# **№ 10 Создание WPF приложения управления базой данных**

## Задание

1. Установите и настройте MS SQL Server.
2. Создайте новую базу данных, а также необходимую инфраструктуру (таблицы, связи межу таблицами, триггеры, хранимые процедуры и т.д.) на основе вариантов в лаб. 2 (желательно 3-4 таблицы, минимум должно **быть 2 таблицы**). Одно из полей БД должно быть графическая информация (фото, логотип, карта проезда и т.п.).
3. создать приложение выполняющего CRUD функции баз данных на основе ADO.NET.
4. Обеспечить следующий функционал:
   1. добавление, редактирование, удаление и просмотр записей из базы данных;
   2. сортировка;
   3. Выполнение запросов, запросов с параметрами и хранимых процедур.
5. При работе с базой данных используйте транзакции.
6. Отображайте информацию в виде таблицы (можно DataGrid). Должны быть кнопки навигации (🡪 🡨 и т.п.). Обязательно использовать привязки данных к ЭУ.
7. Сохраните строку подключения и другие параметры работы приложения в конфигурационном файле (например, App.config).
8. База данных может быть любая реляционная (MS SQL, MySQL, SQLite и т.п., **Access – нельзя**).
9. В случае если после запуска приложения не удается найти БД по указанному пути, она должна создаваться во время работы программы (скрипты создания).

***P.S.*** *По согласованию с преподавателем, для разработки можно взять базу данных из курсового проекта (если такая имеется)*

## Вопросы

1. Что такое Binding в WPF? Какие атрибуты мы можем указывать и для чего?
2. Что такое ADO.NET? Какие существуют режимы работы?
3. Объясните назначение класса SqlConnection.
4. Объясните назначение класса SqlDataAdapter
5. Объясните назначение класса SqlException. Чем данный класс отключается от класса-родителя Exception?
6. Объясните назначение класса SqlCommand.
7. В чем разница между методами ExecuteNonQuery, ExecuteReader и ExecuteScalar?
8. Объясните назначение класса SqlDataReader.
9. Объясните назначение класса SqlParameter.
10. Объясните назначение класса SqlTransaction.
11. Объясните назначение классов DataSet, DataTable, DataColumn, DataRow, DataRelation.

## Краткие теоретические сведения

### Обзор объектов ADO.NET

*Источник данных DataSet*

Основным объектом ADO является источник данных, представленный объектом DataSet. DataSet состоит из объектов типа DataTable и объектов DataRelation. В коде к ним можно обращаться как к свойствам объекта DataSet, то есть, используя точечную нотацию. Свойство Tables возвращает объект типа DataTableCollection, который содержит все объекты DataTable используемой базы данных.

*Таблицы и поля (объекты DataTable и DataColumn)*

Объекты DataTable используются для представления одной из таблиц базы данных в DataSet. В свою очередь, DataTable составляется из объектов DataColumn.

DataColumn - это блок для создания схемы DataTable. Каждый объект DataColumn имеет свойство DataType, которое определяет тип данных, содержащихся в каждом объекте DataColumn. Например, можно ограничить тип данных до целых, строковых и десятичных чисел. Поскольку данные, содержащиеся в DataTable, обычно переносятся обратно в исходный источник данных, необходимо согласовывать тип данных с источником.

*Объекты DataRelation*

Объект DataSet имеет также свойство Relations, возвращающее коллекцию DataRelationCollection, которая в свою очередь состоит из объектов DataRelation. Каждый объект DataRelation выражает отношение между двумя таблицами (сами таблицы связаны по какому-либо полю (столбцу)). Следовательно, эта связь осуществляется через объект DataColumn.

*Строки (объект DataRow)*

Коллекция Rows объекта DataTable возвращает набор строк (записей) заданной таблицы. Эта коллекция используется для изучения результатов запроса к базе данных. Мы можем обращаться к записям таблицы как к элементам простого массива.

*DataAdapter*

DataSet - это специализированный объект, содержащий образ базы данных. Для осуществления взаимодействия между DataSet и собственно источником данных используется объект типа DataAdapter. Само название этого объекта - адаптер, преобразователь, - указывает на его природу. DataAdapter содержит метод Fill() для обновления данных из базы и заполнения DataSet.

*Объекты DBConnection и DBCommand*

Объект DBConnection осуществляет связь с источником данных. Эта связь может быть одновременно использована несколькими командными объектами. Объект DBCommand позволяет послать базе данных команду (как правило, команду SQL или хранимую процедуру). Объекты DBConnection и DBCommand иногда создаются неявно в момент создания объекта DataSet, но их также можно создавать явным образом.

Используйте примеры:

<https://msdn.microsoft.com/ru-ru/library/h0y4a0f6.aspx>

1. **Binding в WPF**:
   * **Binding** в WPF (Windows Presentation Foundation) позволяет связывать данные между источником данных и пользовательским интерфейсом. Это позволяет автоматически обновлять данные в пользовательском интерфейсе при их изменении в источнике данных. Атрибуты, которые можно указывать при использовании Binding, включают **Path** (путь к свойству источника данных), **Mode** (режим изменения данных, например, односторонний или двусторонний), **Source** (источник данных), **UpdateSourceTrigger** (условия обновления данных), **Converter** (конвертер значений), **FallbackValue** (значение по умолчанию при неудачном связывании) и другие.
2. **ADO.NET**:
   * **ADO.NET** (ActiveX Data Objects .NET) — это набор библиотек и классов в .NET Framework, предназначенных для работы с данными. Режимы работы в ADO.NET включают соединенный режим (connected mode), в котором приложение подключается к базе данных напрямую, и разъединенный режим (disconnected mode), в котором данные извлекаются из базы данных, но соединение с базой данных закрывается после получения данных.
3. **SqlConnection**:
   * Класс **SqlConnection** представляет подключение к базе данных SQL Server в ADO.NET. Он используется для открытия соединения с конкретной базой данных на сервере SQL.
4. **SqlDataAdapter**:
   * Класс **SqlDataAdapter** в ADO.NET представляет собой набор данных SQL, который используется для заполнения объектов DataSet и обновления базы данных SQL с использованием команд SQL.
5. **SqlException**:
   * **SqlException** — это класс исключений, который обрабатывает ошибки, возникающие при выполнении операций с базой данных SQL. Отличие SqlException от класса-родителя Exception заключается в том, что SqlException содержит дополнительные свойства, специфичные для ошибок SQL Server, такие как код ошибки SQL.
6. **SqlCommand**:
   * **SqlCommand** в ADO.NET представляет SQL-запрос или хранимую процедуру, которая будет выполнена на базе данных. Он используется для выполнения SQL-команд на подключенной базе данных.
7. **ExecuteNonQuery, ExecuteReader и ExecuteScalar**:
   * **ExecuteNonQuery** используется для выполнения команд SQL, которые не возвращают данные (например, INSERT, UPDATE, DELETE).
   * **ExecuteReader** используется для выполнения команд SQL, которые возвращают результаты (например, SELECT) в виде объекта SqlDataReader.
   * **ExecuteScalar** используется для выполнения команд SQL, которые возвращают одиночное значение (например, агрегатные функции) и возвращает первую колонку первой строки результата запроса.
8. **SqlDataReader**:
   * **SqlDataReader** представляет поток данных, полученных в результате выполнения SQL-запроса с помощью метода ExecuteReader. Он позволяет последовательно считывать результаты запроса.
9. **SqlParameter**:
   * **SqlParameter** представляет параметр SQL-запроса или хранимой процедуры. Он используется для передачи параметров в SQL-команды и защиты от SQL-инъекций.
10. **SqlTransaction**:

* **SqlTransaction** представляет транзакцию базы данных SQL. Он используется для группировки нескольких операций базы данных в единую транзакцию, чтобы обеспечить целостность данных.

1. **DataSet, DataTable, DataColumn, DataRow, DataRelation**:

* **DataSet** представляет набор данных, включая таблицы, столбцы и отношения между таблицами.
* **DataTable** представляет таблицу внутри DataSet.
* **DataColumn** представляет столбец в DataTable.
* **DataRow** представляет строку данных в DataTable.
* **DataRelation** представляет отношение между таблицами в DataSet.