SizeToHeader">
                <DataGridTemplateColumn.CellTemplate>
                    <DataTemplate>
                        <DatePicker SelectedDate="{Binding ShippedDate, Mode=TwoWay,
NotifyOnValidationError=true, ValidatesOnExceptions=true, UpdateSourceTrigger=PropertyChanged}"/>
                    </DataTemplate>
                </DataGridTemplateColumn.CellTemplate>
            </DataGridTemplateColumn>
            <DataGridTextColumn x:Name="shipPostalCodeColumn" Binding="{Binding ShipPostalCode}" Header="Ship
Postal Code" Width="SizeToHeader"/>
```

```
            <DataGridTextColumn x:Name="shipRegionColumn" Binding="{Binding ShipRegion}" Header="Ship Region"
Width="SizeToHeader"/>
            <DataGridTextColumn x:Name="shipViaColumn" Binding="{Binding ShipVia}" Header="Ship Via"
Width="SizeToHeader"/>
        </DataGrid.Columns>
    </DataGrid>
 </Grid>
```

탐색, 추가, 업데이트 및 삭제 단추를 추가 합니다.

Windows Forms 응용 프로그램에서는 데이터베이스의 행을 탐색 하 고 기본 CRUD 작업을 수행 하는 단추가 포함 된 BindingNavigator 개체를 가져옵니다. WPF는 BindingNavigator를 제공 하지 않지만 쉽게 만들 수 있습니다. 이 작업을 수행 하려면 가로 StackPanel 내의 단추를 사용 하 고 단추를 코드의 메서드에 바인딩된 명령과 연결 합니다.

명령 논리에는 (1) 명령, (2) 바인딩, (3) 단추 및 (4) 코드 숨김으로 된 명령 처리기의 네 부분이 있습니다.

XAML에서 명령**,** 바인딩 및 단추 추가

1. 먼저 *mainwindow.xaml* 파일의 요소 내에 명령을 추가 합니다 `Windows.Resources` .

   ```
   <RoutedUICommand x:Key="FirstCommand" Text="First"/>
   <RoutedUICommand x:Key="LastCommand" Text="Last"/>
   <RoutedUICommand x:Key="NextCommand" Text="Next"/>
   <RoutedUICommand x:Key="PreviousCommand" Text="Previous"/>
   <RoutedUICommand x:Key="DeleteCustomerCommand" Text="Delete Customer"/>
   <RoutedUICommand x:Key="DeleteOrderCommand" Text="Delete Order"/>
   <RoutedUICommand x:Key="UpdateCommand" Text="Update"/>
   <RoutedUICommand x:Key="AddCommand" Text="Add"/>
   <RoutedUICommand x:Key="CancelCommand" Text="Cancel"/>
   ```

2. CommandBinding은 이벤트를 `RoutedUICommand` 코드 숨김으로 메서드에 매핑합니다. `CommandBindings` 닫는 태그 뒤에이 요소를 추가 합니다 `Windows.Resources` .

   ```
   <Window.CommandBindings>
       <CommandBinding Command="{StaticResource FirstCommand}" Executed="FirstCommandHandler"/>
       <CommandBinding Command="{StaticResource LastCommand}" Executed="LastCommandHandler"/>
       <CommandBinding Command="{StaticResource NextCommand}" Executed="NextCommandHandler"/>
       <CommandBinding Command="{StaticResource PreviousCommand}" Executed="PreviousCommandHandler"/>
       <CommandBinding Command="{StaticResource DeleteCustomerCommand}"
   Executed="DeleteCustomerCommandHandler"/>
       <CommandBinding Command="{StaticResource DeleteOrderCommand}" Executed="DeleteOrderCommandHandler"/>
       <CommandBinding Command="{StaticResource UpdateCommand}" Executed="UpdateCommandHandler"/>
       <CommandBinding Command="{StaticResource AddCommand}" Executed="AddCommandHandler"/>
       <CommandBinding Command="{StaticResource CancelCommand}" Executed="CancelCommandHandler"/>
   </Window.CommandBindings>
   ```

3. 이제 `StackPanel` 탐색, 추가, 삭제 및 업데이트 단추를 사용 하 여를 추가 합니다. 먼저이 스타일을에 추가 합니다 `Windows.Resources` .

   ```
   <Style x:Key="NavButton" TargetType="{x:Type Button}" BasedOn="{x:Null}">
       <Setter Property="FontSize" Value="24"/>
       <Setter Property="FontFamily" Value="Segoe UI Symbol"/>
       <Setter Property="Margin" Value="2,2,2,0"/>
       <Setter Property="Width" Value="40"/>
       <Setter Property="Height" Value="auto"/>
   </Style>
   ```

   그런 다음이 코드를 `RowDefinitions` 외부 요소에 대 한 바로 뒤에 `Grid` XAML 페이지의 맨 위에 붙여넣습니

다.

```
<StackPanel Orientation="Horizontal" Margin="2,2,2,0" Height="36" VerticalAlignment="Top"
Background="Gainsboro" DataContext="{StaticResource customerViewSource}" d:LayoutOverrides="LeftMargin,
RightMargin, TopMargin, BottomMargin">
    <Button Name="btnFirst" Content="|◄" Command="{StaticResource FirstCommand}" Style="{StaticResource
NavButton}"/>
    <Button Name="btnPrev" Content="◄" Command="{StaticResource PreviousCommand}" Style="{StaticResource
NavButton}"/>
    <Button Name="btnNext" Content="►" Command="{StaticResource NextCommand}" Style="{StaticResource
NavButton}"/>
    <Button Name="btnLast" Content="►|" Command="{StaticResource LastCommand}" Style="{StaticResource
NavButton}"/>
    <Button Name="btnDelete" Content="Delete Customer" Command="{StaticResource DeleteCustomerCommand}"
FontSize="11" Width="120" Style="{StaticResource NavButton}"/>
    <Button Name="btnAdd" Content="New Customer" Command="{StaticResource AddCommand}" FontSize="11"
Width="80" Style="{StaticResource NavButton}"/>
    <Button Content="New Order" Name="btnNewOrder" FontSize="11" Width="80" Style="{StaticResource
NavButton}" Click="NewOrder_click"/>
    <Button Name="btnUpdate" Content="Commit" Command="{StaticResource UpdateCommand}" FontSize="11"
Width="80" Style="{StaticResource NavButton}"/>
    <Button Content="Cancel" Name="btnCancel" Command="{StaticResource CancelCommand}" FontSize="11"
Width="80" Style="{StaticResource NavButton}"/>
</StackPanel>
```

Mainwindow.xaml 클래스에 명령 처리기 추가

코드 숨김이 추가 및 삭제 메서드를 제외 하 고는 최소화 됩니다. 탐색은 CollectionViewSource의 View 속성에서 메서드를 호출 하 여 수행 됩니다. 는 `DeleteOrderCommandHandler` 순서에 따라 하위 삭제를 수행 하는 방법을 보여 줍니다. 연결 된 Order_Details를 먼저 삭제 해야 합니다. 는 `UpdateCommandHandler` 컬렉션에 새 고객 또는 주문을 추가 하거나 사용자가 텍스트 상자에서 변경한 내용으로 기존 고객 또는 주문을 업데이트 합니다.

*MainWindow.xaml.cs*의 mainwindow.xaml 클래스에 이러한 처리기 메서드를 추가 합니다. Customers 테이블의 CollectionViewSource에 다른 이름이 있는 경우 다음 각 방법에서 이름을 조정 해야 합니다.

```
private void LastCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    custViewSource.View.MoveCurrentToLast();
}

private void PreviousCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    custViewSource.View.MoveCurrentToPrevious();
}

private void NextCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    custViewSource.View.MoveCurrentToNext();
}

private void FirstCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    custViewSource.View.MoveCurrentToFirst();
}

private void DeleteCustomerCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    // If existing window is visible, delete the customer and all their orders.
    // In a real application, you should add warnings and allow the user to cancel the operation.
    var cur = custViewSource.View.CurrentItem as Customer;

    var cust = (from c in context.Customers
                where c.CustomerID == cur.CustomerID
                select c).FirstOrDefault();
```

```
    if (cust != null)
    {
        foreach (var ord in cust.Orders.ToList())
        {
            Delete_Order(ord);
        }
        context.Customers.Remove(cust);
    }
    context.SaveChanges();
    custViewSource.View.Refresh();
}

// Commit changes from the new customer form, the new order form,
// or edits made to the existing customer form.
private void UpdateCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    if (newCustomerGrid.IsVisible)
    {
        // Create a new object because the old one
        // is being tracked by EF now.
        Customer newCustomer = new Customer
        {
            Address = add_addressTextBox.Text,
            City = add_cityTextBox.Text,
            CompanyName = add_companyNameTextBox.Text,
            ContactName = add_contactNameTextBox.Text,
            ContactTitle = add_contactTitleTextBox.Text,
            Country = add_countryTextBox.Text,
            CustomerID = add_customerIDTextBox.Text,
            Fax = add_faxTextBox.Text,
            Phone = add_phoneTextBox.Text,
            PostalCode = add_postalCodeTextBox.Text,
            Region = add_regionTextBox.Text
        };

        // Perform very basic validation
        if (newCustomer.CustomerID.Length == 5)
        {
            // Insert the new customer at correct position:
            int len = context.Customers.Local.Count();
            int pos = len;
            for (int i = 0; i < len; ++i)
            {
                if (String.CompareOrdinal(newCustomer.CustomerID, context.Customers.Local[i].CustomerID) < 0)
                {
                    pos = i;
                    break;
                }
            }
            context.Customers.Local.Insert(pos, newCustomer);
            custViewSource.View.Refresh();
            custViewSource.View.MoveCurrentTo(newCustomer);
        }
        else
        {
            MessageBox.Show("CustomerID must have 5 characters.");
        }

        newCustomerGrid.Visibility = Visibility.Collapsed;
        existingCustomerGrid.Visibility = Visibility.Visible;
    }
    else if (newOrderGrid.IsVisible)
    {
        // Order ID is auto-generated so we don't set it here.
        // For CustomerID, address, etc we use the values from current customer.
        // User can modify these in the datagrid after the order is entered.

        Customer currentCustomer = (Customer)custViewSource.View.CurrentItem;
```

```
        Order newOrder = new Order()
        {
            OrderDate = add_orderDatePicker.SelectedDate,
            RequiredDate = add_requiredDatePicker.SelectedDate,
            ShippedDate = add_shippedDatePicker.SelectedDate,
            CustomerID = currentCustomer.CustomerID,
            ShipAddress = currentCustomer.Address,
            ShipCity = currentCustomer.City,
            ShipCountry = currentCustomer.Country,
            ShipName = currentCustomer.CompanyName,
            ShipPostalCode = currentCustomer.PostalCode,
            ShipRegion = currentCustomer.Region
        };

        try
        {
            newOrder.EmployeeID = Int32.Parse(add_employeeIDTextBox.Text);
        }
        catch
        {
            MessageBox.Show("EmployeeID must be a valid integer value.");
            return;
        }

        try
        {
            // Exercise for the reader if you are using Northwind:
            // Add the Northwind Shippers table to the model.

            // Acceptable ShipperID values are 1, 2, or 3.
            if (add_ShipViaTextBox.Text == "1" || add_ShipViaTextBox.Text == "2"
                || add_ShipViaTextBox.Text == "3")
            {
                newOrder.ShipVia = Convert.ToInt32(add_ShipViaTextBox.Text);
            }
            else
            {
                MessageBox.Show("Shipper ID must be 1, 2, or 3 in Northwind.");
                return;
            }
        }
        catch
        {
            MessageBox.Show("Ship Via must be convertible to int");
            return;
        }

        try
        {
            newOrder.Freight = Convert.ToDecimal(add_freightTextBox.Text);
        }
        catch
        {
            MessageBox.Show("Freight must be convertible to decimal.");
            return;
        }

        // Add the order into the EF model
        context.Orders.Add(newOrder);
        ordViewSource.View.Refresh();
    }

    // Save the changes, either for a new customer, a new order
    // or an edit to an existing customer or order.
    context.SaveChanges();
}

// Sets up the form so that user can enter data. Data is later
// saved when user clicks Commit.
```

```
// Saved when user clicks commit.
private void AddCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    existingCustomerGrid.Visibility = Visibility.Collapsed;
    newOrderGrid.Visibility = Visibility.Collapsed;
    newCustomerGrid.Visibility = Visibility.Visible;

    // Clear all the text boxes before adding a new customer.
    foreach (var child in newCustomerGrid.Children)
    {
        var tb = child as TextBox;
        if (tb != null)
        {
            tb.Text = "";
        }
    }
}

private void NewOrder_click(object sender, RoutedEventArgs e)
{
    var cust = custViewSource.View.CurrentItem as Customer;
    if (cust == null)
    {
        MessageBox.Show("No customer selected.");
        return;
    }

    existingCustomerGrid.Visibility = Visibility.Collapsed;
    newCustomerGrid.Visibility = Visibility.Collapsed;
    newOrderGrid.UpdateLayout();
    newOrderGrid.Visibility = Visibility.Visible;
}

// Cancels any input into the new customer form
private void CancelCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    add_addressTextBox.Text = "";
    add_cityTextBox.Text = "";
    add_companyNameTextBox.Text = "";
    add_contactNameTextBox.Text = "";
    add_contactTitleTextBox.Text = "";
    add_countryTextBox.Text = "";
    add_customerIDTextBox.Text = "";
    add_faxTextBox.Text = "";
    add_phoneTextBox.Text = "";
    add_postalCodeTextBox.Text = "";
    add_regionTextBox.Text = "";

    existingCustomerGrid.Visibility = Visibility.Visible;
    newCustomerGrid.Visibility = Visibility.Collapsed;
    newOrderGrid.Visibility = Visibility.Collapsed;
}

private void Delete_Order(Order order)
{
    // Find the order in the EF model.
    var ord = (from o in context.Orders.Local
               where o.OrderID == order.OrderID
               select o).FirstOrDefault();

    // Delete all the order_details that have
    // this Order as a foreign key
    foreach (var detail in ord.Order_Details.ToList())
    {
        context.Order_Details.Remove(detail);
    }

    // Now it's safe to delete the order.
    context.Orders.Remove(ord);
    context.SaveChanges();
```

```
    context.SaveChanges();

    // Update the data grid.
    ordViewSource.View.Refresh();
}

private void DeleteOrderCommandHandler(object sender, ExecutedRoutedEventArgs e)
{
    // Get the Order in the row in which the Delete button was clicked.
    Order obj = e.Parameter as Order;
    Delete_Order(obj);
}
```

애플리케이션 실행

디버깅을 시작하려면 **F5** 키를 누릅니다. 표에서 customer 및 order 데이터를 채우고 탐색 단추가 예상 대로 작동해야 합니다. 데이터를 입력 한 후 새 고객 또는 주문을 모델에 추가 하려면 **커밋** 을 클릭 합니다. 데이터를 저장 하지 않고 새 고객 또는 새 주문 양식으로 돌아가려면 **취소** 를 클릭 합니다. 텍스트 상자에서 기존 고객과 주문을 직접 편집할 수 있으며 이러한 변경 내용은 자동으로 모델에 기록 됩니다.

참고 항목

- .NET용 Visual Studio 데이터 도구
- Entity Framework 설명서

C++용 Visual Studio 데이터 도구

2020-01-06 • 9 minutes to read • Edit Online

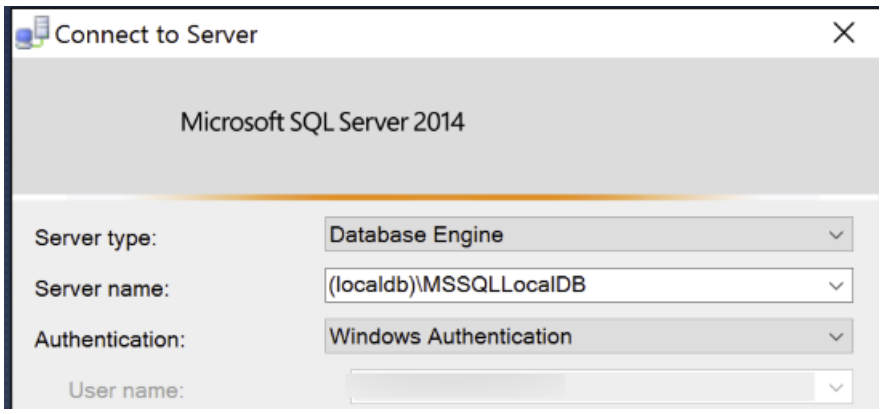
Native C++ 는 데이터 원본에 액세스할 때 가장 빠른 성능을 제공 하는 경우가 많습니다. 그러나 Visual Studio의 응용 C++ 프로그램에 대 한 데이터 도구는 .net 응용 프로그램에 비해 다양 한 기능이 아닙니다. 예를 들어 데이터 **소스 창을 사용** 하 여 데이터 원본을 C++ 디자인 화면으로 끌어서 놓을 수는 없습니다. 개체 관계형 계층이 필요 한 경우에는 직접 작성 하거나 타사 제품을 사용 해야 합니다. Microsoft Foundation Class 라이브러리를 사용 하는 응용 프로그램은 문서 및 뷰와 함께 일부 데이터베이스 클래스를 사용 하 여 메모리에 데이터를 저장 하 고 사용자에 게 표시할 수 있지만, 데이터 바인딩 기능 에서도 마찬가지입니다. 자세한 내용은 시각적 개체 C++의 데이터 액세스 를 참조 하세요.

SQL 데이터베이스에 연결 하기 위해 네이티브 C++ 응용 프로그램은 Windows에 포함 된 ODBC 및 OLE DB 드라이버와 ADO 공급자를 사용할 수 있습니다. 이러한 인터페이스를 지 원하는 모든 데이터베이스에 연결할 수 있습니다. ODBC 드라이버는 표준입니다. OLE DB은 이전 버전과의 호환성을 위해 제공 됩니다. 이러한 데이터 기술에 대 한 자세한 내용은 Windows Data Access 구성 요소를 참조 하세요.

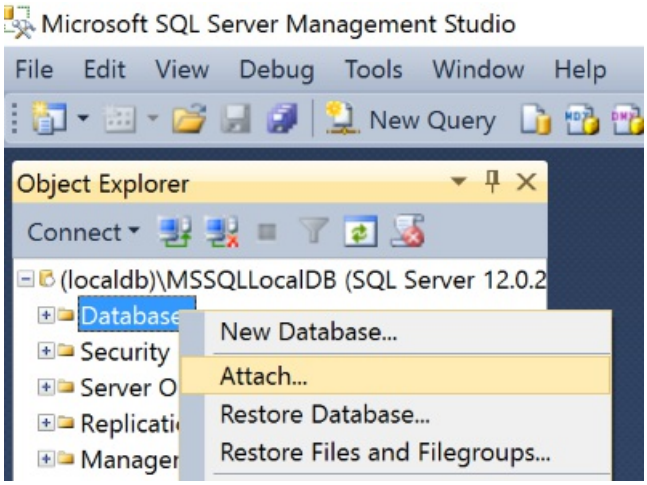
SQL Server 2005 이상에서 사용자 지정 기능을 활용 하려면 SQL Server native client를 사용 합니다. Native client 에는 하나의 네이티브 DLL (동적 연결 라이브러리)에 SQL Server ODBC 드라이버와 SQL Server OLE DB 공급자도 포함 되어 있습니다. 이러한 코드는 네이티브 코드 Api (ODBC, OLE DB 및 ADO)를 사용 하 여 Microsoft SQL Server 하는 응용 프로그램을 지원 합니다. SQL Server Native Client SQL Server Data Tools와 함께 설치 됩니다. 프로그래밍 가이드는 SQL Server native client 프로그래밍을 참조 하세요.

ODBC를 통해 localDB에 연결 하 고 C++ 응용 프로그램에서 SQL Native Client 하려면

1. SQL Server 데이터 도구를 설치합니다.
2. 에 연결할 샘플 SQL 데이터베이스가 필요한 경우 Northwind 데이터베이스를 다운로드 하 고 새 위치에 압축을 풉니다.
3. SQL Server Management Studio를 사용 하 여 압축을 푼 *Northwind.mdf* 파일을 localDB에 연결 합니다. SQL Server Management Studio 시작 되 면 (localdb) \MSSQLLocalDB.에 연결 합니다.



그런 다음 왼쪽 창에서 localdb 노드를 마우스 오른쪽 단추로 클릭 하 고 **연결**을 선택 합니다.



4. ODBC Windows SDK 샘플을 다운로드 하 고 새 위치에 압축을 풉니다. 이 예제에서는 데이터베이스에 연결 하 고 쿼리 및 명령을 실행 하는 데 사용 되는 기본 ODBC 명령을 보여 줍니다. MICROSOFT ODBC (Open Database Connectivity)에서 이러한 함수에 대해 자세히 알아볼 수 있습니다. 솔루션을 처음 로드할 때 (C++ 폴더에 있는 경우) visual studio에서 솔루션을 현재 버전의 visual studio로 업그레이드 하도록 제공 합니다. **예**를 클릭합니다.

5. Native client를 사용 하려면 *헤더* 파일 및 *lib* 파일이 필요 합니다. 이러한 파일에는 SQL .h에 정의 된 ODBC 함수 외에도 SQL Server 관련 된 함수와 정의가 포함 되어 있습니다. **Project** > **속성** > **VC + + 디렉터리**에 다음 포함 디렉터리를 추가 합니다.

   %ProgramFiles%\Microsoft SQL Server\110\SDK\Include

   그리고이 라이브러리 디렉터리는 다음과 같습니다.

   %ProgramFiles%\Microsoft SQL Server\110\SDK\Lib

6. *Odbcsql*에 다음 줄을 추가 합니다. #Define는 관련이 없는 OLE DB 정의가 컴파일될 수 없도록 합니다.

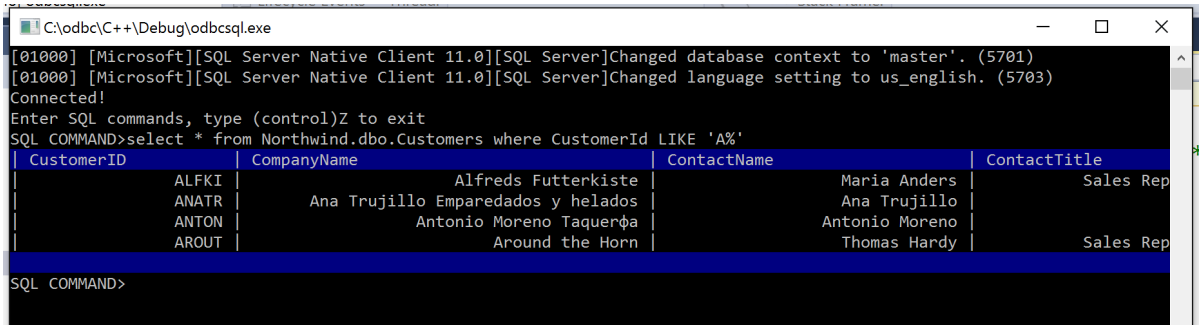
   ```
   #define _SQLNCLI_ODBC_
   #include <sqlncli.h>
   ```

   샘플은 실제로 native client 기능을 사용 하지 않으므로 컴파일 및 실행을 위해 앞의 단계를 수행 하지 않아도 됩니다. 그러나 이제이 기능을 사용할 수 있도록 프로젝트가 구성 되어 있습니다. 자세한 내용은 SQL Server Native Client 프로그래밍을 참조하세요.

7. ODBC 하위 시스템에서 사용할 드라이버를 지정 합니다. 이 샘플은의 드라이버 연결 문자열 특성을 명령줄 인수로 전달 합니다. **디버깅** > **프로젝트** > **속성** 에서 다음 명령 인수를 추가 합니다.

   ```
   DRIVER="SQL Server Native Client 11.0"
   ```

8. **F5** 키를 눌러 애플리케이션을 빌드하고 실행합니다. 데이터베이스를 입력 하 라는 메시지가 표시 되는 드라이버의 대화 상자가 표시 됩니다. `(localdb)\MSSQLLocalDB`를 입력 하 고 **트러스트 된 연결 사용**을 선택 합니다. **확인**을 누릅니다. 연결에 성공 했음을 나타내는 메시지가 포함 된 콘솔이 표시 되어야 합니다. 또한 SQL 문을 입력할 수 있는 명령 프롬프트가 표시 되어야 합니다. 다음 화면에는 예제 쿼리와 결과가 나와 있습니다.



참조

- Visual Studio에서 데이터 액세스

연습: 단일 테이블 상속을 사용 하 여 LINQ to SQL 클래스 만들기 (O/R 디자이너)

2020-01-06 • 13 minutes to read • Edit Online

Visual Studio의 LINQ to SQL 도구 는 일반적으로 관계형 시스템에서 구현 되는 단일 테이블 상속을 지원 합니다. 이 연습은 방법: O/R 디자이너를 사용 하 여 상속 구성 항목에서 제공 하는 일반 단계를 확장 하 고 O/R 디자이너의 상속 사용을 보여 주는 몇 가지 실제 데이터를 제공 합니다.

이 연습에서는 다음 작업을 수행 합니다.

- 데이터베이스 테이블을 만들고 그 안에 데이터를 추가합니다.
- Windows Forms 애플리케이션을 만듭니다.
- LINQ to SQL 파일을 프로젝트에 추가합니다.
- 새 엔터티 클래스를 만듭니다.
- 상속을 사용하도록 엔터티 클래스를 구성합니다.
- 상속된 클래스를 쿼리합니다.
- Windows Form에 데이터를 표시합니다.

상속에 사용될 테이블 만들기

상속이 어떻게 작동 하는지 확인 하려면 작은 `Person` 테이블을 만들고이를 기본 클래스로 사용한 다음이를 상속 하는 `Employee` 개체를 만듭니다.

상속을 보여 주기 위한 기본 테이블을 만들려면

1. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 **테이블** 노드를 마우스 오른쪽 단추로 클릭 하 고 **새 테이블 추가**를 클릭 합니다.

   > **NOTE**
   > Northwind 데이터베이스 또는 테이블을 추가할 수 있는 기타 데이터베이스를 사용할 수 있습니다.

2. **테이블 디자이너**에서 다음 열을 테이블에 추가합니다.

열 이름	데이터 형식	NULL 허용
ID	int	False
Type	int	True
FirstName	nvarchar(200)	False
LastName	nvarchar(200)	False
관리자	int	True

3. ID 열을 기본 키로 설정합니다.

4. 테이블을 저장하고 이름을 **Person**으로 지정합니다.

테이블에 데이터 추가

속성이 올바르게 구성되었는지 확인하기 위해서는 단일 테이블 상속에서 각 클래스에 대한 몇 가지 데이터가 테이블에 필요합니다.

테이블에 데이터를 추가하려면

1. 데이터 뷰에서 테이블을 엽니다. **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 **Person** 테이블을 마우스 오른쪽 단추로 클릭 하 고 **테이블 데이터 표시**를 클릭 합니다.
2. 다음 데이터를 테이블로 복사합니다. (**결과** 창에서 전체 행을 선택 하 여 복사 하 고 테이블에 붙여넣을 수 있습니다.)

ID	**Type**	**FirstName**	**LastName**	**관리자**
1	**1**	**Anne**	**Wallace**	**NULL**
2	**1**	**Carlos**	**Grilo**	**NULL**
3	**1**	**Yael**	**Peled**	**NULL**
4	**2**	**Gatis**	**Ozolins**	**1**
5	**2**	**Andreas**	**Hauser**	**1**
6	**2**	**Tiffany**	**Phuvasate**	**1**
7	**2**	**Alexey**	**Orekhov**	**2**
8	**2**	**Michał**	**Poliszkiewicz**	**2**
9	**2**	**Tai**	**Yee**	**2**
10	**2**	**Fabricio**	**Noriega**	**3**
11	**2**	**Mindy**	**Martin**	**3**
12	**2**	**Ken**	**Kwok**	**3**

새 프로젝트 만들기

테이블을 만든 후에는 상속 구성을 보여 주기 위한 새 프로젝트를 만듭니다.

새 **Windows Forms** 응용 프로그램을 만들려면

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **InheritanceWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   InheritanceWalkthrough 프로젝트가 만들어지고 **솔루션 탐색기**에 추가됩니다.

프로젝트에 LINQ to SQL 클래스 파일 추가

프로젝트에 **LINQ to SQL** 파일을 추가하려면

1. **프로젝트** 메뉴에서 **새 항목 추가**를 클릭합니다.
2. **LINQ to SQL 클래스** 템플릿을 클릭한 다음, **추가**를 클릭합니다.

   .Dbml 파일이 프로젝트에 추가 되 고 **O/R 디자이너가** 열립니다.

O/R 디자이너를 사용하여 상속 만들기

상속 개체를 **도구 상자**에서 디자인 화면으로 끌어 와서 상속을 구성합니다.

상속을 만들려면

1. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 이전에 만든 **Person** 테이블로 이동 합니다.
2. **Person** 테이블을 **O/R 디자이너** 디자인 화면으로 끌어 놓습니다.
3. 두 번째 **Person** 테이블을 **O/R 디자이너로** 끌어 오고 이름을 **Employee**로 변경 합니다.
4. **Person** 개체에서 **Manager** 속성을 삭제합니다.
5. **Employee** 개체에서 **Type**, **ID**, **FirstName** 및 **LastName** 속성을 삭제합니다. 즉, **Manager**를 제외한 모든 속성을 삭제합니다.
6. **도구 상자**의 **개체 관계형 디자이너** 탭에서 **Person** 및 **Employee** 개체 사이에 **상속**을 만듭니다. 이렇게 하려면 **도구 상자**에서 **상속** 항목을 클릭하고 마우스 단추를 놓으세요. 그런 다음, **Employee** 개체를 클릭하 고 **O/R 디자이너**에서 **Person** 개체를 클릭 합니다. 그러면 상속 선의 화살표가 **Person** 개체를 가리킵니다.
7. 디자인 화면에서 **상속** 선을 클릭합니다.
8. **Discriminator Property** 속성을 **Type**으로 설정합니다.
9. **Derived Class Discriminator Value** 속성을 **2**로 설정합니다.
10. **Base Class Discriminator Value** 속성을 **1**로 설정합니다.
11. **Inheritance Default** 속성을 **Person**으로 설정합니다.
12. 프로젝트를 빌드합니다.

상속된 클래스 쿼리 및 폼에 데이터 표시

이제 개체 모델에서 특정 클래스에 대해 쿼리 하는 일부 코드를 폼에 추가 합니다.

LINQ 쿼리를 만들고 폼에 결과를 표시하려면

1. **ListBox**를 **Form1**으로 끌어서 놓습니다.
2. 폼을 두 번 클릭하여 `Form1_Load` 이벤트 처리기를 만듭니다.
3. 다음 코드를 `Form1_Load` 이벤트 처리기에 추가합니다.

```
Dim dc As New DataClasses1DataContext
Dim results = From emp In dc.Persons _
    Where TypeOf emp Is Employee _
    Select emp

For Each Emp As Employee In results
    ListBox1.Items.Add(Emp.LastName)
Next
```

```
NorthwindDataContext dc = new DataClasses1DataContext();
var results = from emp in dc.Persons
              where emp is Employee
              select emp;

foreach(Employee Emp in results)
{
    listBox1.Items.Add(Emp.LastName)
}
```

응용 프로그램 테스트

애플리케이션을 실행하고 목록 상자에 표시된 레코드가 모두 직원(**Type** 열의 값이 2인 레코드)인지 확인합니다.

애플리케이션을 테스트하려면

1. **F5**키를 누릅니다.
2. **Type** 열의 값이 2인 레코드만 표시되는지 확인합니다.
3. 폼을 닫아 디버깅을 (**디버그** 메뉴에서 **디버깅 중지**를 클릭합니다.)

참조

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)
- 방법: 저장 프로시저를 할당하여 업데이트, 삽입 및 삭제 수행(O/R 디자이너)
- LINQ to SQL
- 방법: Visual Basic 또는 C#에서 개체 모델 생성

연습: n 계층 데이터 응용 프로그램 만들기

2020-01-14 • 41 minutes to read • Edit Online

N 계층 데이터 애플리케이션은 데이터에 액세스하며 여러 논리 *계층*으로 구분되는 애플리케이션입니다. 애플리케이션 구성 요소를 개별 계층으로 분리하면 애플리케이션의 확장성과 유지 관리 가능성이 높아집니다. 이는 전체 솔루션을 다시 설계하지 않고도 단일 계층에 적용할 수 있는 새로운 기술을 보다 쉽게 도입할 수 있기 때문입니다. N 계층 아키텍처에는 표시 계층, 중간 계층 및 데이터 계층이 포함됩니다. 중간 계층에는 대개 데이터 액세스 계층, 비즈니스 논리 계층 및 인증, 유효성 검사 등의 공유 구성 요소가 포함됩니다. 데이터 계층에는 관계형 데이터베이스가 포함됩니다. 표시 계층에 액세스하는 최종 사용자로부터 격리된 상태를 유지하기 위해 N 계층 애플리케이션에서는 보통 중요한 정보가 중간 계층의 데이터 액세스 계층에 저장됩니다. 자세한 내용은 N 계층 데이터 응용 프로그램 개요를 참조 하세요.

N 계층 애플리케이션의 여러 계층을 분리하는 방법 중 하나는 애플리케이션에 포함할 각 계층에 대해 개별 프로젝트를 만드는 것입니다. 형식화된 데이터 세트에는 생성된 데이터 세트 및 `DataSet Project` 코드를 포함해야 하는 프로젝트를 결정하는 `TableAdapter` 속성이 포함됩니다.

이 연습에서는 **데이터 세트 디자이너**를 사용하여 데이터 세트 및 `TableAdapter` 코드를 개별 클래스 라이브러리 프로젝트로 분리하는 방법을 보여줍니다. 데이터 집합 및 TableAdapter 코드를 분리 한 후에는 Visual Studio 서비스에서 Windows Communication Foundation Services 및 WCF Data Services 를 만들어 데이터 액세스 계층을 호출 합니다. 마지막으로 Windows Forms 응용 프로그램을 프레젠테이션 계층으로 만듭니다. 이 계층은 데이터 서비스에서 데이터에 액세스합니다.

이 연습에서는 다음 단계를 수행 합니다.

- 여러 프로젝트를 포함 하는 n 계층 솔루션을 새로 만듭니다.
- N 계층 솔루션에 클래스 라이브러리 프로젝트 두 개를 추가합니다.
- **데이터 원본 구성 마법사**를 사용하여 형식화된 데이터 세트를 만듭니다.
- 생성 된 tableadapter 및 데이터 집합 코드를 불연속 프로젝트로 분리 합니다.
- 데이터 액세스 계층으로 호출할 WCF(Windows Communication Foundation) 서비스를 만듭니다.
- 데이터 액세스 계층에서 데이터를 검색하기 위한 함수를 서비스에서 만듭니다.
- 표시 계층으로 사용할 Windows Forms 애플리케이션을 만듭니다.
- 데이터 소스에 바인딩되는 Windows Forms 컨트롤을 만듭니다.
- 데이터 테이블을 채우는 코드를 작성합니다.

이 항목의 비디오 버전을 참조 하세요. Video How to: N 계층 데이터 애플리케이션 만들기.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **.NET 데스크톱 개발** 워크로드의 일부로 또는 개별 구성 요소로서 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. **SQL Server 개체 탐색기** 는 **데이터 저**

장소 및 처리 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

쿼리 편집기 창이 열립니다.

b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

N 계층 솔루션 및 클래스 라이브러리를 만들어 데이터 집합을 저장 합니다 (DataEntityTier).

이 연습의 첫 단계에서는 솔루션과 클래스 라이브러리 프로젝트 두 개를 만듭니다. 첫 번째 클래스 라이브러리에는 데이터 집합 (생성 된 형식화 된 `DataSet` 클래스 및 응용 프로그램의 데이터를 저장 하는 Datatable)이 포함 됩니다. 이 프로젝트는 애플리케이션의 데이터 엔터티 계층으로 사용되며 대개 중간 계층에 배치됩니다. 데이터 집합은 초기 데이터 집합을 만들고 코드를 두 개의 클래스 라이브러리로 자동으로 분리 합니다.

> **NOTE**
> **확인**을 클릭하기 전에 프로젝트와 솔루션의 이름을 올바르게 지정해야 합니다. 그러면 이 연습을 보다 쉽게 완료할 수 있습니다.

N 계층 솔루션 및 **DataEntityTier** 클래스 라이브러리를 만들려면

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **클래스 라이브러리** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **DataEntityTier**로 지정합니다.
5. 솔루션 이름을 **만들어져 ntierwalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   DataEntityTier 프로젝트가 포함된 NTierWalkthrough 솔루션이 만들어져 **솔루션 탐색기**에 추가됩니다.

Tableadapter를 보유할 클래스 라이브러리 만들기 (DataAccessTier)

DataEntityTier 프로젝트를 만든 후 다음 단계에서는 다른 클래스 라이브러리 프로젝트를 만듭니다. 이 프로젝트는 생성 된 Tableadapter를 보유 하며 응용 프로그램의 *데이터 액세스 계층*이라고 합니다. 데이터 액세스 계층은 데이터베이스에 연결하는 데 필요한 정보를 포함하며 대개 중간 계층에 배치됩니다.

Tableadapter에 대 한 별도의 클래스 라이브러리를 만들려면

1. **솔루션 탐색기**에서 솔루션을 마우스 오른쪽 단추로 클릭하고 **추가** > **새 프로젝트**를 선택합니다.
2. **새 프로젝트** 대화 상자의 가운데 창에서 **클래스 라이브러리**를 선택 합니다.
3. 프로젝트 이름을 **DataAccessTier** 로 선택 하 고 **확인**을선택 합니다.

   DataAccessTier 프로젝트가 만들어져 NTierWalkthrough 솔루션에 추가됩니다.

데이터 집합 만들기

다음 단계에서는 형식화된 데이터 세트을 만듭니다. 형식화 된 데이터 집합은 단일 프로젝트에서 dataset 클래스 (`DataTables` 클래스 포함)와 `TableAdapter` 클래스를 사용 하 여 만들어집니다. 모든 클래스는 단일 파일로 생성 됩니다. 데이터 집합 및 Tableadapter를 다른 프로젝트로 분리 하는 경우 다른 프로젝트로 이동 하는 데이터 집합 클

래스로, `TableAdapter` 클래스를 원래 프로젝트에 그대로 둡니다. 따라서 궁극적으로 Tableadapter (DataAccessTier 프로젝트)를 포함 하는 프로젝트에 데이터 집합을 만듭니다. 데이터 집합은 **데이터 소스 구성 마법사**를 사용 하여 만듭니다.

> **NOTE**
> 연결을 만들려면 Northwind 샘플 데이터베이스에 액세스해야 합니다. Northwind 샘플 데이터베이스를 설정 하는 방법에 대 한 자세한 내용은 방법: 예제 데이터베이스 설치를 참조 하세요.

데이터 세트을 만들려면

1. **솔루션 탐색기**에서 **DataAccessTier** 을 선택 합니다.
2. **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다.

   데이터 원본 창이 열립니다.
3. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성 마법사**를 시작합니다.
4. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택 하 고 **다음**을 선택 합니다.
5. **데이터 연결 선택** 페이지에서 다음 작업 중 하나를 수행합니다.

   Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.

   -또는-

   새 연결 을 선택 하 여 **연결 추가** 대화 상자를 엽니다.
6. 데이터베이스에 암호가 필요 하면 중요 한 데이터를 포함 하는 옵션을 선택 하 고 **다음**을 선택 합니다.

   > **NOTE**
   > SQL Server에 연결하는 대신 로컬 데이터베이스 파일을 선택한 경우 프로젝트에 파일을 추가할지를 묻는 메시지가 표시될 수 있습니다. **예** 를 선택 하 여 데이터베이스 파일을 프로젝트에 추가 합니다.
7. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음** 을 선택 합니다.
8. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
9. **Customers** 및 **Orders** 테이블의 확인란을 선택 하 고 **마침**을 선택 합니다.

   NorthwindDataSet이 DataAccessTier 프로젝트에 추가되고 **데이터 원본** 창에 표시됩니다.

데이터 집합에서 Tableadapter를 분리 합니다.

데이터 세트을 만든 후에는 생성된 데이터 세트 클래스를 TableAdapters에서 분리합니다. 이렇게 하려면 **데이터 세트 프로젝트** 속성을 분리된 데이터 세트 클래스를 저장할 프로젝트의 이름으로 설정합니다.

데이터 세트에서 **TableAdapters**를 분리하려면

1. **솔루션 탐색기**에서 **NorthwindDataSet.xsd**를 두 번 클릭하여 **데이터 세트 디자이너**에서 데이터 세트를 엽니다.
2. 디자이너에서 빈 영역을 선택 합니다.
3. **속성** 창에서 **데이터 세트 프로젝트** 노드를 찾습니다.
4. **데이터 집합 프로젝트** 목록에서 **DataEntityTier**를 선택 합니다.
5. **빌드** 메뉴에서 **솔루션 빌드**를 선택합니다.

데이터 세트 및 TableAdapters가 두 클래스 라이브러리 프로젝트로 분리됩니다. 원래 전체 데이터 집합 (`DataAccessTier`)이 포함 된 프로젝트에는 이제 Tableadapter만 포함 됩니다. **데이터 집합 프로젝트** 속성 (`DataEntityTier`)에 지정 된 프로젝트에는 형식화 된 데이터 집합: *NorthwindDataSet* (또는 *NorthwindDataSet.Dataset.Designer.cs*)이 포함 되어 있습니다.

> **NOTE**
> **데이터 세트 프로젝트** 속성을 설정하여 데이터 세트와 TableAdapters를 분리할 때는 프로젝트의 기존 부분 데이터 세트 클래스가 자동으로 이동되지 않습니다. 따라서 데이터 세트 프로젝트로 기존 데이터 세트 부분 클래스를 수동으로 이동해야 합니다.

새 서비스 응용 프로그램 만들기

이 연습에서는 WCF 서비스를 사용 하 여 데이터 액세스 계층에 액세스 하는 방법을 보여 줍니다. 그러면 새 WCF 서비스 응용 프로그램을 만들어 보겠습니다.

새 **WCF** 서비스 애플리케이션을 만들려면

1. **솔루션 탐색기**에서 솔루션을 마우스 오른쪽 단추로 클릭하고 **추가** > **새 프로젝트**를 선택합니다.
2. **새 프로젝트** 대화 상자의 왼쪽 창에서 **WCF**를 선택 합니다. 가운데 창에서 **WCF 서비스 라이브러리**를 선택 합니다.
3. 프로젝트 이름을 **DataService** 로 하 고 **확인**을 선택 합니다.

   DataService 프로젝트가 만들어져 NTierWalkthrough 솔루션에 추가됩니다.

데이터 액세스 계층에서 고객 및 주문 데이터를 반환 하는 메서드 만들기

데이터 서비스는 `GetCustomers` 및 `GetOrders` 데이터 액세스 계층에서 두 메서드를 호출 해야 합니다. 이러한 메서드는 Northwind `Customers` 및 `Orders` 테이블을 반환 합니다. `DataAccessTier` 프로젝트에서 `GetCustomers` 및 `GetOrders` 메서드를 만듭니다.

데이터 액세스 계층에서 **Customers** 테이블을 반환하는 메서드를 만들려면

1. **솔루션 탐색기**에서 **NorthwindDataset** 를 두 번 클릭 하 여 데이터 집합을 엽니다.
2. **Customerstableadapter** 를 마우스 오른쪽 단추로 클릭 하 고 **쿼리 추가**를 클릭 합니다.
3. **명령 유형을 선택하세요.** 페이지에서 기본값인 **SQL 문 사용**을 그대로 유지하고 **다음**을 클릭합니다.
4. **쿼리 형식 선택** 페이지에서 기본값인 **행을 반환하는 SELECT**를 그대로 유지하고 **다음**을 클릭합니다.
5. **SQL SELECT 문을 지정하십시오.** 페이지에서 기본 쿼리를 그대로 유지하고 **다음**을 클릭합니다.
6. **생성할 메서드 선택** 페이지에서 **DataTable 반환** 섹션의 **메서드 이름**에 **GetCustomers**를 입력합니다.
7. **마침**을 클릭합니다.

데이터 액세스 계층에서 **Orders** 테이블을 반환하는 메서드를 만들려면

1. **OrdersTableAdapter**를 마우스 오른쪽 단추로 클릭하고 **쿼리 추가**를 클릭합니다.
2. **명령 유형을 선택하세요.** 페이지에서 기본값인 **SQL 문 사용**을 그대로 유지하고 **다음**을 클릭합니다.
3. **쿼리 형식 선택** 페이지에서 기본값인 **행을 반환하는 SELECT**를 그대로 유지하고 **다음**을 클릭합니다.
4. **SQL SELECT 문을 지정하십시오.** 페이지에서 기본 쿼리를 그대로 유지하고 **다음**을 클릭합니다.
5. **생성할 메서드 선택** 페이지에서 **DataTable 반환** 섹션의 **메서드 이름**에 **GetOrders**를 입력합니다.

6. **마침**을 클릭합니다.

7. **빌드** 메뉴에서 **솔루션 빌드**를 클릭합니다.

데이터 서비스에 데이터 엔터티 및 데이터 액세스 계층에 대 한 참조 추가

데이터 서비스에는 데이터 세트 및 TableAdapters의 정보가 필요하므로 **DataEntityTier** 및 **DataAccessTier** 프로젝트에 대한 참조를 추가합니다.

데이터 서비스에 참조를 추가하려면

1. **솔루션 탐색기**에서 **DataService**를 마우스 오른쪽 단추로 클릭하고 **참조 추가**를 클릭합니다.

2. **참조 추가** 대화 상자에서 **프로젝트** 탭을 클릭합니다.

3. **DataAccessTier** 및 **DataEntityTier** 프로젝트를 모두 선택합니다.

4. **확인**을 클릭합니다.

데이터 액세스 계층에서 GetCustomers 및 GetOrders 메서드를 호출 하는 함수를 서비스에 추가 합니다.

이제 데이터 액세스 계층에 데이터를 반환하는 메서드가 포함되었으므로 데이터 서비스에 데이터 액세스 계층의 메서드를 호출하는 메서드를 만듭니다.

> **NOTE**
> C# 프로젝트의 경우 다음 코드가 컴파일하도록 할 `System.Data.DataSetExtensions` 어셈블리에 대한 참조를 추가해야 합니다.

데이터 서비스에서 **GetCustomers** 및 **GetOrders** 함수를 만들려면

1. **DataService** 프로젝트에서 **IService1.vb** 또는 **IService1.cs**를 두 번 클릭합니다.

2. **여기에 서비스 작업을 추가합니다.** 주석 아래에 다음 코드를 추가합니다.

```
<OperationContract()> _
Function GetCustomers() As DataEntityTier.NorthwindDataSet.CustomersDataTable

<OperationContract()> _
Function GetOrders() As DataEntityTier.NorthwindDataSet.OrdersDataTable
```

```
[OperationContract]
DataEntityTier.NorthwindDataSet.CustomersDataTable GetCustomers();

[OperationContract]
DataEntityTier.NorthwindDataSet.OrdersDataTable GetOrders();
```

3. DataService 프로젝트에서 **Service1.vb** 또는 **Service1.cs**를 두 번 클릭합니다.

4. **Service1** 클래스에 다음 코드를 추가합니다.

```
Public Function GetCustomers() As DataEntityTier.NorthwindDataSet.CustomersDataTable Implements
IService1.GetCustomers
    Dim CustomersTableAdapter1 As New
DataAccessTier.NorthwindDataSetTableAdapters.CustomersTableAdapter
    Return CustomersTableAdapter1.GetCustomers()
End Function

Public Function GetOrders() As DataEntityTier.NorthwindDataSet.OrdersDataTable Implements
IService1.GetOrders
    Dim OrdersTableAdapter1 As New DataAccessTier.NorthwindDataSetTableAdapters.OrdersTableAdapter
    Return OrdersTableAdapter1.GetOrders()
End Function
```

```
public DataEntityTier.NorthwindDataSet.CustomersDataTable GetCustomers()
{
    DataAccessTier.NorthwindDataSetTableAdapters.CustomersTableAdapter
         CustomersTableAdapter1
        = new DataAccessTier.NorthwindDataSetTableAdapters.CustomersTableAdapter();
    return CustomersTableAdapter1.GetCustomers();
}
public DataEntityTier.NorthwindDataSet.OrdersDataTable GetOrders()
{
    DataAccessTier.NorthwindDataSetTableAdapters.OrdersTableAdapter
         OrdersTableAdapter1
        = new DataAccessTier.NorthwindDataSetTableAdapters.OrdersTableAdapter();
    return OrdersTableAdapter1.GetOrders();
}
```

5. **빌드** 메뉴에서 **솔루션 빌드**를 클릭합니다.

데이터 서비스의 데이터를 표시 하는 프레젠테이션 계층 만들기

이제 데이터 액세스 계층을 호출 하는 메서드가 포함 된 데이터 서비스를 솔루션에 포함 했으므로 데이터 서비스를 호출 하 고 사용자에 게 데이터를 제공 하는 다른 프로젝트를 만듭니다. 이 연습에서는 N 계층 애플리케이션의 표시 계층인 Windows Forms 애플리케이션을 만듭니다.

표시 계층 프로젝트를 만들려면

1. **솔루션 탐색기**에서 솔루션을 마우스 오른쪽 단추로 클릭하고 **추가** > **새 프로젝트**를 선택합니다.
2. **새 프로젝트** 대화 상자의 왼쪽 창에서 **Windows 바탕 화면**을 선택 합니다. 가운데 창에서 **Windows Forms 앱**을 선택 합니다.
3. 프로젝트 이름을 **PresentationTier**로 지정하고 **확인**을 클릭합니다.

   PresentationTier 프로젝트가 만들어져 NTierWalkthrough 솔루션에 추가됩니다.

PresentationTier 프로젝트를 시작 프로젝트로 설정

데이터를 표시 하 고 상호 작용 하는 실제 클라이언트 응용 프로그램 이기 때문에 **PresentationTier** 프로젝트를 솔루션의 시작 프로젝트로 설정 합니다.

새 표시 계층 프로젝트를 시작 프로젝트로 설정하려면

- **솔루션 탐색기**에서 **PresentationTier**를 마우스 오른쪽 단추로 클릭하고 **시작 프로젝트로 설정**을 클릭합니다.

프레젠테이션 계층에 대 한 참조 추가

클라이언트 애플리케이션 PresentationTier가 서비스의 메서드에 액세스하려면 데이터 서비스에 대한 서비스 참

조가 필요합니다. WCF 서비스를 통한 형식 공유를 사용하도록 설정하려면 데이터 세트에 대한 참조도 필요합니다. 데이터 서비스를 통해 형식 공유를 사용 하도록 설정할 때까지 부분 데이터 집합 클래스에 추가 된 코드는 프레젠테이션 계층에서 사용할 수 없습니다. 일반적으로 데이터 테이블의 행 및 열 변경 이벤트에 유효성 검사 코드와 같은 코드를 추가 하기 때문에 클라이언트에서이 코드에 액세스 하려고 할 가능성이 높습니다.

표시 계층에 대한 참조를 추가하려면

1. **솔루션 탐색기**에서 **PresentationTier** 을 마우스 오른쪽 단추로 클릭 하 고 **참조 추가**를 선택 합니다.
2. **참조 추가** 대화 상자에서 **프로젝트** 탭을 선택 합니다.
3. **DataEntityTier** 를 선택 하 고 **확인**을 선택 합니다.

표시 계층에 대한 서비스 참조를 추가하려면

1. **솔루션 탐색기**에서 **PresentationTier** 을 마우스 오른쪽 단추로 클릭 하 고 **서비스 참조 추가**을 선택 합니다.
2. **서비스 참조 추가** 대화 상자에서 **검색**을 선택 합니다.
3. **Service1** 을 선택 하 고 **확인**을 선택 합니다.

> **NOTE**
> 현재 컴퓨터에 여러 서비스가 있는 경우이 연습에서 이전에 만든 서비스 (`GetCustomers` 및 `GetOrders` 메서드를 포함 하는 서비스)를 선택 합니다.

Datagridview를 폼에 추가 하 여 데이터 서비스에서 반환 하는 데이터 표시

데이터 서비스에 대한 서비스 참조를 추가하고 나면 **데이터 소스** 창에 서비스에 의해 반환되는 데이터가 자동으로 채워집니다.

데이터 바인딩된 **DataGridView** 두 개를 폼에 추가하려면

1. **솔루션 탐색기**에서 **PresentationTier** 프로젝트를 선택합니다.
2. **데이터 원본** 창에서 **NorthwindDataSet**를 확장하고 **Customers** 노드를 찾습니다.
3. **Customers** 노드를 Form1로 끌어 옵니다.
4. **데이터 원본** 창에서 **Customers** 노드를 확장하고 관련 **Orders** 노드(**Customers** 노드에 중첩된 **Orders** 노드)를 찾습니다.
5. 관련 **Orders** 노드를 Form1로 끌어 옵니다.
6. 폼의 빈 영역을 두 번 클릭하여 `Form1_Load` 이벤트 처리기를 만듭니다.
7. 다음 코드를 `Form1_Load` 이벤트 처리기에 추가합니다.

```
Dim DataSvc As New ServiceReference1.Service1Client
NorthwindDataSet.Customers.Merge(DataSvc.GetCustomers)
NorthwindDataSet.Orders.Merge(DataSvc.GetOrders)
```

```
ServiceReference1.Service1Client DataSvc =
    new ServiceReference1.Service1Client();
northwindDataSet.Customers.Merge(DataSvc.GetCustomers());
northwindDataSet.Orders.Merge(DataSvc.GetOrders());
```

서비스에서 허용 하는 최대 메시지 크기를 늘립니다.

`maxReceivedMessageSize`의 기본값은 `Customers` 및 `Orders` 테이블에서 검색 된 데이터를 저장할 수 있을 만큼 크지 않습니다. 다음 단계에서는 값을 6553600로 늘립니다. 클라이언트에서 값을 변경 하 여 서비스 참조를 자동으로 업데이트 합니다.

> **NOTE**
> 기본값이 작은 이유는 DoS(서비스 거부) 공격에 대한 노출을 제한하기 위해서입니다. 자세한 내용은 MaxReceivedMessageSize를 참조하세요.

maxReceivedMessageSize 값을 늘리려면

1. **솔루션 탐색기**에서 **PresentationTier** 프로젝트의 **app.config** 파일을 두 번 클릭합니다.
2. **maxReceivedMessage** 크기 특성을 찾아 값을 `6553600`으로 변경합니다.

응용 프로그램 테스트

F5를 눌러 애플리케이션을 실행합니다. `Customers` 및 `Orders` 테이블의 데이터는 데이터 서비스에서 검색 되어 폼에 표시 됩니다.

다음 단계

애플리케이션 요구 사항에 따라 Windows 기반 애플리케이션에서 관련 데이터를 저장한 후 몇 단계를 더 수행해야 할 수도 있습니다. 예를 들어 이 애플리케이션을 다음과 같이 개선할 수 있습니다.

- 데이터 세트에 유효성 검사 기능을 추가합니다.
- 데이터를 데이터베이스로 다시 업데이트하기 위한 추가 메서드를 서비스에 추가합니다.

참조

- n 계층 애플리케이션에서 데이터 세트 작업
- 계층적 업데이트
- Visual Studio에서 데이터 액세스

Visual Studio용 호환 데이터베이스 시스템

2020-01-06 • 10 minutes to read • Edit Online

Visual Studio에서 데이터 연결 응용 프로그램을 개발 하려면 일반적으로 로컬 개발 컴퓨터에 데이터베이스 시스템을 설치한 다음 준비가 되 면 프로덕션 환경에 응용 프로그램 및 데이터베이스를 배포 합니다. Visual Studio는 **데이터 저장소 및 처리** 워크 로드의 일부로 컴퓨터에 SQL Server Express LocalDB를 설치 합니다. 이 LocalDB 인스턴스는 데이터 연결 응용 프로그램을 빠르고 쉽게 개발 하는 데 유용 합니다.

.NET 응용 프로그램에서 액세스할 수 있고 Visual Studio data tools 창에 표시 되는 데이터베이스 시스템에는 ADO.NET 데이터 공급자가 있어야 합니다. 공급자는 .NET 응용 프로그램에서 엔터티 데이터 모델을 사용 하려는 경우 Entity Framework를 구체적으로 지원 해야 합니다. 대부분의 공급자는 NuGet 패키지 관리자 또는 Visual Studio Marketplace를 통해 제공 됩니다.

Azure storage Api를 사용 하는 경우 프로덕션 환경에 배포할 준비가 될 때까지 요금을 방지 하기 위해 개발 중에 로컬 컴퓨터에 Azure storage 에뮬레이터를 설치 합니다. 자세한 내용은 Azure Storage 에뮬레이터를 사용 하 여 개발 및 테스트를 참조 하세요.

다음 목록에는 Visual Studio 프로젝트에서 사용할 수 있는 인기 있는 데이터베이스 시스템 중 일부가 포함 되어 있습니다. 목록은 완전 하지 않습니다. Visual Studio 도구와 긴밀 하 게 통합할 수 있는 ADO.NET 데이터 공급자를 제공 하는 타사 공급 업체 목록은 ADO.NET Data providers를 참조 하세요.

Microsoft SQL Server

SQL Server는 Microsoft 주력 데이터베이스 제품입니다. SQL Server 2016은 혁신적인 성능, 고급 보안 및 풍부한 통합 보고 및 분석을 제공 합니다. 단일 컴퓨터에서 사용할 수 있는 확장성이 뛰어난 고성능 비즈니스 분석에서 다양 한 용도로 설계 된 다양 한 버전으로 제공 됩니다. SQL Server Express은 재배포 및 포함에 맞게 조정 된 SQL Server의 완전 한 기능을 갖춘 버전입니다. LocalDB는 구성이 필요 없고 응용 프로그램의 프로세스에서 실행 되는 SQL Server Express의 단순화 된 버전입니다. SQL Server Express 다운로드 페이지에서 또는 두 제품 중 하나를 다운로드할 수 있습니다. 이 섹션의 많은 SQL 예제에서는 SQL Server LocalDB를 사용 합니다. SSMS (SQL Server Management Studio)는 Visual Studio SQL Server 개체 탐색기에서 제공 되는 것 보다 많은 기능을 제공 하는 독립 실행형 데이터베이스 관리 응용 프로그램입니다. 이전 링크에서 SSMS를 가져올 수 있습니다.

Oracle

Oracle 기술 네트워크 페이지에서 oracle 데이터베이스의 유료 또는 무료 버전을 다운로드할 수 있습니다. Entity Framework 및 Tableadapter에 대 한 디자인 타임 지원을 위해 Visual Studio 용 Oracle 개발자 도구가 필요 합니다. Oracle 인스턴트 클라이언트를 비롯 한 다른 공식 Oracle 제품은 NuGet 패키지 관리자를 통해 사용할 수 있습니다. Oracle 온라인 설명서의 지침에 따라 oracle 샘플 스키마를 다운로드할 수 있습니다.

MySQL

MySQL은 기업 및 websites에서 널리 사용 되는 인기 있는 오픈 소스 데이터베이스 시스템입니다. Mysql, Visual Studio 용 MySQL 및 관련 제품에 대 한 다운로드는 Windows의 mysql에 있습니다. 타사는 다양 한 Visual Studio 확장 및 MySQL 용 독립 실행형 관리 응용 프로그램을 제공 합니다. Nuget 패키지 관리자 (**도구** > Nuget 패키지 **관리자** > **솔루션에 대 한 nuget 패키지 관리**)에서 제공 하는 기능을 찾아볼 수 있습니다.

PostgreSQL

PostgreSQL는 무료 오픈 소스 개체 관계형 데이터베이스 시스템입니다. Windows에 설치 하려면 PostgreSQL 다운로드 페이지에서 다운로드할 수 있습니다. 소스 코드에서 PostgreSQL를 빌드할 수도 있습니다. PostgreSQL core 시스템은 C 언어 인터페이스를 포함 합니다. 많은 제 3 자가 .NET 응용 프로그램에서 PostgreSQL를 사용 하

기 위한 NuGet 패키지를 제공 합니다. Nuget 패키지 관리자 (**도구** > Nuget 패키지 **관리자** > **솔루션에 대 한 nuget 패키지 관리**)에서 제공 하는 기능을 찾아볼 수 있습니다. 아마도 npgsql.org에서 가장 인기 있는 패키지를 제공 합니다.

SQLite

SQLite는 응용 프로그램 자체 프로세스에서 실행 되는 임베디드 SQL database 엔진입니다. SQLite 다운로드 페이지에서 다운로드할 수 있습니다. SQLite 용 타사 NuGet 패키지도 사용할 수 있습니다. Nuget 패키지 관리자 (**도구** > Nuget 패키지 **관리자** > **솔루션에 대 한 nuget 패키지 관리**)에서 제공 하는 기능을 찾아볼 수 있습니다.

Firebird

Firebird은 오픈 소스 SQL 데이터베이스 시스템입니다. Firebird 다운로드 페이지에서 다운로드할 수 있습니다. ADO.NET 데이터 공급자는 NuGet 패키지 관리자를 통해 사용할 수 있습니다.

참조

- Visual Studio에서 데이터 액세스
- SQL Server 및 관련 구성 요소의 버전을 확인하는 방법

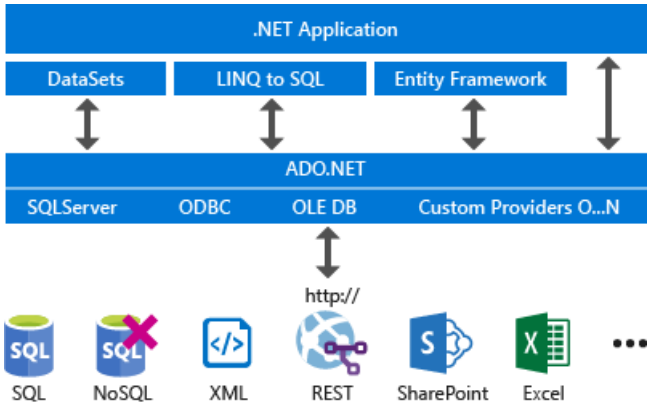
.NET용 Visual Studio 데이터 도구

2020-01-06 • 5 minutes to read • Edit Online

Visual Studio 및 .NET은 데이터베이스에 연결 하 고, 메모리에서 데이터를 모델링 하 고, 사용자 인터페이스에서 데이터를 표시 하기 위한 광범위 한 API 및 도구 지원을 제공 합니다. 데이터 액세스 기능을 제공 하는 .NET 클래스를 ADO.NET라고 합니다. ADO.NET는 Visual Studio의 데이터 도구와 함께 관계형 데이터베이스 및 XML을 지원 하기 위해 주로 설계 되었습니다. 이러한 일, 많은 NoSQL 데이터베이스 공급 업체 또는 제 3 자가 ADO.NET 공급자를 제공 합니다.

.Net Core 는 데이터 집합 및 관련 형식을 제외 하 고 ADO.NET을 지원 합니다. .NET Core를 대상으로 지정 하 고 ORM (개체-관계형 매핑) 계층을 요구 하는 경우 Entity Framework Core를 사용 합니다.

다음 다이어그램에서는 기본 아키텍처의 단순화 된 보기를 보여 줍니다.



일반적인 워크플로

일반적인 워크플로는 다음과 같습니다.

1. 로컬 컴퓨터에 개발 또는 테스트 데이터베이스를 설치 합니다. 데이터베이스 시스템, 도구 및 샘플 설치를 참조 하세요. Azure 데이터 서비스를 사용 하는 경우에는이 단계가 필요 하지 않습니다.

2. Visual Studio에서 데이터베이스 (또는 서비스 또는 로컬 파일)에 대 한 연결을 테스트 합니다. 새 연결 추가를 참조 하세요.

3. 필드 도구를 사용 하 여 새 모델을 생성 하 고 구성 합니다. 새 응용 프로그램에 대 한 기본 권장 사항은 Entity Framework 기반으로 하는 모델입니다. 사용 하는 모델은 응용 프로그램이 상호 작용 하는 데이터 원본입니다. 모델은 데이터베이스 또는 서비스와 응용 프로그램 간에 논리적으로 배치 됩니다. 새 데이터 소스 추가를 참조 하세요.

4. **데이터 소스 창에서** Windows Forms, ASP.NET 또는 Windows Presentation Foundation 디자인 화면으로 데이터 소스를 끌어 사용자가 지정한 방식으로 데이터를 표시 하는 데이터 바인딩 코드를 생성 합니다. Visual Studio에서 데이터에 컨트롤 바인딩을참조 하세요.

5. 비즈니스 규칙, 검색 및 데이터 유효성 검사와 같은 항목에 대 한 사용자 지정 코드를 추가 하거나 기본 데이터베이스에서 노출 하는 사용자 지정 기능을 활용 합니다.

3 단계를 건너뛰고 .NET 응용 프로그램을 프로그래밍 하 여 모델을 사용 하지 않고 데이터베이스에 직접 명령을 실행할 수 있습니다. 이 경우에 관련 설명서를 보면: ADO.NET합니다. **데이터 소스 구성 마법사** 및 디자이너를 사용 하 여 메모리에 사용자 고유의 개체를 채운 다음 해당 개체에 데이터를 바인딩할 때 데이터 바인딩 코드를 생성할 수도 있습니다.

참조

- Visual Studio에서 데이터 액세스

Visual Studio의 Entity Framework Tools

2020-01-14 • 9 minutes to read • Edit Online

Entity Framework는 .NET 개발자가 도메인별 개체를 사용 하 여 관계형 데이터 작업을 수행할 수 있도록 하는 개체-관계형 매핑 기술입니다. 개발자들이 보통 작성해야 하는 데이터 액세스 코드가 대부분 필요하지 않게 됩니다. Entity Framework은 새로운 .NET 응용 프로그램에 권장 되는 ORM (개체 관계형 매핑) 모델링 기술입니다.

Entity Framework Tools는 EF (Entity Framework) 응용 프로그램을 빌드하는 데 도움이 되도록 설계 되었습니다. Entity Framework에 대 한 전체 설명서는 개요-EF 6에서 확인할 수 있습니다.

> **NOTE**
> 이 페이지에 설명 된 Entity Framework Tools는 EF Core에서 지원 되지 않는 .edmx 파일을 생성 하는 데 사용 됩니다 . 기존 데이터베이스에서 EF Core 모델을 생성 하려면 리버스 엔지니어링-EF Core을 참조 하세요. EF 6과 EF Core 간의 차이점에 대 한 자세한 내용은 ef 6 및 EF Core 비교를 참조 하세요.

Entity Framework Tools를 사용 하면 기존 데이터베이스에서 *개념적 모델* 을 만든 다음 개념적 모델을 그래픽으로 시각화 하 고 편집할 수 있습니다. 또는 먼저 개념적 모델을 그래픽으로 만든 후 모델을 지원하는 데이터베이스를 생성할 수 있습니다. 이 두 경우 모두 기본 데이터베이스가 변경될 때 모델을 자동으로 업데이트하고 애플리케이션에 대한 개체 계층 코드를 자동으로 생성할 수 있습니다. 데이터베이스 생성 및 개체 계층 코드 생성 작업은 사용자 지정할 수 있습니다.

Entity Framework 도구는 Visual Studio 설치 관리자에서 **데이터 저장소 및 처리** 워크 로드의 일부로 설치 됩니다. **Sdk, 라이브러리 및 프레임 워크** 범주 아래에 개별 구성 요소로 설치할 수도 있습니다.

다음은 Visual Studio에서 Entity Framework 도구를 구성 하는 특정 도구입니다.

- **Entity Designer**(ADO.NET **엔터티 데이터 모델 디자이너**)를 사용 하 여 엔터티, 연결, 매핑 및 상속 관계를 시각적으로 만들고 수정할 수 있습니다. 또한 **Entity Designer** C# 또는 Visual Basic 개체 계층 코드를 생성 합니다.
- **엔터티 데이터 모델 마법사** 를 사용 하 여 기존 데이터베이스에서 개념적 모델을 생성 하 고 응용 프로그램에 데이터베이스 연결 정보를 추가할 수 있습니다.
- **데이터베이스 만들기 마법사** 를 사용 하 여 먼저 개념적 모델을 만든 다음 모델을 지 원하는 데이터베이스를 만들 수 있습니다.
- 기본 데이터베이스가 변경 된 경우 **모델 업데이트 마법사** 를 사용 하 여 개념적 모델, 저장소 모델 및 매핑을 업데이트할 수 있습니다.

> **NOTE**
> Visual Studio 2010부터 Entity Framework 도구는 SQL Server 2000를 지원 하지 않습니다.

도구는 *.edmx* 파일을 생성 하거나 수정 합니다. 이 *.edmx* 파일에는 개념적 모델, 저장소 모델 및 두 모델 간의 매핑을 설명 하는 정보가 포함 되어 있습니다. 자세한 내용은 EDMX를 참조 하세요.

Entity Framework Power Tools 는 엔터티 데이터 모델를 사용 하는 응용 프로그램을 빌드하는 데 도움이 됩니다. Power tools는 개념적 모델을 생성 하 고, 기존 모델의 유효성을 검사 하 고, 개념적 모델을 기반으로 하는 개체 클래스가 포함 된 소스 코드 파일을 생성 하 고, 모델이 생성 하는 뷰를 포함 하는 소스 코드 파일을 생성할 수 있습니다. 자세한 내용은 미리 생성 된 매핑 뷰를 참조 하세요.

관련 항목

제목	설명
ADO.NET Entity Framework	Entity Framework에서 제공 하는 엔터티 데이터 모델 도구를 사용 하 여 응용 프로그램을 만드는 방법을 설명 합니다.
엔터티 데이터 모델	Entity Framework에서 빌드된 응용 프로그램에서 사용 하는 데이터 작업을 위한 링크와 정보를 제공 합니다.
Entity Framework (EF) 설명서)	Entity Framework를 최대한 활용 하는 데 도움이 되는 비디오, 자습서 및 고급 설명서의 인덱스를 제공 합니다.
ASP.NET 5 응용 프로그램을 새 데이터베이스로	Entity Framework 7을 사용 하 여 새 ASP.NET 5 응용 프로그램을 만드는 방법을 설명 합니다.

참조

- .NET용 Visual Studio 데이터 도구

Visual Studio의 데이터 세트 도구

2020-01-06 • 9 minutes to read • Edit Online

> **NOTE**
> 데이터 집합 및 관련 클래스는 응용 프로그램이 데이터베이스와의 연결을 해제 하는 동안 응용 프로그램에서 메모리의 데이터를 사용할 수 있도록 하는 초기 2000s의 레거시 .NET 기술입니다. 사용자가 데이터를 수정 하 고 변경 내용을 데이터베이스에 다시 저장할 수 있도록 하는 응용 프로그램에 특히 유용 합니다. 데이터 집합은 매우 성공적인 기술로 입증 되었지만 새 .NET 응용 프로그램에서 Entity Framework를 사용 하는 것이 좋습니다. Entity Framework은 테이블 형식 데이터를 개체 모델로 사용 하는 보다 자연 스러운 방법을 제공 하 고 프로그래밍 인터페이스를 단순화 합니다.

`DataSet` 개체는 기본적으로 미니 데이터베이스인 메모리 내 개체입니다. 여기에는 열려 있는 연결을 유지 관리할 필요 없이 하나 이상의 데이터베이스에서 데이터를 저장 하 고 수정할 수 있는 `DataTable`, `DataColumn` 및 `DataRow` 개체가 포함 되어 있습니다. 데이터 집합은 해당 데이터 변경 내용에 대 한 정보를 유지 하므로 응용 프로그램이 다시 연결 되 면 업데이트를 추적 하 고 데이터베이스로 다시 보낼 수 있습니다.

데이터 집합 및 관련 클래스는 .NET API의 System.Data 네임 스페이스에 정의 되어 있습니다. ADO.NET를 사용 하 여 코드에서 동적으로 데이터 집합을 만들고 수정할 수 있습니다. 이 단원의 설명서에서는 Visual Studio 디자이너를 사용 하 여 데이터 집합으로 작업 하는 방법을 보여 줍니다. 디자이너를 통해 생성 된 데이터 집합은 **TableAdapter** 개체를 사용 하 여 데이터베이스와 상호 작용 합니다. 프로그래밍 방식으로 만들어진 데이터 집합은 **DataAdapter** 개체를 사용 합니다. 프로그래밍 방식으로 데이터 집합을 만드는 방법에 대 한 자세한 내용은 dataadapter 및 DataReaders를 참조 하세요.

응용 프로그램에서 데이터베이스의 데이터만 읽고 업데이트, 추가 또는 삭제를 수행 해야 하는 경우 일반적으로 `DataReader` 개체를 사용 하 여 일반 `List` 개체나 다른 컬렉션 개체에 데이터를 검색 하는 것이 더 나은 성능을 얻을 수 있습니다. 데이터를 표시 하는 경우 사용자 인터페이스를 컬렉션에 데이터 바인딩할 수 있습니다.

데이터 집합 워크플로

Visual Studio는 데이터 집합 작업을 간소화 하는 도구를 제공 합니다. 기본 종단 간 워크플로는 다음과 같습니다.

- 데이터 소스 창을 사용 하 여 하나 이상의 데이터 원본에서 새 데이터 집합을 만들 수 있습니다. **데이터 세트 디자이너** 를 사용 하 여 데이터 집합을 구성 하 고 해당 속성을 설정 합니다. 예를 들어 데이터 원본에서 포함할 테이블과 각 테이블의 열을 지정 해야 합니다. 데이터 집합에 필요한 메모리 양을 신중 하 게 선택 합니다. 자세한 내용은 데이터 세트 만들기 및 구성을 참조하세요.
- 외래 키가 올바르게 처리 되도록 테이블 간의 관계를 지정 합니다. 자세한 내용은 tableadapter를 사용 하 여 데이터 집합 채우기를 참조 하세요.
- **TableAdapter 구성 마법사** 를 사용 하 여 데이터 집합을 채우는 쿼리 또는 저장 프로시저와 구현할 데이터베이스 작업 (업데이트, 삭제 등)을 지정할 수 있습니다. 자세한 내용은 다음 항목을 참조하세요.
  - TableAdapter를 사용하여 데이터 세트 채우기
  - 데이터 세트의 데이터 편집
  - 데이터 세트의 데이터 유효성 검사
  - 데이터를 다시 데이터베이스에 저장
- 데이터 집합의 데이터를 쿼리하고 검색 합니다. 자세한 내용은 쿼리 데이터 집합을 참조 하세요. LINQ

to DataSet DataSet 개체의 데이터에 대해 LINQ (통합 언어 쿼리) 를 사용 하도록 설정 합니다. 자세한 내용은 LINQ to DataSet을 참조하세요.

- **데이터 소스** 창을 사용 하 여 사용자 인터페이스 컨트롤을 데이터 집합 또는 개별 열에 바인딩하고 사용자가 편집할 수 있는 열을 지정할 수 있습니다. 자세한 내용은 Visual Studio에서 데이터에 컨트롤 바인딩을 참조 하세요.

데이터 집합 및 N 계층 아키텍처

N 계층 응용 프로그램의 데이터 집합에 대 한 자세한 내용은 n 계층 응용 프로그램에서 데이터 집합 작업을 참조 하세요.

데이터 집합 및 XML

데이터 집합을 XML로 변환 하는 방법에 대 한 자세한 내용은 xml 데이터를 데이터 집합으로 읽어 데이터 집합을 xml로 저장을 참조 하세요.

참조

- .NET용 Visual Studio 데이터 도구

형식화된 데이터 세트 및 형식화되지 않은 데이터 세트

2020-01-06 • 8 minutes to read • Edit Online

형식화 된 데이터 집합은 기본 DataSet 클래스에서 처음 파생 된 데이터 집합이 며 .xsd 파일에 저장 된 **데이터 세트 디자이너**정보를 사용 하 여 강력한 형식의 새 dataset 클래스를 생성 합니다. 스키마의 정보 (테이블, 열 등)가 생성 되 고 새 데이터 집합 클래스에 일련의 첫 번째 클래스 개체 및 속성으로 컴파일됩니다. 형식화 된 데이터 집합은 기본 DataSet 클래스에서 상속 되므로 형식화 된 클래스는 DataSet 클래스의 모든 기능을 사용 하 고 DataSet 클래스의 인스턴스를 매개 변수로 사용 하는 메서드와 함께 사용할 수 있습니다.

이와 달리 형식화 되지 않은 데이터 집합에는 해당 하는 기본 제공 스키마가 없습니다. 형식화 된 데이터 집합에서와 마찬가지로 형식화 되지 않은 데이터 집합은 테이블, 열 등을 포함 하지만 컬렉션 으로만 표시 됩니다. 그러나 형식화 되지 않은 데이터 집합에서 수동으로 테이블 및 기타 데이터 요소를 만든 후에는 데이터 집합의 WriteXmlSchema 메서드를 사용 하 여 데이터 집합의 구조를 스키마로 내보낼 수 있습니다.

형식화 된 데이터 집합 및 형식화 되지 않은 데이터 집합의 데이터 액세스 대비

형식화 된 데이터 집합에 대 한 클래스에는 테이블 및 열의 실제 이름을 사용 하는 개체 모델이 있습니다. 예를 들어 형식화 된 데이터 집합을 사용 하 여 작업 하는 경우 다음과 같은 코드를 사용 하 여 열을 참조할 수 있습니다.

```
// This accesses the CustomerID column in the first row of the Customers table.
string customerIDValue = northwindDataSet.Customers[0].CustomerID;
```

```
' This accesses the CustomerID column in the first row of the Customers table.
Dim customerIDValue As String = NorthwindDataSet.Customers(0).CustomerID
```

이와 대조적으로 형식화 되지 않은 데이터 집합을 사용 하 여 작업 하는 경우 해당 코드는 다음과 같습니다.

```
string customerIDValue = (string)
    dataset1.Tables["Customers"].Rows[0]["CustomerID"];
```

```
Dim customerIDValue As String =
    CType(dataset1.Tables("Customers").Rows(0).Item("CustomerID"), String)
```

형식화 된 액세스는 더 쉽게 읽을 수 있을 뿐만 아니라 Visual Studio **코드 편집기**에서 IntelliSense에 의해 완전히 지원 됩니다. 형식화 된 데이터 집합에 대 한 구문을 사용 하 여 보다 쉽게 작업할 수 있을 뿐만 아니라 컴파일 시간에 형식 검사를 제공 하므로 데이터 집합 멤버에 값을 할당할 때 오류가 발생할 가능성을 크게 줄일 수 있습니다. DataSet 클래스에서 열 이름을 변경한 다음 응용 프로그램을 컴파일하면 빌드 오류가 표시 됩니다. **작업 목록**에서 빌드 오류를 두 번 클릭 하면 이전 열 이름을 참조 하는 코드 줄로 바로 이동할 수 있습니다. 런타임에는 컬렉션이 아니라 컴파일 타임에 액세스를 결정 하기 때문에 형식화 된 데이터 집합의 테이블 및 열에 대 한 액세스도 약간 더 빠릅니다.

형식화 된 데이터 집합에는 많은 이점이 있지만 형식화 되지 않은 데이터 집합은 다양 한 상황에서 유용 합니다. 가장 명백한 시나리오는 데이터 집합에 사용할 수 있는 스키마가 없는 경우입니다. 이는 예를 들어 응용 프로그램이 데이터 집합을 반환 하는 구성 요소와 상호 작용 하지만 해당 구조를 미리 알 수 없는 경우에 발생할 수 있습니다. 마찬가지로 정적이 고 예측 가능한 구조가 없는 데이터로 작업 하는 경우도 있습니다. 이 경우 형식화 된 데이터 집

합을 사용 하는 것은 적합 하지 않습니다. 데이터 구조를 변경 하 여 형식화 된 데이터 집합 클래스를 다시 생성 해야 하기 때문입니다.

더 일반적으로 스키마를 사용할 수 없는 상태에서 데이터 집합을 동적으로 만들 수 있는 경우가 많습니다. 이 경우 데이터를 관계형 방식으로 표현할 수 있는 경우 데이터 집합은 정보를 유지할 수 있는 편리한 구조입니다. 동시에 다른 프로세스에 전달할 정보를 직렬화 하거나 XML 파일을 작성 하는 기능과 같은 데이터 집합의 기능을 활용할 수 있습니다.

참조

- 데이터 세트 도구

TableAdapters를 사용하여 데이터 세트 채우기

2020-03-23 • 27 minutes to read • Edit Online

TableAdapter 구성 요소는 지정한 하나 이상의 쿼리 또는 저장 프로시저에 따라 데이터베이스의 데이터로 데이터 집합을 채웁니다. TableAdapters는 데이터베이스에서 추가, 업데이트 및 삭제를 수행하여 데이터 집합에 대한 변경 내용을 지속할 수도 있습니다. 특정 테이블과 관련이 없는 전역 명령을 발행할 수도 있습니다.

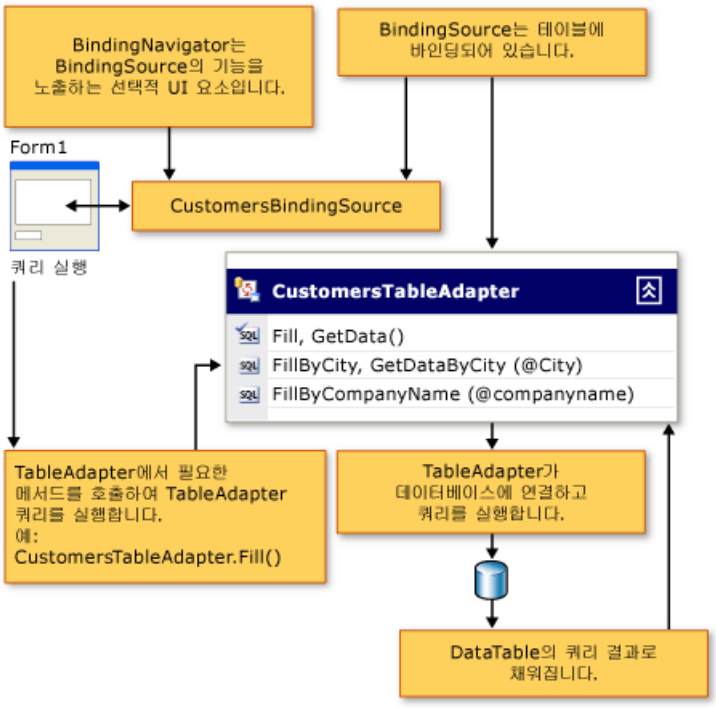
> **NOTE**
> 테이블 어댑터는 비주얼 스튜디오 디자이너에 의해 생성됩니다. 프로그래밍 방식으로 데이터 집합을 만드는 경우 .NET 클래스인 DataAdapter를 사용합니다.

TableAdapter 작업에 대한 자세한 내용은 다음 항목 중 하나로 직접 건너뛸 수 있습니다.

항목	DESCRIPTION
TableAdapter 만들기 및 구성	디자이너를 사용하여 TableAdapters를 만들고 구성하는 방법
매개 변수가 있는 TableAdapter 쿼리 만들기	사용자가 TableAdapter 프로시저 또는 쿼리에 인수를 제공할 수 있도록 하는 방법
TableAdapter를 사용하여 데이터베이스에 직접 액세스	테이블어댑터의 Dbdirect 방법을 사용하는 방법
데이터 세트를 채우는 동안 제약 조건 해제	데이터 업데이트 시 외래 키 제약 조건으로 작업하는 방법
TableAdapter의 기능 확장 방법	TableAdapters에 사용자 지정 코드를 추가하는 방법
XML 데이터를 데이터 세트에 읽어오기	XML작업

TableAdapter 개요

TableAdapters는 데이터베이스에 연결하고 쿼리 또는 저장 프로시저를 실행하고 DataTable을 반환된 데이터로 채우는 디자이너에서 생성한 구성 요소입니다. 또한 TableAdapters는 응용 프로그램에서 업데이트된 데이터를 데이터베이스로 다시 보냅니다. TableAdapter가 연결된 테이블의 스키마에 맞는 데이터를 반환하는 한 TableAdapter에서 원하는 만큼 쿼리를 실행할 수 있습니다. 다음 다이어그램에서는 TableAdapters가 메모리의 데이터베이스 및 기타 개체와 상호 작용하는 방법을 보여 주며 있습니다.



TableAdapter는 **데이터 집합 디자이너**로설계되었지만 TableAdapter 클래스는 의 DataSet중첩 된 클래스로 생성되지 않습니다. 각 데이터 집합에 특정한 별도의 네임스페이스에 있습니다. 예를 `NorthwindDataSet` 들어, 라는 데이터 집합이 있는 경우 , s와 DataTable연결된 `NorthwindDataSet` TableAdapters는 `NorthwindDataSetTableAdapters` 네임스페이스에 있을 것입니다. 특정 TableAdapter에 프로그래밍 방식으로 액세스하려면 TableAdapter의 새 인스턴스를 선언해야 합니다. 다음은 그 예입니다.

```
NorthwindDataSet northwindDataSet = new NorthwindDataSet();

NorthwindDataSetTableAdapters.CustomersTableAdapter customersTableAdapter =
    new NorthwindDataSetTableAdapters.CustomersTableAdapter();

customersTableAdapter.Fill(northwindDataSet.Customers);
```

```
Dim northwindDataSet As New NorthwindDataSet()
Dim customersTableAdapter As New NorthwindDataSetTableAdapters.CustomersTableAdapter()

customersTableAdapter.Fill(northwindDataSet.Customers)
```

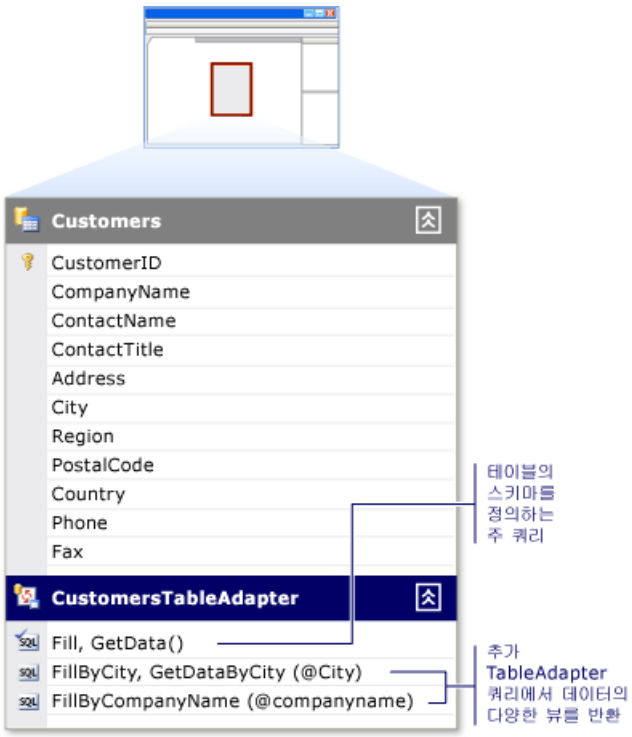
연결된 데이터 테이블 스키마

TableAdapter를 만들 때 초기 쿼리 또는 저장 프로시저를 사용하여 TableAdapter와 연결된 DataTable스키마를 정의합니다. TableAdapter의 `Fill` 메서드(TableAdapter의 관련 DataTable메서드를 채우는)를 호출하여 이 초기 쿼리 또는 저장 프로시저를 실행합니다. TableAdapter의 기본 쿼리에 대한 모든 변경 내용은 연결된 데이터 테이블의 스키마에 반영됩니다. 예를 들어 주 쿼리에서 열을 제거하면 연결된 데이터 테이블에서 열도 제거됩니다. TableAdapter의 추가 쿼리가 주 쿼리에 없는 열을 반환하는 SQL 문을 사용하는 경우 디자이너는 주 쿼리와 추가 쿼리 간에 열 변경 내용을 동기화하려고 시도합니다.

테이블 어댑터 업데이트 명령

TableAdapter의 업데이트 기능은 **TableAdapter 마법사의**기본 쿼리에서 사용할 수 있는 정보의 양에 따라 다릅니다. 예를 들어 여러 테이블(a `JOIN` 사용)에서 값을 가져오도록 구성된 TableAdapters는 기본 데이터베이스로 업데이트를 다시 보낼 수 있는 기능으로 처음에 만들어지지 않고 스칼라 값, 뷰 또는 집계 함수의 결과를 생성하지 않습니다. 그러나 속성 창에서 `INSERT` `UPDATE` 에서 `DELETE` 에서 및 명령을 **Properties** 수동으로 구성할 수 있습니다.

TableAdapter 쿼리



TableAdapters는 연결된 데이터 테이블을 채우기 위해 여러 쿼리를 포함할 수 있습니다. 각 쿼리가 연결된 데이터 테이블과 동일한 스키마를 준수하는 데이터를 반환하는 한 응용 프로그램에 필요한 만큼 TableAdapter에 대한 쿼리를 정의할 수 있습니다. 이 기능을 사용하면 TableAdapter가 서로 다른 기준에 따라 다른 결과를 로드할 수 있습니다.

예를 들어 응용 프로그램에 고객 이름이 있는 테이블이 포함된 경우 특정 문자로 시작하는 모든 고객 이름과 동일한 상태에 있는 모든 고객으로 테이블을 채우는 쿼리를 만들 수 있습니다. 지정된 상태의 `Customers` 고객으로 테이블을 채우려면 다음과 `FillByState` 같이 상태 값에 대한 매개 변수를 취하는 쿼리를 만들 수 `SELECT * FROM Customers WHERE State = @State` 있습니다. `FillByState` 메서드를 호출 하 고 다음과 같은 매개 변수 `CustomerTableAdapter.FillByState("WA")` 값에 전달 하 여 쿼리를 실행 합니다.

TableAdapter의 데이터 테이블과 동일한 스키마의 데이터를 반환하는 쿼리를 추가하는 것 외에도 스칼라(단일) 값을 반환하는 쿼리를 추가할 수 있습니다. 예를 들어 고객 수() `SELECT Count(*) From Customers` 수를 반환하는 쿼리는 `CustomersTableAdapter,` 반환되는 데이터가 테이블의 스키마를 따르지 않는 경우에도 유효합니다.

클리어전에필 속성

기본적으로 TableAdapter의 데이터 테이블을 채우기 위해 쿼리를 실행할 때마다 기존 데이터가 지워지고 쿼리 결과만 테이블에 로드됩니다. 쿼리에서 반환되는 데이터를 `ClearBeforeFill` 데이터 `false` 테이블의 기존 데이터로 추가하거나 병합할 경우 TableAdapter의 속성을 설정합니다. 데이터를 지우는지 여부에 관계없이 업데이트를 유지하려면 업데이트를 데이터베이스에 명시적으로 다시 보내야 합니다. 따라서 테이블을 채우는 다른 쿼리를 실행하기 전에 테이블에 있는 데이터에 대한 변경 내용을 저장해야 합니다. 자세한 내용은 TableAdapter를 사용하여 데이터 업데이트를 참조하십시오.

테이블어댑터 상속

TableAdapters는 구성된 클래스를 캡슐화하여 표준 데이터 어댑터의 기능을 확장합니다. DataAdapter 기본적으로 TableAdapter는 Component 클래스에서 상속되며 DataAdapter 클래스에 캐스팅할 수 없습니다. 클래스에 TableAdapter를 DataAdapter 캐스팅하면 InvalidCastException 오류가 발생합니다. TableAdapter의 기본 클래스를 변경하려면 **데이터 집합 디자이너의** Component TableAdapter의 기본 **클래스** 속성에서 파생되는 클래스를 지정할 수 있습니다.

테이블어댑터 방법 및 속성

TableAdapter 클래스는 .NET 형식이 아닙니다. 즉, 설명서 또는 **개체 브라우저에서** 찾을 수 없습니다. 앞에서 설명한 마법사 중 하나를 사용할 때 디자인 타임에 만들어집니다. TableAdapter를 만들 때 할당된 이름은 작업 중인 테이블의 이름을 기반으로 합니다. 예를 들어, 라는 `Orders` 데이터베이스의 테이블을 기반으로 TableAdapter를 만들 때 TableAdapter이름이 지정됩니다. `OrdersTableAdapter` TableAdapter의 클래스 이름은 **데이터 집합 디자이너의 Name** 속성을 사용하여 변경할 수 있습니다.

다음은 TableAdapters의 일반적으로 사용되는 방법 및 속성입니다.

멤버	DESCRIPTION
`TableAdapter.Fill`	TableAdapter의 관련 데이터 테이블을 TableAdapter `SELECT` 명령의 결과로 채웁니다.
`TableAdapter.Update`	변경 내용을 데이터베이스로 다시 보내고 업데이트의 영향을 받는 행 수를 나타내는 정수를 반환합니다. 자세한 내용은 TableAdapter를 사용하여 데이터 업데이트를 참조하십시오.
`TableAdapter.GetData`	데이터로 DataTable 채워진 새 새 를 반환합니다.
`TableAdapter.Insert`	데이터 테이블에 새 행을 만듭니다. 자세한 내용은 데이터베이스에 새 레코드 삽입을참조하십시오.
`TableAdapter.ClearBeforeFill`	`Fill` 메서드 중 하나를 호출하기 전에 데이터 테이블이 비워지는지 여부를 결정합니다.

테이블어댑터 업데이트 방법

TableAdapters는 데이터베이스에서 읽고 쓰기 위해 데이터 명령을 사용합니다. TableAdapter의 `Fill` 초기(주) 쿼리를 연관된 데이터 테이블의 스키마와 `InsertCommand` `UpdateCommand` `DeleteCommand` `TableAdapter.Update` 메서드와 연결된 의 . 및 명령을 만드는 기준으로 사용합니다. TableAdapter의 `Update` 메서드를 호출하면 **TableAdapter**쿼리 구성 마법사로 추가한 추가 쿼리 중 하나가 아니라 TableAdapter가 원래 구성되었을 때 만들어진 문을 실행합니다.

TableAdapter를 사용하면 일반적으로 수행하는 명령과 동일한 작업을 효과적으로 수행합니다. 예를 들어 어댑터의 `Fill` 메서드를 호출할 때 어댑터는 해당 `SelectCommand` 속성에서 데이터 명령을 실행하고 SqlDataReader데이터 판독기(예: 데이터 판독기)를 사용하여 결과 집합을 데이터 테이블에 로드합니다. 마찬가지로 `Update` 어댑터의 메서드를 호출하면 데이터 테이블의 변경된 `UpdateCommand` `InsertCommand` 각 `DeleteCommand` 레코드에 대해 해당 명령(의 . 및 속성)을 실행합니다.

> **NOTE**
> 기본 쿼리에 충분한 정보가 있는 `InsertCommand` 경우 `UpdateCommand` 의 `DeleteCommand` 및 명령은 TableAdapter가 생성될 때 기본적으로 만들어집니다. TableAdapter의 `SELECT` 기본 쿼리가 단일 테이블 문을 초과하는 경우 디자이너가 및 `InsertCommand` `UpdateCommand` `DeleteCommand` 을 생성할 수 없을 수 있습니다. 이러한 명령이 생성되지 않으면 `TableAdapter.Update` 메서드를 실행할 때 오류가 발생할 수 있습니다.

테이블 어댑터 생성DBDirectMethod

`InsertCommand` 은 `UpdateCommand` 및 `DeleteCommand` , TableAdapters는 데이터베이스에 대해 직접 실행할 수 있는 메서드를 사용하여 만들어집니다. 이러한 메서드 (, `TableAdapter.Insert` `TableAdapter.Update` 및) `TableAdapter.Delete` 직접 호출하여 데이터베이스의 데이터를 조작할 수 있습니다. 즉, 연결된 데이터 테이블에

대해 보류 중인 `TableAdapter.Update` 삽입, 업데이트 및 삭제를 처리하기 위해 호출하는 대신 코드에서 이러한 개별 메서드를 호출할 수 있습니다.

이러한 직접 메서드를 만들지 않으려면 TableAdapter의 **GenerateDbDirectMethod** 속성을 `false` **속성** 창에서 설정합니다. TableAdapter에 추가되는 추가 쿼리는 독립 실행형 쿼리입니다.

nullable 형식에 대한 TableAdapter 지원

TableAdapters는 null `Nullable(Of T)` 형식 `T?` 및 을 지원합니다. Visual Basic의 nullable 형식에 대한 자세한 내용은 Nullable 값 형식을 참조하세요. C#의 null 형식에 대한 자세한 내용은 nullable 형식 사용을참조하십시오.

테이블어댑터관리자 참조

기본적으로 TableAdapterManager 클래스는 관련 테이블을 포함하는 데이터 집합을 만들 때 생성됩니다. 클래스가 생성되지 않도록 하려면 데이터 집합의 `Hierarchical Update` 속성 값을 false로 변경합니다. Windows 양식 또는 WPF 페이지의 디자인 표면에 관계가 있는 테이블을 드래그하면 Visual Studio에서 클래스의 멤버 변수를 선언합니다. 데이터 바인딩을 사용하지 않는 경우 변수를 수동으로 선언해야 합니다.

TableAdapterManager 클래스는 .NET 형식이 아닙니다. 따라서 설명서에서 찾을 수 없습니다. 데이터 집합 만들기 프로세스의 일부로 디자인 타임에 만들어집니다.

다음은 클래스에서 자주 사용하는 메서드 `TableAdapterManager` 및 속성입니다.

멤버	DESCRIPTION
`UpdateAll` 메서드	모든 데이터 테이블의 모든 데이터를 저장합니다.
`BackUpDataSetBeforeUpdate` 속성	`TableAdapterManager.UpdateAll` 메서드를 실행하기 전에 데이터 집합의 백업 복사본을 만들지 여부를 결정합니다. 부울.
테이블이름 `TableAdapter` 속성	TableAdapter를 나타냅니다. 생성된 TableAdapterManager에는 관리하는 각 `TableAdapter` 에 대한 속성이 포함되어 있습니다. 예를 들어 고객 및 주문 테이블이 있는 데이터 집합은 `CustomersTableAdapter` `OrdersTableAdapter` 속성을 포함하는 TableAdapterManager를 사용하여 생성합니다.
`UpdateOrder` 속성	개별 삽입, 업데이트 및 삭제 명령의 순서를 제어합니다. 열거형의 값 `TableAdapterManager.UpdateOrderOption` 중 하나로 설정합니다. 기본적으로 는 `UpdateOrder` **InsertUpdateDelete**로 설정됩니다. 즉, 데이터 집합의 모든 테이블에 대해 삽입, 업데이트 및 삭제가 수행됩니다.

보안

로 설정된 TextCommandType 속성이 있는 데이터 명령을 사용하는 경우 데이터베이스로 전달하기 전에 클라이언트에서 전송되는 정보를 주의 깊게 확인합니다. 악의적인 사용자가 인증되지 않은 액세스 권한을 얻거나 데이터베이스를 손상시키기 위해 수정되었거나 추가된 SQL 문을 전송(주입)할 수도 있습니다. 사용자 입력을 데이터베이스로 전송하기 전에 항상 정보가 유효한지 확인합니다. 가능한 경우 항상 매개 변수화된 쿼리 또는 저장 프로시저를 사용하는 것이 가장 좋습니다.

참고 항목

- 데이터 집합 도구

n 계층 애플리케이션에서 데이터 세트 작업

2020-01-06 • 3 minutes to read • Edit Online

*N 계층 데이터 애플리케이션*은 여러 논리 *계층*으로 구분되는 데이터 중심 애플리케이션입니다. 다시 말해서 N 계층 데이터 애플리케이션은 여러 프로젝트로 구분되며 데이터 액세스 계층, 비즈니스 논리 계층 및 표시 계층이 각 프로젝트에 포함되는 애플리케이션입니다. 자세한 내용은 N 계층 데이터 응용 프로그램 개요를 참조 하세요.

TableAdapter 및 데이터 클래스를 개별 프로젝트로 생성할 수 있도록 형식화된 데이터 세트이 향상되었습니다. 따라서 애플리케이션 계층을 빠르게 분리하고 N 계층 데이터 애플리케이션을 생성하는 기능이 제공됩니다.

형식화 된 데이터 집합의 n 계층 지원을 통해 n 계층 디자인에 대 한 응용 프로그램 아키텍처의 반복적인 개발을 수행할 수 있습니다. 또한 코드를 두 개 이상의 프로젝트로 수동으로 분리 하는 요구 사항을 제거 합니다. **데이터 세트 디자이너**를 사용 하 여 데이터 계층 디자인을 시작 합니다. 애플리케이션 아키텍처에 N 계층 디자인을 적용할 준비가 되면 데이터 세트 클래스를 별도의 프로젝트로 생성하도록 데이터 세트의 **데이터 세트 프로젝트** 속성을 설정합니다.

참조

- DataSet
- TypedTableBase<T>

참조

- N 계층 데이터 애플리케이션 개요
- 연습: n 계층 데이터 애플리케이션 만들기
- n 계층 애플리케이션에서 TableAdapter에 코드 추가
- n 계층 애플리케이션에서 데이터 세트에 코드 추가
- n 계층 데이터 세트에 유효성 검사 추가
- 데이터 세트 및 TableAdapter를 다른 프로젝트로 분리
- 계층적 업데이트
- Visual Studio의 데이터 세트 도구
- Visual Studio에서 데이터 액세스
- TableAdapter 만들기 및 구성
- LINQ to SQL을 사용한 N 계층 및 원격 애플리케이션

데이터베이스 프로젝트 및 데이터 계층 응용 프로그램

2020-01-06 • 7 minutes to read • Edit Online

데이터베이스 프로젝트를 사용 하 여 새 데이터베이스, 새 Dac (데이터 계층 응용 프로그램)를 만들고 기존 데이터베이스 및 데이터 계층 응용 프로그램을 업데이트할 수 있습니다. 데이터베이스 프로젝트와 DAC 프로젝트를 모두 사용 하면 관리 코드 또는 네이티브 코드에 이러한 기술을 적용 하는 것과 거의 동일한 방식으로 데이터베이스 개발 작업에 버전 제어 및 프로젝트 관리 기법을 적용할 수 있습니다. DAC 프로젝트, 데이터베이스 프로젝트 또는 서버 프로젝트를 만들고 버전 제어에 배치 하 여 개발 팀이 데이터베이스 및 데이터베이스 서버에 대 한 변경 내용을 관리 하도록 지원할 수 있습니다. 팀 멤버는 파일을 체크 아웃 하 여 격리 된 개발 환경에서 변경 내용을 적용, 빌드 및 테스트 한 후 팀과 공유할 수 있습니다. 코드 품질을 보장 하기 위해 팀은 변경 내용을 프로덕션에 배포 하기 전에 스테이징 환경에서 데이터베이스의 특정 릴리스에 대 한 모든 변경 내용을 완료 하 고 테스트할 수 있습니다.

데이터 계층 응용 프로그램에서 지원 되는 데이터베이스 기능 목록은 SQL Server 개체에 대 한 DAC 지원을 참조 하세요. 데이터 계층 응용 프로그램에서 지원 하지 않는 기능을 데이터베이스에서 사용 하는 경우에는 데이터베이스 프로젝트를 사용 하 여 데이터베이스에 대 한 변경 내용을 관리 해야 합니다.

일반적인 상위 수준 작업

상위 수준 작업	지원 내용
데이터 계층 응용 프로그램 개발을 시작 합니다. DAC (데이터 계층 응용 프로그램)의 개념은 SQL Server 2008에서 도입 되었습니다. DAC에는 클라이언트-서버 또는 3 계층 응용 프로그램에서 사용 하는 SQL Server 데이터베이스 및 지원 인스턴스 개체에 대 한 정의가 포함 되어 있습니다. DAC에는 데이터베이스 개체 (예: 테이블 및 뷰)와 함께 인스턴스 엔터티 (예: 로그인)가 포함 됩니다. Visual Studio를 사용 하 여 DAC 프로젝트를 만들고, DAC 패키지 파일을 작성 하 고, SQL Server 데이터베이스 엔진 인스턴스에 배포 하기 위해 DAC 패키지 파일을 데이터베이스 관리자에 게 보낼 수 있습니다.	- 데이터 계층 애플리케이션 - SQL Server Management Studio
반복적인 데이터베이스 개발 수행: 개발자는 프로젝트의 일부를 체크 아웃 하 고 격리 된 개발 환경에서 업데이트할 수 있습니다. 이러한 유형의 환경을 사용 하 여 팀의 다른 멤버에 게 영향을 주지 않고 변경 내용을 테스트할 수 있습니다. 변경이 완료 되 면 파일을 다시 버전 제어에 체크 인 합니다. 그러면 다른 팀 멤버가 변경 내용을 가져와 테스트 서버에 빌드하고 배포할 수 있습니다.	- 프로젝트 기반 오프 라인 데이터베이스 개발 (SQL Server Data Tools) transact-sql 디버거 - (SQL Server Management Studio)
프로토타입 생성, 테스트 결과 확인 및 데이터베이스 스크립트 및 개체 수정: Transact-sql 편집기를 사용 하 여 이러한 일반적인 작업 중 하나를 수행할 수 있습니다.	- 쿼리 및 텍스트 편집기 (SQL Server Management Studio)

참조

- .NET용 Visual Studio 데이터 도구

N 계층 데이터 애플리케이션 개요

2020-01-06 • 11 minutes to read • Edit Online

N 계층 데이터 응용 프로그램은 여러 *계층*으로 구분 되는 데이터 응용 프로그램입니다. "분산 응용 프로그램" 및 "다중 계층 응용 프로그램" 이라는 n 계층 응용 프로그램은 클라이언트와 서버 간에 분산 된 불연속 계층으로 처리를 분리 합니다. 데이터에 액세스 하는 응용 프로그램을 개발 하는 경우 응용 프로그램을 구성 하는 다양 한 계층을 명확 하 게 구분 해야 합니다.

일반적인 N 계층 애플리케이션에는 프레젠테이션 계층, 중간 계층 및 데이터 계층이 포함됩니다. N 계층 응용 프로그램에서 다양 한 계층을 분리 하는 가장 쉬운 방법은 응용 프로그램에 포함 하려는 각 계층에 대 한 불연속 프로젝트를 만드는 것입니다. 예를 들어 프레젠테이션 계층은 Windows Forms 응용 프로그램 일 수 있지만 데이터 액세스 논리는 중간 계층에 위치한 클래스 라이브러리 일 수 있습니다. 또한 프레젠테이션 계층은 웹 서비스와 같은 서비스를 통해 중간 계층의 데이터 액세스 논리와 통신할 수 있습니다. 애플리케이션 구성 요소를 별도의 계층으로 분리하면 애플리케이션의 유지 관리성과 확장성이 높아집니다. 이렇게 하려면 전체 솔루션을 다시 디자인 해야 할 필요 없이 단일 계층에 적용할 수 있는 새로운 기술을 더 쉽게 채택할 수 있습니다. 또한 n 계층 응용 프로그램은 일반적으로 프레젠테이션 계층에서 격리를 유지 하는 중간 계층에 중요 한 정보를 저장 합니다.

Visual Studio에는 개발자가 n 계층 응용 프로그램을 만드는 데 도움이 되는 몇 가지 기능이 포함 되어 있습니다.

- 데이터 집합은 데이터 집합 (데이터 엔터티 계층) 및 Tableadapter (데이터 액세스 계층)를 불연속 프로젝트로 분리할 수 있도록 하는 **데이터 집합 프로젝트** 속성을 제공 합니다.
- Visual Studio의 LINQ to SQL 도구 는 DataContext 및 데이터 클래스를 별도의 네임 스페이스로 생성 하는 설정을 제공 합니다. 이렇게 하면 데이터 액세스 및 데이터 엔터티 계층을 논리적으로 분리할 수 있습니다.
- LINQ to SQL 는 응용 프로그램의 여러 계층 으로부터 DataContext를 가져오는 데 사용할 수 있는 Attach 메서드를 제공 합니다. 자세한 내용은 LINQ to SQL를 사용 하는 N 계층 및 원격 응용 프로그램을 참조 하세요.

프레젠테이션 계층

프레젠테이션 계층 은 사용자가 응용 프로그램과 상호 작용 하는 계층입니다. 추가 응용 프로그램 논리도 포함 하는 경우가 많습니다. 일반적인 프레젠테이션 계층 구성 요소에는 다음이 포함 됩니다.

- BindingSource 및 BindingNavigator와 같은 데이터 바인딩 구성 요소입니다.
- 프레젠테이션 계층에서 사용할 엔터티 클래스 LINQ to SQL 같은 데이터의 개체 표현입니다.

프레젠테이션 계층은 일반적으로 서비스 참조 (예: Visual Studio 응용 프로그램의 Windows Communication Foundation 서비스 및 WCF Data Services)를 사용 하 여 중간 계층에 액세스 합니다. 프레젠테이션 계층은 데이터 계층에 직접 액세스 하지 않습니다. 프레젠테이션 계층은 중간 계층의 데이터 액세스 구성 요소를 기준으로 데이터 계층과 통신 합니다.

중간 계층

중간 계층 은 프레젠테이션 계층과 데이터 계층에서 서로 통신 하는 데 사용 하는 계층입니다. 일반적인 중간 계층 구성 요소에는 다음이 포함 됩니다.

- 비즈니스 규칙 및 데이터 유효성 검사 등의 비즈니스 논리
- 데이터 액세스 구성 요소 및 논리 (예:
  - Tableadapter 및 Dataadapter 및 datareaders.

- LINQ to SQL 엔터티 클래스와 같은 데이터의 개체 표현입니다.
- 인증, 권한 부여, 개인 설정 등의 일반적인 응용 프로그램 서비스입니다.

다음 그림에서는 Visual Studio에서 사용할 수 있는 기능 및 기술과 n 계층 응용 프로그램의 중간 계층에 적합할 수 있는 기술을 보여 줍니다.

중간 계층](../data-tools/media/ntiermid.png) 중간 계층 구성 요소 ![

중간 계층은 일반적으로 데이터 연결을 사용 하 여 데이터 계층에 연결 합니다. 이 데이터 연결은 일반적으로 데이터 액세스 구성 요소에 저장 됩니다.

데이터 계층

데이터 계층 은 기본적으로 애플리케이션의 데이터를 저장하는 서버입니다(예: SQL Server를 실행하는 서버).

다음 그림에서는 Visual Studio에서 사용할 수 있는 기능 및 기술과 n 계층 응용 프로그램의 데이터 계층에 적합할 수 있는 기술을 보여 줍니다.

데이터 계층](../data-tools/media/ntierdatatier.png) 데이터 계층 구성 요소 ![

프레젠테이션 계층의 클라이언트에서 직접 데이터 계층에 액세스할 수 없습니다. 대신, 중간 계층의 데이터 액세스 구성 요소는 프레젠테이션과 데이터 계층 간의 통신에 사용 됩니다.

N 계층 개발에 대 한 도움말

다음 항목에서는 n 계층 응용 프로그램 작업에 대 한 정보를 제공 합니다.

데이터 세트 및 TableAdapter를 다른 프로젝트로 분리

연습: N 계층 데이터 애플리케이션 만들기

LINQ to SQL을 사용한 N 계층 및 원격 애플리케이션

참조

- 연습: N 계층 데이터 애플리케이션 만들기
- 계층적 업데이트
- Visual Studio의 데이터 세트 도구
- Visual Studio에서 데이터 액세스

Visual Studio에서 데이터베이스 만들기 및 테이블 추가

2020-01-06 • 16 minutes to read • Edit Online

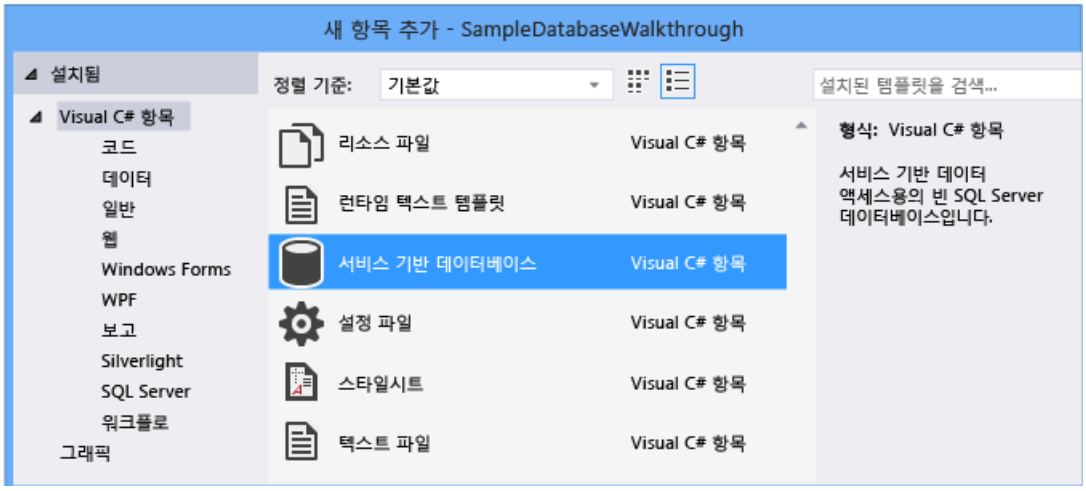
Visual Studio를 사용 하 여 SQL Server Express LocalDB에서 로컬 데이터베이스 파일을 만들고 업데이트할 수 있습니다. Visual Studio의 **SQL Server 개체 탐색기** 도구 창에서 transact-sql 문을 실행 하 여 데이터베이스를 만들 수도 있습니다. 이 항목에서는 *.mdf* 파일을 만들고 테이블 디자이너를 사용 하 여 테이블 및 키를 추가 합니다.

전제 조건

이 연습을 완료 하려면 Visual Studio에 설치 된 **.net 데스크톱 개발** 및 **데이터 저장소 및 처리** 작업을 수행 해야 합니다. 이를 설치 하려면 **Visual Studio 설치 관리자** 열고 수정 하려는 Visual Studio 버전 옆의 **수정 (또는 추가 > 수정**)을 선택 합니다.

프로젝트 및 로컬 데이터베이스 파일 만들기

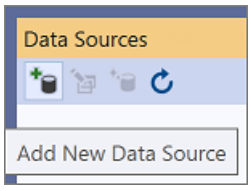
1. 새 **Windows Forms 앱** 프로젝트를 만들고 이름을 **sampledatabasewalkthrough**로로 다시 만듭니다.
2. 메뉴 모음에서 **프로젝트** > **새 항목 추가**를 선택 합니다.
3. 항목 템플릿 목록에서 아래로 스크롤하여 **서비스 기반 데이터베이스**를 선택 합니다.



4. 데이터베이스 이름을 **sampledatabase.mdf**로 지정한 다음 **추가**를 클릭 합니다.

데이터 원본 추가

1. **데이터 소스** 창이 열려 있지 않은 경우 **Shift+alt+D** 를 누르거나 메뉴 모음에서 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 엽니다.
2. **데이터 소스** 창에서 **새 데이터 소스 추가**를 선택 합니다.



데이터 원본 구성 마법사가 열립니다.

3. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택 하 고 **다음**을 선택 합니다.

4. **데이터베이스 모델 선택** 페이지에서 **다음** 을 선택 하 여 기본값 (데이터 집합)을 적용 합니다.
5. **데이터 연결 선택** 페이지의 드롭다운 목록에서 **sampledatabase.mdf** 파일을 선택 하 고 **다음**을 선택 합니다.
6. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 선택 합니다.
7. **데이터베이스 개체 선택** 페이지에서 데이터베이스에 개체가 포함 되어 있지 않다는 메시지가 표시 됩니다. **마침**을 선택합니다.

데이터 연결의 속성 보기

데이터 연결의 속성 창을 열어 *sampledatabase.mdf* 파일에 대 한 연결 문자열을 볼 수 있습니다.

- **보기** > **SQL Server 개체 탐색기** 를 선택 하 여 **SQL Server 개체 탐색기** 창을 엽니다. **(Localdb)\MSSQLLocalDB** > **데이터베이스**를 확장 한 다음 *sampledatabase.mdf* 를 마우스 오른쪽 단추로 클릭 하 고 **속성**을 선택 합니다.
- 또는 해당 창이 아직 열려 있지 않은 경우 **보기** > **서버 탐색기**를 선택할 수 있습니다. **데이터 연결** 노드를 확장 하 고 *sampledatabase.mdf*를 마우스 오른쪽 단추로 클릭 한 다음 **속성**을 선택 하 여 속성 창을 엽니다.

> **TIP**
> 데이터 연결 노드를 확장할 수 없거나 Sampledatabase.mdf 연결이 목록에 없는 경우 서버 탐색기 도구 모음에서 **데이터베이스에 연결** 단추를 선택 합니다. **연결 추가** 대화 상자에서 **데이터 원본**아래에 **Microsoft SQL Server 데이터베이스 파일이** 선택 되어 있는지 확인 한 다음 sampledatabase.mdf 파일을 찾아 선택 합니다. **확인**을선택 하 여 연결 추가를 완료 합니다.

테이블 디자이너를 사용 하 여 테이블 및 키 만들기

이 섹션에서는 두 개의 테이블, 각 테이블의 기본 키 및 몇 개의 샘플 데이터 행을 만듭니다. 또한 외래 키를 만들어 한 테이블의 레코드가 다른 테이블의 레코드에 해당 하는 방식을 지정 합니다.

Customers 테이블 만들기

1. **서버 탐색기**에서 **데이터 연결** 노드를 확장 한 다음 **sampledatabase.mdf** 노드를 확장 합니다.

   데이터 연결 노드를 확장할 수 없거나 Sampledatabase.mdf 연결이 목록에 없는 경우 서버 탐색기 도구 모음에서 **데이터베이스에 연결** 단추를 선택 합니다. **연결 추가** 대화 상자에서 **데이터 원본**아래에 **Microsoft SQL Server 데이터베이스 파일이** 선택 되어 있는지 확인 한 다음 sampledatabase.mdf 파일을 찾아 선택 합니다. **확인**을선택 하 여 연결 추가를 완료 합니다.

2. **테이블** 을 마우스 오른쪽 단추로 클릭 하 고 **새 테이블 추가**를 선택 합니다.

   테이블 디자이너가 열리고 사용자가 만드는 테이블의 단일 열을 나타내는 한 개의 기본 행이 포함된 표가 표시됩니다. 표에 행을 추가하여 테이블에 열을 추가합니다.

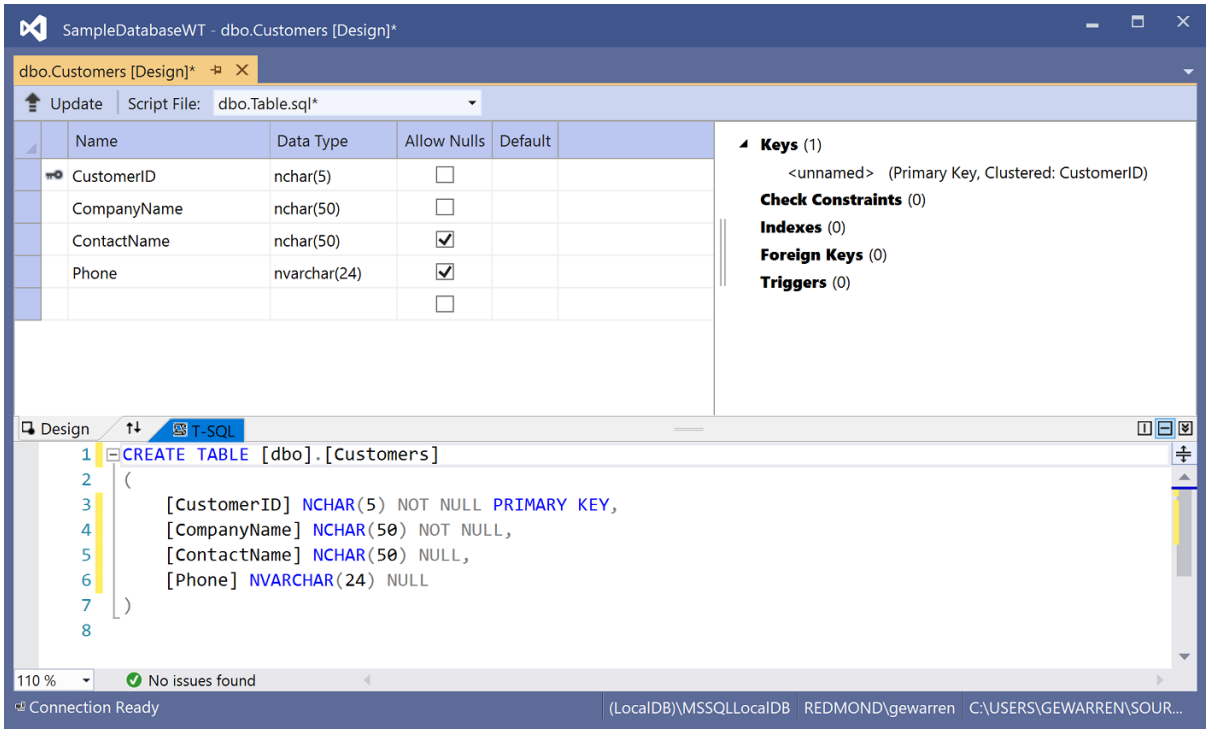
3. 표에서 다음 각 항목에 대한 행을 추가합니다.

열 이름	데이터 형식	NULL 허용
`CustomerID`	`nchar(5)`	False(선택 취소)
`CompanyName`	`nvarchar(50)`	False(선택 취소)
`ContactName`	`nvarchar (50)`	True(선택)
`Phone`	`nvarchar (24)`	True(선택)

4. `CustomerID` 행을 마우스 오른쪽 단추로 클릭 한 다음 **기본 키 설정**을 선택 합니다.
5. 기본 행 (`Id`)을 마우스 오른쪽 단추로 클릭 한 다음 **삭제**를 선택 합니다.
6. 다음 샘플과 일치하도록 스크립트 창에서 첫 번째 줄을 업데이트하여 Customers 테이블 이름을 지정합니다.

```
CREATE TABLE [dbo].[Customers]
```

다음과 같이 표시되어야 합니다.



7. **테이블 디자이너**의 왼쪽 위 모서리에서 **업데이트**를 선택 합니다.
8. **데이터베이스 업데이트 미리 보기** 대화 상자에서 **데이터베이스 업데이트**를 선택 합니다.

Customers 테이블은 로컬 데이터베이스 파일에 생성 됩니다.

Orders 테이블 만들기

1. 다른 테이블을 추가한 다음, 다음 표의 각 항목에 대한 행을 추가합니다.

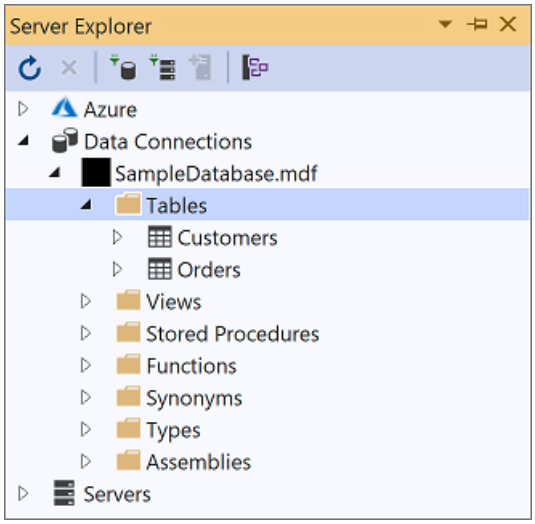
열 이름	데이터 형식	NULL 허용
`OrderID`	`int`	False(선택 취소)
`CustomerID`	`nchar(5)`	False(선택 취소)
`OrderDate`	`datetime`	True(선택)
`OrderQuantity`	`int`	True(선택)

2. **OrderID** 를 기본 키로 설정한 다음 기본 행을 삭제 합니다.
3. 다음 샘플과 일치하도록 스크립트 창에서 첫 번째 줄을 업데이트하여 Orders 테이블 이름을 지정합니다.

```
CREATE TABLE [dbo].[Orders]
```

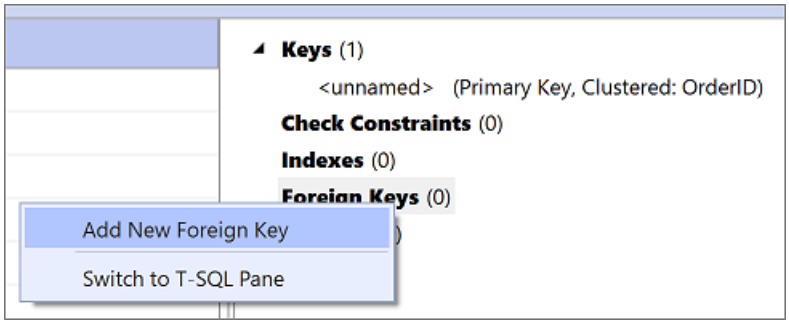
4. **테이블 디자이너**의 왼쪽 위 모서리에서 **업데이트**를 선택 합니다.
5. **데이터베이스 업데이트 미리 보기** 대화 상자에서 **데이터베이스 업데이트**를 선택 합니다.

   Orders 테이블은 로컬 데이터베이스 파일에 생성 됩니다. 서버 탐색기의 **테이블** 노드를 확장 하면 다음 두 테이블이 표시 됩니다.



외래 키 만들기

1. Orders 테이블의 테이블 디자이너 그리드 오른쪽에 있는 컨텍스트 창에서 **외래 키** 를 마우스 오른쪽 단추로 클릭 하 고 **새 외래 키 추가**를 선택 합니다.



2. 표시 되는 텍스트 상자에서 **ToTable** 텍스트를 **고객**으로 바꿉니다.
3. T-sql 창에서 마지막 줄을 다음 샘플과 일치 하도록 업데이트 합니다.

```
CONSTRAINT [FK_Orders_Customers] FOREIGN KEY ([CustomerID]) REFERENCES [Customers]([CustomerID])
```

4. **테이블 디자이너**의 왼쪽 위 모서리에서 **업데이트**를 선택 합니다.
5. **데이터베이스 업데이트 미리 보기** 대화 상자에서 **데이터베이스 업데이트**를 선택 합니다.

   외래 키가 생성 됩니다.

테이블을 데이터로 채우기

1. **서버 탐색기** 또는 **SQL Server 개체 탐색기**에서 예제 데이터베이스에 대 한 노드를 확장 합니다.
2. **테이블** 노드에 대 한 바로 가기 메뉴를 열고 **새로 고침**을 선택한 다음 **테이블** 노드를 확장 합니다.
3. Customers 테이블의 바로 가기 메뉴를 열고 **테이블 데이터 표시**를 선택 합니다.

4. 일부 고객에 게 필요한 모든 데이터를 추가 합니다.

   고객 ID를 5자로 지정할 수 있지만 나중에 이 절차에서 사용할 수 있도록 기억하기 쉬운 1자 정도를 선택하면 됩니다.

5. Orders 테이블에 대 한 바로 가기 메뉴를 열고 **테이블 데이터 표시**를 선택 합니다.

6. 일부 주문의 데이터를 추가 합니다.

   > **IMPORTANT**
   > 모든 주문 ID와 주문 수량이 정수이고 각 고객 ID가 Customers 테이블의 **CustomerID** 열에 지정한 값과 일치하는지 확인합니다.

7. 메뉴 모음에서 **파일** > **모두 저장**을 선택 합니다.

참조

- Visual Studio에서 데이터 액세스

새 연결 추가

2020-01-06 • 6 minutes to read • Edit Online

데이터베이스 또는 서비스에 대 한 연결을 테스트 하 고 **서버 탐색기**, **클라우드 탐색기**또는 **SQL Server 개체 탐색기**를 사용 하 여 데이터베이스 내용과 스키마를 탐색할 수 있습니다. 이러한 창의 기능은 일부 범위에 겹칩니다. 기본 차이점은 다음과 같습니다.

- 서버 탐색기

  기본적으로 Visual Studio에 설치 됩니다. 를 사용 하 여 연결을 테스트 하 고 SQL Server 데이터베이스, ADO.NET 공급자가 설치 된 다른 데이터베이스 및 일부 Azure 서비스를 볼 수 있습니다. 시스템 성능 카운터, 이벤트 로그 및 메시지 큐와 같은 하위 수준 개체도 보여 줍니다. 데이터 원본에 ADO.NET 공급자가 없으면 여기에 표시 되지 않지만 프로그래밍 방식으로 연결 하 여 Visual Studio에서 사용할 수 있습니다.

- 클라우드 탐색기

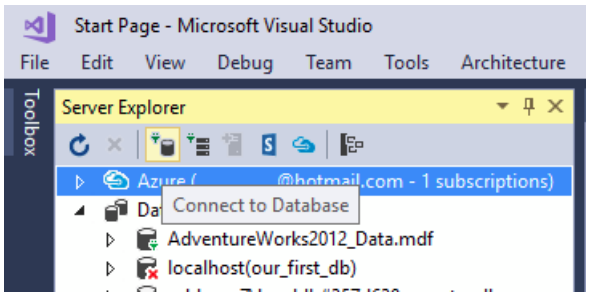
  Visual Studio Marketplace에서 Visual Studio 확장으로이 창을 수동으로 설치 합니다. 는 Azure 서비스를 탐색 하 고 연결 하기 위한 특수 기능을 제공 합니다.

- SQL Server 개체 탐색기

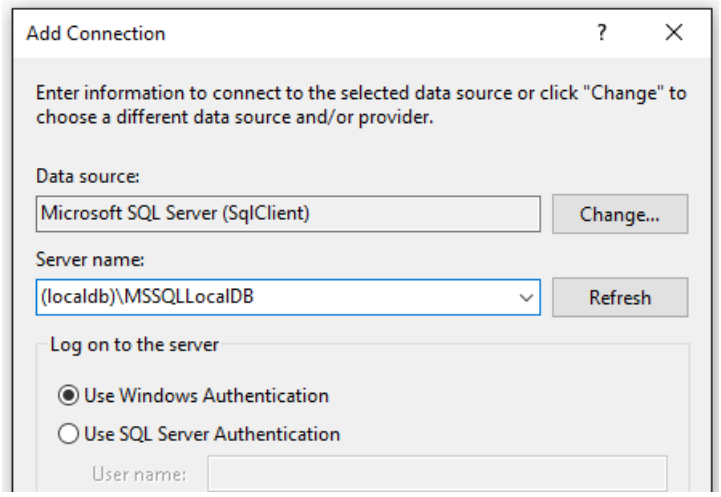
  SQL Server Data Tools와 함께 설치 되 고 **보기** 메뉴 아래에 표시 됩니다. 표시 되지 않으면 제어판의 **프로그램 및 기능** 으로 이동 하 여 Visual Studio를 찾은 다음 **변경** 을 선택 하 여 SQL Server Data Tools에 대 한 확인란을 선택한 후 설치 관리자를 다시 실행 합니다. **SQL Server 개체 탐색기** 를 사용 하 여 SQL 데이터베이스 (ADO.NET 공급자가 있는 경우)를 보고, 새 데이터베이스를 만들고, 스키마를 수정 하 고, 저장 프로시저를 만들고, 연결 문자열을 검색 하 고, 데이터를 볼 수 있습니다. ADO.NET 공급자가 설치 되지 않은 SQL 데이터베이스는 여기에 표시 되지 않지만 프로그래밍 방식으로 연결할 수 있습니다.

서버 탐색기에서 연결 추가

데이터베이스에 대 한 연결을 만들려면 **서버 탐색기**에서 **연결 추가** 아이콘을 클릭 하거나 **데이터 연결** 노드에서 **서버 탐색기** 을 마우스 오른쪽 단추로 클릭 하 고 **연결 추가**를 선택 합니다. 여기에서 다른 서버, SharePoint 서비스 또는 Azure 서비스의 데이터베이스에 연결할 수도 있습니다.

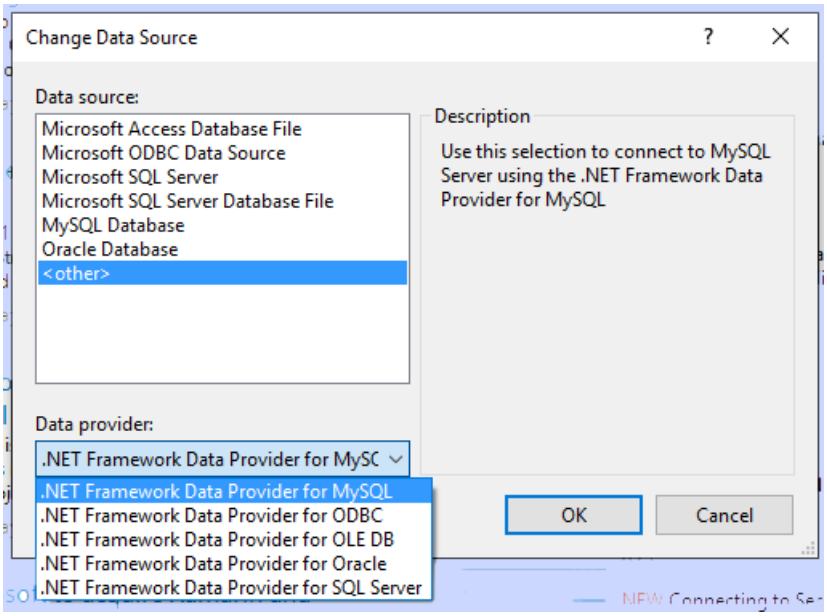


그러면 **연결 추가** 대화 상자가 나타납니다. 여기에서 SQL Server LocalDB 인스턴스의 이름을 입력 했습니다.



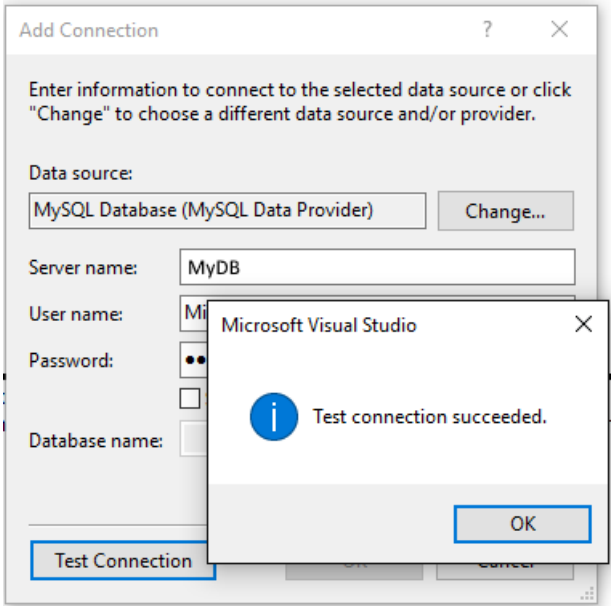
공급자 변경

데이터 원본이 원하는 항목이 아닌 경우 **변경** 단추를 클릭 하 여 새 데이터 원본 및/또는 새 ADO.NET 데이터 공급자를 선택 합니다. 새 공급자는 구성 된 방법에 따라 자격 증명을 요청할 수 있습니다.



연결 테스트

데이터 원본을 선택한 후 **연결 테스트**를 클릭 합니다. 성공 하지 못하면 공급 업체의 설명서에 따라 문제를 해결 해야 합니다.



테스트에 성공 하면 기본 데이터베이스 또는 서비스를 기반으로 하는 *데이터 모델* 을 의미 하는 Visual Studio 용어 인 *데이터 소스*를 만들 준비가 된 것입니다.

참조

- .NET용 Visual Studio 데이터 도구

방법: 연결 문자열 저장 및 편집

2020-01-06 • 9 minutes to read • Edit Online

Visual Studio 응용 프로그램의 연결 문자열은 응용 프로그램 구성 파일 (응용 프로그램 설정이 라고도 함)에 저장되거나 응용 프로그램에서 직접 하드 코딩 됩니다. 애플리케이션 구성 파일에 연결 문자열을 저장하면 애플리케이션 유지 관리 작업을 간소화할 수 있습니다. 연결 문자열을 변경해야 하는 경우 소스 코드에서 문자열을 변경한 다음, 애플리케이션을 다시 컴파일하는 대신 애플리케이션 설정 파일에서 문자열을 업데이트할 수 있습니다.

암호와 같은 중요한 정보를 연결 문자열 내에 저장하면 애플리케이션 보안 문제가 발생할 수 있습니다. 애플리케이션 구성 파일에 저장된 연결 문자열은 암호화되거나 난독 처리되지 않으므로 다른 사람이 파일에 액세스하여 해당 내용을 볼 수 있습니다. 데이터베이스 액세스를 제어할 경우에는 Windows 통합 보안을 사용하는 방법이 더 안전합니다.

Windows 통합 보안을 사용하도록 선택하지 않았는데 데이터베이스에 사용자 이름과 암호가 필요한 경우 연결 문자열에서 사용자 이름과 암호를 생략할 수 있습니다. 그러나 데이터베이스에 정상적으로 연결하려면 애플리케이션이 이 정보를 제공해야 합니다. 예를 들어 이 정보를 입력하라는 메시지를 사용자에게 표시하고 런타임에 연결 문자열을 동적으로 작성하는 대화 상자를 만들 수 있습니다. 데이터베이스로 전송되는 와중에 정보를 가로챌 수 있는 경우에도 보안 문제가 발생할 수 있습니다. 자세한 내용은 연결 정보 보호를 참조하세요.

데이터 소스 구성 마법사 내에서 연결 문자열을 저장 하려면

데이터 소스 구성 마법사의 **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 연결을 저장 하는 옵션을 선택 합니다.

연결 문자열을 애플리케이션 설정에 직접 저장하려면

1. **솔루션 탐색기**에서 **내 프로젝트** 아이콘(Visual Basic) 또는 **속성** 아이콘(C#)을 두 번 클릭하여 **프로젝트 디자이너**를 엽니다.
2. **설정** 탭을 선택합니다.
3. 연결 문자열의 **이름**을 입력합니다. 코드에서 연결 문자열에 액세스할 때 이 이름을 참조합니다.
4. **형식**을 **연결 문자열**로 설정합니다.
5. **범위**는 **애플리케이션**으로 설정된 상태로 유지합니다.
6. **값** 필드에 연결 문자열을 입력 하거나 **값** 필드에서 **줄임표** (...) 단추를 클릭 하 여 연결 **속성** 대화 상자를 열고 연결 문자열을 작성 합니다.

응용 프로그램 설정에 저장 된 연결 문자열 편집

프로젝트 디자이너를 사용하여 애플리케이션 설정에 저장된 연결 정보를 수정할 수 있습니다.

애플리케이션 설정에 저장된 연결 문자열을 편집하려면

1. **솔루션 탐색기**에서 **내 프로젝트** 아이콘(Visual Basic) 또는 **속성** 아이콘(C#)을 두 번 클릭하여 **프로젝트 디자이너**를 엽니다.
2. **설정** 탭을 선택합니다.
3. 편집 하려는 연결을 찾아 **값** 필드에서 텍스트를 선택 합니다.
4. **값** 필드에서 연결 문자열을 편집 하거나 **값** 필드에서 **줄임표** (...) 단추를 클릭 하 여 연결 **속성** 대화 상자를 사용하 여 연결을 편집 합니다.

데이터 집합에 대 한 연결 문자열 편집

데이터 집합의 각 TableAdapter에 대 한 연결 정보를 수정할 수 있습니다.

데이터 집합의 **TableAdapter**에 대 한 연결 문자열을 편집 하려면

1. **솔루션 탐색기**에서 편집 하려는 연결을 포함 하는 데이터 집합 (**.xsd** 파일)을 두 번 클릭 합니다.
2. 편집 하려는 연결이 있는 **TableAdapter** 또는 쿼리를 선택 합니다.
3. **속성** 창에서 **연결 노드**를 확장 합니다.
4. 연결 문자열을 신속 하 게 수정 하려면 **ConnectionString** 속성을 편집 하거나 **연결** 속성에서 아래쪽 화살표를 클릭 하 고 **새 연결**을 선택 합니다.

보안

암호와 같은 중요한 정보를 연결 문자열 내에 저장하면 애플리케이션 보안 문제가 발생할 수 있습니다. 데이터베이스 액세스를 제어할 경우에는 Windows 통합 보안을 사용하는 방법이 더 안전합니다. 자세한 내용은 연결 정보 보호를 참조하세요.

참조

- 연결 추가

Access 데이터베이스의 데이터에 연결

2020-01-06 • 10 minutes to read • Edit Online

Visual Studio를 사용 하 여 Access 데이터베이스 (*.mdb* 파일 또는 *.accdb* 파일)에 연결할 수 있습니다. 연결을 정의 한 후 **데이터 원본** 창에 데이터가 나타납니다. 여기에서 테이블 또는 뷰를 디자인 화면으로 끌 수 있습니다.

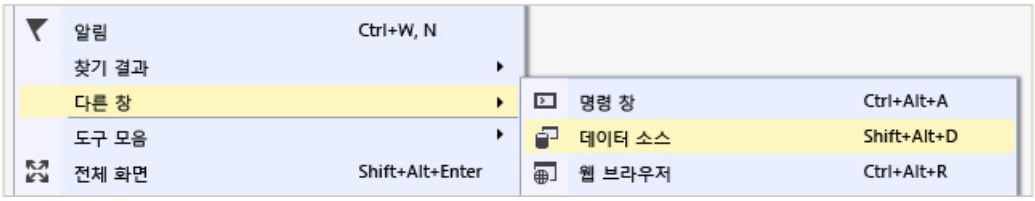
전제 조건

이러한 절차를 사용 하려면 Windows Forms 또는 WPF 프로젝트와 Access 데이터베이스 (*.accdb* 파일) 또는 access 2000-2003 데이터베이스 (*.mdb* 파일)가 필요 합니다. 파일 형식에 해당하는 절차를 따릅니다.

.Accdb 파일에 대 한 데이터 집합 만들기

다음 절차를 사용 하 여 Office 365, 액세스 2013, 액세스 2010 또는 Access 2007로 만든 데이터베이스에 연결 합니다.

1. Visual Studio에서 Windows Forms 또는 WPF 응용 프로그램 프로젝트를 엽니다.
2. **데이터 소스** 창을 열려면 **보기** 메뉴에서 **다른 창** > **데이터 소스**를 선택 합니다.



3. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.

   데이터 원본 구성 마법사가 열립니다.

4. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택 하 고 **다음**을 선택 합니다.
5. **데이터베이스 모델 선택** 페이지에서 **데이터 집합** 을 선택 하 고 **다음**을 선택 합니다.
6. **데이터 연결 선택** 페이지에서 **새 연결**을 선택하여 새 데이터 연결을 구성합니다.

   연결 추가 대화 상자가 열립니다.

7. **데이터 원본을 Microsoft Access 데이터베이스 파일 (OLE DB)** 로 설정 하지 않은 경우에는 **변경** 단추를 선택 합니다.

   데이터 소스 변경 대화 상자가 열립니다. 데이터 원본 목록에서 **Microsoft Access 데이터베이스 파일**을 선택 합니다. **데이터 공급자** 드롭다운에서 **.NET Framework Data Provider OLE DB**를 선택한 다음 **확인**을 선택 합니다.

8. **데이터베이스 파일 이름**옆에 있는 **찾아보기** 를 선택한 다음 *.Accdb* 파일로 이동 하 여 **열기**를 선택 합니다.
9. 사용자 이름 및 암호 (필요한 경우)를 입력 한 다음 **확인**을 선택 합니다.
10. **데이터 연결 선택** 페이지에서 **다음** 을 선택 합니다.

    데이터 파일이 현재 프로젝트에 없다고 알려주는 대화 상자가 표시 될 수 있습니다. **예** 또는 **아니요**를 선택 합니다.

11. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음** 을 선택 합니다.

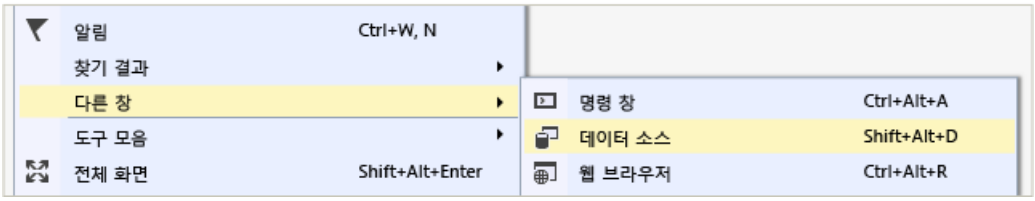
12. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
13. 데이터 집합에 포함 하려는 테이블이 나 뷰를 선택한 다음 **마침**을 선택 합니다.

    데이터 세트가 프로젝트에 추가되고 테이블과 뷰가 **데이터 원본** 창에 나타납니다.

.Mdb 파일에 대 한 데이터 집합 만들기

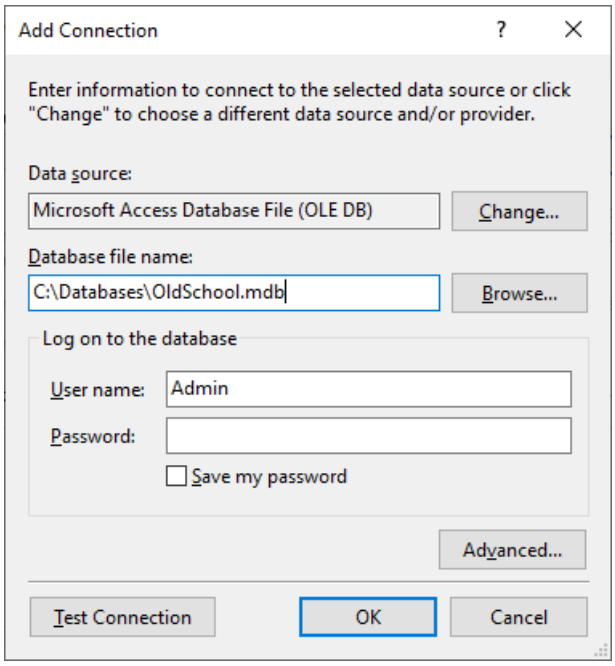
다음 절차에 따라 Access 2000-2003를 사용 하 여 만든 데이터베이스에 연결 합니다.

1. Visual Studio에서 Windows Forms 또는 WPF 응용 프로그램 프로젝트를 엽니다.
2. **보기** 메뉴에서 **다른 Windows** > **데이터 원본**을 선택 합니다.



3. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.

   데이터 원본 구성 마법사가 열립니다.
4. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택 하 고 **다음**을 선택 합니다.
5. **데이터베이스 모델 선택** 페이지에서 **데이터 집합** 을 선택 하 고 **다음**을 선택 합니다.
6. **데이터 연결 선택** 페이지에서 **새 연결**을 선택하여 새 데이터 연결을 구성합니다.
7. 데이터 원본이 **Microsoft Access 데이터베이스 파일 (OLE DB)** 이 아니면 **변경** 을 선택 하 여 **데이터 소스 변경** 대화 상자를 열고 **Microsoft Access 데이터베이스 파일**을 선택한 다음 **확인**을 선택 합니다.
8. **데이터베이스 파일 이름**에서 연결 하려는 *.mdb* 파일의 경로와 이름을 지정한 다음 **확인**을 선택 합니다.



9. **데이터 연결 선택** 페이지에서 **다음** 을 선택 합니다.
10. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음** 을 선택 합니다.
11. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
12. 데이터 집합에서 원하는 테이블 또는 뷰를 선택 하 고 **마침**을 선택 합니다.

데이터 세트가 프로젝트에 추가되고 테이블과 뷰가 **데이터 원본** 창에 나타납니다.

다음 단계

방금 만든 데이터 집합은 **데이터 소스** 창에서 사용할 수 있습니다. 다음 작업 중 어떤 작업이든 수행할 수 있습니다.

- **데이터 소스** 창에서 항목을 선택 하 고 폼 또는 디자인 화면으로 끌어 옵니다 (Visual Studio에서 데이터에 컨트롤 Windows Forms 바인딩 또는 WPF 데이터 바인딩 개요참조).
- **데이터 세트 디자이너**에서 데이터 원본을 열어 데이터 세트를 구성하는 개체를 추가하거나 편집합니다.
- 데이터 집합의 데이터 테이블에 대 한 ColumnChanging 또는 RowChanging 이벤트에 유효성 검사 논리를 추가 합니다 (데이터 집합의 데이터 유효성 검사참조).

참조

- 연결 추가
- WPF 데이터 바인딩 개요
- Windows Forms 데이터 바인딩

새 데이터 원본 추가

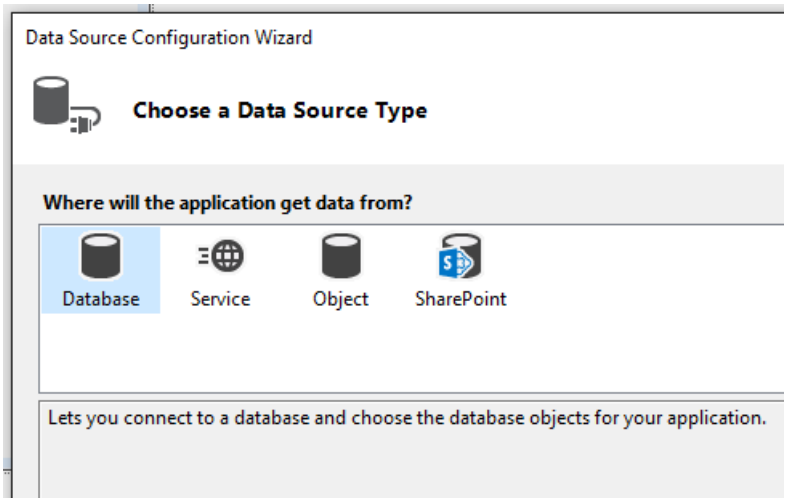
2020-03-23 • 15 minutes to read • Edit Online

Visual Studio의 .NET 데이터 도구의 컨텍스트에서 *데이터 원본이라는* 용어는 데이터 저장소에 연결하고 데이터를 .NET 응용 프로그램에서 사용할 수 있도록 하는 .NET 개체를 나타냅니다. Visual Studio 디자이너는 데이터 원본의 출력을 사용하여 **데이터** 원본 창에서 데이터베이스 개체를 드래그앤드롭할 때 데이터를 양식에 바인딩하는 상용구 코드를 생성할 수 있습니다. 이러한 종류의 데이터 원본은 다음과 같은 것일 수 있습니다.

- 일종의 데이터베이스와 연결된 엔터티 프레임워크 모델의 클래스입니다.
- 일종의 데이터베이스와 연결된 데이터 집합입니다.
- WCF(Windows 통신 재단) 데이터 서비스 또는 REST 서비스와 같은 네트워크 서비스를 나타내는 클래스입니다.
- SharePoint 서비스를 나타내는 클래스입니다.
- 솔루션의 클래스 또는 컬렉션입니다.

> **NOTE**
> 데이터 바인딩 기능, 데이터 집합, 엔터티 프레임워크, LINQ에서 SQL, WCF 또는 SharePoint를 사용하지 않는 경우 "데이터 원본"의 개념이 적용되지 않습니다. SQLCommand 개체를 사용하여 데이터베이스에 직접 연결하고 데이터베이스와 직접 통신하기만 하면 됩니다.

Windows 양식 또는 Windows 프레젠테이션 기초 응용 프로그램에서 **데이터 원본 구성 마법사를** 사용하여 데이터 원본을 만들고 편집합니다. Entity Framework의 경우 먼저 엔터티 클래스를 만든 다음 **새** > 데이터**원본 프로젝트에 추가를** 선택하여 마법사를 시작합니다(이 문서의 자세한 설명).



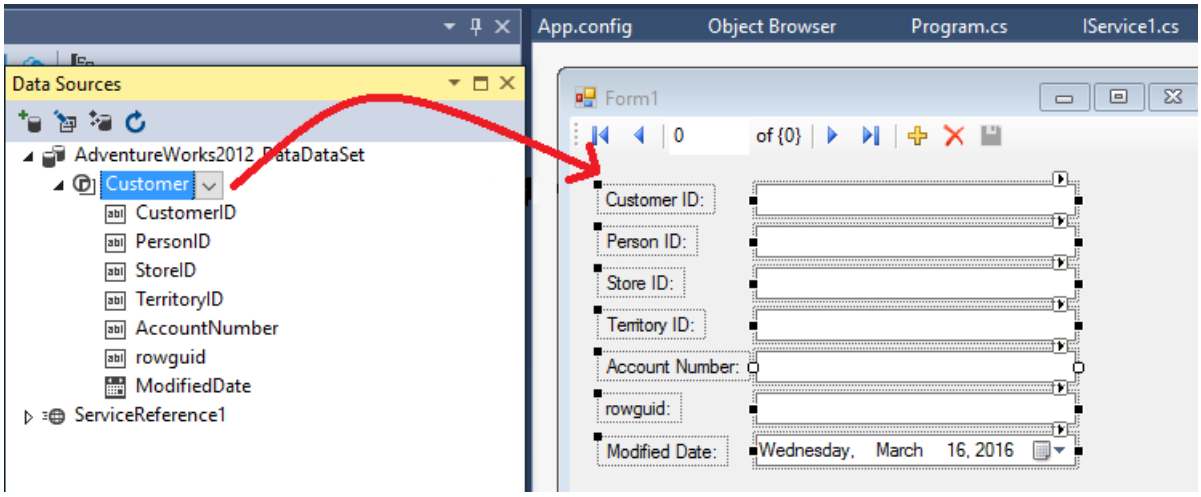
데이터 소스 창

데이터 원본을 만든 후 데이터 **원본** 도구 창에 나타납니다.

> **TIP**
> **데이터 원본** 창을 열려 있는지 확인 하려면 프로젝트가 열려 있는지 확인 한 다음 **Shift+Alt+D**를 누르거나**다른 Windows** > **데이터 원본 보기를** > 선택 합니다.

데이터 원본 창에서 양식 디자인 표면 또는 컨트롤로 데이터 원본을 드래그할 수 있습니다. 이렇게 하면 데이터 저장소의 데이터를 표시하는 상용구 코드가 생성됩니다.

다음 그림에서는 Windows 양식에 삭제된 데이터 집합을 보여 주습니다. 응용 프로그램에서 **F5**를 선택하면 기본 데이터베이스의 데이터가 양식의 컨트롤에 나타납니다.



데이터베이스 또는 데이터베이스 파일의 데이터 원본

데이터베이스 또는 데이터베이스 파일의 데이터 원본으로 사용할 데이터 집합 또는 엔터티 프레임워크 모델을 만들 수 있습니다.

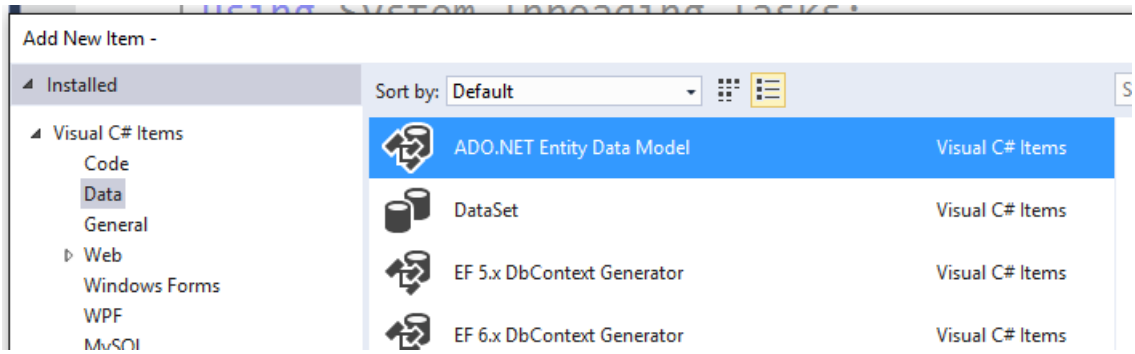
데이터 세트

데이터 집합을 데이터 원본으로 만들려면 **프로젝트** > 새 데이터**원본 추가**를선택하여 **데이터 원본 구성 마법사**를 실행합니다. **데이터베이스** 데이터 원본 유형을 선택하고 프롬프트에 따라 새 데이터베이스 연결 또는 기존 데이터베이스 연결 또는 데이터베이스 파일을 지정합니다.

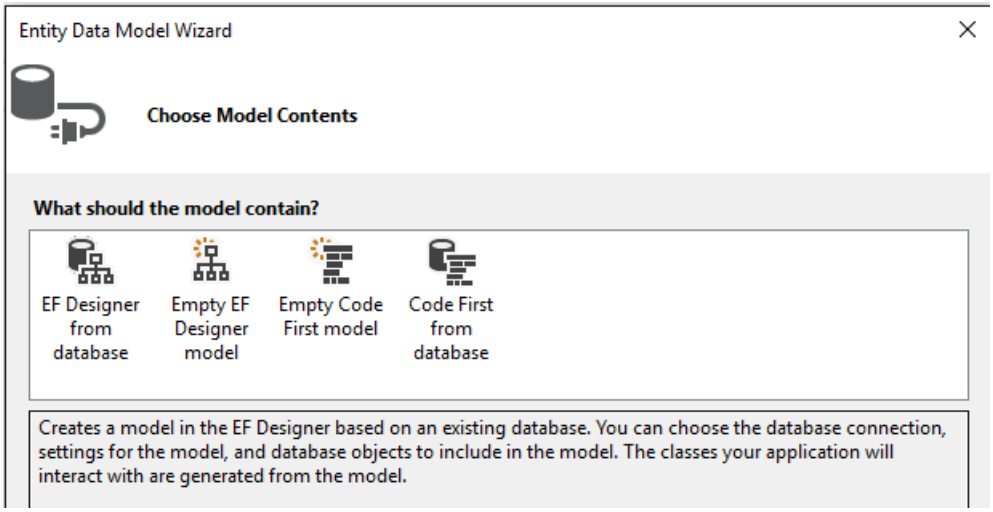
엔터티 클래스

엔터티 프레임워크 모델을 데이터 원본으로 만들려면 다음을 수행합니다.

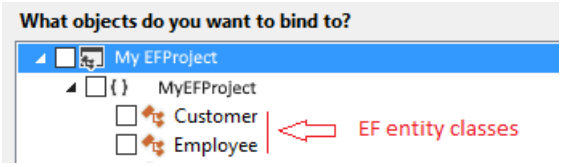
1. 엔터티 **데이터 모델 마법사**를 실행하여 엔터티 클래스를 만듭니다. **프로젝트** > **새 항목** > 추가**ADO.NET 엔터티 데이터 모델**선택 .



2. 모델을 생성할 방법을 선택합니다.



3. 모델을 데이터 원본으로 추가합니다. 생성된 클래스는 **개체** 범주를 선택할 때 **데이터 원본 구성 마법사**에 나타납니다.



서비스에 대한 데이터 원본

서비스에서 데이터 원본을 만들려면 **데이터 원본 구성 마법사**를 실행하고 **서비스** 데이터 원본 유형을 선택합니다. 이는 **서비스 참조 추가** 대화 상자의 바로 가기일 뿐이며, 솔루션 **탐색기에서** 프로젝트를 마우스 오른쪽 단추로 클릭하고 **서비스 참조 추가**를선택하여 액세스할 수도 있습니다.

서비스에서 데이터 원본을 만들 때 Visual Studio는 프로젝트에 서비스 참조를 추가합니다. 또한 Visual Studio는 서비스가 반환하는 개체에 해당하는 프록시 개체를 만듭니다. 예를 들어 데이터 집합을 반환하는 서비스는 프로젝트에 데이터 집합으로 표시됩니다. 특정 형식을 반환하는 서비스는 반환되는 형식으로 프로젝트에 표시됩니다.

다음 유형의 서비스에서 데이터 원본을 만들 수 있습니다.

- WCF Data Services
- WCF 서비스
- 웹 서비스

> **NOTE**
> **데이터 원본** 창에 나타나는 항목은 서비스가 반환하는 데이터에 따라 달라집니다. **데이터 원본 구성 마법사**에서 바인딩 가능한 개체를 만들기에 충분한 정보를 제공하지 않는 서비스도 있습니다. 예를 들어 서비스가 형식이 지정되지 않은 데이터 집합을 반환하는 경우 마법사를 완료할 때 **데이터 원본** 창에 항목이 나타나지 않습니다. 형식이 설정되지 않은 데이터 집합은 스키마를 제공하지 않으므로 마법사에 데이터 원본을 만들 수 있는 정보가 충분하지 않기 때문입니다.

개체의 데이터 원본

데이터 원본 **구성 마법사**를 실행한 다음 **개체** 데이터 원본 형식을 선택하여 하나 이상의 공용 속성을 노출하는 모든 개체에서 데이터 원본을 만들 수 있습니다. 개체의 모든 공용 속성은 **데이터 원본** 창에 표시됩니다. Entity Framework를 사용하고 모델을 생성한 경우 응용 프로그램의 데이터 원본인 엔터티 클래스를 찾을 수 있습니다.

데이터 **개체 선택** 페이지에서 트리 뷰의 노드를 확장하여 바인딩할 개체를 찾습니다. 트리 뷰에는 프로젝트 및 어셈블리 및 프로젝트에서 참조하는 기타 프로젝트에 대한 노드가 포함됩니다.

트리 뷰에 나타나지 않는 어셈블리 또는 프로젝트의 객체에 바인딩하려면 **참조 추가를** 클릭하고 **참조 추가 대화 상자를** 사용하여 어셈블리 또는 프로젝트에 대한 참조를 추가합니다. 참조를 추가하면 어셈블리 또는 프로젝트가 트리 뷰에 추가됩니다.

> **NOTE**
> 개체가 트리 뷰에 나타나기 전에 개체가 포함된 프로젝트를 빌드해야 할 수 있습니다.

> **NOTE**
> 끌어서 놓기 데이터 바인딩을 지원하려면 ITypedList 또는 IListSource 인터페이스를 구현하는 개체에 기본 생성자가 있어야 합니다. 그렇지 않으면 Visual Studio는 데이터 원본 개체를 인스턴스화할 수 없으며 항목을 디자인 표면으로 드래그할 때 오류가 표시됩니다.

공유점 목록의 데이터 원본

데이터 원본 **구성 마법사를** 실행하고 SharePoint 데이터 원본 유형을 선택하여 **SharePoint** 목록에서 데이터 원본을 만들 수 있습니다. SharePoint는 WCF 데이터 서비스를 통해 데이터를 노출하므로 SharePoint 데이터 원본을 만드는 것은 서비스에서 데이터 원본을 만드는 것과 동일합니다. **데이터 원본 구성 마법사에서 SharePoint** 항목을 선택하면 SharePoint 서버를 가리켜 SharePoint 데이터 서비스에 연결하는 **서비스 참조 추가** 대화 상자가 열립니다. 이를 위해서는 쉐어포인트 SDK가 필요합니다.

참고 항목

- .NET용 Visual Studio 데이터 도구

Visual Studio의 LINQ to SQL 도구

2020-01-06 • 15 minutes to read • Edit Online

LINQ to SQL은 Microsoft에서 릴리스된 최초의 개체 관계형 매핑 기술입니다. 이는 기본 시나리오에서 잘 작동 하며 Visual Studio에서 계속 지원 되지만 더 이상 개발 되지 않습니다. 이미 사용 하 고 있는 레거시 응용 프로그램을 유지 관리 하거나 SQL Server를 사용 하는 간단한 응용 프로그램에서 LINQ to SQL를 사용 하 고 다중 테이블 매핑이 필요 하지 않습니다. 일반적으로 개체 관계형 매퍼 계층이 필요한 경우 새 응용 프로그램은 Entity Framework를 사용 해야 합니다.

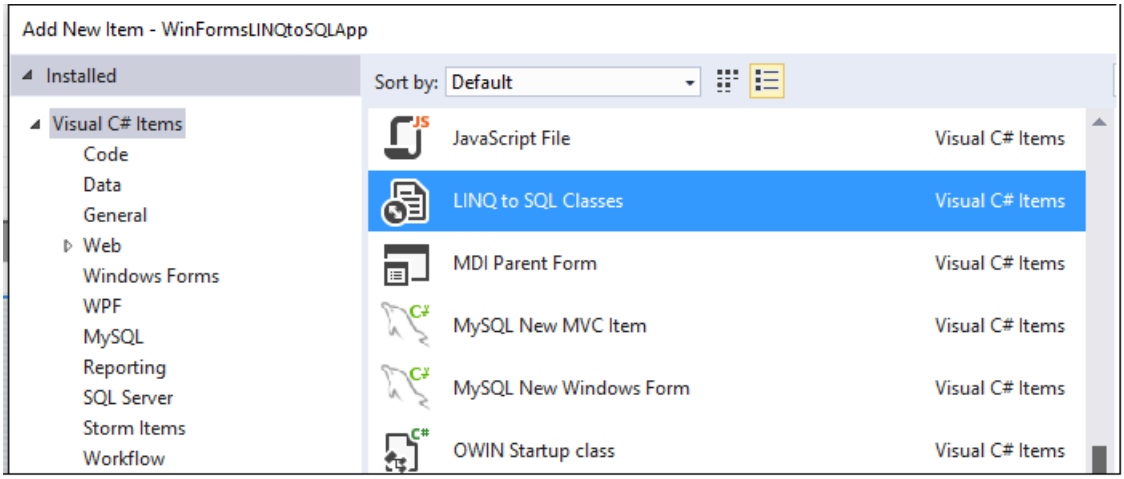
Visual Studio에서 **개체 관계형 디자이너** (**O/R 디자이너**)를 사용 하 여 SQL 테이블을 나타내는 LINQ to SQL 클래스를 만듭니다.

O/R 디자이너 의 디자인 화면에는 왼쪽에 있는 엔터티 창과 오른쪽에 있는 메서드 창이 있습니다. 엔터티 창은 엔터티 클래스, 연결 및 상속 계층을 표시하는 기본 디자인 화면입니다. 메서드 창은 저장 프로시저와 함수에 매핑되는 DataContext 메서드를 표시하는 디자인 화면입니다.

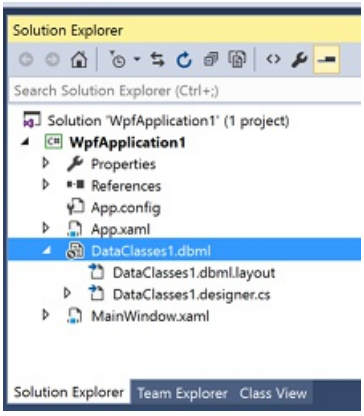
O/R 디자이너 는 데이터베이스의 개체를 기반으로 하는 LINQ to SQL 엔터티 클래스 및 연결 (관계)을 만들기 위한 시각적 디자인 화면을 제공 합니다. 즉, **O/R 디자이너** 는 데이터베이스의 개체에 매핑되는 응용 프로그램에 개체 모델을 만듭니다. 또한 엔터티 클래스와 데이터베이스 간에 데이터를 보내고 받는 강력한 형식의 DataContext를 생성 합니다. 또한 **O/R 디자이너** 는 데이터를 반환 하 고 엔터티 클래스를 채우기 위해 저장 프로시저 및 함수를 DataContext 메서드에 매핑하는 기능을 제공 합니다. 마지막으로 **O/R 디자이너** 는 엔터티 클래스 간에 상속 관계를 디자인 하는 기능을 제공 합니다.

O/R 디자이너 열기

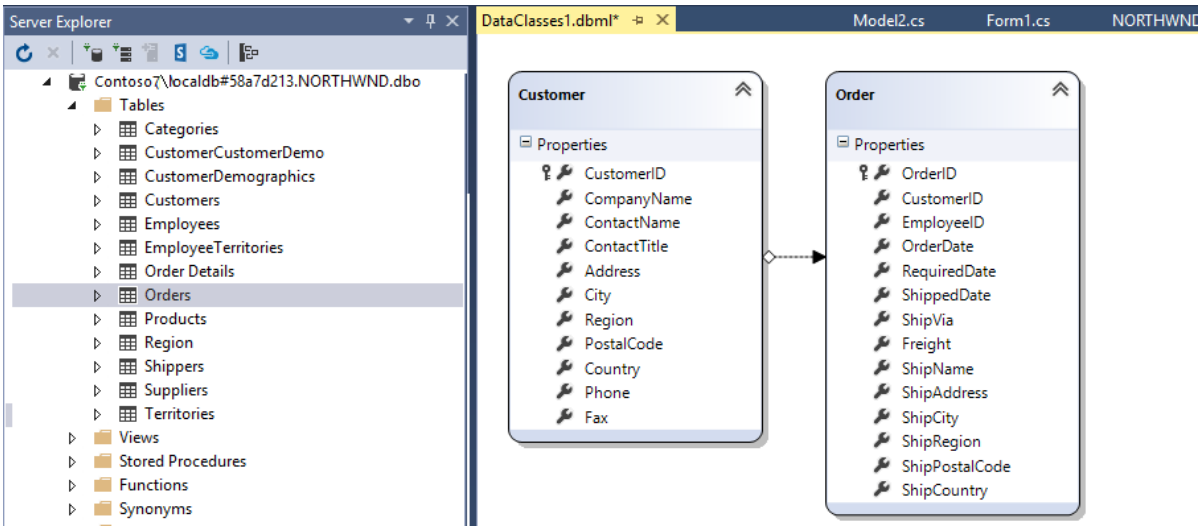
LINQ to SQL 엔터티 모델을 프로젝트에 추가 하려면 프로젝트 > **새 항목 추가 를 선택한 다음** 프로젝트 항목 목록에서 **LINQ to SQL 클래스** 를 선택 합니다.



Visual Studio는 *.dbml* 파일을 만들어 솔루션에 추가 합니다. XML 매핑 파일 및 관련 코드 파일입니다.



.Dbml 파일을 선택 하면 Visual Studio에서 모델을 시각적으로 만들 수 있도록 하는 **O/R 디자이너** 화면을 표시 합니다. 다음 그림에서는 Northwind `Customers` 및 `Orders` 테이블을 **서버 탐색기**에서 끌고 나면의 디자이너를 보여 줍니다. 테이블 간의 관계를 확인 합니다.



> **IMPORTANT**
> **O/R 디자이너** 는 1:1 매핑 관계만 지 원하는 단순 개체 관계형 매퍼입니다. 즉, 엔터티 클래스는 데이터베이스 테이블 또는 뷰와 1:1 매핑 관계만 갖습니다. 엔터티 클래스를 조인 된 테이블에 매핑하는 등의 복잡 한 매핑은 지원 되지 않습니다. 복합 매핑에 Entity Framework를 사용 합니다. 또한 이 디자이너는 단방향 코드 생성기입니다. 이는 디자이너 화면에서 변경한 내용만이 코드 파일에 반영된다는 의미입니다. 코드 파일을 수동으로 변경 해도 **O/R 디자이너**에는 반영 되지 않습니다. 코드 파일에서 수동으로 변경한 모든 내용은 디자이너를 저장하고 코드를 다시 생성할 때 덮어쓰여집니다. 사용자 코드를 추가 하 여 생성 된 클래스를 확장 하는 방법에 대 한 자세한 합니다 **O/R 디자이너**를 참조 하세요 방법: O/R 디자이너에서 생성한 코드 확장.

DataContext 만들기 및 구성

프로젝트에 **LINQ to SQL 클래스** 항목을 추가 하 고 **O/R 디자이너**를 연 후 빈 디자인 화면은 구성 가능한 빈 DataContext를 나타냅니다. 따라서 DataContext는 디자인 화면에 놓여진 첫째 항목에서 제공된 연결 정보를 사용하여 구성됩니다. 따라서 DataContext는 디자인 화면에 놓여진 첫째 항목의 연결 정보를 사용하여 구성됩니다. DataContext 클래스에 대 한 자세한 내용은 DataContext 메서드 (O/R 디자이너)를 참조 하세요.

데이터베이스 테이블 및 뷰에 매핑되는 엔터티 클래스 만들기

서버 탐색기 또는 **데이터베이스 탐색기** 의 데이터베이스 테이블 및 뷰를 **O/R 디자이너**로 끌어와 테이블 및 뷰에 매핑되는 엔터티 클래스를 만들 수 있습니다. 이전 섹션에서 설명한 것처럼 DataContext는 디자인 화면으로 끌어온 첫 번째 항목에서 제공된 연결 정보를 사용하여 구성됩니다. 다른 연결을 사용 하는 후속 항목이 **O/R 디자이너**에 추가 되 면 DataContext에 대 한 연결을 변경할 수 있습니다. 자세한 내용은 방법:

테이블 및 뷰에 매핑된 LINQ to SQL 클래스 만들기 (O/R 디자이너)를 참조 하세요.

저장 프로시저 및 함수를 호출 하는 DataContext 메서드 만들기

서버 탐색기 또는 **데이터베이스 탐색기** 를 **O/R 디자이너로**끌어 저장 프로시저 및 함수를 호출 하는 DataContext 메서드를 만들 수 있습니다. 저장 프로시저 및 함수는 **O/R 디자이너** 에 DataContext의 메서드로 추가 됩니다.

> **NOTE**
> **서버 탐색기** 또는 **데이터베이스 탐색기** 의 저장 프로시저 및 함수를 **O/R 디자이너로**끌어 오면 생성 된 DataContext 메서드의 반환 형식은 항목을 끌어 놓은 위치에 따라 달라 집니다. 자세한 내용은 DataContext 메서드 (O/R 디자이너)를 참조 하세요.

저장 프로시저를 사용 하 여 엔터티 클래스와 데이터베이스 간에 데이터를 저장 하도록 DataContext 구성

앞에서 설명한 대로 저장 프로시저 및 함수를 호출하는 DataContext 메서드를 만들 수 있습니다. 또한 기본 LINQ to SQL 런타임 동작에 사용 되는 저장 프로시저를 할당 하 여 삽입, 업데이트 및 삭제를 수행할 수 있습니다. 자세한 내용은 방법: 저장 프로시저를 할당 하 여 업데이트, 삽입 및 삭제 수행 (O/R 디자이너)을 참조 하세요.

상속 및 O/R 디자이너

다른 개체와 마찬가지로 LINQ to SQL 클래스는 상속을 사용할 수 있고 다른 클래스에서 파생 될 수 있습니다. 데이터베이스에서 상속 관계는 여러 가지 방법으로 만들어집니다. **O/R 디자이너** 는 관계형 시스템에서 주로 구현 되는 단일 테이블 상속 개념을 지원 합니다. 자세한 내용은 방법: O/R 디자이너를 사용 하 여 상속 구성을 참조 하세요.

LINQ to SQL 쿼리

O/R 디자이너 에서 만든 엔터티 클래스는 LINQ (통합 언어 쿼리)와 함께 사용 하도록 설계 되었습니다. 자세한 내용은 방법: 정보 쿼리를 참조 하세요.

생성 된 DataContext 및 엔터티 클래스 코드를 다른 네임 스페이스로 분리

O/R 디자이너 는 DataContext에 대 한 **컨텍스트 네임 스페이스** 및 **엔터티 네임 스페이스** 속성을 제공 합니다. 이들 속성은 DataContext 및 엔터티 클래스 코드가 어떤 네임스페이스로 생성되는지를 결정합니다. 기본적으로 이들 속성은 비어 있으며 DataContext 및 엔터티 클래스는 애플리케이션의 네임스페이스로 생성됩니다. 코드를 애플리케이션의 네임스페이스가 아닌 다른 네임스페이스로 생성하려면 **컨텍스트 네임스페이스** 및/또는 **엔터티 네임스페이스** 속성에 값을 입력합니다.

참조 콘텐츠

- System.Linq
- System.Data.Linq

참조

- LINQ to SQL (.NET Framework)
- Faq (질문과 대답) (.NET Framework)

방법: DataContext 메서드의 반환 형식 변경(O/R 디자이너)

2020-01-06 • 5 minutes to read • Edit Online

저장 프로시저 또는 함수를 기반으로 만들어진 DataContext 메서드의 반환 형식은 **O/R 디자이너**에서 저장 프로시저 또는 함수를 놓는 위치에 따라 달라 집니다. 저장 프로시저 또는 함수에서 반환된 데이터의 스키마가 엔터티 클래스의 모양과 일치하는 경우 항목을 기존 엔터티 클래스에 직접 드롭하면 엔터티 클래스의 반환 형식을 갖는 DataContext 메서드가 만들어집니다. 항목을 **O/R 디자이너**의 빈 영역에 놓으면 자동으로 생성 된 형식을 반환 하는 DataContext 메서드가 만들어집니다. 메서드 창에 추가한 후 DataContext 메서드의 반환 형식을 변경할 수 있습니다. DataContext 메서드의 반환 형식을 검사하거나 변경하려면 해당 메서드를 선택하고 **속성** 창에서 **반환 형식** 속성을 클릭합니다.

> **NOTE**
> 반환 형식을 엔터티 클래스로 설정한 DataContext 메서드를 **속성** 창을 사용하여 자동 생성된 형식을 반환하도록 되돌릴 수 없습니다. 자동으로 생성된 형식을 반환하도록 DataContext 메서드를 되돌리려면 원래 데이터베이스 개체를 **O/R 디자이너**로 끌어와야 합니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

DataContext 메서드의 반환 형식을 자동으로 생성된 형식에서 엔터티 클래스로 변경하려면

1. 메서드 창에서 DataContext 메서드를 선택합니다.
2. **속성** 창에서 **반환 형식**을 선택한 다음, **반환 형식** 목록에서 사용 가능한 엔터티 클래스를 선택합니다. 원하는 엔터티 클래스가 목록에 없으면 **O/R 디자이너** 에서 추가 하거나 만들어 목록에 추가 합니다.
3. *.dbml* 파일을 저장합니다.

DataContext 메서드의 반환 형식을 엔터티 클래스에서 자동으로 생성된 형식으로 다시 변경하려면

1. **메서드** 창에서 DataContext 메서드를 선택한 다음, 삭제합니다.
2. **서버 탐색기** 또는 **데이터베이스 탐색기** 의 데이터베이스 개체를 **O/R 디자이너**의 빈 영역으로 끕니다.
3. *.dbml* 파일을 저장합니다.

참조

- Visual Studio의 LINQ to SQL 도구
- LINQ to SQL
- DataContext 메서드(O/R 디자이너)
- 방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기(O/R 디자이너)

방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기(O/R 디자이너)

2020-01-06 • 7 minutes to read • Edit Online

DataContext 메서드로 **O/R 디자이너** 에 저장 프로시저 및 함수를 추가할 수 있습니다. 메서드를 호출하여 필요한 매개 변수에 전달하면 데이터베이스의 저장 프로시저 또는 함수가 실행되어 DataContext 메서드의 반환 형식으로 데이터를 반환합니다. DataContext 메서드에 대 한 자세한 내용은 DataContext 메서드 (O/R 디자이너)를 참조 하세요.

> **NOTE**
> 저장 프로시저를 사용 하 여 변경 내용이 엔터티 클래스에서 데이터베이스로 저장 될 때 삽입, 업데이트 및 삭제를 수행 하는 기본 LINQ to SQL 런타임 동작을 재정의할 수도 있습니다. 자세한 내용은 방법: 저장 프로시저를 할당 하 여 업데이트, 삽입 및 삭제 수행 (O/R 디자이너)을 참조 하세요.

DataContext 메서드 만들기

서버 탐색기 또는 * * 데이터베이스 탐색기 의 저장 프로시저 또는 함수를 **O/R Designer**로 끌어 DataContext 메서드를 만들 수 있습니다.

> **NOTE**
> 생성 된 DataContext 메서드의 반환 형식은 **O/R 디자이너**에서 저장 프로시저 또는 함수를 놓는 위치에 따라 달라 집니다. drop항목을 기존 엔터티 클래스에 직접 드롭하면 엔터티 클래스의 반환 형식을 갖는 DataContext 메서드가 만들어집니다. **O/R 디자이너** 의 빈 영역에 항목을 놓으면 자동으로 생성 된 형식을 반환 하는 DataContext 메서드가 만들어집니다. **Methods** 창에 추가한 후 DataContext 메서드의 반환 형식을 변경할 수 있습니다. DataContext 메서드의 반환 형식을 검사하거나 변경하려면 해당 메서드를 선택하고 **속성** 창에서 **반환 형식** 속성을 검사합니다. 자세한 내용은 방법: DataContext 메서드의 반환 형식 변경 (O/R 디자이너)을 참조 하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

자동으로 생성된 형식을 반환하는 **DataContext** 메서드를 만들려면

1. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 작업 하 고 있는 데이터베이스의 **저장 프로시저** 노드를 확장 합니다.
2. 원하는 저장 프로시저를 찾아 **O/R 디자이너**의 빈 영역으로 끕니다.

   자동으로 생성된 반환 형식을 갖는 DataContext 메서드가 만들어지고 **메서드** 창에 나타납니다.

엔터티 클래스의 반환 형식을 갖는 **DataContext** 메서드를 만들려면

1. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 작업 하 고 있는 데이터베이스의 **저장 프로시저** 노드를 확장 합니다.
2. 원하는 저장 프로시저를 찾아 **O/R 디자이너**의 기존 엔터티 클래스로 끌어 놓습니다.

   선택한 엔터티 클래스의 반환 형식을 갖는 DataContext 메서드가 만들어지고 **메서드** 창에 나타납니다.

NOTE

기존 반환 형식을 변경 하는 방법은 DataContext 메서드 참조 방법: DataContext 메서드의 반환 형식 변경(O/R 디자이너).

참조

- Visual Studio의 LINQ to SQL 도구
- DataContext 메서드(O/R 디자이너)
- 연습: LINQ to SQL 클래스 만들기
- LINQ to SQL
- Visual Basic의 LINQ 소개
- C#의 LINQ

방법: O/R 디자이너를 사용하여 상속 구성

2020-01-06 • 10 minutes to read • Edit Online

개체 관계형 디자이너 (**O/R 디자이너**)는 관계형 시스템에서 주로 구현 되는 단일 테이블 상속 개념을 지원 합니다. 단일 테이블 상속에서는 부모 정보와 자식 정보에 대한 필드를 모두 포함하는 데이터베이스 테이블이 하나 있습니다. 관계형 데이터의 경우 판별자 열에는 해당 레코드가 어느 클래스에 속해 있는지를 판별하는 값이 포함됩니다.

예를 들어 회사에서 사용 하는 모든 사용자를 포함 하는 `Persons` 테이블이 있다고 가정 합니다. 어떤 사람들은 사원이고 어떤 사람들은 관리자입니다. `Persons` 테이블에는 관리자의 값이 1이 고 직원의 값이 2 인 `EmployeeType` 라는 열이 포함 되어 있습니다. 판별자 열입니다. 이 시나리오에서는 직원 서브클래스를 만든 후 `EmployeeType` 값이 2인 레코드로만 클래스를 채울 수 있습니다. 각 클래스에서 적용되지 않는 열은 제거할 수도 있습니다.

상속을 사용하고 관계형 데이터에 대응하는 개체 모델을 만드는 것은 조금 복잡할 수 있습니다. 다음 절차에서는 **O/R 디자이너**를 사용하여 상속을 구성하는 데 필요한 단계를 요약한 것입니다. 기존 테이블 및 열을 참조 하지 않고 일반적인 단계를 수행 하는 것이 어려울 수 있으므로 데이터를 사용 하는 연습이 제공 됩니다. 사용 하 여 상속을 구성 하는 것에 대 한 자세한 단계별 지침은 **O/R 디자이너**를 참조 하세요 연습: 단일 테이블 상속 (O/R 디자이너)를 사용 하 여 만드는 LINQ to SQL 클래스합니다.

상속된 데이터 클래스를 만들려면

1. 기존 Visual Basic 또는 C# 프로젝트에 **LINQ to SQL 클래스** 항목을 추가 하 여 **O/R 디자이너** 를 엽니다.
2. 기본 클래스로 사용할 테이블을 **O/R 디자이너**에 끌어다 놓습니다.
3. 테이블의 두 번째 복사본을 **O/R 디자이너** 에 끌어 놓고 이름을 바꿉니다. 이것을 파생 클래스 또는 서브클래스라고 합니다.
4. **도구 상자**의 **개체 관계형 디자이너** 탭에서 **상속**을 클릭한 다음, 서브클래스(이름을 바꾼 테이블)를 클릭하고 이를 기본 클래스에 연결합니다.

   > **NOTE**
   > **도구 상자**에서 **상속** 항목을 클릭한 다음, 마우스 단추를 놓고 3단계에서 만든 클래스의 두 번째 복사본을 클릭한 다음, 2단계에서 만든 첫 번째 클래스를 클릭합니다. 상속 선의 화살표는 첫 번째 클래스를 가리킵니다.

5. 각 클래스에서 연결에 사용되지 않으므로 표시하지 않을 개체 속성을 모두 삭제합니다. 연결에 사용 되는 개체 속성을 삭제 하려고 하면 오류가 발생 합니다. 속성 <속성 이름 >은 (는) 연결 <연결 이름 >에 참여 하고 있으므로 삭제할 수 없습니다.

   > **NOTE**
   > 파생 클래스는 기본 클래스에 정의된 속성을 상속하므로 각 클래스에 동일한 열은 정의할 수 없습니다. 열은 속성으로 구현 됩니다. 기본 클래스의 속성에 상속 한정자를 설정 하 여 파생 클래스에서 열 만들기를 사용 하도록 설정할 수 있습니다. 자세한 내용은 상속 기본 사항 (Visual Basic)을 참조 하세요.

6. **O/R 디자이너**에서 상속 선을 선택 합니다.
7. **속성** 창에서 **판별자 속성** 을 클래스의 레코드를 구분 하는 열 이름으로 설정 합니다.
8. **Derived Class Discriminator Value** 속성을 해당 레코드를 상속된 형식으로 지정한 데이터베이스의 값

으로 설정합니다. 이 값은 판별자 열에 저장 되며 상속 된 클래스를 지정 하는 데 사용 됩니다.

9. **Base Class Discriminator Value** 속성을 해당 레코드를 기본 형식으로 지정한 값으로 설정합니다. 이 값은 판별자 열에 저장 되며 기본 클래스를 지정 하는 데 사용 됩니다.

10. 선택적으로 **Inheritance Default** 속성을 사용하여 정의된 상속 코드와 일치하지 않는 행을 로드할 때 사용되는 상속 계층 구조에서 형식을 지정할 수 있습니다. 즉, 레코드의 판별자 열에 **파생 클래스 판별자 값** 또는 **기본 클래스 판별자 값** 속성의 값과 일치 하지 않는 값이 있으면 레코드가 **상속 기본값으로**지정 된 형식으로 로드 됩니다.

참조

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)
- Visual Studio에서 데이터 액세스
- LINQ to SQL
- 연습: 단일 테이블 상속을 사용하여 LINQ to SQL 클래스 만들기(O/R 디자이너)
- 상속 기본 사항(Visual Basic)
- 상속

방법: 테이블 및 뷰에 매핑된 LINQ to SQL 클래스 만들기(O/R 디자이너)

2020-01-06 • 9 minutes to read • Edit Online

데이터베이스 테이블 및 뷰에 매핑되는 LINQ to SQL 클래스를 *엔터티 클래스*라고 합니다. 엔터티 클래스는 레코드에 매핑되고 엔터티 클래스의 개별 속성은 레코드를 구성 하는 개별 열에 매핑됩니다. **서버 탐색기** 또는 **데이터베이스 탐색기** 의 테이블이 나 뷰를 Visual Studio의 LINQ to SQL 도구로끌어 데이터베이스 테이블 또는 뷰를 기반으로 하는 엔터티 클래스를 만듭니다. **O/R 디자이너** 는 클래스를 생성 하 고 특정 LINQ to SQL 특성을 적용 하 여 LINQ to SQL 기능 (DataContext의 데이터 통신 및 편집 기능)을 사용할 수 있도록 합니다. LINQ to SQL 클래스에 대 한 자세한 내용은 LINQ to SQL 개체 모델을 참조 하세요.

> **NOTE**
> **O/R 디자이너** 는 1:1 매핑 관계만 지 원하는 단순 개체 관계형 매퍼입니다. 즉, 엔터티 클래스는 데이터베이스 테이블 또는 뷰와 1:1 매핑 관계만 갖습니다. 엔터티 클래스를 여러 테이블에 매핑하는 복잡한 매핑은 지원되지 않습니다. 그러나 엔터티 클래스를 여러 관련 테이블을 연결하는 뷰에 매핑할 수 있습니다.

데이터베이스 테이블 또는 보기에 매핑된 LINQ to SQL 클래스 만들기

서버 탐색기 또는 **데이터베이스 탐색기** 에서 **O/R Designer** 로 테이블 또는 뷰를 끌어 오면 업데이트를 수행 하는 데 사용 되는 DataContext 메서드 외에도 엔터티 클래스가 만들어집니다.

기본적으로 LINQ to SQL 런타임에서는 업데이트 가능한 엔터티 클래스의 변경 내용을 데이터베이스에 다시 저장하는 논리를 만듭니다. 이 논리는 열 정의 및 기본 키 정보와 같은 테이블 스키마를 기반으로 합니다. 이 동작을 원하지 않는 경우 기본 LINQ to SQL 런타임 동작을 사용 하는 대신 저장 프로시저를 사용 하 여 삽입, 업데이트 및 삭제를 수행 하도록 엔터티 클래스를 구성할 수 있습니다. 자세한 내용은 방법: 저장 프로시저를 할당 하 여 업데이트, 삽입 및 삭제 수행 (O/R 디자이너)을 참조 하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

데이터베이스 테이블 또는 뷰에 매핑된 **LINQ to SQL** 클래스를 만들려면

1. **서버** 또는 **데이터베이스 탐색기**에서 **테이블** 또는 **보기**를 확장하여 사용자 애플리케이션에 사용할 데이터베이스 테이블 또는 보기를 찾습니다.
2. 테이블이 나 뷰를 **O/R 디자이너**로끌어 옵니다.

   엔터티 클래스가 만들어져 디자인 화면에 표시됩니다. 엔터티 클래스에는 선택한 테이블 또는 뷰의 열에 매핑되는 속성이 있습니다.

개체 데이터 원본을 만들어 폼에 데이터 표시

O/R 디자이너를 사용 하 여 엔터티 클래스를 만든 후에는 개체 데이터 소스를 만들고 엔터티 클래스를 사용 하 여 데이터 소스 창을 채울 수 있습니다.

LINQ to SQL 엔터티 클래스 기반의 개체 데이터 소스를 만들려면

1. **빌드** 메뉴에서 **솔루션 빌드**를 클릭하여 프로젝트를 빌드합니다.

2. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.

3. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.

4. **데이터 원본 형식 선택** 페이지에서 **개체**를 클릭한 후, **다음**을 클릭합니다.

5. 노드를 확장하여 클래스를 찾아 선택합니다.

> **NOTE**
> **Customer** 클래스를 사용할 수 없는 경우에는 마법사를 취소하고, 프로젝트를 빌드하고, 마법사를 다시 실행합니다.

6. **마침**을 클릭하여 데이터 원본을 만들고 **데이터 원본** 창에 **Customer** 엔터티 클래스를 추가합니다.

7. 항목을 **데이터 원본** 창에서 폼으로 끌어 옵니다.

참조

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)
- DataContext 메서드(O/R 디자이너)
- 방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기(O/R 디자이너)
- LINQ to SQL 개체 모델
- 연습: 엔터티 클래스의 삽입, 업데이트 및 삭제 동작을 사용자 지정
- 방법: LINQ to SQL 클래스 사이에 연결(관계) 만들기(O/R 디자이너)

방법: O/R 디자이너에서 생성된 코드 확장

2020-01-16 • 5 minutes to read • Edit Online

O/R 디자이너 에서 생성 한 코드는 디자이너 화면에서 엔터티 클래스 및 기타 개체가 변경 될 때 다시 생성 됩니다. 이러한 코드의 다시 생성으로 인해 일반적으로 디자이너에서 코드를 다시 생성하면 생성된 코드에 추가한 모든 코드를 덮어씁니다. **O/R 디자이너** 는 덮어쓰지 않는 코드를 추가할 수 있는 partial 클래스 파일을 생성 하는 기능을 제공 합니다. **O/R 디자이너** 에서 생성 된 코드에 사용자 고유의 코드를 추가 하는 한 가지 예는 LINQ to SQL (엔터티) 클래스에 데이터 유효성 검사를 추가 하는 것입니다. 자세한 내용은 방법: 엔터티 클래스에 유효성 검사 추가를 참조 하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

엔터티 클래스에 코드 추가

Partial 클래스를 만들고 엔터티 클래스에 코드를 추가하려면

1. **O/R 디자이너**에서 새 LINQ to SQL 클래스 파일 (**.dbml** 파일)을 열거나 만듭니다. **솔루션 탐색기** 또는 **데이터베이스 탐색기**에서 **.dbml** 파일을 두 번 클릭 합니다.
2. **O/R 디자이너**에서 유효성 검사를 추가할 클래스를 마우스 오른쪽 단추로 클릭한 다음, **코드 보기**를 클릭 합니다.

   선택한 엔터티 클래스의 partial 클래스와 함께 코드 편집기가 열립니다.
3. 엔터티 클래스의 partial 클래스 선언에 사용자 코드를 추가합니다.

DataContext에 코드 추가

Partial 클래스를 만들고 **DataContext**에 코드를 추가하려면

1. **O/R 디자이너**에서 새 LINQ to SQL 클래스 파일 (**.dbml** 파일)을 열거나 만듭니다. **솔루션 탐색기** 또는 **데이터베이스 탐색기**에서 **.dbml** 파일을 두 번 클릭 합니다.
2. **O/R 디자이너**의 디자이너에서 빈 영역을 마우스 오른쪽 단추로 클릭 한 다음 **코드 보기**를 클릭 합니다.

   DataContext의 partial 클래스와 함께 코드 편집기가 열립니다.
3. DataContext의 partial 클래스 선언에 사용자 코드를 추가합니다.

참조

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)
- LINQ to SQL

방법: LINQ to SQL 클래스 간의 연결 만들기 (O/R 디자이너)

2020-01-06 • 6 minutes to read • Edit Online

LINQ to SQL에서 엔터티 클래스 간의 연결은 데이터베이스 테이블 간의 관계와 비슷합니다. **연결 편집기** 대화 상자를 사용하여 엔터티 클래스 간의 연결을 만들 수 있습니다.

연결 편집기 대화 상자에서 연결을 만들 때 부모 클래스와 자식 클래스를 선택해야 합니다. 부모 클래스는 기본 키가 있는 엔터티 클래스이고, 자식 클래스는 외래 키가 있는 엔터티 클래스입니다. 예를 들어 `Northwind Customers` 및 `Orders` 테이블에 매핑되는 엔터티 클래스를 만든 경우 `Customer` 클래스는 부모 클래스가 되 고 `Order` 클래스는 자식 클래스가 됩니다.

> **NOTE**
> **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 **개체 관계형 디자이너** (O/R 디자이너)로 테이블을 끌어 오면 데이터베이스의 기존 외래 키 관계를 기반으로 연결이 자동으로 만들어집니다.

연결 속성

연결을 만든 후 **O/R 디자이너**에서 연결을 선택하면 **속성** 창에 몇 개의 구성 가능한 속성이 나타납니다. 연결은 관련 클래스 간의 선입니다. 다음 표에서는 연결의 속성에 대 한 설명을 제공 합니다.

속성	설명
Cardinality	연결이 일대다 연결인지 또는 일대일 연결인지를 제어합니다.
자식 속성	연결의 외래 키 쪽에서 자식 레코드의 컬렉션이거나 자식 레코드에 대한 참조인 부모 속성을 만들지 여부를 지정합니다. 예를 들어 `Customer` 와 `Order` 간의 연결에서 **자식 속성이 True**로 설정 된 경우 `Orders` 라는 속성이 부모 클래스에 만들어집니다.
부모 속성	연결된 부모 클래스를 참조하는 자식 클래스의 속성입니다. 예를 들어 `Customer` 와 `Order` 간의 연결에서 주문에 대 한 관련 고객을 참조 하는 `Customer` 라는 속성이 `Order` 클래스에 만들어집니다.
참여 속성	연결 속성을 표시하고 **연결 편집기** 대화 상자를 다시 여는 **줄임표** 단추(...)를 제공합니다.
Unique	외래 대상 열에 고유성 제약 조건이 있는지 여부를 지정합니다.

엔터티 클래스 간에 연결을 만들려면

1. 연결에서 부모 클래스를 나타내는 엔터티 클래스를 마우스 오른쪽 단추로 클릭하고 **추가**를 가리킨 다음, **연결**을 클릭합니다.
2. **연결 편집기** 대화 상자에서 올바른 **부모 클래스**가 선택되었는지 확인합니다.

3. 콤보 상자에서 **자식 클래스**를 선택합니다.

4. 클래스가 관련된 **연결 속성**을 선택합니다. 일반적으로 이는 데이터베이스에 정의된 외래 키 관계에 매핑됩니다. 예를 들어 `Customers` 및 `Orders` 연결에서 **연결 속성** 은 각 클래스에 대 한 `CustomerID`입니다.

5. **확인**을 클릭하여 연결을 만듭니다.

참조

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기
- LINQ to SQL
- DataContext 메서드(O/R 디자이너)
- 방법: 기본 키 표현

방법: 엔터티 클래스에 유효성 검사 추가

2020-01-06 • 12 minutes to read • Edit Online

엔터티 클래스의 *유효성 검사*는 데이터 개체에 입력된 값이 개체의 스키마 제약 조건과 애플리케이션에 대해 설정된 규칙을 따르는지 확인하는 과정입니다. 업데이트를 내부 데이터베이스에 보내기 전에 데이터의 유효성을 검사하면 오류를 줄일 수 있습니다. 또한 애플리케이션과 데이터베이스 간에 발생할 수 있는 잠재적 라운드트립 횟수를 줄일 수 있습니다.

Visual Studio의 LINQ to SQL 도구 는 사용자가 전체 엔터티를 삽입, 업데이트 및 삭제 하는 동안 실행 되는 디자이너 생성 코드와 개별 열이 변경 되는 동안 및 그 후에 실행 되는 디자이너 생성 코드를 확장할 수 있는 부분 메서드를 제공 합니다.

> **NOTE**
> 이 항목에서는 **O/R 디자이너**를 사용 하 여 엔터티 클래스에 유효성 검사를 추가 하는 기본 단계를 제공 합니다. 특정 엔터티 클래스를 참조 하지 않고 이러한 일반 단계를 수행 하기 어려울 수 있기 때문에 실제 데이터를 사용 하는 연습이 제공 됩니다.

특정 열의 값에 대 한 변경 내용에 대 한 유효성 검사 추가

이 절차는 열의 값이 변경될 때 데이터의 유효성을 검사하는 방법을 보여 줍니다. 유효성 검사는 사용자 인터페이스 대신 클래스 정의 내에서 수행되기 때문에 값에 대한 유효성 검사가 실패하면 예외가 throw됩니다. 애플리케이션에서 열 값을 변경하려고 하는 코드에 대한 오류 처리를 구현하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

열의 값을 변경하는 동안 데이터의 유효성을 검사하려면

1. **O/R 디자이너**에서 새 LINQ to SQL 클래스 파일 (**.dbml** 파일)을 열거나 만듭니다. (**솔루션 탐색기**에서 **.dbml** 파일을 두 번 클릭합니다.)
2. **O/R 디자이너**에서 유효성 검사를 추가할 클래스를 마우스 오른쪽 단추로 클릭한 다음, **코드 보기**를 클릭합니다.

   선택한 엔터티 클래스의 partial 클래스와 함께 코드 편집기가 열립니다.
3. 커서를 partial 클래스에 놓습니다.
4. Visual Basic 프로젝트의 경우
   a. **메서드 이름** 목록을 확장합니다.
   b. 유효성 검사를 추가할 열에 대해 **OnCOLUMNNAMEChanging** 메서드를 찾습니다.
   c. `OnCOLUMNNAMEChanging` 메서드가 partial 클래스에 추가됩니다.
   d. 다음 코드를 추가하여 입력된 값을 먼저 확인한 다음 열에 입력된 값이 애플리케이션에서 허용되는지 확인합니다. 다음 `value` 인수는 제안된 값을 포함하고 있으므로 이를 유효한 값으로 확인하는 논리를 추가합니다.

```
If value.HasValue Then
    ' Add code to ensure that the value is acceptable.
    ' If value < 1 Then
    '    Throw New Exception("Invalid data!")
    ' End If
End If
```

C# 프로젝트의 경우:

프로젝트 C# 는 이벤트 처리기를 자동으로 생성 하지 않으므로 IntelliSense를 사용 하 여 열 변경 부분 메서드를 만들 수 있습니다. `partial` 을 입력한 다음 사용 가능한 부분 메서드에 액세스하기 위한 공간을 입력합니다. 유효성 검사를 추가할 열에 대해 열 변경 메서드를 클릭합니다. 다음 코드는 열 변경 부분 메서드를 선택할 때 생성 되는 코드와 비슷합니다.

```
partial void OnCOLUMNNAMEChanging(COLUMNDATATYPE value)
    {
       throw new System.NotImplementedException();
    }
```

엔터티 클래스의 업데이트에 대 한 유효성 검사 추가

변경하는 동안 값을 검사하는 작업 이외에도 전체 엔터티 클래스에 대해 업데이트를 시도할 때 데이터의 유효성을 검사할 수도 있습니다. 비즈니스 규칙에서 여러 열의 값에 대한 비교를 요구하는 경우 업데이트를 시도하는 동안 유효성 검사가 해당 작업을 수행합니다. 다음 절차는 전체 엔터티 클래스를 업데이트하려고 할 때 유효성을 검사하는 방법을 보여 줍니다.

> **NOTE**
> 전체 엔터티 클래스 업데이트에 대한 유효성 검사 코드는 특정 엔터티 클래스의 partial 클래스가 아니라 partial DataContext 클래스에서 실행됩니다.

엔터티 클래스에 대한 업데이트 동안 데이터의 유효성을 검사하려면

1. **O/R 디자이너**에서 새 LINQ to SQL 클래스 파일 (**.dbml** 파일)을 열거나 만듭니다. (**솔루션 탐색기**에서 **.dbml** 파일을 두 번 클릭합니다.)
2. **O/R 디자이너**에서 마우스 오른쪽 단추로 빈 영역을 클릭하고 **코드 보기**를 클릭합니다.

   `DataContext` 의 partial 클래스와 함께 코드 편집기가 열립니다.
3. 커서를 `DataContext` 의 partial 클래스에 놓습니다.
4. Visual Basic 프로젝트의 경우
   a. **메서드 이름** 목록을 확장합니다.
   b. **UpdateENTITYCLASSNAME**를 클릭합니다.
   c. `UpdateENTITYCLASSNAME` 메서드가 partial 클래스에 추가됩니다.
   d. 다음 코드에 표시된 것과 같은 `instance` 인수를 사용하여 개별 열 값에 액세스합니다.

```
If (instance.COLUMNNAME = x) And (instance.COLUMNNAME = y) Then
    Dim ErrorMessage As String = "Invalid data!"
    Throw New Exception(ErrorMessage)
End If
```

C# 프로젝트의 경우:

프로젝트 C# 는 이벤트 처리기를 자동으로 생성 하지 않으므로 IntelliSense를 사용 하 여 부분 `UpdateCLASSNAME` 메서드를 만들 수 있습니다. `partial`을 입력한 다음 사용 가능한 부분 메서드에 액세스하기 위한 공간을 입력합니다. 유효성 검사를 추가 하려는 클래스의 업데이트 메서드를 클릭 합니다. 다음 코드는 `UpdateCLASSNAME` 부분 메서드를 선택할 때 생성 되는 코드와 비슷합니다.

```
partial void UpdateCLASSNAME(CLASSNAME instance)
{
    if ((instance.COLUMNNAME == x) && (instance.COLUMNNAME = y))
    {
        string ErrorMessage = "Invalid data!";
        throw new System.Exception(ErrorMessage);
    }
}
```

참조

- Visual Studio의 LINQ to SQL 도구
- 데이터 유효성 검사
- LINQ to SQL (.NET Framework)

연습: 엔터티 클래스의 삽입, 업데이트 및 삭제 동작 사용자 지정

2020-01-06 • 29 minutes to read • Edit Online

Visual Studio의 LINQ to SQL 도구 는 데이터베이스의 개체를 기반으로 하는 LINQ to SQL 클래스 (엔터티 클래스)를 만들고 편집 하는 시각적 디자인 화면을 제공 합니다. LINQ to SQL를 사용 하 여 LINQ 기술을 사용 하 여 SQL database에 액세스할 수 있습니다. 자세한 내용은 LINQ(Language-Integrated Query)를 참조하세요.

기본적으로 업데이트를 수행 하는 논리는 LINQ to SQL 런타임에서 제공 됩니다. 런타임은 테이블의 스키마 (열 정의 및 기본 키 정보)를 기반으로 기본 `Insert`, `Update` 및 `Delete` 문을 만듭니다. 기본 동작을 사용하지 않으려면 업데이트 동작을 구성하고 데이터베이스의 데이터 작업에 필요한 삽입, 업데이트 및 삭제를 수행하기 위한 특정 저장 프로시저를 지정할 수 있습니다. 엔터티 클래스가 뷰에 매핑되는 때와 같이 기본 동작이 생성되지 않은 경우에도 이렇게 할 수 있습니다. 또한 저장 프로시저를 통해 데이터베이스의 테이블에 액세스해야 하는 경우에 기본 업데이트 동작을 재정의할 수 있습니다. 자세한 내용은 저장 프로시저를 사용 하 여 작업 사용자 지정을 참조 하세요.

> **NOTE**
> 이 연습을 수행하려면 Northwind 데이터베이스의 **InsertCustomer**, **UpdateCustomer** 및 **DeleteCustomer** 저장 프로시저를 사용할 수 있어야 합니다.

이 연습에서는 저장 프로시저를 사용 하 여 데이터를 데이터베이스에 다시 저장 하는 기본 LINQ to SQL 런타임 동작을 재정의 하기 위해 수행 해야 하는 단계를 제공 합니다.

이 연습에서는 다음 작업을 수행 하는 방법에 대해 알아봅니다.

- 새 Windows Forms 응용 프로그램을 만들고이 응용 프로그램에 LINQ to SQL 파일을 추가 합니다.
- Northwind `Customers` 테이블에 매핑되는 엔터티 클래스를 만듭니다.
- LINQ to SQL `Customer` 클래스를 참조 하는 개체 데이터 소스를 만듭니다.
- `Customer` 클래스에 바인딩되는 DataGridView을 포함 하는 Windows Form을 만듭니다.
- 폼의 저장 기능을 구현합니다.
- **O/R 디자이너**에 저장 프로시저를 추가 하 여 DataContext 메서드를 만듭니다.
- 저장 프로시저를 사용 하 여 삽입, 업데이트 및 삭제를 수행 하도록 `Customer` 클래스를 구성 합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. (**SQL Server 개체 탐색기** 은 **Visual Studio 설치 관리자**의 **데이터 저장 및 처리** 워크 로드의 일부로 설치 됩니다.) **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

쿼리 편집기 창이 열립니다.

b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

애플리케이션 만들기 및 LINQ to SQL 클래스 추가

LINQ to SQL 클래스로 작업 하 고 Windows Form에 데이터를 표시 하기 때문에 새 Windows Forms 응용 프로그램을 만들고 LINQ to SQL 클래스 파일을 추가 합니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

LINQ to SQL 클래스를 포함 하는 새 **Windows Forms** 응용 프로그램 프로젝트를 만들려면

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **UpdatingWithSProcsWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   UpdatingWithSProcsWalkthrough 프로젝트를 만들어 **솔루션 탐색기**에 추가합니다.
5. **프로젝트** 메뉴에서 **새 항목 추가**를 클릭합니다.
6. **LINQ to SQL 클래스** 템플릿을 클릭하고 **이름** 상자에 **Northwind.dbml**을 입력합니다.
7. **추가**를 클릭합니다.

   빈 LINQ to SQL 클래스 파일 (**Northwind .dbml**)이 프로젝트에 추가 되 고 **O/R 디자이너가** 열립니다.

Customer 엔터티 클래스 및 개체 데이터 원본 만들기

서버 탐색기 또는 **데이터베이스 탐색기** 에서 **O/R 디자이너로**테이블을 끌어 데이터베이스 테이블에 매핑되는 LINQ to SQL 클래스를 만듭니다. 그러면 데이터베이스의 테이블에 매핑되는 LINQ to SQL 엔터티 클래스가 생성됩니다. 엔터티 클래스를 만든 후 이 클래스를 공용 속성이 있는 다른 클래스처럼 개체 데이터 소스로 사용할 수 있습니다.

Customer 엔터티 클래스를 만들고 이 클래스를 사용하여 데이터 소스를 구성하려면

1. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 Northwind 샘플 데이터베이스의 SQL Server 버전에 있는 **Customer** 테이블을 찾습니다.
2. **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 **O/R 디자이너* 화면으로 **Customers** 노드를 끕니다.

   Customer라는 엔터티 클래스를 만듭니다. 이 클래스에는 Customers 테이블의 열에 해당하는 속성이 있습니다. 이 엔터티 클래스는 Customers 테이블의 단일 고객을 나타내므로 이름이 **Customers**가 아닌 **Customer**로 지정됩니다.

> **NOTE**
> 이러한 이름 바꾸기 동작을 *복수 적용*이라고 합니다. 옵션 대화 상자에서 설정 하거나 해제할 수 있습니다. 자세한 내용은 방법: 복수화 설정 및 해제 (O/R 디자이너)를 참조 하세요.

3. **빌드** 메뉴에서 **UpdatingwithSProcsWalkthrough 빌드**를 클릭하여 프로젝트를 빌드합니다.
4. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.
5. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.
6. **데이터 원본 형식 선택** 페이지에서 **개체**를 클릭한 후, **다음**을 클릭합니다.
7. **UpdatingwithSProcsWalkthrough** 노드를 확장하고 **Customer** 클래스를 찾아 선택합니다.

> **NOTE**
> **Customer** 클래스를 사용할 수 없는 경우에는 마법사를 취소하고, 프로젝트를 빌드하고, 마법사를 다시 실행합니다.

8. **마침**을 클릭하여 데이터 원본을 만들고 **데이터 원본** 창에 **Customer** 엔터티 클래스를 추가합니다.

Windows Form에 고객 데이터를 표시 하는 DataGridView 만들기

데이터 소스 창에서 Windows Form으로 LINQ to SQL 데이터 소스 항목을 끌어와 서 엔터티 클래스에 바인딩된 컨트롤을 만듭니다.

엔터티 클래스에 바인딩된 컨트롤을 추가하려면

1. 디자인 뷰에서 **Form1**을 엽니다.
2. **데이터 원본** 창에서 **Customer** 노드를 **Form1**으로 끌어 옵니다.

> **NOTE**
> **데이터 원본** 창을 표시하려면 **데이터** 메뉴에서 **데이터 원본 표시**를 클릭합니다.

3. 코드 편집기에서 **Form1**을 엽니다.
4. 폼에 다음 코드를 폼에 추가 하 고, 특정 메서드 외부에서 `Form1` 클래스 내에 추가 합니다.

```
Private NorthwindDataContext1 As New NorthwindDataContext
```

```
private NorthwindDataContext northwindDataContext1
    = new NorthwindDataContext();
```

5. `Form_Load` 이벤트에 대한 이벤트 처리기를 만들고 다음 코드를 처리기에 추가합니다.

```
CustomerBindingSource.DataSource = NorthwindDataContext1.Customers
```

```
customerBindingSource.DataSource
    = northwindDataContext1.Customers;
```

저장 기능 구현

기본적으로 저장 단추를 사용할 수 없으며 저장 기능이 구현되지 않습니다. 또한 개체 데이터 소스에 대해 데이터 바인딩된 컨트롤을 만들 때 변경된 데이터를 데이터베이스에 저장하는 코드가 자동으로 추가되지 않습니다. 이 섹션에서는 저장 단추를 사용 하도록 설정 하 고 LINQ to SQL 개체에 대 한 저장 기능을 구현 하는 방법을 설명 합니다.

저장 기능을 구현하려면

1. 디자인 뷰에서 **Form1**을 엽니다.
2. **CustomerBindingNavigator**에서 저장 단추(플로피 디스크 아이콘이 있는 단추)를 선택합니다.
3. **속성** 창에서 **Enabled** 속성을 **True**로 설정합니다.
4. 저장 단추를 두 번 클릭하여 이벤트 처리기를 만들고 코드 편집기로 전환합니다.
5. 저장 단추 이벤트 처리기에 다음 코드를 추가합니다.

```
NorthwindDataContext1.SubmitChanges()
```

```
northwindDataContext1.SubmitChanges();
```

업데이트를 수행 하기 위한 기본 동작 (삽입, 업데이트 및 삭제) 재정의

기본 업데이트 동작을 재정의하려면

1. **O/R 디자이너**에서 LINQ to SQL 파일을 엽니다. (**솔루션 탐색기**에서 **Northwind.dbml** 파일을 두 번 클릭합니다.)
2. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 Northwind 데이터베이스 **저장 프로시저** 노드를 확장하고 **InsertCustomers**, **UpdateCustomers** 및 **DeleteCustomers** 저장 프로시저를 찾습니다.
3. 세 개의 저장 프로시저를 모두 **O/R 디자이너로**끌어 옵니다.

   이러한 저장 프로시저가 메서드 창에 DataContext 메서드로 추가됩니다. 자세한 내용은 DataContext 메서드 (O/R 디자이너)를 참조 하세요.
4. **O/R 디자이너**에서 **Customer** 엔터티 클래스를 선택 합니다.
5. **속성** 창에서 **삽입** 속성을 선택합니다.
6. **런타임 사용** 옆의 줄임표(...) 를 클릭하여 **동작 구성** 대화 상자를 엽니다.
7. **사용자 지정**을 선택합니다.
8. **사용자 지정** 목록에서 **InsertCustomers** 메서드를 선택합니다.
9. **적용**을 클릭하여 선택한 클래스 및 동작에 대한 구성을 저장합니다.

   > **NOTE**
   > 계속해서 클래스/동작 조합을 변경한 후 **적용**을 클릭하여 해당하는 각 조합에 대한 동작을 구성할 수 있습니다. **적용**을 클릭 하기 전에 클래스 또는 동작을 변경한 경우 변경 내용을 적용 하는 기회를 제공 하는 경고 대화 상자가 나타납니다.
10. **동작** 목록에서 **업데이트**를 선택합니다.
11. **사용자 지정**을 선택합니다.

12. **사용자 지정** 목록에서 **UpdateCustomers** 메서드를 선택합니다.

    메서드 인수 및 **클래스 속성** 목록을 살펴보면 테이블의 일부 열에 두 개의 **메서드 인수**와 두 개의 **클래스 속성**이 있습니다. 이를 사용하면 손쉽게 변경 내용을 추적하고 동시성 위반을 확인하는 문을 만들 수 있습니다.

13. **Original_CustomerID** 메서드 인수를 **CustomerID(Original)** 클래스 속성에 매핑합니다.

    > **NOTE**
    > 기본적으로 메서드 인수는 이름이 일치하는 경우 클래스 속성에 매핑됩니다. 속성 이름이 변경되거나 더 이상 테이블 및 엔터티 클래스 간에 일치하지 않아 **O/R 디자이너**에서 올바른 매핑을 결정할 수 없는 경우에는 매핑할 해당 클래스 속성을 선택해야 합니다. 또한 메서드 인수를 매핑할 올바른 클래스 속성이 없는 경우 **클래스 속성** 값을 **(없음)**으로 설정할 수 있습니다.

14. **적용**을 클릭하여 선택한 클래스 및 동작에 대한 구성을 저장합니다.

15. **목록** 목록에서 **삭제**를 선택합니다.

16. **사용자 지정**을 선택합니다.

17. **사용자 지정** 목록에서 **DeleteCustomers** 메서드를 선택합니다.

18. **Original_CustomerID** 메서드 인수를 **CustomerID(Original)** 클래스 속성에 매핑합니다.

19. **확인**을 클릭합니다.

> **NOTE**
> 이 특정 연습에서는 문제가 되지 않지만 삽입 중에 id (자동 증분), rowguidcol (데이터베이스에서 생성 된 GUID) 및 timestamp 열에 대해 데이터베이스에서 생성 된 값을 자동으로 처리 하는 것이 LINQ to SQL. update. 데이터베이스에서 생성된 값이 다른 형식의 열에 있으면 null 값이라는 예기치 않은 결과가 발생합니다. 데이터베이스에서 생성 된 값을 반환 하려면 IsDbGenerated를 `true`으로 수동으로 설정 하 고 AutoSync, AutoSync, OnInsert또는 AutoSync중 하나로 AutoSync 해야 합니다.

응용 프로그램 테스트

애플리케이션을 다시 실행하여 **UpdateCustomers** 저장 프로시저가 데이터베이스에서 고객 레코드를 올바로 업데이트하는지 확인합니다.

1. **F5**키를 누릅니다.
2. 그리드에서 레코드를 수정하여 업데이트 동작을 테스트합니다.
3. 새 레코드를 추가하여 삽입 동작을 테스트합니다.
4. 저장 단추를 클릭하여 변경 내용을 데이터베이스에 다시 저장합니다.
5. 폼을 닫아 디버깅을
6. **F5** 키를 눌러 업데이트된 레코드와 새로 삽입한 레코드가 있는지 확인합니다.
7. 3단계에서 만든 새 레코드를 삭제하여 삭제 동작을 테스트합니다.
8. 저장 단추를 클릭하여 변경 내용을 전송하고 데이터베이스에서 삭제한 레코드를 제거합니다.
9. 폼을 닫아 디버깅을
10. **F5** 키를 눌러 삭제한 레코드가 데이터베이스에서 제거되었는지 확인합니다.

> **NOTE**
> 애플리케이션에서 SQL Server Express 버전을 사용하는 경우 데이터베이스 파일의 **출력 디렉터리로 복사** 속성 값에 따라 10단계에서 **F5** 키를 누를 때 변경 내용이 표시되지 않을 수 있습니다.

다음 단계

응용 프로그램 요구 사항에 따라 LINQ to SQL 엔터티 클래스를 만든 후 몇 단계를 더 수행 해야 할 수도 있습니다. 이 애플리케이션에서 개선할 수 있는 몇 가지 사항은 다음과 같습니다.

- 업데이트 동안 동시성 검사를 구현합니다. 자세한 내용은 낙관적 동시성: 개요를 참조 하세요.
- LINQ 쿼리를 추가하여 데이터를 필터링합니다. 자세한 내용은 LINQ 쿼리 소개C#()를 참조 하세요.

참조

- Visual Studio의 LINQ to SQL 도구
- DataContext 메서드
- 방법: 저장 프로시저를 할당 하 여 업데이트, 삽입 및 삭제 수행
- LINQ to SQL
- LINQ to SQL 쿼리

방법: 저장 프로시저를 할당하여 업데이트, 삽입 및 삭제 수행(O/R 디자이너)

2020-01-06 • 11 minutes to read • Edit Online

저장 프로시저를 **O/R 디자이너**에 추가하여 일반적인 DataContext 메서드로 실행할 수 있습니다. 또한 엔터티 클래스의 변경 내용이 데이터베이스에 저장 된 경우 (예: SubmitChanges 메서드를 호출 하는 경우) 삽입, 업데이트 및 삭제를 수행 하는 기본 LINQ to SQL 런타임 동작을 재정의 하는 데 사용할 수 있습니다.

> **NOTE**
> 클라이언트로 다시 보내야 하는 값(예: 저장 프로시저에서 계산된 값)을 저장 프로시저에서 반환하는 경우 저장 프로시저에 출력 매개 변수를 만듭니다. 출력 매개 변수를 사용할 수 없는 경우 O/R 디자이너에서 생성된 재정의를 사용하지 말고 부분 메서드(Partial Method) 구현을 작성합니다. 데이터베이스에서 생성된 값에 매핑되는 멤버는 INSERT 또는 UPDATE 작업이 성공적으로 완료된 후 적절한 값으로 설정되어야 합니다. 자세한 내용은 기본 동작 재정의에서 개발자의 책임을 참조 하세요.

> **NOTE**
> LINQ to SQL은 id (자동 증분), rowguidcol (데이터베이스에서 생성 된 GUID) 및 timestamp 열에 대해 데이터베이스에서 생성 된 값을 자동으로 처리 합니다. 데이터베이스에서 생성된 값이 다른 형식의 열에 있으면 null 값이라는 예기치 않은 결과가 발생합니다. 데이터베이스에서 생성 된 값을 반환 하려면 IsDbGenerated를 **true** 로 수동으로 설정 하 고 AutoSync, AutoSync, OnInsert또는 AutoSync중 하나로 AutoSync 해야 합니다.

엔터티 클래스의 업데이트 동작 구성

기본적으로 LINQ to SQL 엔터티 클래스의 데이터에 적용 된 변경 내용으로 데이터베이스를 업데이트 (삽입, 업데이트 및 삭제) 하는 논리는 LINQ to SQL 런타임에서 제공 됩니다. 런타임에서는 열 및 기본 키 정보와 같은 테이블 스키마를 기반으로 기본 INSERT, UPDATE 및 DELETE 명령을 만듭니다. 기본 동작을 원하지 않는 경우 테이블의 데이터를 조작 하는 데 필요한 삽입, 업데이트 및 삭제를 수행 하기 위한 특정 저장 프로시저를 할당 하 여 업데이트 동작을 구성할 수 있습니다. 엔터티 클래스가 뷰에 매핑되는 때와 같이 기본 동작이 생성되지 않은 경우에도 이렇게 할 수 있습니다. 또한 저장 프로시저를 통해 데이터베이스의 테이블에 액세스해야 하는 경우에 기본 업데이트 동작을 재정의할 수 있습니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

저장 프로시저를 지정하여 엔터티 클래스의 기본 동작을 재정의하려면

1. 디자이너에서 **LINQ to SQL** 파일을 엽니다. (**솔루션 탐색기**에서 **.dbml** 파일을 두 번 클릭합니다.)
2. **서버 탐색기** 또는 **데이터베이스 탐색기**에서 **저장 프로시저**를 확장하여 엔터티 클래스의 삽입, 업데이트 및/또는 삭제 명령에 사용할 저장 프로시저를 찾습니다.
3. 저장 프로시저를 **O/R 디자이너**로 끌어 놓습니다.

   저장 프로시저는 메서드 창에 DataContext 메서드로 추가됩니다. 자세한 내용은 DataContext 메서드 (O/R 디자이너)를 참조 하세요.

4. 업데이트 수행을 위해 저장 프로시저를 사용하려는 엔터티 클래스를 선택합니다.
5. **속성** 창에서 재정의할 **삽입**, **업데이트** 또는 **삭제** 명령을 선택합니다.
6. **런타임 사용** 옆의 줄임표(...)를 클릭하여 **동작 구성** 대화 상자를 엽니다.
7. **사용자 지정**을 선택합니다.
8. **사용자 지정** 목록에서 원하는 저장 프로시저를 선택합니다.
9. **메서드 인수** 및 **클래스 속성** 목록을 살펴보고 **메서드 인수**가 적절한 **클래스 속성**에 매핑되어 있는지 확인합니다. 원래 메서드 인수 (`Original_<ArgumentName>`)를 `Update` 및 `Delete` 명령의 원래 속성 (`<PropertyName> (Original)`)에 매핑합니다.

   > **NOTE**
   > 기본적으로 메서드 인수는 이름이 일치하는 경우 클래스 속성에 매핑됩니다. 속성 이름이 변경되어서 더 이상 테이블과 엔터티 클래스 간에 일치하지 않으면 디자이너에서 올바른 매핑을 결정할 수 없는 경우 매핑할 해당 클래스 속성을 선택해야 합니다.

10. **확인** 또는 **적용**을 클릭합니다.

    > **NOTE**
    > 각 변경을 수행한 후에 **적용** 을 클릭 하면 각 클래스 및 동작 조합의 동작을 계속 구성할 수 있습니다. **적용**을 클릭하기 전에 클래스 또는 동작을 변경 하면 경고 대화 상자가 표시 되 고 변경 내용을 적용할 수 있는 기회가 제공 됩니다.

업데이트의 기본 런타임 논리 사용으로 되돌아가려면 **속성** 창에서 **삽입**, **업데이트** 또는 **삭제** 명령 옆의 줄임표를 클릭한 다음, **동작 구성** 대화 상자에서 **런타임 사용**을 선택합니다.

참조

- Visual Studio의 LINQ to SQL 도구
- DataContext 메서드
- LINQ to SQL (.NET Framework)
- 삽입, 업데이트 및 삭제 작업 (.NET Framework)

방법: 복수형 설정 및 해제(O/R 디자이너)

2020-01-06 • 3 minutes to read • Edit Online

기본적으로 이름이로 끝나는 데이터베이스 개체를 **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 Visual Studio의 LINQ to SQL 도구로끌면 생성 된 엔터티 클래스의 이름이 복수형에서 단일로 변경 됩니다. 이렇게 하면 인스턴스화된 엔터티 클래스를 데이터의 단일 레코드에 매핑하는 것을 보다 정확하게 나타낼 수 있습니다. 예를 들어 **O/R 디자이너** 에 `Customers` 테이블을 추가 하면 클래스가 단일 고객에 대 한 데이터만 보유 하므로 `Customer` 라는 엔터티 클래스가 생성 됩니다.

> **NOTE**
> 복수 적용은 기본적으로 Visual Studio 영어 버전에만 적용됩니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

복수 적용을 설정 및 해제하려면

1. **도구** 메뉴에서 **옵션**을 클릭합니다.
2. **옵션** 대화 상자에서 **데이터베이스 도구**를 확장합니다.

   > **NOTE**
   > **데이터베이스 도구** 노드가 표시되지 않은 경우에는 **모든 설정 표시**를 선택합니다.

3. **O/R 디자이너**를 클릭합니다.
4. **Name의 복수화** 을 **Enabled** = **False** 로 설정 하 여 클래스 이름을 변경 하지 않도록 **O/R 디자이너** 를 설정 합니다.
5. **O/R 디자이너**에 추가 된 개체의 클래스 이름에 복수화 규칙을 적용 하려면 **이름 복수화** 을 **Enabled** = **True** 로 설정 합니다.

참조

- Visual Studio의 LINQ to SQL 도구
- LINQ to SQL
- Visual Studio에서 데이터 액세스

방법: Visual Studio에서 데이터 집합 만들기 및 구성

2020-01-06 • 10 minutes to read • Edit Online

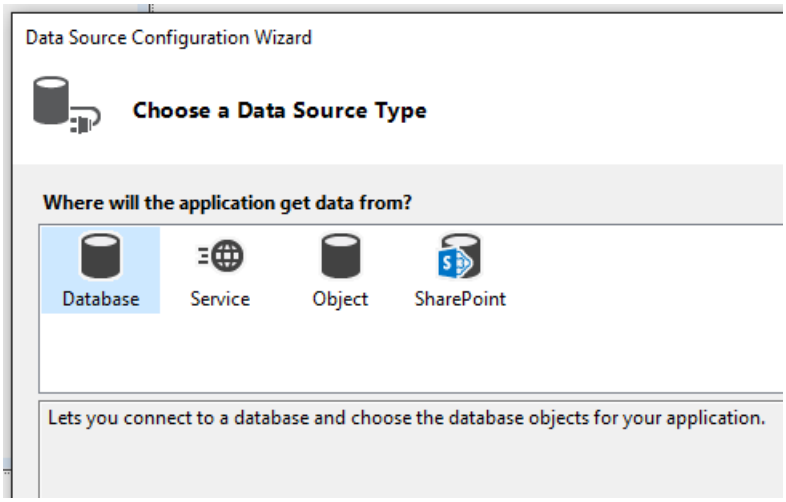
데이터 집합은 데이터베이스의 데이터를 메모리에 저장 하 고 변경 내용 추적을 지원 하 여 데이터베이스에 항상 연결 되어 있지 않아도 해당 데이터에 대 한 CRUD (만들기, 읽기, 업데이트 및 삭제) 작업을 가능 하 게 하는 개체 집합입니다. 데이터 집합은 데이터 비즈니스 응용 프로그램에 대 한 간단한 *양식을* 위해 설계 되었습니다. 새 응용 프로그램의 경우 Entity Framework를 사용 하 여 메모리에 데이터를 저장 하 고 모델링 하는 것이 좋습니다. 데이터 집합을 사용 하려면 데이터베이스 개념에 대 한 기본 지식이 있어야 합니다.

디자인 타임에 **데이터 소스 구성 마법사**를 사용 하 여 Visual Studio에서 형식화 된 DataSet 클래스를 만들 수 있습니다. 데이터 집합을 프로그래밍 방식으로 만드는 방법에 대 한 자세한 내용은 데이터 집합 만들기 (ADO.NET)를 참조 하십시오.

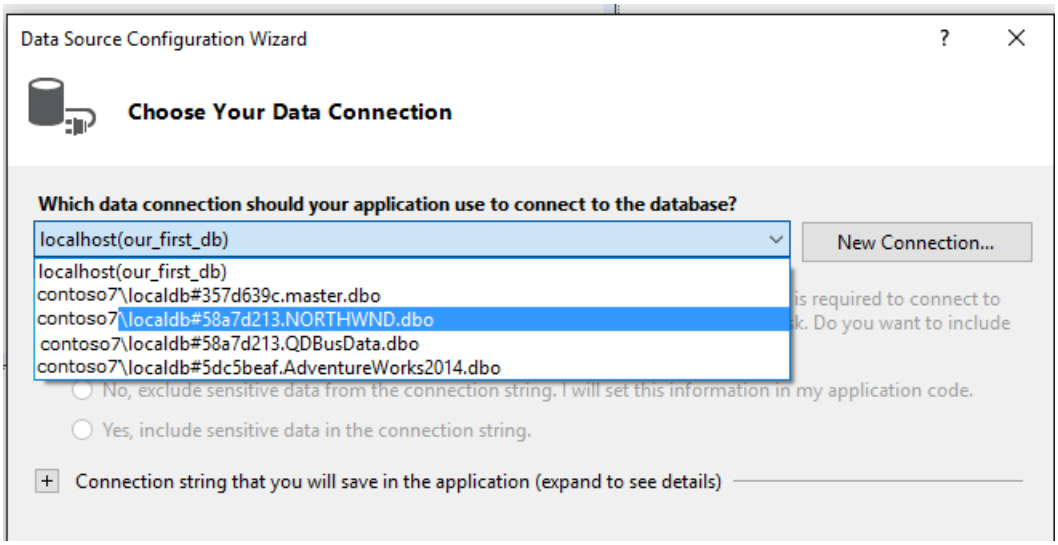
데이터 소스 구성 마법사를 사용 하 여 새 데이터 집합 만들기

1. Visual Studio에서 프로젝트를 연 다음 **프로젝트** > **새 데이터 소스 추가** 를 선택 하 여 **데이터 소스 구성 마법사**를 시작 합니다.

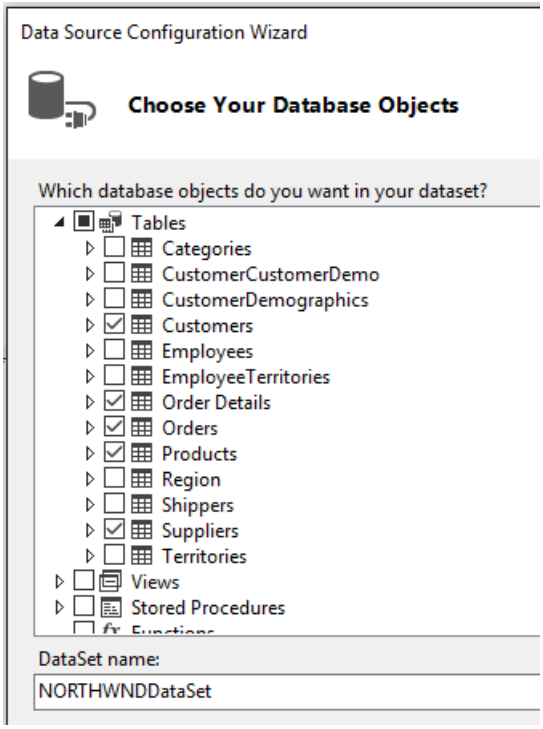
2. 연결할 데이터 원본의 유형을 선택 합니다.



3. 데이터 집합의 데이터 원본으로 사용할 데이터베이스를 선택 합니다.

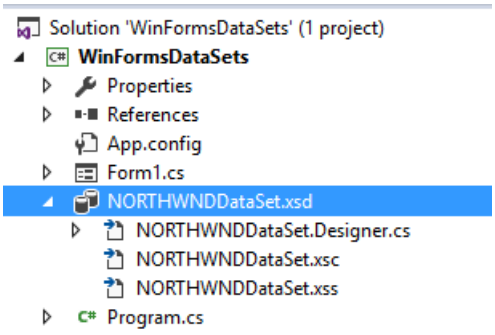


4. 데이터 집합에 표시할 테이블 (또는 개별 열), 저장 프로시저, 함수 및 뷰를 데이터베이스에서 선택 합니다.

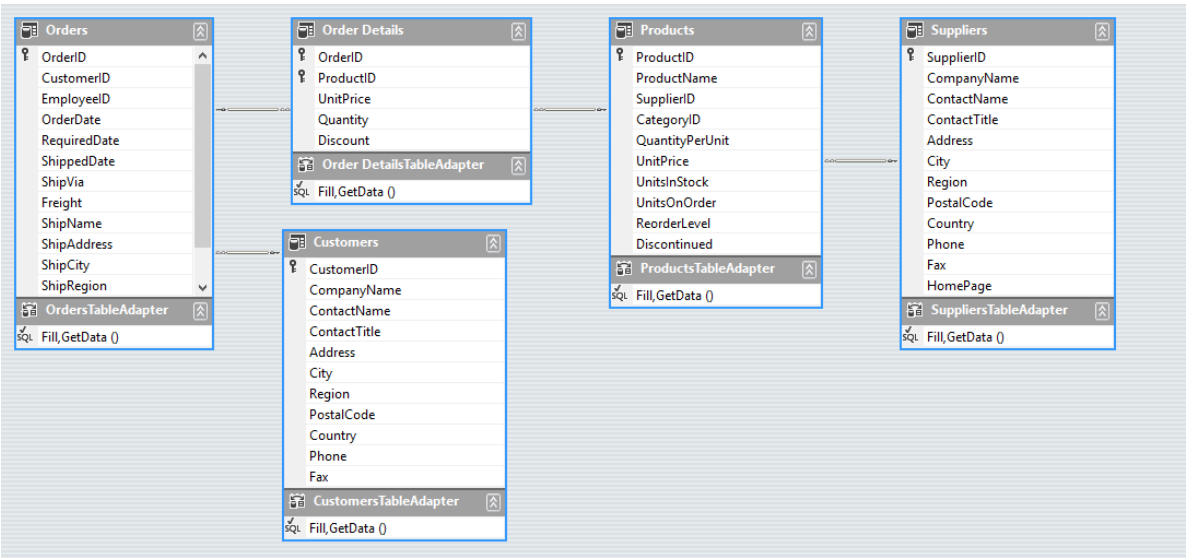


5. **마침**을 클릭합니다.

   데이터 집합은 **솔루션 탐색기** 노드로 표시 됩니다.

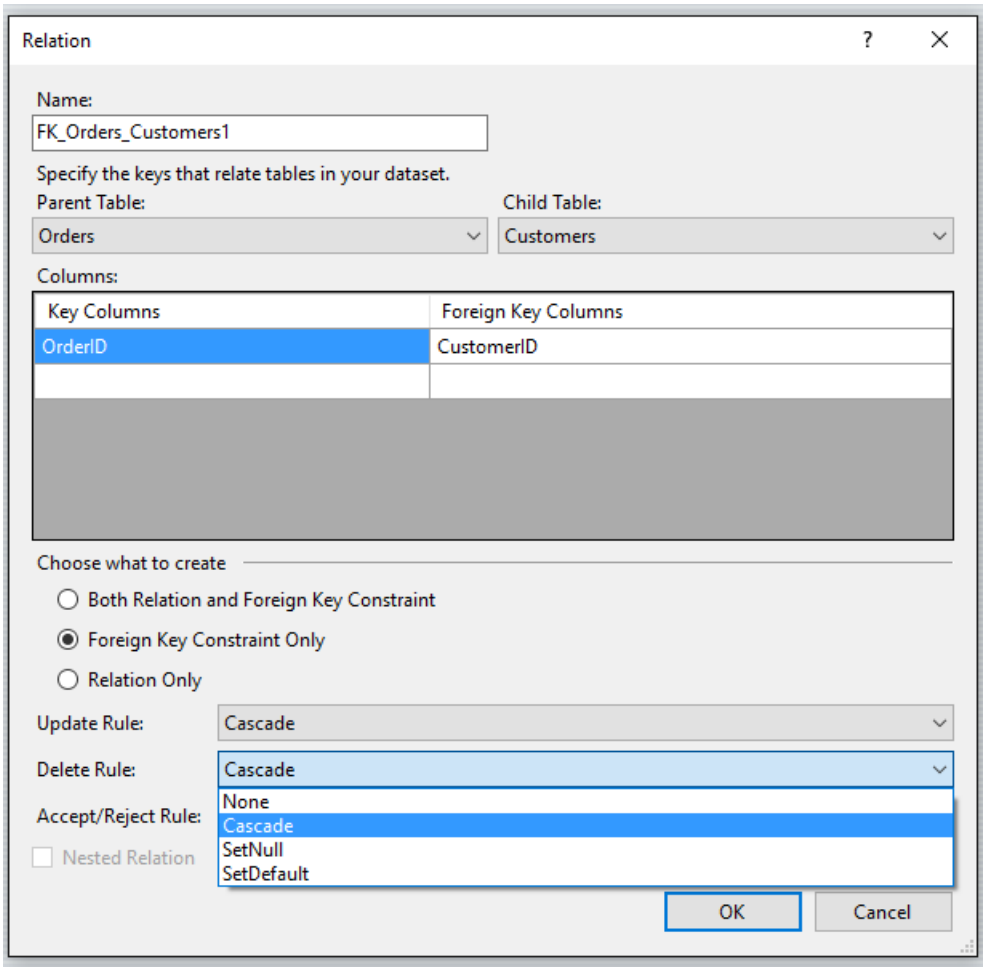


6. **솔루션 탐색기** 에서 데이터 집합 노드를 클릭 하 여 데이터 집합 **디자이너**에서 데이터 집합을 엽니다. 데이터 집합의 각 테이블에는 맨 아래에 표시 되는 `TableAdapter` 개체가 연결 되어 있습니다. 테이블 어댑터는 데이터 집합을 채우는 데 사용 되며 필요에 따라 데이터베이스에 명령을 보낼 수 있습니다.

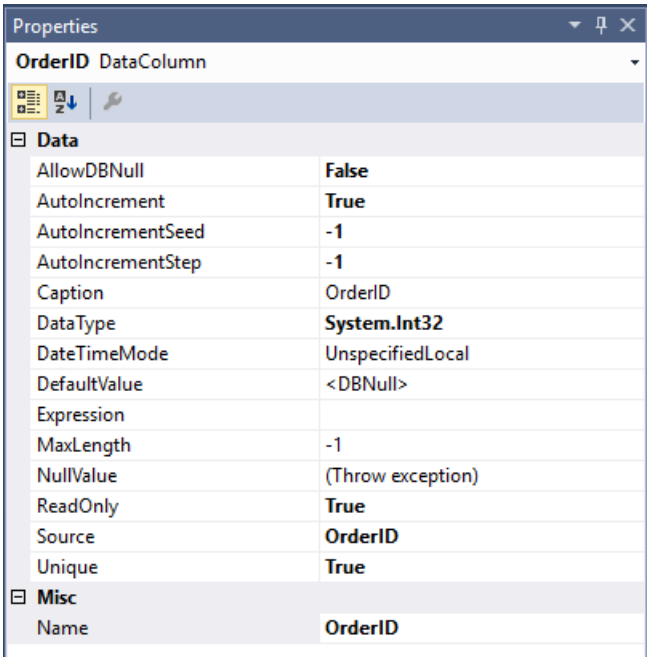


7. 테이블을 연결 하는 관계 선은 데이터베이스에 정의 된 테이블 관계를 나타냅니다. 기본적으로 데이터베이스의 foreign key 제약 조건은 업데이트 및 삭제 규칙이 없음으로 설정 된 관계로만 표시 됩니다. 일반적

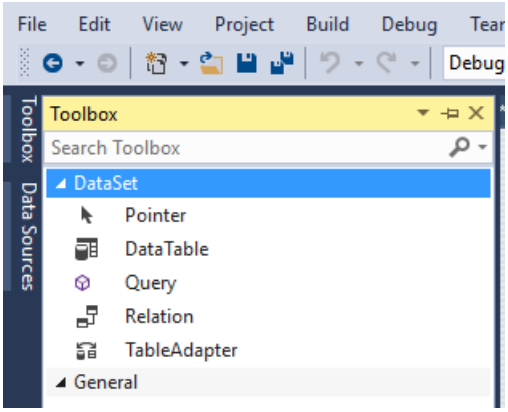
으로 원하는 것입니다. 그러나 해당 줄을 클릭 하 여 **관계** 대화 상자를 표시할 수 있습니다 .이 대화 상자에서 계층적 업데이트의 동작을 변경할 수 있습니다. 자세한 내용은 데이터 집합의 관계 및 계층적 업데이트를 참조 하세요.



8. 테이블에서 테이블, 테이블 어댑터 또는 열 이름을 클릭 하 여 **속성** 창에서 해당 속성을 확인 합니다. 여기에서 일부 값을 수정할 수 있습니다. 원본 데이터베이스가 아니라 데이터 집합을 수정 하는 것만 명심 하세요.



9. 새 테이블 또는 테이블 어댑터를 데이터 집합에 추가 하거나, 기존 테이블 어댑터에 대 한 새 쿼리를 추가 하거나, **도구 상자** 탭에서 해당 항목을 끌어 테이블 간에 새 관계를 지정할 수 있습니다. 이 탭은 **데이터 집합 디자이너** 에 포커스가 있을 때 나타납니다.

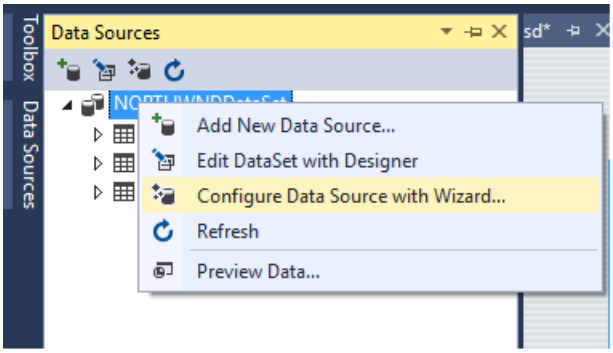


그런 다음 데이터를 사용 하 여 데이터 집합을 채우는 방법을 지정 하려고 할 수 있습니다. 이렇게 하려면 **TableAdapter 구성 마법사**를 사용 합니다. 자세한 내용은 tableadapter를 사용 하 여 데이터 집합 채우기를 참조 하세요.

기존 데이터 집합에 데이터베이스 테이블 또는 기타 개체 추가

이 절차에서는 데이터 집합을 처음 만들 때 사용한 것과 동일한 데이터베이스에서 테이블을 추가 하는 방법을 보여 줍니다.

1. **솔루션 탐색기** 에서 데이터 집합 노드를 클릭 하 여 **데이터 집합 디자이너** 를 포커스로 가져옵니다.
2. Visual Studio의 왼쪽 여백에 있는 **데이터 원본** 탭을 클릭 하거나 검색 상자에 **데이터 소스** 를 입력 합니다.
3. 데이터 집합 노드를 마우스 오른쪽 단추로 클릭 하 고 **마법사를 사용 하 여 데이터 원본 구성**을 선택 합니다.



4. 마법사를 사용 하 여 데이터 집합에 추가할 추가 테이블, 저장 프로시저 또는 기타 데이터베이스 개체를 지정 합니다.

데이터 집합에 독립 실행형 데이터 테이블 추가

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다.
2. **도구 상자** 의 **데이터 집합** 탭에서 **데이터 세트 디자이너**로 DataTable 클래스를 끌어 옵니다.
3. 열을 추가 하 여 데이터 테이블을 정의 합니다. 테이블을 마우스 오른쪽 단추로 클릭 하 고 > **열 추가** 를 선택 합니다. **속성** 창을 사용 하 여 열의 데이터 형식을 설정 하 고 필요한 경우 키를 설정 합니다.

독립 실행형 테이블은 데이터를 채울 수 있도록 독립 실행형 테이블에 `Fill` 논리를 구현 해야 합니다. 독립 실행형 데이터 테이블을 채우는 방법에 대 한 자세한 내용은 DataAdapter에서 데이터 집합 채우기를 참조 하세요.

참조

- Visual Studio의 데이터 세트 도구

- 데이터 세트에서의 관계
- 계층적 업데이트
- TableAdapter를 사용하여 데이터 세트 채우기

데이터 세트 간 관계 만들기

2020-01-06 • 16 minutes to read • Edit Online

관련 데이터 테이블이 포함 된 데이터 집합은 DataRelation 개체를 사용 하 여 테이블 간의 부모/자식 관계를 표시 하 고 서로 관련 된 레코드를 반환 합니다. **데이터 소스 구성 마법사**또는 **데이터 세트 디자이너**를 사용 하 여 데이터 집합에 관련 테이블을 추가 하면 DataRelation 개체가 생성 및 구성 됩니다.

DataRelation 개체는 다음 두 가지 기능을 수행 합니다.

- 작업 중인 레코드와 관련 된 레코드를 사용할 수 있게 만들 수 있습니다. 부모 레코드 (GetChildRows)에 있는 경우 자식 레코드를 제공 하 고 자식 레코드 (GetParentRow)로 작업 하는 경우 부모 레코드를 제공 합니다.
- 부모 레코드를 삭제할 때 관련 자식 레코드를 삭제 하는 등의 참조 무결성에 대 한 제약 조건을 적용할 수 있습니다.

DataRelation 개체의 실제 조인과 함수 간의 차이점을 이해 하는 것이 중요 합니다. 진정한 조인에서 레코드는 부모 및 자식 테이블에서 가져오며 단일 플랫 레코드 집합에 배치 됩니다. DataRelation 개체를 사용 하는 경우 새 레코드 집합을 만들 수 없습니다. 대신 DataRelation은 테이블 간의 관계를 추적 하 고 부모 및 자식 레코드를 동기화 된 상태로 유지 합니다.

DataRelation 개체 및 제약 조건

DataRelation 개체는 다음 제약 조건을 만들고 적용 하는 데도 사용 됩니다.

- 테이블의 열에 중복 항목이 포함 되지 않도록 보장 하는 unique 제약 조건입니다.
- 데이터 집합의 부모 및 자식 테이블 간에 참조 무결성을 유지 하는 데 사용할 수 있는 외래 키 제약 조건입니다.

DataRelation 개체에서 지정 하는 제약 조건은 자동으로 적절 한 개체를 만들거나 속성을 설정 하 여 구현 됩니다. DataRelation 개체를 사용 하 여 외래 키 제약 조건을 만드는 경우 ForeignKeyConstraint 클래스의 인스턴스가 DataRelation 개체의 ChildKeyConstraint 속성에 추가 됩니다.

Unique 제약 조건은 데이터 열의 Unique 속성을 `true`으로 설정 하거나 DataRelation 개체의 ParentKeyConstraint 속성에 UniqueConstraint 클래스의 인스턴스를 추가 하 여 구현 됩니다. 데이터 집합의 제약 조건 일시 중단에 대 한 자세한 내용은 데이터 집합을 채우는 동안 제약 조건해제를 참조 하세요.

참조 무결성 규칙

외래 키 제약 조건의 일부로 세 지점에서 적용 되는 참조 무결성 규칙을 지정할 수 있습니다.

- 부모 레코드가 업데이트 되는 경우
- 부모 레코드가 삭제 되는 경우
- 변경 내용이 수락 되거나 거부 되는 경우

지정할 수 있는 규칙은 Rule 열거에 지정 되며 다음 표에 나와 있습니다.

외래 키 제약 조건 규칙	동작
Cascade	부모 레코드에 대 한 변경 (업데이트 또는 삭제)은 자식 테이블의 관련 레코드에도 적용 됩니다.

외래 키 제약 조건 규칙	동작
SetNull	자식 레코드는 삭제 되지 않지만 자식 레코드의 외래 키는 DBNull로 설정 됩니다. 이 설정을 사용 하면 자식 레코드를 "고아"로 남겨둘 수 있습니다. 즉, 부모 레코드와 관계가 없습니다. **참고:** 이 규칙을 사용 하면 자식 테이블에 잘못 된 데이터가 생성 될 수 있습니다.
SetDefault	관련 자식 레코드의 외래 키는 열의 DefaultValue 속성으로 설정 된 대로 기본값으로 설정 됩니다.
None	관련 자식 레코드는 변경 되지 않습니다. 이 설정을 사용 하면 자식 레코드에 잘못 된 부모 레코드에 대 한 참조가 포함 될 수 있습니다.

데이터 집합 테이블의 업데이트에 대 한 자세한 내용은 데이터를 데이터베이스에 다시 저장을 참조 하세요.

제약 조건 전용 관계

DataRelation 개체를 만들 때는 제약 조건을 적용 하는 데만 관계를 사용 하도록 지정할 수 있습니다. 즉, 관련 레코드에 액세스 하는 데에도 사용 되지 않습니다. 이 옵션을 사용 하 여 약간 더 효율적이 고 관련 레코드 기능을 사용 하는 것 보다 더 작은 메서드를 포함 하는 데이터 집합을 생성할 수 있습니다. 그러나 관련 레코드에는 액세스할 수 없습니다. 예를 들어 제약 조건 전용 관계로 인해 자식 레코드가 아직 있는 부모 레코드를 삭제할 수 없으며, 부모를 통해 자식 레코드에 액세스할 수 없습니다.

데이터 세트 디자이너에서 수동으로 데이터 관계 만들기

Visual Studio에서 데이터 디자인 도구를 사용 하 여 데이터 테이블을 만들 때 데이터 원본에서 정보를 수집할 수 있는 경우 자동으로 관계가 만들어집니다. **도구 상자**의 데이터 **집합** 탭에서 수동으로 데이터 테이블을 추가 하는 경우 관계를 수동으로 만들어야 할 수 있습니다. 프로그래밍 방식으로 DataRelation 개체를 만드는 방법에 대 한 자세한 내용은 DataRelations 추가를 참조 하세요.

데이터 테이블 간의 관계는 관계의 일 대 다 측면을 보여 주는 키와 무한대 문자를 사용 하 여 **데이터 세트 디자이너**의 선으로 나타납니다. 기본적으로 관계의 이름은 디자인 화면에 표시 되지 않습니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

두 데이터 테이블 간에 관계를 만들려면

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.
2. **관계 개체를 데이터 집합** 도구 상자에서 관계의 자식 데이터 테이블로 끌어 놓습니다.

   관계 **대화 상자가** 열리고 **관계** 개체를 끌어온 테이블로 **자식 테이블** 상자를 채웁니다.
3. **부모 테이블** 상자에서 부모 테이블을 선택 합니다. 부모 테이블은 일 대 다 관계의 "일" 쪽에 있는 레코드를 포함 합니다.
4. **자식 테이블** 상자에 올바른 자식 테이블이 표시 되는지 확인 합니다. 자식 테이블에는 일 대 다 관계의 "다" 쪽에 있는 레코드가 포함 되어 있습니다.
5. **이름** 상자에 관계 이름을 입력 하거나 선택한 테이블을 기반으로 기본 이름을 그대로 둡니다. 코드의 실제 DataRelation 개체 이름입니다.
6. **키 열** 과 **외래 키 열** 목록에서 테이블을 조인 하는 열을 선택 합니다.

7. 관계, 제약 조건 또는 둘 다를 만들지 여부를 선택 합니다.

8. **중첩 관계** 상자를 선택 하거나 선택 취소 합니다. 이 옵션을 선택 하면 Nested 속성이 `true`로 설정 되 고 해당 행이 XML 데이터로 작성 되거나 XmlDataDocument와 동기화 될 때 관계의 자식 행이 부모 열 내에 중첩 됩니다. 자세한 내용은 중첩 Datarelation합니다.

9. 이러한 테이블의 레코드를 변경 하는 경우 적용 되는 규칙을 설정 합니다. 자세한 내용은 Rule를 참조하세요.

10. **확인** 을 클릭 하 여 관계를 만듭니다. 두 테이블 사이에 관계 선이 디자이너에 나타납니다.

데이터 세트 디자이너에 관계 이름을 표시 하려면

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.

2. **데이터** 메뉴에서 **관계 레이블 표시** 명령을 선택 하 여 관계 이름을 표시 합니다. 이 명령을 지워 관계 이름을 숨깁니다.

참조

- Visual Studio에서 데이터 세트 만들기 및 구성

연습: 데이터 세트 디자이너을 사용 하 여 데이터 집합 만들기

2020-01-06 • 9 minutes to read • Edit Online

이 연습에서는 **데이터 세트 디자이너**를 사용 하 여 데이터 집합을 만듭니다. 이 문서에서는 새 프로젝트를 만들고 새 프로젝트에 새 **데이터 집합** 항목을 추가 하는 과정을 안내 합니다. 마법사를 사용 하지 않고 데이터베이스의 테이블을 기반으로 테이블을 만드는 방법을 배웁니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. Visual Studio 설치 관리자에서 SQL Server Express LocalDB는 **데이터 저장소 및 처리** 워크 로드의 일부로 설치 되거나 개별 구성 요소로 설치 될 수 있습니다.

2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

새 Windows Forms 애플리케이션 프로젝트 만들기

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **Datasetdesignerwalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   Visual Studio는 **솔루션 탐색기** 에 프로젝트를 추가 하 고 디자이너에서 새 폼을 표시 합니다.

응용 프로그램에 새 데이터 집합 추가

1. **프로젝트** 메뉴에서 **새 항목 추가**를 선택합니다.

   새 항목 추가 대화 상자가 나타납니다.

2. 왼쪽 창에서 **데이터**를 선택한 다음 가운데 창에서 데이터 **집합** 을 선택 합니다.
3. 데이터 집합의 이름을 **NorthwindDataset**로 지정한 다음 **추가**를 선택 합니다.

   Visual Studio에서 **NorthwindDataset** 라는 파일을 프로젝트에 추가 하 고 **데이터 세트 디자이너**에서 엽니다.

서버 탐색기에서 데이터 연결 만들기

1. **보기** 메뉴에서 **서버 탐색기**를 클릭합니다.
2. **서버 탐색기**에서 **데이터베이스에 연결** 단추를 클릭 합니다.
3. Northwind 샘플 데이터베이스에 대 한 연결을 만듭니다.

데이터 집합에 테이블 만들기

이 섹션에서는 데이터 집합에 테이블을 추가 하는 방법에 대해 설명 합니다.

Customers 테이블을 만들려면

1. **서버 탐색기**에서 만든 데이터 연결을 확장 한 다음 **테이블** 노드를 확장 합니다.
2. **서버 탐색기** 에서 **데이터 세트 디자이너**로 **Customers** 테이블을 끕니다.

   Customers 데이터 테이블과 **Customerstableadapter** 가 데이터 집합에 추가 됩니다.

Orders 테이블을 만들려면

- **서버 탐색기** 에서 **데이터 세트 디자이너**로 **Orders** 테이블을 끌어 옵니다.

  Customers 테이블과 **Orders** 테이블 간의 **주문** 데이터 테이블, **orderstableadapter**및 데이터 관계가 데이터 집합에 추가 됩니다.

OrderDetails 테이블을 만들려면

- **서버 탐색기** 에서 **데이터 세트 디자이너**로 **주문 정보** 테이블을 끕니다.

  Order Details 데이터 테이블, **OrderDetailsTableAdapter**및 **Orders** 테이블과 **OrderDetails** 테이블 간의 데이터 관계가 데이터 집합에 추가 됩니다.

다음 단계

- 데이터 집합을 저장 합니다.
- **데이터 소스** 창에서 항목을 선택 하 고 폼으로 끌어 놓습니다. 자세한 내용은 Windows Forms 컨트롤을 Visual Studio의 데이터에 바인딩을 참조 하세요.
- Tableadapter에 쿼리를 추가 합니다.
- 데이터 집합의 데이터 테이블에 대 한 ColumnChanging 또는 RowChanging 이벤트에 유효성 검사 논리를 추가 합니다. 자세한 내용은 데이터 집합의 데이터 유효성 검사를 참조 하세요.

참조

- Visual Studio에서 데이터 세트 만들기 및 구성
- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩
- Visual Studio에서 데이터에 컨트롤 바인딩
- 데이터 유효성 검사

연습: 데이터 세트 디자이너에서 DataTable 만들기

2020-01-06 • 4 minutes to read • Edit Online

이 연습에서는 **데이터 세트 디자이너**를 사용 하 여 TableAdapter 없이 DataTable를 만드는 방법을 설명 합니다. Tableadapter를 포함 하는 데이터 테이블을 만드는 방법에 대 한 자세한 내용은 Tableadapter 만들기 및 구성을 참조 하세요.

새 Windows Forms 애플리케이션 만들기

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창**에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **DataTableWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   DataTableWalkthrough 프로젝트가 만들어지고 **솔루션 탐색기**에 추가 됩니다.

응용 프로그램에 새 데이터 집합 추가

1. **프로젝트** 메뉴에서 **새 항목 추가**를 선택합니다.

   새 항목 추가 대화 상자가 나타납니다.
2. 왼쪽 창에서 **데이터**를 선택한 다음 가운데 창에서 데이터 **집합** 을 선택 합니다.
3. **추가**를 선택합니다.

   Visual Studio에서 **DataSet1** 라는 파일을 프로젝트에 추가 하 고 **데이터 세트 디자이너**에서 엽니다.

데이터 집합에 새 DataTable 추가

1. **도구 상자** 의 **데이터 집합** 탭에서 **데이터 세트 디자이너**로 **DataTable** 을 끌어 옵니다.

   DataTable1 이라는 테이블이 데이터 집합에 추가 됩니다.
2. **DataTable1** 의 제목 표시줄을 클릭 하 고 이름을 `Music` 로 바꿉니다.

DataTable에 열 추가

1. **Music** 테이블을 마우스 오른쪽 단추로 클릭 합니다. **추가**를 가리킨 다음 **열**을 클릭 합니다.
2. 열 이름을 `SongID` 로 합니다.
3. **속성** 창에서 DataType 속성을 System.Int16로 설정합니다.
4. 이 프로세스를 반복 하 고 다음 열을 추가 합니다.

   `SongTitle` : System.String

   `Artist` : System.String

   `Genre` : System.String

테이블에 대 한 기본 키 설정

모든 데이터 테이블에는 기본 키가 있어야 합니다. 기본 키는 데이터 테이블의 특정 레코드를 고유 하 게 식별 합니다.

기본 키를 설정 하려면 **SongID** 열을 마우스 오른쪽 단추로 클릭 한 다음 **기본 키 설정**을 클릭 합니다. 키 아이콘이 **SongID** 열 옆에 표시 됩니다.

프로젝트 저장

DataTableWalkthrough 프로젝트를 저장 하려면 **파일** 메뉴에서 **모두 저장**을 선택 합니다.

참조

- Visual Studio에서 데이터 세트 만들기 및 구성
- Visual Studio에서 데이터에 컨트롤 바인딩
- 데이터 유효성 검사

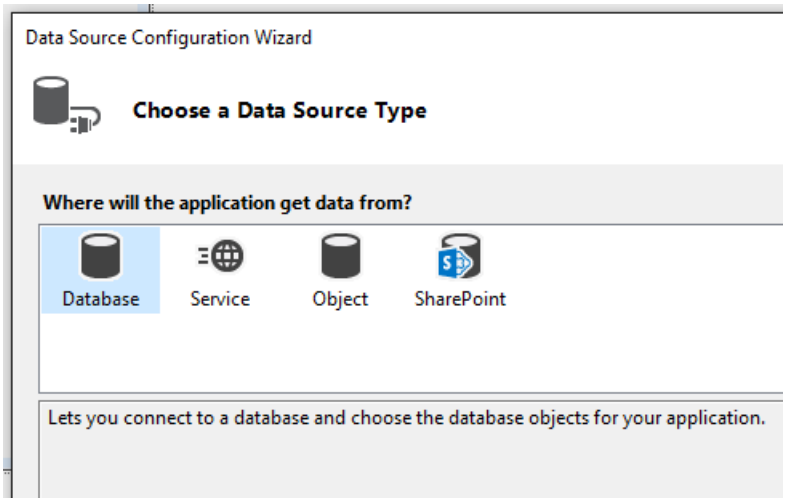
TableAdapter 만들기 및 구성

2020-01-06 • 18 minutes to read • Edit Online

Tableadapter는 응용 프로그램과 데이터베이스 간의 통신을 제공 합니다. 데이터베이스에 연결 하 여 쿼리 또는 저장 프로시저를 실행 하 고 새 데이터 테이블을 반환 하거나 반환 된 데이터로 기존 DataTable을 채웁니다. Tableadapter는 응용 프로그램에서 업데이트 된 데이터를 다시 데이터베이스로 보낼 수도 있습니다.

Tableadapter는 다음 작업 중 하나를 수행할 때 생성 됩니다.

- **서버 탐색기** 에서 **데이터 세트 디자이너**으로 데이터베이스 개체를 끌어 옵니다.
- 데이터 소스 구성 마법사를 실행 하 고 **데이터베이스** 또는 **웹 서비스** 데이터 원본 유형 중 하나를 선택 합니다.



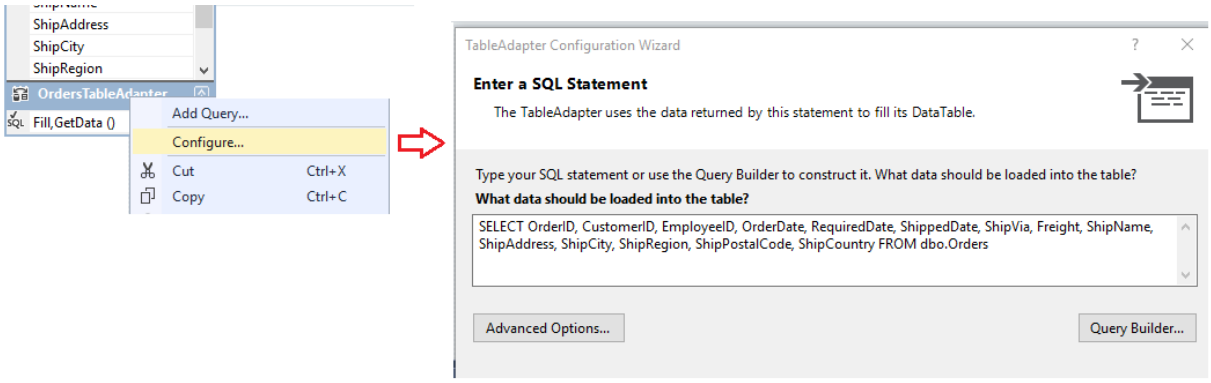
또한 **도구 상자** 에서 tableadapter를 **데이터 세트 디자이너** 화면의 빈 영역으로 끌어서 새 tableadapter를 만들고 데이터 소스를 사용 하 여 구성할 수 있습니다.

Tableadapter에 대 한 소개는 tableadapter를 사용 하 여 데이터 집합 채우기를 참조 하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

TableAdapter 구성 마법사 사용

Tableadapter 구성 마법사 를 실행 하 여 tableadapter 및 연결 된 datatable을 만들거나 편집 합니다. **데이터 세트 디자이너**에서 기존 TableAdapter를 마우스 오른쪽 단추로 클릭 하 여 구성할 수 있습니다.



데이터 세트 디자이너 포커스가 있을 때 도구 상자에서 새 TableAdapter를 끌면 마법사가 시작 되 고 TableAdapter가 연결 해야 하는 데이터 소스를 지정 하 라는 메시지가 표시 됩니다. 다음 페이지에서 마법사는 데이터베이스와 통신 하는 데 사용 해야 하는 명령의 종류 (SQL 문 또는 저장 프로시저)를 묻는 메시지를 표시 합니다. 데이터 원본에 이미 연결 된 TableAdapter를 구성 하는 경우에는이 내용이 표시 되지 않습니다.

- 데이터베이스에 대 한 올바른 사용 권한이 있는 경우 기본 데이터베이스에 새 저장 프로시저를 만들 수 있는 옵션이 있습니다. 이러한 권한이 없는 경우에는이 옵션을 사용할 수 없습니다.
- TableAdapter의 **SELECT**, **INSERT**, **UPDATE** 및 **DELETE** 명령에 대해 기존 저장 프로시저를 실행 하도록 선택할 수도 있습니다. 예를 들어 **Update** 명령에 할당 된 저장 프로시저는 `TableAdapter.Update()` 메서드가 호출 될 때 실행 됩니다.

선택한 저장 프로시저에서 데이터 테이블의 해당 열로 매개 변수를 매핑합니다. 예를 들어 저장 프로시저가 테이블의 `CompanyName` 열에 전달 하는 `@CompanyName` 이라는 매개 변수를 허용 하는 경우 `@CompanyName` 매개 변수의 **원본 열** 을 `CompanyName` 로 설정 합니다.

> **NOTE**
> SELECT 명령에 할당 되는 저장 프로시저는 마법사의 다음 단계에서 이름을 지정 하는 TableAdapter의 메서드를 호출 하여 실행 합니다. 기본 메서드는 `Fill` 이므로 일반적으로 SELECT 프로시저를 실행 하는 데 사용 되는 코드는 `TableAdapter.Fill(tableName)` 됩니다. `Fill` 에서 기본 이름을 변경 하는 경우 사용자가 할당 한 이름으로 `Fill` 를 대체 하 고 "TableAdapter"를 TableAdapter의 실제 이름 (예: `CustomersTableAdapter`)으로 바꿉니다.

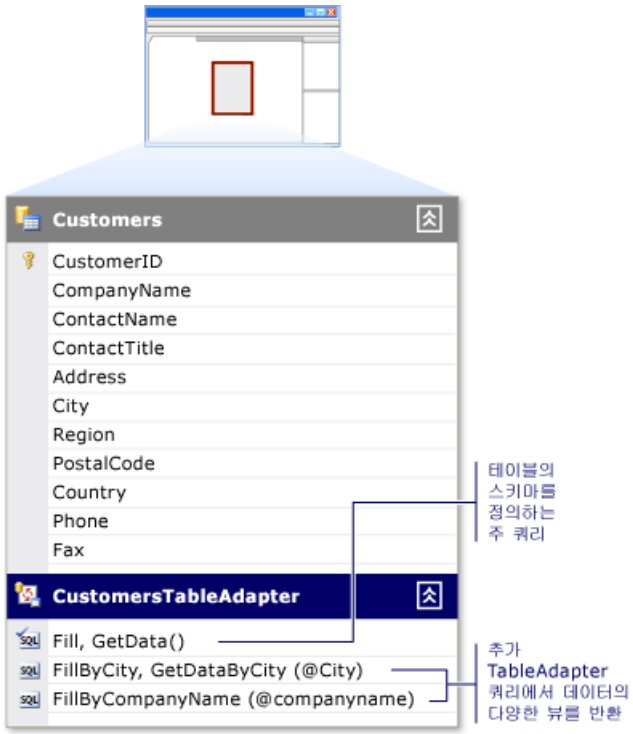
- 데이터베이스 옵션 **에 직접 업데이트를 전송 하는 Create 메서드** 를 선택 하는 것은 `GenerateDBDirectMethods` 속성을 true로 설정 하는 것과 같습니다. 원래 SQL 문에서 충분한 정보를 제공하지 않거나 쿼리가 업데이트할 수 없는 쿼리인 경우에는 이 옵션을 사용할 수 없습니다. 예를 들어 **조인** 쿼리와 단일 (스칼라) 값을 반환 하는 쿼리에서 이러한 상황이 발생할 수 있습니다.

마법사의 **고급 옵션** 을 사용 하 여 다음 작업을 수행할 수 있습니다.

- **SQL 문 생성** 페이지에 정의 된 SELECT 문을 기반으로 INSERT, UPDATE 및 DELETE 문을 생성 합니다.
- 낙관적 동시성 사용
- INSERT 및 UPDATE 문을 실행 한 후 데이터 테이블을 새로 고칠지 여부를 지정 합니다.

TableAdapter의 Fill 메서드 구성

TableAdapter의 테이블 스키마를 변경 해야 하는 경우가 있습니다. 이렇게 하려면 TableAdapter의 기본 `Fill` 메서드를 수정 합니다. Tableadapter는 연결 된 데이터 테이블의 스키마를 정의 하는 기본 `Fill` 메서드를 사용 하 여 생성 됩니다. 기본 `Fill` 방법은 TableAdapter를 처음 구성할 때 입력 한 쿼리 또는 저장 프로시저를 기반으로 합니다. 데이터 집합 디자이너의 데이터 테이블 아래에 있는 첫 번째 (최상위) 메서드입니다.



TableAdapter의 주 `Fill` 메서드에 대 한 모든 변경 내용은 연결 된 데이터 테이블의 스키마에 반영 됩니다. 예를 들어 주 `Fill` 메서드의 쿼리에서 열을 제거 하면 연결 된 데이터 테이블 에서도 해당 열이 제거 됩니다. 또한 주 `Fill` 메서드에서 열을 제거 하면 해당 TableAdapter에 대 한 추가 쿼리에서 해당 열이 제거 됩니다.

TableAdapter 쿼리 구성 마법사를 사용 하 여 TableAdapter에 대 한 추가 쿼리를 만들고 편집할 수 있습니다. 이러한 추가 쿼리는 스칼라 값을 반환 하는 경우를 제외 하 고 테이블 스키마를 준수 해야 합니다. 각 추가 쿼리에는 지정 된 이름이 있습니다.

다음 예제에서는 `FillByCity`이라는 추가 쿼리를 호출 하는 방법을 보여 줍니다.

```
CustomersTableAdapter.FillByCity(NorthwindDataSet.Customers, "Seattle")
```

새 쿼리를 사용 하 여 **TableAdapter** 쿼리 구성 마법사를 시작 하려면

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다.
2. 새 쿼리를 만드는 경우 **도구 상자** 의 **데이터 집합** 탭에서 DataTable로 **쿼리** 개체를 끌거나 TableAdapter의 바로 가기 메뉴에서 **쿼리 추가** 를 선택 합니다. **쿼리** 개체를 **데이터 세트 디자이너**의 빈 영역으로 끌어와 연결 된 DataTable없이 TableAdapter를 만들 수도 있습니다. 이러한 쿼리는 단일 (스칼라) 값만 반환 하거나 데이터베이스에 대해 UPDATE, INSERT 또는 DELETE 명령을 실행할 수 있습니다.
3. **데이터 연결 선택** 화면에서 쿼리에서 사용할 연결을 선택 하거나 만듭니다.

   > **NOTE**
   > 이 화면은 디자이너에서 사용할 적절 한 연결을 결정할 수 없는 경우 나 사용할 수 있는 연결이 없는 경우에만 표시 됩니다.

4. **명령 유형 선택** 화면에서 데이터베이스에서 데이터를 가져오는 다음 방법 중에서 선택 합니다.
   - **Sql 문을 사용** 하 여 데이터베이스에서 데이터를 선택 하는 sql 문을 입력할 수 있습니다.
   - **새 저장 프로시저 만들기** 를 사용 하면 마법사에서 지정 된 SELECT 문을 기반으로 데이터베이스에 새 저장 프로시저를 만들 수 있습니다.
   - **기존 저장 프로시저를 사용** 하면 쿼리를 실행할 때 기존 저장 프로시저를 실행할 수 있습니다.

기존 쿼리에서 **TableAdapter** 쿼리 구성 마법사를 시작 하려면

- 기존 TableAdapter 쿼리를 편집 하는 경우 쿼리를 마우스 오른쪽 단추로 클릭 한 다음 바로 가기 메뉴에서 **구성** 을 선택 합니다.

> **NOTE**
>
> TableAdapter의 주 쿼리를 마우스 오른쪽 단추로 클릭 하면 TableAdapter 및 DataTable 스키마가 다시 됩니다. 그러나 TableAdapter에서 추가 쿼리를 마우스 오른쪽 단추로 클릭 하면 선택한 쿼리만 구성 됩니다. Tableadapter **구성 마법사** 는 tableadapter 정의를 다시 구성 하는 반면 **Tableadapter 쿼리 구성 마법사** 는 선택한 쿼리를 다시 구성 합니다.

TableAdapter에 전역 쿼리를 추가 하려면

- 전역 쿼리는 단일 (스칼라) 값을 반환 하거나 값을 반환 하지 않는 SQL 쿼리입니다. 일반적으로 전역 함수는 삽입, 업데이트 및 삭제와 같은 데이터베이스 작업을 수행 합니다. 또한 테이블의 고객 수 또는 특정 주문의 모든 항목에 대 한 총 요금 같은 정보를 집계 합니다.

  도구 상자 의 **데이터 집합** 탭에서 **데이터 세트 디자이너**의 빈 영역으로 **쿼리** 개체를 끌어서 전역 쿼리를 추가 합니다.

- 원하는 작업 (예: `SELECT COUNT(*) AS CustomerCount FROM Customers`)을 수행 하는 쿼리를 제공 합니다.

> **NOTE**
>
> **쿼리** 개체를 **데이터 세트 디자이너** 직접 끌면 스칼라 (단일) 값만 반환 하는 메서드가 만들어집니다. 선택한 쿼리나 저장 프로시저가 둘 이상의 값을 반환할 수 있지만 마법사로 만든 메서드는 단일 값만 반환 합니다. 예를 들어 쿼리는 반환 된 데이터에서 첫 번째 행의 첫 번째 열을 반환할 수 있습니다.

참조

- TableAdapter를 사용하여 데이터 세트 채우기

매개 변수가 있는 TableAdapter 쿼리 만들기

2020-01-06 • 8 minutes to read • Edit Online

매개 변수가 있는 쿼리는 쿼리의 WHERE 절 조건을 충족하는 데이터를 반환합니다. 예를 들어 고객 목록을 반환하는 SQL 문 끝에 `WHERE City = @City`를 추가하여 특정 구/군/시의 고객만 표시하도록 고객 목록을 매개 변수화할 수 있습니다.

데이터 세트 디자이너에서 매개 변수가 있는 TableAdapter 쿼리를 만듭니다. **데이터** 메뉴에서 **데이터 소스 매개 변수화** 명령을 사용 하 여 Windows 응용 프로그램에서 만들 수도 있습니다. **데이터 소스 매개 변수화** 명령은 매개 변수 값을 입력 하 고 쿼리를 실행할 수 있는 폼에 컨트롤을 만듭니다.

> **NOTE**
> 매개 변수가 있는 쿼리를 생성할 때 코딩 하는 데이터베이스와 관련 된 매개 변수 표기법을 사용 합니다. 예를 들어 Access 및 OleDb 데이터 소스는 물음표 '?'를 사용하여 매개 변수를 표기하므로 WHERE 절은 `WHERE City = ?`와 같습니다.

매개 변수가 있는 TableAdapter 쿼리 만들기

데이터 세트 디자이너에서 매개 변수가 있는 쿼리를 만들려면

- 원하는 매개 변수가 포함된 WHERE 절을 SQL 문에 추가하여 새 TableAdapter를 만듭니다. 자세한 내용은 Tableadapter 만들기 및 구성을 참조 하세요.

  -또는-

- 원하는 매개 변수가 포함된 WHERE 절을 SQL 문에 추가하여 기존 TableAdapter에 쿼리를 추가합니다.

데이터 바인딩된 폼을 디자인하면서 매개 변수가 있는 쿼리를 만들려면

1. 데이터 세트에 이미 바인딩되어 있는 폼의 컨트롤을 선택합니다. 자세한 내용은 Windows Forms 컨트롤을 Visual Studio의 데이터에 바인딩을 참조 하세요.
2. **데이터** 메뉴에서 **쿼리 추가**를 선택 합니다.
3. 원하는 매개 변수가 포함된 WHERE 절을 SQL 문에 추가하여 **검색 조건 작성기** 대화 상자에서 필요한 작업을 완료합니다.

기존 데이터 바인딩된 폼에 쿼리를 추가하려면

1. **Windows Forms 디자이너**에서 폼을 엽니다.
2. **데이터** 메뉴에서 **쿼리 추가** 또는 **데이터 스마트 태그**를 선택 합니다.

   > **NOTE**
   > **데이터** 메뉴에서 **쿼리 추가**를 사용할 수 없는 경우 매개 변수화를 추가할 데이터 소스가 표시되어 있는 폼의 컨트롤을 선택합니다. 예를 들어 폼의 DataGridView 컨트롤에 데이터가 표시되는 경우 해당 컨트롤을 선택합니다. 폼의 개별 컨트롤에 데이터가 표시되는 경우에는 데이터 바인딩된 컨트롤을 선택합니다.

3. **데이터 원본 테이블 선택** 영역에서 매개 변수화를 추가 하려는 테이블을 선택 합니다.
4. 새 쿼리를 만드는 경우 **새 쿼리 이름** 상자에 이름을 입력합니다.

   -또는-

기존 쿼리 이름 상자에서 쿼리를 선택합니다.

5. **쿼리 텍스트** 상자에 매개 변수를 사용 하는 쿼리를 입력 합니다.
6. **확인**을 선택합니다.

   매개 변수를 입력하기 위한 컨트롤과 **로드** 단추가 ToolStrip 컨트롤의 폼에 추가됩니다.

Null 값 쿼리

현재 값이 없는 레코드를 쿼리하려면 TableAdapter 매개 변수에 null 값을 할당할 수 있습니다. 예를 들어 `WHERE` 절에 `ShippedDate` 매개 변수가 있는 다음 쿼리를 살펴보세요.

```
SELECT CustomerID, OrderDate, ShippedDate
FROM Orders
WHERE (ShippedDate = @ShippedDate) OR (ShippedDate IS NULL)
```

TableAdapter에 대 한 쿼리 인 경우 다음 코드와 함께 제공 되지 않은 모든 주문을 쿼리할 수 있습니다.

```
ordersTableAdapter.FillByShippedDate(northwindDataSet.Orders, null);
```

```
OrdersTableAdapter.FillByShippedDate(NorthwindDataSet.Orders, Nothing)
```

쿼리가 null 값을 허용 하도록 설정 하려면 다음을 수행 합니다.

1. **데이터 세트 디자이너**에서 null 매개 변수 값을 허용 해야 하는 TableAdapter 쿼리를 선택 합니다.
2. **속성** 창에서 **매개 변수**를 선택 하 고 줄임표 (...) 단추를 클릭 하 여 **매개 변수 컬렉션 편집기**를 엽니다.
3. Null 값을 허용 하는 매개 변수를 선택 하 고 **Allowdbnull** 속성을 `true`로 설정 합니다.

참조

- TableAdapter를 사용하여 데이터 세트 채우기

TableAdapter를 사용하여 데이터베이스에 직접 액세스

2020-01-06 • 4 minutes to read • Edit Online

`InsertCommand`, `UpdateCommand` 및 `DeleteCommand` 외에도 Tableadapter는 데이터베이스에 대해 직접 실행할 수 있는 메서드를 사용 하 여 생성 됩니다. 이러한 메서드 (`TableAdapter.Insert`, `TableAdapter.Update` 및 `TableAdapter.Delete`)를 호출 하 여 데이터베이스에서 직접 데이터를 조작할 수 있습니다.

이러한 직접 메서드를 만들지 않으려면 **속성** 창에서 TableAdapter의 `GenerateDbDirectMethods` 속성을 `false` 로 설정 합니다. Tableadapter의 주 쿼리와 함께 TableAdapter에 쿼리를 추가 하는 경우 이러한 `DbDirect` 메서드를 생성 하지 않는 독립 실행형 쿼리를 사용할 수 있습니다.

데이터베이스에 직접 명령 보내기

수행 하려는 작업을 수행 하는 TableAdapter `DbDirect` 메서드를 호출 합니다.

데이터베이스에 새 레코드를 직접 삽입 하려면

- TableAdapter의 `Insert` 메서드를 호출 하 여 각 열에 대 한 값을 매개 변수로 전달 합니다. 다음 절차에서는 Northwind 데이터베이스의 `Region` 테이블을 예로 사용 합니다.

  > **NOTE**
  > 사용할 수 있는 인스턴스가 없는 경우 사용할 TableAdapter를 인스턴스화합니다.

```
Dim regionTableAdapter As New NorthwindDataSetTableAdapters.RegionTableAdapter

regionTableAdapter.Insert(5, "NorthWestern")
```

```
NorthwindDataSetTableAdapters.RegionTableAdapter regionTableAdapter =
    new NorthwindDataSetTableAdapters.RegionTableAdapter();

regionTableAdapter.Insert(5, "NorthWestern");
```

데이터베이스에서 직접 레코드를 업데이트 하려면

- TableAdapter의 `Update` 메서드를 호출 하 여 각 열에 대 한 새 값과 원래 값을 매개 변수로 전달 합니다.

  > **NOTE**
  > 사용할 수 있는 인스턴스가 없는 경우 사용할 TableAdapter를 인스턴스화합니다.

```
Dim regionTableAdapter As New NorthwindDataSetTableAdapters.RegionTableAdapter

regionTableAdapter.Update(1, "East", 1, "Eastern")
```

```
NorthwindDataSetTableAdapters.RegionTableAdapter regionTableAdapter =
    new NorthwindDataSetTableAdapters.RegionTableAdapter();

regionTableAdapter.Update(1, "East", 1, "Eastern");
```

데이터베이스에서 직접 레코드를 삭제 하려면

- TableAdapter의 `Delete` 메서드를 호출 하 여 각 열에 대 한 값을 `Delete` 메서드의 매개 변수로 전달 합니다. 다음 절차에서는 Northwind 데이터베이스의 `Region` 테이블을 예로 사용 합니다.

> **NOTE**
> 사용할 수 있는 인스턴스가 없는 경우 사용할 TableAdapter를 인스턴스화합니다.

```
Dim regionTableAdapter As New NorthwindDataSetTableAdapters.RegionTableAdapter

regionTableAdapter.Delete(5, "NorthWestern")
```

```
NorthwindDataSetTableAdapters.RegionTableAdapter regionTableAdapter =
    new NorthwindDataSetTableAdapters.RegionTableAdapter();

regionTableAdapter.Delete(5, "NorthWestern");
```

참조

- TableAdapter를 사용하여 데이터 세트 채우기

데이터 세트를 채우는 동안 제약 조건 해제

2020-01-06 • 3 minutes to read • Edit Online

데이터 집합에 제약 조건 (예: foreign key 제약 조건)이 포함 된 경우 데이터 집합에 대해 수행 되는 작업의 순서와 관련 된 오류를 발생 시킬 수 있습니다. 예를 들어 관련 된 부모 레코드를 로드 하기 전에 자식 레코드를 로드 하면 제약 조건을 위반 하 여 오류가 발생할 수 있습니다. 자식 레코드를 로드 하는 즉시 제약 조건은 관련 된 부모 레코드를 확인 하 고 오류를 발생 시킵니다.

임시 제약 조건 일시 중단을 허용 하는 메커니즘이 없는 경우 자식 테이블에 레코드를 로드 하려고 할 때마다 오류가 발생 합니다. 데이터 집합의 모든 제약 조건을 일시 중단 하는 또 다른 방법은 BeginEdit및 EndEdit 속성을 사용 하는 것입니다.

> **NOTE**
> 제약 조건이 해제 되 면 유효성 검사 이벤트 (예: ColumnChanging 및 RowChanging)가 발생 하지 않습니다.

프로그래밍 방식으로 업데이트 제약 조건을 일시 중단 하려면

- 다음 예제에서는 데이터 집합에서 제약 조건 확인을 일시적으로 해제 하는 방법을 보여 줍니다.

```
dataSet1.EnforceConstraints = false;
// Perform some operations on the dataset
dataSet1.EnforceConstraints = true;
```

```
DataSet1.EnforceConstraints = False
' Perform some operations on the dataset
DataSet1.EnforceConstraints = True
```

데이터 세트 디자이너를 사용 하 여 업데이트 제약 조건을 일시 중단 하려면

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.
2. **속성** 창에서 EnforceConstraints 속성을 `false`로 설정합니다.

참조

- TableAdapter를 사용하여 데이터 세트 채우기
- 데이터 세트에서의 관계

TableAdapter의 기능 확장

2020-01-06 • 3 minutes to read • Edit Online

TableAdapter의 partial 클래스 파일에 코드를 추가 하 여 TableAdapter의 기능을 확장할 수 있습니다.

Tableadapter를 정의 하는 코드는 **데이터 세트 디자이너**에서 tableadapter가 변경 되거나 마법사에서 tableadapter의 구성을 수정할 때 다시 생성 됩니다. TableAdapter를 다시 생성 하는 동안 코드가 삭제 되지 않도록 하려면 TableAdapter의 partial 클래스 파일에 코드를 추가 합니다.

Partial 클래스를 사용 하면 특정 클래스에 대 한 코드를 여러 물리적 파일 간에 분할할 수 있습니다. 자세한 내용은 부분 또는 부분 (형식)을 참조 하세요.

코드에서 Tableadapter 찾기

Tableadapter는 **데이터 세트 디자이너**를 사용 하 여 디자인 되었지만 생성 된 tableadapter 클래스는 DataSet의 중첩 클래스가 아닙니다. Tableadapter는 TableAdapter의 연결 된 데이터 집합 이름을 기반으로 하는 네임 스페이스에 있습니다. 예를 들어 응용 프로그램에 `HRDataSet`이라는 데이터 집합이 포함 되어 있으면 Tableadapter는 `HRDataSetTableAdapters` 네임 스페이스에 있습니다. (명명 규칙은이 패턴을 따릅니다: *DatasetName* + `TableAdapters`).

다음 예제에서는 `CustomersTableAdapter` TableAdapter가 `NorthwindDataSet`있는 프로젝트에 있다고 가정 합니다.

TableAdapter의 **partial** 클래스를 만들려면

1. **프로젝트** 메뉴로 이동 하 고 **클래스 추가**를 선택 하 여 프로젝트에 새 클래스를 추가 합니다.
2. 클래스 이름을 `CustomersTableAdapterExtended`로 지정합니다.
3. **추가**를 선택합니다.
4. 다음과 같이 코드를 프로젝트의 올바른 네임 스페이스 및 partial 클래스 이름으로 바꿉니다.

```
namespace NorthwindDataSetTableAdapters
{
    public partial class CustomersTableAdapter
    {
        // Add user code here. For example:
        public override string ToString()
        {
            return "Overridden in the partial class.";
        }
    }
}
```

```
Namespace NorthwindDataSetTableAdapters

    Partial Class CustomersTableAdapter

        ' Add user code here. For example:
        Public Overrides Function ToString() As String
            Return "Overridden in the partial class."
        End Function
    End Class
End Namespace
```

참조

- TableAdapter를 사용하여 데이터 세트 채우기

XML 데이터를 데이터 세트에 읽어오기

2020-01-06 • 11 minutes to read • Edit Online

ADO.NET는 XML 데이터 작업을 위한 간단한 메서드를 제공 합니다. 이 연습에서는 XML 데이터를 데이터 집합에 로드 하는 Windows 응용 프로그램을 만듭니다. 그러면 데이터 집합이 DataGridView 컨트롤에 표시 됩니다. 마지막으로 XML 파일의 내용을 기반으로 하는 XML 스키마가 텍스트 상자에 표시 됩니다.

새 프로젝트 만들기

또는 Visual Basic에 대 한 새 **Windows Forms 앱** 프로젝트를 만듭니다. C# 프로젝트 이름을 **ReadingXML**로 합니다.

데이터 집합으로 읽어올 XML 파일 생성

이 연습에서는 XML 데이터를 데이터 집합으로 읽는 방법을 중점적으로 설명 하므로 XML 파일의 내용이 제공 됩니다.

1. **프로젝트** 메뉴에서 **새 항목 추가**를 선택합니다.
2. **Xml 파일**을 선택 하 고 파일 이름을 **authors .xml**로 지정한 다음 **추가**를 선택 합니다.

   XML 파일이 디자이너로 로드 되 고 편집할 준비가 됩니다.
3. XML 선언 아래에 다음 XML 데이터를 편집기에 붙여 넣습니다.

```
<Authors_Table>
  <authors>
    <au_id>172-32-1176</au_id>
    <au_lname>White</au_lname>
    <au_fname>Johnson</au_fname>
    <phone>408 496-7223</phone>
    <address>10932 Bigge Rd.</address>
    <city>Menlo Park</city>
    <state>CA</state>
    <zip>94025</zip>
    <contract>true</contract>
  </authors>
  <authors>
    <au_id>213-46-8915</au_id>
    <au_lname>Green</au_lname>
    <au_fname>Margie</au_fname>
    <phone>415 986-7020</phone>
    <address>309 63rd St. #411</address>
    <city>Oakland</city>
    <state>CA</state>
    <zip>94618</zip>
    <contract>true</contract>
  </authors>
  <authors>
    <au_id>238-95-7766</au_id>
    <au_lname>Carson</au_lname>
    <au_fname>Cheryl</au_fname>
    <phone>415 548-7723</phone>
    <address>589 Darwin Ln.</address>
    <city>Berkeley</city>
    <state>CA</state>
    <zip>94705</zip>
    <contract>true</contract>
  </authors>
  <authors>
    <au_id>267-41-2394</au_id>
    <au_lname>Hunter</au_lname>
    <au_fname>Anne</au_fname>
    <phone>408 286-2428</phone>
    <address>22 Cleveland Av. #14</address>
    <city>San Jose</city>
    <state>CA</state>
    <zip>95128</zip>
    <contract>true</contract>
  </authors>
  <authors>
    <au_id>274-80-9391</au_id>
    <au_lname>Straight</au_lname>
    <au_fname>Dean</au_fname>
    <phone>415 834-2919</phone>
    <address>5420 College Av.</address>
    <city>Oakland</city>
    <state>CA</state>
    <zip>94609</zip>
    <contract>true</contract>
  </authors>
</Authors_Table>
```

4. **파일** 메뉴에서 **authors 저장**을 선택 합니다.

사용자 인터페이스 만들기

이 응용 프로그램에 대 한 사용자 인터페이스는 다음과 같이 구성 됩니다.

- XML 파일의 내용을 데이터로 표시 하는 DataGridView 컨트롤입니다.

- XML 파일에 대 한 XML 스키마를 표시 하는 TextBox 컨트롤입니다.
- 두 Button 컨트롤.
  - 한 단추는 XML 파일을 데이터 집합으로 읽어 DataGridView 컨트롤에 표시 합니다.
  - 두 번째 단추는 데이터 집합에서 스키마를 추출 하 고 StringWriter를 통해 해당 스키마를 TextBox 컨트롤에 표시 합니다.

컨트롤을 폼에 추가하려면

1. 디자인 뷰에서 `Form1`를 엽니다.
2. **도구 상자**에서 다음 컨트롤을 폼으로 끌어옵니다.
   - 한 DataGridView 컨트롤
   - 한 TextBox 컨트롤
   - 두 Button 컨트롤
3. 다음 속성을 설정합니다.

CONTROL	속성	설정
`TextBox1`	**Multiline**	`true`
	ScrollBars	**세로**
`Button1`	**Name**	`ReadXmlButton`
	Text	`Read XML`
`Button2`	**Name**	`ShowSchemaButton`
	Text	`Show Schema`

XML 데이터를 수신 하는 데이터 집합 만들기

이 단계에서는 `authors`라는 새 데이터 집합을 만듭니다. 데이터 집합에 대 한 자세한 내용은 Visual Studio의 데이터 집합 도구를 참조 하세요.

1. **솔루션 탐색기**에서 **Form1**의 원본 파일을 선택한 다음 **솔루션 탐색기** 도구 모음에서 **디자이너 보기** 단추를 선택 합니다.
2. 도구 상자, 데이터 탭에서 **데이터 집합** 을 **Form1**로 끌어 옵니다.
3. **데이터 집합 추가** 대화 상자에서 **형식화 되지 않은 데이터 집합**을 선택 하 고 **확인**을 선택 합니다.

   DataSet1 가 구성 요소 트레이에 추가 됩니다.
4. **속성** 창에서 `AuthorsDataSet`의 **이름** 및 DataSetName 속성을 설정 합니다.

XML 파일을 데이터 집합으로 읽도록 이벤트 처리기를 만듭니다.

Xml 읽기 단추를 클릭 하면 xml 파일이 데이터 집합으로 읽혀집니다. 그런 다음 데이터 집합에 바인딩하는 DataGridView 컨트롤에 대 한 속성을 설정 합니다.

1. **솔루션 탐색기**에서 **Form1**을 선택 하 고 **솔루션 탐색기** 도구 모음에서 **디자이너 보기** 단추를 선택 합니

다.

2. **XML 읽기** 단추를 선택 합니다.

   `ReadXmlButton_Click` 이벤트 처리기에서 **코드 편집기** 가 열립니다.

3. `ReadXmlButton_Click` 이벤트 처리기에 다음 코드를 입력 합니다.

```
private void ReadXmlButton_Click(object sender, EventArgs e)
{
    string filePath = "Complete path where you saved the XML file";

    AuthorsDataSet.ReadXml(filePath);

    dataGridView1.DataSource = AuthorsDataSet;
    dataGridView1.DataMember = "authors";
}
```

```
Private Sub ReadXmlButton_Click() Handles ReadXmlButton.Click

    Dim filePath As String = "Complete path where you saved the XML file"

    AuthorsDataSet.ReadXml(filePath)

    DataGridView1.DataSource = AuthorsDataSet
    DataGridView1.DataMember = "authors"
End Sub
```

4. `ReadXMLButton_Click` 이벤트 처리기 코드에서 `filepath =` 항목을 올바른 경로로 변경 합니다.

텍스트 상자에 스키마를 표시 하는 이벤트 처리기를 만듭니다.

스키마 표시 단추를 클릭 하면 스키마로 채워지고 TextBox컨트롤에 표시 되는 StringWriter 개체가 만들어집니다.

1. **솔루션 탐색기**에서 **Form1**을 선택 하 고 **디자이너 보기** 단추를 선택 합니다.

2. **스키마 표시** 단추를 선택 합니다.

   `ShowSchemaButton_Click` 이벤트 처리기에서 **코드 편집기** 가 열립니다.

3. `ShowSchemaButton_Click` 이벤트 처리기에 다음 코드를 붙여넣습니다.

```
private void ShowSchemaButton_Click(object sender, EventArgs e)
{
    System.IO.StringWriter swXML = new System.IO.StringWriter();
    AuthorsDataSet.WriteXmlSchema(swXML);
    textBox1.Text = swXML.ToString();
}
```

```
Private Sub ShowSchemaButton_Click() Handles ShowSchemaButton.Click

    Dim swXML As New System.IO.StringWriter()
    AuthorsDataSet.WriteXmlSchema(swXML)
    TextBox1.Text = swXML.ToString
End Sub
```

양식 테스트

이제 폼을 테스트 하 여 예상 대로 동작 하는지 확인할 수 있습니다.

1. **F5 키** 를 선택 하 여 응용 프로그램을 실행 합니다.
2. **XML 읽기** 단추를 선택 합니다.

   DataGridView는 XML 파일의 내용을 표시 합니다.
3. **스키마 표시** 단추를 선택 합니다.

   텍스트 상자에 XML 파일에 대 한 XML 스키마가 표시 됩니다.

다음 단계

이 연습에서는 xml 파일을 데이터 집합으로 읽는 방법 뿐만 아니라 XML 파일의 내용에 따라 스키마를 만들 수 있는 기본 사항을 배웁니다. 다음 작업을 수행할 수 있는 몇 가지 작업은 다음과 같습니다.

- 데이터 집합의 데이터를 편집 하 고 XML로 다시 작성 합니다. 자세한 내용은 WriteXml를 참조하세요.
- 데이터 집합의 데이터를 편집 하 고 데이터베이스에 기록 합니다.

참조

- Visual Studio에서 데이터 액세스
- Visual Studio의 XML 도구

데이터 세트의 데이터 편집

2020-01-06 • 13 minutes to read • Edit Online

모든 데이터베이스의 테이블에서 데이터를 편집 하는 것 처럼 데이터 테이블의 데이터를 편집 합니다. 이 프로세스에는 테이블의 레코드 삽입, 업데이트 및 삭제가 포함 될 수 있습니다. 데이터 바인딩된 폼에서 사용자가 편집할 수 있는 필드를 지정할 수 있습니다. 이러한 경우 데이터 바인딩 인프라는 모든 변경 내용 추적을 처리 하 여 나중에 변경 내용을 데이터베이스로 다시 전송할 수 있도록 합니다. 프로그래밍 방식으로 데이터를 편집 하 고 해당 변경 내용을 데이터베이스에 다시 보내려면 변경 내용 추적을 수행 하는 개체 및 메서드를 사용 해야 합니다.

실제 데이터를 변경 하는 것 외에도 DataTable을 쿼리하여 데이터의 특정 행을 반환할 수도 있습니다. 예를 들어 개별 행, 특정 버전의 행 (원래 및 제안), 변경 된 행 또는 오류가 있는 행을 쿼리할 수 있습니다.

데이터 집합의 행을 편집 하려면

DataTable에서 기존 행을 편집 하려면 편집 하려는 DataRow를 찾은 다음 업데이트 된 값을 원하는 열에 할당 해야 합니다.

편집 하려는 행의 인덱스를 모르는 경우 `FindBy` 메서드를 사용 하 여 기본 키로 검색 합니다.

```
NorthwindDataSet.CustomersRow customersRow =
    northwindDataSet1.Customers.FindByCustomerID("ALFKI");

customersRow.CompanyName = "Updated Company Name";
customersRow.City = "Seattle";;
```

```
Dim customersRow As NorthwindDataSet.CustomersRow
customersRow = NorthwindDataSet1.Customers.FindByCustomerID("ALFKI")

customersRow.CompanyName = "Updated Company Name"
customersRow.City = "Seattle"
```

행 인덱스를 알고 있으면 다음과 같이 행에 액세스 하 여 행을 편집할 수 있습니다.

```
northwindDataSet1.Customers[4].CompanyName = "Updated Company Name";
northwindDataSet1.Customers[4].City = "Seattle";
```

```
NorthwindDataSet1.Customers(4).CompanyName = "Updated Company Name"
NorthwindDataSet1.Customers(4).City = "Seattle"
```

데이터 집합에 새 행을 삽입 하려면

데이터 바인딩된 컨트롤을 사용 하는 응용 프로그램은 일반적으로 BindingNavigator 컨트롤의 **새로 추가** 단추를 통해 새 레코드를 추가 합니다.

데이터 집합에 새 레코드를 수동으로 추가 하려면 DataTable에서 메서드를 호출 하 여 새 데이터 행을 만듭니다. 그런 다음 DataTable의 DataRow 컬렉션 (Rows)에 행을 추가 합니다.

```
NorthwindDataSet.CustomersRow newCustomersRow =
    northwindDataSet1.Customers.NewCustomersRow();

newCustomersRow.CustomerID = "ALFKI";
newCustomersRow.CompanyName = "Alfreds Futterkiste";

northwindDataSet1.Customers.Rows.Add(newCustomersRow);
```

```
Dim newCustomersRow As NorthwindDataSet.CustomersRow
newCustomersRow = NorthwindDataSet1.Customers.NewCustomersRow()

newCustomersRow.CustomerID = "ALFKI"
newCustomersRow.CompanyName = "Alfreds Futterkiste"

NorthwindDataSet1.Customers.Rows.Add(newCustomersRow)
```

데이터 집합에서 데이터 원본에 업데이트를 전송 하는 데 필요한 정보를 유지 하려면 Delete 메서드를 사용 하 여 데이터 테이블에서 행을 제거 합니다. 예를 들어 응용 프로그램에서 TableAdapter (또는 DataAdapter)를 사용 하는 경우 TableAdapter의 `Update` 메서드는 데이터베이스에서 DeletedRowState 있는 행을 삭제 합니다.

응용 프로그램이 업데이트를 다시 데이터 원본에 보낼 필요가 없는 경우 데이터 행 컬렉션 (Remove)에 직접 액세스 하 여 레코드를 제거할 수 있습니다.

데이터 테이블에서 레코드를 삭제 하려면

- DataRow의 Delete 메서드를 호출 합니다.

  이 메서드는 레코드를 물리적으로 제거 하지 않습니다. 대신 삭제를 위해 레코드를 표시 합니다.

  > **NOTE**
  > DataRowCollection의 count 속성을 가져오는 경우 결과 개수는 삭제 하도록 표시 된 레코드를 포함 합니다. 삭제 하도록 표시 되지 않은 레코드의 정확한 수를 가져오려면 각 레코드의 RowState 속성을 확인 하는 컬렉션을 반복할 수 있습니다. 삭제 하도록 표시 된 레코드에는 DeletedRowState 있습니다. 또는 행 상태를 기준으로 필터링 하는 데이터 집합의 데이터 뷰를 만들고 여기에서 count 속성을 가져올 수 있습니다.

다음 예에서는 Delete 메서드를 호출 하 여 `Customers` 테이블의 첫 번째 행을 삭제 된 것으로 표시 하는 방법을 보여 줍니다.

```
northwindDataSet1.Customers.Rows[0].Delete();
```

```
NorthwindDataSet1.Customers.Rows(0).Delete()
```

변경 된 행이 있는지 확인

데이터 집합의 레코드가 변경 될 때 해당 변경 내용에 대 한 정보는 커밋할 때까지 저장 됩니다. 데이터 집합 또는 데이터 테이블의 `AcceptChanges` 메서드를 호출 하거나 TableAdapter 또는 데이터 어댑터의 `Update` 메서드를 호출 할 때 변경 내용을 커밋합니다.

변경 내용은 각 데이터 행에서 다음 두 가지 방법으로 추적 됩니다.

- 각 데이터 행은 해당 RowState 관련 된 정보 (예: Added, Modified, Deleted또는 Unchanged)를 포함 합니다.
- 변경 된 각 데이터 행에는 해당 행의 여러 버전 (DataRowVersion), 원래 버전 (변경 전) 및 현재 버전 (변경 후)이 포함 됩니다. 변경이 보류 중인 기간 (RowChanging 이벤트에 응답할 수 있는 시간) 동안 세 번째 버전 (제

안 된 버전)도 사용할 수 있습니다.

데이터 집합의 HasChanges 메서드는 데이터 집합에 변경 내용이 있을 경우 `true` 을 반환 합니다. 변경 된 행이 존재 하는지 확인 한 후 DataSet 또는 DataTable의 `GetChanges` 메서드를 호출 하 여 변경 된 행 집합을 반환할 수 있습니다.

행이 변경 되었는지 확인 하려면

- 데이터 집합의 HasChanges 메서드를 호출 하 여 변경 된 행을 확인 합니다.

다음 예에서는 HasChanges 메서드에서 반환 값을 확인 하 여 `NorthwindDataset1` 라는 데이터 집합에 변경 된 행이 있는지 여부를 확인 하는 방법을 보여 줍니다.

```
if (northwindDataSet1.HasChanges())
{
    // Changed rows were detected, add appropriate code.
}
else
{
    // No changed rows were detected, add appropriate code.
}
```

```
If NorthwindDataSet1.HasChanges() Then

    ' Changed rows were detected, add appropriate code.
Else
    ' No changed rows were detected, add appropriate code.
End If
```

변경 유형 확인

DataRowState 열거형의 값을 HasChanges 메서드로 전달 하 여 데이터 집합에서 수행 된 변경 내용 유형을 확인할 수도 있습니다.

행에 대 한 변경 유형을 확인 하려면

- HasChanges 메서드에 DataRowState 값을 전달 합니다.

다음 예제에서는 `NorthwindDataset1` 라는 데이터 집합을 확인 하 여 새 행이 추가 되었는지 확인 하는 방법을 보여 줍니다.

```
if (northwindDataSet1.HasChanges(DataRowState.Added))
{
    // New rows have been added to the dataset, add appropriate code.
}
else
{
    // No new rows have been added to the dataset, add appropriate code.
}
```

```
If NorthwindDataSet1.HasChanges(DataRowState.Added) Then

    ' New rows have been added to the dataset, add appropriate code.
Else
    ' No new rows have been added to the dataset, add appropriate code.
End If
```

오류가 있는 행을 찾으려면

개별 열과 데이터 행을 사용할 때 오류가 발생할 수 있습니다. `HasErrors` 속성을 확인 하 여 오류가 DataSet, DataTable또는 DataRow에 있는지 확인할 수 있습니다.

1. `HasErrors` 속성을 확인 하 여 데이터 집합에 오류가 있는지 확인 합니다.
2. `HasErrors` 속성이 `true`인 경우 테이블의 컬렉션을 반복 하 고 행을 통해를 반복 하 여 오류가 있는 행을 찾습니다.

```
private void FindErrors()
{
    if (dataSet1.HasErrors)
    {
        foreach (DataTable table in dataSet1.Tables)
        {
            if (table.HasErrors)
            {
                foreach (DataRow row in table.Rows)
                {
                    if (row.HasErrors)
                    {
                        // Process error here.
                    }
                }
            }
        }
    }
}
```

```
Private Sub FindErrors()
    Dim table As Data.DataTable
    Dim row As Data.DataRow

    If DataSet1.HasErrors Then

        For Each table In DataSet1.Tables
            If table.HasErrors Then

                For Each row In table.Rows
                    If row.HasErrors Then

                        ' Process error here.
                    End If
                Next
            End If
        Next
    End If
End Sub
```

참조

- Visual Studio의 데이터 세트 도구

데이터 세트의 데이터 유효성 검사

2020-01-06 • 30 minutes to read • Edit Online

데이터 유효성 검사는 데이터 개체에 입력 된 값이 데이터 집합의 스키마 내에서 제약 조건을 준수 하는지 확인 하는 프로세스입니다. 또한 유효성 검사 프로세스는 이러한 값이 응용 프로그램에 대해 설정 된 규칙을 따라 수행 되는지 확인 합니다. 기본 데이터베이스에 업데이트를 보내기 전에 데이터의 유효성을 검사 하는 것이 좋습니다. 이렇게 하면 응용 프로그램과 데이터베이스 간의 잠재적 왕복 수 뿐만 아니라 오류가 줄어듭니다.

데이터 집합 자체에 유효성 검사를 작성 하 여 데이터 집합에 기록 되 고 있는 데이터가 유효한 지 확인할 수 있습니다. 데이터 집합은 업데이트가 수행 되는 방식에 관계 없이 (폼의 컨트롤에서 직접 또는 구성 요소 내에서) 또는 다른 방식으로 데이터를 확인할 수 있습니다. 데이터 집합은 데이터베이스 백 엔드와 달리 응용 프로그램의 일부 이기 때문에 응용 프로그램별 유효성 검사를 빌드하는 논리적 장소입니다.

응용 프로그램에 유효성 검사를 추가 하는 가장 좋은 장소는 데이터 집합의 partial 클래스 파일에 있습니다. Visual Basic 또는 Visual C#에서 **데이터 세트 디자이너** 을 열고 유효성 검사를 만들 열 또는 테이블을 두 번 클릭 합니다. 이 작업은 ColumnChanging 또는 RowChanging 이벤트 처리기를 자동으로 만듭니다.

데이터의 유효성 검사

데이터 집합 내에서 유효성 검사는 다음과 같은 방법으로 수행 됩니다.

- 변경 하는 동안 개별 데이터 열의 값을 검사할 수 있는 응용 프로그램별 유효성 검사를 만듭니다. 자세한 내용은 방법: 열 변경 중 데이터 유효성 검사를 참조 하세요.
- 전체 데이터 행을 변경 하는 동안 데이터를 확인할 수 있는 고유한 응용 프로그램별 유효성 검사를 만듭니다. 자세한 내용은 방법: 행을 변경 하는 동안 데이터 유효성 검사를 참조 하세요.
- 데이터 집합의 실제 스키마 정의의 일부로 키, unique 제약 조건 등을 만듭니다.
- MaxLength, AllowDBNull및 Unique과 같은 DataColumn 개체의 속성을 설정 합니다.

레코드에서 변경이 발생 하는 경우 DataTable 개체에 의해 발생 하는 몇 가지 이벤트는 다음과 같습니다.

- ColumnChanging 및 ColumnChanged 이벤트는 개별 열을 변경할 때마다 발생 합니다. ColumnChanging 이벤트는 특정 열의 변경 내용에 대 한 유효성을 검사 하려는 경우에 유용 합니다. 제안 된 변경에 대 한 정보는 이벤트와 함께 인수로 전달 됩니다.
- RowChanging 및 RowChanged 이벤트는 행을 변경한 후에 발생 합니다. RowChanging 이벤트는 더 일반적입니다. 이는 행에서 변경 내용이 발생 하지만 변경 된 열을 알 수 없음을 나타냅니다.

기본적으로 열을 변경할 때마다 4 개의 이벤트가 발생 합니다. 첫 번째는 변경 중인 특정 열에 대 한 ColumnChanging 및 ColumnChanged 이벤트입니다. 다음은 RowChanging 및 RowChanged 이벤트입니다. 행이 여러 번 변경 되 면 각 변경에 대해 이벤트가 발생 합니다.

> **NOTE**
> 데이터 행의 BeginEdit 메서드는 개별 열이 변경 된 후 RowChanging 및 RowChanged 이벤트를 해제 합니다. 이 경우 RowChanging 및 RowChanged 이벤트가 한 번만 발생 하면 EndEdit 메서드가 호출 될 때까지 이벤트가 발생 하지 않습니다. 자세한 내용은 데이터 집합을 채우는 동안 제약 조건해제를 참조 하세요.

선택 하는 이벤트는 유효성 검사를 수행할 세부적인 방법에 따라 달라 집니다. 열이 변경 될 때 즉시 오류를 catch 하는 것이 중요 한 경우 ColumnChanging 이벤트를 사용 하 여 유효성 검사를 빌드합니다. 그렇지 않으면 RowChanging 이벤트를 사용 하 여 한 번에 여러 오류를 catch 할 수 있습니다. 또한 데이터를 구조화 하 여 한

열의 값이 다른 열의 내용에 따라 유효성을 검사 하도록 하려면 RowChanging 이벤트 중에 유효성 검사를 수행 합니다.

레코드가 업데이트 되 면 변경 내용이 발생 하 고 변경 내용이 적용 된 후에 응답할 수 있는 이벤트가 DataTable 개체에 발생 합니다.

응용 프로그램에서 형식화 된 데이터 집합을 사용 하는 경우 강력한 형식의 이벤트 처리기를 만들 수 있습니다. 그러면 처리기를 만들 수 있는 네 가지 형식화 된 이벤트 `dataTableNameRowChanging`, `dataTableNameRowChanged`, `dataTableNameRowDeleting` 및 `dataTableNameRowDeleted` 추가 됩니다. 이러한 형식화 된 이벤트 처리기는 코드를 더 쉽게 작성 하 고 읽을 수 있도록 테이블의 열 이름이 포함 된 인수를 전달 합니다.

데이터 업데이트 이벤트

EVENT	설명
ColumnChanging	열의 값이 변경 되 고 있습니다. 이 이벤트는 제안 된 새 값과 함께 행과 열을 사용자에 게 전달 합니다.
ColumnChanged	열의 값이 변경 되었습니다. 이벤트는 제안 된 값과 함께 행과 열을 사용자에 게 전달 합니다.
RowChanging	DataRow 개체에 대 한 변경 내용은 데이터 집합으로 다시 커밋됩니다. BeginEdit 메서드를 호출 하지 않은 경우 ColumnChanging 이벤트가 발생 한 직후에 열을 변경할 때마다 RowChanging 이벤트가 발생 합니다. 변경을 수행 하기 전에 BeginEdit를 호출한 경우 EndEdit 메서드를 호출 하는 경우에만 RowChanging 이벤트가 발생 합니다. 이벤트는 수행 중인 작업 유형 (변경, 삽입 등)을 나타내는 값과 함께 사용자에 게 행을 전달 합니다.
RowChanged	행이 변경 되었습니다. 이벤트는 수행 중인 작업 유형 (변경, 삽입 등)을 나타내는 값과 함께 사용자에 게 행을 전달 합니다.
RowDeleting	행을 삭제 하 고 있습니다. 이벤트는 수행 중인 작업 유형 (delete)을 나타내는 값과 함께 행을 사용자에 게 전달 합니다.
RowDeleted	행이 삭제 되었습니다. 이벤트는 수행 중인 작업 유형 (delete)을 나타내는 값과 함께 행을 사용자에 게 전달 합니다.

ColumnChanging, RowChanging 및 RowDeleting 이벤트는 업데이트 프로세스 중에 발생 합니다. 이러한 이벤트를 사용 하 여 데이터의 유효성을 검사 하거나 다른 유형의 처리를 수행할 수 있습니다. 이러한 이벤트 중에 업데이트가 진행 되 고 있으므로 예외를 throw 하 여 업데이트를 취소할 수 있습니다. 이렇게 하면 업데이트를 완료할 수 없습니다.

ColumnChanged, RowChanged 및 RowDeleted 이벤트는 업데이트가 성공적으로 완료 될 때 발생 하는 알림 이벤트입니다. 이러한 이벤트는 성공적인 업데이트를 기반으로 하 여 추가 작업을 수행 하려는 경우에 유용 합니다.

열 변경 중 데이터 유효성 검사

> **NOTE**
> **데이터 세트 디자이너** 유효성 검사 논리를 데이터 집합에 추가할 수 있는 partial 클래스를 만듭니다. 디자이너에서 생성 된 데이터 집합은 partial 클래스의 코드를 삭제 하거나 변경 하지 않습니다.

ColumnChanging 이벤트에 응답 하 여 데이터 열의 값이 변경 되 면 데이터의 유효성을 검사할 수 있습니다. 이 이벤트는 발생 하면 현재 열에 대해 제안 되는 값을 포함 하는 이벤트 인수 (ProposedValue)를 전달 합니다. `e.ProposedValue`의 내용에 따라 다음을 수행할 수 있습니다.

- 아무 작업도 수행 하지 않고 제안 된 값을 적용 합니다.
- 열 변경 이벤트 처리기 내에서 열 오류 (SetColumnError)를 설정 하 여 제안 된 값을 거부 합니다.
- 필요에 따라 ErrorProvider 컨트롤을 사용 하 여 사용자에 게 오류 메시지를 표시 합니다. 자세한 내용은 ErrorProvider 구성 요소를 참조하세요.

RowChanging 이벤트 중에 유효성 검사를 수행할 수도 있습니다.

행을 변경 하는 동안 데이터 유효성 검사

유효성을 검사 하려는 각 열에 응용 프로그램의 요구 사항을 충족 하는 데이터가 포함 되어 있는지 확인 하는 코드를 작성할 수 있습니다. 이렇게 하려면 제안 된 값이 허용 되지 않는 경우 오류가 포함 되어 있음을 나타내도록 열을 설정 합니다. 다음 예에서는 `Quantity` 열이 0 이하인 경우 열 오류를 설정 합니다. 행 변경 이벤트 처리기는 다음 예제와 유사 합니다.

행이 변경 될 때 데이터의 유효성을 검사 하려면 **(Visual Basic)**

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.
2. 유효성을 검사할 테이블의 제목 표시줄을 두 번 클릭 합니다. 이 작업을 수행 하면 데이터 집합의 partial 클래스 파일에 DataTable의 RowChanging 이벤트 처리기가 자동으로 만들어집니다.

   > **TIP**
   > 테이블 이름 왼쪽을 두 번 클릭 하 여 행 변경 이벤트 처리기를 만듭니다. 테이블 이름을 두 번 클릭 하면 편집할 수 있습니다.

```
Private Sub Order_DetailsDataTable_Order_DetailsRowChanging(
    ByVal sender As System.Object,
    ByVal e As Order_DetailsRowChangeEvent
  ) Handles Me.Order_DetailsRowChanging

    If CType(e.Row.Quantity, Short) <= 0 Then
        e.Row.SetColumnError("Quantity", "Quantity must be greater than 0")
    Else
        e.Row.SetColumnError("Quantity", "")
    End If
End Sub
```

행이 변경 될 때 데이터의 유효성**C#**을 검사 하려면 **()**

1. **데이터 세트 디자이너**에서 데이터 세트를 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.
2. 유효성을 검사할 테이블의 제목 표시줄을 두 번 클릭 합니다. 이 작업은 DataTable에 대 한 부분 클래스 파일을 만듭니다.

> **NOTE**
> **데이터 세트 디자이너** 는 RowChanging 이벤트에 대 한 이벤트 처리기를 자동으로 만들지 않습니다. RowChanging 이벤트를 처리 하는 메서드를 만들고 코드를 실행 하 여 테이블의 초기화 메서드에서 이벤트를 연결 해야 합니다.

3. Partial 클래스에 다음 코드를 복사 합니다.

```
public override void EndInit()
{
    base.EndInit();
    Order_DetailsRowChanging += TestRowChangeEvent;
}

public void TestRowChangeEvent(object sender, Order_DetailsRowChangeEvent e)
{
    if ((short)e.Row.Quantity <= 0)
    {
        e.Row.SetColumnError("Quantity", "Quantity must be greater than 0");
    }
    else
    {
        e.Row.SetColumnError("Quantity", "");
    }
}
```

변경 된 행을 검색 하려면

데이터 테이블의 각 행에는 DataRowState 열거형의 값을 사용 하 여 해당 행의 현재 상태를 추적 하는 RowState 속성이 있습니다. DataSet 또는 DataTable의 `GetChanges` 메서드를 호출 하 여 데이터 집합 또는 데이터 테이블에서 변경 된 행을 반환할 수 있습니다. 데이터 집합의 HasChanges 메서드를 호출 하 여 `GetChanges` 를 호출 하기 전에 변경 내용이 존재 하는지 확인할 수 있습니다.

> **NOTE**
> AcceptChanges 메서드를 호출 하 여 데이터 집합 또는 데이터 테이블에 대 한 변경 내용을 커밋한 후 `GetChanges` 메서드는 데이터를 반환 하지 않습니다. 응용 프로그램에서 변경 된 행을 처리 해야 하는 경우 `AcceptChanges` 메서드를 호출 하기 전에 변경 내용을 처리 해야 합니다.

데이터 집합 또는 데이터 테이블의 GetChanges 메서드를 호출 하면 변경 된 레코드만 포함 된 새 데이터 집합 또는 데이터 테이블이 반환 됩니다. 특정 레코드 (예: 새 레코드 또는 수정 된 레코드만)를 가져오려는 경우 DataRowState 열거형의 값을 `GetChanges` 메서드에 매개 변수로 전달할 수 있습니다.

DataRowVersion 열거를 사용 하 여 행을 처리 하기 전에 행에 있던 원래 값과 같은 다른 버전의 행에 액세스할 수 있습니다.

데이터 집합에서 변경 된 모든 레코드를 가져오려면

- 데이터 집합의 GetChanges 메서드를 호출 합니다.

  다음 예에서는 `changedRecords` 라는 새 데이터 집합을 만들고 `dataSet1` 라는 다른 데이터 집합의 모든 변경 된 레코드로 채웁니다.

```
DataSet changedRecords = dataSet1.GetChanges();
```

```
Dim changedRecords As DataSet = DataSet1.GetChanges()
```

데이터 테이블에서 변경 된 모든 레코드를 가져오려면

- DataTable의 GetChanges 메서드를 호출 합니다.

  다음 예에서는 `changedRecordsTable` 라는 새 데이터 테이블을 만들고 `dataTable1` 라는 다른 데이터 테이블의 모든 변경 된 레코드로 채웁니다.

```
DataTable changedRecordsTable = dataTable1.GetChanges();
```

```
Dim changedRecordsTable As DataTable = dataTable1.GetChanges()
```

특정 행 상태를 포함 하는 모든 레코드를 가져오려면

- 데이터 집합 또는 데이터 테이블의 `GetChanges` 메서드를 호출 하 고 DataRowState 열거형 값을 인수로 전달 합니다.

  다음 예제에서는 `addedRecords` 라는 새 데이터 집합을 만들고 `dataSet1` 데이터 집합에 추가 된 레코드를 사용 하 여이를 채우는 방법을 보여 줍니다.

```
DataSet addedRecords = dataSet1.GetChanges(DataRowState.Added);
```

```
Dim addedRecords As DataSet = DataSet1.GetChanges(DataRowState.Added)
```

  다음 예에서는 `Customers` 테이블에 최근에 추가 된 모든 레코드를 반환 하는 방법을 보여 줍니다.

```
private NorthwindDataSet.CustomersDataTable GetNewRecords()
{
    return (NorthwindDataSet.CustomersDataTable)
        northwindDataSet1.Customers.GetChanges(DataRowState.Added);
}
```

```
Private Function GetNewRecords() As NorthwindDataSet.CustomersDataTable

    Return CType(NorthwindDataSet1.Customers.GetChanges(Data.DataRowState.Added),
        NorthwindDataSet.CustomersDataTable)
End Function
```

DataRow의 원래 버전에 액세스

데이터 행이 변경 되 면 데이터 집합은 행의 원래 (Original) 버전과 새 (Current) 버전을 모두 유지 합니다. 예를 들어 `AcceptChanges` 메서드를 호출 하기 전에 응용 프로그램은 DataRowVersion 열거형에 정의 된 것과 같은 다른 버전의 레코드에 액세스 하 고 그에 따라 변경 내용을 처리할 수 있습니다.

> **NOTE**
> 서로 다른 버전의 행은 편집 된 후 `AcceptChanges` 메서드가 호출 되기 전에만 존재 합니다. `AcceptChanges` 메서드를 호출한 후에는 현재 버전과 원래 버전이 동일 합니다.

열 인덱스 (또는 열 이름)와 함께 DataRowVersion 값을 전달 하면 해당 열의 특정 행 버전에서 값이 반환 됩니다.

변경 된 열은 ColumnChanging 및 ColumnChanged 이벤트 중에 식별 됩니다. 유효성 검사를 위해 다른 행 버전을 검사 하는 것이 좋습니다. 그러나 일시적으로 제약 조건을 일시 중단 한 경우 이러한 이벤트는 발생 하지 않으며 변경 된 열을 프로그래밍 방식으로 식별 해야 합니다. Columns 컬렉션을 반복 하 고 다른 DataRowVersion 값을 비교 하 여이 작업을 수행할 수 있습니다.

레코드의 원래 버전을 가져오려면

- 반환할 행의 DataRowVersion를 전달 하 여 열의 값에 액세스 합니다.

  다음 예에서는 DataRowVersion 값을 사용 하 여 DataRow에서 `CompanyName` 필드의 원래 값을 가져오는 방법을 보여 줍니다.

```
string originalCompanyName;
originalCompanyName = northwindDataSet1.Customers[0]
    ["CompanyName", DataRowVersion.Original].ToString();
```

```
Dim originalCompanyName = NorthwindDataSet1.Customers(0)(
   "CompanyName", DataRowVersion.Original).ToString()
```

DataRow의 현재 버전에 액세스 합니다.

레코드의 현재 버전을 가져오려면

- 열의 값에 액세스 한 다음 반환할 행의 버전을 나타내는 매개 변수를 인덱스에 추가 합니다.

  다음 예에서는 DataRowVersion 값을 사용 하 여 DataRow에서 `CompanyName` 필드의 현재 값을 가져오는 방법을 보여 줍니다.

```
string currentCompanyName;
currentCompanyName = northwindDataSet1.Customers[0]
    ["CompanyName", DataRowVersion.Current].ToString();
```

```
Dim currentCompanyName = NorthwindDataSet1.Customers(0)(
    "CompanyName", DataRowVersion.Current).ToString()
```

참조

- Visual Studio의 데이터 세트 도구
- 방법: Windows Forms DataGridView 컨트롤의 데이터 유효성 검사
- 방법: Windows Forms ErrorProvider 구성 요소를 사용하여 폼 유효성에 대한 오류 아이콘 표시

데이터를 다시 데이터베이스에 저장

2020-01-06 • 56 minutes to read • Edit Online

데이터 집합은 데이터의 메모리 내 복사본입니다. 해당 데이터를 수정 하는 경우 해당 변경 내용을 데이터베이스에 다시 저장 하는 것이 좋습니다. 다음 세 가지 방법 중 하나로이 작업을 수행 합니다.

- TableAdapter의 `Update` 메서드 중 하나를 호출 하 여
- TableAdapter의 `DBDirect` 메서드 중 하나를 호출 하 여
- 데이터 집합에 데이터 집합의 다른 테이블과 관련 된 테이블이 포함 되어 있는 경우 Visual Studio가 생성 하는 TableAdapterManager에서 `UpdateAll` 메서드를 호출 합니다.

Windows Form 또는 XAML 페이지의 컨트롤에 데이터 집합 테이블을 바인딩하는 경우 데이터 바인딩 아키텍처에서 모든 작업을 수행 합니다.

Tableadapter에 대해 잘 알고 있는 경우 다음 항목 중 하나로 직접 이동할 수 있습니다.

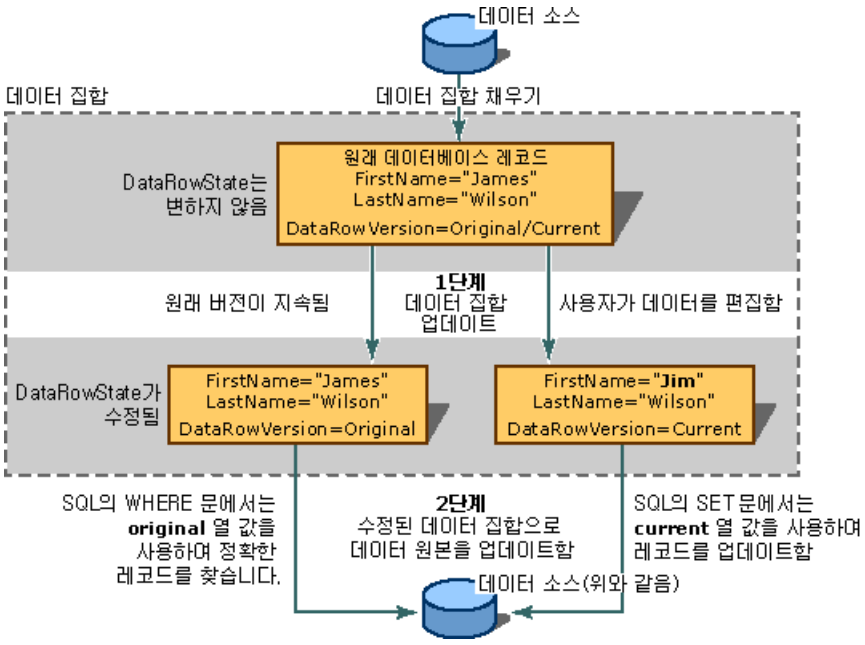
항목	설명
데이터베이스에 새 레코드 삽입	Tableadapter 또는 Command 개체를 사용 하 여 업데이트 및 삽입을 수행 하는 방법
TableAdapter를 사용하여 데이터 업데이트	Tableadapter를 사용 하 여 업데이트를 수행 하는 방법
계층적 업데이트	두 개 이상의 관련 테이블이 포함 된 데이터 집합에서 업데이트를 수행 하는 방법
동시성 예외 처리	두 사용자가 동시에 데이터베이스의 동일한 데이터를 변경 하려고 시도 하는 경우 예외를 처리 하는 방법
방법: 트랜잭션을 사용하여 데이터 저장	시스템을 사용 하 여 트랜잭션에 데이터를 저장 하는 방법 트랜잭션 네임 스페이스 및 TransactionScope 개체
트랜잭션에 데이터 저장	트랜잭션 내에서 데이터베이스에 데이터를 저장 하는 방법을 보여 주는 Windows Forms 응용 프로그램을 만드는 연습
데이터베이스에 데이터 저장(여러 테이블)	레코드를 편집 하 고 여러 테이블의 변경 내용을 데이터베이스에 다시 저장 하는 방법
개체에서 데이터베이스로 데이터 저장	TableAdapter DbDirect 메서드를 사용 하 여 데이터 집합에 없는 개체에서 데이터베이스로 데이터를 전달 하는 방법
TableAdapter DBDirect 메서드를 사용하여 데이터 저장	TableAdapter를 사용 하 여 SQL 쿼리를 데이터베이스에 직접 보내는 방법
데이터 세트를 XML로 저장	XML 문서에 데이터 집합을 저장 하는 방법

2 단계 업데이트

데이터 원본 업데이트는 2 단계 프로세스로 진행 됩니다. 첫 번째 단계는 새 레코드, 변경 된 레코드 또는 삭제 된 레코드를 사용 하 여 데이터 집합을 업데이트 하는 것입니다. 응용 프로그램에서 해당 변경 내용을 데이터 원본

으로 다시 보내지 않는 경우 업데이트가 완료 된 것입니다.

변경 내용을 데이터베이스로 다시 보내는 경우 두 번째 단계가 필요 합니다. 데이터 바인딩된 컨트롤을 사용 하지 않는 경우 데이터 집합을 채우는 데 사용한 것과 동일한 TableAdapter (또는 데이터 어댑터)의 `Update` 메서드를 수동으로 호출 해야 합니다. 그러나 다른 어댑터를 사용 하 여 데이터 원본 간에 데이터를 이동 하거나 여러 데이터 원본을 업데이트할 수도 있습니다. 데이터 바인딩을 사용 하지 않고 관련 테이블에 대 한 변경 내용을 저장 하는 경우 자동으로 생성 된 `TableAdapterManager` 클래스의 변수를 수동으로 인스턴스화한 다음 해당 `UpdateAll` 메서드를 호출 해야 합니다.



데이터 집합은 행 컬렉션을 포함 하는 테이블의 컬렉션을 포함 합니다. 나중에 기본 데이터 원본을 업데이트 하려는 경우에는 행을 추가 하거나 제거할 때 `DataTable.DataRowCollection` 속성에서 메서드를 사용 해야 합니다. 이러한 메서드는 데이터 원본을 업데이트 하는 데 필요한 변경 내용 추적을 수행 합니다. Rows 속성에서 `RemoveAt` 컬렉션을 호출 하는 경우 삭제는 데이터베이스에 다시 전달 되지 않습니다.

데이터 집합 병합

데이터 집합을 다른 데이터 집합과 *병합* 하 여 데이터 집합의 내용을 업데이트할 수 있습니다. 여기에는 *원본* 데이터 집합의 내용을 호출 하는 데이터 집합 (*대상* 데이터 집합 이라고 함)에 복사 하는 작업이 포함 됩니다. 데이터 집합을 병합할 때 원본 데이터 집합의 새 레코드가 대상 데이터 집합에 추가 됩니다. 또한 원본 데이터 집합의 추가 열이 대상 데이터 집합에 추가 됩니다. 로컬 데이터 집합이 있고 다른 응용 프로그램에서 두 번째 데이터 집합을 가져오는 경우 데이터 집합을 병합 하는 것이 유용 합니다. 또한 XML web services와 같은 구성 요소에서 두 번째 데이터 집합을 가져오는 경우 또는 여러 데이터 집합의 데이터를 통합 해야 하는 경우에도 유용 합니다.

데이터 집합을 병합할 때 대상 데이터 집합의 기존 수정 내용을 유지할지 여부를 Merge 메서드에 알려 주는 부울 인수 (`preserveChanges`)를 전달할 수 있습니다. 데이터 집합은 여러 버전의 레코드를 유지 하기 때문에 두 개 이상의 레코드가 병합 되 고 있다는 점을 명심 해야 합니다. 다음 표에서는 두 데이터 집합의 레코드를 병합 하는 방법을 보여 줍니다.

DATAROWVERSION	대상 데이터 세트	원본 데이터 세트
원래 색	James Wilson	James C. Wilson
현재	Jim Wilson	James C. Wilson

위의 표에서 `preserveChanges=false targetDataset.Merge(sourceDataset)` 를 사용 하 여 Merge 메서드를 호출 하면 다음 데이터가 반환 됩니다.

DATAROWVERSION	대상 데이터 세트	원본 데이터 세트
원래 색	James C. Wilson	James C. Wilson
현재	James C. Wilson	James C. Wilson

`preserveChanges = true targetDataset.Merge(sourceDataset, true)`를 사용 하 여 Merge 메서드를 호출 하면 다음 데이터가 반환 됩니다.

DATAROWVERSION	대상 데이터 세트	원본 데이터 세트
원래 색	James C. Wilson	James C. Wilson
현재	Jim Wilson	James C. Wilson

Caution

`preserveChanges = true` 시나리오에서 대상 데이터 집합의 레코드에 대해 RejectChanges 메서드를 호출 하면 *원본* 데이터 집합에서 원래 데이터로 되돌아갑니다. 즉, 대상 데이터 집합을 사용 하 여 원래 데이터 원본을 업데이트 하려고 하면 업데이트할 원래 행을 찾지 못할 수 있습니다. 데이터 원본에서 업데이트 된 레코드로 다른 데이터 집합을 채운 다음, 동시성 위반을 방지 하기 위해 병합을 수행 하 여 동시성 위반을 방지할 수 있습니다. 데이터 집합을 채운 후 다른 사용자가 데이터 소스의 레코드를 수정 하면 동시성 위반이 발생 합니다.

제약 조건 업데이트

기존 데이터 행을 변경 하려면 개별 열의 데이터를 추가 하거나 업데이트 합니다. 데이터 집합에 제약 조건 (예: 외래 키 또는 null을 허용 하지 않는 제약 조건)이 포함 된 경우 업데이트할 때 레코드가 일시적으로 오류 상태에 있을 수 있습니다. 즉, 한 열에 대 한 업데이트를 완료 한 후 다음 항목을 가져오기 전에는 오류 상태가 될 수 있습니다.

중간 제약 조건 위반을 방지 하기 위해 업데이트 제약 조건을 일시적으로 중단할 수 있습니다. 이는 두 가지 용도로 사용됩니다.

- 한 열 업데이트를 완료 했지만 다른 열 업데이트를 시작 하지 않은 후 오류가 발생 하지 않도록 방지 합니다.
- 특정 업데이트 이벤트 (종종 유효성 검사에 사용 되는 이벤트)가 발생 하지 않도록 합니다.

> **NOTE**
> Windows Forms datagrid에 기본 제공 되는 데이터 바인딩 아키텍처는 포커스가 행 외부로 이동할 때까지 제약 조건 검사를 일시 중단 하며, BeginEdit, EndEdit또는 CancelEdit 메서드를 명시적으로 호출할 필요가 없습니다.

데이터 집합에서 Merge 메서드가 호출 되 면 제약 조건이 자동으로 비활성화 됩니다. 병합이 완료 되 면 사용 하도록 설정할 수 없는 데이터 집합에 대 한 제약 조건이 있으면 ConstraintException throw 됩니다. 이 경우 EnforceConstraints 속성이 `false,`로 설정 되 고 EnforceConstraints 속성을 `true`로 다시 설정 하기 전에 모든 제약 조건 위반을 확인 해야 합니다.

업데이트를 완료 한 후에도 업데이트 이벤트를 다시 사용 하도록 설정 하 고 발생 시키는 제약 조건 검사를 다시 활성화할 수 있습니다.

이벤트를 일시 중단 하는 방법에 대 한 자세한 내용은 데이터 집합을 채우는 동안 제약 조건해제를 참조 하세요.

데이터 집합 업데이트 오류

데이터 집합의 레코드를 업데이트 하면 오류가 발생할 수 있습니다. 예를 들어 잘못 된 형식의 데이터를 열 또는

너무 긴 데이터 또는 다른 무결성 문제가 있는 데이터에 실수로 쓸 수 있습니다. 또는 업데이트 이벤트의 모든 단계에서 사용자 지정 오류를 발생 시킬 수 있는 응용 프로그램별 유효성 검사가 있을 수 있습니다. 자세한 내용은 데이터 집합의 데이터 유효성 검사를 참조 하세요.

변경 내용에 대 한 정보 유지 관리

데이터 집합의 변경 내용에 대 한 정보는 두 가지 방법으로 유지 됩니다. 즉, 변경 된 내용 (RowState)을 나타내는 행을 플래그 지정 하 고 레코드의 여러 복사본을 유지 합니다 (DataRowVersion). 프로세스는이 정보를 사용 하 여 데이터 집합에서 변경 된 내용을 확인 하 고 데이터 원본에 적절 한 업데이트를 보낼 수 있습니다.

RowState 속성

DataRow 개체의 RowState 속성은 특정 데이터 행의 상태에 대 한 정보를 제공 하는 값입니다.

다음 표에서는 DataRowState 열거형의 가능한 값에 대해 자세히 설명 합니다.

DATAROWSTATE 값	설명
Added	행이 DataRowCollection에 항목으로 추가 되었습니다. 이 상태의 행은 마지막 AcceptChanges 메서드를 호출한 경우 존재 하지 않기 때문에 해당 하는 원래 버전이 없습니다.
Deleted	DataRow 개체의 Delete를 사용 하 여 행을 삭제 했습니다.
Detached	행 생성 되었지만의 일부가 아닌 DataRowCollection합니다. DataRow 개체는 만들어진 직후, 컬렉션에 추가 되기 전에, 그리고 컬렉션에서 제거 된 후에이 상태가 됩니다.
Modified	행의 열 값이 변경 되었습니다.
Unchanged	행이 이후 변경 되지 AcceptChanges 가 마지막으로 호출 합니다.

DataRowVersion 열거형

데이터 집합은 여러 버전의 레코드를 유지 관리 합니다. DataRowVersion 필드는 DataRow 개체의 GetChildRows 메서드 또는 Item[] 속성을 사용 하 여 DataRow에 있는 값을 검색할 때 사용 됩니다.

다음 표에서는 DataRowVersion 열거형의 가능한 값에 대해 자세히 설명 합니다.

DATAROWVERSION 값	설명
Current	최신 버전의 레코드에는 AcceptChanges가 마지막으로 호출 된 이후 레코드에 대해 수행 된 모든 수정 내용이 포함 되어 있습니다. 행이 삭제 된 경우에는 현재 버전이 없습니다.
Default	데이터 집합 스키마 또는 데이터 원본에 의해 정의 된 레코드의 기본값입니다.
Original	레코드의 원래 버전은 데이터 집합에서 변경 내용이 마지막으로 커밋된 시점의 레코드 복사본입니다. 실제로이는 일반적으로 데이터 원본에서 읽은 레코드의 버전입니다.

DATAROWVERSION 값	설명
Proposed	업데이트 중에 일시적으로 사용할 수 있는 (즉, BeginEdit 메서드와 EndEdit 메서드를 호출한 시간 사이에) 임시 버전의 레코드를 사용할 수 있습니다. 일반적으로 RowChanging와 같은 이벤트에 대 한 처리기에서 제안 된 레코드 버전에 액세스 합니다. CancelEdit 메서드를 호출 하면 변경 내용이 취소 되 고 데이터 행의 제안 된 버전이 삭제 됩니다.

원본 및 현재 버전은 업데이트 정보가 데이터 원본으로 전송 되는 경우에 유용 합니다. 일반적으로 데이터 원본에 업데이트를 보낼 때 데이터베이스의 새 정보는 레코드의 현재 버전에 있습니다. 원래 버전의 정보는 업데이트할 레코드를 찾는 데 사용 됩니다.

예를 들어 레코드의 기본 키가 변경 된 경우에는 변경 내용을 업데이트 하기 위해 데이터 원본에서 올바른 레코드를 찾는 방법이 필요 합니다. 원래 버전이 없는 경우 레코드는 데이터 원본에 추가 되어 원치 않는 추가 레코드뿐만 아니라 부정확 하 고 오래 된 레코드 하나에 있을 수 있습니다. 두 버전은 동시성 제어에도 사용 됩니다. 데이터 원본에 있는 레코드와 원래 버전을 비교 하 여 레코드가 데이터 집합에 로드 된 후 변경 되었는지 확인할 수 있습니다.

제안 된 버전은 데이터 집합에 대 한 변경 내용을 실제로 커밋하기 전에 유효성 검사를 수행 해야 하는 경우에 유용 합니다.

레코드가 변경 된 경우에도 해당 행의 원래 버전이 나 현재 버전이 항상 존재 하지는 않습니다. 테이블에 새 행을 삽입 하는 경우 원래 버전이 없으며 현재 버전만 있습니다. 마찬가지로, 테이블의 `Delete` 메서드를 호출 하 여 행을 삭제 하는 경우에는 원래 버전이 있지만 현재 버전은 없습니다.

데이터 행의 HasVersion 메서드를 쿼리하여 레코드의 특정 버전이 존재 하는지 테스트할 수 있습니다. 열 값을 요청할 때 DataRowVersion 열거형 값을 선택적 인수로 전달 하 여 레코드의 두 버전에 액세스할 수 있습니다.

변경 된 레코드 가져오기

일반적으로 데이터 집합의 모든 레코드를 업데이트 하지 않는 것이 좋습니다. 예를 들어 사용자가 많은 레코드를 표시 하는 Windows Forms DataGridView 컨트롤로 작업할 수 있습니다. 그러나 사용자는 몇 개의 레코드만 업데이트 하 고 삭제 한 다음 새 레코드만 삽입할 수 있습니다. 데이터 집합 및 데이터 테이블은 수정 된 행만 반환 하는 메서드 (`GetChanges`)를 제공 합니다.

데이터 테이블 (GetChanges) 또는 데이터 집합 (GetChanges)의 `GetChanges` 메서드를 사용 하 여 변경 된 레코드의 하위 집합을 만들 수 있습니다. 데이터 테이블에 대해 메서드를 호출 하면 변경 된 레코드만 포함 된 테이블의 복사본을 반환 합니다. 마찬가지로, 데이터 집합에서 메서드를 호출 하는 경우 변경 된 레코드만 포함 된 새 데이터 집합을 가져옵니다.

`GetChanges` 자체는 변경 된 모든 레코드를 반환 합니다. 반면, 원하는 DataRowState를 `GetChanges` 메서드에 매개 변수로 전달 하 여 새로 추가 된 레코드, 삭제 하도록 표시 된 레코드, 분리 된 레코드 또는 수정 된 레코드 중에서 변경 된 레코드의 하위 집합을 지정할 수 있습니다.

변경 된 레코드의 하위 집합을 가져오는 작업은 처리를 위해 다른 구성 요소로 레코드를 전송 하려는 경우에 유용 합니다. 전체 데이터 집합을 전송 하는 대신 구성 요소에 필요한 레코드만 가져옴으로써 다른 구성 요소와 통신 하는 오버 헤드를 줄일 수 있습니다.

데이터 집합의 변경 내용 커밋

데이터 집합에서 변경이 수행 되 면 변경 된 행의 RowState 속성이 설정 됩니다. 원본 및 현재 버전의 레코드는 RowVersion 속성으로 설정 되 고 유지 관리 되며 사용할 수 있습니다. 이러한 변경 된 행의 속성에 저장 된 메타데이터는 데이터 원본에 대 한 올바른 업데이트를 전송 하는 데 필요 합니다.

변경 내용이 데이터 원본의 현재 상태를 반영 하는 경우 더 이상이 정보를 유지 관리할 필요가 없습니다. 일반적

으로 데이터 집합 및 해당 소스가 동기화 되는 경우는 두 번입니다.

- 원본에서 데이터를 읽는 경우와 같이 데이터 집합에 정보를 로드 한 직후
- 변경 내용을 데이터베이스에 전송 하는 데 필요한 변경 정보를 잃지 않으므로 데이터 집합에서 데이터 원본으로 변경 내용을 전송한 후 (이전에는 그렇지 않음)

AcceptChanges 메서드를 호출 하 여 데이터 집합에 대 한 보류 중인 변경 내용을 커밋할 수 있습니다. 일반적으로 AcceptChanges는 다음과 같은 시간에 호출 됩니다.

- 데이터 집합을 로드 한 후 TableAdapter의 `Fill` 메서드를 호출 하 여 데이터 집합을 로드 하는 경우 어댑터는 자동으로 변경 내용을 커밋합니다. 그러나 다른 데이터 집합을 병합 하 여 데이터 집합을 로드 하는 경우에는 변경 내용을 수동으로 커밋해야 합니다.

  > **NOTE**
  > 어댑터의 `AcceptChangesDuringFill` 속성을 `false`로 설정 하 여 `Fill` 메서드를 호출할 때 어댑터가 자동으로 변경 내용을 커밋하는 것을 방지할 수 있습니다. `false`로 설정 된 경우 채우기 중에 삽입 된 각 행의 RowState Added로 설정 됩니다.

- XML web services와 같은 다른 프로세스에 데이터 집합 변경 내용을 전송 합니다.

  Caution
  이 방법으로 변경 내용을 커밋하면 변경 정보가 지워집니다. 응용 프로그램이 데이터 집합에서 변경 된 내용을 확인 하는 데 필요한 작업을 완료 한 후에 변경 내용을 커밋하지 않습니다.

이 메서드는 다음을 수행 합니다.

- Current 버전의 레코드를 Original 버전에 쓰고 원래 버전을 덮어씁니다.
- RowState 속성이 Deleted로 설정 된 모든 행을 제거 합니다.
- Unchanged레코드의 RowState 속성을 설정 합니다.

AcceptChanges 메서드는 세 가지 수준에서 사용할 수 있습니다. DataRow 개체에서이 메서드를 호출 하 여 해당 행에 대 한 변경 내용을 커밋할 수 있습니다. 또한 DataTable 개체에서이 메서드를 호출 하 여 테이블의 모든 행을 커밋할 수 있습니다. 마지막으로 DataSet 개체에서이 메서드를 호출 하 여 데이터 집합의 모든 테이블에 있는 모든 레코드의 보류 중인 모든 변경 내용을 커밋할 수 있습니다.

다음 표에서는 메서드가 호출 되는 개체에 따라 커밋된 변경 내용을 설명 합니다.

메서드	결과
System.Data.DataRow.AcceptChanges	특정 행에 대해서만 변경 내용이 커밋됩니다.
System.Data.DataTable.AcceptChanges	특정 테이블의 모든 행에 대 한 변경 내용이 커밋됩니다.
System.Data.DataSet.AcceptChanges	데이터 집합의 모든 테이블에 있는 모든 행에 대 한 변경 내용이 커밋됩니다.

> **NOTE**
> TableAdapter의 `Fill` 메서드를 호출 하 여 데이터 집합을 로드 하는 경우에는 변경 내용을 명시적으로 수락할 필요가 없습니다. 기본적으로 `Fill` 메서드는 데이터 테이블 채우기를 완료 한 후 `AcceptChanges` 메서드를 호출 합니다.

관련 메서드인 RejectChangesOriginal 버전을 다시 Current 버전의 레코드에 복사 하 여 변경 효과를 취소 합니다. 또한 각 레코드의 RowState를 Unchanged으로 다시 설정 합니다.

데이터 유효성 검사

응용 프로그램의 데이터가 전달 되는 프로세스의 요구 사항을 충족 하는지 확인 하기 위해 종종 유효성 검사를 추가 해야 합니다. 이렇게 하려면 폼의 사용자 항목이 올바른지, 다른 응용 프로그램에 의해 응용 프로그램에 전송 된 데이터의 유효성을 검사 하거나, 구성 요소 내에서 계산 된 정보가 데이터 원본의 제약 조건에 포함 되는지 확인 해야 합니다. 및 응용 프로그램 요구 사항.

다음과 같은 여러 가지 방법으로 데이터의 유효성을 검사할 수 있습니다.

- 비즈니스 계층에서 데이터의 유효성을 검사 하는 코드를 응용 프로그램에 추가 합니다. 데이터 집합은이 작업을 수행할 수 있는 한 곳입니다. 데이터 집합은 열 및 행 값이 변경 될 때 변경 내용의 유효성을 검사 하는 기능과 같은 백 엔드 유효성 검사의 이점 중 일부를 제공 합니다. 자세한 내용은 데이터 집합의 데이터 유효성 검사를 참조 하세요.
- 프레젠테이션 계층에서 폼에 유효성 검사를 추가 합니다. 자세한 내용은 Windows Forms에서 사용자 입력 유효성 검사를 참조하세요.
- 데이터 백 엔드에서 데이터 원본으로 데이터를 보내고 (예: 데이터베이스) 데이터를 수락 하거나 거부 하도록 허용 합니다. 데이터의 유효성을 검사 하 고 오류 정보를 제공 하기 위한 정교한 기능이 있는 데이터베이스를 사용 하 여 작업 하는 경우에는 데이터의 출처에 상관 없이 데이터의 유효성을 검사할 수 있으므로 실용적인 방법이 될 수 있습니다. 그러나이 방법은 응용 프로그램별 유효성 검사 요구 사항을 수용 하지 못할 수 있습니다. 또한 데이터 원본의 유효성을 검사 하는 경우 백 엔드에서 발생 하는 유효성 검사 오류를 응용 프로그램에서 얼마나 쉽게 해결할 수 있는지에 따라 데이터 원본에 대 한 많은 왕복이 발생할 수 있습니다.

  > **IMPORTANT**
  > Text로 설정 된 CommandType 속성을 사용 하 여 데이터 명령을 사용 하는 경우 클라이언트에서 전송 된 정보를 데이터베이스에 전달 하기 전에 주의 깊게 확인 합니다. 악의적인 사용자가 인증되지 않은 액세스 권한을 얻거나 데이터베이스를 손상시키기 위해 수정되었거나 추가된 SQL 문을 전송(주입)할 수도 있습니다. 사용자 입력을 데이터베이스로 전송 하기 전에 항상 정보가 유효한 지 확인 하십시오. 가능 하면 항상 매개 변수가 있는 쿼리 또는 저장 프로시저를 사용 하는 것이 좋습니다.

데이터 원본에 업데이트 전송

데이터 집합에서 변경한 후에는 변경 내용을 데이터 원본에 전송할 수 있습니다. 가장 일반적으로 TableAdapter (또는 데이터 어댑터)의 `Update` 메서드를 호출 하 여이 작업을 수행 합니다. 메서드는 데이터 테이블의 각 레코드를 반복 하 여 필요한 업데이트 유형 (있는 경우)을 결정 하 고 적절 한 명령을 실행 합니다.

업데이트를 수행 하는 방법에 대 한 예시는 응용 프로그램에서 단일 데이터 테이블이 포함 된 데이터 집합을 사용 한다고 가정 합니다. 응용 프로그램은 데이터베이스에서 두 개의 행을 페치합니다. 검색 후 메모리 내 데이터 테이블은 다음과 같습니다.

```
(RowState)     CustomerID   Name             Status
(Unchanged)    c200         Robert Lyon      Good
(Unchanged)    c400         Nancy Buchanan    Pending
```

응용 프로그램이 김소미의 상태를 "기본 설정"으로 변경 합니다. 이러한 변경으로 인해 해당 행의 RowState 속성 값이 Unchanged에서 Modified로 변경 됩니다. 첫 번째 행에 대 한 RowState 속성 값은 Unchanged남아 있습니다. 이제 데이터 테이블은 다음과 같습니다.

```
(RowState)     CustomerID   Name             Status
(Unchanged)    c200         Robert Lyon      Good
(Modified)     c400         Nancy Buchanan    Preferred
```

이제 애플리케이션에서 `Update` 메서드를 호출하여 데이터 세트을 데이터베이스로 전송합니다. 메서드는 각 행을 차례로 검사 합니다. 첫 번째 행의 경우 메서드는 데이터베이스에서 원래 가져온 이후 해당 행이 변경 되지 않았기 때문에 SQL 문을 데이터베이스에 전송 하지 않습니다.

그러나 두 번째 행의 `Update` 메서드는 자동으로 올바른 데이터 명령을 호출 하 고 데이터베이스로 전송 합니다. SQL 문의 특정 구문은 기본 데이터 저장소에서 지 원하는 SQL 언어에 따라 달라 집니다. 하지만 전송 된 SQL 문의 다음과 같은 일반적인 특성은 중요 합니다.

- 전송 된 SQL 문은 UPDATE 문입니다. 어댑터는 RowState 속성 값이 Modified되므로 UPDATE 문을 사용 하는 것을 알고 있습니다.
- 전송 된 SQL 문에는 UPDATE 문의 대상이 `CustomerID = 'c400'` 행 임을 나타내는 WHERE 절이 포함 되어 있습니다. SELECT 문의이 부분은 대상 테이블의 기본 키 이기 때문 `CustomerID` 에 다른 모든 행의 대상 행을 구분 합니다. WHERE 절에 대 한 정보는 행을 식별 하는 데 필요한 값이 변경 된 경우 레코드의 원래 버전 (`DataRowVersion.Original`)에서 파생 됩니다.
- 전송 된 SQL 문에는 수정 된 열의 새 값을 설정 하는 SET 절이 포함 되어 있습니다.

> **NOTE**
> TableAdapter의 `UpdateCommand` 속성이 저장 프로시저의 이름으로 설정 된 경우 어댑터는 SQL 문을 생성 하지 않습니다. 대신 전달 된 적절 한 매개 변수를 사용 하 여 저장 프로시저를 호출 합니다.

매개 변수 전달

일반적으로 매개 변수를 사용 하 여 데이터베이스에서 업데이트할 레코드의 값을 전달 합니다. TableAdapter의 `Update` 메서드에서 UPDATE 문을 실행 하는 경우 매개 변수 값을 입력 해야 합니다. 적절 한 데이터 명령 (이 경우 TableAdapter의 `UpdateCommand` 개체)에 대 한 `Parameters` 컬렉션에서 이러한 값을 가져옵니다.

Visual Studio 도구를 사용 하 여 데이터 어댑터를 생성 한 경우 `UpdateCommand` 개체는 문의 각 매개 변수 자리 표시자에 해당 하는 매개 변수 컬렉션을 포함 합니다.

각 매개 변수의 System.Data.SqlClient.SqlParameter.SourceColumn 속성은 데이터 테이블의 열을 가리킵니다. 예를 들어 `au_id` 및 `Original_au_id` 매개 변수에 대 한 `SourceColumn` 속성은 데이터 테이블에서 만든이 id를 포함하는 모든 열로 설정 됩니다. 어댑터의 `Update` 메서드가 실행 되 면 업데이트 되는 레코드에서 author id 열을 읽고 해당 값을 문으로 채웁니다.

UPDATE 문에서는 레코드에 기록 될 새 값과 이전 값 (레코드를 데이터베이스에서 찾을 수 있도록)을 둘 다 지정 해야 합니다. 따라서 각 값에 대해 두 개의 매개 변수, 즉 SET 절과 WHERE 절에 대 한 다른 매개 변수가 있습니다. 두 매개 변수는 업데이트 되는 레코드에서 데이터를 읽었지만 매개 변수의 SourceVersion 속성에 따라 서로 다른 버전의 열 값을 가져옵니다. SET 절의 매개 변수는 현재 버전을 가져오고 WHERE 절에 대 한 매개 변수는 원래 버전을 가져옵니다.

> **NOTE**
> 데이터 어댑터의 RowChanging 이벤트에 대 한 이벤트 처리기에서 일반적으로 수행 하는 코드를 통해 `Parameters` 컬렉션의 값을 직접 설정할 수도 있습니다.

참조

- Visual Studio의 데이터 세트 도구
- TableAdapter 만들기 및 구성
- TableAdapter를 사용하여 데이터 업데이트
- Visual Studio에서 데이터에 컨트롤 바인딩

- 데이터 유효성 검사
- 방법: 엔터티 추가, 수정 및 삭제(WCF 데이터 서비스)

데이터베이스에 새 레코드 삽입

2020-01-06 • 9 minutes to read • Edit Online

데이터베이스에 새 레코드를 삽입 하려면 `TableAdapter.Update` 메서드나 TableAdapter의 DBDirect 메서드 중 하나 (특히 `TableAdapter.Insert` 메서드)를 사용할 수 있습니다. 자세한 내용은 TableAdapter를 참조 하세요.

응용 프로그램에서 Tableadapter를 사용 하지 않는 경우 명령 개체 (예: SqlCommand)를 사용 하 여 데이터베이스에 새 레코드를 삽입할 수 있습니다.

응용 프로그램에서 데이터 집합을 사용 하 여 데이터를 저장 하는 경우 `TableAdapter.Update` 메서드를 사용 합니다. `Update` 메서드는 데이터베이스에 모든 변경 내용 (업데이트, 삽입 및 삭제)을 보냅니다.

응용 프로그램에서 개체를 사용 하 여 데이터를 저장 하는 경우 또는 데이터베이스에 새 레코드를 만드는 방법을 보다 세밀 하 게 제어 하려면 `TableAdapter.Insert` 메서드를 사용 합니다.

TableAdapter에 `Insert` 메서드가 없는 경우 저장 프로시저를 사용 하도록 TableAdapter를 구성 했거나 해당 `GenerateDBDirectMethods` 속성이 `false`로 설정 되어 있음을 의미 합니다. TableAdapter의 `GenerateDBDirectMethods` 속성을 **데이터 세트 디자이너**내에서 `true`로 설정 하 고 데이터 집합을 저장 합니다. 이렇게 하면 TableAdapter가 다시 생성 됩니다. TableAdapter에 `Insert` 메서드가 아직 없는 경우 테이블은 개별 행을 구분 하는 데 충분 한 스키마 정보를 제공 하지 않을 수 있습니다. 예를 들어 테이블에 기본 키 집합이 없을 수 있습니다.

Tableadapter를 사용 하 여 새 레코드 삽입

Tableadapter는 응용 프로그램의 요구 사항에 따라 데이터베이스에 새 레코드를 삽입 하는 다양 한 방법을 제공 합니다.

응용 프로그램에서 데이터 집합을 사용 하 여 데이터를 저장 하는 경우 데이터 집합의 원하는 DataTable에 새 레코드를 추가한 다음 `TableAdapter.Update` 메서드를 호출할 수 있습니다. `TableAdapter.Update` 메서드는 DataTable 변경 내용을 데이터베이스 (수정 및 삭제 된 레코드 포함)로 보냅니다.

TableAdapter. Update 메서드를 사용 하 여 데이터베이스에 새 레코드를 삽입 하려면

1. 새 DataRow를 만들고 Rows 컬렉션에 추가 하 여 원하는 DataTable에 새 레코드를 추가 합니다.
2. 새 행이 DataTable에 추가 된 후 `TableAdapter.Update` 메서드를 호출 합니다. 전체 DataSet, DataTable, DataRow의 배열 또는 단일 DataRow를 전달 하 여 업데이트할 데이터의 양을 제어할 수 있습니다.

   다음 코드에서는 DataTable에 새 레코드를 추가한 다음 `TableAdapter.Update` 메서드를 호출 하 여 새 행을 데이터베이스에 저장 하는 방법을 보여 줍니다. 이 예에서는 Northwind 데이터베이스의 `Region` 테이블을 사용 합니다.

```
' Create a new row.
Dim newRegionRow As NorthwindDataSet.RegionRow
newRegionRow = Me.NorthwindDataSet._Region.NewRegionRow()
newRegionRow.RegionID = 5
newRegionRow.RegionDescription = "NorthWestern"

' Add the row to the Region table
Me.NorthwindDataSet._Region.Rows.Add(newRegionRow)

' Save the new row to the database
Me.RegionTableAdapter.Update(Me.NorthwindDataSet._Region)
```

```
// Create a new row.
NorthwindDataSet.RegionRow newRegionRow;
newRegionRow = northwindDataSet.Region.NewRegionRow();
newRegionRow.RegionID = 5;
newRegionRow.RegionDescription = "NorthWestern";

// Add the row to the Region table
this.northwindDataSet.Region.Rows.Add(newRegionRow);

// Save the new row to the database
this.regionTableAdapter.Update(this.northwindDataSet.Region);
```

TableAdapter. Insert 메서드를 사용 하 여 데이터베이스에 새 레코드를 삽입 하려면

응용 프로그램에서 개체를 사용 하 여 데이터를 저장 하는 경우 `TableAdapter.Insert` 메서드를 사용 하 여 데이터베이스에 새 행을 직접 만들 수 있습니다. `Insert` 메서드는 각 열에 대 한 개별 값을 매개 변수로 받아들입니다. 메서드를 호출 하면 전달 된 매개 변수 값을 사용 하 여 데이터베이스에 새 레코드가 삽입 됩니다.

- TableAdapter의 `Insert` 메서드를 호출 하 여 각 열에 대 한 값을 매개 변수로 전달 합니다.

다음 절차에서는 `TableAdapter.Insert` 메서드를 사용 하 여 행을 삽입 하는 방법을 보여 줍니다. 이 예에서는 Northwind 데이터베이스의 `Region` 테이블에 데이터를 삽입 합니다.

> **NOTE**
> 사용할 수 있는 인스턴스가 없는 경우 사용할 TableAdapter를 인스턴스화합니다.

```
Dim regionTableAdapter As New NorthwindDataSetTableAdapters.RegionTableAdapter

regionTableAdapter.Insert(5, "NorthWestern")
```

```
NorthwindDataSetTableAdapters.RegionTableAdapter regionTableAdapter =
    new NorthwindDataSetTableAdapters.RegionTableAdapter();

regionTableAdapter.Insert(5, "NorthWestern");
```

명령 개체를 사용 하 여 새 레코드 삽입

명령 개체를 사용 하 여 새 레코드를 데이터베이스에 직접 삽입할 수 있습니다.

명령 개체를 사용 하 여 데이터베이스에 새 레코드를 삽입 하려면

- 새 명령 개체를 만든 다음 `Connection`, `CommandType` 및 `CommandText` 속성을 설정 합니다.

다음 예에서는 명령 개체를 사용 하 여 데이터베이스에 레코드를 삽입 하는 방법을 보여 줍니다. Northwind 데이터베이스의 `Region` 테이블에 데이터를 삽입 합니다.

```
Dim sqlConnection1 As New System.Data.SqlClient.SqlConnection("YOUR CONNECTION STRING")

Dim cmd As New System.Data.SqlClient.SqlCommand
cmd.CommandType = System.Data.CommandType.Text
cmd.CommandText = "INSERT Region (RegionID, RegionDescription) VALUES (5, 'NorthWestern')"
cmd.Connection = sqlConnection1

sqlConnection1.Open()
cmd.ExecuteNonQuery()
sqlConnection1.Close()
```

```
System.Data.SqlClient.SqlConnection sqlConnection1 =
    new System.Data.SqlClient.SqlConnection("YOUR CONNECTION STRING");

System.Data.SqlClient.SqlCommand cmd = new System.Data.SqlClient.SqlCommand();
cmd.CommandType = System.Data.CommandType.Text;
cmd.CommandText = "INSERT Region (RegionID, RegionDescription) VALUES (5, 'NorthWestern')";
cmd.Connection = sqlConnection1;

sqlConnection1.Open();
cmd.ExecuteNonQuery();
sqlConnection1.Close();
```

.NET 보안

연결 하려는 데이터베이스에 대 한 액세스 권한과 원하는 테이블에 대 한 삽입을 수행할 수 있는 권한이 있어야 합니다.

참조

- 데이터를 다시 데이터베이스에 저장

TableAdapter를 사용하여 데이터 업데이트

2020-01-06 • 4 minutes to read • Edit Online

데이터 집합의 데이터를 수정 하 고 유효성을 검사 한 후에는 TableAdapter의 `Update` 메서드를 호출 하 여 업데이트 된 데이터를 데이터베이스로 다시 보낼 수 있습니다. `Update` 메서드는 단일 데이터 테이블을 업데이트 하 고 테이블의 각 데이터 행에 대 한 RowState에 따라 올바른 명령 (INSERT, UPDATE 또는 DELETE)을 실행 합니다. 데이터 집합에 관련 테이블이 있으면 Visual Studio에서 업데이트를 수행 하는 데 사용 하는 TableAdapterManager 클래스를 생성 합니다. TableAdapterManager 클래스를 사용 하면 데이터베이스에 정의 된 외래 키 제약 조건을 기반으로 올바른 순서로 업데이트가 수행 됩니다. 데이터 바인딩된 컨트롤을 사용 하는 경우 데이터 바인딩 아키텍처는 tableAdapterManager 클래스의 멤버 변수를 만듭니다.

> **NOTE**
> 데이터 집합의 내용을 사용 하 여 데이터 소스를 업데이트 하려고 하면 오류가 발생할 수 있습니다. 오류를 방지 하려면 어댑터의 `Update` 메서드를 호출 하는 코드를 `try` / `catch` 블록 내에 배치 하는 것이 좋습니다.

데이터 원본을 업데이트 하는 정확한 방법은 비즈니스 요구 사항에 따라 다를 수 있지만 다음 단계를 포함 합니다.

1. `try` / `catch` 블록에서 어댑터의 `Update` 메서드를 호출 합니다.
2. 예외가 catch 되 면 오류가 발생 한 데이터 행을 찾습니다.
3. 데이터 행의 문제를 조정 하 여 (가능 하면 프로그래밍 방식으로 또는 사용자에 게 잘못 된 행을 표시 하 여 수정할 수 있음) 업데이트를 다시 시도 합니다 (HasErrors, GetErrors).

데이터베이스에 데이터 저장

TableAdapter의 `Update` 메서드를 호출 합니다. 데이터베이스에 쓸 값을 포함 하는 데이터 테이블의 이름을 전달 합니다.

TableAdapter를 사용 하 여 데이터베이스를 업데이트 하려면

- TableAdapter의 `Update` 메서드를 `try` / `catch` 블록으로 묶습니다. 다음 예에서는 `try` / `catch` 블록 내에서 `NorthwindDataSet`의 `Customers` 테이블 내용을 업데이트 하는 방법을 보여 줍니다.

```
try
{
    this.Validate();
    this.customersBindingSource.EndEdit();
    this.customersTableAdapter.Update(this.northwindDataSet.Customers);
    MessageBox.Show("Update successful");
}
catch (System.Exception ex)
{
    MessageBox.Show("Update failed");
}
```

```
Try
    Me.Validate()
    Me.CustomersBindingSource.EndEdit()
    Me.CustomersTableAdapter.Update(Me.NorthwindDataSet.Customers)
    MsgBox("Update successful")

Catch ex As Exception
    MsgBox("Update failed")
End Try
```

참조

- 데이터를 다시 데이터베이스에 저장

계층적 업데이트

2020-01-06 • 24 minutes to read • Edit Online

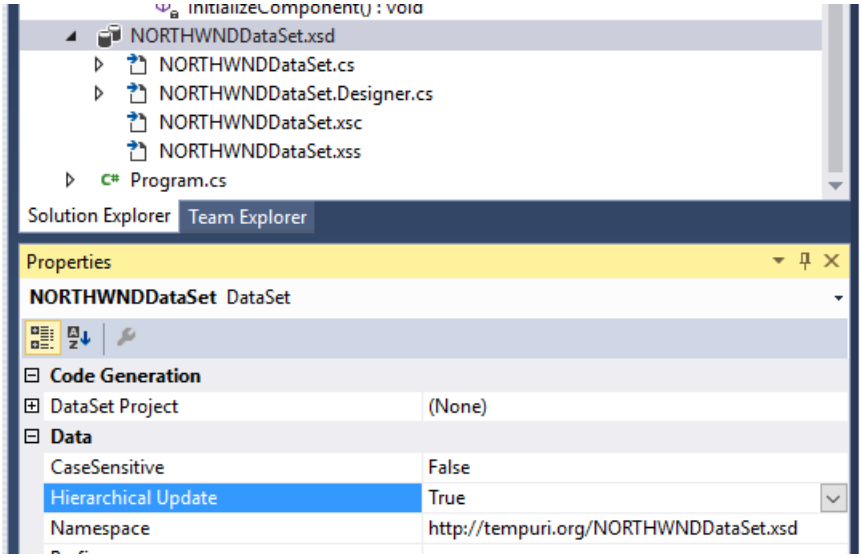
*계층적 업데이트*는 참조 무결성 규칙을 유지 하면서 업데이트 된 데이터를 두 개 이상의 관련 테이블이 있는 데이터 집합에서 데이터베이스에 다시 저장 하는 프로세스를 의미 합니다. *참조 무결성*은 관련 레코드 삽입, 업데이트 및 삭제 동작을 제어 하는 데이터베이스의 제약 조건에서 제공 하는 일관성 규칙을 나타냅니다. 예를 들어 고객 레코드 만들기를 적용 하는 참조 무결성을 적용 하 여 해당 고객에 대 한 주문을 만들도록 허용 합니다. 데이터 집합의 관계에 대 한 자세한 내용은 데이터 집합의 관계를 참조 하세요.

계층적 업데이트 기능은 `TableAdapterManager`를 사용 하 여 형식화 된 데이터 집합의 `TableAdapter`를 관리 합니다. `TableAdapterManager` 구성 요소는 .NET 형식이 아닌 Visual Studio에서 생성 된 클래스입니다. **데이터 소스** 창에서 Windows FORM 또는 WPF 페이지로 테이블을 끌어 오면 Visual Studio에서 TableAdapterManager 형식의 변수를 폼 이나 페이지에 추가 하 고 디자이너의 구성 요소 트레이에 표시 합니다. `TableAdapterManager` 클래스에 대 한 자세한 내용은 tableadapter의 TableAdapterManager 참조 섹션을 참조 하세요.

기본적으로 데이터 집합은 관련 테이블을 "관계 전용"으로 처리 합니다. 즉, foreign key 제약 조건을 적용 하지 않습니다. **데이터 세트 디자이너**를 사용 하 여 디자인 타임에 해당 설정을 수정할 수 있습니다. 두 테이블 간의 관계 선을 선택 하 여 **관계** 대화 상자를 표시 합니다. 여기에서 변경한 내용에 따라 관련 테이블의 변경 내용을 데이터베이스에 다시 보낼 때 `TableAdapterManager`의 동작 방식이 결정 됩니다.

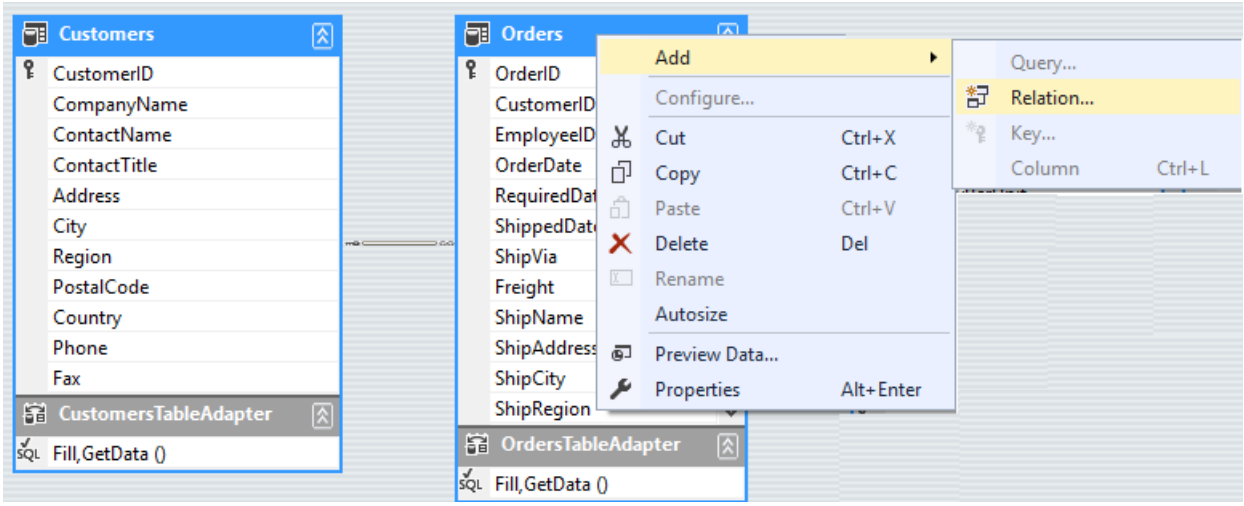
데이터 집합에서 계층적 업데이트를 사용 하도록 설정

기본적으로 계층적 업데이트는 프로젝트에서 추가 되거나 만들어진 모든 새 데이터 집합에 대해 사용 하도록 설정 됩니다. 데이터 집합에서 형식화 된 데이터 집합의 **계층적 업데이트** 속성을 **True** 또는 **False**로 설정 하 여 계층적 업데이트를 설정 하거나 해제 합니다.



테이블 간에 새 관계 만들기

두 테이블 간에 새 관계를 만들려면 데이터 세트 디자이너에서 각 테이블의 제목 표시줄을 선택 하 고 마우스 오른쪽 단추를 클릭 한 다음 **관계 추가**를 선택 합니다.



외래 키 제약 조건, 연계 업데이트 및 삭제 이해

생성 된 데이터 집합 코드에서 데이터베이스의 foreign key 제약 조건 및 연계 동작을 만드는 방법을 이해 하는 것이 중요 합니다.

기본적으로 데이터 집합의 데이터 테이블은 데이터베이스의 관계와 일치 하는 관계 (DataRelation)를 사용 하 여 생성 됩니다. 그러나 데이터 집합의 관계는 외래 키 제약 조건으로 생성 되지 않습니다. DataRelation UpdateRule 또는 DeleteRule 적용 되지 않은 경우에 **만 관계로** 구성 됩니다.

기본적으로 연계 업데이트 및/또는 연계 된 삭제가 설정 된 상태에서 데이터베이스 관계가 설정 된 경우에도 연계 업데이트 및 연계 삭제는 해제 됩니다. 예를 들어 새 고객과 새 주문을 만든 다음 데이터를 저장 하려고 하면 데이터베이스에 정의 된 foreign key 제약 조건과 충돌이 발생할 수 있습니다. 자세한 내용은 데이터 집합을 채우는 동안 제약 조건해제를 참조 하세요.

업데이트를 수행 하는 순서 설정

업데이트를 수행 하는 순서를 설정 하면 데이터 집합의 모든 테이블에 수정 된 모든 데이터를 저장 하는 데 필요한 개별 삽입, 업데이트 및 삭제 순서가 설정 됩니다. 계층적 업데이트를 사용 하는 경우 삽입이 먼저 수행 된 다음 업데이트 및 삭제 됩니다. `TableAdapterManager`는 먼저 업데이트를 수행 하 고 삽입 한 다음 삭제 하도록 설정 할 수 있는 `UpdateOrder` 속성을 제공 합니다.

> **NOTE**
> 업데이트 순서는 모두 포함 된다는 것을 이해 하는 것이 중요 합니다. 즉, 업데이트를 수행 하는 경우 데이터 집합의 모든 테이블에 대 한 삽입 및 삭제 작업이 수행 됩니다.

`UpdateOrder` 속성을 설정 하려면 데이터 소스 창 에서 폼으로 항목을 끌어 온 후 구성 요소 트레이에서 `TableAdapterManager`를 선택 하 고 **속성** 창에서 `UpdateOrder` 속성을 설정 합니다.

계층적 업데이트를 수행 하기 전에 데이터 집합의 백업 복사본 만들기

`TableAdapterManager.UpdateAll()` 메서드를 호출 하 여 데이터를 저장 하면 `TableAdapterManager`는 단일 트랜잭션으로 각 테이블에 대 한 데이터를 업데이트 하려고 시도 합니다. 테이블에 대 한 업데이트의 일부가 실패 하면 전체 트랜잭션이 롤백됩니다. 대부분의 경우 롤백은 응용 프로그램을 원래 상태로 되돌립니다.

그러나 경우에 따라 백업 복사본에서 데이터 집합을 복원 하는 것이 좋습니다. 자동 증분 값을 사용 하는 경우이에 대 한 한 가지 예가 발생할 수 있습니다. 예를 들어 저장 작업에 성공 하지 못하면 자동 증분 값이 데이터 집합에서 다시 설정 되지 않으며 데이터 집합은 자동 증분 값을 계속 만듭니다. 이로 인해 응용 프로그램에서 허용 되지 않는 번호 매기기 간격이 남아 있습니다. 이 문제가 발생 하는 경우 `TableAdapterManager`은 트랜잭션이 실패 할 경우 기존 데이터 집합을 백업 복사본으로 대체 하는 `BackupDataSetBeforeUpdate` 속성을 제공 합니다.

NOTE

백업 복사본은 `TableAdapterManager.UpdateAll` 메서드가 실행 되는 동안에만 메모리에 있습니다. 따라서이 백업 데이터 집합은 원본 데이터 집합을 대체 하거나 `TableAdapterManager.UpdateAll` 메서드의 실행이 완료 되는 즉시 범위를 벗어나기 때문에이 백업 데이터 집합에 대 한 프로그래밍 방식의 액세스는 없습니다.

생성 된 저장 코드를 수정 하 여 계층적 업데이트 수행

`TableAdapterManager.UpdateAll` 메서드를 호출한 다음 관련 테이블이 포함된 데이터 세트의 이름을 전달하여 데이터 세트의 관련 데이터 테이블에서 데이터베이스로 변경 내용을 저장합니다. 예를 들어 NorthwindDataset에 포함된 모든 테이블의 업데이트를 백 엔드 데이터베이스로 보내려면 `TableAdapterManager.UpdateAll(NorthwindDataset)` 메서드를 실행합니다.

데이터 원본 창에서 항목을 놓으면 코드가 `Form_Load` 이벤트에 자동으로 추가되어 각 테이블을 채웁니다(`TableAdapter.Fill` 메서드). 또한 데이터 세트의 데이터를 데이터베이스에 다시 저장할 수 있도록 BindingNavigator의 **저장** 단추 클릭 이벤트에도 코드가 추가됩니다(`TableAdapterManager.UpdateAll` 메서드).

생성된 저장 코드에는 `CustomersBindingSource.EndEdit` 메서드를 호출하는 코드 줄도 포함됩니다. 구체적으로 말하면 폼에 추가 된 첫 번째 BindingSource의 EndEdit 메서드를 호출 합니다. 즉,이 코드는 **데이터 소스** 창에서 폼으로 끌어온 첫 번째 테이블에 대해서만 생성 됩니다. EndEdit 호출에서는 현재 편집 중인 데이터 바인딩된 컨트롤의 프로세스에 포함된 모든 변경 내용을 커밋합니다. 따라서 데이터 바인딩된 컨트롤에 계속 포커스가 있는 상태에서 **저장** 단추를 클릭하면 실제 저장 전에 해당 컨트롤에서 보류 중인 모든 편집 내용이 커밋됩니다(`TableAdapterManager.UpdateAll` 메서드).

NOTE

데이터 세트 디자이너 는 폼에 끌어 놓은 첫 번째 테이블의 `BindingSource.EndEdit` 코드를 추가 합니다. 그러므로 폼의 각 관련 테이블에 대해 `BindingSource.EndEdit` 메서드를 호출하는 코드 줄을 추가해야 합니다. 이 연습에서는 `OrdersBindingSource.EndEdit` 메서드 호출을 추가해야 합니다.

저장 전에 관련 테이블로 변경 내용을 커밋하도록 코드를 업데이트하려면

1. BindingNavigator에서 **저장** 단추를 두 번 클릭하여 코드 편집기에서 **Form1**을 엽니다.
2. `OrdersBindingSource.EndEdit` 메서드를 호출하는 줄 뒤에 `CustomersBindingSource.EndEdit` 메서드를 호출하는 코드 줄을 추가합니다. **저장** 단추 클릭 이벤트 내의 코드는 다음과 같습니다.

```
Me.Validate()
Me.CustomersBindingSource.EndEdit()
Me.OrdersBindingSource.EndEdit()
Me.TableAdapterManager.UpdateAll(Me.NorthwindDataSet)
```

```
this.Validate();
this.customersBindingSource.EndEdit();
this.ordersBindingSource.EndEdit();
this.tableAdapterManager.UpdateAll(this.northwindDataSet);
```

이처럼 데이터를 데이터베이스에 저장하기 전에 관련 자식 테이블에 대해 변경 내용을 커밋해야 할 뿐 아니라 새 자식 레코드를 데이터 세트에 추가하기 전에 새로 만든 부모 레코드도 커밋해야 할 수 있습니다. 즉, 새 부모 레코드 (`Customer`)를 데이터 집합에 추가 해야 할 수 있습니다. 외래 키 제약 조건을 사용 하면 새 자식 레코드 (`Orders`)를 데이터 집합에 추가할 수 있습니다. 자식 `BindingSource.AddingNew` 이벤트를 사용하면 이 작업을 수행할 수 있습니다.

> **NOTE**
> 새 부모 레코드를 커밋해야 하는지 여부는 데이터 원본에 바인딩하는 데 사용 되는 컨트롤의 형식에 따라 달라 집니다. 이 연습에서는 개별 컨트롤을 사용 하 여 부모 테이블에 바인딩합니다. 이렇게 하려면 새 부모 레코드를 커밋하는 추가 코드가 필요 합니다. 부모 레코드가 DataGridView와 같은 복합 바인딩 컨트롤에 표시 되는 경우에는 부모 레코드에 대 한 추가 EndEdit 호출이 필요 하지 않습니다. 컨트롤의 기본 데이터 바인딩 기능이 새 레코드 커밋을 처리하기 때문입니다.

새 자식 레코드를 추가하기 전에 데이터 세트에서 부모 레코드를 커밋하는 코드를 추가하려면

1. `OrdersBindingSource.AddingNew` 이벤트에 대한 이벤트 처리기를 만듭니다.
   - 디자인 뷰에서 **Form1** 을 열고 구성 요소 트레이에서 **OrdersBindingSource** 를 선택한 다음 **속성** 창에서 **이벤트** 를 선택 하 고 **system.windows.forms.dataconnector.addingnew** 이벤트를 두 번 클릭 합니다.
2. `CustomersBindingSource.EndEdit` 메서드를 호출 하는 이벤트 처리기에 코드 줄을 추가 합니다. `OrdersBindingSource_AddingNew` 이벤트 처리기의 코드는 다음과 같습니다.

```
Me.CustomersBindingSource.EndEdit()
```

```
this.customersBindingSource.EndEdit();
```

TableAdapterManager 참조

기본적으로 `TableAdapterManager` 클래스는 관련 테이블을 포함 하는 데이터 집합을 만들 때 생성 됩니다. 클래스가 생성 되지 않도록 하려면 데이터 집합의 `Hierarchical Update` 속성 값을 false로 변경 합니다. 관계가 있는 테이블을 Windows Form 또는 WPF 페이지의 디자인 화면으로 끌어 오면 Visual Studio에서 클래스의 멤버 변수를 선언 합니다. 데이터 바인딩을 사용 하지 않는 경우 변수를 수동으로 선언 해야 합니다.

`TableAdapterManager` 클래스는 .NET 형식이 아닙니다. 따라서 설명서에서 확인할 수 없습니다. 디자인 타임에 데이터 집합 생성 프로세스의 일부로 만들어집니다.

다음은 `TableAdapterManager` 클래스의 자주 사용 되는 메서드와 속성입니다.

MEMBER	설명
`UpdateAll` 메서드	모든 데이터 테이블의 모든 데이터를 저장 합니다.
`BackUpDataSetBeforeUpdate` 속성	`TableAdapterManager.UpdateAll` 메서드를 실행 하기 전에 데이터 집합의 백업 복사본을 만들지 여부를 결정 합니다. 부울.
tableName `TableAdapter` 속성	`TableAdapter`를 나타냅니다. 생성 된 `TableAdapterManager`는 관리 하는 각 `TableAdapter`에 대 한 속성을 포함 합니다. 예를 들어 Customers 및 Orders 테이블이 포함 된 데이터 집합은 `CustomersTableAdapter` 및 `OrdersTableAdapter` 속성을 포함 하는 `TableAdapterManager`를 사용 하 여 생성 됩니다.

MEMBER	설명
`UpdateOrder` 속성	개별 insert, update 및 delete 명령의 순서를 제어 합니다. 이를 `TableAdapterManager.UpdateOrderOption` 열거형의 값 중 하나로 설정 합니다. 기본적으로 `UpdateOrder`은 **Insertupdatedelete**로 설정 됩니다. 즉, 데이터 집합의 모든 테이블에 대 한 삽입, 업데이트 및 삭제 작업이 수행 됩니다.

참조

- 데이터를 다시 데이터베이스에 저장

동시성 예외 처리

2020-01-06 • 22 minutes to read • Edit Online

동시성 예외 (System.Data.DBConcurrencyException)는 두 사용자가 동시에 데이터베이스의 동일한 데이터를 변경하려고 할 때 발생 합니다. 이 연습에서는 DBConcurrencyException를 catch 하 고 오류를 발생 시킨 행을 찾아 처리하는 방법에 대 한 전략을 파악 하는 방법을 보여 주는 Windows 응용 프로그램을 만듭니다.

이 연습에서는 다음 프로세스를 수행 합니다.

1. 새 **Windows Forms 애플리케이션** 프로젝트를 만듭니다.
2. Northwind Customers 테이블을 기반으로 새 데이터 집합을 만듭니다.
3. DataGridView를 사용 하 여 폼을 만들어 데이터를 표시 합니다.
4. Northwind 데이터베이스의 Customers 테이블 데이터를 사용 하 여 데이터 집합을 채웁니다.
5. **서버 탐색기** 의 **테이블 데이터 표시** 기능을 사용 하 여 Customers 테이블의 데이터에 액세스 하 고 레코드를 변경할 수 있습니다.
6. 동일한 레코드를 다른 값으로 변경 하 고, 데이터 집합을 업데이트 하 고, 변경 내용을 데이터베이스에 쓰려고 시도 하 여 동시성 오류가 발생 합니다.
7. 오류를 파악 한 다음 레코드의 다른 버전을 표시 하 여 사용자가 계속 해 서 데이터베이스를 업데이트 하거나 업데이트를 취소할지 여부를 결정할 수 있도록 합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

새 프로젝트 만들기

새 Windows Forms 응용 프로그램을 만들어 시작 합니다.

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.

3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.

4. 프로젝트 이름을 **ConcurrencyWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   ConcurrencyWalkthrough 프로젝트가 만들어지고 **솔루션 탐색기**에 추가 되 고 디자이너에서 새 양식이 열립니다.

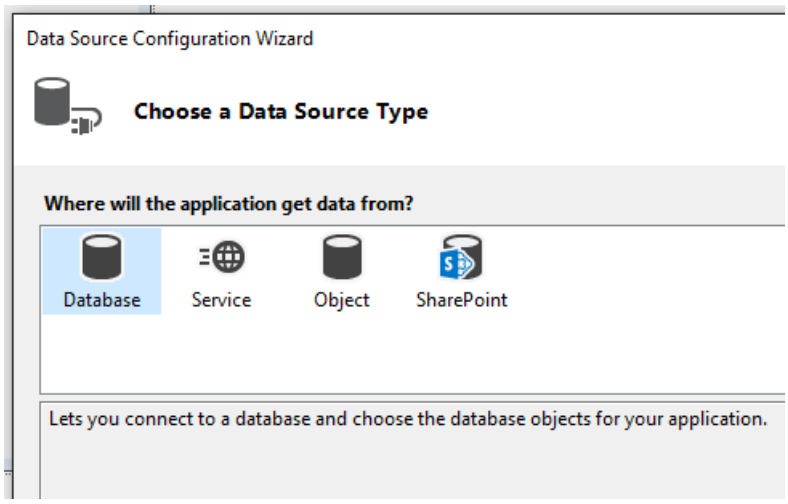
Northwind 데이터 집합 만들기

다음으로 **NorthwindDataSet**라는 데이터 집합을 만듭니다.

1. **데이터** 메뉴에서 **새 데이터 소스 추가**를 선택 합니다.

   데이터 소스 구성 마법사가 열립니다.

2. **데이터 소스 형식 선택** 화면에서 **데이터베이스**를 선택 합니다.



3. 사용 가능한 연결 목록에서 Northwind 샘플 데이터베이스에 대 한 연결을 선택 합니다. 연결 목록에서 연결을 사용할 수 없는 경우 **새 연결**을 선택 합니다.

> **NOTE**
> 로컬 데이터베이스 파일에 연결 하는 경우 프로젝트에 파일을 추가할지 묻는 메시지가 표시 되 면 **아니요** 를 선택 합니다.

4. **응용 프로그램 구성 파일에 연결 문자열 저장** 화면에서 **다음**을 선택 합니다.

5. **테이블** 노드를 확장 하 고 **Customers** 테이블을 선택 합니다. 데이터 집합의 기본 이름은 **NorthwindDataSet**여야 합니다.

6. **마침** 을 선택 하 여 프로젝트에 데이터 집합을 추가 합니다.

데이터 바인딩된 DataGridView 컨트롤 만들기

이 섹션에서는 **데이터 소스** 창에서 Windows Form으로 **Customers** 항목을 끌어서 System.Windows.Forms.DataGridView를 만듭니다.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다.

2. **데이터 소스** 창에서 **NorthwindDataSet** 노드를 확장 한 다음 **Customers** 테이블을 선택 합니다.

3. 테이블 노드에서 아래쪽 화살표를 선택 하 고 드롭다운 목록에서 **DataGridView** 를 선택 합니다.

4. 테이블을 폼의 빈 영역으로 끕니다.

Customersdatagridview라는 DataGridView 컨트롤과 **customersdatagridview**라는 BindingNavigator BindingSource에 바인딩되는 폼에 추가 됩니다. 이는 NorthwindDataSet의 Customers 테이블에 바인딩됩니다.

양식 테스트

이제 폼을 테스트 하 여이 시점까지 예상 대로 동작 하는지 확인할 수 있습니다.

1. **F5 키** 를 선택 하 여 응용 프로그램을 실행 합니다.

   이 폼에는 Customers 테이블의 데이터로 채워진 DataGridView 컨트롤이 표시 됩니다.

2. **디버그** 메뉴에서 **디버깅 중지**를 선택합니다.

동시성 오류 처리

오류를 처리 하는 방법은 응용 프로그램을 제어 하는 특정 비즈니스 규칙에 따라 달라 집니다. 이 연습에서는 동시성 오류를 처리 하는 방법에 대 한 예제로 다음 전략을 사용 합니다.

응용 프로그램은 사용자에 게 세 가지 버전의 레코드를 제공 합니다.

- 데이터베이스의 현재 레코드
- 데이터 집합에 로드 된 원본 레코드
- 데이터 집합에서 제안 된 변경 내용

그런 다음 사용자는 데이터베이스를 제안 된 버전으로 덮어쓰거나 업데이트를 취소 하 고 데이터베이스의 새 값을 사용 하 여 데이터 집합을 새로 고칠 수 있습니다.

동시성 오류 처리를 사용 하도록 설정 하려면

1. 사용자 지정 오류 처리기를 만듭니다.
2. 사용자에 게 선택 항목을 표시 합니다.
3. 사용자의 응답을 처리 합니다.
4. 업데이트를 다시 보내거나 데이터 집합의 데이터를 다시 설정 합니다.

동시성 예외를 처리 하는 코드 추가

업데이트를 수행 하려고 할 때 예외가 발생 하면 일반적으로 발생 한 예외에 의해 제공 된 정보를 사용 하 여 작업을 수행 하려고 합니다. 이 섹션에서는 데이터베이스 업데이트를 시도 하는 코드를 추가 합니다. 또한 발생할 수 있는 모든 DBConcurrencyException 및 기타 예외를 처리 합니다.

> **NOTE**
> `CreateMessage` 및 `ProcessDialogResults` 메서드는 연습의 뒷부분에서 추가 됩니다.

1. `Form1_Load` 메서드 아래에 다음 코드를 추가 합니다.

```
private void UpdateDatabase()
{
    try
    {
        this.customersTableAdapter.Update(this.northwindDataSet.Customers);
        MessageBox.Show("Update successful");
    }
    catch (DBConcurrencyException dbcx)
    {
        DialogResult response = MessageBox.Show(CreateMessage((NorthwindDataSet.CustomersRow)
            (dbcx.Row)), "Concurrency Exception", MessageBoxButtons.YesNo);

        ProcessDialogResult(response);
    }
    catch (Exception ex)
    {
        MessageBox.Show("An error was thrown while attempting to update the database.");
    }
}
```

```
Private Sub UpdateDatabase()

    Try
        Me.CustomersTableAdapter.Update(Me.NorthwindDataSet.Customers)
        MsgBox("Update successful")

    Catch dbcx As Data.DBConcurrencyException
        Dim response As Windows.Forms.DialogResult

        response = MessageBox.Show(CreateMessage(CType(dbcx.Row, NorthwindDataSet.CustomersRow)),
            "Concurrency Exception", MessageBoxButtons.YesNo)

        ProcessDialogResult(response)

    Catch ex As Exception
        MsgBox("An error was thrown while attempting to update the database.")
    End Try
End Sub
```

2. `CustomersBindingNavigatorSaveItem_Click` 메서드를 대체 하 여 다음과 같이 `UpdateDatabase` 메서드를 호출 합니다.

```
private void customersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    UpdateDatabase();
}
```

```
Private Sub CustomersBindingNavigatorSaveItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CustomersBindingNavigatorSaveItem.Click
    UpdateDatabase()
End Sub
```

사용자에 게 선택 항목 표시

방금 작성 한 코드는 `CreateMessage` 프로시저를 호출 하 여 사용자에 게 오류 정보를 표시 합니다. 이 연습에서는 메시지 상자를 사용 하 여 레코드의 다른 버전을 사용자에 게 표시 합니다. 이를 통해 사용자는 레코드를 변경 내용으로 덮어쓸지 아니면 편집을 취소할지를 선택할 수 있습니다. 사용자가 메시지 상자에서 단추를 클릭 하 여 옵션을 선택 하면 응답이 `ProcessDialogResult` 메서드에 전달 됩니다.

코드 편집기에 다음 코드를 추가 하 여 메시지를 만듭니다. `UpdateDatabase` 메서드 아래에 다음 코드를 입력 합니

다.

```
private string CreateMessage(NorthwindDataSet.CustomersRow cr)
{
    return
        "Database: " + GetRowData(GetCurrentRowInDB(cr), DataRowVersion.Default) + "\n" +
        "Original: " + GetRowData(cr, DataRowVersion.Original) + "\n" +
        "Proposed: " + GetRowData(cr, DataRowVersion.Current) + "\n" +
        "Do you still want to update the database with the proposed value?";
}


//-----------------------------------------------------------------------
// This method loads a temporary table with current records from the database
// and returns the current values from the row that caused the exception.
//-----------------------------------------------------------------------
private NorthwindDataSet.CustomersDataTable tempCustomersDataTable =
    new NorthwindDataSet.CustomersDataTable();

private NorthwindDataSet.CustomersRow GetCurrentRowInDB(NorthwindDataSet.CustomersRow RowWithError)
{
    this.customersTableAdapter.Fill(tempCustomersDataTable);

    NorthwindDataSet.CustomersRow currentRowInDb =
        tempCustomersDataTable.FindByCustomerID(RowWithError.CustomerID);

    return currentRowInDb;
}


//-----------------------------------------------------------------------
// This method takes a CustomersRow and RowVersion
// and returns a string of column values to display to the user.
//-----------------------------------------------------------------------
private string GetRowData(NorthwindDataSet.CustomersRow custRow, DataRowVersion RowVersion)
{
    string rowData = "";

    for (int i = 0; i < custRow.ItemArray.Length ; i++ )
    {
        rowData = rowData + custRow[i, RowVersion].ToString() + " ";
    }
    return rowData;
}
```

```
Private Function CreateMessage(ByVal cr As NorthwindDataSet.CustomersRow) As String
    Return "Database: " & GetRowData(GetCurrentRowInDB(cr),
                                     Data.DataRowVersion.Default) & vbCrLf &
           "Original: " & GetRowData(cr, Data.DataRowVersion.Original) & vbCrLf &
           "Proposed: " & GetRowData(cr, Data.DataRowVersion.Current) & vbCrLf &
           "Do you still want to update the database with the proposed value?"
End Function


'------------------------------------------------------------------------
' This method loads a temporary table with current records from the database
' and returns the current values from the row that caused the exception.
'------------------------------------------------------------------------
Private TempCustomersDataTable As New NorthwindDataSet.CustomersDataTable

Private Function GetCurrentRowInDB(
    ByVal RowWithError As NorthwindDataSet.CustomersRow
    ) As NorthwindDataSet.CustomersRow

    Me.CustomersTableAdapter.Fill(TempCustomersDataTable)

    Dim currentRowInDb As NorthwindDataSet.CustomersRow =
        TempCustomersDataTable.FindByCustomerID(RowWithError.CustomerID)

    Return currentRowInDb
End Function


'------------------------------------------------------------------------
' This method takes a CustomersRow and RowVersion
' and returns a string of column values to display to the user.
'------------------------------------------------------------------------
Private Function GetRowData(ByVal custRow As NorthwindDataSet.CustomersRow,
    ByVal RowVersion As Data.DataRowVersion) As String

    Dim rowData As String = ""

    For i As Integer = 0 To custRow.ItemArray.Length - 1
        rowData &= custRow.Item(i, RowVersion).ToString() & " "
    Next

    Return rowData
End Function
```

사용자의 응답을 처리 합니다.

또한 메시지 상자에 대 한 사용자의 응답을 처리 하는 코드도 필요 합니다. 옵션은 데이터베이스의 현재 레코드를 제안 된 변경 내용으로 덮어쓰거나 로컬 변경 내용을 취소 하 고 현재 데이터베이스에 있는 레코드를 사용 하 여 데이터 테이블을 새로 고치는 것입니다. 사용자가 **예**를 선택 하면 *preserveChanges* 인수를 **true**로 설정 하 여 Merge 메서드가 호출 됩니다. 그러면 레코드의 원래 버전이 데이터베이스의 레코드와 일치 하기 때문에 업데이트에 성공 하 게 됩니다.

이전 섹션에서 추가 된 코드 아래에 다음 코드를 추가 합니다.

```
// This method takes the DialogResult selected by the user and updates the database
// with the new values or cancels the update and resets the Customers table
// (in the dataset) with the values currently in the database.

private void ProcessDialogResult(DialogResult response)
{
    switch (response)
    {
        case DialogResult.Yes:
            northwindDataSet.Merge(tempCustomersDataTable, true, MissingSchemaAction.Ignore);
            UpdateDatabase();
            break;

        case DialogResult.No:
            northwindDataSet.Merge(tempCustomersDataTable);
            MessageBox.Show("Update cancelled");
            break;
    }
}
```

```
' This method takes the DialogResult selected by the user and updates the database
' with the new values or cancels the update and resets the Customers table
' (in the dataset) with the values currently in the database.

Private Sub ProcessDialogResult(ByVal response As Windows.Forms.DialogResult)

    Select Case response

        Case Windows.Forms.DialogResult.Yes
            NorthwindDataSet.Customers.Merge(TempCustomersDataTable, True)
            UpdateDatabase()

        Case Windows.Forms.DialogResult.No
            NorthwindDataSet.Customers.Merge(TempCustomersDataTable)
            MsgBox("Update cancelled")
    End Select
End Sub
```

양식 테스트

이제 폼을 테스트 하 여 예상 대로 동작 하는지 확인할 수 있습니다. 동시성 위반을 시뮬레이트하려면 NorthwindDataSet을 채운 후 데이터베이스에서 데이터를 변경 합니다.

1. **F5 키** 를 선택 하 여 응용 프로그램을 실행 합니다.
2. 양식이 표시 되 면 실행을 그대로 유지 하 고 Visual Studio IDE로 전환 합니다.
3. **보기** 메뉴에서 **서버 탐색기**를 선택합니다.
4. **서버 탐색기**에서 응용 프로그램이 사용 하는 연결을 확장 한 다음 **테이블** 노드를 확장 합니다.
5. **Customers** 테이블을 마우스 오른쪽 단추로 클릭 한 다음 **테이블 데이터 표시**를 선택 합니다.
6. 첫 번째 레코드 (ALFKI)에서 **ContactName** 를 **민 Anders2**로 변경 합니다.

   > **NOTE**
   > 다른 행으로 이동 하 여 변경 내용을 커밋합니다.

7. ConcurrencyWalkthrough의 실행 중인 폼으로 전환 합니다.

8. 폼의 첫 번째 레코드 (**ALFKI**)에서 **ContactName** 를 **민 Anders1**로 변경 합니다.
9. **저장** 단추를 선택합니다.

   동시성 오류가 발생 하 고 메시지 상자가 나타납니다.

   아니요 를 선택 하면 업데이트를 취소 하 고 현재 데이터베이스에 있는 값으로 데이터 집합을 업데이트 합니다. **예** 를 선택 하면 데이터베이스에 제안 된 값이 기록 됩니다.

참조

- 데이터를 다시 데이터베이스에 저장

방법: 트랜잭션을 사용하여 데이터 저장

2020-01-06 • 3 minutes to read • Edit Online

System.Transactions 네임 스페이스를 사용 하 여 트랜잭션에 데이터를 저장 합니다. TransactionScope 개체를 사용 하 여 자동으로 관리 되는 트랜잭션에 참여할 수 있습니다.

프로젝트는 *System. 트랜잭션* 어셈블리에 대 한 참조를 사용 하 여 만들어지지 않으므로 트랜잭션을 사용 하는 프로젝트에 대 한 참조를 수동으로 추가 해야 합니다.

트랜잭션을 구현 하는 가장 쉬운 방법은 `using` 문에서 TransactionScope 개체를 인스턴스화하는 것입니다. 자세한 내용은 using 문및 using 문을 참조 하세요. `using` 문 내에서 실행 되는 코드는 트랜잭션에 참여 합니다.

트랜잭션을 커밋하려면 using 블록의 마지막 문으로 Complete 메서드를 호출 합니다.

트랜잭션을 롤백하려면 Complete 메서드를 호출 하기 전에 예외를 throw 합니다.

System.object에 대 한 참조를 추가 하려면

1. **프로젝트** 메뉴에서 **참조 추가**를 선택합니다.
2. **.Net** 탭 (SQL Server 프로젝트에 대 한**SQL Server** 탭)에서 **시스템 트랜잭션을**선택한 다음 **확인**을 선택 합니다.

   *System.object*에 대 한 참조가 프로젝트에 추가 됩니다.

트랜잭션에 데이터를 저장 하려면

- 트랜잭션을 포함 하는 using 문 내에 데이터를 저장 하는 코드를 추가 합니다. 다음 코드에서는 using 문에서 TransactionScope 개체를 만들고 인스턴스화하는 방법을 보여 줍니다.

```
Using updateTransaction As New Transactions.TransactionScope

    ' Add code to save your data here.
    ' Throw an exception to roll back the transaction.

    ' Call the Complete method to commit the transaction
    updateTransaction.Complete()
End Using
```

```
using (System.Transactions.TransactionScope updateTransaction =
    new System.Transactions.TransactionScope())
{
    // Add code to save your data here.
    // Throw an exception to roll back the transaction.

    // Call the Complete method to commit the transaction
    updateTransaction.Complete();
}
```

참조

- 데이터를 다시 데이터베이스에 저장
- 연습: 트랜잭션에 데이터 저장

연습: 트랜잭션에 데이터 저장

2020-01-06 • 15 minutes to read • Edit Online

이 연습에서는 System.Transactions 네임 스페이스를 사용 하 여 트랜잭션에 데이터를 저장 하는 방법을 보여 줍니다. 이 연습에서는 Windows Forms 응용 프로그램을 만듭니다. 데이터 소스 구성 마법사를 사용 하 여 Northwind 샘플 데이터베이스의 두 테이블에 대 한 데이터 집합을 만듭니다. Windows form에 데이터 바인딩된 컨트롤을 추가 하 고 BindingNavigator의 저장 단추에 대 한 코드를 수정 하 여 TransactionScope 내에서 데이터베이스를 업데이트 합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. Visual Studio 설치 관리자에서 SQL Server Express LocalDB는 **.net 데스크톱 개발** 워크 로드의 일부로 또는 개별 구성 요소로 설치할 수 있습니다.

2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 애플리케이션 만들기

첫 번째 단계는 **Windows Forms 응용 프로그램**을 만드는 것입니다.

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.

2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.

3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.

4. 프로젝트 이름을 **SavingDataInATransactionWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   SavingDataInATransactionWalkthrough 프로젝트가 만들어져 **솔루션 탐색기**에 추가됩니다.

데이터베이스 데이터 원본 만들기

이 단계에서는 **데이터 소스 구성 마법사** 를 사용 하 여 Northwind 샘플 데이터베이스의 `Customers` 및 `Orders` 테이블을 기반으로 데이터 원본을 만듭니다.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다.

2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성 마법사**를 시작합니다.

3. **데이터 소스 형식 선택** 화면에서 **데이터베이스**를 선택 하 고 **다음**을 선택 합니다.

4. **데이터 연결 선택** 화면에서 다음 중 하나를 수행 합니다.

   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.

     -또는-

   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작하고 Northwind 데이터베이스에 대한 연결을 만듭니다.

5. 데이터베이스에 암호가 필요 하면 중요 한 데이터를 포함 하는 옵션을 선택 하 고 **다음**을 선택 합니다.

6. **응용 프로그램 구성 파일에 연결 문자열 저장** 화면에서 **다음**을 선택 합니다.

7. **데이터베이스 개체 선택** 화면에서 **테이블** 노드를 확장 합니다.

8. `Customers` 및 `Orders` 테이블을 선택 하 고 **마침**을 선택 합니다.

   NorthwindDataSet가 프로젝트에 추가되고 `Customers` 및 `Orders` 테이블이 **데이터 원본** 창에 나타납니다.

폼에 컨트롤 추가

데이터 원본 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다.

1. **데이터 소스** 창에서 **Customers** 노드를 확장 합니다.

2. 주 **Customers** 노드를 **데이터 원본** 창에서 **Form1**으로 끌어서 놓습니다.

   DataGridView 컨트롤과 레코드 탐색에 사용되는 도구 모음인 BindingNavigator가 폼에 나타납니다. NorthwindDataSet, `CustomersTableAdapter`, BindingSource및 BindingNavigator가 구성 요소 트레이에 나타납니다.

3. 관련 **Orders** 노드 (주 **주문** 노드가 아닌, **Fax** 열 아래의 관련 자식 테이블 노드)를 **customersdatagridview**아래에 있는 폼으로 끕니다.

   DataGridView가 폼에 나타납니다. 구성 요소 트레이에 `OrdersTableAdapter` 및 BindingSource 표시 됩니다.

시스템 트랜잭션 어셈블리에 대 한 참조 추가

트랜잭션은 System.Transactions 네임스페이스를 사용합니다. system.transactions 어셈블리에 대한 프로젝트 참조는 기본적으로 추가되어 있지 않으므로 수동으로 추가해야 합니다.

System.Transactions DLL 파일에 대한 참조를 추가하려면

1. **프로젝트** 메뉴에서 **참조 추가**를 선택합니다.

2. **.Net** 탭에서 **시스템 트랜잭션** 을 선택 하 고 **확인**을 선택 합니다.

   System.Transactions에 대한 참조가 프로젝트에 추가됩니다.

BindingNavigator의 SaveItem 단추에서 코드 수정

폼에 끌어 놓은 첫 번째 테이블의 경우 코드는 기본적으로 BindingNavigator의 저장 단추 `click` 이벤트에 추가 됩니다. 추가 테이블을 업데이트하려면 코드를 수동으로 추가해야 합니다. 이 연습에서는 저장 단추의 click 이벤트 처리기에서 기존 저장 코드를 리팩터링 합니다. 또한 행을 추가 하거나 삭제 해야 하는지 여부에 따라 특정 업데이트 기능을 제공 하는 몇 가지 메서드를 추가로 만들 수 있습니다.

자동으로 생성된 저장 코드를 수정하려면

1. **Customersbindingnavigator** 에서 **저장** 단추 (플로피 디스크 아이콘이 있는 단추)를 선택 합니다.

2. `CustomersBindingNavigatorSaveItem_Click` 메서드를 다음 코드로 바꿉니다.

```
Private Sub CustomersBindingNavigatorSaveItem_Click() Handles CustomersBindingNavigatorSaveItem.Click
    UpdateData()
End Sub

Private Sub UpdateData()
    Me.Validate()
    Me.CustomersBindingSource.EndEdit()
    Me.OrdersBindingSource.EndEdit()

    Using updateTransaction As New Transactions.TransactionScope

        DeleteOrders()
        DeleteCustomers()
        AddNewCustomers()
        AddNewOrders()

        updateTransaction.Complete()
        NorthwindDataSet.AcceptChanges()
    End Using
End Sub
```

```
private void customersBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    UpdateData();
}

private void UpdateData()
{
    this.Validate();
    this.customersBindingSource.EndEdit();
    this.ordersBindingSource.EndEdit();

    using (System.Transactions.TransactionScope updateTransaction =
        new System.Transactions.TransactionScope())
    {
        DeleteOrders();
        DeleteCustomers();
        AddNewCustomers();
        AddNewOrders();

        updateTransaction.Complete();
        northwindDataSet.AcceptChanges();
    }
}
```

관련 데이터의 변경을 조정하는 순서는 다음과 같습니다.

- 자식 레코드를 삭제 합니다. 이 경우 `Orders` 테이블에서 레코드를 삭제 합니다.
- 부모 레코드를 삭제 합니다. 이 경우 `Customers` 테이블에서 레코드를 삭제 합니다.
- 부모 레코드를 삽입 합니다. 이 경우 `Customers` 테이블에 레코드를 삽입 합니다.
- 자식 레코드를 삽입 합니다. 이 경우 `Orders` 테이블에 레코드를 삽입 합니다.

기존 주문을 삭제하려면

- 다음 `DeleteOrders` 메서드를 **Form1**에 추가합니다.

```
Private Sub DeleteOrders()

    Dim deletedOrders As NorthwindDataSet.OrdersDataTable
    deletedOrders = CType(NorthwindDataSet.Orders.GetChanges(Data.DataRowState.Deleted),
        NorthwindDataSet.OrdersDataTable)

    If Not IsNothing(deletedOrders) Then
        Try
            OrdersTableAdapter.Update(deletedOrders)

        Catch ex As Exception
            MessageBox.Show("DeleteOrders Failed")
        End Try
    End If
End Sub
```

```
private void DeleteOrders()
{
    NorthwindDataSet.OrdersDataTable deletedOrders;
    deletedOrders = (NorthwindDataSet.OrdersDataTable)
        northwindDataSet.Orders.GetChanges(DataRowState.Deleted);

    if (deletedOrders != null)
    {
        try
        {
            ordersTableAdapter.Update(deletedOrders);
        }
        catch (System.Exception ex)
        {
            MessageBox.Show("DeleteOrders Failed");
        }
    }
}
```

기존 고객을 삭제하려면

- 다음 `DeleteCustomers` 메서드를 **Form1**에 추가합니다.

```
Private Sub DeleteCustomers()

    Dim deletedCustomers As NorthwindDataSet.CustomersDataTable
    deletedCustomers = CType(NorthwindDataSet.Customers.GetChanges(Data.DataRowState.Deleted),
        NorthwindDataSet.CustomersDataTable)

    If Not IsNothing(deletedCustomers) Then
        Try
            CustomersTableAdapter.Update(deletedCustomers)

        Catch ex As Exception
            MessageBox.Show("DeleteCustomers Failed" & vbCrLf & ex.Message)
        End Try
    End If
End Sub
```

```
private void DeleteCustomers()
{
    NorthwindDataSet.CustomersDataTable deletedCustomers;
    deletedCustomers = (NorthwindDataSet.CustomersDataTable)
        northwindDataSet.Customers.GetChanges(DataRowState.Deleted);

    if (deletedCustomers != null)
    {
        try
        {
            customersTableAdapter.Update(deletedCustomers);
        }
        catch (System.Exception ex)
        {
            MessageBox.Show("DeleteCustomers Failed");
        }
    }
}
```

새 고객을 추가하려면

- 다음 `AddNewCustomers` 메서드를 **Form1**에 추가합니다.

```
Private Sub AddNewCustomers()

    Dim newCustomers As NorthwindDataSet.CustomersDataTable
    newCustomers = CType(NorthwindDataSet.Customers.GetChanges(Data.DataRowState.Added),
        NorthwindDataSet.CustomersDataTable)

    If Not IsNothing(newCustomers) Then
        Try
            CustomersTableAdapter.Update(newCustomers)

        Catch ex As Exception
            MessageBox.Show("AddNewCustomers Failed" & vbCrLf & ex.Message)
        End Try
    End If
End Sub
```

```
private void AddNewCustomers()
{
    NorthwindDataSet.CustomersDataTable newCustomers;
    newCustomers = (NorthwindDataSet.CustomersDataTable)
        northwindDataSet.Customers.GetChanges(DataRowState.Added);

    if (newCustomers != null)
    {
        try
        {
            customersTableAdapter.Update(newCustomers);
        }
        catch (System.Exception ex)
        {
            MessageBox.Show("AddNewCustomers Failed");
        }
    }
}
```

새 주문을 추가하려면

- 다음 `AddNewOrders` 메서드를 **Form1**에 추가합니다.

```
Private Sub AddNewOrders()

    Dim newOrders As NorthwindDataSet.OrdersDataTable
    newOrders = CType(NorthwindDataSet.Orders.GetChanges(Data.DataRowState.Added),
        NorthwindDataSet.OrdersDataTable)

    If Not IsNothing(newOrders) Then
        Try
            OrdersTableAdapter.Update(newOrders)

        Catch ex As Exception
            MessageBox.Show("AddNewOrders Failed" & vbCrLf & ex.Message)
        End Try
    End If
End Sub
```

```
private void AddNewOrders()
{
    NorthwindDataSet.OrdersDataTable newOrders;
    newOrders = (NorthwindDataSet.OrdersDataTable)
        northwindDataSet.Orders.GetChanges(DataRowState.Added);

    if (newOrders != null)
    {
        try
        {
            ordersTableAdapter.Update(newOrders);
        }
        catch (System.Exception ex)
        {
            MessageBox.Show("AddNewOrders Failed");
        }
    }
}
```

애플리케이션 실행

F5 키를 눌러 애플리케이션을 실행합니다.

참조

- 방법: 트랜잭션을 사용하여 데이터 저장
- 데이터를 다시 데이터베이스에 저장

데이터베이스에 데이터 저장(여러 테이블)

2020-01-06 • 18 minutes to read • Edit Online

애플리케이션 개발에서 가장 일반적인 시나리오는 Windows 애플리케이션의 폼에 데이터를 표시하고 데이터를 편집한 다음 업데이트된 데이터를 데이터베이스로 다시 보내는 것입니다. 이 연습에서는 두 관련 테이블의 데이터를 표시하는 폼을 만들고, 레코드를 편집한 다음 변경 내용을 데이터베이스에 다시 저장하는 방법을 보여줍니다. 이 예에서는 Northwind 샘플 데이터베이스의 `Customers` 및 `Orders` 테이블을 사용합니다.

TableAdapter의 `Update` 메서드를 호출하여 애플리케이션의 데이터를 데이터베이스에 다시 저장할 수 있습니다. **데이터 소스** 창에서 폼으로 테이블을 끌어다 놓으면 데이터를 저장 하는 데 필요한 코드가 자동으로 추가 됩니다. 양식에 추가 된 추가 테이블에는이 코드를 수동으로 추가 해야 합니다. 이 연습에서는 둘 이상의 테이블에서 업데이트를 저장하는 코드를 추가하는 방법을 보여줍니다.

이 연습에서 설명하는 작업은 다음과 같습니다.

- 데이터 소스 구성 마법사를 사용 하 여 응용 프로그램에서 데이터 소스를 만들고 구성 합니다.
- 데이터 소스 창에서 항목의 컨트롤을 설정 합니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.
- **데이터 원본** 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만듭니다.
- 데이터 집합의 각 테이블에서 몇 개의 레코드를 수정 합니다.
- 데이터 세트의 업데이트된 데이터를 데이터베이스로 다시 보내도록 코드를 수정합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 응용 프로그램 만들기

또는 Visual Basic에 대 한 새 **Windows Forms 앱** 프로젝트를 만듭니다. C# 프로젝트 이름을 **UpdateMultipleTablesWalkthrough**로 지정합니다.

데이터 원본 만들기

이 단계에서는 **데이터 원본 구성 마법사**를 사용하여 Northwind 데이터베이스에서 데이터 원본을 만듭니다. 연결을 만들려면 Northwind 샘플 데이터베이스에 액세스해야 합니다. Northwind 샘플 데이터베이스를 설정 하는 방법에 대 한 자세한 내용은 방법: 샘플 데이터베이스 설치를 참조 하세요.

1. **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다.

   데이터 원본 창이 열립니다.

2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성 마법사**를 시작합니다.

3. **데이터 소스 형식 선택** 화면에서 **데이터베이스**를 선택 하 고 **다음**을 선택 합니다.

4. **데이터 연결 선택** 화면에서 다음 중 하나를 수행 합니다.

   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.

     -또는-

   - **새 연결**을 선택하여 **연결 추가 또는 수정** 대화 상자를 엽니다.

5. 데이터베이스에 암호가 필요 하면 중요 한 데이터를 포함 하는 옵션을 선택 하 고 **다음**을 선택 합니다.

6. **응용 프로그램 구성 파일에 연결 문자열 저장**에서 **다음**을 선택 합니다.

7. **데이터베이스 개체 선택** 화면에서 **테이블** 노드를 확장 합니다.

8. **Customers** 및 **Orders** 테이블을 선택한 다음 **마침**을 선택 합니다.

   NorthwindDataSet가 프로젝트에 추가되고 테이블이 **데이터 원본** 창에 나타납니다.

만들 컨트롤 설정

이 연습에서 `Customers` 테이블의 데이터는 개별 컨트롤에 데이터가 표시 되는 **세부 정보** 레이아웃에 있습니다. `Orders` 테이블의 데이터는 DataGridView 컨트롤에 표시 되는 **그리드** 레이아웃에 있습니다.

데이터 소스 창에서 항목에 대한 삭제 유형을 설정하려면

1. **데이터 소스** 창에서 **Customers** 노드를 확장 합니다.

2. **Customers** 노드의 컨트롤 목록에서 **세부 정보** 를 선택 하 여 **customers** 테이블의 컨트롤을 개별 컨트롤로 변경 합니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.

데이터 바인딩된 폼 만들기

데이터 원본 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다.

1. 주 **Customers** 노드를 **데이터 원본** 창에서 **Form1**으로 끌어서 놓습니다.

   설명 레이블이 있는 데이터 바인딩된 컨트롤이 레코드 탐색을 위한 도구 모음인 BindingNavigator와 함께 폼에 나타납니다. NorthwindDataSet, `CustomersTableAdapter`, BindingSource및 BindingNavigator가 구성 요소 트레이에 나타납니다.

2. 관련 **Orders** 노드를 **데이터 원본** 창에서 **Form1**으로 끌어 옵니다.

> **NOTE**
>
> **Fax** 열 아래에 있는 관련 **Orders** 노드는 **Customers** 노드의 자식 노드입니다.

DataGridView 컨트롤과 레코드 탐색에 사용되는 도구 모음인 BindingNavigator가 폼에 나타납니다. 구성 요소 트레이에 `OrdersTableAdapter` 및 BindingSource 표시 됩니다.

데이터베이스를 업데이트 하는 코드 추가

Customers 및 **Orders** TableAdapters의 `Update` 메서드를 호출하여 데이터베이스를 업데이트할 수 있습니다. 기본적으로 데이터베이스에 업데이트를 보내기 위해BindingNavigator의 **저장** 단추에 대 한 이벤트 처리기가 폼의 코드에 추가 됩니다. 이 절차에서는 업데이트를 올바른 순서로 보내도록 코드를 수정 합니다. 이렇게 하면 참조 무결성 오류가 발생 하는 가능성이 없어집니다. 또한 이 코드는 try-catch 블록에서 업데이트 호출을 래핑하여 오류 처리를 구현합니다. 애플리케이션의 요구 사항에 맞게 코드를 수정할 수 있습니다.

> **NOTE**
> 명확 하 게 하기 위해이 연습에서는 트랜잭션을 사용 하지 않습니다. 그러나 두 개 이상의 관련 테이블을 업데이트 하는 경우에는 트랜잭션 내의 모든 업데이트 논리를 포함 합니다. 트랜잭션은 변경 내용을 커밋하기 전에 데이터베이스의 모든 관련 변경 내용이 성공 하도록 하는 프로세스입니다. 자세한 내용은 트랜잭션 및 동시성을 참조 하세요.

애플리케이션에 업데이트 논리를 추가하려면

1. BindingNavigator에서 **저장** 단추를 선택 합니다. 그러면 `bindingNavigatorSaveItem_Click` 이벤트 처리기에 대 한 코드 편집기가 열립니다.

2. 관련 TableAdapter의 `Update` 메서드를 호출하도록 이벤트 처리기의 코드를 바꿉니다. 다음 코드는 먼저 각 DataRowState(Deleted, Added 및 Modified)에 대해 업데이트된 정보를 저장할 3개 임시 데이터 테이블을 만듭니다. 업데이트는 올바른 순서 대로 실행 됩니다. 이 코드는 다음과 같습니다.

```
Me.Validate()
Me.OrdersBindingSource.EndEdit()
Me.CustomersBindingSource.EndEdit()

Dim deletedOrders As NorthwindDataSet.OrdersDataTable = CType(
    NorthwindDataSet.Orders.GetChanges(Data.DataRowState.Deleted), NorthwindDataSet.OrdersDataTable)

Dim newOrders As NorthwindDataSet.OrdersDataTable = CType(
    NorthwindDataSet.Orders.GetChanges(Data.DataRowState.Added), NorthwindDataSet.OrdersDataTable)

Dim modifiedOrders As NorthwindDataSet.OrdersDataTable = CType(
    NorthwindDataSet.Orders.GetChanges(Data.DataRowState.Modified), NorthwindDataSet.OrdersDataTable)


Try
    ' Remove all deleted orders from the Orders table.
    If Not deletedOrders Is Nothing Then
        OrdersTableAdapter.Update(deletedOrders)
    End If

    ' Update the Customers table.
    CustomersTableAdapter.Update(NorthwindDataSet.Customers)

    ' Add new orders to the Orders table.
    If Not newOrders Is Nothing Then
        OrdersTableAdapter.Update(newOrders)
    End If

    ' Update all modified Orders.
    If Not modifiedOrders Is Nothing Then
        OrdersTableAdapter.Update(modifiedOrders)
    End If

    NorthwindDataSet.AcceptChanges()

Catch ex As Exception
    MsgBox("Update failed")

Finally
    If Not deletedOrders Is Nothing Then
        deletedOrders.Dispose()
    End If

    If Not newOrders Is Nothing Then
        newOrders.Dispose()
    End If

    If Not modifiedOrders Is Nothing Then
        modifiedOrders.Dispose()
    End If
End Try
```

```
this.Validate();
this.ordersBindingSource.EndEdit();
this.customersBindingSource.EndEdit();

NorthwindDataSet.OrdersDataTable deletedOrders = (NorthwindDataSet.OrdersDataTable)
    northwindDataSet.Orders.GetChanges(DataRowState.Deleted);

NorthwindDataSet.OrdersDataTable newOrders = (NorthwindDataSet.OrdersDataTable)
    northwindDataSet.Orders.GetChanges(DataRowState.Added);

NorthwindDataSet.OrdersDataTable modifiedOrders = (NorthwindDataSet.OrdersDataTable)
    northwindDataSet.Orders.GetChanges(DataRowState.Modified);

try
{
    // Remove all deleted orders from the Orders table.
    if (deletedOrders != null)
    {
        ordersTableAdapter.Update(deletedOrders);
    }

    // Update the Customers table.
    customersTableAdapter.Update(northwindDataSet.Customers);

    // Add new orders to the Orders table.
    if (newOrders != null)
    {
        ordersTableAdapter.Update(newOrders);
    }

    // Update all modified Orders.
    if (modifiedOrders != null)
    {
        ordersTableAdapter.Update(modifiedOrders);
    }

    northwindDataSet.AcceptChanges();
}

catch (System.Exception ex)
{
    MessageBox.Show("Update failed");
}

finally
{
    if (deletedOrders != null)
    {
        deletedOrders.Dispose();
    }
    if (newOrders != null)
    {
        newOrders.Dispose();
    }
    if (modifiedOrders != null)
    {
        modifiedOrders.Dispose();
    }
}
```

응용 프로그램 테스트

1. **F5**키를 누릅니다.

2. 각 테이블에 포함된 레코드 하나 이상의 데이터를 변경해 봅니다.

3. **저장** 단추를 선택합니다.

4. 데이터베이스의 값을 점검하여 변경 내용이 저장되었는지 확인합니다.

참조

- 데이터를 다시 데이터베이스에 저장

개체에서 데이터베이스로 데이터 저장

2020-01-06 • 8 minutes to read • Edit Online

개체의 값을 TableAdapter의 DBDirect 메서드 중 하나로 전달 하 여 개체의 데이터를 데이터베이스에 저장할 수 있습니다 (예: `TableAdapter.Insert`). 자세한 내용은 TableAdapter를 참조 하세요.

개체 컬렉션의 데이터를 저장 하려면 개체의 컬렉션을 반복 하 고 (예: for next loop) TableAdapter의 `DBDirect` 메서드 중 하나를 사용 하 여 각 개체의 값을 데이터베이스로 보냅니다.

기본적으로 `DBDirect` 메서드는 데이터베이스에 대해 직접 실행할 수 있는 TableAdapter에 생성 됩니다. 이러한 메서드는 직접 호출할 수 있으며 데이터베이스에 업데이트를 전송 하기 위해 변경 내용을 조정 하기 위해 DataSet 또는 DataTable 개체가 필요 하지 않습니다.

> **NOTE**
> TableAdapter를 구성 하는 경우 주 쿼리는 `DBDirect` 메서드를 만들 수 있는 충분 한 정보를 제공 해야 합니다. 예를 들어 기본 키 열이 정의 되지 않은 테이블의 데이터를 쿼리하도록 TableAdapter가 구성 된 경우 `DBDirect` 메서드를 생성 하지 않습니다.

TABLEADAPTER DBDIRECT 메서드	설명
`TableAdapter.Insert`	데이터베이스에 새 레코드를 추가 하 고 개별 열 값을 메서드 매개 변수로 전달할 수 있도록 합니다.
`TableAdapter.Update`	데이터베이스의 기존 레코드를 업데이트 합니다. `Update` 메서드는 원래 열 값과 새 열 값을 메서드 매개 변수로 사용 합니다. 원래 값을 사용 하 여 원래 레코드를 찾은 다음 새 값을 사용 하 여 해당 레코드를 업데이트 합니다. `TableAdapter.Update` 메서드는 DataSet, DataTable, DataRow또는 DataRows의 배열을 메서드 매개 변수로 사용 하 여 데이터 집합의 변경 내용을 데이터베이스에 다시 조정 하는 데도 사용 됩니다.
`TableAdapter.Delete`	메서드 매개 변수로 전달 된 원래 열 값을 기준으로 데이터베이스에서 기존 레코드를 삭제 합니다.

개체의 새 레코드를 데이터베이스에 저장 하려면

- `TableAdapter.Insert` 메서드에 값을 전달 하 여 레코드를 만듭니다.

  다음 예에서는 `currentCustomer` 개체의 값을 `TableAdapter.Insert` 메서드에 전달 하 여 `Customers` 테이블에 새 고객 레코드를 만듭니다.

```
private void AddNewCustomers(Customer currentCustomer)
{
    customersTableAdapter.Insert(
        currentCustomer.CustomerID,
        currentCustomer.CompanyName,
        currentCustomer.ContactName,
        currentCustomer.ContactTitle,
        currentCustomer.Address,
        currentCustomer.City,
        currentCustomer.Region,
        currentCustomer.PostalCode,
        currentCustomer.Country,
        currentCustomer.Phone,
        currentCustomer.Fax);
}
```

```
Private Sub AddNewCustomer(ByVal currentCustomer As Customer)

    CustomersTableAdapter.Insert(
        currentCustomer.CustomerID,
        currentCustomer.CompanyName,
        currentCustomer.ContactName,
        currentCustomer.ContactTitle,
        currentCustomer.Address,
        currentCustomer.City,
        currentCustomer.Region,
        currentCustomer.PostalCode,
        currentCustomer.Country,
        currentCustomer.Phone,
        currentCustomer.Fax)
End Sub
```

개체의 기존 레코드를 데이터베이스로 업데이트 하려면

- `TableAdapter.Update` 메서드를 호출 하 여 레코드를 업데이트 하 고 레코드를 업데이트 하기 위해 새 값을 전달 하 여 레코드를 수정 하 고 원래 값을 전달 하 여 레코드를 찾습니다.

> **NOTE**
> 개체는 `Update` 메서드에 전달 하기 위해 원래 값을 유지 해야 합니다. 이 예제에서는 `orig` 접두사가 포함 된 속성을 사용 하 여 원래 값을 저장 합니다.

다음 예에서는 `Customer` 개체의 새 값과 원래 값을 `TableAdapter.Update` 메서드로 전달 하 여 `Customers` 테이블의 기존 레코드를 업데이트 합니다.

```
private void UpdateCustomer(Customer cust)
{
    customersTableAdapter.Update(
        cust.CustomerID,
        cust.CompanyName,
        cust.ContactName,
        cust.ContactTitle,
        cust.Address,
        cust.City,
        cust.Region,
        cust.PostalCode,
        cust.Country,
        cust.Phone,
        cust.Fax,
        cust.origCustomerID,
        cust.origCompanyName,
        cust.origContactName,
        cust.origContactTitle,
        cust.origAddress,
        cust.origCity,
        cust.origRegion,
        cust.origPostalCode,
        cust.origCountry,
        cust.origPhone,
        cust.origFax);
}
```

```
Private Sub UpdateCustomer(ByVal cust As Customer)

        CustomersTableAdapter.Update(
        cust.CustomerID,
        cust.CompanyName,
        cust.ContactName,
        cust.ContactTitle,
        cust.Address,
        cust.City,
        cust.Region,
        cust.PostalCode,
        cust.Country,
        cust.Phone,
        cust.Fax,
        cust.origCustomerID,
        cust.origCompanyName,
        cust.origContactName,
        cust.origContactTitle,
        cust.origAddress,
        cust.origCity,
        cust.origRegion,
        cust.origPostalCode,
        cust.origCountry,
        cust.origPhone,
        cust.origFax)
End Sub
```

데이터베이스에서 기존 레코드를 삭제 하려면

- `TableAdapter.Delete` 메서드를 호출 하 고 원래 값을 전달 하 여 레코드를 찾는 방법으로 레코드를 삭제 합니다.

> **NOTE**
> 개체는 `Delete` 메서드에 전달 하기 위해 원래 값을 유지 해야 합니다. 이 예제에서는 `orig` 접두사가 포함 된 속성을 사용 하 여 원래 값을 저장 합니다.

다음 예에서는 `Customer` 개체의 원래 값을 `TableAdapter.Delete` 메서드로 전달 하 여 `Customers` 테이블에서 레코드를 삭제 합니다.

```
private void DeleteCustomer(Customer cust)
{
    customersTableAdapter.Delete(
        cust.origCustomerID,
        cust.origCompanyName,
        cust.origContactName,
        cust.origContactTitle,
        cust.origAddress,
        cust.origCity,
        cust.origRegion,
        cust.origPostalCode,
        cust.origCountry,
        cust.origPhone,
        cust.origFax);
}
```

```
Private Sub DeleteCustomer(ByVal cust As Customer)

    CustomersTableAdapter.Delete(
        cust.origCustomerID,
        cust.origCompanyName,
        cust.origContactName,
        cust.origContactTitle,
        cust.origAddress,
        cust.origCity,
        cust.origRegion,
        cust.origPostalCode,
        cust.origCountry,
        cust.origPhone,
        cust.origFax)
End Sub
```

.NET 보안

데이터베이스의 테이블에서 선택한 `INSERT`, `UPDATE` 또는 `DELETE` 를 수행할 수 있는 권한이 있어야 합니다.

참조

- 데이터를 다시 데이터베이스에 저장

TableAdapter DBDirect 메서드를 사용하여 데이터 저장

2020-01-06 • 16 minutes to read • Edit Online

이 연습에서는 TableAdapter의 DBDirect 메서드를 사용 하 여 데이터베이스에 대해 직접 SQL 문을 실행 하기 위한 자세한 지침을 제공 합니다. TableAdapter의 DBDirect 메서드는 데이터베이스 업데이트에 대한 상세 제어 수준을 제공합니다. 응용 프로그램에서 필요에 따라 개별 `Insert`, `Update` 및 `Delete` 메서드를 호출 하 여 특정 SQL 문 및 저장 프로시저를 실행 하는 데 사용할 수 있습니다 .이 메서드는 업데이트, 삽입 및 삭제 문을 모두 한 번 호출 하는 오버 로드 된 `Update` 메서드와는 반대로 수행 합니다.

이 연습에서는 다음 작업을 수행하는 방법을 배웁니다.

- 새 **Windows Forms 애플리케이션**을 만듭니다.
- 데이터 소스 구성 마법사를 사용 하 여 데이터 집합을 만들고 구성 합니다.
- **데이터 원본** 창에서 항목을 끌어오는 경우 폼에 만들어질 컨트롤을 선택합니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.
- **데이터 원본** 창에서 폼으로 항목을 끌어 데이터 바인딩된 폼을 만듭니다.
- 데이터베이스에 직접 액세스 하 고 삽입, 업데이트 및 삭제를 수행 하는 메서드를 추가 합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 애플리케이션 만들기

첫 번째 단계는 **Windows Forms 응용 프로그램**을 만드는 것입니다.

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.

4. 프로젝트 이름을 **TableAdapterDbDirectMethodsWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   TableAdapterDbDirectMethodsWalkthrough 프로젝트가 만들어져 **솔루션 탐색기**에 추가됩니다.

데이터베이스에서 데이터 원본 만들기

이 단계에서는 **데이터 원본 구성 마법사**를 사용하여 Northwind 샘플 데이터베이스의 `Region` 테이블을 기반으로 하는 데이터 원본을 만듭니다. 연결을 만들려면 Northwind 샘플 데이터베이스에 액세스해야 합니다. Northwind 샘플 데이터베이스를 설정 하는 방법에 대 한 자세한 내용은 방법: 샘플 데이터베이스 설치를 참조 하세요.

데이터 소스를 만들려면

1. **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다.

   데이터 원본 창이 열립니다.

2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성 마법사**를 시작합니다.

3. **데이터 소스 형식 선택** 화면에서 **데이터베이스**를 선택 하 고 **다음**을 선택 합니다.

4. **데이터 연결 선택** 화면에서 다음 중 하나를 수행 합니다.

   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.

     -또는-

   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.

5. 데이터베이스에 암호가 필요 하면 중요 한 데이터를 포함 하는 옵션을 선택 하 고 **다음**을 선택 합니다.

6. **응용 프로그램 구성 파일에 연결 문자열 저장** 화면에서 **다음**을 선택 합니다.

7. **데이터베이스 개체 선택** 화면에서 **테이블** 노드를 확장 합니다.

8. `Region` 테이블을 선택 하 고 **마침**을 선택 합니다.

   NorthwindDataSet가 프로젝트에 추가되고 `Region` 테이블이 **데이터 원본** 창에 나타납니다.

폼에 데이터를 표시 하는 컨트롤 추가

데이터 원본 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만듭니다.

Windows form에서 데이터 바인딩된 컨트롤을 만들려면 주 **지역** 노드를 **데이터 소스** 창에서 폼으로 끌어 옵니다.

DataGridView 컨트롤과 레코드 탐색에 사용되는 도구 모음인 BindingNavigator가 폼에 나타납니다. NorthwindDataSet, `RegionTableAdapter`, BindingSource및 BindingNavigator가 구성 요소 트레이에 나타납니다.

개별 **TableAdapter DbDirect** 메서드를 호출하는 단추를 추가하려면

1. Button 컨트롤 3개를 **도구 상자**에서 **Form1**의 **RegionDataGridView** 아래로 끌어 옵니다.

2. 각 단추에 대해 다음 **이름** 및 **텍스트** 속성을 설정합니다.

이름	텍스트
`InsertButton`	삽입
`UpdateButton`	업데이트

이름	텍스트
`DeleteButton`	**삭제**

데이터베이스에 새 레코드를 삽입하는 코드를 추가하려면

1. **InsertButton** 를 선택 하 여 click 이벤트에 대 한 이벤트 처리기를 만들고 코드 편집기에서 폼을 엽니다.
2. `InsertButton_Click` 이벤트 처리기를 다음 코드로 바꿉니다.

```
Private Sub InsertButton_Click() Handles InsertButton.Click

    Dim newRegionID As Integer = 5
    Dim newRegionDescription As String = "NorthEastern"

    Try
        RegionTableAdapter1.Insert(newRegionID, newRegionDescription)

    Catch ex As Exception
        MessageBox.Show("Insert Failed")
    End Try

    RefreshDataset()
End Sub


Private Sub RefreshDataset()
    Me.RegionTableAdapter1.Fill(Me.NorthwindDataSet1._Region)
End Sub
```

```
private void InsertButton_Click(object sender, EventArgs e)
{
    Int32 newRegionID = 5;
    String newRegionDescription = "NorthEastern";

    try
    {
        regionTableAdapter1.Insert(newRegionID, newRegionDescription);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Insert Failed");
    }
    RefreshDataset();
}


private void RefreshDataset()
{
    this.regionTableAdapter1.Fill(this.northwindDataSet1.Region);
}
```

데이터베이스에서 레코드를 업데이트하는 코드를 추가하려면

1. **UpdateButton**을 두 번 클릭하여 클릭 이벤트에 대한 이벤트 처리기를 만들고 코드 편집기에서 폼을 엽니다.
2. `UpdateButton_Click` 이벤트 처리기를 다음 코드로 바꿉니다.

```
Private Sub UpdateButton_Click() Handles UpdateButton.Click

    Dim newRegionID As Integer = 5

    Try
        RegionTableAdapter1.Update(newRegionID, "Updated Region Description", 5, "NorthEastern")

    Catch ex As Exception
        MessageBox.Show("Update Failed")
    End Try

    RefreshDataset()
End Sub
```

```
private void UpdateButton_Click(object sender, EventArgs e)
{
    Int32 newRegionID = 5;

    try
    {
        regionTableAdapter1.Update(newRegionID, "Updated Region Description", 5, "NorthEastern");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Update Failed");
    }
    RefreshDataset();
}
```

데이터베이스에서 레코드를 삭제 하는 코드를 추가 하려면

1. **Deletebutton** 을 선택 하 여 click 이벤트에 대 한 이벤트 처리기를 만들고 코드 편집기에서 폼을 엽니다.
2. `DeleteButton_Click` 이벤트 처리기를 다음 코드로 바꿉니다.

```
Private Sub DeleteButton_Click() Handles DeleteButton.Click

    Try
        RegionTableAdapter1.Delete(5, "Updated Region Description")

    Catch ex As Exception
        MessageBox.Show("Delete Failed")
    End Try

    RefreshDataset()
End Sub
```

```
private void DeleteButton_Click(object sender, EventArgs e)
{
    try
    {
        regionTableAdapter1.Delete(5, "Updated Region Description");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Delete Failed");
    }
    RefreshDataset();
}
```

애플리케이션 실행

- **F5 키** 를 선택 하 여 응용 프로그램을 실행 합니다.
- **삽입** 단추를 선택 하 고 새 레코드가 표에 표시 되는지 확인 합니다.
- **업데이트** 단추를 선택 하 고 레코드가 표에서 업데이트 되었는지 확인 합니다.
- **삭제** 단추를 선택 하 고 레코드가 표에서 제거 되었는지 확인 합니다.

다음 단계

응용 프로그램 요구 사항에 따라 데이터 바인딩된 폼을 만든 후 몇 단계를 더 수행 해야 할 수도 있습니다. 이 연습에서 보완할 수 있는 사항은 다음과 같습니다.

- 폼에 검색 기능을 추가합니다.
- **데이터 원본** 창에서 **마법사로 데이터 세트 구성**을 선택하여 추가 테이블을 데이터 세트에 추가합니다. 관련 노드를 폼으로 끌어 관련 데이터를 표시하는 컨트롤을 추가할 수 있습니다. 자세한 내용은 데이터 집합의 관계를 참조 하세요.

참조

- 데이터를 다시 데이터베이스에 저장

데이터 세트를 XML로 저장

2020-01-06 • 2 minutes to read • Edit Online

데이터 집합에서 사용 가능한 XML 메서드를 호출 하 여 데이터 집합의 XML 데이터에 액세스 합니다. XML 형식으로 데이터를 저장 하기 위해 DataSet의 WriteXml 메서드 또는 GetXml 메서드를 호출할 수 있습니다.

GetXml 메서드를 호출 하면 XML 형식의 데이터 집합에 있는 모든 데이터 테이블의 데이터를 포함 하는 문자열이 반환 됩니다.

WriteXml 메서드를 호출 하면 XML 형식 데이터를 지정한 파일로 보냅니다.

데이터 집합의 데이터를 변수에 XML로 저장 하려면

- 합니다 GetXml 메서드가 반환 되는 String합니다. String 형식의 변수를 선언 하 고 GetXml 메서드의 결과에 할당 합니다.

```
Dim xmlData As String = NorthwindDataSet.GetXml()
```

```
string xmlData = northwindDataSet.GetXml();
```

데이터 집합의 데이터를 파일에 XML로 저장 하려면

- WriteXml 메서드에 몇 가지 오버 로드가 있습니다. 변수를 선언 하 고 파일을 저장할 올바른 경로를 할당 합니다. 다음 코드에서는 데이터를 파일에 저장 하는 방법을 보여 줍니다.

```
Dim filePath As String = "ENTER A VALID FILEPATH"
NorthwindDataSet.WriteXml(filePath)
```

```
string filePath = "ENTER A VALID FILEPATH";
northwindDataSet.WriteXml(filePath);
```

참조

- 데이터를 다시 데이터베이스에 저장

데이터 세트 쿼리

2020-01-06 • 12 minutes to read • Edit Online

데이터 집합에서 특정 레코드를 검색 하려면 DataTable에서 `FindBy` 메서드를 사용 하 고, 테이블의 Rows 컬렉션을 반복 하는 고유한 foreach 문을 작성 하거나 LINQ to DataSet를 사용 합니다.

데이터 집합 대/소문자 구분

데이터 집합 내에서 테이블 및 열 이름은 기본적으로 대/소문자를 구분 하지 않습니다. 즉, "Customers" 라는 데이터 집합의 테이블을 "customers" 라고도 합니다. 이는 SQL Server을 포함 하 여 많은 데이터베이스의 명명 규칙과 일치 합니다. SQL Server에서 기본 동작은 데이터 요소의 이름이 대/소문자만 구분 될 수 없다는 것입니다.

> **NOTE**
> 데이터 집합과 달리 XML 문서는 대/소문자를 구분 하므로 스키마에 정의 된 데이터 요소의 이름은 대/소문자를 구분 합니다. 예를 들어 스키마 프로토콜을 사용 하면 스키마에서 "Customers" 라는 테이블과 "customers" 라는 다른 테이블을 정의할 수 있습니다. 이 경우 대/소문자만 다른 요소가 포함 된 스키마를 사용 하 여 dataset 클래스를 생성할 때 이름 충돌이 발생할 수 있습니다.

하지만 대/소문자 구분은 데이터 집합 내에서 데이터를 해석 하는 방법에 대 한 요인이 될 수 있습니다. 예를 들어 데이터 집합 테이블에서 데이터를 필터링 하는 경우 비교 시 대/소문자를 구분 하는지 여부에 따라 검색 조건이 다른 결과를 반환할 수 있습니다. 데이터 집합의 CaseSensitive 속성을 설정 하 여 필터링, 검색 및 정렬의 대/소문자 구분 여부를 제어할 수 있습니다. 데이터 집합의 모든 테이블은 기본적으로이 속성의 값을 상속 합니다. 테이블의 CaseSensitive 속성을 설정 하 여 개별 테이블에 대해이 속성을 재정의할 수 있습니다.

데이터 테이블에서 특정 행 찾기

기본 키 값을 사용 하 여 형식화 된 데이터 집합에서 행을 찾으려면

- 행을 찾으려면 테이블의 기본 키를 사용 하는 강력한 형식의 `FindBy` 메서드를 호출 합니다.

  다음 예에서는 `CustomerID` 열이 `Customers` 테이블의 기본 키입니다. 즉, 생성 된 `FindBy` 메서드가 `FindByCustomerID`됩니다. 이 예제에서는 생성 된 `FindBy` 메서드를 사용 하 여 변수에 특정 DataRow를 할당 하는 방법을 보여 줍니다.

```
NorthwindDataSet.CustomersRow customersRow =
    northwindDataSet1.Customers.FindByCustomerID("ALFKI");
```

```
Dim customersRow As NorthwindDataSet.CustomersRow
customersRow = NorthwindDataSet1.Customers.FindByCustomerID("ALFKI")
```

기본 키 값을 사용 하 여 형식화 되지 않은 데이터 집합에서 행을 찾으려면

- DataRowCollection 컬렉션의 Find 메서드를 호출 하 여 기본 키를 매개 변수로 전달 합니다.

  다음 예제에서는 `foundRow` 라는 새 행을 선언 하 고이를 Find 메서드의 반환 값에 할당 하는 방법을 보여 줍니다. 기본 키가 있는 경우에는 열 인덱스 1의 내용이 메시지 상자에 표시 됩니다.

```
string s = "primaryKeyValue";
DataRow foundRow = dataSet1.Tables["AnyTable"].Rows.Find(s);

if (foundRow != null)
{
    MessageBox.Show(foundRow[0].ToString());
}
else
{
    MessageBox.Show("A row with the primary key of " + s + " could not be found");
}
```

```
Dim s As String = "primaryKeyValue"
Dim foundRow As DataRow = DataSet1.Tables("AnyTable").Rows.Find(s)

If foundRow IsNot Nothing Then
    MsgBox(foundRow(1).ToString())
Else
    MsgBox("A row with the primary key of " & s & " could not be found")
End If
```

열 값으로 행 찾기

열의 값을 기준으로 행을 찾으려면

- 데이터 테이블은 Select 메서드에 전달 된 식을 기반으로 DataRow의 배열을 반환 하는 Select 메서드를 사용 하 여 생성 됩니다. 유효한 식을 만드는 방법에 대 한 자세한 내용은 Expression 속성에 대 한 페이지의 "식 구문" 섹션을 참조 하세요.

  다음 예제에서는 DataTable의 Select 메서드를 사용 하 여 특정 행을 찾는 방법을 보여 줍니다.

```
DataRow[] foundRows;
foundRows = dataSet1.Tables["Customers"].Select("CompanyName Like 'A%'");
```

```
Dim foundRows() As Data.DataRow
foundRows = DataSet1.Tables("Customers").Select("CompanyName Like 'A%'")
```

관련 레코드 액세스

데이터 집합의 테이블이 관련 된 경우 DataRelation 개체를 사용 하 여 다른 테이블에서 관련 레코드를 사용할 수 있습니다. 예를 들어 `Customers` 및 `Orders` 테이블을 포함 하는 데이터 집합을 사용할 수 있습니다.

DataRelation 개체를 사용 하 여 부모 테이블에서 DataRow의 GetChildRows 메서드를 호출 하 여 관련 레코드를 찾을 수 있습니다. 이 메서드는 관련 된 자식 레코드의 배열을 반환 합니다. 또는 자식 테이블에서 DataRow의 GetParentRow 메서드를 호출할 수 있습니다. 이 메서드는 부모 테이블에서 단일 DataRow 반환 합니다.

이 페이지에서는 형식화 된 데이터 집합을 사용 하는 예제를 제공 합니다. 형식화 되지 않은 데이터 집합에서 관계를 탐색 하는 방법은 DataRelations 탐색을 참조 하세요.

> **NOTE**
>
> Windows Forms 응용 프로그램에서 작업 중이 고 데이터 바인딩 기능을 사용 하 여 데이터를 표시 하는 경우 디자이너에서 생성 한 폼을 사용 하면 응용 프로그램에 충분 한 기능을 제공할 수 있습니다. 자세한 내용은 Visual Studio에서 데이터에 컨트롤 바인딩을 참조 하세요. 특히 데이터 집합의 관계를 참조 하세요.

다음 코드 예제에서는 형식화 된 데이터 집합에서 위쪽 및 아래쪽 관계를 탐색 하는 방법을 보여 줍니다. 이 코드 예제에서는 형식화 된 DataRows (`NorthwindDataSet.OrdersRow`) 및 생성 된 FindBy*PrimaryKey* (`FindByCustomerID`) 메서드를 사용 하 여 원하는 행을 찾고 관련 레코드를 반환 합니다. 예제는 다음을 수행 하는 경우에만 올바르게 컴파일 및 실행 됩니다.

- `Customers` 테이블이 있는 `NorthwindDataSet` 이라는 데이터 집합의 인스턴스입니다.
- `Orders` 테이블입니다.
- 두 테이블을 관련 `FK_Orders_Customers` 이라는 관계입니다.

또한 반환 될 레코드에 대 한 데이터를 두 테이블에 채워야 합니다.

선택한 부모 레코드의 자식 레코드를 반환 하려면

- 특정 `Customers` 데이터 행의 GetChildRows 메서드를 호출 하 고 `Orders` 테이블에서 행의 배열을 반환 합니다.

```
string custID = "ALFKI";
NorthwindDataSet.OrdersRow[] orders;

orders = (NorthwindDataSet.OrdersRow[])northwindDataSet.Customers.
    FindByCustomerID(custID).GetChildRows("FK_Orders_Customers");

MessageBox.Show(orders.Length.ToString());
```

```
Dim customerID As String = "ALFKI"
Dim orders() As NorthwindDataSet.OrdersRow

orders = CType(NorthwindDataSet.Customers.FindByCustomerID(customerID).
    GetChildRows("FK_Orders_Customers"), NorthwindDataSet.OrdersRow())

MessageBox.Show(orders.Length.ToString())
```

선택한 자식 레코드의 부모 레코드를 반환 하려면

- 특정 `Orders` 데이터 행의 GetParentRow 메서드를 호출 하 고 `Customers` 테이블에서 단일 행을 반환 합니다.

```
int orderID = 10707;
NorthwindDataSet.CustomersRow customer;

customer = (NorthwindDataSet.CustomersRow)northwindDataSet.Orders.
    FindByOrderID(orderID).GetParentRow("FK_Orders_Customers");

MessageBox.Show(customer.CompanyName);
```

```
Dim orderID As Integer = 10707
Dim customer As NorthwindDataSet.CustomersRow

customer = CType(NorthwindDataSet.Orders.FindByOrderID(orderID).
    GetParentRow("FK_Orders_Customers"), NorthwindDataSet.CustomersRow)

MessageBox.Show(customer.CompanyName)
```

참조

- Visual Studio의 데이터 세트 도구

n 계층 데이터 세트에 유효성 검사 추가

2020-01-16 • 15 minutes to read • Edit Online

N 계층 솔루션으로 분리 된 데이터 집합에 유효성 검사를 추가 하는 것은 기본적으로 단일 파일 데이터 집합에 유효성 검사를 추가 하는 것과 같습니다 (단일 프로젝트의 데이터 집합). 데이터에 대 한 유효성 검사를 수행 하기 위한 제안 된 위치는 데이터 테이블의 ColumnChanging 및/또는 RowChanging 이벤트 중입니다.

데이터 집합은 데이터 집합에 있는 데이터 테이블의 열 및 행 변경 이벤트에 사용자 코드를 추가할 수 있는 partial 클래스를 만드는 기능을 제공 합니다. N 계층 솔루션의 데이터 집합에 코드를 추가 하는 방법에 대 한 자세한 내용은 n 계층 응용 프로그램의 데이터 집합에 코드 추가및 n 계층 응용 프로그램에서 Tableadapter에 코드 추가를 참조 하세요. Partial 클래스에 대 한 자세한 내용은 방법: 클래스를 부분 클래스 (클래스 디자이너) 또는 partial 클래스 및 메서드로분할을 참조 하세요.

> **NOTE**
> **데이터 집합 프로젝트** 속성을 설정 하 여 tableadapter의 데이터 집합을 분리 하는 경우 프로젝트의 기존 부분 데이터 집합 클래스는 자동으로 이동 되지 않습니다. 기존 부분 데이터 집합 클래스는 데이터 집합 프로젝트로 수동으로 이동 해야 합니다.

> **NOTE**
> 데이터 집합 디자이너는 ColumnChanging 및 RowChanging 이벤트 C# 에 대해 이벤트 처리기를 자동으로 만들지 않습니다. 이벤트 처리기를 수동으로 만들고 이벤트 처리기를 기본 이벤트에 연결 해야 합니다. 다음 절차에서는 Visual Basic와 C#모두에서 필요한 이벤트 처리기를 만드는 방법을 설명 합니다.

개별 열에 대 한 변경 내용 유효성 검사

ColumnChanging 이벤트를 처리 하 여 개별 열에 있는 값의 유효성을 검사 합니다. 열 값이 수정 되 면 발생 하는 ColumnChanging 이벤트입니다. **데이터 세트 디자이너**에서 원하는 열을 두 번 클릭 하 여 ColumnChanging 이벤트에 대 한 이벤트 처리기를 만듭니다.

처음으로 열을 두 번 클릭 하면 디자이너에서 ColumnChanging 이벤트에 대 한 이벤트 처리기를 생성 합니다. 특정 열을 테스트 하는 `If...Then` 문도 생성 됩니다. 예를 들어 Northwind Orders 테이블에서 **RequiredDate** 열을 두 번 클릭 하면 다음 코드가 생성 됩니다.

```
Private Sub OrdersDataTable_ColumnChanging(ByVal sender As System.Object, ByVal e As
System.Data.DataColumnChangeEventArgs) Handles Me.ColumnChanging
    If (e.Column.ColumnName = Me.RequiredDateColumn.ColumnName) Then
        ' Add validation code here.
    End If
End Sub
```

> **NOTE**
> 프로젝트 C# 에서 데이터 세트 디자이너는 데이터 집합 및 데이터 집합의 개별 테이블에 대 한 partial 클래스를 만듭니다. 데이터 세트 디자이너는 Visual Basic C# 처럼 ColumnChanging 및 RowChanging 이벤트에 대 한 이벤트 처리기를 자동으로 만들지 않습니다. 프로젝트 C# 에서 이벤트를 처리 하 고 메서드를 기본 이벤트에 후크 하는 메서드를 수동으로 생성 해야 합니다. 다음 절차에서는 Visual Basic와 C#모두에서 필요한 이벤트 처리기를 만드는 단계를 제공 합니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

개별 열 값을 변경 하는 동안 유효성 검사를 추가 하려면

1. **솔루션 탐색기**에서 *.xsd* 파일을 두 번 클릭 하 여 데이터 집합을 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.

2. 유효성을 검사할 열을 두 번 클릭 합니다. 이 작업은 ColumnChanging 이벤트 처리기를 만듭니다.

   > **NOTE**
   > 데이터 세트 디자이너는 C# 이벤트에 대 한 이벤트 처리기를 자동으로 만들지 않습니다. 에서 C# 이벤트를 처리 하는 데 필요한 코드는 다음 섹션에 포함 되어 있습니다. `SampleColumnChangingEvent` 생성 된 다음 EndInit 메서드의 ColumnChanging 이벤트에 연결 됩니다.

3. 코드를 추가 하 여 `e.ProposedValue`에 응용 프로그램의 요구 사항을 충족 하는 데이터가 포함 되어 있는지 확인 합니다. 제안 된 값이 허용 되지 않는 경우 열에 오류가 포함 되어 있음을 나타내는 열을 설정 합니다.

   다음 코드 예에서는 **Quantity** 열에 0 보다 큰 값이 포함 되어 있는지 확인 합니다. **Quantity** 가 0 보다 작거나 같으면 열이 오류로 설정 됩니다. **Quantity** 가 0 보다 크면 `Else` 절에서 오류를 지웁니다. 열 변경 이벤트 처리기의 코드는 다음과 유사 합니다.

```
If (e.Column.ColumnName = Me.QuantityColumn.ColumnName) Then
    If CType(e.ProposedValue, Short) <= 0 Then
        e.Row.SetColumnError(e.Column, "Quantity must be greater than 0")
    Else
        e.Row.SetColumnError(e.Column, "")
    End If
End If
```

```
// Add this code to the DataTable partial class.

public override void EndInit()
{
    base.EndInit();
    // Hook up the ColumnChanging event
    // to call the SampleColumnChangingEvent method.
    ColumnChanging += SampleColumnChangingEvent;
}

public void SampleColumnChangingEvent(object sender, System.Data.DataColumnChangeEventArgs e)
{
    if (e.Column.ColumnName == QuantityColumn.ColumnName)
    {
        if ((short)e.ProposedValue <= 0)
        {
            e.Row.SetColumnError("Quantity", "Quantity must be greater than 0");
        }
        else
        {
            e.Row.SetColumnError("Quantity", "");
        }
    }
}
```

전체 행의 변경 내용 유효성 검사

RowChanging 이벤트를 처리 하 여 전체 행의 값에 대 한 유효성을 검사 합니다. RowChanging 이벤트는 모든 열의 값이 커밋될 때 발생 합니다. 한 열의 값이 다른 열의 값에 의존 하는 경우 RowChanging 이벤트에서 유효성을 검사 해야 합니다. 예를 들어 Northwind의 Orders 테이블에서 OrderDate 및 RequiredDate을 고려 합니다.

주문을 입력 하는 경우 유효성 검사를 수행 하면 OrderDate의 RequiredDate 이전에 주문이 입력 되지 않습니다. 이 예에서는 RequiredDate 열과 OrderDate 열 모두에 대 한 값을 비교 해야 하므로 개별 열 변경의 유효성을 검사 하는 것은 적합 하지 않습니다.

데이터 세트 디자이너테이블의 제목 표시줄에서 테이블 이름을 두 번 클릭 하 여 RowChanging 이벤트에 대 한 이벤트 처리기를 만듭니다.

전체 행을 변경 하는 동안 유효성 검사를 추가 하려면

1. **솔루션 탐색기**에서 *.xsd* 파일을 두 번 클릭 하 여 데이터 집합을 엽니다. 자세한 내용은 연습: 데이터 세트 디자이너에서 데이터 집합 만들기를 참조 하세요.

2. 디자이너에서 데이터 테이블의 제목 표시줄을 두 번 클릭 합니다.

   Partial 클래스는 `RowChanging` 이벤트 처리기를 사용 하 여 생성 되 고 코드 편집기에서 열립니다.

   > **NOTE**
   > 데이터 세트 디자이너는 프로젝트의 C# RowChanging 이벤트에 대 한 이벤트 처리기를 자동으로 만들지 않습니다. RowChanging 이벤트를 처리 하 고 코드를 실행 한 다음 테이블의 초기화 메서드에서 이벤트를 후크 하는 메서드를 만들어야 합니다.

3. Partial 클래스 선언 내에 사용자 코드를 추가 합니다.

4. 다음 코드에서는 RowChanging 이벤트 중에 유효성을 검사 하는 사용자 코드를 추가할 위치를 보여 줍니다. 이 C# 예제에는 이벤트 처리기 메서드를 `OrdersRowChanging` 이벤트에 후크 하는 코드도 포함 되어 있습니다.

```
Partial Class OrdersDataTable
    Private Sub OrdersDataTable_OrdersRowChanging(ByVal sender As System.Object, ByVal e As
OrdersRowChangeEvent) Handles Me.OrdersRowChanging
        ' Add logic to validate columns here.
        If e.Row.RequiredDate <= e.Row.OrderDate Then
            ' Set the RowError if validation fails.
            e.Row.RowError = "Required Date cannot be on or before the OrderDate"
        Else
            ' Clear the RowError when validation passes.
            e.Row.RowError = ""
        End If
    End Sub
End Class
```

```
partial class OrdersDataTable
{
    public override void EndInit()
    {
        base.EndInit();
        // Hook up the event to the
        // RowChangingEvent method.
        OrdersRowChanging += RowChangingEvent;
    }

    public void RowChangingEvent(object sender, OrdersRowChangeEvent e)
    {
        // Perform the validation logic.
        if (e.Row.RequiredDate <= e.Row.OrderDate)
        {
            // Set the row to an error when validation fails.
            e.Row.RowError = "Required Date cannot be on or before the OrderDate";
        }
        else
        {
            // Clear the RowError if validation passes.
            e.Row.RowError = "";
        }
    }
}
```

참조

- N 계층 데이터 애플리케이션 개요
- 연습: N 계층 데이터 애플리케이션 만들기
- 데이터 세트의 데이터 유효성 검사

n 계층 애플리케이션에서 데이터 세트에 코드 추가

2020-01-06 • 5 minutes to read • Edit Online

*DatasetName*에 코드를 추가 하는 대신 데이터 집합에 대 한 partial 클래스 파일을 만들고 코드를 추가 하 여 데이터 집합의 기능을 확장할 수 있습니다. 데이터 집합 디자이너 파일). Partial 클래스를 사용 하면 특정 클래스에 대 한 코드를 여러 물리적 파일로 분할할 수 있습니다. 자세한 내용은 partial 또는 partial 클래스 및 메서드를 참조 하세요.

데이터 집합을 정의 하는 코드는 형식화 된 데이터 집합에서 데이터 집합 정의가 변경 될 때마다 생성 됩니다. 이 코드는 데이터 집합의 구성을 수정 하는 마법사를 실행 하는 동안 변경 작업을 수행할 때도 생성 됩니다. 데이터 집합을 다시 생성 하는 동안 코드가 삭제 되지 않도록 하려면 데이터 집합의 partial 클래스 파일에 코드를 추가 합니다.

데이터 집합 및 TableAdapter 코드를 분리 한 후에는 기본적으로 각 프로젝트의 불연속 클래스 파일이 생성 됩니다. 원본 프로젝트에는 TableAdapter 코드를 포함 하는 *DatasetName* (또는 *DatasetName.Designer.cs*) 라는 파일이 있습니다. **데이터 집합 프로젝트** 속성에 지정 된 프로젝트에는 이름이 *DatasetName* (또는 *DatasetName.DataSet.Designer.cs*) 인 파일이 있습니다. 이 파일에는 데이터 집합 코드가 포함 됩니다.

> **NOTE**
> **데이터 집합 프로젝트** 속성을 설정 하 여 데이터 집합 및 tableadapter를 분리 하는 경우 프로젝트의 기존 부분 데이터 집합 클래스는 자동으로 이동 되지 않습니다. 기존 데이터 집합 partial 클래스는 데이터 집합 프로젝트로 수동으로 이동 해야 합니다.

> **NOTE**
> 유효성 검사 코드를 추가 해야 하는 경우 형식화 된 데이터 집합은 ColumnChanging 및 RowChanging 이벤트 처리기를 생성 하는 기능을 제공 합니다. 자세한 내용은 n 계층 데이터 집합에 유효성 검사 추가를 참조 하세요.

N 계층 응용 프로그램의 데이터 집합에 코드를 추가 하려면

1. *.Xsd* 파일이 포함 된 프로젝트를 찾습니다.
2. **.Xsd** 파일을 선택 하 여 데이터 집합을 엽니다.
3. 코드를 추가 하려는 데이터 테이블 (제목 표시줄의 테이블 이름)을 마우스 오른쪽 단추로 클릭 한 다음 **코드 보기**를 선택 합니다.

   Partial 클래스가 만들어지고 코드 편집기에서 열립니다.
4. Partial 클래스 선언 내에 코드를 추가 합니다.

   다음 예제에서는 NorthwindDataSet의 CustomersDataTable에 코드를 추가할 위치를 보여 줍니다.

```
Partial Public Class CustomersDataTable
    ' Add code here to add functionality
    ' to the CustomersDataTable.
End Class
```

```
partial class CustomersDataTable
{
    // Add code here to add functionality
    // to the CustomersDataTable.
}
```

참조

- N 계층 데이터 애플리케이션 개요
- n 계층 애플리케이션에서 TableAdapter에 코드 추가
- TableAdapter 만들기 및 구성
- 계층적 업데이트 개요
- Visual Studio의 데이터 집합 도구

n 계층 애플리케이션에서 TableAdapter에 코드 추가

2020-01-16 • 5 minutes to read • Edit Online

TableAdapter의 partial 클래스 파일을 만들고 코드를 추가 (*DatasetName* 파일에 코드를 추가 하는 대신) 하 여 tableadapter의 기능을 확장할 수 있습니다. Partial 클래스를 사용 하면 특정 클래스에 대 한 코드를 여러 물리적 파일로 분할할 수 있습니다. 자세한 내용은 부분 또는 부분 (형식)을 참조 하세요.

TableAdapter를 정의 하는 코드는 데이터 집합에서 TableAdapter가 변경 될 때마다 생성 됩니다. 이 코드는 TableAdapter의 구성을 수정 하는 마법사를 실행 하는 동안 변경 된 경우에도 생성 됩니다. TableAdapter를 다시 생성 하는 동안 코드가 삭제 되지 않도록 하려면 TableAdapter의 partial 클래스 파일에 코드를 추가 합니다.

데이터 집합 및 TableAdapter 코드를 분리 한 후에는 기본적으로 각 프로젝트의 불연속 클래스 파일이 생성 됩니다. 원본 프로젝트에는 TableAdapter 코드를 포함 하는 *DatasetName* (또는 *DatasetName.Designer.cs*) 라는 파일이 있습니다. **데이터 집합 프로젝트** 속성에 지정 된 프로젝트에는 데이터 집합 코드를 포함 하는 *DatasetName* (또는 *DatasetName.DataSet.Designer.cs*) 라는 파일이 있습니다.

> **NOTE**
> **데이터 세트 프로젝트** 속성을 설정하여 데이터 세트와 TableAdapters를 분리할 때는 프로젝트의 기존 부분 데이터 세트 클래스가 자동으로 이동되지 않습니다. 기존 부분 데이터 집합 클래스는 데이터 집합 프로젝트로 수동으로 이동 해야 합니다.

> **NOTE**
> 데이터 집합은 유효성 검사가 필요할 때 ColumnChanging 및 RowChanging 이벤트 처리기를 생성 하는 기능을 제공 합니다. 자세한 내용은 n 계층 데이터 집합에 유효성 검사 추가를 참조 하세요.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

N 계층 응용 프로그램에서 TableAdapter에 사용자 코드를 추가 하려면

1. *.Xsd* 파일이 포함 된 프로젝트를 찾습니다.
2. *.Xsd* 파일을 두 번 클릭 하 여 **데이터 세트 디자이너**을 엽니다.
3. 코드를 추가할 TableAdapter를 마우스 오른쪽 단추로 클릭 한 다음 **코드 보기**를 선택 합니다.

   Partial 클래스가 만들어지고 코드 편집기에서 열립니다.
4. Partial 클래스 선언 내에 코드를 추가 합니다.
5. 다음 예제에서는 `NorthwindDataSet`의 `CustomersTableAdapter`에 코드를 추가할 위치를 보여 줍니다.

```
Partial Public Class CustomersTableAdapter
    ' Add code here to add functionality
    ' to the CustomersTableAdapter.
End Class
```

```
public partial class CustomersTableAdapter
{
    // Add code here to add functionality
    // to the CustomersTableAdapter.
}
```

참조

- N 계층 데이터 애플리케이션 개요
- n 계층 애플리케이션에서 데이터 세트에 코드 추가
- TableAdapter 만들기 및 구성
- 계층적 업데이트 개요

데이터 세트 및 TableAdapter를 다른 프로젝트로 분리

2020-01-06 • 7 minutes to read • Edit Online

형식화 된 데이터 집합은 tableadapter 및 데이터 집합 클래스가 개별 프로젝트로 생성 될 수 있도록 향상 되었습니다. 이를 통해 응용 프로그램 계층을 신속 하 게 분리 하 고 n 계층 데이터 응용 프로그램을 생성할 수 있습니다.

다음 절차에서는 **데이터 세트 디자이너** 를 사용 하 여 생성 된 TableAdapter 코드가 포함 된 프로젝트와 별도의 프로젝트에 데이터 집합 코드를 생성 하는 프로세스에 대해 설명 합니다.

별도의 데이터 집합 및 Tableadapter

TableAdapter 코드에서 데이터 집합 코드를 분리 하는 경우 데이터 집합 코드가 포함 된 프로젝트를 현재 솔루션에 배치 해야 합니다. 이 프로젝트를 현재 솔루션에 배치 하지 않으면 **속성** 창의 **데이터 집합 프로젝트** 목록에서 사용할 수 없습니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

데이터 집합을 다른 프로젝트로 분리 하려면

1. 데이터 집합 (*.xsd* 파일)이 포함 된 솔루션을 엽니다.

   > **NOTE**
   > 솔루션에 데이터 집합 코드를 분리 하려는 프로젝트가 포함 되어 있지 않으면 프로젝트를 만들거나 솔루션에 기존 프로젝트를 추가 합니다.

2. **솔루션 탐색기** 에서 형식화 된 데이터 집합 파일 (*.xsd* 파일)을 두 번 클릭 하 여 **데이터 세트 디자이너**에서 데이터 집합을 엽니다.

3. **데이터 세트 디자이너**빈 영역을 선택 합니다.

4. **속성** 창에서 **데이터 집합 프로젝트** 노드를 찾습니다.

5. **데이터 집합 프로젝트** 목록에서 데이터 집합 코드를 생성 하려는 프로젝트의 이름을 선택 합니다.

   데이터 집합 코드를 생성 하려는 프로젝트를 선택 하면 **데이터 집합 파일** 속성이 기본 파일 이름으로 채워집니다. 필요한 경우이 이름을 변경할 수 있습니다. 또한 특정 디렉터리에 데이터 집합 코드를 생성 하려는 경우 **프로젝트 폴더** 속성을 폴더 이름으로 설정할 수 있습니다.

   > **NOTE**
   > **데이터 집합 프로젝트** 속성을 설정 하 여 데이터 집합 및 tableadapter를 분리 하는 경우 프로젝트의 기존 부분 데이터 집합 클래스는 자동으로 이동 되지 않습니다. 기존 부분 데이터 집합 클래스는 데이터 집합 프로젝트로 수동으로 이동 해야 합니다.

6. 데이터 집합을 저장 합니다.

데이터 집합 코드는 **데이터 집합 프로젝트** 속성에서 선택한 프로젝트로 생성 되 고, **TableAdapter** 코드는 현재 프로젝트로 생성 됩니다.

데이터 집합 및 TableAdapter 코드를 분리 한 후에는 기본적으로 각 프로젝트의 불연속 클래스 파일이 생성 됩니다. 원본 프로젝트에는 TableAdapter 코드를 포함 하는 *DatasetName* (또는 *DatasetName.Designer.cs*) 라는 파일이 있습니다. **데이터 집합 프로젝트** 속성에 지정 된 프로젝트에는 데이터 집합 코드를 포함 하는 *DatasetName* (또는 *DatasetName.DataSet.Designer.cs*) 라는 파일이 있습니다.

> **NOTE**
> 생성 된 클래스 파일을 보려면 데이터 집합 또는 TableAdapter 프로젝트를 선택 합니다. 그런 다음 **솔루션 탐색기**에서 **모든 파일 표시**를 선택 합니다.

참조

- N 계층 데이터 애플리케이션 개요
- 연습: N 계층 데이터 애플리케이션 만들기
- 계층적 업데이트
- Visual Studio에서 데이터 액세스
- ADO.NET

Visual Studio에서 데이터에 컨트롤 바인딩

2020-01-06 • 9 minutes to read • Edit Online

데이터를 컨트롤에 바인딩하여 애플리케이션 사용자에게 데이터를 표시할 수 있습니다. **데이터 소스** 창에서 Visual Studio의 화면에 있는 컨트롤 또는 디자인 화면으로 항목을 끌어 이러한 데이터 바인딩된 컨트롤을 만들 수 있습니다.

이 항목에서는 데이터 바인딩된 컨트롤을 만드는 데 사용할 수 있는 데이터 소스에 대해 설명합니다. 또한 데이터 바인딩과 관련된 일반적인 작업에 대해서도 설명합니다. 데이터 바인딩된 컨트롤을 만드는 방법에 대 한 자세한 내용은 Visual studio에서 데이터에 컨트롤 Windows Forms 바인딩 및 visual studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.

데이터 원본

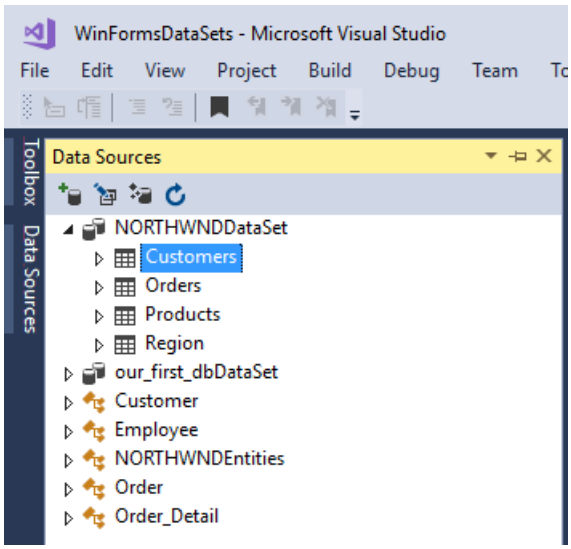
데이터 바인딩 컨텍스트에서 데이터 원본은 사용자 인터페이스에 바인딩할 수 있는 메모리의 데이터를 나타냅니다. 실제로 데이터 소스는 Entity Framework 클래스, 데이터 집합, .NET 프록시 개체에서 캡슐화 되는 서비스 끝점, LINQ to SQL 클래스 또는 .NET 개체나 컬렉션 일 수 있습니다. 일부 데이터 소스의 경우 **데이터 원본** 창에서 항목을 끌어 데이터에 바인딩된 컨트롤을 만들도록 설정하지만 모든 데이터 원본이 그러한 것은 아닙니다. 다음 표에서는 어떠한 데이터 소스가 지원되는지 보여 줍니다.

데이터 소스	WINDOWS FORMS 디자이너에서의 끌어서 놓기 지원	WPF 디자이너에서의 끌어서 놓기 지원	SILVERLIGHT 디자이너에서의 끌어서 놓기 지원
데이터 집합	예	예	아니요
엔터티 데이터 모델	예[1]	예	예
LINQ to SQL 클래스	No[2]	No[2]	No[2]
WCF Data Services, WCF 서비스, 웹 서비스 등의 서비스	예	예	예
개체	예	예	예
SharePoint	예	예	예

1. **엔터티 데이터 모델** 마법사를 사용 하 여 모델을 생성 한 다음 해당 개체를 디자이너로 끌어 옵니다.
2. LINQ to SQL 클래스는 **데이터 원본** 창에 표시되지 않습니다. 그러나 LINQ to SQL 클래스에 기반하는 개체 데이터 소스를 새로 만든 다음 해당 개체를 디자이너로 끌어 와 데이터 바인딩된 컨트롤을 만들 수는 있습니다. 자세한 내용은 연습: LINQ to SQL 클래스 만들기 (O-R 디자이너)를 참조 하세요.

데이터 소스 창

데이터 원본은 프로젝트에서 **데이터 원본** 창의 항목으로 사용할 수 있습니다. 이 창은 양식 디자인 화면이 프로젝트의 활성 창이 면 표시 되 고 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 프로젝트를 열 수 있습니다. 이 창에서 항목을 끌어 기본 데이터에 바인딩된 컨트롤을 만들 수 있으며, 마우스 오른쪽 단추를 클릭 하여 데이터 소스를 구성할 수도 있습니다.



데이터 원본 창에 표시되는 각 데이터 형식의 경우 디자이너로 항목을 끌 때 기본 컨트롤이 만들어집니다. **데이터 소스** 창에서 항목을 끌기 전에 만들어진 컨트롤을 변경할 수 있습니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.

컨트롤을 데이터에 바인딩하는 것과 관련된 작업

다음 표에서는 컨트롤을 데이터에 바인딩하기 위해 수행 하는 가장 일반적인 몇 가지 작업을 보여 줍니다.

작업	추가 정보
데이터 원본 창을 엽니다.	편집기에서 디자인 화면을 열고 > **데이터 원본 보기** 를 선택 합니다.
프로젝트에 데이터 원본을 추가합니다.	새 데이터 소스 추가
데이터 원본 창의 항목을 디자이너로 끌 때 만들어지는 컨트롤을 설정합니다.	데이터 소스 창에서 끌어올 때 만들 컨트롤 설정
데이터 원본 창의 항목과 연결되는 컨트롤 목록을 수정합니다.	데이터 소스 창에 사용자 지정 컨트롤 추가
데이터 바인딩된 컨트롤을 만듭니다.	Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩 Visual Studio에서 데이터에 WPF 컨트롤 바인딩
개체 또는 컬렉션에 바인딩합니다.	Visual Studio에서 개체 바인딩
UI에 표시 되는 데이터를 필터링 합니다.	Windows Forms 애플리케이션에서 데이터 필터링 및 정렬
컨트롤의 캡션을 사용자 지정 합니다.	Visual Studio에서 데이터 바인딩된 컨트롤에 대한 캡션을 만드는 방식 사용자 지정

참조

- .NET용 Visual Studio 데이터 도구
- Windows Forms 데이터 바인딩

데이터 소스 창에서 끌어올 때 만들 컨트롤 설정

2020-01-06 • 9 minutes to read • Edit Online

데이터 소스 창에서 WPF 디자이너나 Windows Forms designer로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다. **데이터 소스** 창의 각 항목에는 디자이너로 끌어올 때 생성 되는 기본 컨트롤이 있습니다. 그러나 다른 컨트롤을 만들도록 선택할 수 있습니다.

데이터 테이블 또는 개체에 대해 만들 컨트롤 설정

데이터 **소스** 창에서 데이터 테이블이 나 개체를 나타내는 항목을 끌기 전에 모든 데이터를 하나의 컨트롤로 표시 하거나 개별 컨트롤에 각 열 또는 속성을 표시 하도록 선택할 수 있습니다.

이 컨텍스트에서 용어 *개체* 는 사용자 지정 비즈니스 개체, 엔터티 (엔터티 데이터 모델) 또는 서비스에서 반환 된 개체를 참조 합니다.

데이터 테이블이 나 개체에 대해 만들 컨트롤을 설정 하려면

1. **WPF** 디자이너나 **Windows Forms** designer가 열려 있어야 합니다.
2. **데이터 소스** 창에서 설정 하려는 데이터 테이블이 나 개체를 나타내는 항목을 선택 합니다.

   > **TIP**
   > **데이터 소스** 창이 열려 있지 않은 경우 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 열 수 있습니다.

3. 항목에 대 한 드롭다운 메뉴를 클릭 하 고 메뉴에서 다음 항목 중 하나를 클릭 합니다.

   - 각 데이터 필드를 별도의 컨트롤에 표시 하려면 **세부 정보**를 클릭 합니다. 데이터 항목을 디자이너로 끌어 오면이 작업은 각 컨트롤에 대 한 레이블과 함께 부모 데이터 테이블이 나 개체의 각 열 또는 속성에 대해 서로 다른 데이터 바인딩된 컨트롤을 만듭니다.
   - 모든 데이터를 단일 컨트롤로 표시 하려면 목록에서 WPF 응용 프로그램의 **DataGrid** 또는 **목록** , Windows Forms 응용 프로그램에서 **DataGridView** 와 같은 다른 컨트롤을 선택 합니다.

   사용 가능한 컨트롤 목록은 열려 있는 디자이너, 프로젝트에서 대상으로 하는 .NET 버전 및 **도구 상자**에 데이터 바인딩을 지 원하는 사용자 지정 컨트롤을 추가 했는지 여부에 따라 달라 집니다. 만들려는 컨트롤이 사용 가능한 컨트롤 목록에 없으면 목록에 컨트롤을 추가할 수 있습니다. 자세한 내용은 데이터 소스 창에 사용자 지정 컨트롤 추가를 참조 하세요.

   데이터 **소스** 창에서 데이터 테이블 또는 개체의 컨트롤 목록에 추가할 수 있는 사용자 지정 Windows Forms 컨트롤을 만드는 방법에 대 한 자세한 내용은 복합 데이터 바인딩을 지 원하는 Windows Forms 사용자 정의 컨트롤 만들기를 참조 하세요.

데이터 열 또는 속성에 대해 만들 컨트롤 설정

열 또는 개체의 속성을 나타내는 항목을 **데이터 소스** 창에서 디자이너로 끌기 전에 컨트롤을 만들 수 있도록 설정할 수 있습니다.

열 또는 속성에 대해 만들 컨트롤을 설정 하려면

1. **WPF** 디자이너나 **Windows Forms** designer가 열려 있어야 합니다.
2. **데이터 원본** 창에서 원하는 테이블 또는 개체를 확장 하 여 해당 열 또는 속성을 표시 합니다.
3. 만들 컨트롤을 설정 하려는 각 열 또는 속성을 선택 합니다.

4. 열 또는 속성에 대 한 드롭다운 메뉴를 클릭 한 다음 항목을 디자이너로 끌 때 만들려는 컨트롤을 선택 합니다.

   사용 가능한 컨트롤 목록은 열려 있는 디자이너, 프로젝트에서 대상으로 하는 .NET 버전 및 **도구 상자**에 추가한 데이터 바인딩을 지 원하는 사용자 지정 컨트롤에 따라 다릅니다. 만들려는 컨트롤이 사용 가능한 컨트롤 목록에 있으면 목록에 컨트롤을 추가할 수 있습니다. 자세한 내용은 데이터 소스 창에 사용자 지정 컨트롤 추가를 참조 하세요.

   데이터 **소스** 창에서 데이터 열 또는 속성에 대 한 컨트롤 목록에 추가할 수 있는 사용자 지정 컨트롤을 만드는 방법을 알아보려면 단순 데이터 바인딩을 지 원하는 Windows Forms 사용자 정의 컨트롤 만들기를 참조 하세요.

   열 또는 속성에 대 한 컨트롤을 만들지 않으려면 드롭다운 메뉴에서 **없음** 을 선택 합니다. 이 방법은 부모 테이블이 나 개체를 디자이너에 끌어 놓을 수 있지만 특정 열 또는 속성을 포함 하지 않으려는 경우에 유용 합니다.

참조

- Visual Studio에서 데이터에 컨트롤 바인딩

데이터 소스 창에 사용자 지정 컨트롤 추가

2020-01-06 • 12 minutes to read • Edit Online

데이터 소스 창에서 디자인 화면으로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 때 사용자가 만드는 컨트롤의 형식을 선택할 수 있습니다. 창의 각 항목에는 선택할 수 있는 컨트롤을 표시 하는 드롭다운 목록이 있습니다. 각 항목과 연결 된 컨트롤 집합은 항목의 데이터 형식에 따라 결정 됩니다. 만들려는 컨트롤이 목록에 표시 되지 않는 경우이 항목의 지침에 따라 컨트롤을 목록에 추가할 수 있습니다.

데이터 소스 창에서 항목에 대해 만들 데이터 바인딩된 컨트롤을 선택 하는 방법에 대 한 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.

바인딩 가능한 컨트롤 목록 사용자 지정

데이터 소스 창에서 특정 데이터 형식의 항목에 대해 사용 가능한 컨트롤 목록에서 컨트롤을 추가 하거나 제거 하려면 다음 단계를 수행 합니다.

데이터 형식에 대해 나열할 컨트롤을 선택 하려면

1. WPF 디자이너나 Windows Forms 디자이너이 열려 있는지 확인 합니다.
2. **데이터 소스** 창에서 창에 추가한 데이터 원본의 일부인 항목을 클릭 한 다음 해당 항목에 대 한 드롭다운 메뉴를 클릭 합니다.

   > **TIP**
   > 데이터 소스 창이 열려 있지 않으면 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 엽니다.

3. 드롭다운 메뉴에서 **사용자 지정**을 클릭 합니다. 다음 대화 상자 중 하나가 열립니다.
   - **Windows Forms 디자이너** 열려 있으면 **옵션** 대화 상자의 **데이터 UI 사용자 지정** 페이지가 열립니다. 자세한 내용은 데이터 UI 사용자 지정 옵션 대화 상자를 참조 하세요.
   - **WPF Designer** 가 열려 있으면 **컨트롤 바인딩 사용자 지정** 대화 상자가 열립니다.
4. 대화 상자의 **데이터 형식** 드롭다운 목록에서 데이터 형식을 선택 합니다.
   - 테이블 또는 개체의 컨트롤 목록을 사용자 지정 하려면 **[목록]** 을 선택 합니다.
   - 테이블의 열 또는 개체의 속성에 대 한 컨트롤 목록을 사용자 지정 하려면 기본 데이터 저장소에서 열 또는 속성의 데이터 형식을 선택 합니다.
   - 사용자 정의 셰이프가 있는 데이터 개체를 표시 하는 컨트롤 목록을 사용자 지정 하려면 **[기타]** 를 선택 합니다. 예를 들어 응용 프로그램에 특정 개체의 둘 이상의 속성 데이터를 표시 하는 사용자 지정 컨트롤이 있는 경우 **[기타]** 를 선택 합니다.
5. **연결 된 컨트롤** 상자에서 선택한 데이터 형식에 대해 사용할 수 있는 각 컨트롤을 선택 하거나 목록에서 제거 하려는 컨트롤의 선택을 취소 합니다.

   > **NOTE**
   > 선택 하려는 컨트롤이 **연결 된 컨트롤** 상자에 표시 되지 않는 경우 컨트롤을 목록에 추가 해야 합니다. 자세한 내용은 연결 된 컨트롤 추가를 참조 하세요.

6. **확인**을 클릭합니다.

7. **데이터 소스** 창에서 하나 이상의 컨트롤과 연결 된 데이터 형식의 항목을 클릭 한 다음 항목에 대 한 드롭 다운 메뉴를 클릭 합니다.

   연결 된 컨트롤 상자에서 선택한 컨트롤이 이제 해당 항목의 드롭다운 메뉴에 표시 됩니다.

연결 된 컨트롤 추가

컨트롤을 데이터 형식과 연결 하려고 하지만 컨트롤이 **연결 된 컨트롤** 상자에 표시 되지 않는 경우 컨트롤을 목록에 추가 해야 합니다. 컨트롤이 현재 솔루션 또는 참조 된 어셈블리에 있어야 합니다. 또한 **도구 상자** 에서 사용할 수 있어야 하 고 컨트롤의 데이터 바인딩 동작을 지정 하는 특성이 있어야 합니다.

연결 된 컨트롤 목록에 컨트롤을 추가 하려면 다음을 수행 합니다.

1. **도구 상자** 를 마우스 오른쪽 단추로 클릭 하 고 **항목 선택**을 선택 하 여 원하는 컨트롤을 **도구 상자** 에 추가 합니다.

   컨트롤에는 다음 특성 중 하나가 있어야 합니다.

특성	설명
DefaultBindingPropertyAttribute	TextBox같은 데이터의 단일 열 또는 속성을 표시 하는 간단한 컨트롤에 대해이 특성을 구현 합니다.
ComplexBindingPropertiesAttribute	DataGridView와 같은 데이터 목록 또는 테이블을 표시 하는 컨트롤에 대해이 특성을 구현 합니다.
LookupBindingPropertiesAttribute	데이터의 목록이 나 테이블을 표시 하는 컨트롤에 대해이 특성을 구현 하 고 ComboBox같은 단일 열 이나 속성을 제공 해야 합니다.

2. Windows Forms의 경우 **옵션** 대화 상자에서 **데이터 UI 사용자 지정** 페이지를 엽니다. 또는 WPF의 경우 **컨트롤 바인딩 사용자 지정** 대화 상자를 엽니다. 자세한 내용은 데이터 형식에 대 한 바인딩 가능한 컨트롤 목록 사용자 지정을 참조 하세요.

3. 연결 된 **컨트롤** 상자에 방금 **도구 상자** 에 추가한 컨트롤이 표시 됩니다.

> **NOTE**
> 현재 솔루션 또는 참조 되는 어셈블리 내에 있는 컨트롤만 연결 된 컨트롤의 목록에 추가할 수 있습니다. 또한 컨트롤은 위의 표에 있는 데이터 바인딩 특성 중 하나를 구현 해야 합니다. 데이터 소스 창에서 사용할 수 없는 사용자 지정 컨트롤에 데이터를 바인딩하려면 컨트롤을 **도구 상자** 에서 디자인 화면으로 끌어 온 다음 바인딩할 항목을 **데이터 소스** 창에서 컨트롤로 끌어 옵니다.

참조

- Visual Studio에서 데이터에 컨트롤 바인딩
- 데이터 UI 사용자 지정 옵션 대화 상자

Visual Studio에서 데이터에 WPF 컨트롤 바인딩

2020-01-06 • 18 minutes to read • Edit Online

데이터를 WPF 컨트롤에 바인딩하여 애플리케이션 사용자에게 데이터를 표시할 수 있습니다. 이러한 데이터 바인딩된 컨트롤을 만들려면 **데이터 소스** 창에서 Visual Studio의 WPF Designer로 항목을 끌어 옵니다. 이 항목에서는 데이터 바인딩된 WPF 애플리케이션을 만드는 데 사용할 수 있는 가장 일반적인 몇 가지 작업, 도구 및 클래스에 대해 설명합니다.

Visual Studio에서 데이터 바인딩된 컨트롤을 만드는 방법에 대 한 일반적인 내용은 Visual studio에서 데이터에 컨트롤 바인딩을 참조 하세요. WPF 데이터 바인딩에 대한 자세한 내용은 데이터 바인딩 개요를 참조하세요.

WPF 컨트롤을 데이터에 바인딩하는 것과 관련된 작업

다음 표에서는 **데이터 원본** 창에서 WPF Designer로 항목을 끌어서 수행할 수 있는 작업을 보여 줍니다.

작업	추가 정보
새 데이터 바인딩된 컨트롤을 만듭니다. 기존 컨트롤을 데이터에 바인딩합니다.	데이터 세트로 WPF 컨트롤 바인딩
부모-자식 관계로 관련 데이터를 표시하는 컨트롤을 만듭니다. 그러면 사용자가 어느 한 컨트롤에서 부모 데이터 레코드를 선택하면 선택한 부모 레코드의 관련 자식 데이터가 다른 한 컨트롤에 표시됩니다.	WPF 애플리케이션에서 관련 데이터 표시
한 테이블의 정보를 다른 테이블의 외래 키 필드 값을 기반으로 표시하는 *조회 테이블*을 만듭니다.	WPF 애플리케이션에서 조회 테이블 만들기
데이터베이스의 이미지에 컨트롤을 바인딩합니다.	데이터베이스의 그림에 컨트롤 바인딩

유효한 놓기 대상

데이터 원본 창의 항목을 WPF Designer의 유효한 놓기 대상으로 끌어 올 수 있습니다. 유효한 놓기 대상에는 크게 두 가지 종류 즉, 컨테이너와 컨트롤이 있습니다. 컨테이너는 일반적으로 컨트롤을 포함하는 사용자 인터페이스 요소입니다. 예를 들어, 표는 컨테이너이므로 창입니다.

생성된 XAML 및 코드

데이터 소스 창에서 WPF Designer로 항목을 끌어 오면 Visual Studio에서 새 데이터 바인딩된 컨트롤을 정의 하거나 기존 컨트롤을 데이터 소스에 바인딩하는 XAML 생성 합니다. 일부 데이터 원본의 경우에는 데이터 소스를 데이터로 채우는 코드에 코드를 생성 하는 코드도 생성 됩니다.

다음 표에서는 Visual Studio에서 **데이터** 소스 창에 있는 각 데이터 소스 형식에 대해 생성 하는 XAML 및 코드를 보여 줍니다.

데이터 소스	데이터 소스에 컨트롤을 바인딩하는 XAML을 생성합니다.	데이터 소스를 데이터로 채우는 코드를 생성합니다.
데이터 집합	예	예

데이터 소스	데이터 소스에 컨트롤을 바인딩하는 XAML을 생성합니다.	데이터 소스를 데이터로 채우는 코드를 생성합니다.
엔터티 데이터 모델	예	예
서비스	예	아니요
개체	예	아니요

데이터 집합

데이터 소스 창에서 디자이너로 테이블이 나 열을 끌어 오면 Visual Studio에서 다음을 수행 하는 XAML을 생성 합니다.

- 항목을 끌어 온 컨테이너의 리소스에 데이터 세트과 새 CollectionViewSource를 추가합니다. CollectionViewSource는 데이터 세트에 있는 데이터를 탐색하고 표시하는 데 사용할 수 있는 개체입니다.
- 컨트롤에 대한 데이터 바인딩을 만듭니다. 디자이너의 기존 컨트롤로 항목을 끌면 XAML이 컨트롤을 항목에 바인딩합니다. 항목을 컨테이너로 끌어 오면 XAML은 끌어 온 항목에 대해 선택한 컨트롤을 만들고 컨트롤을 항목에 바인딩합니다. 이 컨트롤은 새로운 Grid 내에 만들어집니다.

또한 Visual Studio에서는 코드 숨김 파일에 대해 다음과 같은 변경 작업도 수행합니다.

- 이 컨트롤이 들어 있는 Loaded 요소에 대한 UI 이벤트 처리기를 만듭니다. 이 이벤트 처리기는 테이블을 데이터로 채우고 컨테이너의 리소스에서 CollectionViewSource를 검색한 다음 첫 번째 데이터 항목을 현재 항목으로 설정합니다. Loaded 이벤트 처리기가 이미 있는 경우 Visual Studio는 기존 이벤트 처리기에이 코드를 추가 합니다.

엔터티 데이터 모델

데이터 소스 창에서 디자이너로 엔터티 또는 엔터티 속성을 끌어 오면 Visual Studio에서 다음을 수행 하는 XAML을 생성 합니다.

- 항목을 끌어 온 컨테이너의 리소스에 새 CollectionViewSource를 추가합니다. CollectionViewSource는 엔터티에 있는 데이터를 탐색하고 표시하는 데 사용할 수 있는 개체입니다.
- 컨트롤에 대한 데이터 바인딩을 만듭니다. 디자이너의 기존 컨트롤로 항목을 끌면 XAML이 컨트롤을 항목에 바인딩합니다. 항목을 컨테이너로 끌면 XAML는 끌어 온 항목에 대해 선택한 컨트롤을 만들고 컨트롤을 항목에 바인딩합니다. 이 컨트롤은 새로운 Grid 내에 만들어집니다.

또한 Visual Studio에서는 코드 숨김 파일에 대해 다음과 같은 변경 작업도 수행합니다.

- 디자이너로 끌어 온 엔터티(또는 디자이너로 끌어 온 속성이 들어 있는 엔터티)에 대한 쿼리를 반환하는 새 메서드를 추가합니다. 새 메서드에는 `Get<EntityName>Query` 이름이 있습니다. 여기서 `\<EntityName>` 는 엔터티의 이름입니다.
- 이 컨트롤이 들어 있는 Loaded 요소에 대한 UI 이벤트 처리기를 만듭니다. 이벤트 처리기는 `Get<EntityName>Query` 메서드를 호출 하 여 엔터티를 데이터로 채우고 컨테이너의 리소스에서 CollectionViewSource를 검색 한 다음 첫 번째 데이터 항목을 현재 항목으로 만듭니다. Loaded 이벤트 처리기가 이미 있는 경우 Visual Studio는 기존 이벤트 처리기에이 코드를 추가 합니다.

서비스

데이터 소스 창에서 디자이너로 서비스 개체 또는 속성을 끌어 오면 Visual Studio에서 데이터 바인딩된 컨트롤을 만들거나 기존 컨트롤을 개체나 속성에 바인딩하는 XAML을 생성 합니다. 그러나 Visual Studio는 프록시 서비스 개체를 데이터로 채우는 코드를 생성 하지 않습니다. 이 코드를 직접 작성해야 합니다. 이 작업을 수행 하는 방법을 보여 주는 예제는 WCF 데이터 서비스에 WPF 컨트롤 바인딩을 참조 하세요.

Visual Studio에서는 다음을 수행하는 XAML을 생성합니다.

- 항목을 끌어 온 컨테이너의 리소스에 새 CollectionViewSource를 추가합니다. CollectionViewSource는 서비스에서 반환하는 개체에 있는 데이터를 탐색하고 표시하는 데 사용할 수 있는 개체입니다.
- 컨트롤에 대한 데이터 바인딩을 만듭니다. 디자이너의 기존 컨트롤로 항목을 끌면 XAML이 컨트롤을 항목에 바인딩합니다. 항목을 컨테이너로 끌면 XAML는 끌어 온 항목에 대해 선택한 컨트롤을 만들고 컨트롤을 항목에 바인딩합니다. 이 컨트롤은 새로운 Grid 내에 만들어집니다.

개체

데이터 소스 창에서 디자이너로 개체 또는 속성을 끌어 오면 Visual Studio에서 데이터 바인딩된 컨트롤을 만들거나 기존 컨트롤을 개체나 속성에 바인딩하는 XAML을 생성 합니다. 그러나 Visual Studio에서는 데이터를 사용하 여 개체를 채우는 코드를 생성 하지 않습니다. 이 코드를 직접 작성해야 합니다.

> **NOTE**
> 사용자 지정 클래스는 public 이어야 하며 기본적으로 매개 변수가 없는 생성자가 있어야 합니다. 구문에서 "점"을 포함 하는 중첩 된 클래스 일 수 없습니다. 자세한 내용은 WPF에 대 한 XAML 및 사용자 지정 클래스를 참조 하세요.

Visual Studio는 다음을 수행 하는 XAML을 생성 합니다.

- 항목을 끌어 온 컨테이너의 리소스에 새 CollectionViewSource를 추가합니다. CollectionViewSource는 개체에 있는 데이터를 탐색하고 표시하는 데 사용할 수 있는 개체입니다.
- 컨트롤에 대한 데이터 바인딩을 만듭니다. 디자이너의 기존 컨트롤로 항목을 끌면 XAML이 컨트롤을 항목에 바인딩합니다. 항목을 컨테이너로 끌어 오면 XAML은 끌어 온 항목에 대해 선택한 컨트롤을 만들고 컨트롤을 항목에 바인딩합니다. 이 컨트롤은 새로운 Grid 내에 만들어집니다.

참조

- Visual Studio에서 데이터에 컨트롤 바인딩

데이터 세트로 WPF 컨트롤 바인딩

2020-01-06 • 23 minutes to read • Edit Online

이 연습에서는 데이터 바인딩된 컨트롤을 포함 하는 WPF 응용 프로그램을 만듭니다. 이러한 컨트롤은 데이터 세트에서 캡슐화된 제품 레코드에 바인딩됩니다. 또한 제품을 탐색 하 고 제품 레코드에 대 한 변경 내용을 저장 하는 단추도 추가 합니다.

이 연습에서는 다음 작업을 수행합니다.

- AdventureWorksLT 샘플 데이터베이스의 데이터에서 생성되는 데이터 세트 및 WPF 애플리케이션을 만듭니다.
- **데이터 원본** 창에서 WPF 디자이너의 창으로 데이터 테이블을 끌어 데이터 바인딩된 컨트롤 집합을 만듭니다.
- 제품 레코드를 앞뒤로 탐색하는 데 사용할 단추를 만듭니다.
- 사용자가 제품 레코드에 대해 적용한 변경 내용을 데이터 테이블 및 기본 데이터 소스에 저장하는 단추를 만듭니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

전제 조건

이 연습을 완료하려면 다음 구성 요소가 필요합니다.

- Visual Studio
- AdventureWorks Light (AdventureWorksLT) 예제 데이터베이스가 연결 된 실행 중인 SQL Server 또는 SQL Server Express 인스턴스에 액세스 합니다. CodePlex 보관에서 AdventureWorksLT 데이터베이스를 다운로드할 수 있습니다.

또한 다음 개념에 대한 지식은 연습을 완료하는 데 반드시 필요하지는 않지만 사전에 파악해 두면 유용할 수 있습니다.

- 데이터 세트 및 TableAdapter. 자세한 내용은 Visual Studio 데이터 집합 도구 및 TableAdapters를 참조하세요.
- WPF 데이터 바인딩. 자세한 내용은 데이터 바인딩 개요를 참조하세요.

프로젝트를 만듭니다.

제품 레코드를 표시 하는 새 WPF 프로젝트를 만듭니다.

1. Visual Studio를 엽니다.
2. **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 선택합니다.
3. **Visual Basic** 또는 **Visual C#** 을 확장한 다음, **Windows**를 선택합니다.
4. **WPF 앱** 프로젝트 템플릿을 선택 합니다.

5. **이름** 상자에 **AdventureWorksProductsEditor** 를 입력 한 다음 **확인**을 선택 합니다.

1. Visual Studio를 엽니다.
2. 시작 창에서 **새 프로젝트 만들기**를 선택합니다.
3. C# **WPF 앱** 프로젝트 템플릿을 검색 하 고 단계에 따라 프로젝트를 만들고 **AdventureWorksProductsEditor**프로젝트 이름을 지정 합니다.

AdventureWorksProductsEditor 프로젝트가 만들어집니다.

응용 프로그램에 대 한 데이터 집합 만들기

데이터 바인딩된 컨트롤을 만들려면 먼저 애플리케이션의 데이터 모델을 정의하고 **데이터 원본** 창에 추가해야 합니다. 이 연습에서는 데이터 모델로 사용할 데이터 세트을 만듭니다.

1. **데이터** 메뉴에서 **데이터 소스 표시**를 클릭합니다.

   데이터 원본 창이 열립니다.

2. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.

   데이터 원본 구성 마법사가 열립니다.

3. **데이터 원본 형식 선택** 페이지에서 **데이터베이스**를 선택한 후, **다음**을 클릭합니다.

4. **데이터베이스 모델 선택** 페이지에서 **데이터 세트**를 선택한 후, **다음**을 클릭합니다.

5. **데이터 연결 선택** 페이지에서 다음 옵션 중 하나를 선택합니다.

   - AdventureWorksLT 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택한 후, **다음**을 클릭합니다.
   - **새 연결**을 클릭하고 AdventureWorksLT 데이터베이스에 대한 연결을 만듭니다.

6. **애플리케이션 구성 파일에 연결 문자열 저장** 페이지에서 **예, 다음으로 연결을 저장합니다.** 확인란을 선택한 후, **다음**을 클릭합니다.

7. **데이터베이스 개체 선택** 페이지에서 **테이블**을 확장한 다음, **Product(SalesLT)** 테이블을 선택합니다.

8. **마침**을 클릭합니다.

   Visual Studio는 프로젝트에 새 `AdventureWorksLTDataSet.xsd` 파일을 추가 하 고 해당 **adventureworksltdataset.xsd 파일이** 항목을 **데이터 소스** 창에 추가 합니다. `AdventureWorksLTDataSet.xsd` 파일은 `AdventureWorksLTDataSet` 이라는 형식화 된 데이터 집합과 `ProductTableAdapter`라는 TableAdapter를 정의 합니다. 이 연습 뒷부분에서 `ProductTableAdapter`를 사용하여 데이터 세트에 데이터를 채우고 변경 내용을 데이터베이스에 다시 저장합니다.

9. 프로젝트를 빌드합니다.

TableAdapter의 기본 fill 메서드를 편집 합니다.

데이터 세트을 데이터로 채우려면 `Fill`의 `ProductTableAdapter` 메서드를 사용합니다. 기본적으로 `Fill` 메서드는 Product 테이블의 모든 데이터 행을 `ProductDataTable`의 `AdventureWorksLTDataSet`에 채웁니다. 이 메서드가 행 하위 집합만 반환하도록 수정할 수 있습니다. 이 연습에서는 사진이 있는 제품의 행만 반환하도록 `Fill` 메서드를 수정합니다.

1. **솔루션 탐색기**에서 *AdventureWorksLTDataSet.xsd* 파일을 두 번 클릭합니다.

   데이터 세트 디자이너가 열립니다.

2. 디자이너에서 **채우기, GetData()** 쿼리를 마우스 오른쪽 단추로 클릭하고 **구성**을 선택합니다.

   TableAdapter 구성 마법사가 열립니다.

3. **SQL 문을 입력하세요.** 페이지의 텍스트 상자에서 `SELECT` 문 뒤에 다음 WHERE 절을 추가합니다.

```
WHERE ThumbnailPhotoFileName <> 'no_image_available_small.gif'
```

4. **마침**을 클릭합니다.

사용자 인터페이스 정의

WPF 디자이너에서 XAML을 수정하여 창에 여러 단추를 추가합니다. 이 연습 뒷부분에서 사용자가 이러한 단추를 사용해 제품 레코드를 스크롤하고 변경 내용을 저장할 수 있도록 하는 코드를 추가합니다.

1. **솔루션 탐색기**에서 *MainWindow.xaml*을 두 번 클릭합니다.

   WPF 디자이너에서 창이 열립니다.

2. 디자이너의 XAML 뷰에서 `<Grid>` 태그 사이에 다음 코드를 추가합니다.

```
<Grid.RowDefinitions>
    <RowDefinition Height="75" />
    <RowDefinition Height="625" />
</Grid.RowDefinitions>
<Button HorizontalAlignment="Left" Margin="22,20,0,24" Name="backButton" Width="75"><</Button>
<Button HorizontalAlignment="Left" Margin="116,20,0,24" Name="nextButton" Width="75">></Button>
<Button HorizontalAlignment="Right" Margin="0,21,46,24" Name="saveButton" Width="110">Save
changes</Button>
```

3. 프로젝트를 빌드합니다.

데이터 바인딩된 컨트롤 만들기

`Product` 테이블을 **데이터 소스** 창에서 WPF 디자이너로 끌어 고객 레코드를 표시 하는 컨트롤을 만듭니다.

1. **데이터 소스** 창에서 **Product** 노드의 드롭다운 메뉴를 클릭하고 **정보**를 선택합니다.
2. **Product** 노드를 확장합니다.
3. 이 예에서는 일부 필드를 표시하지 않을 것이므로 다음 노드 옆의 드롭다운 메뉴를 클릭하고 **없음**을 선택합니다.
   - ProductCategoryID
   - ProductModelID
   - ThumbnailPhotoFileName
   - rowguid
   - ModifiedDate
4. **ThumbNailPhoto** 노드 옆의 드롭다운 메뉴를 클릭하고 **이미지**를 선택합니다.

> **NOTE**
> 기본적으로 **데이터 원본** 창에서 사진을 나타내는 항목의 기본 컨트롤은 **없음**으로 설정되어 있습니다. 사진은 데이터베이스에서 바이트 배열로 저장되는데 바이트 배열은 단순한 바이트의 배열에서 대형 애플리케이션의 실행 파일에 이르기까지 모든 항목을 포함할 수 있기 때문입니다.

5. **데이터 원본** 창에서 단추가 포함된 행 아래의 그리드 행으로 **Product** 노드를 끌어 옵니다.

   Visual Studio는 **Product** 테이블의 데이터에 바인딩되는 컨트롤 집합을 정의하는 XAML이 생성됩니다. 또한 데이터를 로드하는 코드도 생성됩니다. 생성 된 XAML 및 코드에 대 한 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.

6. 디자이너에서 **제품 ID** 레이블 옆의 텍스트 상자를 클릭합니다.

7. **속성** 창에서 **IsReadOnly** 속성 옆의 확인란을 선택합니다.

제품 레코드 탐색

사용자가 **<** 및 **>** 단추를 사용하여 제품 레코드를 스크롤할 수 있도록 하는 코드를 추가합니다.

1. 디자이너의 창 화면에서 **<** 단추를 두 번 클릭합니다.

   Visual Studio가 코드 숨김 파일을 열고 Click 이벤트에 대해 새 `backButton_Click` 이벤트 처리기를 만듭니다.

2. `Window_Loaded` 이벤트 처리기를 수정하여 `ProductViewSource`, `AdventureWorksLTDataSet` 및 `AdventureWorksLTDataSetProductTableAdapter`가 메서드 외부에 있으며 전체 양식에 액세스할 수 있도록 합니다. 이러한 항목을 폼에 대해 전역적으로 선언 하 고 다음과 같이 `Window_Loaded` 이벤트 처리기 내에 할당합니다.

```
private AdventureWorksProductsEditor.AdventureWorksLTDataSet AdventureWorksLTDataSet;
private AdventureWorksProductsEditor.AdventureWorksLTDataSetTableAdapters.ProductTableAdapter
adventureWorksLTDataSetProductTableAdapter;
private System.Windows.Data.CollectionViewSource productViewSource;

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    AdventureWorksLTDataSet = ((AdventureWorksProductsEditor.AdventureWorksLTDataSet)
(this.FindResource("adventureWorksLTDataSet")));
    // Load data into the table Product. You can modify this code as needed.
    adventureWorksLTDataSetProductTableAdapter = new
AdventureWorksProductsEditor.AdventureWorksLTDataSetTableAdapters.ProductTableAdapter();
    adventureWorksLTDataSetProductTableAdapter.Fill(AdventureWorksLTDataSet.Product);
    productViewSource = ((System.Windows.Data.CollectionViewSource)
(this.FindResource("productViewSource")));
    productViewSource.View.MoveCurrentToFirst();
}
```

```
Dim ProductViewSource As System.Windows.Data.CollectionViewSource
Dim AdventureWorksLTDataSet As AdventureWorksProductsEditor.AdventureWorksLTDataSet
Dim AdventureWorksLTDataSetProductTableAdapter As
AdventureWorksProductsEditor.AdventureWorksLTDataSetTableAdapters.ProductTableAdapter

Private Sub Window_Loaded(ByVal sender As System.Object, ByVal e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
    AdventureWorksLTDataSet = CType(Me.FindResource("AdventureWorksLTDataSet"),
AdventureWorksProductsEditor.AdventureWorksLTDataSet)
    'Load data into the table Product. You can modify this code as needed.
    AdventureWorksLTDataSetProductTableAdapter = New
AdventureWorksProductsEditor.AdventureWorksLTDataSetTableAdapters.ProductTableAdapter()
    AdventureWorksLTDataSetProductTableAdapter.Fill(AdventureWorksLTDataSet.Product)
    ProductViewSource = CType(Me.FindResource("ProductViewSource"),
System.Windows.Data.CollectionViewSource)
    ProductViewSource.View.MoveCurrentToFirst()
End Sub
```

3. 다음 코드를 `backButton_Click` 이벤트 처리기에 추가합니다.

```
if (productViewSource.View.CurrentPosition > 0)
{
    productViewSource.View.MoveCurrentToPrevious();
}
```

```
If ProductViewSource.View.CurrentPosition > 0 Then
    ProductViewSource.View.MoveCurrentToPrevious()
End If
```

4. 디자이너로 돌아가서 **>** 단추를 두 번 클릭합니다.

5. 다음 코드를 `nextButton_Click` 이벤트 처리기에 추가합니다.

```
if (productViewSource.View.CurrentPosition < ((CollectionView)productViewSource.View).Count - 1)
{
    productViewSource.View.MoveCurrentToNext();
}
```

```
If ProductViewSource.View.CurrentPosition < CType(ProductViewSource.View, CollectionView).Count - 1
Then
    ProductViewSource.View.MoveCurrentToNext()
End If
```

제품 레코드 변경 내용 저장

사용자가 **변경 내용 저장** 단추를 사용하여 제품 레코드 변경 내용을 저장할 수 있도록 하는 코드를 추가합니다.

1. 디자이너에서 **변경 내용 저장** 단추를 두 번 클릭합니다.

   Visual Studio가 코드 숨김 파일을 열고 Click 이벤트에 대해 새 `saveButton_Click` 이벤트 처리기를 만듭니다.

2. 다음 코드를 `saveButton_Click` 이벤트 처리기에 추가합니다.

```
adventureWorksLTDataSetProductTableAdapter.Update(AdventureWorksLTDataSet.Product);
```

```
AdventureWorksLTDataSetProductTableAdapter.Update(AdventureWorksLTDataSet.Product)
```

> **NOTE**
> 이 예에서는 `Save`의 `TableAdapter` 메서드를 사용하여 변경 내용을 저장합니다. 이 연습에서는 데이터 테이블을 하나만 변경하므로 이러한 방식이 적절합니다. 여러 데이터 테이블의 변경 내용을 저장해야 하는 경우에는 데이터 세트과 함께 생성되는 `UpdateAll`의 `TableAdapterManager` 메서드를 사용할 수 있습니다. 자세한 내용은 tableadapter를 참조 하십시오.

응용 프로그램 테스트

애플리케이션을 빌드 및 실행합니다. 제품 레코드를 보고 업데이트할 수 있는지 확인합니다.

1. **F5**키를 누릅니다.

   애플리케이션이 빌드되고 실행됩니다. 다음 사항을 확인합니다.

   - 텍스트 상자에 사진이 포함된 첫 번째 제품 레코드의 데이터가 표시됩니다. 이 제품의 ID는 713이고 이름은 **Long-Sleeve Logo Jersey, S**입니다.
   - > 또는 < 단추를 클릭하여 다른 제품 레코드를 탐색할 수 있습니다.

2. 제품 레코드 중 하나에서 **크기** 값을 변경한 다음, **변경 내용 저장**을 클릭합니다.
3. 애플리케이션을 닫은 다음, Visual Studio에서 **F5**를 눌러 애플리케이션을 다시 시작합니다.
4. 변경한 제품 레코드로 이동하여 변경 내용이 유지되었는지를 확인합니다.
5. 애플리케이션을 닫습니다.

다음 단계

이 연습을 완료 한 후에는 다음과 같은 관련 작업을 수행해 볼 수 있습니다.

- Visual Studio의 **데이터 원본** 창을 사용하여 WPF 컨트롤을 다른 형식의 데이터 원본에 바인딩하는 방법을 알아봅니다. 자세한 내용은 WPF 컨트롤을 WCF data service에 바인딩을 참조 하세요.
- Visual Studio의 **데이터 원본** 창을 사용하여 WPF 컨트롤에 관련 데이터(즉, 부모-자식 관계가 있는 데이터)를 표시하는 방법을 알아봅니다. 자세한 내용은 연습: WPF 앱에서 관련 데이터 표시를 참조 하세요.

참조

- Visual Studio에서 데이터에 WPF 컨트롤 바인딩
- Visual Studio의 데이터 세트 도구
- 데이터 바인딩 개요

WCF 데이터 서비스에 WPF 컨트롤 바인딩

2020-01-06 • 27 minutes to read • Edit Online

이 연습에서는 데이터 바인딩된 컨트롤을 포함하는 WPF 애플리케이션을 만듭니다. 컨트롤은 WCF 데이터 서비스에서 캡슐화 된 고객 레코드에 바인딩됩니다. 또한 고객이 레코드를 보고 업데이트하는 데 사용할 수 있는 단추도 추가합니다.

이 연습에서는 다음 작업을 수행합니다.

- AdventureWorksLT 샘플 데이터베이스의 데이터에서 생성되는 엔터티 데이터 모델을 만듭니다.
- 엔터티 데이터 모델의 데이터를 WPF 응용 프로그램에 노출 하는 WCF 데이터 서비스를 만듭니다.
- **데이터 원본** 창에서 WPF 디자이너로 항목을 끌어 데이터 바인딩된 컨트롤 집합을 만듭니다.
- 고객 레코드를 앞뒤로 탐색하는 데 사용할 단추를 만듭니다.
- 컨트롤의 데이터 변경 내용을 WCF 데이터 서비스 및 기본 데이터 소스에 저장 하는 단추를 만듭니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

전제 조건

이 연습을 완료하려면 다음 구성 요소가 필요합니다.

- Visual Studio
- AdventureWorksLT 샘플 데이터베이스가 연결된 SQL Server 또는 SQL Server Express의 실행 중인 인스턴스 액세스 권한. CodePlex 웹 사이트에서 AdventureWorksLT 데이터베이스를 다운로드할 수 있습니다.

또한 다음 개념에 대한 지식은 연습을 완료하는 데 반드시 필요하지는 않지만 사전에 파악해 두면 유용할 수 있습니다.

- WCF Data Services.
- WCF Data Services의 데이터 모델
- 엔터티 데이터 모델 및 ADO.NET Entity Framework 자세한 내용은 Entity Framework 개요를 참조 하세요.
- WPF 데이터 바인딩. 자세한 내용은 데이터 바인딩 개요를 참조하세요.

서비스 프로젝트 만들기

1. 또는 Visual Basic C# **ASP.NET 웹 응용 프로그램** 프로젝트를 만들어이 연습을 시작 합니다. 프로젝트 이름을 **AdventureWorksService**로 합니다.
2. **솔루션 탐색기**에서 **Default.aspx**를 오른쪽 마우스 단추로 클릭하고 **삭제**를 선택합니다. 이 파일은 연습에 필요 하지 않습니다.

서비스에 대 한 엔터티 데이터 모델 만들기

WCF 데이터 서비스를 사용 하 여 응용 프로그램에 데이터를 노출 하려면 서비스에 대 한 데이터 모델을 정의 해야

합니다. WCF 데이터 서비스는 두 가지 유형의 데이터 모델, 즉 엔터티 데이터 모델 및 IQueryable<T> 인터페이스를 구현 하는 CLR (공용 언어 런타임) 개체를 사용 하 여 정의 되는 사용자 지정 데이터 모델을 지원 합니다. 이 연습에서는 데이터 모델에 대해 엔터티 데이터 모델을 만듭니다.

1. **프로젝트** 메뉴에서 **새 항목 추가**를 클릭합니다.
2. 설치된 템플릿 목록에서 **데이터**를 클릭한 다음, **ADO.NET 엔터티 데이터 모델** 프로젝트 항목을 선택합니다.
3. 이름을 `AdventureWorksModel.edmx`로 변경 하 고 **추가**를 클릭 합니다.

   엔터티 데이터 모델 마법사가 열립니다.
4. **모델 콘텐츠 선택** 페이지에서 **데이터베이스에서 생성**을 클릭한 후, **다음**을 클릭합니다.
5. **데이터 연결 선택** 페이지에서 다음 옵션 중 하나를 선택합니다.
   - AdventureWorksLT 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
   - **새 연결**을 클릭하고 AdventureWorksLT 데이터베이스에 대한 연결을 만듭니다.
6. **데이터 연결 선택** 페이지에서 **다른 이름으로 App.Config의 엔터티 연결 설정 저장** 옵션이 선택되어 있는지 확인한 후, **다음**을 클릭합니다.
7. **데이터베이스 개체 선택** 페이지에서 **테이블**을 확장한 다음, **SalesOrderHeader** 테이블을 선택합니다.
8. **마침**을 클릭합니다.

서비스 만들기

엔터티 데이터 모델의 데이터를 WPF 응용 프로그램에 노출 하는 WCF 데이터 서비스를 만듭니다.

1. **프로젝트** 메뉴에서 **새 항목 추가**를 선택합니다.
2. **설치된 템플릿** 목록에서 **웹**을 클릭한 다음, **WCF 데이터 서비스** 프로젝트 항목을 선택합니다.
3. **이름** 상자에 `AdventureWorksService.svc`를 입력 하 고 **추가**를 클릭 합니다.

   Visual Studio에서 프로젝트에 `AdventureWorksService.svc`를 추가 합니다.

서비스 구성

작성한 엔터티 데이터 모델에 대해 작동하도록 서비스를 구성해야 합니다.

1. `AdventureWorks.svc` 코드 파일에서 **AdventureWorksService** 클래스 선언을 다음 코드로 바꿉니다.

```
public class AdventureWorksService : DataService<AdventureWorksLTEntities>
{
    // This method is called only once to initialize service-wide policies.
    public static void InitializeService(IDataServiceConfiguration config)
    {
        config.SetEntitySetAccessRule("SalesOrderHeaders", EntitySetRights.All);
    }
}
```

```
Public Class AdventureWorksService
    Inherits DataService(Of AdventureWorksLTEntities)

    ' This method is called only once to initialize service-wide policies.
    Public Shared Sub InitializeService(ByVal config As IDataServiceConfiguration)
        config.SetEntitySetAccessRule("SalesOrderHeaders", EntitySetRights.All)
        config.UseVerboseErrors = True
    End Sub
End Class
```

이 코드는 **AdventureWorksService** 클래스를 업데이트 하여 엔터티 데이터 모델의 `AdventureWorksLTEntities` 개체 컨텍스트 클래스에서 작동 하는 DataService<T>에서 파생 됩니다. 또한 서비스의 클라이언트에 `InitializeService` 엔터티에 대한 모든 읽기/쓰기 권한을 허용하도록 `SalesOrderHeader` 메서드를 업데이트합니다.

2. 프로젝트를 빌드하고 오류가 없이 빌드되는지 확인합니다.

WPF 클라이언트 응용 프로그램 만들기

WCF 데이터 서비스의 데이터를 표시 하려면 서비스를 기반으로 하는 데이터 원본을 사용 하여 새 WPF 응용 프로그램을 만듭니다. 이 연습 뒷부분에서 애플리케이션에 데이터 바인딩된 컨트롤을 추가합니다.

1. **솔루션 탐색기**에서 솔루션 노드를 마우스 오른쪽 단추로 클릭하고, **추가**를 클릭하고, **새 프로젝트**를 선택합니다.
2. **새 프로젝트** 대화 상자에서 **Visual C#** 또는 **Visual Basic**을 확장한 다음, **Windows**를 선택합니다.
3. **WPF 애플리케이션** 프로젝트 템플릿을 선택합니다.
4. **이름** 상자에 `AdventureWorksSalesEditor`를 입력하고 **확인**을 클릭합니다.

   Visual Studio에서 솔루션에 `AdventureWorksSalesEditor` 프로젝트를 추가 합니다.
5. **데이터** 메뉴에서 **데이터 소스 표시**를 클릭합니다.

   데이터 원본 창이 열립니다.
6. **데이터 소스** 창에서 **새 데이터 소스 추가**를 클릭합니다.

   데이터 원본 구성 마법사가 열립니다.
7. 마법사의 **데이터 원본 형식 선택** 페이지에서 **서비스**를 선택한 후, **다음**을 클릭합니다.
8. **서비스 참조 추가** 대화 상자에서 **검색**을 클릭합니다.

   Visual Studio는 현재 솔루션에서 사용 가능한 서비스를 검색 하고 **서비스** 상자에서 사용 가능한 서비스 목록에 `AdventureWorksService.svc`를 추가 합니다.
9. **네임스페이스** 상자에 **AdventureWorksService**를 입력합니다.
10. **서비스** 상자에서 **AdventureWorksService.svc**를 클릭한 다음, **확인**을 클릭합니다.

    Visual Studio에서 서비스 정보를 다운로드한 다음, **데이터 원본 구성** 마법사로 돌아갑니다.
11. **서비스 참조 추가** 페이지에서 **마침**을 클릭합니다.

    Visual Studio는 서비스에서 반환하는 데이터를 나타내는 노드를 **데이터 원본** 창에 추가합니다.

사용자 인터페이스 정의

WPF 디자이너에서 XAML을 수정하여 창에 여러 단추를 추가합니다. 이 연습 뒷부분에서 사용자가 이러한 단추를

사용해 판매 레코드를 보고 업데이트할 수 있도록 하는 코드를 추가합니다.

1. **솔루션 탐색기**에서 **MainWindow.xaml**을 두 번 클릭합니다.

   WPF 디자이너에서 창이 열립니다.

2. 디자이너의 XAML 뷰에서 `<Grid>` 태그 사이에 다음 코드를 추가합니다.

```
<Grid.RowDefinitions>
    <RowDefinition Height="75" />
    <RowDefinition Height="525" />
</Grid.RowDefinitions>
<Button HorizontalAlignment="Left" Margin="22,20,0,24" Name="backButton" Width="75"><</Button>
<Button HorizontalAlignment="Left" Margin="116,20,0,24" Name="nextButton" Width="75">></Button>
<Button HorizontalAlignment="Right" Margin="0,21,46,24" Name="saveButton" Width="110">Save
changes</Button>
```

3. 프로젝트를 빌드합니다.

데이터 바인딩된 컨트롤 만들기

`SalesOrderHeaders` 노드를 **데이터 소스** 창에서 디자이너로 끌어 고객 레코드를 표시 하는 컨트롤을 만듭니다.

1. **데이터 원본** 창에서 **SalesOrderHeaders** 노드의 드롭다운 메뉴를 클릭하고 **정보**를 선택합니다.
2. **SalesOrderHeaders** 노드를 확장합니다.
3. 이 예에서는 일부 필드를 표시하지 않을 것이므로 다음 노드 옆의 드롭다운 메뉴를 클릭하고 **없음**을 선택합니다.

   - **CreditCardApprovalCode**
   - **ModifiedDate**
   - **OnlineOrderFlag**
   - **RevisionNumber**
   - **Rowguid**

   이 작업을 수행하면 다음 단계에서 이러한 노드에 대해 데이터 바인딩된 컨트롤이 작성되지 않습니다. 이 연습에서는 최종 사용자가이 데이터를 볼 필요가 없다고 가정 합니다.

4. **데이터 원본** 창에서 단추가 포함된 행 아래의 데이터 그리드 행으로 **SalesOrderHeaders** 노드를 끌어 옵니다.

   Visual Studio는 **Product** 테이블의 데이터에 바인딩되는 컨트롤 집합을 만드는 XAML 및 코드를 생성합니다. 생성 된 XAML 및 코드에 대 한 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조하세요.

5. 디자이너에서 **고객 ID** 레이블 옆의 텍스트 상자를 클릭합니다.
6. **속성** 창에서 **IsReadOnly** 속성 옆의 확인란을 선택합니다.
7. 다음의 각 텍스트 상자에 대해 **IsReadOnly** 속성을 설정합니다.

   - **구매 주문 번호**
   - **판매 주문 ID**
   - **판매 주문 번호**

서비스에서 데이터 로드

서비스 프록시 개체를 사용 하 여 서비스에서 판매 데이터를 로드 합니다. 그런 다음 WPF 창에서 CollectionViewSource에 대 한 데이터 소스에 반환 된 데이터를 할당 합니다.

1. 디자이너를 만들려면를 `Window_Loaded` 이벤트 처리기에서 텍스트를 두 번 클릭: **MainWindow**.

2. 이벤트 처리기를 다음 코드로 바꿉니다. 이 코드의 *localhost*주소는 사용 중인 개발 컴퓨터의 로컬 호스트 주소로 바꿔야 합니다.

```
private AdventureWorksService.AdventureWorksLTEntities dataServiceClient;
private System.Data.Services.Client.DataServiceQuery<AdventureWorksService.SalesOrderHeader>
salesQuery;
private CollectionViewSource ordersViewSource;

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    // TODO: Modify the port number in the following URI as required.
    dataServiceClient = new AdventureWorksService.AdventureWorksLTEntities(
        new Uri("http://localhost:45899/AdventureWorksService.svc"));
    salesQuery = dataServiceClient.SalesOrderHeaders;

    ordersViewSource = ((CollectionViewSource)(this.FindResource("salesOrderHeadersViewSource")));
    ordersViewSource.Source = salesQuery.Execute();
    ordersViewSource.View.MoveCurrentToFirst();
}
```

```
Private DataServiceClient As AdventureWorksService.AdventureWorksLTEntities
Private SalesQuery As System.Data.Services.Client.DataServiceQuery(Of
AdventureWorksService.SalesOrderHeader)
Private OrdersViewSource As CollectionViewSource

Private Sub Window_Loaded(ByVal Sender As Object, ByVal e As RoutedEventArgs) Handles MyBase.Loaded

    ' TODO: Modify the port number in the following URI as required.
    DataServiceClient = New AdventureWorksService.AdventureWorksLTEntities( _
    New Uri("http://localhost:32415/AdventureWorksService.svc"))
    SalesQuery = DataServiceClient.SalesOrderHeaders

    OrdersViewSource = CType(Me.FindResource("SalesOrderHeadersViewSource"), CollectionViewSource)
    OrdersViewSource.Source = SalesQuery.Execute()
    OrdersViewSource.View.MoveCurrentToFirst()
End Sub
```

판매 레코드 탐색

사용자가 < 및 > 단추를 사용하여 판매 레코드를 스크롤할 수 있도록 하는 코드를 추가합니다.

1. 디자이너의 창 화면에서 < 단추를 두 번 클릭합니다.

   Visual Studio가 코드 숨김 파일을 열고 Click 이벤트에 대해 새 `backButton_Click` 이벤트 처리기를 만듭니다.

2. 다음 코드를 생성된 `backButton_Click` 이벤트 처리기에 추가합니다.

```
if (ordersViewSource.View.CurrentPosition > 0)
    ordersViewSource.View.MoveCurrentToPrevious();
```

```
If OrdersViewSource.View.CurrentPosition > 0 Then
    OrdersViewSource.View.MoveCurrentToPrevious()
End If
```

3. 디자이너로 돌아와서 **>** 단추를 두 번 클릭합니다.

   Visual Studio가 코드 숨김 파일을 열고 Click 이벤트에 대해 새 `nextButton_Click` 이벤트 처리기를 만듭니다.

4. 다음 코드를 생성된 `nextButton_Click` 이벤트 처리기에 추가합니다.

```
if (ordersViewSource.View.CurrentPosition < ((CollectionView)ordersViewSource.View).Count - 1)
{
    ordersViewSource.View.MoveCurrentToNext();
}
```

```
If OrdersViewSource.View.CurrentPosition < CType(OrdersViewSource.View, CollectionView).Count - 1 Then
    OrdersViewSource.View.MoveCurrentToNext()
End If
```

판매 레코드 변경 내용 저장

사용자가 **변경 내용 저장** 단추를 사용하여 판매 레코드 변경 내용을 확인 및 저장할 수 있도록 하는 코드를 추가합니다.

1. 디자이너에서 **변경 내용 저장** 단추를 두 번 클릭합니다.

   Visual Studio가 코드 숨김 파일을 열고 Click 이벤트에 대해 새 `saveButton_Click` 이벤트 처리기를 만듭니다.

2. 다음 코드를 `saveButton_Click` 이벤트 처리기에 추가합니다.

```
AdventureWorksService.SalesOrderHeader currentOrder =
(AdventureWorksService.SalesOrderHeader)ordersViewSource.View.CurrentItem;
dataServiceClient.UpdateObject(currentOrder);
dataServiceClient.SaveChanges();
```

```
Dim CurrentOrder As AdventureWorksService.SalesOrderHeader = CType(OrdersViewSource.View.CurrentItem,
AdventureWorksService.SalesOrderHeader)

DataServiceClient.UpdateObject(CurrentOrder)
DataServiceClient.SaveChanges()
```

응용 프로그램 테스트

애플리케이션을 빌드하고 실행하여 고객 레코드를 보고 업데이트할 수 있는지 확인합니다.

1. **빌드** 메뉴에서 **솔루션 빌드**를 클릭 합니다. 솔루션이 오류 없이 빌드되는지 확인합니다.
2. **Ctrl+f5**키를 누릅니다.

   Visual Studio는 디버깅하지 않고 **AdventureWorksService** 프로젝트를 시작합니다.

3. **솔루션 탐색기**에서 **AdventureWorksSalesEditor** 프로젝트를 마우스 오른쪽 단추로 클릭합니다.
4. 오른쪽 클릭 메뉴 (상황에 맞는 메뉴)의 **디버그**에서 **새 인스턴스 시작**을 클릭 합니다.

애플리케이션이 실행됩니다. 다음 사항을 확인합니다.

- 판매 주문 ID가 **71774**인 첫 번째 판매 레코드의 서로 다른 데이터 필드가 텍스트 상자에 표시됩니다.
- > 또는 < 단추를 클릭하여 다른 판매 레코드를 탐색할 수 있습니다.

5. 판매 레코드 중 하나의 **주석** 상자에 텍스트를 입력한 다음, **변경내용 저장**을 클릭합니다.
6. 애플리케이션을 닫았다가 Visual Studio에서 다시 시작합니다.
7. 변경한 판매 레코드로 이동하여 애플리케이션을 닫았다가 다시 열어도 변경 내용이 그대로 유지되는지 확인합니다.
8. 애플리케이션을 닫습니다.

다음 단계

이 연습을 완료하고 나면 다음과 같은 관련 작업을 수행할 수 있습니다.

- Visual Studio의 **데이터 원본** 창을 사용하여 WPF 컨트롤을 다른 형식의 데이터 원본에 바인딩하는 방법을 알아봅니다. 자세한 내용은 WPF 컨트롤을 데이터 집합에 바인딩을 참조 하세요.
- Visual Studio의 **데이터 원본** 창을 사용하여 WPF 컨트롤에 관련 데이터(즉, 부모-자식 관계가 있는 데이터)를 표시하는 방법을 알아봅니다. 자세한 내용은 연습: WPF 응용 프로그램에서 관련 데이터 표시를 참조 하세요.

참조

- Visual Studio에서 데이터에 WPF 컨트롤 바인딩
- 데이터 세트로 WPF 컨트롤 바인딩
- WCF 개요 (.NET Framework)
- Entity Framework 개요 (.NET Framework)
- 데이터 바인딩 개요 (.NET Framework)

WPF 애플리케이션에서 조회 테이블 만들기

2020-01-06 • 12 minutes to read • Edit Online

용어 *조회 테이블* (*조회 바인딩*이라고 함)은 다른 테이블의 외래 키 필드 값을 기반으로 한 데이터 테이블의 정보를 표시 하는 컨트롤을 설명 합니다. **데이터 소스** 창에서 부모 테이블 또는 개체의 주 노드를 관련 자식 테이블의 열 또는 속성에 이미 바인딩된 컨트롤로 끌어 조회 테이블을 만들 수 있습니다.

예를 들어 sales 데이터베이스에 `Orders` 테이블이 있다고 가정 합니다. `Orders` 테이블의 각 레코드에는 주문이 배치 된 고객을 나타내는 `CustomerID` 포함 되어 있습니다. `CustomerID` 은 `Customers` 테이블의 고객 레코드를 가리키는 외래 키입니다. `Orders` 테이블의 주문 목록을 표시 하는 경우 `CustomerID` 대신 실제 고객 이름을 표시 하는 것이 좋습니다. 고객 이름이 `Customers` 테이블에 있으므로 고객 이름을 표시 하는 조회 테이블을 만들어야 합니다. 조회 테이블은 `Orders` 레코드의 `CustomerID` 값을 사용 하 여 관계를 탐색 하 고 고객 이름을 반환 합니다.

조회 테이블을 만들려면

1. 관련 데이터가 포함 된 다음 데이터 원본 중 하나를 프로젝트에 추가 합니다.

   - 데이터 집합 또는 엔터티 데이터 모델입니다.
   - WCF 데이터 서비스, WCF 서비스 또는 웹 서비스입니다. 자세한 내용은 방법: 서비스의 데이터에 연결을 참조 하세요.
   - 개체. 자세한 내용은 Visual Studio에서 개체에 바인딩을 참조 하세요.

   > **NOTE**
   > 조회 테이블을 만들려면 먼저 두 개의 관련 테이블이 나 개체가 프로젝트의 데이터 원본으로 존재 해야 합니다.

2. **WPF 디자이너**를 열고 디자이너에 **데이터 소스** 창의 항목에 대 한 유효한 놓기 대상인 컨테이너가 포함 되어 있는지 확인 합니다.

   유효한 놓기 대상에 대 한 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.

3. **데이터** 메뉴에서 **데이터 원본 표시**를 클릭하여 **데이터 원본** 창을 엽니다.

4. 부모 테이블이 나 개체 및 관련 자식 테이블이 나 개체를 볼 수 있을 때까지 **데이터 소스** 창에서 노드를 확장 합니다.

   > **NOTE**
   > 관련 자식 테이블이 나 개체는 부모 테이블이 나 개체 아래에 확장 가능한 자식 노드로 표시 되는 노드입니다.

5. 자식 노드에 대 한 드롭다운 메뉴를 클릭 하 고 **세부 정보**를 선택 합니다.

6. 자식 노드를 확장 합니다.

7. 자식 노드 아래에서 자식 및 부모 데이터와 관련 된 항목에 대 한 드롭다운 메뉴를 클릭 합니다. 앞의 예제에서 이는 **CustomerID** 노드입니다. 조회 바인딩을 지 원하는 다음 형식의 컨트롤 중 하나를 선택 합니다.

   - **ComboBox**
   - **ListBox**

- ListView

> **NOTE**
> 목록에 **ListBox** 또는 **ListView** 컨트롤이 표시 되지 않는 경우 목록에 이러한 컨트롤을 추가할 수 있습니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.

- Selector에서 파생 되는 모든 사용자 지정 컨트롤입니다.

> **NOTE**
> **데이터 소스** 창에서 항목에 대해 선택할 수 있는 컨트롤 목록에 사용자 지정 컨트롤을 추가 하는 방법에 대한 자세한 내용은 데이터 소스 창에 사용자 지정 컨트롤 추가를 참조 하세요.

8. 자식 노드를 **데이터 소스** 창에서 WPF 디자이너의 컨테이너로 끌어 옵니다. 앞의 예제에서 자식 노드는 **Orders** 노드입니다.

   Visual Studio는 끌어 온 각 항목에 대해 새로운 데이터 바인딩된 컨트롤을 만드는 XAML을 생성 합니다. 또한 XAML은 자식 테이블 또는 개체의 새 CollectionViewSource를 놓기 대상의 리소스에 추가 합니다. 일부 데이터 원본의 경우에는 Visual Studio에서 테이블 또는 개체에 데이터를 로드 하는 코드도 생성 합니다. 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.

9. **데이터 소스** 창에서 이전에 만든 조회 바인딩 컨트롤로 부모 노드를 끌어 옵니다. 앞의 예제에서 부모 노드는 **Customers** 노드입니다.

   Visual Studio는 컨트롤의 일부 속성을 설정 하 여 조회 바인딩을 구성 합니다. 다음 표에서는 Visual Studio에서 수정 하는 속성을 보여 줍니다. 필요한 경우 XAML 또는 **속성** 창에서 이러한 속성을 변경할 수 있습니다.

속성	설정 설명
ItemsSource	이 속성은 컨트롤에 표시 되는 데이터를 가져오는 데 사용 되는 컬렉션 또는 바인딩을 지정 합니다. Visual Studio는 컨트롤로 끌어 온 부모 데이터의 CollectionViewSource으로이 속성을 설정 합니다.
DisplayMemberPath	이 속성은 컨트롤에 표시 되는 데이터 항목의 경로를 지정 합니다. Visual Studio는 기본 키 뒤에 문자열 데이터 형식이 있는 부모 데이터의 첫 번째 열 또는 속성으로이 속성을 설정 합니다. 부모 데이터에 다른 열 이나 속성을 표시 하려는 경우이 속성을 다른 속성의 경로로 변경 합니다.
SelectedValue	Visual Studio는이 속성을 디자이너로 끌어 온 자식 데이터의 열 또는 속성에 바인딩합니다. 부모 데이터의 외래 키입니다.
SelectedValuePath	Visual Studio는 부모 데이터의 외래 키인 자식 데이터의 열 또는 속성 경로에이 속성을 설정 합니다.

참조

- Visual Studio에서 데이터에 WPF 컨트롤 바인딩
- WPF 애플리케이션에서 관련 데이터 표시

- 연습: WPF 애플리케이션에서 관련 데이터 표시

WPF 애플리케이션에서 관련 데이터 표시

2020-01-06 • 6 minutes to read • Edit Online

일부 응용 프로그램에서는 여러 테이블이 나 부모-자식 관계에서 서로 관련 된 엔터티에서 제공 되는 데이터로 작업 하는 것이 좋습니다. 예를 들어 `Customers` 테이블의 고객을 표시 하는 그리드를 표시 하려고 할 수 있습니다. 사용자가 특정 고객을 선택 하면 다른 표에는 관련 `Orders` 테이블에서 해당 고객에 대 한 주문이 표시 됩니다.

데이터 소스 창에서 WPF 디자이너로 항목을 끌어 관련 데이터를 표시 하는 데이터 바인딩된 컨트롤을 만들 수 있습니다.

관련 레코드를 표시 하는 컨트롤을 만들려면

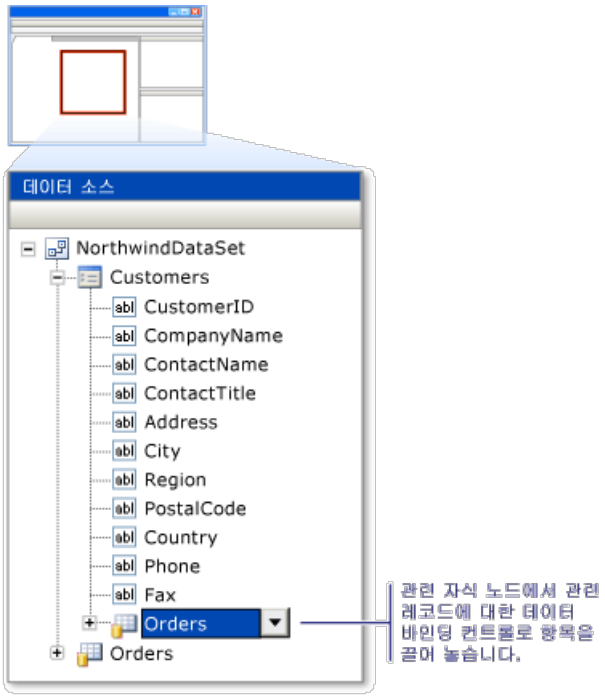
1. **데이터** 메뉴에서 **데이터 원본 표시**를 클릭하여 **데이터 원본** 창을 엽니다.
2. **새 데이터 원본 추가**를 클릭하고 **데이터 원본 구성 마법사** 완료합니다.
3. WPF 디자이너를 열고 디자이너에 **데이터 소스** 창의 항목에 대 한 유효한 놓기 대상인 컨테이너가 포함 되어 있는지 확인 합니다.

   유효한 놓기 대상에 대 한 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.
4. **데이터 소스** 창에서 관계의 부모 테이블이 나 개체를 나타내는 노드를 확장 합니다. 부모 테이블 또는 개체가 일 대 다 관계의 "일" 쪽에 있습니다.
5. 부모 노드 또는 부모 노드의 모든 개별 항목을 **데이터 소스** 창에서 디자이너의 유효한 놓기 대상으로 끌어 옵니다.

   Visual Studio는 끌어 온 각 항목에 대해 새로운 데이터 바인딩된 컨트롤을 만드는 XAML을 생성 합니다. 또한 XAML은 부모 테이블이 나 개체에 대 한 새 CollectionViewSource를 놓기 대상의 리소스에 추가 합니다. 일부 데이터 원본의 경우에는 Visual Studio에서 부모 테이블이 나 개체에 데이터를 로드 하는 코드도 생성 합니다. 자세한 내용은 Visual Studio에서 데이터에 WPF 컨트롤 바인딩을 참조 하세요.
6. **데이터 소스** 창에서 관련 자식 테이블이 나 개체를 찾습니다. 관련 자식 테이블 및 개체가 부모 노드의 데이터 목록 아래쪽에 확장 가능한 노드로 표시 됩니다.
7. 자식 노드 (또는 자식 노드의 모든 개별 항목)를 **데이터 소스** 창에서 디자이너의 유효한 놓기 대상으로 끌어 옵니다.

   Visual Studio는 끌어 온 각 항목에 대해 새로운 데이터 바인딩된 컨트롤을 만드는 XAML을 생성 합니다. 또한 XAML은 자식 테이블 또는 개체의 새 CollectionViewSource를 놓기 대상의 리소스에 추가 합니다. 이 새 CollectionViewSource는 방금 디자이너로 끌어온 부모 테이블이 나 개체의 속성에 바인딩됩니다. 일부 데이터 원본의 경우에는 Visual Studio에서 자식 테이블이 나 개체에 데이터를 로드 하는 코드도 생성 합니다.

   다음 그림에서는 **데이터 소스** 창에서 데이터 집합에 있는 **Customers** 테이블의 관련 **Orders** 테이블을 보여 줍니다.



참조

- Visual Studio에서 데이터에 WPF 컨트롤 바인딩
- WPF 애플리케이션에서 조회 테이블 만들기

데이터베이스의 그림에 컨트롤 바인딩

2020-01-06 • 4 minutes to read • Edit Online

데이터 소스 창을 사용 하 여 데이터베이스의 이미지를 응용 프로그램의 컨트롤에 바인딩할 수 있습니다. 예를 들어, WPF 응용 프로그램의 Image 컨트롤에 이미지를 바인딩하거나 Windows Forms 응용 프로그램의 PictureBox 컨트롤에 바인딩할 수 있습니다.

데이터베이스의 사진은 일반적으로 바이트 배열로 저장 됩니다. 바이트 배열로 저장 되는 **데이터 소스** 창의 항목에는 기본적으로 **없음** 으로 설정 되어 있습니다. 바이트 배열에는 간단한 바이트 배열의 모든 항목이 포함 될 수 있기 때문입니다. 이미지를 나타내는 **데이터 소스** 창에서 바이트 배열 항목에 대 한 데이터 바인딩된 컨트롤을 만들려면 만들 컨트롤을 선택 해야 합니다.

다음 절차에서는 **데이터 소스** 창이 이미 이미지에 바인딩된 항목으로 채워진 것으로 가정 합니다.

데이터베이스의 그림을 컨트롤에 바인딩하려면

1. 컨트롤을 추가 하려는 디자인 화면이 WPF 디자이너 또는 Windows Forms 디자이너에서 열려 있는지 확인 합니다.
2. **데이터 원본** 창에서 원하는 테이블 또는 개체를 확장 하 여 해당 열 또는 속성을 표시 합니다.

   > **TIP**
   > **데이터 소스** 창이 열려 있지 않으면 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 엽니다.

3. 이미지 데이터를 포함 하는 열 또는 속성을 선택 하 고 드롭다운 컨트롤 목록에서 다음 컨트롤 중 하나를 선택 합니다.
   - WPF designer가 열려 있으면 **이미지**를 선택 합니다.
   - Windows Forms designer가 열려 있으면 **PictureBox**를 선택 합니다.
   - 또는 데이터 바인딩을 지원 하 고 이미지를 표시할 수 있는 다른 컨트롤을 선택할 수 있습니다. 사용할 컨트롤이 사용 가능한 컨트롤 목록에 없으면 목록에 추가 하 여 선택할 수 있습니다 (예를 들어, 자세한 내용은 데이터 소스 창에 사용자 지정 컨트롤 추가를 참조 하세요.

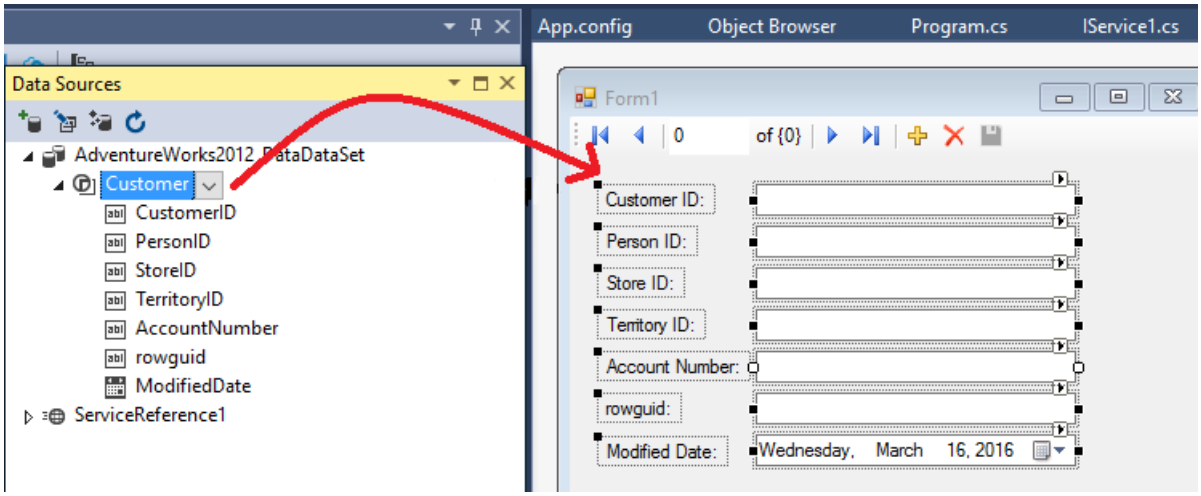
참조

- Visual Studio에서 데이터에 WPF 컨트롤 바인딩

Windows Forms 컨트롤을 Visual Studio의 데이터에 바인딩

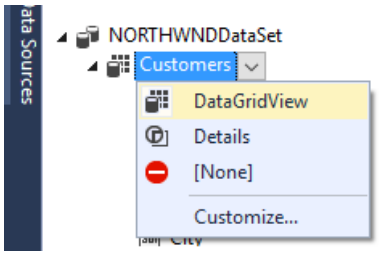
2020-01-06 • 9 minutes to read • Edit Online

데이터를 Windows Forms에 바인딩하여 응용 프로그램 사용자에 게 데이터를 표시할 수 있습니다. 이러한 데이터 바인딩된 컨트롤을 만들려면 **데이터 소스** 창에서 Visual Studio의 Windows Forms 디자이너로 항목을 끌어 옵니다.



> **TIP**
> **데이터 소스** 창이 표시 되지 않는 경우 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하거나 **Shift**+**Alt**+**D**를 눌러 열 수 있습니다. **데이터 소스** 창을 보려면 Visual Studio에서 프로젝트를 열어야 합니다.

항목을 끌기 전에 바인딩하려는 컨트롤의 형식을 설정할 수 있습니다. 테이블 자체를 선택 하는지 아니면 개별 열을 선택 하는지에 따라 다른 값이 표시 됩니다. 사용자 지정 값을 설정할 수도 있습니다. 테이블의 경우 **세부 정보** 는 각 열이 별도의 컨트롤에 바인딩되어 있음을 의미 합니다.



BindingSource 및 BindingNavigator 컨트롤

BindingSource 구성 요소는 두가지 용도로 사용됩니다. 먼저 컨트롤을 데이터에 바인딩할 때 추상화 계층을 제공 합니다. 양식의 컨트롤은 데이터 소스에 직접 연결 하는 대신 BindingSource 구성 요소에 바인딩됩니다. 둘째, 개체의 컬렉션을 관리할 수 있습니다. BindingSource에 형식을 추가 하면 해당 형식의 목록이 생성 됩니다.

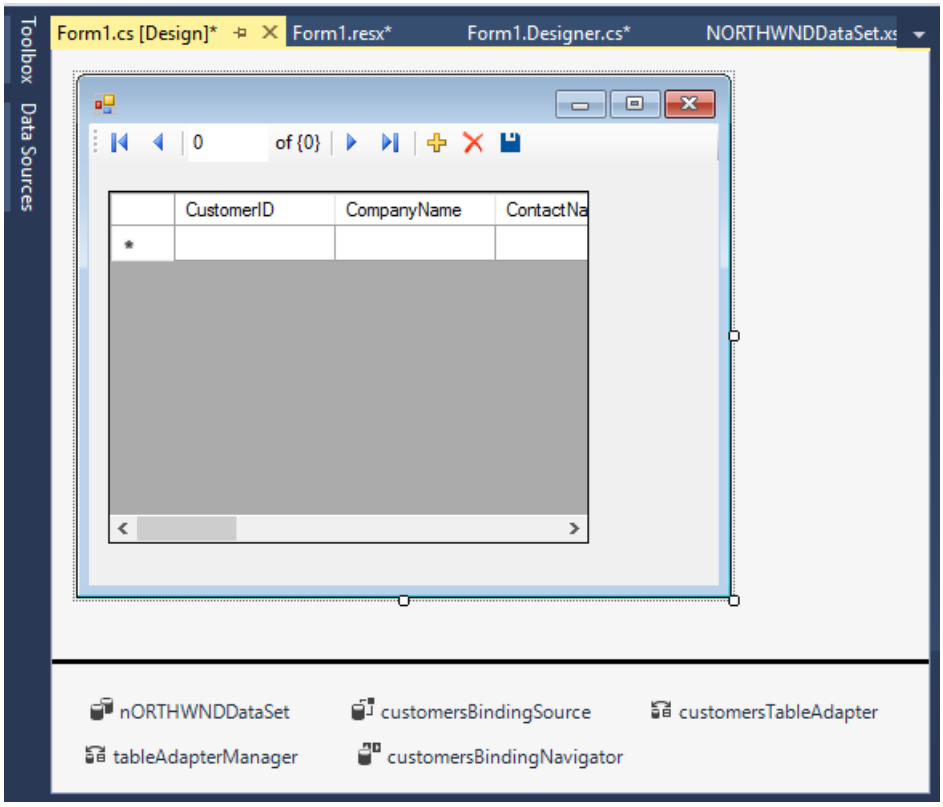
BindingSource 구성 요소에 대 한 자세한 내용은 다음을 참조 하세요.

- BindingSource 구성 요소
- BindingSource 구성 요소 개요
- BindingSource 구성 요소 아키텍처

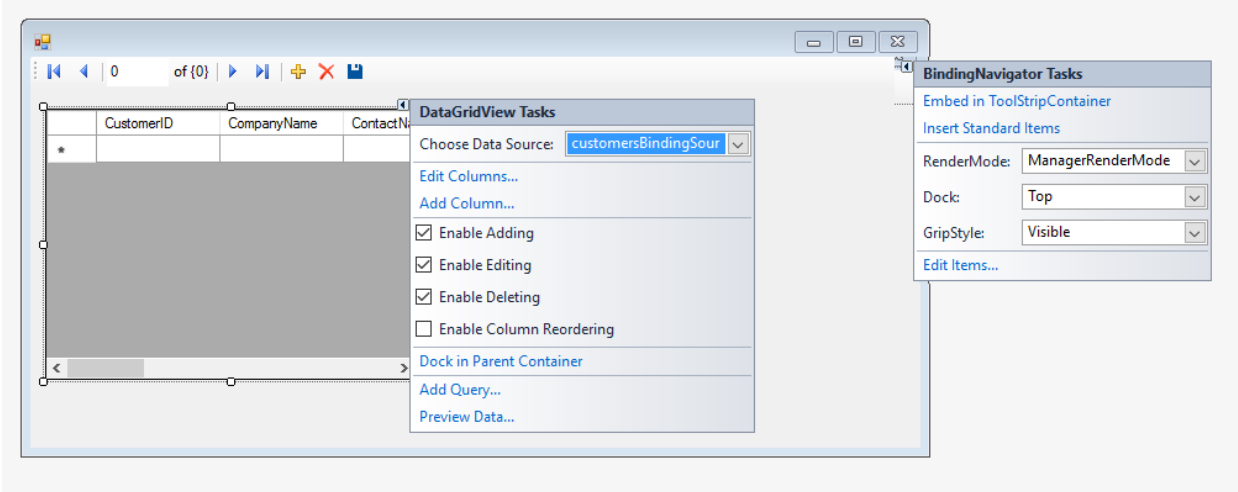
BindingNavigator 컨트롤 은 Windows 응용 프로그램에 표시 되는 데이터를 탐색 하기 위한 사용자 인터페이스를 제공 합니다.

DataGridView 컨트롤의 데이터에 바인딩

DataGridView 컨트롤의 경우 전체 테이블이 해당 단일 컨트롤에 바인딩됩니다. **DataGridView** 를 폼으로 끌어오면 레코드 탐색을 위한 도구 스트립 (BindingNavigator)도 표시 됩니다. 데이터 집합, TableAdapter, BindingSource및 BindingNavigator 구성 요소 트레이에 나타납니다. 다음 그림에서는 Customers 테이블에 Orders 테이블과의 관계가 있으므로 TableAdapterManager 도 추가 되었습니다. 이러한 변수는 모두 자동 생성 코드에서 form 클래스의 private 멤버로 선언 됩니다. **DataGridView** 를 채우기 위한 자동 생성 코드는 `Form_Load` 이벤트 처리기에 있습니다. 데이터베이스를 업데이트 하기 위해 데이터를 저장 하는 코드는 **BindingNavigator**의 `Save` 이벤트 처리기에 있습니다. 필요에 따라이 코드를 이동 하거나 수정할 수 있습니다.



각각의 오른쪽 위 모퉁이에 있는 스마트 태그를 클릭 하 여 **DataGridView** 및 **BindingNavigator** 의 동작을 사용자 지정할 수 있습니다.

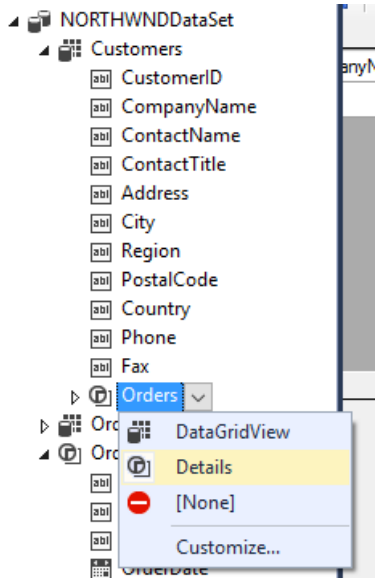


응용 프로그램에 필요한 컨트롤을 **데이터 소스** 창에서 사용할 수 없는 경우 컨트롤을 추가할 수 있습니다. 자세한 내용은 데이터 소스 창에 사용자 지정 컨트롤 추가를 참조 하세요.

데이터 소스 창에서 폼에 이미 있는 컨트롤로 항목을 끌어와 서 컨트롤을 데이터에 바인딩할 수도 있습니다. 이미 데이터에 바인딩된 컨트롤의 데이터 바인딩이 가장 최근에 끌어온 항목으로 다시 설정 됩니다. 유효한 놓기 대상이 되려면 컨트롤이 **데이터 소스** 창에서 끌어 온 항목의 기본 데이터 형식을 표시할 수 있어야 합니다. 예를 들어 CheckBox 날짜를 표시할 수 없기 때문에 데이터 형식이 DateTime 인 항목을 CheckBox로 끌어 올 수 없습니다.

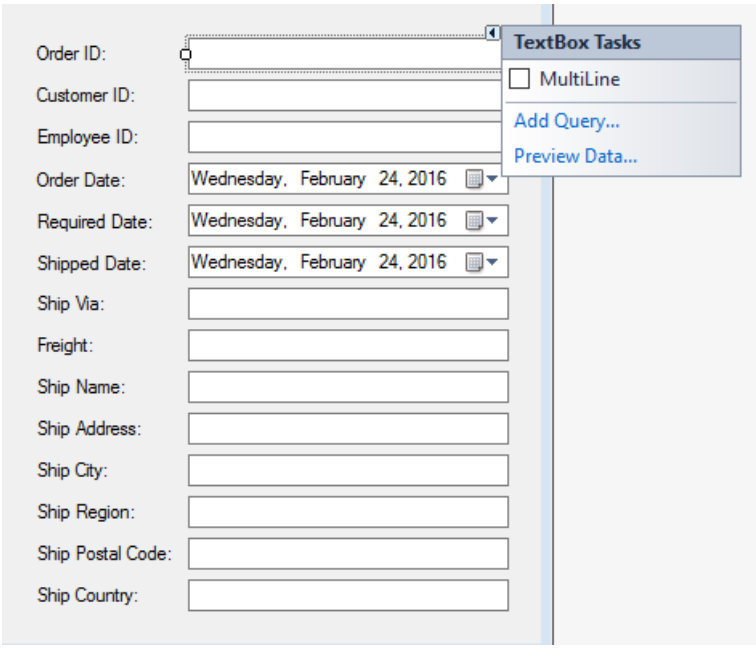
개별 컨트롤의 데이터에 바인딩

데이터 소스를 **세부 정보**에 바인딩하는 경우 데이터 집합의 각 열이 별도의 컨트롤에 바인딩됩니다.



> **IMPORTANT**
> 위의 그림에서는 Orders 테이블이 아니라 Customers 테이블의 Orders 속성에서 끌어 옵니다. `Customer.Orders` 속성에 바인딩하면 **DataGridView** 에서 만든 탐색 명령이 details 컨트롤에 즉시 반영 됩니다. Orders 테이블에서 끌어 오면 컨트롤이 데이터 집합에 계속 바인딩되어 있지만 **DataGridView**와 동기화 되지 않습니다.

다음 그림은 Customers 테이블의 Orders 속성이 **데이터 소스** 창의 **세부 정보** 에 바인딩된 후 폼에 추가 되는 기본 데이터 바인딩된 컨트롤을 보여 줍니다.



또한 각 컨트롤에는 스마트 태그가 있습니다. 이 태그를 사용 하면 해당 컨트롤에만 적용 되는 사용자 지정이 가능 합니다.

참조

- Visual Studio의 데이터에 컨트롤 바인딩
- Windows Forms의 데이터 바인딩 (.NET Framework)

Windows Forms 애플리케이션에서 데이터 필터링 및 정렬

2020-01-06 • 2 minutes to read • Edit Online

Filter 속성을 원하는 레코드를 반환 하는 문자열 식으로 설정 하 여 데이터를 필터링 합니다.

Sort 속성을 정렬할 열 이름으로 설정 하 여 데이터를 정렬 합니다. 내림차순으로 정렬 하려면 `DESC`을 추가 하 고, 오름차순으로 정렬 하려면 `ASC`를 추가 합니다.

> **NOTE**
> 응용 프로그램에서 BindingSource 구성 요소를 사용 하지 않는 경우 DataView 개체를 사용 하 여 데이터를 필터링 하 고 정렬 할 수 있습니다. 자세한 내용은 Dataview합니다.

BindingSource 구성 요소를 사용 하 여 데이터를 필터링 하려면

- Filter 속성을 반환 하려는 식으로 설정 합니다. 예를 들어 다음 코드는 "B"로 시작 하는 `CompanyName` 있는 고객을 반환 합니다.

```
customersBindingSource.Filter = "CompanyName like 'B'";
```

```
CustomersBindingSource.Filter = "CompanyName like 'B'"
```

BindingSource 구성 요소를 사용 하 여 데이터를 정렬 하려면

- Sort 속성을 정렬 하려는 열로 설정 합니다. 예를 들어 다음 코드는 `CompanyName` 열에 대 한 고객을 내림차순으로 정렬 합니다.

```
customersBindingSource.Sort = "CompanyName Desc";
```

```
CustomersBindingSource.Sort = "CompanyName Desc"
```

참조

- Visual Studio에서 데이터에 컨트롤 바인딩

데이터 바인딩된 컨트롤에서 데이터를 저장하기 전에 In-Process 편집 커밋

2020-01-06 • 4 minutes to read • Edit Online

데이터 바인딩된 컨트롤에서 값을 편집할 때 사용자는 현재 레코드를 탐색 하 여 컨트롤이 바인딩되는 기본 데이터 소스에 업데이트 된 값을 커밋해야 합니다. 데이터 소스 창 에서 폼으로 항목을 끌어 놓으면 첫 번째 항목은 BindingNavigator의 **저장** 단추 클릭 이벤트에 코드를 생성 합니다. 이 코드는 BindingSource의 EndEdit 메서드를 호출 합니다. 따라서 EndEdit 메서드에 대 한 호출은 폼에 추가 된 첫 번째 BindingSource에 대해서만 생성 됩니다.

EndEdit 호출에서는 현재 편집 중인 데이터 바인딩된 컨트롤의 프로세스에 포함된 모든 변경 내용을 커밋합니다. 따라서 데이터 바인딩된 컨트롤에 계속 포커스가 있는 상태에서 **저장** 단추를 클릭하면 실제 저장 전에 해당 컨트롤에서 보류 중인 모든 편집 내용이 커밋됩니다(`TableAdapterManager.UpdateAll` 메서드).

사용자가 변경 내용을 커밋하지 않고 저장 프로세스의 일부로 데이터를 저장 하려고 하는 경우에도 자동으로 변경 내용을 커밋하는 응용 프로그램을 구성할 수 있습니다.

> **NOTE**
> 디자이너는 폼에 끌어 놓은 첫 번째 항목에 대해서만 `BindingSource.EndEdit` 코드를 추가 합니다. 따라서 폼의 각 BindingSource에 대 한 EndEdit 메서드를 호출 하는 코드 줄을 추가 해야 합니다. 각 BindingSource에 대 한 EndEdit 메서드를 호출 하는 코드 줄을 수동으로 추가할 수 있습니다. 또는 폼에 `EndEditOnAllBindingSources` 메서드를 추가 하 고 저장을 수행 하기 전에 호출할 수 있습니다.

다음 코드는 LINQ (통합 언어 쿼리) 쿼리를 사용 하 여 모든 BindingSource 구성 요소를 반복 하 고 폼의 각 BindingSource에 대해 EndEdit 메서드를 호출 합니다.

폼의 모든 BindingSource 구성 요소에 대해 EndEdit를 호출 하려면

1. BindingSource 구성 요소를 포함 하는 폼에 다음 코드를 추가 합니다.

```
private void EndEditOnAllBindingSources()
{
    var BindingSourcesQuery =
        from Component bindingSources in this.components.Components
        where bindingSources is BindingSource
        select bindingSources;

    foreach (BindingSource bindingSource in BindingSourcesQuery)
    {
        bindingSource.EndEdit();
    }
}
```

```
Private Sub EndEditOnAllBindingSources()
    Dim BindingSourcesQuery = From bindingsources In Me.components.Components
                  Where (TypeOf bindingsources Is Windows.Forms.BindingSource)
                  Select bindingsources

    For Each bindingSource As Windows.Forms.BindingSource In BindingSourcesQuery
        bindingSource.EndEdit()
    Next
End Sub
```

2. 폼의 데이터를 저장 하기 위한 호출 바로 앞에 다음 코드 줄을 추가 합니다 (`TableAdapterManager.UpdateAll()` 메서드).

```
EndEditOnAllBindingSources();
```

```
Me.EndEditOnAllBindingSources()
```

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩
- 계층적 업데이트

Windows Forms 애플리케이션에서 조회 테이블 만들기

2020-01-06 • 8 minutes to read • Edit Online

용어 *조회 표에서* 는 두 개의 관련 된 데이터 테이블에 바인딩된 컨트롤을 설명 합니다. 이러한 조회 컨트롤은 두 번째 테이블에서 선택한 값에 따라 첫 번째 테이블의 데이터를 표시 합니다.

부모 테이블의 주 노드 (데이터 소스 창에서)를 연결 된 자식 테이블의 열에 이미 바인딩되어 있는 폼의 컨트롤로 끌어 조회 테이블을 만들 수 있습니다.

예를 들어 sales 데이터베이스에 `Orders` 테이블이 있다고 가정 합니다. `Orders` 테이블의 각 레코드에는 주문이 배치 된 고객을 나타내는 `CustomerID` 포함 되어 있습니다. `CustomerID` 은 `Customers` 테이블의 고객 레코드를 가리키는 외래 키입니다. 이 시나리오에서는 **데이터 소스** 창에서 `Orders` 테이블을 확장 하 고 주 노드를 **자세히** 로 설정 합니다. 그런 다음 ComboBox (또는 조회 바인딩을 지 원하는 다른 컨트롤)를 사용 하도록 `CustomerID` 열을 설정 하고 `Orders` 노드를 폼으로 끌어 옵니다. 마지막으로 `Customers` 노드를 관련 열에 바인딩된 컨트롤 (이 경우에는 ComboBox `CustomerID` 열에 바인딩)로 끌어옵니다.

조회 컨트롤을 바인딩하려면

1. 프로젝트를 연 상태에서 **다른 Windows** > **데이터 원본** > **보기** 를 선택 하 여 **데이터 소스** 창을 엽니다.

   > **NOTE**
   > 조회 테이블을 사용 하려면 **데이터 소스** 창에서 두 개의 관련 테이블이 나 개체를 사용할 수 있어야 합니다. 자세한 내용은 데이터 집합의 관계를 참조 하세요.

2. 부모 테이블 및 모든 열과 관련 자식 테이블 및 모든 열을 볼 수 있을 때까지 **데이터 소스** 창에서 노드를 확장 합니다.

   > **NOTE**
   > 자식 테이블 노드는 부모 테이블에서 확장 가능한 자식 노드로 표시 되는 노드입니다.

3. 자식 테이블의 노드에 있는 컨트롤 목록에서 **세부 정보** 를 선택 하 여 자식 테이블의 놓기 형식을 **자세히** 로 변경 합니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.

4. 두 테이블을 연결 하는 노드 (이전 예제의 `CustomerID` 노드)를 찾습니다. 컨트롤 목록에서 **ComboBox** 를 선택 하 여 삭제 유형을 ComboBox로 변경 합니다.

5. 주 자식 테이블 노드를 **데이터 소스** 창에서 폼으로 끌어 옵니다.

   데이터 바인딩된 컨트롤 (설명 레이블 포함)과 도구 스트립 (BindingNavigator)이 폼에 나타납니다. 데이터 집합, TableAdapter, BindingSource및 BindingNavigator 구성 요소 트레이에 나타납니다.

6. 이제 **데이터 소스** 창에서 조회 컨트롤 (ComboBox)로 직접 주 부모 테이블 노드를 끌어 옵니다.

   이제 조회 바인딩이 설정 됩니다. 컨트롤에 설정 된 특정 속성은 다음 표를 참조 하세요.

속성	설정 설명
DataSource	Visual Studio는 사용자가 컨트롤로 끌어 온 테이블에 대해 작성된 BindingSource로 이 속성을 설정합니다. 컨트롤을 만들 때 작성된 BindingSource가 아닙니다. 조정 해야 하는 경우 표시 하려는 열이 포함 된 테이블의 BindingSource 설정 합니다.
DisplayMember	Visual Studio는 컨트롤로 끄는 테이블에 대해 문자열 데이터 형식을 포함하는 기본 키 다음의 첫 번째 열로 이 속성을 설정합니다. 조정 해야 하는 경우 표시 하려는 열 이름으로 설정 합니다.
ValueMember	Visual Studio는 이 속성을 기본 키에 포함되는 첫 번째 열로 설정하거나 키가 정의되어 있지 않으면 테이블의 첫 번째 열로 설정합니다. 조정 해야 하는 경우 표시 하려는 열이 포함 된 테이블의 기본 키로 설정 합니다.
SelectedValue	Visual Studio는이 속성을 **데이터 소스** 창에서 삭제 된 원래 열로 설정 합니다. 조정 해야 하는 경우 관련 테이블의 외래 키 열로 설정 합니다.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩

데이터 검색을 위한 Windows Form 만들기

2020-01-06 • 15 minutes to read • Edit Online

일반적인 애플리케이션 시나리오에서는 선택한 데이터를 폼에 표시합니다. 특정 고객의 주문이나 특정 주문의 정보를 표시하려는 경우를 예로 들 수 있습니다. 이 시나리오에서는 사용자가 폼에 정보를 입력하면 해당 사용자의 입력을 매개 변수로 사용하여 쿼리가 실행됩니다. 즉 매개 변수가 있는 쿼리를 기준으로 데이터가 선택됩니다. 쿼리는 사용자가 입력한 기준을 만족하는 데이터만 반환합니다. 이 연습에서는 특정 구/군/시의 고객을 반환하는 쿼리를 만들고 사용자 인터페이스를 수정하여, 사용자가 구/군/시 이름을 입력한 후 단추를 눌러 쿼리를 실행할 수 있도록 하는 방법을 보여줍니다.

매개 변수가 있는 쿼리를 사용하면 데이터베이스가 레코드를 빠르게 필터링하도록 함으로써 애플리케이션의 효율성을 높일 수 있습니다. 반면 전체 데이터베이스 테이블을 요청하여 네트워크를 통해 전송한 다음, 애플리케이션 논리를 사용하여 원하는 레코드를 찾는 경우 애플리케이션의 속도와 효율성이 떨어질 수 있습니다.

검색 조건 작성기 대화 상자를 사용 하 여 매개 변수가 있는 쿼리를 TableAdapter에 추가 하 고 매개 변수 값을 허용 하 고 쿼리를 실행 하는 컨트롤을 제어할 수 있습니다. **데이터** 메뉴 또는 TableAdapter 스마트 태그에서 **쿼리 추가** 명령을 선택하여 대화 상자를 엽니다.

이 연습에서 설명하는 작업은 다음과 같습니다.

- **데이터 소스 구성** 마법사를 사용 하 여 응용 프로그램에서 데이터 소스를 만들고 구성 합니다.
- **데이터 소스** 창에서 항목의 놓기 형식을 설정 합니다.
- **데이터 원본** 창에서 양식으로 항목을 끌어 데이터를 표시하는 컨트롤을 만듭니다.
- 폼에 데이터를 표시하기 위한 컨트롤을 추가합니다.
- **검색 조건 작성기** 대화 상자를 완료 합니다.
- 매개 변수를 폼에 입력 하 고 매개 변수가 있는 쿼리를 실행 합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.

2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 **Visual Studio 설치 관리자**에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 응용 프로그램 만들기

또는 Visual Basic에 대 한 새 **Windows Forms 앱** 프로젝트를 만듭니다. C# 프로젝트 이름을 **WindowsSearchForm**로 지정합니다.

데이터 원본 만들기

이 단계에서는 **데이터 원본 구성** 마법사를 사용하여 데이터베이스에서 데이터 원본을 만듭니다.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.
2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성** 마법사를 시작합니다.
3. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택하고 **다음**을 클릭합니다.
4. **데이터 연결 선택** 페이지에서 다음 중 한 가지를 수행합니다.
   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.
5. 데이터베이스에 암호가 필요하면 중요한 데이터를 포함하는 옵션을 선택한 후, **다음**을 클릭합니다.
6. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 클릭 합니다.
7. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
8. **고객** 테이블을 선택한 다음, **마침**을 클릭합니다.

   NorthwindDataSet가 프로젝트에 추가되면 **고객** 테이블이 **데이터 원본** 창에 나타납니다.

양식 만들기

데이터 원본 창에서 양식으로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다.

1. **데이터 원본** 창에서 **Customers** 노드를 확장합니다.
2. **고객** 노드를 **데이터 원본** 창에서 양식으로 끌어옵니다.

   DataGridView와 레코드 탐색에 사용되는 도구 스트립(BindingNavigator)이 폼에 나타납니다. NorthwindDataSet, CustomersTableAdapter, BindingSource 및 BindingNavigator가 구성 요소 트레이에 나타납니다.

쿼리에 매개 변수화 (검색 기능) 추가

검색 조건 작성기 대화 상자를 사용 하 여 원래 쿼리에 WHERE 절을 추가할 수 있습니다.

1. DataGridView 컨트롤을 선택한 다음, **데이터** 메뉴에서 **쿼리 추가**를 선택합니다.
2. **검색 조건 작성기** 대화 상자의 **새 쿼리 이름** 영역에 **fillbycity** 를 입력 합니다.
3. 쿼리의 **쿼리 텍스트** 영역에 `WHERE City = @City`를 추가합니다.

   이 쿼리는 다음과 같아야 합니다.

```
SELECT CustomerID, CompanyName, ContactName, ContactTitle,
     Address, City, Region, PostalCode, Country, Phone, Fax
FROM Customers
WHERE City = @City
```

> **NOTE**
> Access 및 OLE DB 데이터 원본은 물음표 ('? ')를 사용 하 여 매개 변수를 표시 하므로 WHERE 절은 `WHERE City = ?` 와 같습니다.

4. **확인**을 클릭하여 **검색 조건 작성기** 대화 상자를 닫습니다.

   FillByCityToolStrip이 양식에 추가됩니다.

응용 프로그램 테스트

응용 프로그램을 실행 하면 양식이 열리고 매개 변수를 입력으로 사용할 수 있습니다.

1. **F5** 키를 눌러 애플리케이션을 실행합니다.
2. **도시** 텍스트 상자에 **런던**을 입력한 다음, **FillByCity**를 클릭합니다.

   데이터 표는 조건을 충족 하는 고객으로 채워집니다. 이 예제의 데이터 표에는 **도시** 열의 값이 **런던**인 고객만 표시됩니다.

다음 단계

애플리케이션 요구 사항에 따라 매개 변수가 있는 폼을 만든 후 몇 단계를 더 수행해야 할 수도 있습니다. 이 연습에서 보완할 수 있는 사항은 다음과 같습니다.

- 관련 데이터를 표시하는 컨트롤을 추가합니다. 자세한 내용은 데이터 집합의 관계를 참조 하세요.
- 데이터 세트을 편집하여 데이터베이스 개체를 추가하거나 편집합니다. 자세한 내용은 데이터 세트 만들기 및 구성을 참조하세요.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩

단순 데이터 바인딩을 지원하는 Windows Forms 사용자 정의 컨트롤 만들기

2020-01-06 • 19 minutes to read • Edit Online

Windows 애플리케이션에서 폼에 데이터를 표시할 때는 **도구 상자**에서 기존 컨트롤을 선택할 수도 있고, 표준 컨트롤에서는 제공되지 않는 기능이 애플리케이션에 필요한 경우에는 사용자 지정 컨트롤을 작성할 수도 있습니다. 이 연습에서는 DefaultBindingPropertyAttribute를 구현하는 컨트롤을 만드는 방법을 보여줍니다. DefaultBindingPropertyAttribute를 구현하는 컨트롤은 데이터에 바인딩할 수 있는 속성 한 개를 포함할 수 있습니다. 이러한 컨트롤은 TextBox 또는 CheckBox와 비슷합니다.

컨트롤 작성에 대 한 자세한 내용은 디자인 타임에 Windows Forms 컨트롤 개발을 참조 하세요.

데이터 바인딩 시나리오에 사용할 컨트롤을 작성할 때는 다음 데이터 바인딩 특성 중 하나를 구현 해야 합니다.

데이터 바인딩 특성 사용
단일 데이터 열이나 속성을 표시하는 DefaultBindingPropertyAttribute 등의 단순 컨트롤에 대해 TextBox를 구현합니다. 이 연습 페이지에서 해당 프로세스에 대해 설명합니다.
데이터 목록 또는 테이블을 표시하는 ComplexBindingPropertiesAttribute 등의 컨트롤에 대해 DataGridView를 구현합니다. 자세한 내용은 복합 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.
데이터 목록 또는 테이블을 표시하는 동시에 단일 열이나 속성도 제공해야 하는 LookupBindingPropertiesAttribute 등의 컨트롤에 대해 ComboBox를 구현합니다. 자세한 내용은 조회 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.

이 연습에서는 테이블 내 단일 열의 데이터를 표시하는 단순 컨트롤을 만듭니다. 이 예에서는 Northwind 샘플 데이터베이스 `Phone` 테이블의 `Customers` 열을 사용합니다. 간단한 사용자 정의 컨트롤은 MaskedTextBox를 사용하 고 마스크를 전화 번호로 설정 하 여 고객의 전화 번호를 표준 전화 번호 형식으로 표시 합니다.

이 연습에서는 다음 작업을 수행하는 방법을 배웁니다.

- 새 **Windows Forms 애플리케이션**을 만듭니다.
- 프로젝트에 새 **사용자 정의 컨트롤**을 추가합니다.
- 사용자 컨트롤을 시각적으로 디자인합니다.
- `DefaultBindingProperty` 특성을 구현합니다.
- **데이터 소스 구성** 마법사를 사용 하 여 데이터 집합을 만듭니다.
- 새 컨트롤을 사용하도록 **데이터 원본** 창의 **Phone** 열을 설정합니다.
- 새 컨트롤에 데이터를 표시할 폼을 만듭니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.

2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 **Visual Studio 설치 관리자**에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 애플리케이션 만들기

첫 번째 단계는 **Windows Forms 응용 프로그램**을 만드는 것입니다.

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **SimpleControlWalkthrough**로 지정한 다음 **확인**을 선택 합니다.

   SimpleControlWalkthrough 프로젝트가 만들어져 **솔루션 탐색기**에 추가됩니다.

프로젝트에 사용자 정의 컨트롤 추가

이 연습에서는 **사용자 정의 컨트롤**에서 간단한 데이터 바인딩 가능 컨트롤을 만듭니다.
SimpleControlWalkthrough 프로젝트에 **사용자 정의 컨트롤** 항목을 추가 합니다.

1. **프로젝트** 메뉴에서 **사용자 컨트롤 추가**를 선택합니다.
2. 이름 영역에 **PhoneNumberBox**를 입력하고 **추가**를 클릭합니다.

   PhoneNumberBox 컨트롤이 **솔루션 탐색기**에 추가되고 디자이너에서 열립니다.

PhoneNumberBox 컨트롤 디자인

이 연습은 기존 MaskedTextBox를 확장 하 여 **PhoneNumberBox** 컨트롤을 만듭니다.

1. MaskedTextBox도구 상자**에서 사용자 컨트롤의 디자인 화면으로** 를 끌어 옵니다.
2. 방금 끌어 온 MaskedTextBox에서 스마트 태그를 선택하고 **마스크 설정**을 선택합니다.
3. **입력 마스크** 대화 상자에서 **전화 번호**를 선택하고 **확인**을 클릭하여 마스크를 설정합니다.

필요한 데이터 바인딩 특성 추가

데이터 바인딩을 지원하는 단순 컨트롤에 대해 DefaultBindingPropertyAttribute를 구현합니다.

1. **PhoneNumberBox** 컨트롤을 코드 보기로 전환 합니다. (**보기** 메뉴에서 **코드**를 선택합니다.)
2. **PhoneNumberBox** 의 코드를 다음으로 바꿉니다.

```
using System.Windows.Forms;

namespace CS
{
    [System.ComponentModel.DefaultBindingProperty("PhoneNumber")]
    public partial class PhoneNumberBox : UserControl
    {
        public string PhoneNumber
        {
            get{ return maskedTextBox1.Text; }
            set{ maskedTextBox1.Text = value; }
        }

        public PhoneNumberBox()
        {
            InitializeComponent();
        }
    }
}
```

```
<System.ComponentModel.DefaultBindingProperty("PhoneNumber")>
Public Class PhoneNumberBox

    Public Property PhoneNumber() As String
        Get
            Return MaskedTextBox1.Text
        End Get
        Set(ByVal value As String)
            MaskedTextBox1.Text = value
        End Set
    End Property
End Class
```

3. **빌드** 메뉴에서 **솔루션 빌드**를 선택합니다.

데이터베이스에서 데이터 원본 만들기

이 단계에서는 **데이터 원본 구성** 마법사를 사용하여 Northwind 샘플 데이터베이스의 `Customers` 테이블을 기반으로 하는 데이터 원본을 만듭니다. 연결을 만들려면 Northwind 샘플 데이터베이스에 액세스해야 합니다. Northwind 샘플 데이터베이스를 설정 하는 방법에 대 한 자세한 내용은 방법: 샘플 데이터베이스 설치를 참조 하세요.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.
2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성** 마법사를 시작합니다.
3. **데이터 원본 형식 선택** 페이지에서 **데이터베이스**를 선택한 후, **다음**을 클릭합니다.
4. **데이터 연결 선택** 페이지에서 다음 중 한 가지를 수행합니다.
   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.
5. 데이터베이스에 암호가 필요하면 중요한 데이터를 포함하는 옵션을 선택한 후, **다음**을 클릭합니다.
6. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 클릭 합니다.
7. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
8. `Customers` 테이블을 선택한 다음, **마침**을 클릭합니다.

NorthwindDataSet가 프로젝트에 추가되고 `Customers` 테이블이 **데이터 원본** 창에 나타납니다.

PhoneNumberBox 컨트롤을 사용 하도록 phone 열 설정

데이터 원본 창 내에서 항목을 폼으로 끌기 전에 만들 컨트롤을 설정할 수 있습니다.

1. 디자이너에서 **Form1**을 엽니다.
2. **데이터 원본** 창에서 **Customers** 노드를 확장합니다.
3. **Customers** 노드에서 드롭다운 화살표를 클릭하고 컨트롤 목록에서 **정보**를 선택합니다.
4. **Phone** 열에서 드롭다운 화살표를 클릭하고 **Customize**를 선택합니다.
5. **데이터 UI 사용자 지정 옵션** 대화 상자의 **연결된 컨트롤** 목록에서 **PhoneNumberBox**를 선택합니다.
6. **Phone** 열에서 드롭다운 화살표를 클릭하고 **PhoneNumberBox**를 선택합니다.

폼에 컨트롤 추가

데이터 원본 창에서 폼으로 항목을 끌어서 데이터 바인딩된 컨트롤을 만들 수 있습니다.

폼에서 데이터 바인딩된 컨트롤을 만들려면 주 **Customers** 노드를 **데이터 소스** 창에서 폼으로 끌고 **PhoneNumberBox** 컨트롤이 **휴대폰** 열의 데이터를 표시 하는 데 사용 되는지 확인 합니다.

설명 레이블이 있는 데이터 바인딩된 컨트롤이 레코드 탐색을 위한 도구 모음인 BindingNavigator와 함께 폼에 나타납니다. NorthwindDataSet, CustomersTableAdapter, BindingSource 및 BindingNavigator가 구성 요소 트레이에 나타납니다.

애플리케이션 실행

F5 키를 눌러 애플리케이션을 실행합니다.

다음 단계

애플리케이션 요구 사항에 따라 데이터 바인딩을 지원하는 컨트롤을 만든 후 몇 단계를 더 수행해야 할 수도 있습니다. 일반적으로 수행하는 몇 가지 단계는 다음과 같습니다.

- 다른 애플리케이션에서 다시 사용할 수 있도록 컨트롤 라이브러리에 사용자 지정 컨트롤을 배치합니다.
- 보다 복잡한 데이터 바인딩 시나리오를 지원하는 컨트롤을 만듭니다. 자세한 내용은 복합 데이터 바인딩을 지 원하는 Windows Forms 사용자 정의 컨트롤 만들기 및 조회 데이터 바인딩을 지 원하는 Windows Forms 사용자 정의 컨트롤 만들기를 참조 하세요.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩
- 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정

복합 데이터 바인딩을 지원하는 Windows Forms 사용자 정의 컨트롤 만들기

2020-01-06 • 17 minutes to read • Edit Online

Windows 응용 프로그램의 폼에 데이터를 표시할 때는 **도구 상자**에서 기존 컨트롤을 선택할 수 있습니다. 또는 응용 프로그램에 표준 컨트롤에서 사용할 수 없는 기능이 필요한 경우 사용자 지정 컨트롤을 제작할 수 있습니다. 이 연습에서는 ComplexBindingPropertiesAttribute를 구현하는 컨트롤을 만드는 방법을 보여줍니다. ComplexBindingPropertiesAttribute를 구현하는 컨트롤은 데이터에 바인딩할 수 있는 `DataSource` 및 `DataMember` 속성을 포함합니다. 이러한 컨트롤은 DataGridView 또는 ListBox와 비슷합니다.

컨트롤 작성에 대 한 자세한 내용은 디자인 타임에 Windows Forms 컨트롤 개발을 참조 하세요.

데이터 바인딩 시나리오에 사용할 컨트롤을 작성할 때는 다음 데이터 바인딩 특성 중 하나를 구현해야 합니다.

데이터 바인딩 특성 사용
단일 데이터 열이나 속성을 표시하는 DefaultBindingPropertyAttribute 등의 단순 컨트롤에 대해 TextBox를 구현합니다. 자세한 내용은 단순 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.
데이터 목록 또는 테이블을 표시하는 ComplexBindingPropertiesAttribute 등의 컨트롤에 대해 DataGridView를 구현합니다. 이 연습 페이지에서 해당 프로세스에 대해 설명합니다.
데이터 목록 또는 테이블을 표시하는 동시에 단일 열이나 속성도 제공해야 하는 LookupBindingPropertiesAttribute 등의 컨트롤에 대해 ComboBox를 구현합니다. 자세한 내용은 조회 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.

이 연습에서는 테이블의 데이터 행을 표시하는 복합 컨트롤을 만듭니다. 이 예에서는 Northwind 샘플 데이터베이스의 `Customers` 테이블을 사용합니다. 복합 사용자 컨트롤은 사용자 지정 컨트롤의 DataGridView에 Customers 테이블을 표시합니다.

이 연습에서는 다음 작업을 수행 하는 방법에 대해 알아봅니다.

- 프로젝트에 새 **사용자 정의 컨트롤**을 추가합니다.
- 사용자 컨트롤을 시각적으로 디자인합니다.
- `ComplexBindingProperty` 특성을 구현합니다.
- 데이터 소스 구성 마법사를 사용 하 여 데이터 집합을 만듭니다.
- 새 복합 컨트롤을 사용 하도록 데이터 소스 창 에서 **Customers** 테이블을 설정 합니다.
- 새 컨트롤을 **데이터 원본** 창에서 **Form1**으로 끌어 추가합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 앱 프로젝트 만들기

첫 번째 단계는 C# 또는 Visual Basic에 대 한 **Windows Forms 앱** 프로젝트를 만드는 것입니다. 프로젝트 이름을 **ComplexControlWalkthrough**로 지정합니다.

프로젝트에 사용자 정의 컨트롤 추가

이 연습에서는 **사용자 정의 컨트롤**에서 복잡 한 데이터 바인딩 가능 컨트롤을 만들기 때문에 **사용자 정의 컨트롤** 항목을 프로젝트에 추가 합니다.

1. **프로젝트** 메뉴에서 **사용자 컨트롤 추가**를 선택합니다.
2. **이름** 영역에 **ComplexDataGridView**를 입력하고 **추가**를 클릭합니다.

   ComplexDataGridView 컨트롤이 **솔루션 탐색기**에 추가되고 디자이너에서 열립니다.

ComplexDataGridView 컨트롤 디자인

사용자 정의 컨트롤에 DataGridView를 추가 하려면 **도구 상자** 에서 사용자 컨트롤의 디자인 화면으로 DataGridView을 끌어 옵니다.

필요한 데이터 바인딩 특성 추가

데이터 바인딩을 지원하는 복합 컨트롤에 대해 ComplexBindingPropertiesAttribute를 구현할 수 있습니다.

1. **ComplexDataGridView** 컨트롤을 코드 보기로 전환합니다. (**보기** 메뉴에서 **코드**를 선택합니다.)
2. `ComplexDataGridView`의 코드를 다음 코드로 바꿉니다.

```
using System.Windows.Forms;

namespace CS
{
    [System.ComponentModel.ComplexBindingProperties("DataSource", "DataMember")]
    public partial class ComplexDataGridView : UserControl
    {
        public object DataSource
        {
            get{ return dataGridView1.DataSource; }
            set{ dataGridView1.DataSource = value; }
        }

        public string DataMember
        {
            get{ return dataGridView1.DataMember; }
            set{ dataGridView1.DataMember = value; }
        }

        public ComplexDataGridView()
        {
            InitializeComponent();
        }
    }
}
```

```
<System.ComponentModel.ComplexBindingProperties("DataSource", "DataMember")>
Public Class ComplexDataGridView

    Public Property DataSource() As Object
        Get
            Return DataGridView1.DataSource
        End Get
        Set(ByVal value As Object)
            DataGridView1.DataSource = value
        End Set
    End Property

    Public Property DataMember() As String
        Get
            Return DataGridView1.DataMember
        End Get
        Set(ByVal value As String)
            DataGridView1.DataMember = value
        End Set
    End Property
End Class
```

3. **빌드** 메뉴에서 **솔루션 빌드**를 선택합니다.

데이터베이스에서 데이터 원본 만들기

데이터 소스 구성 마법사를 사용 하 여 Northwind 샘플 데이터베이스의 `Customers` 테이블을 기반으로 데이터 원본을 만듭니다.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.

2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성** 마법사를 시작합니다.

3. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택하고 **다음**을 클릭합니다.

4. **데이터 연결 선택** 페이지에서 다음 중 한 가지를 수행합니다.

- Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
- **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.

5. 데이터베이스에 암호가 필요하면 중요한 데이터를 포함하는 옵션을 선택한 후, **다음**을 클릭합니다.
6. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 클릭 합니다.
7. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
8. `Customers` 테이블을 선택한 다음, **마침**을 클릭합니다.

   NorthwindDataSet가 프로젝트에 추가되고 `Customers` 테이블이 **데이터 원본** 창에 나타납니다.

ComplexDataGridView 컨트롤을 사용 하도록 Customers 테이블 설정

데이터 원본 창 내에서 항목을 폼으로 끌기 전에 만들 컨트롤을 설정할 수 있습니다.

1. 디자이너에서 **Form1**을 엽니다.
2. **데이터 원본** 창에서 **Customers** 노드를 확장합니다.
3. **Customers** 노드에서 드롭다운 화살표를 클릭하고 **Customize**를 선택합니다.
4. **데이터 UI 사용자 지정 옵션** 대화 상자의 **연결된 컨트롤** 목록에서 **ComplexDataGridView**를 선택합니다.
5. `Customers` 테이블에서 드롭다운 화살표를 클릭하고 컨트롤 목록에서 **ComplexDataGridView**를 선택합니다.

폼에 컨트롤 추가

데이터 원본 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다. 주 **Customers** 노드를 **데이터 원본** 창에서 폼으로 끌어서 놓습니다. **Complexdatagridview** 컨트롤이 테이블의 데이터를 표시 하는 데 사용 되는지 확인 합니다.

애플리케이션 실행

F5 키를 눌러 애플리케이션을 실행합니다.

다음 단계

애플리케이션 요구 사항에 따라 데이터 바인딩을 지원하는 컨트롤을 만든 후 몇 단계를 더 수행해야 할 수도 있습니다. 일반적으로 수행하는 몇 가지 단계는 다음과 같습니다.

- 다른 애플리케이션에서 다시 사용할 수 있도록 컨트롤 라이브러리에 사용자 지정 컨트롤을 배치합니다.
- 조회 시나리오를 지원하는 컨트롤을 만듭니다. 자세한 내용은 조회 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩
- 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정
- Windows Forms 컨트롤

조회 데이터 바인딩을 지원하는 Windows Forms 사용자 정의 컨트롤 만들기

2020-01-06 • 18 minutes to read • Edit Online

Windows Forms에 데이터를 표시할 때는 **도구 상자**에서 기존 컨트롤을 선택할 수도 있고, 표준 컨트롤에서는 제공되지 않는 기능이 애플리케이션에 필요한 경우에는 사용자 지정 컨트롤을 작성할 수도 있습니다. 이 연습에서는 LookupBindingPropertiesAttribute를 구현하는 컨트롤을 만드는 방법을 보여줍니다. LookupBindingPropertiesAttribute를 구현하는 컨트롤은 데이터에 바인딩할 수 있는 속성 3개를 포함할 수 있습니다. 이러한 컨트롤은 ComboBox와 비슷합니다.

컨트롤 작성에 대 한 자세한 내용은 디자인 타임에 Windows Forms 컨트롤 개발을 참조 하세요.

데이터 바인딩 시나리오에 사용할 컨트롤을 작성할 때는 다음 데이터 바인딩 특성 중 하나를 구현해야 합니다.

데이터 바인딩 특성 사용
단일 데이터 열이나 속성을 표시하는 DefaultBindingPropertyAttribute 등의 단순 컨트롤에 대해 TextBox를 구현합니다. 자세한 내용은 단순 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.
데이터 목록 또는 테이블을 표시하는 ComplexBindingPropertiesAttribute 등의 컨트롤에 대해 DataGridView를 구현합니다. 자세한 내용은 복합 데이터 바인딩을 지 원하는 사용자 정의 컨트롤 Windows Forms 만들기를 참조 하세요.
데이터 목록 또는 테이블을 표시하는 동시에 단일 열이나 속성도 제공해야 하는 LookupBindingPropertiesAttribute 등의 컨트롤에 대해 ComboBox를 구현합니다. 이 연습 페이지에서 해당 프로세스에 대해 설명합니다.

이 연습에서는 두 테이블의 데이터에 바인딩되는 조회 컨트롤을 만듭니다. 이 예에서는 Northwind 샘플 데이터베이스의 `Customers` 및 `Orders` 테이블을 사용합니다. 조회 컨트롤은 `Orders` 테이블의 `CustomerID` 필드에 바인딩됩니다. 이 값을 사용 하 여 `Customers` 테이블에서 `CompanyName`를 조회 합니다.

이 연습에서는 다음 작업을 수행 하는 방법에 대해 알아봅니다.

- 새 **Windows Forms 애플리케이션**을 만듭니다.
- 프로젝트에 새 **사용자 정의 컨트롤**을 추가합니다.
- 사용자 컨트롤을 시각적으로 디자인합니다.
- `LookupBindingProperty` 특성을 구현합니다.
- **데이터 소스 구성** 마법사를 사용 하 여 데이터 집합을 만듭니다.
- 새 컨트롤을 사용하도록 **데이터 원본** 창에서 **Orders** 테이블의 **CustomerID** 열을 설정합니다.
- 새 컨트롤에 데이터를 표시할 폼을 만듭니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 앱 프로젝트 만들기

첫 번째 단계는 **Windows Forms 응용 프로그램** 프로젝트를 만드는 것입니다.

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창 **에서 C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **Lookupcontrolwalkthrough**으로 지정한 다음 **확인**을 선택 합니다.

   LookupControlWalkthrough 프로젝트가 생성되고 **솔루션 탐색기**에 추가됩니다.

프로젝트에 사용자 정의 컨트롤 추가

이 연습에서는 **사용자 정의 컨트롤**에서 조회 컨트롤을 만들 것이므로 **사용자 정의 컨트롤** 항목을 **LookupControlWalkthrough** 프로젝트에 추가합니다.

1. **프로젝트** 메뉴에서 **사용자 정의 컨트롤 추가**를 선택합니다.
2. **이름** 영역에 `LookupBox`를 입력 한 다음 **추가**를 클릭 합니다.

   LookupBox 컨트롤이 **솔루션 탐색기**에 추가되고 디자이너에서 열립니다.

LookupBox 컨트롤 디자인

LookupBox 컨트롤을 디자인 하려면 ComboBox를 **도구 상자** 에서 사용자 컨트롤의 디자인 화면으로 끌어 옵니다.

필요한 데이터 바인딩 특성 추가

데이터 바인딩을 지원하는 조회 컨트롤에 대해 LookupBindingPropertiesAttribute를 구현할 수 있습니다.

1. **LookupBox** 컨트롤을 코드 보기로 전환합니다. (**보기** 메뉴에서 **코드**를 선택합니다.)
2. `LookupBox`의 코드를 다음 코드로 바꿉니다.

```
<System.ComponentModel.LookupBindingProperties("DataSource", "DisplayMember", "ValueMember",
"LookupMember")>
Public Class LookupBox

    Public Property DataSource() As Object
        Get
            Return ComboBox1.DataSource
        End Get
        Set(ByVal value As Object)
            ComboBox1.DataSource = value
        End Set
    End Property

    Public Property DisplayMember() As String
        Get
            Return ComboBox1.DisplayMember
        End Get
        Set(ByVal value As String)
            ComboBox1.DisplayMember = value
        End Set
    End Property

    Public Property ValueMember() As String
        Get
            Return ComboBox1.ValueMember
        End Get
        Set(ByVal value As String)
            ComboBox1.ValueMember = value
        End Set
    End Property

    Public Property LookupMember() As String
        Get
            Return ComboBox1.SelectedValue.ToString()
        End Get
        Set(ByVal value As String)
            ComboBox1.SelectedValue = value
        End Set
    End Property
End Class
```

```
using System.Windows.Forms;

namespace CS
{
    [System.ComponentModel.LookupBindingProperties("DataSource", "DisplayMember", "ValueMember",
"LookupMember")]
    public partial class LookupBox : UserControl
    {
        public object DataSource
        {
            get{ return comboBox1.DataSource; }
            set{ comboBox1.DataSource = value; }
        }

        public string DisplayMember
        {
            get{ return comboBox1.DisplayMember; }
            set{ comboBox1.DisplayMember = value; }
        }

        public string ValueMember
        {
            get{ return comboBox1.ValueMember; }
            set{ comboBox1.ValueMember = value; }
        }

        public string LookupMember
        {
            get{ return comboBox1.SelectedValue.ToString(); }
            set{ comboBox1.SelectedValue = value; }
        }


        public LookupBox()
        {
            InitializeComponent();
        }
    }
}
```

3. **빌드** 메뉴에서 **솔루션 빌드**를 선택합니다.

데이터베이스에서 데이터 원본 만들기

이 단계에서는 **데이터 원본 구성** 마법사를 사용하여 Northwind 샘플 데이터베이스의 `Customers` 및 `Orders` 테이블을 기반으로 하는 데이터 원본을 만듭니다.

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.
2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성** 마법사를 시작합니다.
3. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택하고 **다음**을 클릭합니다.
4. **데이터 연결 선택** 페이지에서 다음 중 한 가지를 수행합니다.
   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.
5. 데이터베이스에 암호가 필요하면 중요한 데이터를 포함하는 옵션을 선택한 후, **다음**을 클릭합니다.
6. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 클릭 합니다.

7. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.

8. `Customers` 및 `Orders` 테이블을 선택한 다음, **마침**을 클릭합니다.

   NorthwindDataSet가 프로젝트에 추가되고 `Customers` 및 `Orders` 테이블이 **데이터 원본** 창에 나타납니다.

LookupBox 컨트롤을 사용 하도록 Orders 테이블의 CustomerID 열 설정

데이터 원본 창 내에서 항목을 폼으로 끌기 전에 만들 컨트롤을 설정할 수 있습니다.

1. 디자이너에서 **Form1**을 엽니다.

2. **데이터 원본** 창에서 **Customers** 노드를 확장합니다.

3. **Orders** 노드(**Customers** 노드에서 **Fax** 열 아래의 노드)를 확장합니다.

4. **Orders** 노드에서 드롭다운 화살표를 클릭하고 컨트롤 목록에서 **Details**를 선택합니다.

5. **Orders** 노드에서 **CustomerID** 열의 드롭다운 화살표를 클릭하고 **Customize**를 선택합니다.

6. **데이터 UI 사용자 지정 옵션** 대화 상자의 **연결된 컨트롤** 목록에서 **LookupBox**를 선택합니다.

7. **확인**을 클릭합니다.

8. **CustomerID** 열에서 드롭다운 화살표를 클릭하고 **LookupBox**를 선택합니다.

폼에 컨트롤 추가

데이터 원본 창에서 **Form1**로 항목을 끌어 데이터 바인딩된 컨트롤을 만들 수 있습니다.

Windows Form에서 데이터 바인딩된 컨트롤을 만들려면 **Orders** 노드를 **데이터 소스** 창에서 Windows Form으로 끌고, **lookupbox** 컨트롤이 `CustomerID` 열의 데이터를 표시 하는 데 사용 되는지 확인 합니다.

Customers 테이블에서 CompanyName을 조회 하는 컨트롤 바인딩

조회 바인딩을 설정 하려면 **데이터 소스** 창에서 주 **Customers** 노드를 선택 하 고이를 **Form1**의 **customeridlookupbox** 에 있는 콤보 상자로 끌어 놓습니다.

그러면 `Customers` 테이블의 `CompanyName`이 표시되도록 데이터 바인딩이 설정되고 `Orders` 테이블의 `CustomerID` 값은 유지됩니다.

애플리케이션 실행

- **F5** 키를 눌러 애플리케이션을 실행합니다.
- 일부 레코드를 탐색해 보고 `CompanyName`이 `LookupBox` 컨트롤에 표시되는지 확인합니다.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩

폼 간에 데이터 전달

2020-01-06 • 17 minutes to read • Edit Online

이 연습에서는 폼 간에 데이터를 전달하기 위한 단계별 지침을 제공합니다. Northwind의 customers 및 orders 테이블을 사용 하면 한 폼에서 고객을 선택할 수 있으며, 두 번째 폼에는 선택한 고객의 주문이 표시 됩니다. 이 연습에서는 첫 번째 폼에서 데이터를 받는 두 번째 폼에서 메서드를 만드는 방법을 보여 줍니다.

> **NOTE**
> 이 연습에서는 폼 간에 데이터를 전달하는 방식 중 하나만을 보여줍니다. 데이터를 수신 하기 위한 두 번째 생성자 만들기를 비롯 하 여 폼에 데이터를 전달 하는 다른 옵션이 있습니다. 또는 첫 번째 폼의 데이터로 설정할 수 있는 공용 속성을 만듭니다.

이 연습에서 설명하는 작업은 다음과 같습니다.

- 새 **Windows Forms 응용 프로그램** 프로젝트를 만듭니다.
- 데이터 소스 구성 마법사를 사용 하 여 데이터 집합 만들기 및 구성
- **데이터 원본** 창에서 항목을 끌어오는 경우 폼에 만들어질 컨트롤을 선택합니다. 자세한 내용은 데이터 소스 창에서 끌어올 때 만들 컨트롤 설정을 참조 하세요.
- **데이터 원본** 창에서 폼으로 항목을 끌어 데이터 바인딩된 컨트롤을 만듭니다.
- 데이터를 표시하는 표가 포함된 두 번째 폼을 만듭니다.
- 특정 고객의 주문을 가져오는 TableAdapter 쿼리를 만듭니다.
- 폼 간에 데이터를 전달합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. Visual Studio 설치 관리자에서 SQL Server Express LocalDB는 **데이터 저장소 및 처리** 워크 로드의 일부로 설치 되거나 개별 구성 요소로 설치 될 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.
   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. SQL Server 개체 탐색기는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

      쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

      잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

Windows Forms 앱 프로젝트 만들기

1. Visual Studio의 **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 차례로 선택합니다.
2. 왼쪽 창에서 **C# 시각적 개체** 또는 **Visual Basic** 을 확장 한 다음 **Windows 데스크톱**을 선택 합니다.
3. 가운데 창에서 **Windows Forms 앱** 프로젝트 형식을 선택 합니다.
4. 프로젝트 이름을 **PassingDataBetweenForms**로 지정한 다음 **확인**을 선택 합니다.

   PassingDataBetweenForms 프로젝트가 만들어져 **솔루션 탐색기**에 추가됩니다.

데이터 원본 만들기

1. 데이터 **소스** 창을 열려면 **데이터** 메뉴에서 **데이터 소스 표시**를 클릭 합니다.
2. **데이터 원본** 창에서 **새 데이터 원본 추가**를 선택하여 **데이터 원본 구성** 마법사를 시작합니다.
3. **데이터 소스 형식 선택** 페이지에서 **데이터베이스** 를 선택하고 **다음**을 클릭합니다.
4. **데이터베이스 모델 선택** 페이지에서 **데이터 세트**가 지정되어 있는지 확인한 후, **다음**을 클릭합니다.
5. **데이터 연결 선택** 페이지에서 다음 중 한 가지를 수행합니다.
   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.
   - **새 연결**을 선택하여 **연결 추가/수정** 대화 상자를 시작합니다.
6. 데이터베이스에 암호가 필요하며 중요한 데이터를 포함하도록 옵션이 설정되어 있으면 해당 옵션을 선택한 후, **다음**을 클릭합니다.
7. **응용 프로그램 구성 파일에 연결 문자열 저장** 페이지에서 **다음**을 클릭 합니다.
8. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장합니다.
9. **Customers** 및 **Orders** 테이블을 선택한 다음, **마침**을 클릭합니다.

   NorthwindDataSet가 프로젝트에 추가되고 **Customers** 및 **Orders** 테이블이 **데이터 원본** 창에 나타납니다.

첫 번째 폼 만들기 (Form1)

데이터 원본 창에서 폼으로 **Customers** 노드를 끌어 데이터 바인딩된 표(DataGridView)를 만들 수 있습니다.

폼에서 데이터 바인딩된 표를 만들려면

- 주 **Customers** 노드를 **데이터 원본** 창에서 **Form1**으로 끌어서 놓습니다.

  DataGridView와 레코드 탐색에 사용되는 도구 모음(BindingNavigator)이 **Form1**에 나타납니다. NorthwindDataSet, CustomersTableAdapter, BindingSource 및 BindingNavigator가 구성 요소 트레이에 나타납니다.

두 번째 폼 만들기

데이터를 전달할 두 번째 폼을 만듭니다.

1. **프로젝트** 메뉴에서 **Windows Form 추가**를 선택합니다.
2. 기본 이름인 **Form2**를 그대로 두고 **추가**를 클릭합니다.
3. 주 **Orders** 노드를 **데이터 원본** 창에서 **Form2**로 끌어 옵니다.

   DataGridView와 레코드 탐색에 사용되는 도구 모음(BindingNavigator)이 **Form2**에 나타납니다. NorthwindDataSet, CustomersTableAdapter, BindingSource 및 BindingNavigator가 구성 요소 트레이에 나타

납니다.

4. 구성 요소 트레이에서 **OrdersBindingNavigator**를 삭제합니다.

   OrdersBindingNavigator가 **Form2**에서 사라집니다.

TableAdapter 쿼리 추가

Form2에 TableAdapter 쿼리를 추가 하 여 Form1에서 선택한 고객에 대 한 주문을 로드 합니다.

1. **솔루션 탐색기**에서 **NorthwindDataSet.xsd** 파일을 두 번 클릭합니다.
2. **OrdersTableAdapter**를 마우스 오른쪽 단추로 클릭하고 **쿼리 추가**를 선택합니다.
3. 기본 옵션인 **SQL 문 사용**을 그대로 둔 후, **다음**을 클릭합니다.
4. 기본 옵션인 **행을 반환하는 SELECT**를 그대로 둔 후, **다음**을 클릭합니다.
5. 쿼리에 WHERE 절을 추가하여 `CustomerID`에 따라 `Orders`를 반환합니다. 이 쿼리는 다음과 같아야 합니다.

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight,
ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, ShipCountry
FROM Orders
WHERE CustomerID = @CustomerID
```

> **NOTE**
> 데이터베이스에 대한 올바른 매개 변수 구문을 확인합니다. 예를 들어 Microsoft Access에서 WHERE 절은 다음과 같습니다. `WHERE CustomerID = ?`.

6. **다음**을 클릭합니다.
7. **Fill a DataTableMethod Name**에 `FillByCustomerID`을 입력 합니다.
8. **DataTable 반환** 옵션 선택을 취소한 후, **다음**을 클릭합니다.
9. **마침**을 클릭합니다.

에 데이터를 전달 하는 메서드를 Form2에 만듭니다.

1. **Form2**를 마우스 오른쪽 단추로 클릭하고 **코드 보기**를 선택하여 **코드 편집기**에서 **Form2**를 엽니다.
2. 다음 코드를 **Form2**의 `Form2_Load` 메서드 뒤에 추가합니다.

```
Friend Sub LoadOrders(ByVal CustomerID As String)
    OrdersTableAdapter.FillByCustomerID(NorthwindDataSet.Orders, CustomerID)
End Sub
```

```
internal void LoadOrders(String CustomerID)
{
    ordersTableAdapter.FillByCustomerID(northwindDataSet.Orders, CustomerID);
}
```

Form1에서 데이터를 전달 하 고 Form2를 표시 하는 메서드를 만듭니다.

1. **Form1**에서 Customer 데이터 표를 마우스 오른쪽 단추로 클릭한 다음, **속성**을 클릭합니다.

2. **속성** 창에서 **이벤트**를 클릭합니다.
3. **CellDoubleClick** 이벤트를 두 번 클릭합니다.

   코드 편집기가 나타납니다.
4. 다음 샘플과 일치하도록 메서드 정의를 업데이트합니다.

```
private void customersDataGridView_DoubleClick(object sender, EventArgs e)
{
    System.Data.DataRowView SelectedRowView;
    NorthwindDataSet.CustomersRow SelectedRow;

    SelectedRowView = (System.Data.DataRowView)customersBindingSource.Current;
    SelectedRow = (NorthwindDataSet.CustomersRow)SelectedRowView.Row;

    Form2 OrdersForm = new Form2();
    OrdersForm.LoadOrders(SelectedRow.CustomerID);
    OrdersForm.Show();
}
```

```
Private Sub CustomersDataGridView_DoubleClick() Handles CustomersDataGridView.DoubleClick

    Dim SelectedRowView As Data.DataRowView
    Dim SelectedRow As NorthwindDataSet.CustomersRow

    SelectedRowView = CType(CustomersBindingSource.Current, System.Data.DataRowView)
    SelectedRow = CType(SelectedRowView.Row, NorthwindDataSet.CustomersRow)

    Dim OrdersForm As New Form2
    OrdersForm.LoadOrders(SelectedRow.CustomerID)
    OrdersForm.Show()
End Sub
```

앱 실행

- **F5** 키를 눌러 애플리케이션을 실행합니다.
- **Form1**에서 고객 레코드를 두 번 클릭하여 해당 고객의 주문이 포함된 **Form2**를 엽니다.

다음 단계

애플리케이션 요구 사항에 따라 폼 간에 데이터를 전달한 후 몇 단계를 더 수행해야 할 수도 있습니다. 이 연습에서 보완할 수 있는 사항은 다음과 같습니다.

- 데이터 세트을 편집하여 데이터베이스 개체를 추가하거나 편집합니다. 자세한 내용은 데이터 세트 만들기 및 구성을 참조하세요.
- 데이터를 데이터베이스로 다시 보내는 기능을 추가합니다. 자세한 내용은 데이터를 데이터베이스에 다시 저장을 참조 하세요.

참조

- Visual Studio에서 데이터에 Windows Forms 컨트롤 바인딩

Visual Studio에서 개체를 데이터 원본으로 바인딩

2020-01-06 • 20 minutes to read • Edit Online

Visual Studio는 사용자 지정 개체를 응용 프로그램의 데이터 소스로 사용 하기 위한 디자인 타임 도구를 제공 합니다. UI 컨트롤에 바인딩하는 개체의 데이터베이스에서 데이터를 저장 하려는 경우 Entity Framework를 사용 하 여 클래스를 생성 하는 것이 좋습니다. Entity Framework 모든 상용구 변경 추적 코드를 자동으로 생성 합니다. 즉, DbSet 개체에서 AcceptChanges를 호출 하면 로컬 개체에 대 한 모든 변경 내용이 데이터베이스에 자동으로 유지 됩니다. 자세한 내용은 Entity Framework 설명서를 참조 하세요.

> **TIP**
> 응용 프로그램이 이미 데이터 집합을 기반으로 하는 경우에만이 문서의 개체 바인딩에 대 한 접근 방식을 고려해 야 합니다. 데이터 집합에 이미 익숙한 경우에도 이러한 접근 방식을 사용할 수 있으며, 처리할 데이터는 테이블 형식이 며 너무 복잡 하거나 너무 크지 않습니다. DataReader를 사용 하 여 개체에 직접 데이터를 로드 하 고 데이터 바인딩을 사용 하지 않고 UI를 수동으로 업데이트 하는 것을 포함 하는 더 간단한 예제를 보려면 ADO.NET를 사용 하 여 간단한 데이터 응용 프로그램 만들기

개체 요구 사항

Visual Studio에서 데이터 디자인 도구를 사용 하는 사용자 지정 개체의 유일한 요구 사항은 개체에 public 속성이 하나 이상 있어야 한다는 것입니다.

일반적으로 사용자 지정 개체에는 응용 프로그램의 데이터 소스 역할을 하는 특정 인터페이스, 생성자 또는 특성이 필요 하지 않습니다. 그러나 데이터 **소스** 창에서 디자인 화면으로 개체를 끌어 데이터 바인딩된 컨트롤을 만들고 개체가 ITypedList 또는 IListSource 인터페이스를 구현 하는 경우 개체에는 기본 생성자가 있어야 합니다. 그렇지 않으면 Visual Studio에서 데이터 소스 개체를 인스턴스화할 수 없으며, 항목을 디자인 화면으로 끌 때 오류가 표시 됩니다.

사용자 지정 개체를 데이터 원본으로 사용 하는 예

개체를 데이터 원본으로 사용할 때 응용 프로그램 논리를 구현 하는 다양 한 방법이 있지만 SQL 데이터베이스에는 Visual Studio에서 생성 한 TableAdapter 개체를 사용 하 여 단순화할 수 있는 몇 가지 표준 작업이 있습니다. 이 페이지에서는 Tableadapter를 사용 하 여 이러한 표준 프로세스를 구현 하는 방법을 설명 합니다. 사용자 지정 개체를 만드는 방법에 대 한 지침을 제공 하지는 않습니다. 예를 들어 일반적으로 개체의 특정 구현이 나 응용 프로그램의 논리에 관계 없이 다음과 같은 표준 작업을 수행 합니다.

- 데이터를 개체로 로드 (일반적으로 데이터베이스에서)
- 개체의 형식화 된 컬렉션을 만듭니다.
- 컬렉션에서 개체를 추가 하거나 제거 합니다.
- 폼의 사용자에 게 개체 데이터 표시
- 개체의 데이터 변경/편집
- 개체의 데이터를 데이터베이스에 다시 저장 합니다.

개체에 데이터 로드

이 예에서는 Tableadapter를 사용 하 여 개체에 데이터를 로드 합니다. 기본적으로 Tableadapter는 데이터베이스에서 데이터를 가져오고 데이터 테이블을 채우는 두 가지 종류의 메서드를 사용 하 여 생성 됩니다.

- `TableAdapter.Fill` 메서드는 기존 데이터 테이블을 반환 된 데이터로 채웁니다.
- `TableAdapter.GetData` 메서드는 데이터로 채워진 새 데이터 테이블을 반환 합니다.

데이터를 사용 하 여 사용자 지정 개체를 로드 하는 가장 쉬운 방법은 `TableAdapter.GetData` 메서드를 호출 하 고, 반환 된 데이터 테이블의 행 컬렉션을 반복 하 고 각 개체를 각 행의 값으로 채우는 것입니다. TableAdapter에 추가 된 쿼리에 대해 채워진 데이터 테이블을 반환 하는 `GetData` 메서드를 만들 수 있습니다.

> **NOTE**
> Visual Studio는 기본적으로 `Fill` 하 고 `GetData` TableAdapter 쿼리의 이름을 지정할 수 있지만 이러한 이름을 유효한 메서드 이름으로 변경할 수 있습니다.

다음 예에서는 데이터 테이블의 행을 반복 하 고 데이터를 사용 하 여 개체를 채우는 방법을 보여 줍니다.

```
private void LoadCustomers()
{
    NorthwindDataSet.CustomersDataTable customerData =
        customersTableAdapter1.GetTop5Customers();

    foreach (NorthwindDataSet.CustomersRow customerRow in customerData)
    {
        Customer currentCustomer = new Customer();
        currentCustomer.CustomerID = customerRow.CustomerID;
        currentCustomer.CompanyName = customerRow.CompanyName;

        if (customerRow.IsAddressNull() == false)
        {
            currentCustomer.Address = customerRow.Address;
        }

        if (customerRow.IsCityNull() == false)
        {
            currentCustomer.City = customerRow.City;
        }

        if (customerRow.IsContactNameNull() == false)
        {
            currentCustomer.ContactName = customerRow.ContactName;
        }

        if (customerRow.IsContactTitleNull() == false)
        {
            currentCustomer.ContactTitle = customerRow.ContactTitle;
        }

        if (customerRow.IsCountryNull() == false)
        {
            currentCustomer.Country = customerRow.Country;
        }

        if (customerRow.IsFaxNull() == false)
        {
            currentCustomer.Fax = customerRow.Fax;
        }

        if (customerRow.IsPhoneNull() == false)
        {
            currentCustomer.Phone = customerRow.Phone;
        }

        if (customerRow.IsPostalCodeNull() == false)
        {
            currentCustomer.PostalCode = customerRow.PostalCode;
        }

        if (customerRow.IsRegionNull() == false)
        {
            currentCustomer.Region = customerRow.Region;
        }

        LoadOrders(currentCustomer);
        customerBindingSource.Add(currentCustomer);
    }
}
```

```
Private Sub LoadCustomers()
    Dim customerData As NorthwindDataSet.CustomersDataTable =
        CustomersTableAdapter1.GetTop5Customers()

    Dim customerRow As NorthwindDataSet.CustomersRow

    For Each customerRow In customerData
        Dim currentCustomer As New Customer()
        With currentCustomer

            .CustomerID = customerRow.CustomerID
            .CompanyName = customerRow.CompanyName

            If Not customerRow.IsAddressNull Then
                .Address = customerRow.Address
            End If

            If Not customerRow.IsCityNull Then
                .City = customerRow.City
            End If

            If Not customerRow.IsContactNameNull Then
                .ContactName = customerRow.ContactName
            End If

            If Not customerRow.IsContactTitleNull Then
                .ContactTitle = customerRow.ContactTitle
            End If

            If Not customerRow.IsCountryNull Then
                .Country = customerRow.Country
            End If

            If Not customerRow.IsFaxNull Then
                .Fax = customerRow.Fax
            End If

            If Not customerRow.IsPhoneNull Then
                .Phone = customerRow.Phone
            End If

            If Not customerRow.IsPostalCodeNull Then
                .PostalCode = customerRow.PostalCode
            End If

            If Not customerRow.Is_RegionNull Then
                .Region = customerRow._Region
            End If

        End With

        LoadOrders(currentCustomer)
        CustomerBindingSource.Add(currentCustomer)
    Next
End Sub
```

개체의 형식화 된 컬렉션 만들기

개체에 대 한 컬렉션 클래스를 만들거나 BindingSource 구성 요소에서 자동으로 제공 하는 형식화 된 컬렉션을 사용할 수 있습니다.

개체에 대 한 사용자 지정 컬렉션 클래스를 만들 때 BindingList<T>에서 상속 하는 것이 좋습니다. 이 제네릭 클래스는 컬렉션을 관리 하는 기능 뿐만 아니라 Windows Forms의 데이터 바인딩 인프라에 알림을 보내는 이벤트를 발생 시키는 기능을 제공 합니다.

BindingSource에서 자동으로 생성 된 컬렉션은 형식화 된 컬렉션에 대 한 BindingList<T>를 사용 합니다. 응용 프

로그램에 추가 기능이 필요 하지 않은 경우 BindingSource내에서 컬렉션을 유지할 수 있습니다. 자세한 내용은 참조 하세요. 합니다 List 의 속성을 BindingSource 클래스.

> **NOTE**
> 컬렉션에 BindingList<T>의 기본 구현에서 제공 하지 않는 기능이 필요한 경우 필요에 따라 클래스에 추가할 수 있도록 사용자 지정 컬렉션을 만들어야 합니다.

다음 코드에서는 강력한 형식의 `Order` 개체 컬렉션에 대 한 클래스를 만드는 방법을 보여 줍니다.

```
/// <summary>
/// A collection of Orders
/// </summary>
public class Orders: System.ComponentModel.BindingList<Order>
{
    // Add any additional functionality required by your collection.
}
```

```
''' <summary>
''' A collection of Orders
''' </summary>
Public Class Orders
    Inherits System.ComponentModel.BindingList(Of Order)

    ' Add any additional functionality required by your collection.

End Class
```

컬렉션에 개체 추가

사용자 지정 컬렉션 클래스 또는 BindingSource의 `Add` 메서드를 호출 하 여 컬렉션에 개체를 추가 합니다.

> **NOTE**
> `Add` 메서드는 BindingList<T>에서 상속할 때 사용자 지정 컬렉션에 대해 자동으로 제공 됩니다.

다음 코드에서는 BindingSource의 형식화 된 컬렉션에 개체를 추가 하는 방법을 보여 줍니다.

```
Customer currentCustomer = new Customer();
customerBindingSource.Add(currentCustomer);
```

```
Dim currentCustomer As New Customer()
CustomerBindingSource.Add(currentCustomer)
```

다음 코드는 BindingList<T>에서 상속 되는 형식화 된 컬렉션에 개체를 추가 하는 방법을 보여 줍니다.

> **NOTE**
> 이 예제에서 `Orders` 컬렉션은 `Customer` 개체의 속성입니다.

```
Order currentOrder = new Order();
currentCustomer.Orders.Add(currentOrder);
```

```
Dim currentOrder As New Order()
currentCustomer.Orders.Add(currentOrder)
```

컬렉션에서 개체 제거

사용자 지정 컬렉션 클래스 또는 BindingSource의 `Remove` 또는 `RemoveAt` 메서드를 호출 하 여 컬렉션에서 개체를 제거 합니다.

> **NOTE**
> `Remove` 및 `RemoveAt` 메서드는 BindingList<T>에서 상속할 때 사용자 지정 컬렉션에 대해 자동으로 제공 됩니다.

다음 코드에서는 RemoveAt 메서드를 사용 하 여 BindingSource의 형식화 된 컬렉션에서 개체를 찾고 제거 하는 방법을 보여 줍니다.

```
int customerIndex = customerBindingSource.Find("CustomerID", "ALFKI");
customerBindingSource.RemoveAt(customerIndex);
```

```
Dim customerIndex As Integer = CustomerBindingSource.Find("CustomerID", "ALFKI")
CustomerBindingSource.RemoveAt(customerIndex)
```

사용자에 게 개체 데이터 표시

사용자에 게 개체의 데이터를 표시 하려면 **데이터 소스 구성** 마법사를 사용 하 여 개체 데이터 소스를 만든 다음 **데이터** 소스 창에서 전체 개체 또는 개별 속성을 폼으로 끌어 옵니다.

개체의 데이터 수정

Windows Forms 컨트롤에 데이터 바인딩된 사용자 지정 개체의 데이터를 편집 하려면 바인딩된 컨트롤의 데이터를 편집 하거나 개체의 속성에서 직접 편집 하면 됩니다. 데이터 바인딩 아키텍처는 개체의 데이터를 업데이트 합니다.

응용 프로그램에서 변경 내용을 추적 해야 하 고 제안 된 변경 내용을 원래 값으로 롤백하는 경우 개체 모델에서이 기능을 구현 해야 합니다. 데이터 테이블에서 제안 된 변경 내용을 추적 하는 방법에 대 한 예는 DataRowState, HasChanges및 GetChanges를 참조 하세요.

개체의 데이터를 데이터베이스에 다시 저장 합니다.

개체의 값을 TableAdapter의 DBDirect 메서드로 전달 하 여 데이터를 다시 데이터베이스에 저장 합니다.

Visual Studio는 데이터베이스에 대해 직접 실행할 수 있는 DBDirect 메서드를 만듭니다. 이러한 메서드에는 DataSet 또는 DataTable 개체가 필요 하지 않습니다.

TABLEADAPTER DBDIRECT 메서드	설명
`TableAdapter.Insert`	데이터베이스에 새 레코드를 추가 하 여 개별 열 값을 메서드 매개 변수로 전달할 수 있도록 합니다.
`TableAdapter.Update`	데이터베이스의 기존 레코드를 업데이트 합니다. Update 메서드는 원래 열 값과 새 열 값을 메서드 매개 변수로 사용 합니다. 원래 값을 사용 하 여 원래 레코드를 찾은 다음 새 값을 사용 하 여 해당 레코드를 업데이트 합니다. `TableAdapter.Update` 메서드는 DataSet, DataTable, DataRow또는 DataRows의 배열을 메서드 매개 변수로 사용 하 여 데이터 집합의 변경 내용을 데이터베이스에 다시 조정 하는 데도 사용 됩니다.

TableAdapter DBDirect 메서드	설명
`TableAdapter.Delete`	메서드 매개 변수로 전달 된 원래 열 값을 기준으로 데이터베이스에서 기존 레코드를 삭제 합니다.

개체의 컬렉션에서 데이터를 저장 하려면 개체의 컬렉션을 반복 합니다. 예를 들어 for next 루프를 사용 합니다. TableAdapter의 DBDirect 메서드를 사용 하 여 각 개체의 값을 데이터베이스로 보냅니다.

다음 예에서는 `TableAdapter.Insert` DBDirect 메서드를 사용 하 여 새 고객을 데이터베이스에 직접 추가 하는 방법을 보여 줍니다.

```
private void AddNewCustomers(Customer currentCustomer)
{
    customersTableAdapter.Insert(
        currentCustomer.CustomerID,
        currentCustomer.CompanyName,
        currentCustomer.ContactName,
        currentCustomer.ContactTitle,
        currentCustomer.Address,
        currentCustomer.City,
        currentCustomer.Region,
        currentCustomer.PostalCode,
        currentCustomer.Country,
        currentCustomer.Phone,
        currentCustomer.Fax);
}
```

```
Private Sub AddNewCustomer(ByVal currentCustomer As Customer)

    CustomersTableAdapter.Insert(
        currentCustomer.CustomerID,
        currentCustomer.CompanyName,
        currentCustomer.ContactName,
        currentCustomer.ContactTitle,
        currentCustomer.Address,
        currentCustomer.City,
        currentCustomer.Region,
        currentCustomer.PostalCode,
        currentCustomer.Country,
        currentCustomer.Phone,
        currentCustomer.Fax)
End Sub
```

참조

- Visual Studio에서 데이터에 컨트롤 바인딩

Visual Studio에서 데이터 바인딩된 컨트롤에 대한 캡션을 만드는 방식 사용자 지정

2020-01-16 • 10 minutes to read • Edit Online

데이터 소스 창 에서 디자이너로 항목을 끌어 오면 특별 한 고려 사항이 있습니다. 두 개 이상의 단어가 함께 연결 될 경우 캡션 레이블의 열 이름이 더 읽기 쉬운 문자열로 다시 포맷 됩니다.

HKEY_CURRENT_USER \Software\microsoft\visualstudio\15.0\data designer 레지스트리 키에서 **Smartcaptionexpression**, **Smartcaptionexpression**및 **smartcaptionexpression** 값을 설정 하 여 이러한 레이블이 생성 되는 방식을 사용자 지정할 수 있습니다.

HKEY_CURRENT_USER \Software\microsoft\visualstudio\16.0\data designer 레지스트리 키에서 **Smartcaptionexpression**, **Smartcaptionexpression**및 **smartcaptionexpression** 값을 설정 하 여 이러한 레이블이 생성 되는 방식을 사용자 지정할 수 있습니다.

> **NOTE**
> 이 레지스트리 키는 만들 때까지 존재 하지 않습니다.

스마트 캡션은 **Smartcaptionexpression** 값의 값에 입력 된 정규식에 의해 제어 됩니다. **데이터 디자이너** 레지스트리 키를 추가 하면 캡션 레이블을 제어 하는 기본 정규식이 재정의 됩니다. 정규식에 대 한 자세한 내용은 Visual Studio에서 정규식 사용을 참조 하세요.

다음 표에서는 캡션 레이블을 제어 하는 레지스트리 값에 대해 설명 합니다.

레지스트리 항목	설명
SmartCaptionExpression	패턴을 일치 시키기 위해 사용 하는 정규식입니다.
SmartCaptionReplacement	**Smartcaptionexpression**에 일치 하는 그룹을 표시 하는 형식입니다.
SmartCaptionSuffix	캡션의 끝에 추가할 선택적 문자열입니다.

다음 표에서는 이러한 레지스트리 값에 대 한 내부 기본 설정을 나열 합니다.

레지스트리 항목	기본값	설명
SmartCaptionExpression	(\\p{Ll})(\\p{Lu})|_+	소문자와 대문자 또는 밑줄을 찾습니다.
SmartCaptionReplacement	$1 $2	**$1** 은 식의 첫 번째 괄호에서 일치 하는 문자를 나타내며, **$2** 은 두 번째 괄호에 일치 하는 모든 문자를 나타냅니다. 첫 번째 일치 항목, 공백, 두 번째 일치 항목을 대체 합니다.
SmartCaptionSuffix	:	반환 된 문자열에 추가 되는 문자를 나타냅니다. 예를 들어 캡션이 `Company Name` 되는 경우 접미사를 사용 하 여 `Company Name:`

Caution

레지스트리 편집기에서 원하는 작업을 수행 하는 경우에는 주의 해야 합니다. 레지스트리를 편집 하기 전에 백업 합니다. 레지스트리 편집기를 잘못 사용 하면 운영 체제를 다시 설치 해야 할 수 있는 심각한 문제가 발생할 수 있습니다. Microsoft는 레지스트리 편집기를 잘못 사용 하 여 발생 하는 문제를 해결할 수 있도록 보장 하지 않습니다. 레지스트리 편집기 사용에 따른 결과는 사용자의 책임입니다.

레지스트리 백업, 편집 및 복원에 대 한 자세한 내용은 advanced users에 대 한 Windows 레지스트리 정보를 참조 하십시오.

데이터 소스 창의 스마트 캡션 동작 수정

1. **시작** 을 클릭 한 다음 **실행**을 클릭 하 여 명령 창을 엽니다.
2. **실행** 대화 상자에 `regedit`을 입력 하 고 **확인**을 클릭 합니다.
3. **HKEY_CURRENT_USER** > **Software** > **Microsoft** > **VisualStudio** 노드를 확장 합니다.
4. **15.0** 노드를 마우스 오른쪽 단추로 클릭 하 고 `Data Designers`이라는 새 **키** 를 만듭니다.
4. **16.0** 노드를 마우스 오른쪽 단추로 클릭 하 고 `Data Designers`이라는 새 **키** 를 만듭니다.
5. **데이터 디자이너** 노드를 마우스 오른쪽 단추로 클릭 하 고 다음과 같은 세 개의 새 문자열 값을 만듭니다.
   - `SmartCaptionExpression`
   - `SmartCaptionReplacement`
   - `SmartCaptionSuffix`
6. **Smartcaptionexpression** 값을 마우스 오른쪽 단추로 클릭 하 고 **수정**을 선택 합니다.
7. **데이터 소스** 창에서 사용 하려는 정규식을 입력 합니다.
8. **Smartcaptionreplacement** 값을 마우스 오른쪽 단추로 클릭 하 고 **수정**을 선택 합니다.
9. 정규식에서 일치 하는 패턴을 표시 하려는 방식으로 형식이 지정 된 대체 문자열을 입력 합니다.
10. **Smartcaptionsuffix** 값을 마우스 오른쪽 단추로 클릭 하 고 **수정**을 선택 합니다.
11. 캡션의 끝에 표시할 문자를 입력 합니다.

    다음 번에 **데이터 소스** 창에서 항목을 끌면 제공 된 새 레지스트리 값을 사용 하 여 캡션 레이블이 생성 됩니다.

스마트 캡션 기능 해제

1. **시작** 을 클릭 한 다음 **실행**을 클릭 하 여 명령 창을 엽니다.
2. **실행** 대화 상자에 `regedit`을 입력 하 고 **확인**을 클릭 합니다.
3. **HKEY_CURRENT_USER** > **Software** > **Microsoft** > **VisualStudio** 노드를 확장 합니다.
4. **15.0** 노드를 마우스 오른쪽 단추로 클릭 하 고 `Data Designers`이라는 새 **키** 를 만듭니다.
4. **16.0** 노드를 마우스 오른쪽 단추로 클릭 하 고 `Data Designers`이라는 새 **키** 를 만듭니다.
5. **데이터 디자이너** 노드를 마우스 오른쪽 단추로 클릭 하 고 다음과 같은 세 개의 새 문자열 값을 만듭니다.
   - `SmartCaptionExpression`
   - `SmartCaptionReplacement`
   - `SmartCaptionSuffix`
6. **Smartcaptionexpression** 항목을 마우스 오른쪽 단추로 클릭 하 고 **수정**을 선택 합니다.

7. 값에 `(.*)`을 입력 합니다. 그러면 전체 문자열이 일치 합니다.

8. **Smartcaptionreplacement** 항목을 마우스 오른쪽 단추로 클릭 하 고 **수정**을 선택 합니다.

9. 값에 `$1`을 입력 합니다. 이렇게 하면 문자열을 일치 하는 값으로 대체 하 여 변경 되지 않은 상태로 유지 되도록 전체 문자열을 대체 합니다.

   다음 번에 **데이터 소스** 창에서 항목을 끌면 수정 되지 않은 캡션으로 캡션 레이블이 생성 됩니다.

참조

- Visual Studio에서 데이터에 컨트롤 바인딩

Windows Communication Foundation 서비스 및 Visual Studio의 WCF.NET 데이터 서비스

2020-01-06 • 37 minutes to read • Edit Online

Visual Studio는 배포 응용 프로그램을 만들기 위한 Microsoft 기술 Windows Communication Foundation (WCF) 및 WCF Data Services을 사용 하기 위한 도구를 제공 합니다. 이 항목에서는 Visual Studio 관점의 서비스를 소개 합니다. 전체 설명서는 WCF Data Services 4.5을 참조 하세요.

WCF 란?

WCF (Windows Communication Foundation)는 안전 하 고 신뢰할 수 있으며 트랜잭션 되 고 상호 운용할 수 있는 분산 응용 프로그램을 만들기 위한 통합 프레임 워크입니다. ASMX 웹 서비스, .NET Remoting, 엔터프라이즈 서비스 (DCOM) 및 MSMQ와 같은 이전 프로세스 간 통신 기술을 대체 합니다. WCF는 통합 프로그래밍 모델에서 모든 기술의 기능을 함께 제공 합니다. 이렇게 하면 배포 응용 프로그램을 개발 하는 환경이 간소화 됩니다.

WCF Data Services 정의

WCF Data Services는 OData (Open Data) 프로토콜 표준의 구현입니다. WCF Data Services를 사용 하면 테이블 형식 데이터를 일련의 REST Api로 노출 하 여 GET, POST, PUT 또는 DELETE와 같은 표준 HTTP 동사를 사용 하 여 데이터를 반환할 수 있습니다. 서버 쪽에서는 새 OData 서비스를 만들기 위해 WCF Data Services ASP.NET Web API 으로 대체 됩니다. WCF Data Services 클라이언트 라이브러리는 Visual Studio (**Project** > **서비스 참조 추가**)의 .Net 응용 프로그램에서 OData 서비스를 사용 하는 데 적합 합니다. 자세한 내용은 WCF Data Services 4.5를 참조하세요.

WCF 프로그래밍 모델

WCF 프로그래밍 모델은 WCF 서비스와 WCF 클라이언트 라는 두 엔터티 간의 통신을 기반으로 합니다. 프로그래밍 모델은 .NET의 System.ServiceModel 네임 스페이스에 캡슐화 되어 있습니다.

WCF 서비스

WCF 서비스는 서비스와 클라이언트 간의 계약을 정의 하는 인터페이스를 기반으로 합니다. 다음 코드와 같이 ServiceContractAttribute 특성으로 표시 됩니다.

```
[ServiceContract]
public interface IService1
```

```
<ServiceContract()>
Public Interface IService1
```

WCF 서비스에 의해 노출 되는 함수 또는 메서드를 OperationContractAttribute 특성으로 표시 하 여 정의 합니다.

```
[OperationContract]
string GetData(string value);
```

```
<OperationContract()>
Function GetData(ByVal value As String) As String
```

또한 복합 형식을 DataContractAttribute 특성으로 표시 하 여 serialize 된 데이터를 노출할 수 있습니다. 이렇게 하면 클라이언트에서 데이터를 바인딩할 수 있습니다.

인터페이스와 해당 메서드를 정의한 후에는 인터페이스를 구현 하는 클래스에 캡슐화 됩니다. 단일 WCF 서비스 클래스는 여러 서비스 계약을 구현할 수 있습니다.

WCF 서비스는 *끝점*이라고 하는 항목을 통해 사용 하기 위해 노출 됩니다. 끝점은 서비스와 통신 하는 유일한 방법을 제공 합니다. 다른 클래스와 마찬가지로 직접 참조를 통해 서비스에 액세스할 수 없습니다.

끝점은 주소, 바인딩 및 계약으로 구성 됩니다. 주소는 서비스의 위치를 정의 합니다. URL, FTP 주소 또는 네트워크 또는 로컬 경로일 수 있습니다. 바인딩은 서비스와 통신 하는 방법을 정의 합니다. WCF 바인딩은 HTTP 또는 FTP와 같은 프로토콜, Windows 인증 또는 사용자 이름 및 암호 등의 보안 메커니즘을 지정 하는 다양 한 모델을 제공 합니다. 계약은 WCF 서비스 클래스에 의해 노출 되는 작업을 포함 합니다.

단일 WCF 서비스에 대해 여러 끝점을 노출할 수 있습니다. 이렇게 하면 여러 클라이언트가 서로 다른 방식으로 동일한 서비스와 통신할 수 있습니다. 예를 들어, 은행 서비스는 직원을 위한 끝점을 제공 하 고, 다른 주소, 바인딩 및/또는 계약을 사용 하는 외부 고객에 게 다른 끝점을 제공할 수 있습니다.

WCF 클라이언트**(WCF client)**

WCF 클라이언트는 응용 프로그램이 WCF 서비스와 통신할 수 있도록 하는 *프록시*에 대해 정의 된 끝점과 일치 하는 끝점으로 구성 됩니다. 프록시는 *app.config* 파일의 클라이언트 쪽에서 생성 되 고 서비스에서 노출 하는 형식 및 메서드에 대 한 정보를 포함 합니다. 여러 끝점을 노출 하는 서비스의 경우 클라이언트는 HTTP를 통해 통신 하 고 Windows 인증을 사용 하는 등의 요구에 가장 적합 한 항목을 선택할 수 있습니다.

WCF 클라이언트를 만든 후에는 다른 개체와 마찬가지로 코드에서 서비스를 참조 합니다. 예를 들어 앞에 표시 된 `GetData` 메서드를 호출 하려면 다음과 같은 코드를 작성 합니다.

```
private void button1_Click(System.Object sender, System.EventArgs e)
{
    ServiceReference1.Service1Client client = new
        ServiceReference1.Service1Client();
    string returnString;

    returnString = client.GetData(textBox1.Text);
    label1.Text = returnString;
}
```

```
  Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
      Dim client As New ServiceReference1.Service1Client
      Dim returnString As String

      returnString = client.GetData(TextBox1.Text)
      Label1.Text = returnString
  End Sub
```

Visual Studio의 WCF 도구

Visual Studio는 WCF 서비스와 WCF 클라이언트를 만드는 데 도움이 되는 도구를 제공 합니다. 도구를 보여 주는 연습은 연습: Windows Forms에서 간단한 WCF 서비스 만들기를 참조 하세요.

WCF 서비스 만들기 및 테스트

WCF Visual Studio 템플릿을 기반으로 사용 하 여 서비스를 신속 하 게 만들 수 있습니다. 그런 다음 WCF 서비스 자동 호스트 및 WCF 테스트 클라이언트를 사용 하 여 서비스를 디버깅 하 고 테스트할 수 있습니다. 이러한 도구는 빠르고 편리한 디버그 및 테스트 주기를 제공 하 고 초기 단계에서 호스팅 모델에 커밋하지 않아도 되는 요구 사항을 제거 합니다.

WCF 템플릿

WCF Visual Studio 템플릿은 서비스 개발을 위한 기본 클래스 구조를 제공 합니다. **새 프로젝트 추가** 대화 상자에서 사용할 수 있는 몇 가지 WCF 템플릿이 있습니다. 여기에는 WCF 서비스 ILibrary 프로젝트, WCF 서비스 웹

사이트 및 WCF 서비스 항목 템플릿이 포함 됩니다.

템플릿을 선택 하면 서비스 계약, 서비스 구현 및 서비스 구성에 대 한 파일이 추가 됩니다. 필요한 모든 특성이 이미 추가 되었으며 간단한 "Hello World" 유형의 서비스를 만들고 코드를 작성할 필요가 없습니다. 물론 실제 서비스에 대 한 함수 및 메서드를 제공 하는 코드를 추가 하려고 하지만 템플릿에서 기본 토대를 제공 합니다.

WCF 템플릿에 대 한 자세한 내용은 Wcf Visual Studio 템플릿을 참조 하세요.

WCF 서비스 호스트

Visual Studio 디버거를 시작할 때 (**F5**키를 눌러) wcf 서비스 프로젝트에 대해 Wcf 서비스 호스트 도구가 자동으로 시작 되어 서비스를 로컬로 호스팅합니다. WCF 서비스 호스트는 WCF 서비스 프로젝트에서 서비스를 열거 하 고 프로젝트의 구성을 로드 한 다음 찾은 각 서비스에 대 한 호스트를 인스턴스화합니다.

WCF 서비스 호스트를 사용 하 여 개발 중에 추가 코드를 작성 하거나 특정 호스트를 커밋하지 않고도 WCF 서비스를 테스트할 수 있습니다.

WCF 서비스 호스트에 대 한 자세한 내용은 wcf 서비스 호스트 (wcfsvchost.exe)를 참조 하세요.

WCF 테스트 클라이언트

WCF 테스트 클라이언트 도구를 사용 하 여 테스트 매개 변수를 입력 하 고, 해당 입력을 WCF 서비스에 제출 하 고, 서비스가 다시 보내는 응답을 볼 수 있습니다. WCF 서비스 호스트와 결합할 때 편리한 서비스 테스트 환경을 제공 합니다. *% ProgramFiles (x86)% \ Microsoft Visual Studio\2017\Enterprise\Common7\IDE* 폴더에서 도구를 찾습니다.

F5 키를 눌러 wcf 서비스 프로젝트를 디버깅할 때 Wcf 테스트 클라이언트는 구성 파일에 정의 된 서비스 끝점 목록을 열고 표시 합니다. 매개 변수를 테스트 하 고 서비스를 시작 하 고이 프로세스를 반복 하 여 서비스를 지속적으로 테스트 하 고 유효성을 검사할 수 있습니다.

WCF 테스트 클라이언트에 대해 자세히 알아보려면 wcf 테스트 클라이언트 (wcftestclient.exe)를 참조 하세요.

Visual Studio에서 **WCF** 서비스 액세스

Visual Studio는 WCF 클라이언트 만들기, **서비스 참조 추가** 대화 상자를 사용 하 여 추가 하는 서비스에 대 한 프록시 및 끝점을 자동으로 생성 하는 작업을 간소화 합니다. 필요한 모든 구성 정보는 *app.config* 파일에 추가 됩니다. 대부분의 경우에는이 서비스를 사용 하기 위해 서비스를 인스턴스화해야 합니다.

서비스 참조 추가 대화 상자를 사용 하 여 서비스에 대 한 주소를 입력 하거나 솔루션에 정의 된 서비스를 검색할 수 있습니다. 이 대화 상자는 서비스 목록 및 해당 서비스에서 제공 하는 작업을 반환 합니다. 또한 코드에서 서비스를 참조 하는 네임 스페이스를 정의할 수 있습니다.

서비스 참조 구성 대화 상자를 사용 하 여 서비스에 대 한 구성을 사용자 지정할 수 있습니다. 서비스의 주소를 변경 하 고, 액세스 수준, 비동기 동작 및 메시지 계약 형식을 지정 하 고, 형식 재사용을 구성할 수 있습니다.

방법: 서비스 끝점 선택

WCF (일부 Windows Communication Foundation) 서비스는 클라이언트가 서비스와 통신할 수 있는 여러 끝점을 노출 합니다. 예를 들어 서비스는 HTTP 바인딩과 사용자 이름 및 암호 보안을 사용 하는 하나의 끝점과 FTP 및 Windows 인증을 사용 하는 두 번째 끝점을 노출할 수 있습니다. 첫 번째 끝점은 방화벽 외부에서 서비스에 액세스 하는 응용 프로그램에서 사용 될 수 있지만, 두 번째 끝점은 인트라넷에서 사용 될 수 있습니다.

이러한 경우 서비스 참조의 생성자에 대 한 매개 변수로 `endpointConfigurationName` 를 지정할 수 있습니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

서비스 끝점을 선택 하려면

1. **솔루션 탐색기** 에서 프로젝트 노드를 마우스 오른쪽 단추로 클릭 하 고 **서비스 참조 추가**를 선택 하 여 WCF 서비스에 대 한 참조를 추가 합니다.

2. 코드 편집기에서 서비스 참조에 대 한 생성자를 추가 합니다.

```
Dim proxy As New ServiceReference.Service1Client(
```

```
ServiceReference.Service1Client proxy = new ServiceReference.Service1Client(
```

> **NOTE**
> *ServiceReference* 을 서비스 참조의 네임 스페이스로 바꾸고 *Service1Client* 을 서비스 이름으로 바꿉니다.

3. 생성자에 대 한 오버 로드가 포함 된 IntelliSense 목록이 표시 됩니다. `endpointConfigurationName As String` 오버 로드를 선택 합니다.

4. 오버 로드 뒤에 `=` *configurationname*을 입력 합니다. 여기서 *configurationname* 은 사용 하려는 끝점의 이름입니다.

> **NOTE**
> 사용 가능한 끝점의 이름을 모르는 경우 *app.config* 파일에서 찾을 수 있습니다.

WCF 서비스에 사용할 수 있는 끝점을 찾으려면

1. **솔루션 탐색기**에서 서비스 참조가 포함 된 프로젝트의 **app.config 파일을** 마우스 오른쪽 단추로 클릭 한 다음 **열기**를 클릭 합니다. 파일이 코드 편집기에 표시 됩니다.

2. 파일에서 `<Client>` 태그를 검색 합니다.

3. `<Client>` 태그 아래에서 `<Endpoint>`로 시작 하는 태그를 검색 합니다.

   서비스 참조에서 여러 끝점을 제공 하는 경우 두 개 이상의 `<Endpoint` 태그가 있습니다.

4. `<EndPoint>` 태그 안에 `name="`*SomeService*`"` 매개 변수 (여기서 *SomeService* 은 끝점 이름을 나타냄)를 찾을 수 있습니다. 서비스 참조에 대 한 생성자의 `endpointConfigurationName As String` 오버 로드에 전달 될 수 있는 끝점의 이름입니다.

방법: 비동기적으로 서비스 메서드 호출

WCF (Windows Communication Foundation) 서비스의 대부분의 메서드는 동기적 또는 비동기적으로 호출할 수 있습니다. 메서드를 비동기적으로 호출 하면 메서드가 저속 연결을 통해 작동할 때 호출 되는 동안 응용 프로그램이 계속 작동할 수 있습니다.

기본적으로 프로젝트에 서비스 참조를 추가 하는 경우 메서드를 동기적으로 호출 하도록 구성 됩니다. **서비스 참조 구성** 대화 상자에서 설정을 변경 하 여 메서드를 비동기적으로 호출 하는 동작을 변경할 수 있습니다.

> **NOTE**
> 이 옵션은 서비스 별로 설정 됩니다. 서비스에 대 한 한 메서드가 비동기적으로 호출 되는 경우 모든 메서드를 비동기적으로 호출 해야 합니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

서비스 메서드를 비동기적으로 호출 하려면

1. **솔루션 탐색기**에서 서비스 참조를 선택 합니다.
2. **프로젝트** 메뉴에서 **서비스 참조 구성**을 클릭 합니다.
3. **서비스 참조 구성** 대화 상자에서 **비동기 작업 생성** 확인란을 선택 합니다.

방법: 서비스에서 반환 되는 데이터 바인딩

다른 데이터 소스를 컨트롤에 바인딩할 수 있는 것 처럼 WCF (Windows Communication Foundation) 서비스에서 반환 되는 데이터를 컨트롤에 바인딩할 수 있습니다. WCF 서비스에 대 한 참조를 추가 하는 경우 서비스에 데이터를 반환 하는 복합 형식이 포함 되어 있으면 **데이터 소스** 창에 자동으로 추가 됩니다.

WCF 서비스에서 반환 하는 단일 데이터 필드에 컨트롤을 바인딩하려면

1. **데이터** 메뉴에서 **데이터 소스 표시**를 클릭합니다.

   데이터 소스 창이 나타납니다.
2. **데이터 소스** 창에서 서비스 참조에 대 한 노드를 확장 합니다. 서비스 표시에서 반환 된 모든 복합 형식입니다.
3. 형식에 대 한 노드를 확장 합니다. 해당 형식에 대 한 데이터 필드가 나타납니다.
4. 필드를 선택 하 고 드롭다운 화살표를 클릭 하 여 데이터 형식에 사용할 수 있는 컨트롤의 목록을 표시 합니다.
5. 바인딩하려는 컨트롤의 형식을 클릭 합니다.
6. 필드를 폼으로 끌어 놓습니다. 컨트롤은 BindingSource 구성 요소 및 BindingNavigator 구성 요소와 함께 폼에 추가 됩니다.
7. 바인딩하려는 다른 모든 필드에 대해 4 ~ 6 단계를 반복 합니다.

WCF 서비스에서 반환 하는 복합 형식에 컨트롤을 바인딩하려면

1. **데이터** 메뉴에서 **데이터 소스 표시**를 선택 합니다. **데이터 소스** 창이 나타납니다.
2. **데이터 소스** 창에서 서비스 참조에 대 한 노드를 확장 합니다. 서비스 표시에서 반환 된 모든 복합 형식입니다.
3. 형식에 대 한 노드를 선택 하 고 드롭다운 화살표를 클릭 하 여 사용 가능한 옵션 목록을 표시 합니다.
4. 표 형태의 데이터를 표시 하려면 **DataGridView** 를 클릭 하거나 개별 컨트롤에 데이터를 표시 하려면 **세부 정보** 를 클릭 합니다.
5. 노드를 폼으로 끌어 옵니다. 컨트롤은 BindingSource 구성 요소 및 BindingNavigator 구성 요소와 함께 폼에 추가 됩니다.

방법: 서비스를 구성 하 여 기존 형식 다시 사용

서비스 참조가 프로젝트에 추가 되 면 서비스에 정의 된 모든 형식이 로컬 프로젝트에 생성 됩니다. 대부분의 경우 서비스에서 공용 .NET 형식을 사용 하거나 형식이 공유 라이브러리에 정의 된 경우 중복 형식을 만듭니다.

이 문제를 방지 하기 위해 참조 된 어셈블리의 형식이 기본적으로 공유 됩니다. 하나 이상의 어셈블리에 대해 형

식 공유를 사용 하지 않도록 설정 하려면 **서비스 참조 구성** 대화 상자에서이 작업을 수행할 수 있습니다.

단일 어셈블리에서 형식 공유를 사용 하지 않도록 설정 하려면

1. **솔루션 탐색기**에서 서비스 참조를 선택 합니다.
2. **프로젝트** 메뉴에서 **서비스 참조 구성**을 클릭 합니다.
3. **서비스 참조 구성** 대화 상자에서 **지정 된 참조 된 어셈블리의 형식 재사용**을 선택 합니다.
4. 형식 공유를 사용 하도록 설정할 각 어셈블리에 대 한 확인란을 선택 합니다. 어셈블리에 대해 형식 공유를 사용 하지 않도록 설정 하려면 확인란을 선택 하지 않은 상태로 둡니다.

모든 어셈블리에서 형식 공유를 사용 하지 않도록 설정 하려면

1. **솔루션 탐색기**에서 서비스 참조를 선택 합니다.
2. **프로젝트** 메뉴에서 **서비스 참조 구성**을 클릭 합니다.
3. **서비스 참조 구성** 대화 상자에서 **참조 된 어셈블리의 형식 재사용** 확인란의 선택을 취소 합니다.

관련 항목

제목	설명
연습: Windows Forms에서 간단한 WCF 서비스 만들기	Visual Studio에서 WCF 서비스를 만들고 사용 하는 방법에 대 한 단계별 데모를 제공 합니다.
연습: WPF 및 Entity Framework를 사용하여 WCF 데이터 서비스 만들기	Visual Studio에서 WCF Data Services를 만들고 사용 하는 방법에 대 한 단계별 데모를 제공 합니다.
WCF 개발 도구 사용	Visual Studio에서 WCF 서비스를 만들고 테스트 하는 방법을 설명 합니다.
	방법: WCF 데이터 서비스 참조 추가, 업데이트 또는 제거
서비스 참조 문제 해결	서비스 참조에서 발생할 수 있는 몇 가지 일반적인 오류와 이러한 오류를 방지 하는 방법을 보여 줍니다.
WCF 서비스 디버그	WCF 서비스를 디버그할 때 발생할 수 있는 일반적인 디버깅 문제와 기술에 대해 설명 합니다.
연습: N 계층 데이터 애플리케이션 만들기	형식화된 데이터 세트을 만들고 TableAdapter 및 데이터 세트 코드를 여러 프로젝트로 분리하는 단계별 지침을 제공합니다.
서비스 참조 구성 대화 상자	**서비스 참조 구성** 대화 상자의 사용자 인터페이스 요소에 대해 설명 합니다.

참조

- System.ServiceModel
- System.Data.Services

참조

- .NET용 Visual Studio 데이터 도구

개념적 모델 작업 (WCF Data Services)

2020-01-06 • 3 minutes to read • Edit Online

개념적 모델을 사용 하 여 데이터베이스의 데이터를 설명 하는 경우 데이터베이스 스키마와 개체 모델 간을 앞뒤로 변환 하지 않고 개체를 통해 데이터를 쿼리할 수 있습니다.

WCF Data Services 애플리케이션과 함께 개념적 모델을 사용할 수 있습니다. 다음 항목에서는 개념적 모델을 통해 데이터를 쿼리 하는 방법을 보여 줍니다.

항목	설명
방법: 데이터 서비스 쿼리 실행	.NET 응용 프로그램에서 데이터 서비스를 쿼리 하는 방법을 보여 줍니다.
방법: 프로젝트 쿼리 결과	데이터 서비스 쿼리를 통해 반환 되는 데이터의 양을 줄이는 방법을 보여 줍니다.

개념적 모델을 사용 하는 경우 도메인에 일치 하는 언어로 유효한 데이터 종류를 정의할 수 있습니다. 모델에 유효한 데이터를 정의 하거나 엔터티 또는 데이터 서비스에서 수행 하는 작업에 유효성 검사를 추가할 수 있습니다.

다음 항목에서는 WCF Data Services 애플리케이션에 유효성 검사를 추가하는 방법을 보여 줍니다.

항목	설명
방법: 데이터 서비스 메시지 가로채기	데이터 서비스 작업에 유효성 검사를 추가 하는 방법을 보여 줍니다.

다음 항목에서는 엔터티에 대 한 작업을 수행 하 여 데이터를 만들고, 업데이트 하 고, 삭제 하는 방법을 보여 줍니다.

항목	설명
방법: 엔터티 추가, 수정 및 삭제	데이터 서비스에서 엔터티 데이터를 만들고, 업데이트 하 고, 삭제 하는 방법을 보여 줍니다.
방법: 엔터티 관계 정의	데이터 서비스에서 관계를 만들거나 변경 하는 방법을 보여 줍니다.

참조

- Windows Communication Foundation 서비스 및 Visual Studio의 WCF Data Services
- 데이터 서비스 쿼리

방법: 서비스의 데이터에 연결

2020-01-06 • 5 minutes to read • Edit Online

데이터 소스 구성 마법사 를 실행 하 고 **데이터 소스 형식 선택** 페이지에서 **서비스** 를 선택 하 여 서비스에서 반환 된 데이터에 응용 프로그램을 연결 합니다.

마법사가 완료 되 면 서비스 참조가 프로젝트에 추가 되 고 데이터 소스 창에서 바로 사용할 수 있습니다.

> **NOTE**
> **데이터 원본** 창에 표시되는 항목은 서비스에서 반환하는 정보에 따라 달라집니다. **데이터 원본 구성 마법사**에서 바인딩 가능한 개체를 만들기에 충분한 정보를 제공하지 않는 서비스도 있습니다. 예를 들어 서비스에서 형식화 되지 않은 데이터 집합을 반환 하는 경우 마법사를 완료할 때 **데이터 소스** 창에 항목이 표시 되지 않습니다. 이는 형식화 되지 않은 데이터 집합이 스키마를 제공 하지 않으므로 마법사에 데이터 소스를 만들 수 있는 충분 한 정보가 없기 때문입니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

응용 프로그램을 서비스에 연결 하려면

1. **데이터** 메뉴에서 **새 데이터 소스 추가**를 클릭합니다.
2. **데이터 소스 형식 선택** 페이지에서 **서비스** 를 선택 하 고 **다음**을 클릭 합니다.
3. 사용할 서비스의 주소를 입력 하거나 **검색** 을 클릭 하 여 현재 솔루션에서 서비스를 찾은 다음 **이동**을 클릭 합니다.
4. 필요에 따라 기본값 대신 새 **네임 스페이스** 를 입력할 수 있습니다.

   > **NOTE**
   > **고급** 을 클릭 하 여 서비스 참조 구성 대화 상자를 엽니다.

5. **확인** 을 클릭 하 여 프로젝트에 서비스 참조를 추가 합니다.
6. **마침**을 클릭합니다.

   데이터 원본이 **데이터 원본** 창에 추가됩니다.

다음 단계

응용 프로그램에 기능을 추가 하려면 **데이터 소스** 창에서 항목을 선택 하 고 폼으로 끌어서 바인딩된 컨트롤을 만듭니다. 자세한 내용은 Visual Studio에서 데이터에 컨트롤 바인딩을 참조 하세요.

참조

- WCF 데이터 서비스에 WPF 컨트롤 바인딩
- Windows Communication Foundation 서비스 및 Visual Studio의 WCF 데이터 서비스

연습: Windows Forms에서 간단한 WCF 서비스 만들기

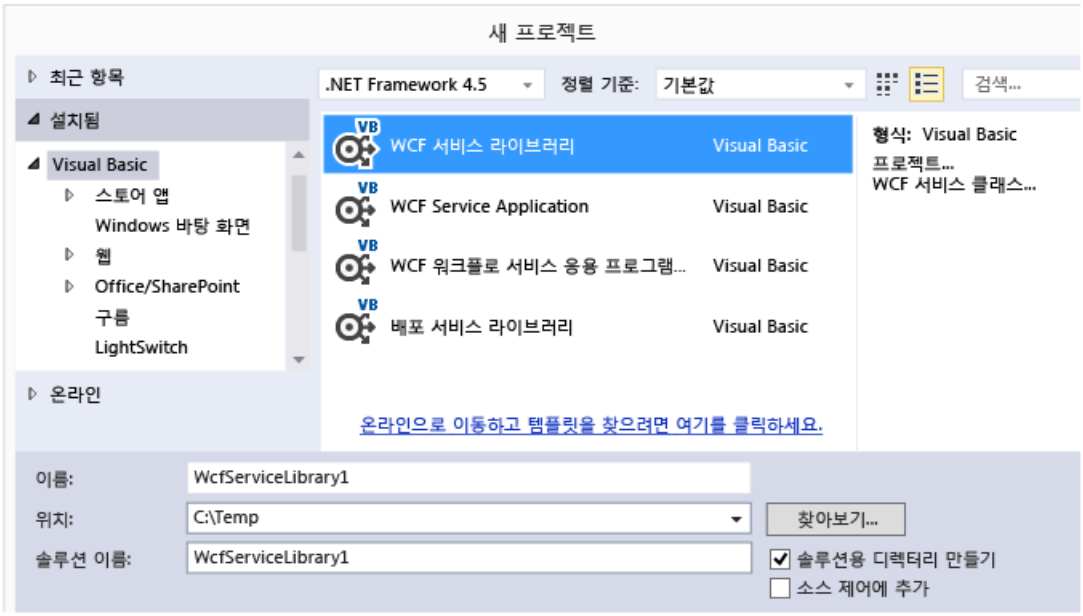
2020-01-06 • 9 minutes to read • Edit Online

이 연습에서는 WCF (simple Windows Communication Foundation) 서비스를 만들고 테스트 한 다음 Windows Forms 응용 프로그램에서 액세스 하는 방법을 보여 줍니다.

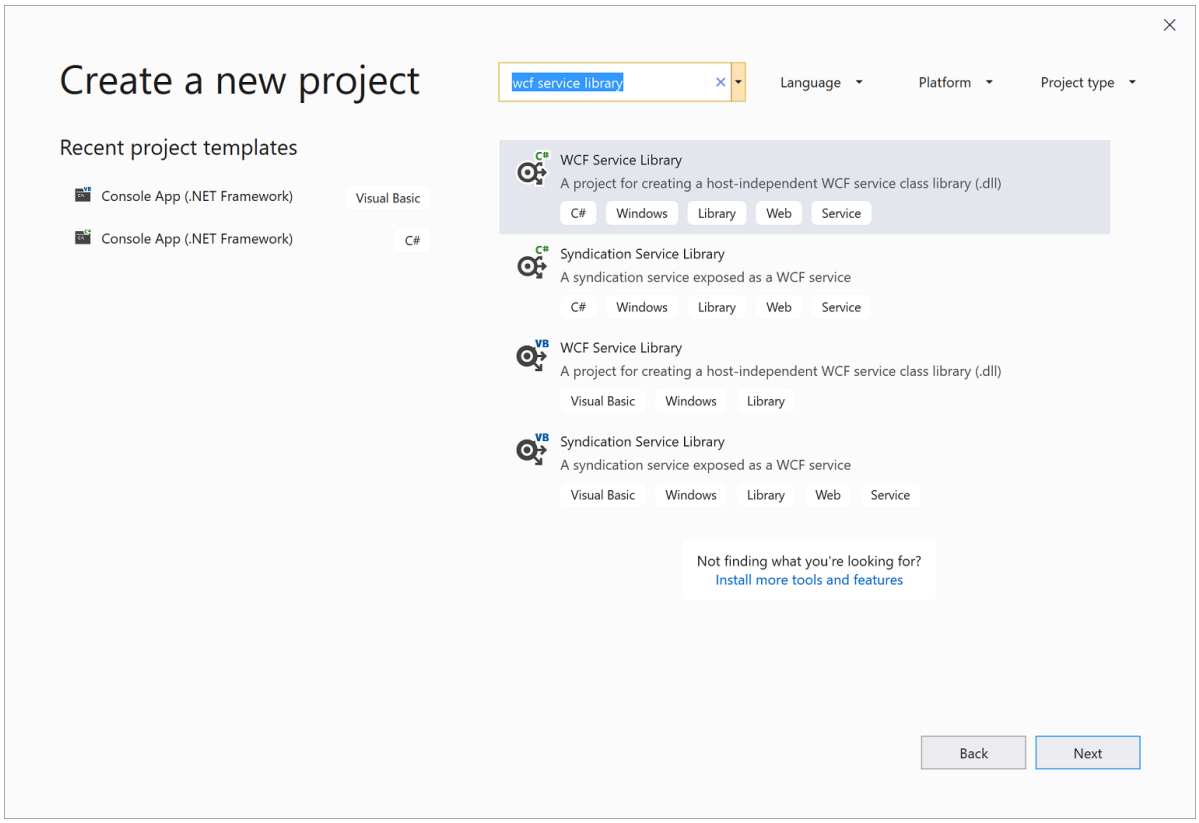
> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

서비스 만들기

1. Visual Studio를 엽니다.
2. **파일** 메뉴에서 **새로 만들기** > **프로젝트**를 선택 합니다.
3. **새 프로젝트** 대화 상자에서 **Visual Basic** 또는 **시각적 C#** 노드를 확장 하 고 wcf, **wcf 서비스 라이브러리**를 **차례로**선택 합니다.
4. **확인**을 클릭하여 프로젝트를 만듭니다.



2. 시작 창에서 **새 프로젝트 만들기**를 선택합니다.
3. **새 프로젝트 만들기** 페이지의 검색 상자에 **wcf 서비스 라이브러리** 를 입력 합니다. C# **WCF 서비스 라이브러리**의 또는 Visual Basic 템플릿을 선택 하 고 **다음**을 클릭 합니다.

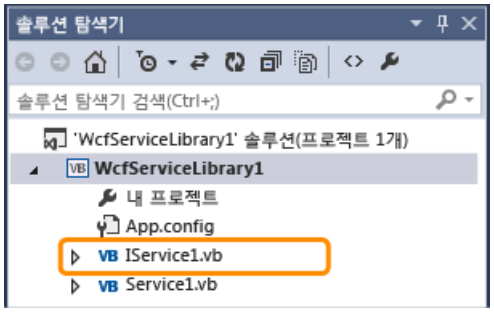


> **TIP**
>
> 템플릿이 표시 되지 않는 경우 Visual Studio의 **Windows Communication Foundation** 구성 요소를 설치 해야 할 수 있습니다. **추가 도구 및 기능 설치** 를 선택 하 여 Visual Studio 설치 관리자을 엽니다. **개별 구성 요소** 탭을 선택 하 고 **개발 작업**으로 스크롤한 다음 **Windows Communication Foundation**를 선택 합니다. **수정**을 클릭합니다.

4. **새 프로젝트 구성** 페이지에서 **만들기**를 클릭 합니다.

> **NOTE**
>
> 이렇게 하면 테스트 및 액세스할 수 있는 작업 서비스가 만들어집니다. 다음 두 단계는 다른 데이터 형식을 사용하도록 기본 메서드를 수정하는 방법을 보여 줍니다. 실제 애플리케이션에서는 서비스에 사용자 고유의 함수를 추가할 수도 있습니다.

5. **솔루션 탐색기**에서 **IService1** 또는 **IService1.cs**를 두 번 클릭 합니다.



다음 줄을 찾습니다.

```
[OperationContract]
string GetData(int value);
```

```
<OperationContract()>
Function GetData(ByVal value As Integer) As String
```

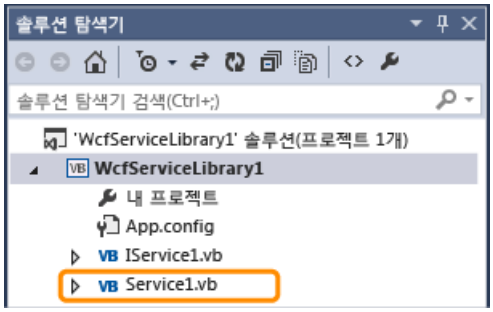
`value` 매개 변수의 형식을 문자열로 변경 합니다.

```
[OperationContract]
string GetData(string value);
```

```
<OperationContract()>
Function GetData(ByVal value As String) As String
```

위의 코드에서 `<OperationContract()>` 또는 `[OperationContract]` 특성을 확인합니다. 이러한 특성은 서비스에서 노출하는 모든 메서드에 필요합니다.

6. **솔루션 탐색기**에서 **Service1** 또는 **Service1.cs**를 두 번 클릭 합니다.



다음 줄을 찾습니다.

```
Public Function GetData(ByVal value As Integer) As String Implements IService1.GetData
    Return String.Format("You entered: {0}", value)
End Function
```

```
public string GetData(int value)
{
    return string.Format("You entered: {0}", value);
}
```

`value` 매개 변수의 형식을 문자열로 변경 합니다.

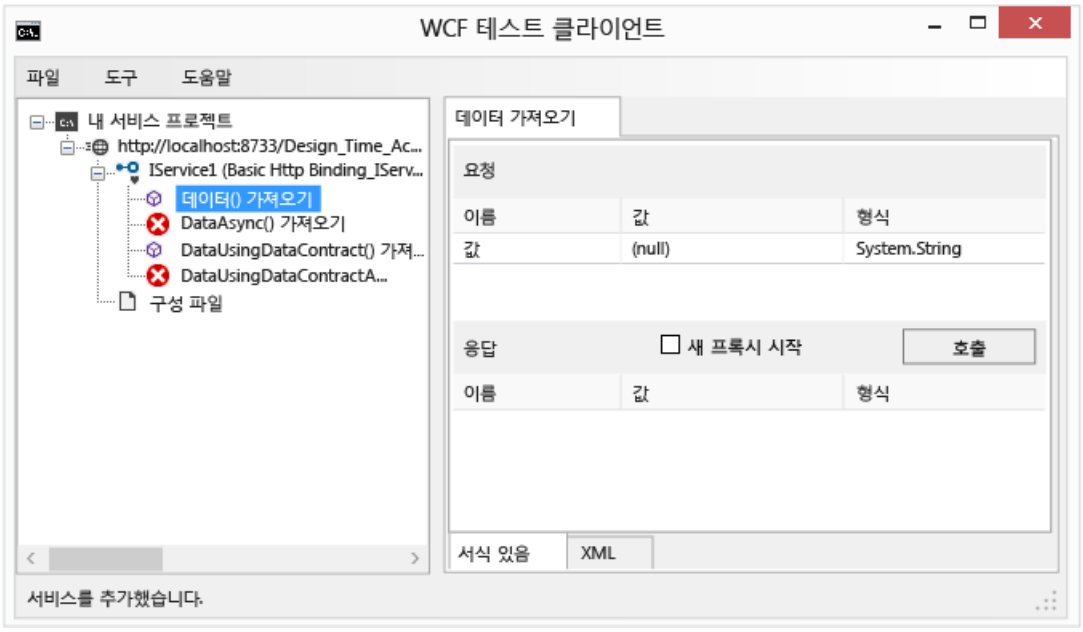
```
public string GetData(string value)
{
    return string.Format("You entered: {0}", value);
}
```

```
Public Function GetData(ByVal value As String) As String Implements IService1.GetData
    Return String.Format("You entered: {0}", value)
End Function
```

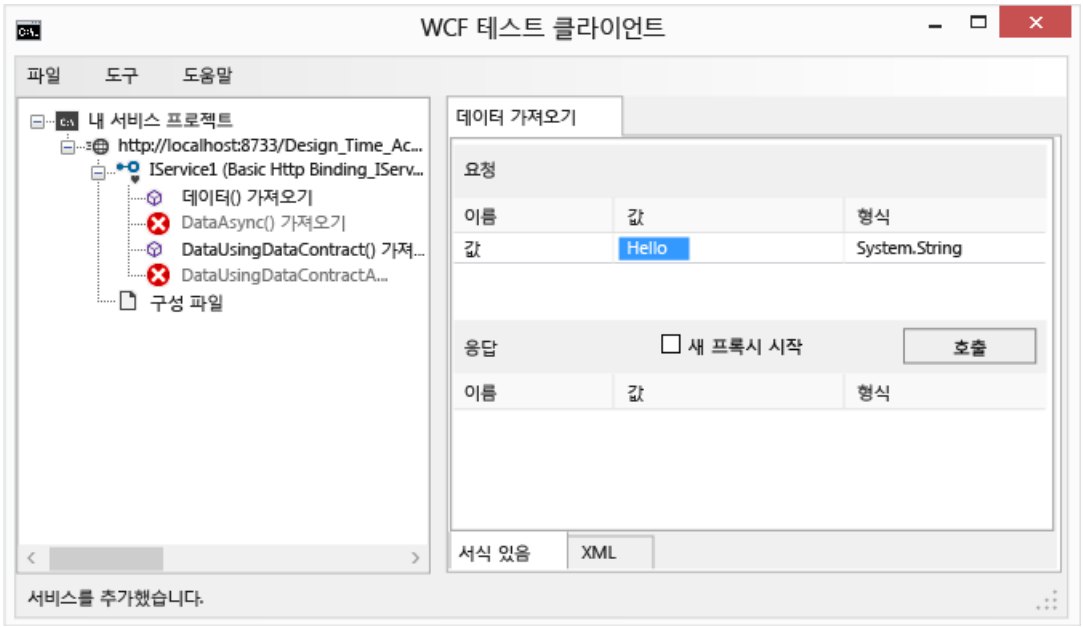
서비스 테스트

1. **F5** 키를 눌러 서비스를 실행합니다. **WCF 테스트 클라이언트** 폼이 표시 되 고 서비스가 로드 됩니다.
2. **WCF 테스트 클라이언트** 폼에서 **IService1** 아래의 **GetData()** 메서드를 두 번 클릭합니다. **GetData** 탭이

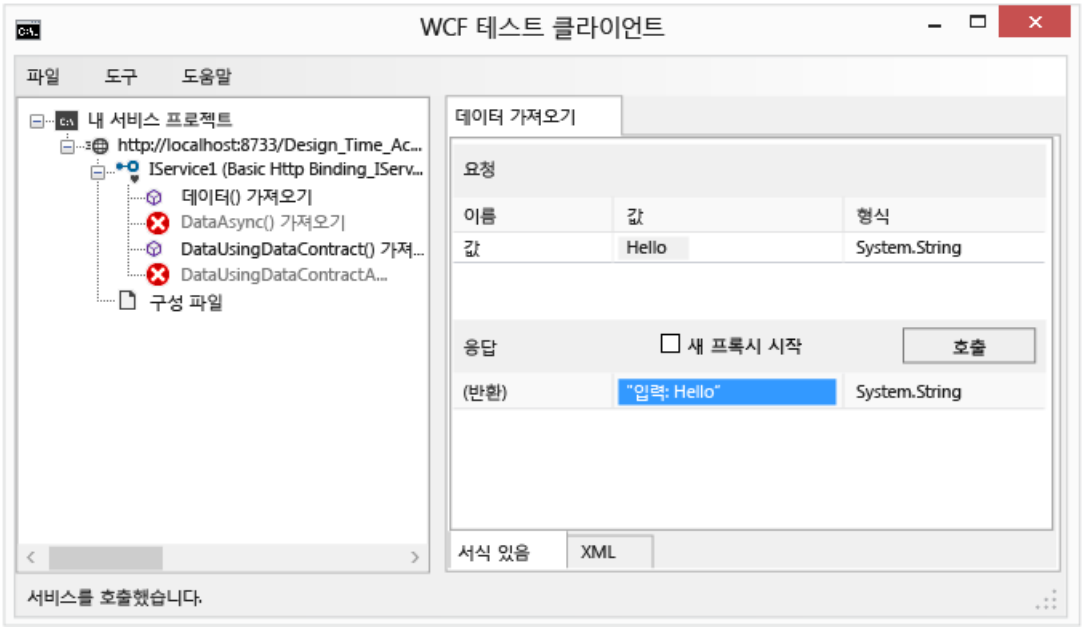
나타납니다.



3. **요청** 상자에서 **값** 필드와 형식 `Hello`를 선택합니다.



4. **호출** 단추를 클릭합니다. **보안 경고** 대화 상자가 표시 되 면 **확인**을 클릭 합니다. 결과는 **응답** 상자에 표시 됩니다.

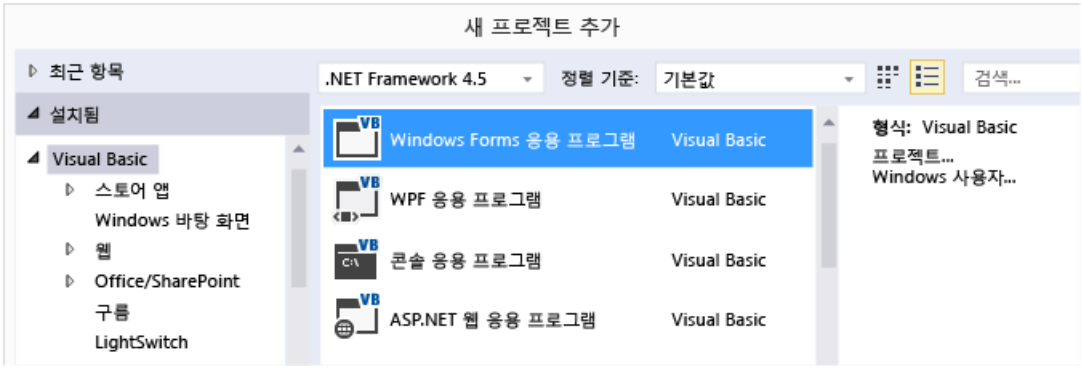


5. **파일** 메뉴에서 **끝내기**를 클릭하여 테스트 폼을 닫습니다.

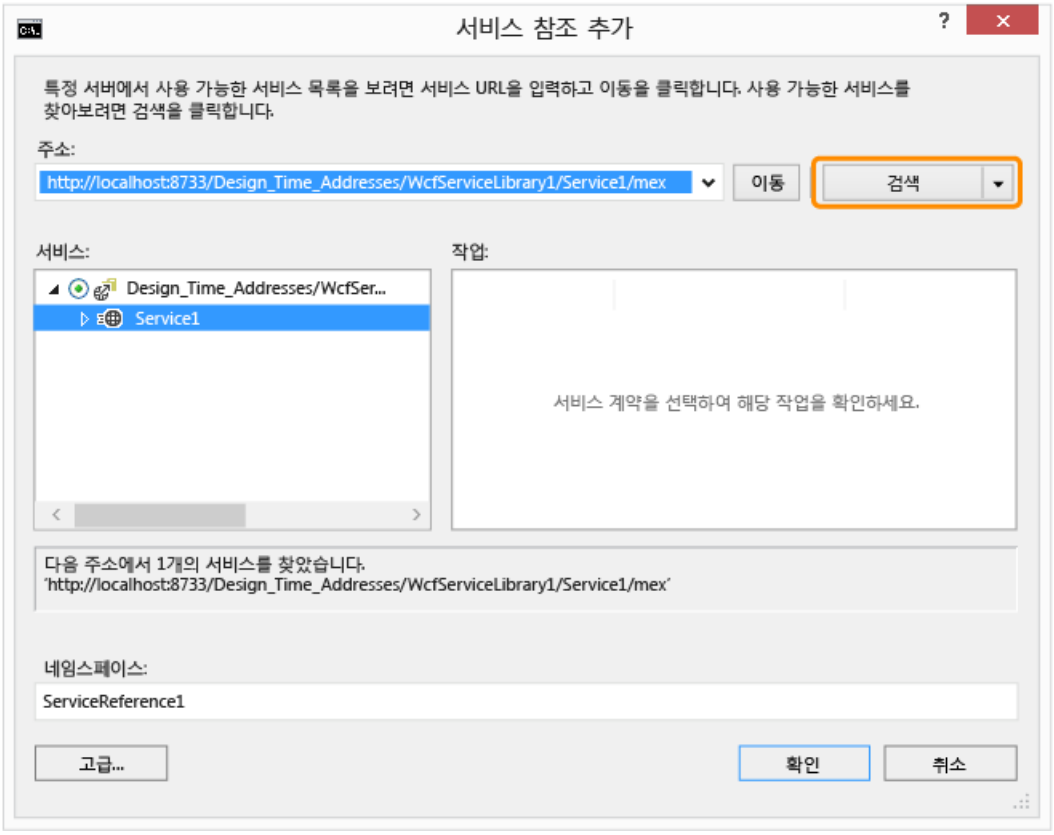
서비스 액세스

WCF 서비스 참조

1. **파일** 메뉴에서 **추가**를 가리킨 다음, **새 프로젝트**를 클릭합니다.
2. **새 프로젝트** 대화 상자에서 **Visual Basic** 또는 **시각적 C#** 노드를 확장 하 고 **Windows**를 선택한 다음 **Windows Forms 응용 프로그램**을 선택 합니다. **확인**을 클릭하여 프로젝트를 엽니다.



3. **WindowsApplication1**을 마우스 오른쪽 단추로 클릭하고 **서비스 참조 추가**를 클릭합니다. **서비스 참조 추가** 대화 상자가 나타납니다.
4. **서비스 참조 추가** 대화 상자에서 **검색**을 클릭합니다.



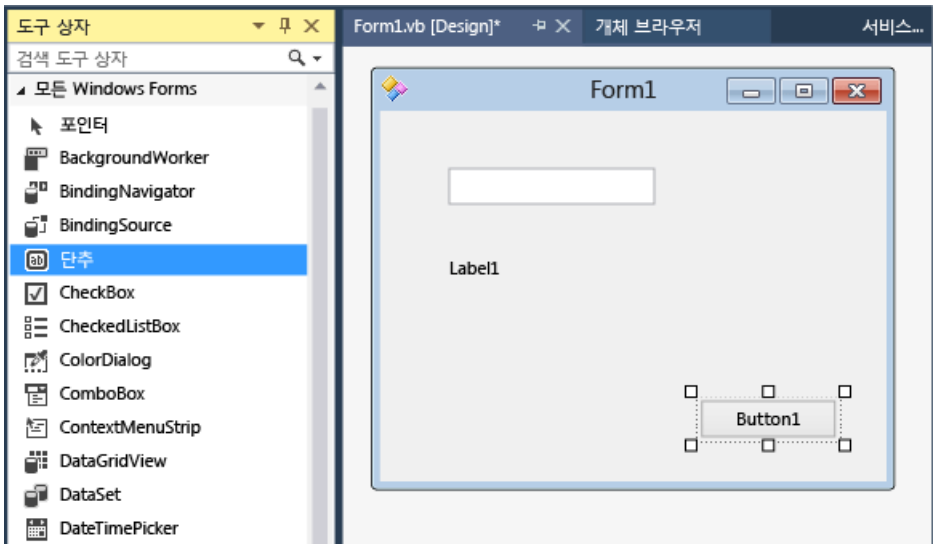
Service1 이 **서비스** 창에 표시 됩니다.

5. **확인**을 클릭하여 서비스 참조를 추가합니다.

클라이언트 애플리케이션을 빌드합니다

1. Windows Forms 디자이너를 아직 열지 않은 경우 **솔루션 탐색기**에서 **Form1.vb** 또는 **Form1.cs**을 두 번 클릭하여 엽니다.

2. **도구 상자**에서 `TextBox` 컨트롤, `Label` 컨트롤 및 `Button` 컨트롤을 폼으로 끌어옵니다.



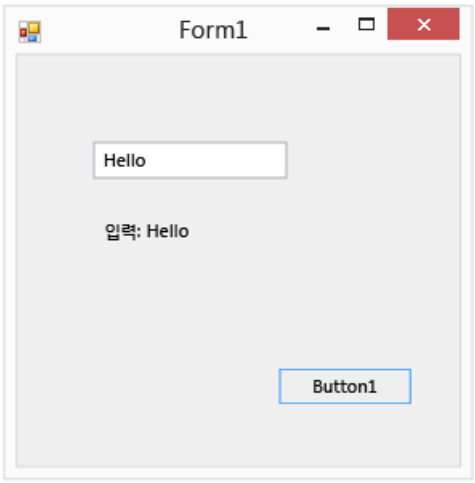
3. `Button`을 두 번 클릭하고 다음 코드를 `Click` 이벤트 처리기에 추가합니다.

```
private void button1_Click(System.Object sender, System.EventArgs e)
{
    ServiceReference1.Service1Client client = new
        ServiceReference1.Service1Client();
    string returnString;

    returnString = client.GetData(textBox1.Text);
    label1.Text = returnString;
}
```

```
  Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
      Dim client As New ServiceReference1.Service1Client
      Dim returnString As String

      returnString = client.GetData(TextBox1.Text)
      Label1.Text = returnString
  End Sub
```

4. **솔루션 탐색기**에서 **WindowsApplication1**을 마우스 오른쪽 단추로 클릭한 다음, **시작 프로젝트로 설정**을 클릭합니다.

5. **F5** 키를 눌러 프로젝트를 실행합니다. 텍스트를 입력하고 단추를 클릭합니다. 레이블은 "입력 했습니다."를 표시 하 고 입력 한 텍스트를 표시 합니다.



참조

- Windows Communication Foundation 서비스 및 Visual Studio의 WCF Data Services

연습: WPF 및 Entity Framework를 사용하여 WCF 데이터 서비스 만들기

2020-01-06 • 25 minutes to read • Edit Online

이 연습에서는 ASP.NET 웹 애플리케이션에서 호스팅되는 WCF Data Service를 만든 다음, Windows Forms 애플리케이션에서 이 서비스에 액세스하는 방법을 보여줍니다.

이 연습에서는 다음을 수행 합니다.

- WCF Data Service를 호스팅하는 웹 애플리케이션을 만듭니다.
- Northwind 데이터베이스의 `Customers` 테이블을 나타내는 엔터티 데이터 모델를 만듭니다.
- WCF Data Service를 만듭니다.
- 클라이언트 애플리케이션을 만들고 WCF Data Service에 대한 참조를 추가합니다.
- 서비스에 대한 데이터 바인딩을 활성화하고 사용자 인터페이스를 생성합니다.
- 필요한 경우 애플리케이션에 필터링 기능을 추가합니다.

전제 조건

이 연습에서는 SQL Server Express LocalDB 및 Northwind 샘플 데이터베이스를 사용 합니다.

1. LocalDB SQL Server Express 없는 경우 SQL Server Express 다운로드 페이지에서 또는 **Visual Studio 설치 관리자**를 통해 설치 합니다. **Visual Studio 설치 관리자**에서 **데이터 저장소 및 처리** 워크 로드의 일부로 또는 개별 구성 요소로 SQL Server Express LocalDB를 설치할 수 있습니다.
2. 다음 단계를 수행 하 여 Northwind 샘플 데이터베이스를 설치 합니다.

   a. Visual Studio에서 **SQL Server 개체 탐색기** 창을 엽니다. **SQL Server 개체 탐색기** 는 **데이터 저장소 및 처리** 워크 로드의 일부로 Visual Studio 설치 관리자에 설치 됩니다. **SQL Server** 노드를 확장 합니다. LocalDB 인스턴스를 마우스 오른쪽 단추로 클릭 하 고 **새 쿼리**를 선택 합니다.

   쿼리 편집기 창이 열립니다.

   b. Northwind transact-sql 스크립트 를 클립보드에 복사 합니다. 이 T-sql 스크립트는 Northwind 데이터베이스를 처음부터 만들어 데이터로 채웁니다.

   c. T-sql 스크립트를 쿼리 편집기에 붙여 넣은 다음 **실행** 단추를 선택 합니다.

   잠시 후 쿼리 실행이 완료 되 고 Northwind 데이터베이스가 만들어집니다.

서비스 만들기

WCF Data Service를 만들려면 웹 프로젝트를 추가하고 엔터티 데이터 모델을 만든 다음, 이 모델에서 서비스를 만듭니다.

첫 번째 단계에서는 서비스를 호스트 하는 웹 프로젝트를 추가 합니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

웹 프로젝트를 만들려면

1. 메뉴 모음에서 **파일** > **새로 만들기** > **프로젝트**를 선택합니다.
2. **새 프로젝트** 대화 상자에서 **Visual Basic** 또는 **Visual C#** 및 **웹** 노드를 확장한 다음, **ASP.NET 웹 애플리케이션** 템플릿을 선택합니다.
3. **이름** 텍스트 상자에 **NorthwindWeb**을 입력한 다음, **확인** 단추를 선택합니다.
4. **새 ASP.NET 프로젝트** 대화 상자의 **템플릿 선택** 목록에서 **비어 있음**을 선택한 다음, **확인** 단추를 선택합니다.

다음 단계에서는 Northwind 데이터베이스의 `Customers` 테이블을 나타내는 엔터티 데이터 모델를 만듭니다.

엔터티 데이터 모델을 만들려면

1. 메뉴 모음에서 **프로젝트** > **새 항목 추가**를 선택합니다.
2. **새 항목 추가** 대화 상자에서 **데이터** 노드를 선택한 다음, **ADO.NET 엔터티 데이터 모델** 항목을 선택합니다.
3. **이름** 텍스트 상자에 `NorthwindModel`를 입력 한 다음 **추가** 단추를 선택 합니다.

   엔터티 데이터 모델 마법사가 나타납니다.
4. 엔터티 데이터 모델 마법사의 **모델 콘텐츠 선택** 페이지에서 **데이터베이스의 EF 디자이너** 항목을 선택한 후, **다음** 단추를 선택합니다.
5. **데이터 연결 선택** 페이지에서 다음 단계 중 하나를 수행합니다.

   - Northwind 샘플 데이터베이스에 대한 데이터 연결이 드롭다운 목록에 표시되면 해당 연결을 선택합니다.

     -또는-

   - **새 연결** 단추를 선택하여 새 데이터 연결을 구성합니다. 자세한 내용은 새 연결 추가를 참조 하세요.
6. 데이터베이스에 암호가 필요한 경우 **예, 중요한 데이터를 연결 문자열에 포함합니다.** 옵션 단추를 선택한 후, **다음** 단추를 선택합니다.

   > **NOTE**
   > 대화 상자가 나타나는 경우 **예**를 선택하여 프로젝트에 파일을 저장합니다.
7. **사용자 버전 선택** 페이지에서 **Entity Framework 5.0** 옵션 단추를 선택한 후, **다음** 단추를 선택합니다.

   > **NOTE**
   > 최신 버전의 Entity Framework 6을 WCF Services와 함께 사용하려면 WCF Data Services Entity Framework Provider NuGet 패키지를 설치해야 합니다. Entity Framework 6 +에서 WCF Data Services 5.6.0 사용을참조 하세요.
8. **데이터베이스 개체 선택** 페이지에서 **테이블** 노드를 확장하고 **Customers** 확인란을 선택한 다음, **마침** 단추를 선택합니다.

엔터티 모델 다이어그램이 표시 되 고 *NorthwindModel* 파일이 프로젝트에 추가 됩니다.

다음 단계에서는 데이터 서비스를 만들고 테스트 합니다.

데이터 서비스를 만들려면

1. 메뉴 모음에서 **프로젝트** > **새 항목 추가**를 선택합니다.
2. **새 항목 추가** 대화 상자에서 **웹** 노드를 선택한 다음, **WCF Data Service 5.6** 항목을 선택합니다.
3. **이름** 텍스트 상자에 `NorthwindCustomers`를 입력 한 다음 **추가** 단추를 선택 합니다.

   코드 편집기에 **NorthwindCustomers.svc** 파일이 표시됩니다.
4. **코드 편집기**에서 첫 번째 `TODO:` 주석을 찾아 다음 코드로 바꿉니다.

   ```
   Inherits DataService(Of northwindEntities)
   ```

   ```
   public class NorthwindCustomers : DataService<northwindEntities>
   ```
5. `InitializeService` 이벤트 처리기의 주석을 다음 코드로 바꿉니다.

   ```
   config.SetEntitySetAccessRule("*", EntitySetRights.All)
   ```

   ```
   config.SetEntitySetAccessRule("*", EntitySetRights.All);
   ```
6. 메뉴 모음에서 **디버그** > **디버깅 하지 않고 시작** 을 선택 하 여 서비스를 실행 합니다. 브라우저 창이 열리고 서비스에 대 한 XML 스키마가 표시 됩니다.
7. **주소** 표시줄에 **NorthwindCustomers**에 대 한 URL의 끝에 `Customers`를 입력 한 다음 **enter** 키를 선택 합니다.

   `Customers` 테이블의 데이터에 대 한 XML 표현이 표시 됩니다.

   > **NOTE**
   > Internet Explorer가 이 데이터를 RSS 피드로 잘못 해석하는 경우도 있습니다. RSS 피드를 표시하는 옵션은 반드시 비활성화되어 있어야 합니다. 자세한 내용은 서비스 참조 문제 해결을 참조하세요.
8. 브라우저 창을 닫습니다.

다음 단계에서는 서비스를 사용 하는 Windows Forms 클라이언트 응용 프로그램을 만듭니다.

클라이언트 애플리케이션 만들기

클라이언트 애플리케이션을 만들려면 두 번째 프로젝트를 추가하고, 프로젝트에 서비스 참조를 추가하고, 데이터 원본을 구성하고, 서비스의 데이터를 표시할 사용자 인터페이스를 만듭니다.

첫 번째 단계에서는 솔루션에 Windows Forms 프로젝트를 추가 하 고이를 시작 프로젝트로 설정 합니다.

클라이언트 애플리케이션을 만들려면

1. 메뉴 모음에서 파일, **추가** > **새 프로젝트**를 선택 합니다.
2. **새 프로젝트** 대화 상자에서 **Visual Basic** 또는 **시각적 C#** 노드를 확장 하 고 **Windows** 노드를 선택한 다음 **Windows Forms 응용 프로그램**을 선택 합니다.

3. **이름** 텍스트 상자에 `NorthwindClient`를 입력하고 **확인** 단추를 선택합니다.
4. **솔루션 탐색기**에서 **NorthwindClient** 프로젝트 노드를 선택합니다.
5. 메뉴 모음에서 **프로젝트**, **시작 프로젝트로 설정**을 차례로 선택합니다.

다음 단계에서는 웹 프로젝트의 WCF Data Service에 대 한 서비스 참조를 추가 합니다.

서비스 참조를 추가하려면

1. 메뉴 모음에서 **프로젝트** > **서비스 참조 추가**를 선택 합니다.
2. **서비스 참조 추가** 대화 상자에서 **검색** 단추를 선택합니다.

   NorthwindCustomers 서비스의 URL이 **주소** 필드에 표시됩니다.
3. **확인** 단추를 선택하여 서비스 참조를 추가합니다.

다음 단계에서는 서비스에 대 한 데이터 바인딩을 사용 하도록 데이터 소스를 구성 합니다.

서비스에 대한 데이터 바인딩을 사용하려면

1. 메뉴 모음에서 **보기** > **다른 창** > **데이터 소스**를 선택 합니다.

   데이터 원본 창이 열립니다.
2. **데이터 원본** 창에서 **새 데이터 원본 추가** 단추를 선택합니다.
3. **데이터 원본 구성 마법사**의 **데이터 소스 형식 선택** 페이지에서 **개체**를 선택한 후, **다음** 단추를 선택합니다.
4. **데이터 개체 선택** 페이지에서 **NorthwindClient** 노드를 확장한 다음, **NorthwindClient.ServiceReference1** 노드를 확장합니다.
5. **Customer** 확인란을 선택한 다음, **마침** 단추를 선택합니다.

다음 단계에서는 서비스의 데이터를 표시 하는 사용자 인터페이스를 만듭니다.

사용자 인터페이스를 만들려면

1. **데이터 원본** 창에서 **Customers** 노드에 대한 바로 가기 메뉴를 열고 **복사**를 선택합니다.
2. **Form1.vb** 또는 **Form1.c** 폼 디자이너에서 바로 가기 메뉴를 열고 **붙여넣기**를 선택합니다.

   DataGridView 컨트롤, BindingSource 구성 요소 및 BindingNavigator 구성 요소가 폼에 추가됩니다.
3. **CustomersDataGridView** 컨트롤을 선택한 다음, **속성** 창에서 **Dock** 속성을 **Fill**로 설정합니다.
4. **솔루션 탐색기**에서 **Form1** 노드에 대 한 바로 가기 메뉴를 열고 **코드 보기** 를 선택 하 여 코드 편집기를 열고 파일 맨 위에 다음 `Imports` 또는 `Using` 문을 추가 합니다.

   ```
   Imports NorthwindClient.ServiceReference1
   ```

   ```
   using NorthwindClient.ServiceReference1;
   ```
5. 다음 코드를 `Form1_Load` 이벤트 처리기에 추가합니다.

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim proxy As New NorthwindEntities _
(New Uri("http://localhost:53161/NorthwindCustomers.svc/"))
        Me.CustomersBindingSource.DataSource = proxy.Customers
    End Sub
```

```
private void Form1_Load(object sender, EventArgs e)
{
NorthwindEntities proxy = new NorthwindEntities(new
Uri("http://localhost:53161/NorthwindCustomers.svc/"));
this.CustomersBindingSource.DataSource = proxy.Customers;
}
```

6. **솔루션 탐색기**에서 **NorthwindCustomers.svc** 파일에 대한 바로 가기 메뉴를 열고 **브라우저에서 보기**를 선택합니다. Internet Explorer가 열리고 서비스에 대 한 XML 스키마가 표시 됩니다.

7. Internet Explorer 주소 표시줄에서 URL을 복사합니다.

8. 4단계에서 추가한 코드에서 `http://localhost:53161/NorthwindCustomers.svc/`를 선택하여 방금 복사한 URL로 바꿉니다.

9. 메뉴 모음에서 **디버그** > **디버깅 시작** 을 선택 하 여 응용 프로그램을 실행 합니다. 고객 정보가 표시 됩니다.

   이제 NorthwindCustomers 서비스의 고객 목록을 표시하는 애플리케이션이 만들어졌습니다. 이 서비스를 통해 추가 데이터를 노출하려면 Northwind 데이터베이스의 다른 테이블을 포함하도록 엔터티 데이터 모델을 수정하면 됩니다.

다음 선택적 단계에서는 서비스에서 반환 되는 데이터를 필터링 하는 방법에 대해 알아봅니다.

필터링 기능 추가

이 단계에서는 고객의 구/군/시를 기준으로 데이터를 필터링 하도록 응용 프로그램을 사용자 지정 합니다.

구/군/시 기준 필터링을 추가하려면

1. **솔루션 탐색기**에서 **Form1.vb** 또는 **Form1.cs** 노드의 바로 가기 메뉴를 열고 **열기**를 선택합니다.

2. Button도구 상자에서 **TextBox 컨트롤 및** 컨트롤을 폼에 추가합니다.

3. Button 컨트롤에 대 한 바로 가기 메뉴를 열고 **코드 보기**를 선택한 후 `Button1_Click` 이벤트 처리기에서 다음 코드를 추가 합니다.

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim proxy As New northwindEntities _
(New Uri("http://localhost:53161/NorthwindCustomers.svc"))
        Dim city As String = TextBox1.Text

        If city <> "" Then
            Me.CustomersBindingSource.DataSource = From c In _
         proxy.Customers Where c.City = city
        End If

    End Sub
```

```
private void Button1_Click(object sender, EventArgs e)
{
ServiceReference1.northwindModel.northwindEntities proxy = new northwindEntities(new
Uri("http://localhost:53161/NorthwindCustomers.svc"));
string city = TextBox1.Text;

if (!string.IsNullOrEmpty(city)) {
this.CustomersBindingSource.DataSource = from c in proxy.Customers where c.City == city;
}

}
```

4. 앞의 코드에서 `http://localhost:53161/NorthwindCustomers.svc`를 `Form1_Load` 이벤트 처리기의 URL로 바꿉니다.
5. 메뉴 모음에서 **디버그** > **디버깅 시작** 을 선택 하 여 응용 프로그램을 실행 합니다.
6. 텍스트 상자에 **London**을 입력한 다음, 단추를 선택합니다. London의 고객만 표시됩니다.

참조

- Windows Communication Foundation 서비스 및 Visual Studio의 WCF Data Services
- 방법: WCF 데이터 서비스 참조 추가, 업데이트 또는 제거

서비스 참조 문제 해결

2020-01-06 • 11 minutes to read • Edit Online

이 항목에서는 Visual Studio에서 WCF (Windows Communication Foundation) 또는 WCF Data Services 참조를 사용 하 여 작업할 때 발생할 수 있는 일반적인 문제에 대해 설명 합니다.

서비스에서 데이터를 반환 하는 동안 오류 발생

서비스에서 `DataSet` 또는 `DataTable`을 반환 하면 "들어오는 메시지의 최대 크기 할당량을 초과 했습니다." 예외가 표시 될 수 있습니다. 기본적으로 일부 바인딩의 `MaxReceivedMessageSize` 속성은 서비스 거부 공격에 대 한 노출을 제한 하기 위해 상대적으로 작은 값으로 설정 됩니다. 예외를 방지 하기 위해이 값을 늘릴 수 있습니다. 자세한 내용은 MaxReceivedMessageSize를 참조하세요.

이 오류를 해결하려면

1. **솔루션 탐색기**에서 *app.config* 파일을 두 번 클릭 하 여 엽니다.
2. `MaxReceivedMessageSize` 속성을 찾아 더 큰 값으로 변경 합니다.

내 솔루션에서 서비스를 찾을 수 없습니다.

서비스 참조 추가 대화 상자에서 **검색** 단추를 클릭 하면 솔루션에 있는 하나 이상의 WCF 서비스 라이브러리 프로젝트가 서비스 목록에 나타나지 않습니다. 이는 서비스 라이브러리가 솔루션에 추가 되었지만 아직 컴파일되지 않은 경우에 발생할 수 있습니다.

이 오류를 해결하려면

- **솔루션 탐색기**에서 WCF 서비스 라이브러리 프로젝트를 마우스 오른쪽 단추로 클릭 하 고 **빌드**를 클릭 합니다.

원격 데스크톱을 통해 서비스에 액세스 하는 동안 오류 발생

사용자가 원격 데스크톱 연결을 통해 웹 호스팅 WCF 서비스에 액세스 하 고 사용자에 게 관리 권한이 없는 경우 NTLM 인증을 사용 합니다. 사용자에 게 관리 권한이 없는 경우 사용자에 게 다음 오류 메시지가 표시 될 수 있습니다. "HTTP 요청은 클라이언트 인증 스키마 ' 익명 '으로 인증 되지 않습니다. 서버에서 받은 인증 헤더가 ' NTLM ' 이었습니다.

이 오류를 해결하려면

1. 웹 사이트 프로젝트에서 **속성** 페이지를 엽니다.
2. **시작 옵션** 탭에서 **NTLM 인증** 확인란의 선택을 취소 합니다.

> **NOTE**
> WCF 서비스를 단독으로 포함 하는 웹 사이트의 경우에만 NTLM 인증을 해제 해야 합니다. WCF 서비스에 대 한 보안은 *web.config 파일의* 구성을 통해 관리 됩니다. 이렇게 하면 NTLM 인증이 필요 하지 않습니다.

생성 된 클래스 설정에 대 한 액세스 수준이 아무런 영향을 주지 않습니다.

서비스 참조 구성 대화 상자의 **생성 된 클래스에 대 한 액세스 수준** 옵션을 **내부** 또는 **Friend** 로 설정 하면 항

상 작동 하지 않을 수 있습니다. 대화 상자에서 옵션을 설정 하는 것 처럼 보이는 경우에도 `Public`의 액세스 수준으로 결과 지원 클래스가 생성 됩니다.

이는 XmlSerializer을 사용 하 여 serialize 된 것과 같은 특정 형식에 대 한 알려진 제한 사항입니다.

서비스 코드 디버깅 오류

클라이언트 코드에서 WCF 서비스에 대 한 코드를 한 단계씩 코드 실행 하면 누락 된 기호와 관련 된 오류가 발생할 수 있습니다. 솔루션에 포함 된 서비스를 솔루션에서 이동 하거나 제거 하는 경우 이러한 문제가 발생할 수 있습니다.

현재 솔루션의 일부인 WCF 서비스에 대 한 참조를 처음 추가 하면 서비스 프로젝트와 서비스 클라이언트 프로젝트 간에 명시적 빌드 종속성이 추가 됩니다. 이렇게 하면 클라이언트가 항상 최신 서비스 이진 파일에 액세스할 수 있습니다 .이는 클라이언트 코드를 서비스 코드로 단계별로 실행 하는 것과 같은 디버깅 시나리오에서 특히 중요합니다.

서비스 프로젝트가 솔루션에서 제거 되는 경우이 명시적 빌드 종속성이 무효화 됩니다. Visual Studio는 필요에 따라 서비스 프로젝트를 다시 작성 하는 것을 더 이상 보장할 수 없습니다.

이 오류를 해결 하려면 서비스 프로젝트를 수동으로 다시 빌드해야 합니다.

1. **도구** 메뉴에서 **옵션**을 클릭합니다.
2. **옵션** 대화 상자에서 **프로젝트 및 솔루션**을 확장 한 다음 **일반**을 선택 합니다.
3. **고급 빌드 구성 표시** 확인란이 선택 되어 있는지 확인 한 다음 **확인**을 클릭 합니다.
4. WCF 서비스 프로젝트를 로드 합니다.
5. **Configuration Manager** 대화 상자에서 **활성 솔루션 구성을 디버그**로 설정 합니다. 자세한 내용은 방법: 구성 만들기 및 편집을 참조하세요.
6. **솔루션 탐색기**에서 WCF 서비스 프로젝트를 선택 합니다.
7. **빌드** 메뉴에서 **다시** 빌드를 클릭 하 여 WCF 서비스 프로젝트를 다시 빌드합니다.

브라우저에 표시 되지 WCF Data Services

WCF Data Service에서 데이터의 XML 표현을 보려고 하면 Internet Explorer에서 데이터를 RSS 피드로 잘못 해석할 수 있습니다. RSS 피드를 표시 하는 옵션이 사용 하지 않도록 설정 되어 있는지 확인 합니다.

이 오류를 해결 하려면 RSS 피드를 사용 하지 않도록 설정 합니다.

1. Internet Explorer의 **도구** 메뉴에서 **인터넷 옵션**을 클릭합니다.
2. **콘텐츠** 탭의 **피드** 섹션에서 **설정**을 클릭 합니다.
3. **피드 설정** 대화 상자에서 **피드 읽기용 보기 설정** 확인란의 선택을 취소 한 다음 **확인**을 클릭 합니다.
4. **확인**을 클릭하여 **인터넷 옵션** 대화 상자를 닫습니다.

참조

- Windows Communication Foundation 서비스 및 Visual Studio의 WCF Data Services

서비스 참조 구성 대화 상자

2020-01-06 • 9 minutes to read • Edit Online

서비스 참조 구성 대화 상자를 사용 하 여 WCF (Windows Communication Foundation) 서비스의 동작을 구성할 수 있습니다.

서비스 참조 구성 대화 상자에 액세스하려면 **솔루션 탐색기**에서 서비스 참조를 마우스 오른쪽 단추로 클릭하고 **서비스 참조 구성**을 선택합니다. **서비스 참조 추가 대화 상자**에서 **고급** 단추를 클릭하여 대화 상자에 액세스할 수도 있습니다.

작업 목록

- WCF 서비스가 호스팅되는 주소를 변경하려면 **주소** 필드에 새 주소를 입력합니다.
- WCF 클라이언트에서 클래스의 액세스 수준을 변경하려면 **생성된 클래스에 대한 액세스 수준** 목록에서 액세스 수준 키워드를 선택합니다.
- WCF 서비스의 메서드를 비동기적으로 호출하려면 **비동기 작업 생성** 확인란을 선택합니다.
- WCF 클라이언트에서 메시지 계약 형식을 생성하려면 **항상 메시지 계약 생성** 확인란을 선택합니다.
- WCF 클라이언트에 대해 목록 또는 사전 컬렉션 형식을 지정하려면 **컬렉션 형식** 및 **사전 컬렉션 형식** 목록에서 형식을 선택합니다.
- 형식 공유를 사용하지 않도록 설정하려면 **참조된 어셈블리의 형식 재사용** 확인란의 선택을 취소합니다. 일부 참조된 어셈블리에 대해 형식 공유를 사용하도록 설정하려면 **참조된 어셈블리의 형식 재사용** 확인란을 선택하고, **참조된 어셈블리 중 지정된 어셈블리의 형식 재사용**을 선택하고, **참조된 어셈블리 목록**에서 원하는 참조를 선택합니다.

UI 요소 목록

Address

서비스 참조가 서비스를 검색 하는 웹 주소를 업데이트 합니다. 예를 들어 개발 중에 서비스는 개발 서버에서 호스트 된 후 나중에 프로덕션 서버로 이동 하 여 주소를 변경할 수 있습니다.

> **NOTE**
> Address 요소는 **서비스 참조 추가 대화 상자**에서 **서비스 참조 구성** 대화 상자가 표시된 경우에는 사용할 수 없습니다.

생성된 클래스에 대한 액세스 수준

WCF 클라이언트 클래스에 대한 코드 액세스 수준을 결정합니다.

> **NOTE**
> 웹 사이트 프로젝트의 경우 이 옵션은 항상 `Public`으로 설정되며 변경할 수 없습니다. 자세한 내용은 서비스 참조 문제 해결을 참조하세요.

동기 작업 생성

WCF 서비스 메서드를 동기적으로 (기본값) 또는 비동기적으로 호출 하는지 여부를 결정 합니다.

작업 기반 작업 생성

비동기 코드를 작성할 때이 옵션을 사용 하면 .NET 4에서 도입 된 TPL (작업 병렬 라이브러리)을 활용할 수 있습니다. TPL (작업 병렬 라이브러리)을 참조 하세요.

항상 메시지 계약 생성

WCF 클라이언트에 대해 메시지 계약 형식이 생성 되는지 여부를 결정 합니다. 메시지 계약에 대한 자세한 내용은 메시지 계약 사용을 참조하세요.

컬렉션 형식

WCF 클라이언트에 대한 목록 컬렉션 형식을 지정합니다. 기본 유형은 Array입니다.

사전 컬렉션 형식

WCF 클라이언트에 대한 사전 컬렉션 형식을 지정합니다. 기본 유형은 Dictionary<TKey,TValue>입니다.

참조된 어셈블리의 형식 재사용

WCF 클라이언트가 서비스를 추가 하거나 업데이트할 때 새 형식을 생성 하는 대신 참조 된 어셈블리에 이미 있는 항목을 다시 사용 하려고 하는지 여부를 결정 합니다. 기본적으로 이 옵션은 선택되어 있습니다.

모든 참조된 어셈블리의 형식 재사용

이를 선택 하면 가능한 경우 **참조 된 어셈블리 목록의** 모든 형식이 다시 사용 됩니다. 기본적으로 이 옵션이 선택 됩니다.

참조된 어셈블리 중 지정된 어셈블리의 형식 재사용

이를 선택 하면 **참조 된 어셈블리 목록** 에서 선택한 유형만 다시 사용 됩니다.

참조된 어셈블리 목록

프로젝트 또는 웹 사이트에 대 한 참조 어셈블리의 목록을 포함 합니다. **지정 된 참조 된 어셈블리의 형식 재사용**을 선택 하는 경우 개별 어셈블리를 선택 하거나 선택 취소할 수 있습니다.

웹 참조 추가

웹 참조 추가 대화 상자를 표시합니다.

> **NOTE**
> 이 옵션은 .NET Framework 버전 2.0을 대상으로 하는 프로젝트에만 사용 해야 합니다.

> **NOTE**
> **웹 참조 추가** 단추는 **서비스 참조 추가 대화 상자**에서 **서비스 참조 구성** 대화 상자가 표시 되는 경우에만 사용할 수 있습니다.

참조

- 방법: 웹 서비스에 참조 추가
- Windows Communication Foundation 서비스 및 WCF Data Services

방법: WCF 데이터 서비스 참조 추가, 업데이트 또는 제거

2020-06-08 • 14 minutes to read • Edit Online

*서비스 참조*를 사용 하면 프로젝트에서 하나 이상의에 액세스할 수 있습니다 WCF Data Services . **서비스 참조 추가** 대화 상자를 사용 하 여 WCF Data Services 로컬 영역 네트워크 또는 인터넷에서 현재 솔루션의를 검색할 수 있습니다.

솔루션 탐색기 의 **연결된 서비스** 노드를 사용 하 여 **Microsoft WCF Web Service Reference Provider**에 액세스할 수 있으며,이를 통해 WCF (Windows Communication Foundation) 데이터 서비스 참조를 관리할 수 있습니다.

> **NOTE**
> 이 문서의 일부 Visual Studio 사용자 인터페이스 요소에 대한 다른 이름 또는 위치가 컴퓨터에 표시될 수 있습니다. 다른 버전의 Visual Studio 또는 다른 환경 설정을 사용 중일 수 있습니다. 자세한 내용은 IDE 개인 설정을 참조하세요.

WCF 서비스 참조 추가

외부 서비스에 대 한 참조를 추가 하려면

1. **솔루션 탐색기**에서 서비스를 추가 하려는 프로젝트의 이름을 마우스 오른쪽 단추로 클릭 한 다음 **서비스 참조 추가**를 클릭 합니다.

   서비스 참조 추가 대화 상자가 나타납니다.

2. **주소** 상자에 서비스에 대 한 URL을 입력 하 고 **이동** 을 클릭 하 여 서비스를 검색 합니다. 서비스에서 사용자 이름 및 암호 보안을 구현 하는 경우 사용자 이름 및 암호를 입력 하 라는 메시지가 표시 될 수 있습니다.

   > **NOTE**
   > 신뢰할 수 있는 원본의 서비스만 참조해야 합니다. 신뢰할 수 없는 원본에서 참조를 추가하면 보안이 손상될 수 있습니다.

   또한 올바른 서비스 메타 데이터를 찾은 이전 15 개 Url을 저장 하는 **주소** 목록에서 URL을 선택할 수 있습니다.

   검색을 수행 하는 동안 진행률 표시줄이 표시 됩니다. **중지**를 클릭 하 여 언제 든 지 검색을 중지할 수 있습니다.

3. **서비스** 목록에서 사용 하려는 서비스에 대 한 노드를 확장 하 고 엔터티 집합을 선택 합니다.

4. 참조에 사용하려는 네임스페이스를 **네임스페이스** 상자에 입력합니다.

5. **확인**을 클릭하여 프로젝트에 대한 참조를 추가합니다.

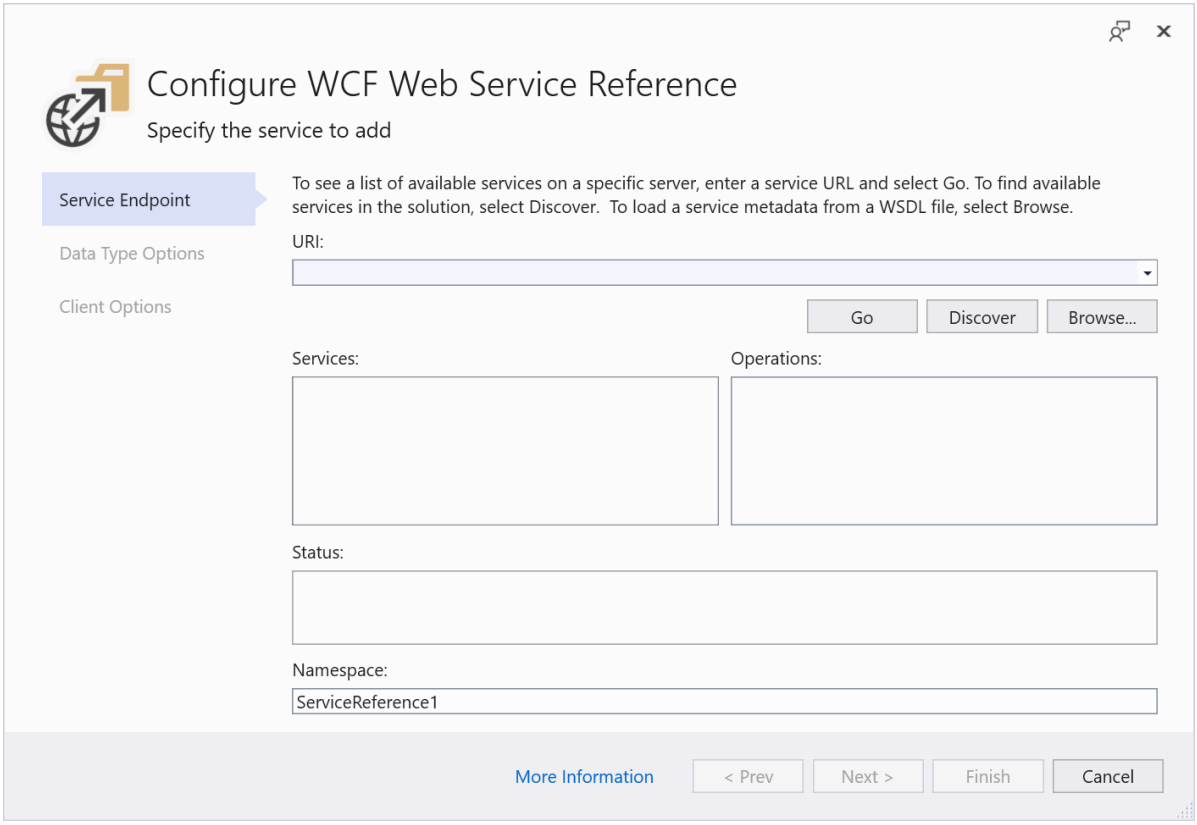
   서비스 클라이언트 (프록시)가 생성 되 고 서비스를 설명 하는 메타 데이터가 *app.config* 파일에 추가 됩니다.

1. **솔루션 탐색기**에서 **연결된 서비스** 노드를 두 번 클릭 하거나 누릅니다.

   서비스 구성 탭이 열립니다.

2. **Microsoft WCF Web Service Reference Provider**를 선택 합니다.

   WCF 웹 서비스 참조 구성 대화 상자가 나타납니다.



3. **URI** 상자에 서비스에 대 한 URL을 입력 한 다음 **이동** 을 클릭 하 여 서비스를 검색 합니다. 서비스에서 사용자 이름 및 암호 보안을 구현 하는 경우 사용자 이름 및 암호를 입력 하 라는 메시지가 표시 될 수 있습니다.

   > **NOTE**
   > 신뢰할 수 있는 원본의 서비스만 참조해야 합니다. 신뢰할 수 없는 원본에서 참조를 추가하면 보안이 손상될 수 있습니다.

   유효한 서비스 메타 데이터를 찾은 이전 15 개 Url을 저장 하는 **URI** 목록에서 URL을 선택할 수도 있습니다.

   검색을 수행 하는 동안 진행률 표시줄이 표시 됩니다. **중지**를 클릭 하 여 언제 든 지 검색을 중지할 수 있습니다.

4. **서비스** 목록에서 사용 하려는 서비스에 대 한 노드를 확장 하 고 엔터티 집합을 선택 합니다.

5. 참조에 사용하려는 네임스페이스를 **네임스페이스** 상자에 입력합니다.

6. **마침** 을 클릭 하 여 프로젝트에 참조를 추가 합니다.

   서비스 클라이언트 (프록시)가 생성 되 고 서비스를 설명 하는 메타 데이터가 *app.config* 파일에 추가 됩니다.

현재 솔루션의 서비스에 대 한 참조를 추가 하려면

1. **솔루션 탐색기**에서 서비스를 추가 하려는 프로젝트의 이름을 마우스 오른쪽 단추로 클릭 한 다음 **서비스 참조 추가**를 클릭 합니다.

   서비스 참조 추가 대화 상자가 나타납니다.

2. **검색**을 클릭 합니다.

   WCF Data Services현재 솔루션의 모든 서비스 (및 WCF 서비스)가 **서비스** 목록에 추가 됩니다.

3. **서비스** 목록에서 사용 하려는 서비스에 대 한 노드를 확장 하 고 엔터티 집합을 선택 합니다.
4. 참조에 사용하려는 네임스페이스를 **네임스페이스** 상자에 입력합니다.
5. **확인**을 클릭하여 프로젝트에 대한 참조를 추가합니다.

   서비스 클라이언트 (프록시)가를 생성 하 고, 서비스를 설명 하는 메타 데이터가 *app.config* 파일에 추가 됩니다.

1. **솔루션 탐색기**에서 **연결된 서비스** 노드를 두 번 클릭 하거나 누릅니다.

   서비스 구성 탭이 열립니다.
2. **Microsoft WCF Web Service Reference Provider**를 선택 합니다.

   WCF 웹 서비스 참조 구성 대화 상자가 나타납니다.
3. **검색**을 클릭 합니다.

   WCF Data Services현재 솔루션의 모든 서비스 (및 WCF 서비스)가 **서비스** 목록에 추가 됩니다.
4. **서비스** 목록에서 사용 하려는 서비스에 대 한 노드를 확장 하 고 엔터티 집합을 선택 합니다.
5. 참조에 사용하려는 네임스페이스를 **네임스페이스** 상자에 입력합니다.
6. **마침** 을 클릭 하 여 프로젝트에 참조를 추가 합니다.

   서비스 클라이언트 (프록시)가를 생성 하 고, 서비스를 설명 하는 메타 데이터가 *app.config* 파일에 추가 됩니다.

서비스 참조 업데이트

엔터티 데이터 모델는 경우에 WCF Data Services 따라 변경 됩니다. 이 경우 서비스 참조를 업데이트 해야 합니다.

서비스 참조를 업데이트 하려면

- **솔루션 탐색기**에서 서비스 참조를 마우스 오른쪽 단추로 클릭 한 다음 **서비스 참조 업데이트**를 클릭 합니다.

  원본 위치에서 참조를 업데이트 하는 동안 진행률 대화 상자가 표시 되 고 메타 데이터의 변경 내용을 반영 하도록 서비스 클라이언트가 다시 생성 됩니다.

서비스 참조 제거

서비스 참조가 더 이상 사용 되지 않는 경우 솔루션에서 제거할 수 있습니다.

서비스 참조를 제거 하려면

- **솔루션 탐색기**에서 서비스 참조를 마우스 오른쪽 단추로 클릭 한 다음 **삭제**를 클릭 합니다.

  서비스 클라이언트는 솔루션에서 제거 되 고, 서비스를 설명 하는 메타 데이터는 *app.config* 파일에서 제거 됩니다.

  > **NOTE**
  > 서비스 참조를 참조 하는 모든 코드는 수동으로 제거 해야 합니다.

참고 항목

- Visual Studio에서 Windows Communication Foundation Services 및 WCF data Services

.mdf 파일 업그레이드

2020-01-06 • 12 minutes to read • Edit Online

이 항목에서는 새 버전의 Visual Studio를 설치한 후 데이터베이스 파일 (*.mdf*)을 업그레이드 하기 위한 옵션에 대해 설명 합니다. 여기에는 다음 태스크에 대 한 지침이 포함 됩니다.

- 최신 버전의 SQL Server Express LocalDB를 사용 하도록 데이터베이스 파일 업그레이드
- 최신 버전의 SQL Server Express을 사용 하도록 데이터베이스 파일 업그레이드
- Visual Studio에서 데이터베이스 파일에 대 한 작업을 수행 하지만 이전 버전의 SQL Server Express 또는 LocalDB와의 호환성을 유지 합니다.
- 기본 데이터베이스 엔진 SQL Server Express 설정

Visual Studio를 사용 하 여 이전 버전의 SQL Server Express 또는 LocalDB를 사용 하 여 만든 데이터베이스 파일 (*.mdf*)을 포함 하는 프로젝트를 열 수 있습니다. 그러나 Visual Studio에서 프로젝트를 계속 개발 하려면 해당 버전의 SQL Server Express 또는 LocalDB가 Visual Studio와 같은 컴퓨터에 설치 되어 있거나 데이터베이스 파일을 업그레이드 해야 합니다. 데이터베이스 파일을 업그레이드 하는 경우 이전 버전의 SQL Server Express 또는 LocalDB를 사용 하 여 액세스할 수 없습니다.

이전 버전의 SQL Server Express 또는 LocalDB를 통해 만든 데이터베이스 파일의 버전이 현재 설치 되어 있는 SQL Server Express 또는 LocalDB의 인스턴스와 호환 되지 않는 경우 해당 데이터베이스 파일을 업그레이드 하 라는 메시지가 표시 될 수도 있습니다. 이 문제를 해결 하기 위해 Visual Studio에서 파일을 업그레이드할지 묻는 메시지를 표시 합니다.

> **IMPORTANT**
> 데이터베이스 파일을 업그레이드 하기 전에 백업 하는 것이 좋습니다.

> **WARNING**
> LocalDB 2014 (V12) 32 비트에서 LocalDB 2016 (V13) 이상으로 만든 *.mdf* 파일을 업그레이드 하는 경우에는 버전의 localdb 32에서 다시 파일을 열 수 없습니다.

데이터베이스를 업그레이드 하기 전에 다음 조건을 고려 합니다.

- 이전 버전 및 최신 버전의 Visual Studio에서 프로젝트를 사용 하려면 업그레이드 하지 마세요.
- 응용 프로그램이 LocalDB 대신 SQL Server Express를 사용 하는 환경에서 사용 되는 경우 업그레이드 하지 마세요.
- 응용 프로그램에서 원격 연결을 사용 하는 경우에는 LocalDB에서 해당 연결을 허용 하지 않으므로 업그레이드 하지 마십시오.
- 응용 프로그램이 인터넷 정보 서비스 (IIS)를 사용 하는 경우 업그레이드 하지 않습니다.
- 샌드박스 환경에서 데이터베이스 응용 프로그램을 테스트 하지만 데이터베이스를 관리 하지 않으려는 경우 업그레이드 하는 것이 좋습니다.

LocalDB 버전을 사용 하도록 데이터베이스 파일을 업그레이드 하려면

1. **서버 탐색기**에서 **데이터베이스에 연결** 단추를 선택 합니다.

2. **연결 추가** 대화 상자에서 다음 정보를 지정 합니다.

   - **데이터 원본**: `Microsoft SQL Server (SqlClient)`
   - **서버 이름**:
     - 기본 버전을 사용 하려면: `(localdb)\MSSQLLocalDB`. 설치 된 Visual Studio 버전 및 첫 번째 LocalDB 인스턴스를 만든 시간에 따라 ProjectV12 또는 ProjectV13를 지정 합니다. **SQL Server 개체 탐색기** 의 **MSSQLLocalDB** 노드에는 가리키는 버전이 표시 됩니다.
     - 특정 버전을 사용 하려면: `(localdb)\ProjectsV12` 또는 `(localdb)\ProjectsV13`를 사용 합니다. 여기서 V12는 LocalDB 2014이 고 V13는 LocalDB 2016입니다.
   - **데이터베이스 파일 연결**: 기본 *.mdf* 파일의 실제 경로입니다.
   - **논리적 이름**: 파일을 사용 하려는 이름입니다.

3. **확인** 단추를 선택합니다.

4. 메시지가 표시 되 면 **예** 단추를 선택 하 여 파일을 업그레이드 합니다.

   데이터베이스가 업그레이드 되 고 LocalDB 데이터베이스 엔진에 연결 되며 이전 버전의 LocalDB와 더 이상 호환 되지 않습니다.

연결에 대 한 바로 가기 메뉴를 열고 **연결 수정**을 선택 하 여 LocalDB를 사용 하도록 SQL Server Express 연결을 수정할 수도 있습니다. **연결 수정** 대화 상자에서 서버 이름을 `(LocalDB)\MSSQLLocalDB`로 변경 합니다. **고급 속성** 대화 상자에서 **사용자 인스턴스가 False**로 설정 되어 있는지 확인 합니다.

SQL Server Express 버전을 사용 하도록 데이터베이스 파일을 업그레이드 하려면

1. 데이터베이스에 대 한 연결에 대 한 바로 가기 메뉴에서 **연결 수정**을 선택 합니다.
2. **연결 수정** 대화 상자에서 **고급** 단추를 선택 합니다.
3. **고급 속성** 대화 상자에서 서버 이름을 변경 하지 않고 **확인** 단추를 선택 합니다.

   데이터베이스 파일은 SQL Server Express 현재 버전과 일치 하도록 업그레이드 됩니다.

Visual Studio에서 데이터베이스를 사용 하지만 **SQL Server Express**와의 호환성을 유지 하려면

- Visual Studio에서 프로젝트를 업그레이드 하지 않고 엽니다.
  - 프로젝트를 실행 하려면 **F5** 키를 선택 합니다.
  - 데이터베이스를 편집 하려면 **솔루션 탐색기**에서 *.mdf* 파일을 열고 **서버 탐색기** 에서 노드를 확장 하 여 데이터베이스 작업을 수행 합니다.

기본 데이터베이스 엔진 **SQL Server Express** 만들려면

1. 메뉴 모음에서 **도구** > **옵션**을 선택합니다.
2. **옵션** 대화 상자에서 **데이터베이스 도구** 옵션을 확장 한 다음 **데이터 연결**을 선택 합니다.
3. **SQL Server 인스턴스 이름** 입력란에 사용 하려는 SQL Server Express 또는 LocalDB 인스턴스의 이름을 지정 합니다. 인스턴스에 이름이 지정 되지 않은 경우 `.\SQLEXPRESS or (LocalDB)\MSSQLLocalDB`를 지정 합니다.
4. **확인** 단추를 선택합니다.

   SQL Server Express는 응용 프로그램에 대 한 기본 데이터베이스 엔진입니다.

참조

- Visual Studio에서 데이터 액세스

포럼에서 데이터 액세스 오류 문제 해결

2020-01-06 • 2 minutes to read • Edit Online

MSDN(Microsoft Developer Network) 공개 포럼에서 오류 및 경고 문제 해결 관련 지원 정보를 찾을 수 있습니다. 다음은 MSDN에서 사용할 수 있는 몇 가지 데이터 관련 포럼입니다.

- Windows Forms 데이터 컨트롤 및 데이터 바인딩
- ADO.NET 데이터 집합 포럼
- ADO.NET Entity Framework 및 LINQ to Entities
- WCF Data Services 포럼
- SQL Server 데이터 액세스 포럼
- LINQ to SQL 포럼
- ADO.NET 데이터 공급자 포럼

SQL Server에 대 한 연결 문제를 해결 하는 방법에 대 한 자세한 내용은 SQL Server 데이터베이스 엔진에 대 한 연결 문제 해결을 참조 하세요.

참조

- .NET용 Visual Studio 데이터 도구

DataContext 메서드(O/R 디자이너)

2020-01-06 • 12 minutes to read • Edit Online

Visual Studio의 LINQ to SQL 도구컨텍스트에서 DataContext 메서드는 데이터베이스에서 저장 프로시저 및 함수를 실행 하는 DataContext 클래스의 메서드입니다.

DataContext 클래스는 SQL Server 데이터베이스와 해당 데이터베이스에 매핑되는 LINQ to SQL 엔터티 클래스 간의 연결 통로 역할을 하는 LINQ to SQL 클래스입니다. DataContext 클래스에는 데이터베이스에 연결 하 고 데이터베이스의 데이터를 조작 하기 위한 연결 문자열 정보 및 메서드가 포함 되어 있습니다. 기본적으로 DataContext 클래스에는 LINQ to SQL 클래스에서 데이터베이스로 업데이트 된 데이터를 보내는 SubmitChanges 메서드와 같이 사용자가 호출할 수 있는 여러 메서드가 포함 되어 있습니다. 또한 저장 프로시저 및 함수에 매핑되는 DataContext 메서드를 추가로 만들 수 있습니다. 즉, 이러한 사용자 지정 메서드를 호출 하면 DataContext 메서드가 매핑되는 데이터베이스의 저장 프로시저 또는 함수가 실행 됩니다. 클래스를 확장 하는 메서드를 추가하는 것처럼 DataContext 클래스에 새 메서드를 추가할 수 있습니다. 그러나 **O/R 디자이너**컨텍스트의 DataContext 메서드에 대 한 토론에서 설명 하는 저장 프로시저 및 함수에 매핑되는 DataContext 메서드입니다.

메서드 창

저장 프로시저 및 함수에 매핑되는 DataContext 메서드는 **O/R 디자이너**의 **메서드** 창에 표시 됩니다. **메서드** 창은 기본 디자인 화면인 **엔터티** 창 옆에 있는 창입니다. **메서드** 창에는 **O/R 디자이너**를 사용 하 여 만든 모든 DataContext 메서드가 나열 됩니다. 기본적으로 **메서드** 창은 비어 있습니다. **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 저장 프로시저 또는 함수를 **O/R 디자이너**로 끌어 DataContext 메서드를 만들고 **메서드** 창을 채웁니다. 자세한 내용은 방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기 (O/R 디자이너)를 참조 하세요.

> **NOTE**
> **O/R 디자이너** 를 마우스 오른쪽 단추로 클릭 한 다음 메서드 창 **숨김** 또는 **메서드 창 표시**를 클릭 하거나 바로 가기 키 **CTRL+1**을 사용 하 여 메서드 창을 열고 닫습니다.

DataContext 메서드의 두 가지 형식

DataContext 메서드는 데이터베이스의 저장 프로시저와 함수에 매핑되는 메서드로 **O/R 디자이너**의 **메서드** 창에서 DataContext 메서드를 만들고 추가할 수 있습니다. DataContext 메서드에는 다음과 같이 결과 세트를 하나 이상 반환하는 형식과 반환하지 않는 형식이 있습니다.

- 결과 집합을 하나 이상 반환하는 DataContext 메서드

  애플리케이션이 저장 프로시저 및 함수를 데이터베이스에서 실행하고 결과를 반환하는 작업만 수행할 때 이 DataContext 메서드를 만듭니다. 자세한 내용은 방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기 (O/R 디자이너), ISingleResult<t > 및 IMultipleResults를 참조 하세요.

- 특정 엔터티 클래스에 대한 Inserts, Updates, Deletes 등과 같이 결과 집합을 반환하지 않는 DataContext 메서드

  애플리케이션에서 엔터티 클래스와 데이터베이스 간에 수정된 데이터를 저장할 때 기본 LINQ to SQL 동작을 사용하는 대신 저장 프로시저를 실행해야 하는 경우 이 DataContext 메서드를 만듭니다. 자세한 내용은 방법: 저장 프로시저를 할당 하 여 업데이트, 삽입 및 삭제 수행 (O/R 디자이너)을 참조 하세요.

DataContext 메서드의 반환 형식

서버 탐색기 또는 **데이터베이스 탐색기** 의 저장 프로시저 및 함수를 **O/R 디자이너**로끌어 오면 생성 된 DataContext 메서드의 반환 형식은 항목을 끌어 놓은 위치에 따라 달라 집니다. 항목을 기존 엔터티 클래스에 직접 놓으면 엔터티 클래스의 반환 형식을 사용 하 여 DataContext 메서드가 만들어집니다. **O/R 디자이너** 의 빈 영역 (두 창)에 항목을 놓으면 자동으로 생성 된 형식을 반환 하는 DataContext 메서드가 만들어집니다. 자동으로 생성 되는 형식은 저장 프로시저 또는 함수에서 반환 된 필드에 매핑되는 저장 프로시저 또는 함수 이름 및 속성과 일치 하는 이름을 갖습니다.

> **NOTE**
> 메서드 창에 추가한 후 DataContext 메서드의 반환 형식을 변경할 수 있습니다. DataContext 메서드의 반환 형식을 검사 하거나 변경하려면 해당 메서드를 선택하고 **속성** 창에서 **반환 형식** 속성을 검사합니다. 자세한 내용은 방법: DataContext 메서드의 반환 형식 변경 (O/R 디자이너)을 참조 하세요.

데이터베이스에서 O/R 디자이너 화면으로 끌어 온 개체는 데이터베이스의 개체 이름에 따라 자동으로 이름이 지정 됩니다. 같은 개체를 두 번 이상 끌면 이름을 구별 하는 새 이름의 끝에 숫자가 추가 됩니다. 데이터베이스 개체 이름에 공백이 포함되어 있거나 Visual Basic 또는 C#에서 지원되지 않는 문자가 사용되었으면 해당 공백이나 잘못된 문자가 밑줄로 대체됩니다.

참조

- Visual Studio의 LINQ to SQL 도구
- LINQ to SQL
- 저장 프로시저
- 방법: 저장 프로시저 및 함수에 매핑된 DataContext 메서드 만들기(O/R 디자이너)
- 방법: 저장 프로시저를 할당하여 업데이트, 삽입 및 삭제 수행(O/R 디자이너)
- 연습: 엔터티 클래스의 삽입, 업데이트 및 삭제 동작을 사용자 지정
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)

데이터 클래스 상속(O/R 디자이너)

2019-12-02 • 7 minutes to read • Edit Online

다른 개체와 마찬가지로 LINQ to SQL 클래스는 상속을 사용할 수 있고 다른 클래스에서 파생 될 수 있습니다. 코드에서 클래스 하나가 다른 클래스에서 상속되도록 선언하여 개체 간의 상속 관계를 지정할 수 있습니다. 데이터베이스에서 상속 관계는 여러 가지 방법으로 만들어집니다. **개체 관계형 디자이너** (**O/R 디자이너**)는 관계형 시스템에서 주로 구현 되는 단일 테이블 상속 개념을 지원 합니다.

단일 테이블 상속에서는 기본 클래스와 파생 클래스의 열을 모두 포함하는 데이터베이스 테이블이 하나 있습니다. 관계형 데이터의 경우 판별자 열에는 해당 레코드가 어느 클래스에 속해 있는지를 판별하는 값이 포함됩니다. 예를 들어 회사에서 사용 하는 모든 사용자를 포함 하는 `Persons` 테이블이 있다고 가정 합니다. 어떤 사람들은 사원이고 어떤 사람들은 관리자입니다. @No__t_0 테이블에는 관리자의 값이 1이 고 직원의 값이 2 인 `Type` 라는 열이 포함 되어 있습니다. @No__t_0 열은 판별자 열입니다. 이 시나리오에서는 직원의 서브 클래스를 만들고 `Type` 값이 2 인 레코드로만 클래스를 채울 수 있습니다.

O/R 디자이너를 사용하여 엔터티 클래스에서 상속을 구성하려면 상속 데이터가 들어 있는 단일 테이블을 디자이너에 두 번, 즉 상속 계층 구조의 각 클래스에 대해 테이블을 한 번씩 끌어 와야 합니다. 디자이너에 테이블을 추가한 후에는 **개체 관계형 디자이너** 도구 상자의 상속 항목을 사용하여 두 테이블을 연결한 다음, **속성** 창에서 네 가지 상속 속성을 설정합니다.

상속 속성

다음 표에는 상속 속성과 해당 설명이 나와 있습니다.

속성	설명
판별자 속성	현재 레코드가 속해 있는 클래스를 결정하는 속성이며 열에 매핑됩니다.
기본 클래스 판별자 값	**판별자 속성**으로 지정된 열에서 레코드가 기본 클래스에 속함을 나타내는 값입니다.
파생된 클래스 판별자 값	**판별자 속성**으로 지정된 속성에서 레코드가 파생 클래스에 속함을 나타내는 값입니다.
상속 기본값	**판별자 속성** 으로 지정 된 속성의 값이 **기본 클래스 판별자 값** 또는 **파생 클래스 판별자 값**과 일치 하지 않을 때 채워지는 클래스입니다.

상속을 사용하고 관계형 데이터에 대응하는 개체 모델을 만드는 것은 다소 복잡할 수 있습니다. 이 항목에서는 상속을 구성하는 데 필요한 기본 개념과 개별 속성에 대한 정보를 제공합니다. 다음 항목에는 **O/R 디자이너**를 사용하 여 상속을 구성 하는 방법에 대 한 자세한 설명이 나와 있습니다.

항목	설명
방법: O/R 디자이너를 사용하여 상속 구성	**O/R 디자이너**를 사용 하 여 단일 테이블 상속을 사용 하는 엔터티 클래스를 구성 하는 방법을 설명 합니다.
연습: 단일 테이블 상속을 사용하여 LINQ to SQL 클래스 만들기(O/R 디자이너)	**O/R 디자이너**를 사용 하 여 단일 테이블 상속을 사용 하는 엔터티 클래스를 구성 하는 방법에 대 한 단계별 지침을 제공 합니다.

참고 항목

- Visual Studio의 LINQ to SQL 도구
- 연습: LINQ to SQL 클래스 만들기(O-R 디자이너)
- 연습: 단일 테이블 상속을 사용하여 LINQ to SQL 클래스 만들기(O/R 디자이너)
- 시작

선택한 클래스가 하나 이상의 DataContext 메서드의 반환 형식으로 사용되므로 해당 클래스를 삭제할 수 없습니다.

2020-01-16 • 2 minutes to read • Edit Online

하나 이상의 DataContext 메서드 반환 형식은 선택한 엔터티 클래스입니다. DataContext 메서드에 대 한 반환 형식으로 사용 되는 엔터티 클래스를 삭제 하면 프로젝트 컴파일이 실패 합니다. 선택한 엔터티 클래스를 삭제하려면 해당 엔터티 클래스를 사용하는 DataContext 메서드를 식별하고 해당 메서드의 반환 형식을 서로 다른 엔터티 클래스로 설정합니다.

DataContext 메서드의 반환 형식을 원래 자동으로 생성된 형식으로 되돌리려면 우선 **메서드** 창에서 DataContext 메서드를 삭제한 다음, 개체를 **서버 탐색기/데이터베이스 탐색기**에서 **O/R 디자이너**로 다시 끌어 옵니다.

이 오류를 해결하려면

1. **메서드** 창에서 DataContext 메서드를 선택하고 **속성** 창에서 **반환 형식** 속성을 검사하여 엔터티 클래스를 반환 형식으로 사용하는 DataContext 메서드를 식별합니다.
2. **반환 형식**을 서로 다른 엔터티 클래스로 설정하거나 메서드 창에서 DataContext 메서드를 삭제합니다.

참조

- Visual Studio의 LINQ to SQL 도구

연결 문자열에 일반 텍스트 암호가 있는 자격 증명이 포함되어 있으며 통합 보안을 사용하지 않습니다.

2020-01-06 • 2 minutes to read • Edit Online

중요한 정보가 포함된 연결 문자열을 현재 DBML 파일 및 응용 프로그램 구성 파일에 저장하시겠습니까? 중요한 정보를 포함하지 않고 연결 문자열을 저장하려면 **아니요**를 클릭하세요.

연결 문자열에 포함된 암호와 같이 중요한 정보가 들어 있는 데이터 연결을 사용하는 경우 연결 문자열에 중요한 정보를 포함시키거나 제외시켜 프로젝트의 DBML 파일 및 애플리케이션 구성 파일에 저장하는 옵션이 제공됩니다.

WARNING

연결 속성의 **애플리케이션 설정** 속성을 명시적으로 **False**로 설정하면 DBML 파일에 암호가 추가됩니다.

저장 옵션

- 중요 한 정보를 사용 하 여 연결 문자열을 저장 하려면 **예**를 선택 합니다.

  연결 문자열이 애플리케이션 설정대로 저장됩니다. 연결 문자열에는 중요한 정보가 일반 텍스트로 포함됩니다. DBML 파일에는 중요한 데이터가 포함되지 않습니다.

- 중요 한 정보를 포함 하지 않고 연결 문자열을 저장 하려면 **아니요**를 선택 합니다.

  연결 문자열이 애플리케이션 설정대로 저장되지만 암호는 포함되지 않습니다.

참조

- Visual Studio의 LINQ to SQL 도구

<property name> 속성은 <association name> 연결에 참여하고 있으므로 삭제할 수 없음

2019-12-02 • 2 minutes to read • Edit Online

선택한 속성이 오류 메시지에 표시된 클래스 간 연결의 **연결 속성**으로 설정되었습니다. 속성이 데이터 클래스 간 연결에 참여 중인 경우에는 해당 속성을 삭제할 수 없습니다.

연결 속성을 데이터 클래스의 다른 속성으로 설정하면 원하는 속성을 삭제할 수 있습니다.

이 오류를 해결하려면

1. **O/R 디자이너**에서 오류 메시지에 표시된 데이터 클래스를 연결하는 연결 선을 선택합니다.
2. 해당 선을 두 번 클릭하여 **연결 편집기** 대화 상자를 엽니다.
3. **연결 속성**에서 속성을 제거합니다.
4. 속성 삭제를 다시 한 번 시도합니다.

참조

- Visual Studio의 LINQ to SQL 도구

<property name> 속성을 삭제할 수 없음

2019-12-02 • 2 minutes to read • Edit Online

<property name> 속성은 <class name> 및 <class name> 간 상속의 **판별자 속성**으로 설정되어 있으므로 삭제할 수 없음

선택한 속성이 오류 메시지에 표시된 클래스 간 상속의 **판별자 속성**으로 설정되었습니다. 속성이 데이터 클래스 간 상속 구성에 참여 중인 경우에는 해당 속성을 삭제할 수 없습니다.

판별자 속성을 데이터 클래스의 다른 속성으로 설정하면 원하는 속성을 삭제할 수 있습니다.

이 오류를 해결하려면

1. **O/R 디자이너**에서 오류 메시지에 표시된 데이터 클래스를 연결하는 상속 선을 선택합니다.
2. **판별자** 속성을 다른 속성으로 설정합니다.
3. 속성 삭제를 다시 한 번 시도합니다.

참조

- Visual Studio의 LINQ to SQL 도구

애플리케이션 설정 파일에서 연결 속성이 없거나 잘못되었습니다.

2020-01-06 • 2 minutes to read • Edit Online

응용 프로그램 설정 파일의 연결 속성이 없거나 잘못되었습니다. *.dbml* 파일의 연결 문자열이 대신 사용되었습니다.

.dbml 파일에는 찾을 수 없는 애플리케이션 설정 파일의 연결 문자열에 대한 참조가 포함되어 있습니다. 이 메시지는 정보 제공을 위한 것이며 **확인**을 클릭할 때 연결 문자열 설정이 만들어집니다.

이 메시지에 응답 하려면 **확인**을 선택 합니다. *.dbml* 파일에 포함된 연결 정보는 애플리케이션 설정에 추가됩니다.

참조

- Visual Studio의 LINQ to SQL 도구

<association name> 연결을 만들 수 없습니다. - 속성 형식이 일치하지 않는 경우

2019-12-02 • 2 minutes to read • Edit Online

<association name> 연결을 만들 수 없습니다. - 속성 형식이 일치하지 않는 경우. 속성에 일치하는 형식이 없습니다: <property names>.

연결이 **연결 편집기** 대화 상자에서 선택한 **연결 속성**에 의해 정의되었습니다. 연결의 각 측에 대한 속성은 동일한 데이터 형식이어야 합니다.

메시지에 나열된 속성은 동일한 데이터 형식을 갖지 않습니다.

이 오류를 해결하려면

1. 메시지를 검사하여 메시지에서 호출된 속성을 기록합니다.
2. **확인**을 클릭하여 대화 상자를 닫습니다.
3. **연결 속성**을 검사하고 동일한 데이터 형식의 속성을 선택합니다.
4. **확인**을 클릭합니다.

참고 항목

- Visual Studio의 LINQ to SQL 도구
- 방법: LINQ to SQL 클래스 (O/R 디자이너) 간에 연결을 만듭니다

경고: 동작 구성 대화 상자에 적용되지 않은 변경이 있음

2019-12-02 • 2 minutes to read • Edit Online

경고. 동작 구성 대화 상자에 적용되지 않은 변경 내용이 있습니다. 변경 내용을 적용하시겠습니까?

동작 구성 대화 상자를 사용 하 여 사용 가능한 모든 클래스에 대 한 `Insert`, `Update` 및 `Delete` 동작을 구성할 수 있습니다. 이 메시지는 새 **클래스** 및 **동작** 조합을 선택하고 이전 변경 내용이 아직 적용되지 않은 경우 나타납니다.

변경 옵션

- 변경 내용을 적용 하 고 계속 하려면 **예**를 클릭 합니다. 변경 내용은 선택한 **클래스** 및 **동작**에 적용 됩니다.
- 이전 변경 내용을 취소 하 고 계속 하려면 **아니요**를 클릭 합니다.

참조

- Visual Studio의 LINQ to SQL 도구

지원되지 않는 데이터베이스 공급자에서 데이터베이스 공급자를 선택했습니다.

2020-01-16 • 2 minutes to read • Edit Online

O/R 디자이너 는 SQL Server에 대 한 .NET Framework Data Provider (System.Data.SqlClient)만 지원 합니다. **확인**을 클릭하고 지원되지 않은 데이터베이스 공급자의 개체를 사용하여 계속 작업을 할 수 있지만 런타임에 예상치 못한 동작이 발생할 수도 있습니다.

> **NOTE**
> .NET Framework Data Provider for SQL Server를 사용하는 데이터 연결만 지원됩니다.

Options

- **확인**을 클릭하여 지원되지 않는 데이터베이스 공급자를 사용하는 연결에 매핑할 엔터티 클래스를 계속 디자인합니다. 지원되지 않는 데이터베이스 공급자를 사용하면 예상치 못한 동작이 발생할 수도 있습니다.
- 작업을 중지 하려면 **취소** 를 클릭 합니다. SQL Server에 대 한 .NET Framework 공급자를 사용 하는 다른 데이터 연결을 만들거나 사용 합니다.

참조

- Visual Studio의 LINQ to SQL 도구

이 관련 메서드는 다음과 같은 기본 삽입, 업데이트 및 삭제 메서드를 지원하는 메서드입니다.

2019-12-02 • 2 minutes to read • Edit Online

이 관련 메서드는 다음과 같은 기본 `Insert`, `Update` 또는 `Delete` 메서드에 대 한 지원 메서드입니다. 이 관련 메서드를 삭제하면 이러한 메서드도 삭제됩니다. 계속하시겠습니까?

선택한 `DataContext` 메서드는 **O/R 디자이너**에서 엔터티 클래스 중 하나에 대 한 `Insert`, `Update` 또는 `Delete` 메서드 중 하나로 사용 됩니다. 선택한 메서드를 삭제 하면이 메서드를 사용 하는 엔터티 클래스가 업데이트 중에 삽입, 업데이트 또는 삭제를 수행 하는 기본 런타임 동작으로 되돌아갑니다.

선택한 메서드 옵션

- 선택한 메서드를 삭제 하 여 엔터티 클래스가 런타임 업데이트를 사용 하도록 하려면 **예**를 클릭 합니다.

  선택한 메서드가 삭제 되 고이 메서드를 사용 하 여 업데이트 동작을 재정의 하는 모든 클래스가 기본 LINQ to SQL 런타임 동작을 사용 하 여 되돌립니다.

- 선택한 메서드를 변경 하지 않고 메시지 상자를 닫으려면 **아니요**를 클릭 합니다.

  메시지 상자가 닫히고 변경되지 않습니다.

참고 항목

- Visual Studio의 LINQ to SQL 도구

선택한 항목 중 하나 이상이 디자이너에서 지원되지 않는 데이터 형식을 포함하고 있습니다.

2020-01-06 • 2 minutes to read • Edit Online

O/r 디자이너에서 **서버 탐색기** 또는 **데이터베이스 탐색기** 끌어온 항목 중 하나 이상이 **o/r 디자이너**에서 지원하지 않는 데이터 형식 (예: CLR 사용자 정의 형식)을 포함 합니다.

이 오류를 해결하려면

1. 지원되지 않는 데이터 형식이 들어 있지 않은 테이블을 기초로 뷰를 만듭니다.
2. **서버 탐색기** 또는 **데이터베이스 탐색기** 디자이너에서 뷰를 끌어 놓습니다.

참조

- Visual Studio의 LINQ to SQL 도구

DataContext 메서드의 반환 형식 변경은 취소할 수 없습니다.

2019-12-02 • 3 minutes to read • Edit Online

DataContext 메서드의 반환 형식을 변경하면 실행 취소할 수 없습니다. 자동으로 생성된 형식으로 되돌리려면 항목을 **서버 탐색기** 또는 **데이터베이스 탐색기**에서 O/R 디자이너로 끌어와야 합니다. 반환 형식을 변경하시겠습니까?

DataContext 메서드의 반환 형식은 O/R 디자이너에서 항목을 드롭하는 위치에 따라 달라집니다. 항목을 기존 엔터티 클래스에 직접 드롭하면 엔터티 클래스의 반환 형식을 갖는 DataContext 메서드가 만들어집니다. 항목을 O/R 디자이너의 빈 영역에 놓으면 자동으로 생성된 형식을 반환하는 DataContext 메서드가 만들어집니다. 메서드 창에 추가한 후 DataContext 메서드의 반환 형식을 변경할 수 있습니다. DataContext 메서드의 반환 형식을 검사하거나 변경하려면 해당 메서드를 선택하고 **속성** 창에서 **반환 형식** 속성을 클릭합니다.

DataContext의 반환 형식을 변경하려면

- **예**를 클릭합니다.

반환 형식을 변경하지 않고 메시지 상자를 끝내려면

- **아니요**를 클릭합니다.

반환 형식을 변경한 후 원래 반환 형식으로 되돌리려면

1. O/R 디자이너에서 DataContext 메서드를 선택한 다음, 삭제합니다.
2. **서버 탐색기/데이터베이스 탐색기**에서 항목을 찾아 O/R 디자이너로 끌어옵니다.

   원래 기본 반환 형식을 갖는 DataContext 메서드가 만들어집니다.

참조

- Visual Studio의 LINQ to SQL 도구

디버깅하는 동안에는 디자이너를 수정할 수 없습니다.

2020-01-16 • 2 minutes to read • Edit Online

이 메시지는 애플리케이션이 디버그 모드로 실행 중일 때 **O/R 디자이너**에서 항목을 수정하려는 경우 나타납니다. 애플리케이션이 디버그 모드로 실행 중인 경우 **O/R 디자이너**는 읽기 전용입니다.

이 오류를 해결 하려면 **디버그** 메뉴에서 **디버깅 중지** 를 선택 합니다. 응용 프로그램에서 디버깅을 중지 하 고 **O/R 디자이너**에서 항목을 수정할 수 있습니다.

참조

- Visual Studio의 LINQ to SQL 도구

선택한 연결에서 지원되지 않는 데이터베이스 공급자를 사용합니다.

2020-01-06 • 2 minutes to read • Edit Online

이 메시지는 SQL Server에 대 한 .NET Framework Data Provider를 사용 하지 않는 항목을 **서버 탐색기** 또는 **데이터베이스 탐색기** 에서 Visual Studio의 LINQ to SQL 도구로끌 때 나타납니다.

O/R 디자이너 는 SQL Server에 대해 .NET Framework 공급자를 사용 하는 데이터 연결만 지원 합니다. Microsoft SQL Server 또는 Microsoft SQL Server 데이터베이스 파일에 대한 연결만 유효합니다.

이 오류를 해결 하려면 SQL Server에 대 한 .NET Framework Data Provider를 사용 하는 데이터 연결의 항목만 **O/R 디자이너**에 추가 합니다.

참조

- System.Data.SqlClient
- Visual Studio의 LINQ to SQL 도구

디자이너에 추가 하려는 개체가 디자이너와 다른 데이터 연결을 사용 합니다.

2020-01-06 • 3 minutes to read • Edit Online

디자이너에 추가하려는 개체가 디자이너에서 현재 사용 중인 데이터 연결이 아닌 다른 데이터 연결을 사용합니다. 디자이너에서 사용 중인 연결을 바꾸시겠습니까?

개체 관계형 디자이너 (**O/R 디자이너**)에 항목을 추가 하는 경우 모든 항목은 하나의 공유 데이터 연결을 사용 합니다. 디자인 화면은 표면의 모든 개체에 대해 단일 연결을 사용 하는 DataContext를 나타냅니다. 디자이너에서 현재 사용 중인 데이터 연결과 다른 데이터 연결을 사용 하는 디자이너에 개체를 추가 하는 경우이 메시지가 나타납니다. 이 오류를 해결하려면 기존 연결 유지를 선택하세요. 이 항목을 선택하면 선택한 개체가 추가되지 않습니다. 또는 개체 추가를 선택하고 DataContext 연결을 새 연결로 다시 설정할 수 있습니다.

연결 옵션

- 기존 연결을 선택한 개체에서 사용 하는 연결로 바꾸려면 **예**를 클릭 합니다.

  선택한 개체는 **O/R 디자이너**에 추가 되 고 *DataContext. 연결은* 새 연결로 설정 됩니다.

  > **NOTE**
  > **예**를 클릭 하면 **O/R 디자이너** 의 모든 엔터티 클래스가 새 연결에 매핑됩니다.

- 기존 연결을 계속 사용 하 고 선택한 개체의 추가를 취소 하려면 **아니요**를 클릭 합니다.

  작업이 취소됩니다. *DataContext.Connection*이 기존 연결로 설정되어 있습니다.

참조

- Visual Studio의 LINQ to SQL 도구

<association name> 연결을 만들 수 없습니다. - 속성이 두 번 나열된 경우

2019-12-02 • 2 minutes to read • Edit Online

<association name> 연결을 만들 수 없습니다. 동일한 속성이 한 번을 초과해 나열되었습니다. <property name>.

연결이 **연결 편집기** 대화 상자에서 선택한 **연결 속성**에 의해 정의되었습니다. 속성은 연결의 각 클래스에 대해 한 번만 나열될 수 있습니다.

메시지의 속성은 부모 또는 자식 클래스의 **연결 속성**에 한 번을 초과해 나타납니다.

이 문제를 해결하려면

- 메시지를 검사하여 메시지에 지정된 속성을 기록합니다.
- **확인**을 클릭하여 메시지 상자를 닫습니다.
- **연결 속성**을 검사하여 중복된 엔트리를 제거합니다.
- **확인**을 클릭합니다.

참조

- Visual Studio의 LINQ to SQL 도구
- 방법: LINQ to SQL 클래스 간의 연결 만들기 (O/R 디자이너)

데이터베이스 개체 <object name>의 스키마 정보를 검색할 수 없음

2020-01-06 • 2 minutes to read • Edit Online

이 메시지는 일반적으로 **서버 탐색기 Explorer** 또는 **데이터베이스 탐색기**에서 개체를 클립보드에 복사하고 데이터베이스에서 삭제한 다음, 디자이너에 붙여넣은 경우에 나타납니다. 즉, 데이터베이스 개체가 더 이상 존재하지 않으므로 이 메시지가 나타납니다.

참조

- Visual Studio의 LINQ to SQL 도구

선택한 데이터베이스 개체 중 하나 이상이 대상 클래스의 스키마와 일치하지 않는 스키마를 반환합니다.

2020-01-16 • 2 minutes to read • Edit Online

선택한 하나 이상의 데이터베이스 개체에서 대상 데이터 클래스 스키마와 일치하지 않는 스키마를 반환합니다. 디자이너에 아무 것도 추가되지 않았습니다.

데이터베이스 개체를 기존 엔터티 클래스로 끌어 오면 데이터베이스 개체에서 반환된 데이터는 대상 엔터티 클래스의 스키마와 일치해야 합니다. 올바른 데이터베이스 개체를 선택했는지, 그리고 올바른 엔터티 클래스를 대상으로 했는지 확인하세요.

이 오류를 해결하려면

1. **확인**을 클릭하여 대화 상자를 닫습니다.
2. **O/R 디자이너**에서 데이터베이스 개체를 끌어 놓는 대상 클래스의 스키마와 일치하는 데이터를 반환하는 데이터베이스 개체를 선택합니다.

참조

- Visual Studio의 LINQ to SQL 도구