Введение в ADO.NET

Введение в ADO.NET.

1. Что такое ADO.NET?
2. Исторический экскурс в технологии доступа к данным
3. Сравнительный анализ технологий доступа к данным.
4. Сравнительный анализ понятий драйвер и провайдер
5. Пространства ADO.NET
6. Модели работы ADO.NET
7. Концепция интерфейсов и базовых классов ADO.NET
8. Обзорный пример использования ADO.NET для доступа к источнику данных

ADO.NET (ActiveX Data Objects) - часть платформы .NET Framework, представляющая различные службы для доступа к реляционным данным.

Пользовательские приложения, могут использовать ADO.NET для соединения, обработки и обновления данных в различных источниках.

Исторический экскурс в технологии доступа к данным

ADO.NET

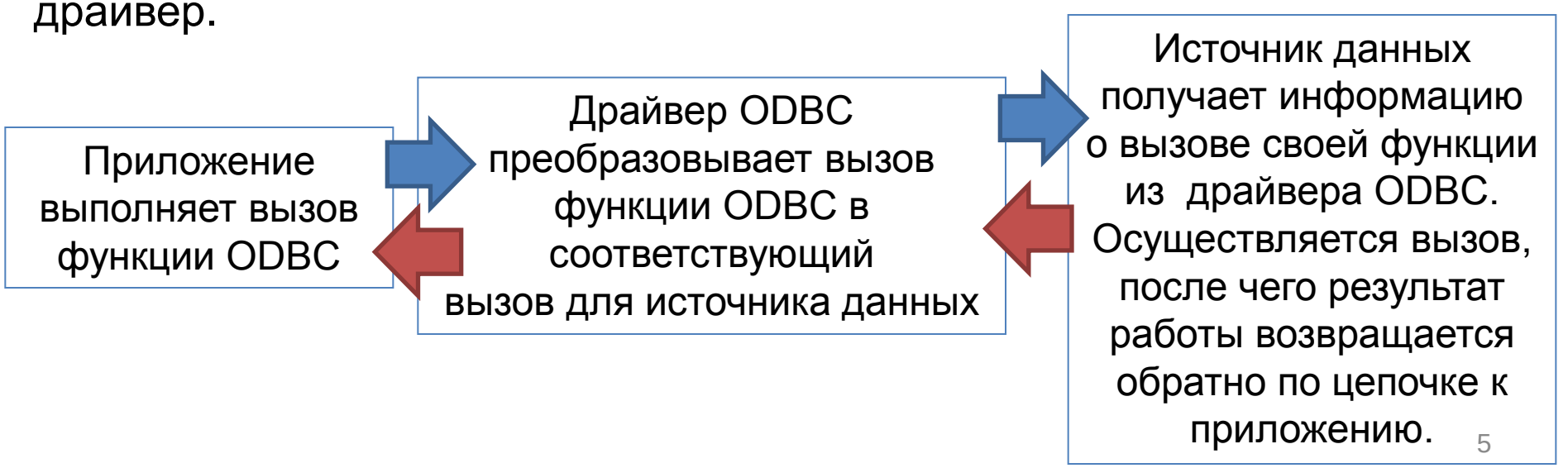
- Существует большой выбор баз данных, которыми можно воспользоваться для реализации потребностей вашего приложения.
- Изначально, каждый производитель баз данных предлагал свой набор функций для работы с ней, причем никакого стандарта этих функций не было.
- Для связи с каждой базой данных использовались разные функции, каждый разработчик тратил большое количество времени на знакомство и погружение в тонкости API конкретной базы данных.
- Естественно такая ситуация не могла устраивать программистов. В силу этого началась разработка некоторой унифицированной технологии для доступа к различным базам данных.
- Первостепенную роль в этом сыграла компания Microsoft.

Исторический экскурс в технологии доступа к данным

ADO.NET

1. **ODBC (Open DataBase Connectivity)** – программный интерфейс (API), который позволяет единообразно обращаться к различным источникам данных не задумываясь о тонкостях и особенностях данного источника.

Важным понятием технологии ODBC является **драйвер**.
Драйвер – набор библиотек для доступа к тому или иному источнику данных ODBC. Например, если вам нужен, доступен через ODBC к СУБД MS SQL Server следует проинсталлировать соответствующий драйвер.



Исторический экскурс в технологии доступа к данным

2. **DAO (Data Access Objects)** - основана на технологии баз данных Microsoft Jet. *Microsoft Jet* - процессор баз данных, предназначенный для Microsoft Access.

Данный процессор является первым объектно-ориентированным интерфейсом для связи с Microsoft Access (в отличие от технологии ODBC, где доступ реализован в процедурном стиле).

DAO обеспечивал доступ и к другим СУБД, например таким как MS SQL Server, Firebird, MySQL и т.д. При возникновении такой ситуации DAO использовал ODBC, т.е. производилось преобразование вызовов DAO в соответствующие вызовы ODBC, что естественно замедляло работу приложения.

Разработка закончилась в 2001 году.

Исторический экскурс в технологии доступа к данным

3. **OLE DB (Object Linking and Embedding Database)** – набор COM интерфейсов, которые позволяют приложениям обращаться к различным источникам данных (как реляционным, так и другим).

Схема взаимодействия OLE DB и приложения схожа с ODBC.

Провайдер – набор объектов, реализующих множество COM интерфейсов для доступа к источнику данных. Например, провайдер для доступа к файловому хранилищу, провайдер для доступа к локальной папке, провайдер для взаимодействия с MySQL.

OLE DB была разработана для использования в рамках C и C++, а также для языков с C-подобными вызовами функций.

Исторический экскурс в технологии доступа к данным

4. **Технология ADO (ActiveX Data Objects)** предоставляет набор объектов для высокоуровневого доступа к источникам данных вне зависимости от их типа.

Технологию ADO могут использовать программисты C++, VB, JScript и другие. Фактически ADO является надстройкой над OLE DB, упрощающей особенности использования OLE DB. Естественно из-за этого есть некоторые потери в скорости взаимодействия по сравнению с чистым OLE DB.

5. **ADO.NET** – технология доступа к данным в рамках платформы .Net Framework. Она содержит набор пространств для доступа к технологиям OLE DB и ODBC, СУБД MS SQL Server и Oracle.

Архитектура ADO.NET

ADO.NET

Двумя основными компонентами ADO.NET является

- **поставщик данных** .NET Framework;
- **автономная модель** хранения данных.

Поставщик данных .NET Framework используется для **соединения** с базой данных, **выполнения команд** и **получение результатов** выполнения команд.

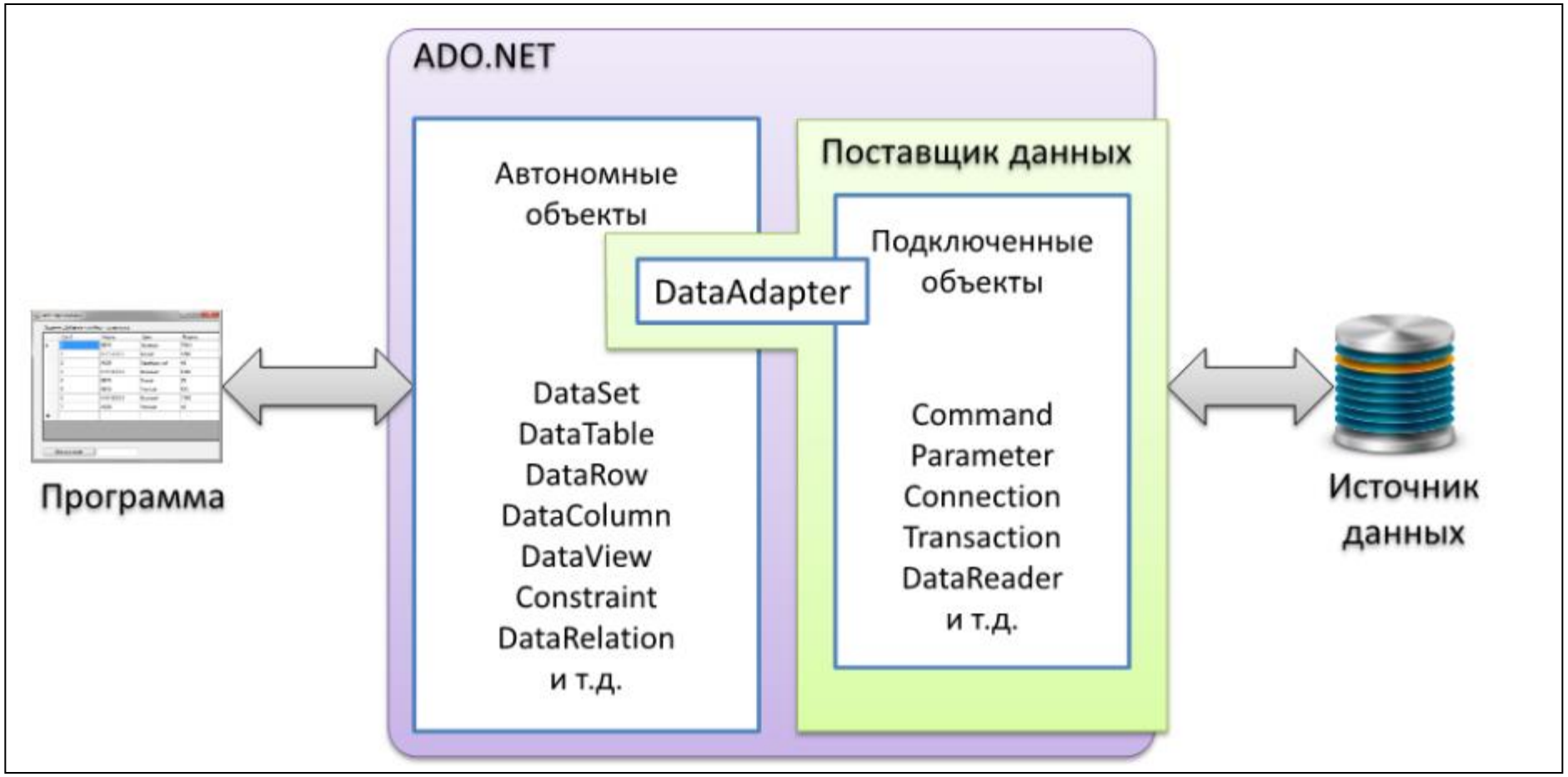
Автономная часть архитектуры, представленная в виде класса DataSet, является расположенным в оперативной памяти кэшем данных, для **хранения результатов, получаемых от поставщика данных**.

Достоинства:

Разделение архитектуры на две независимые части позволяет использовать технологию ADO.NET для построения многоуровневых приложений, а так же для создания приложений, использующих различные источники данных.

Архитектура ADO.NET

ADO.NET



Архитектура ADO.NET

ADO.NET

Особенности ADO.NET

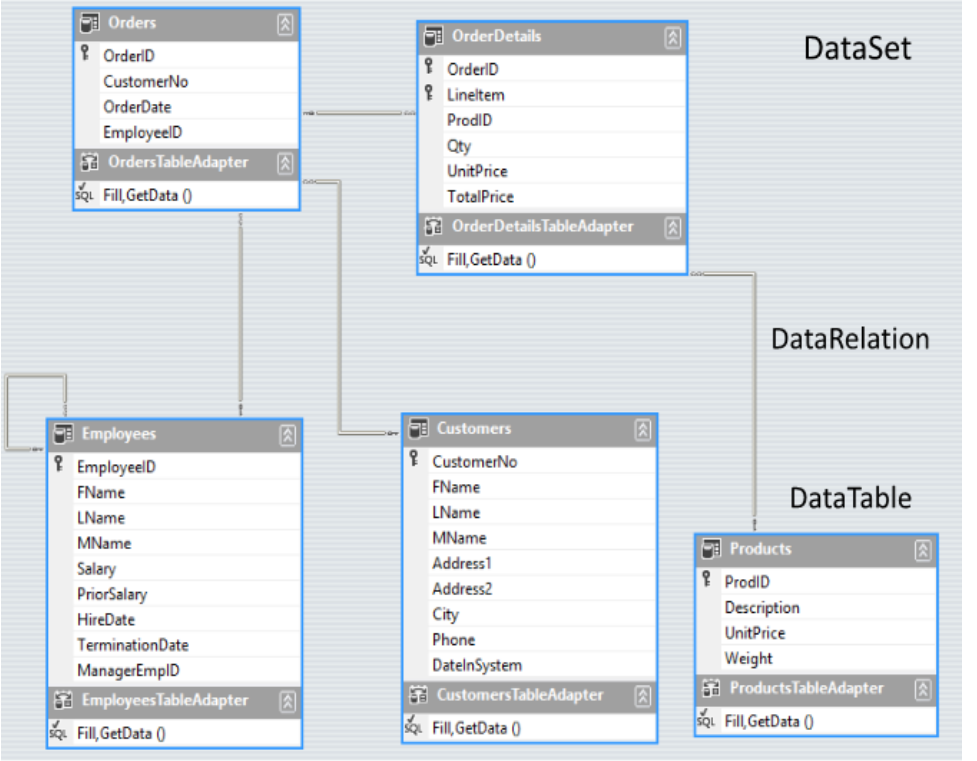
- Все классы, предоставляемые технологией ADO.NET можно отнести к подключаемой части или автономной.
- Единственное исключение - класс **DataAdapter**, который является посредником между подключенной и автономной частями ADO.NET.
- Объект **DataAdapter** представляет собой набор команд для заполнения данными автономной части, а так же для передачи отложенных изменений обратно в источник данных

Автономная часть

ADO.NET

Главным объектом автономной части ADO.NET является объект **DataSet**.

На абстрактном уровне объект **DataSet** является виртуальной базой данных, содержащей в себе таблицы с данными и отношения между таблицами.



Таблицы, содержащиеся в объекте **DataSet** являются экземплярами класса **DataTable**, а связи являются экземплярами класса **DataRelation**.

Автономная часть ADO.NET находится в пространстве имен System.Data.

Поставщики данных .NET Framework

ADO.NET

Поставщик данных .NET Framework используется для установления соединения с базой данных, выполнения команд и получения результатов.

Поставщики .NET Framework для разных источников данных:

- **Поставщик данных SQL Server** – используется для приложений, работающих с базами данных **MS SQL Server**.
- **Поставщик данных OLE DB** – используется для источников данных **Microsoft Access и Microsoft Excel**.
- **Поставщик данных ODBC** – используется для приложений, работающих с источником данных через **ODBC**.
- **Поставщик данных Oracle** – используется для приложений, работающих с **Oracle**

Поставщики данных .NET Framework

ADO.NET

Основные классы подключаемой части ADO.NET.

- **Connection** - класс, позволяющий устанавливать подключение к источнику данных.
- **Transaction** - класс, предоставляющий транзакцию для указанной команды
- **Command** – класс, представляющий исполняемую команду в базовом источнике данных.
- **Parameter** – класс, предоставляющий параметры, для указанной команды.
- **DataReader** – класс, представляющий собой эквивалент конвейерного курсора с возможностью только чтения данных в прямом направлении.

Концепция интерфейсов и базовых классов ADO.NET

ADO.NET

Тип объекта	Базовый класс	Соответствующие интерфейсы	Описание
Connection	**DbConnection**	**IDbConnection**	Позволяет подключаться к хранилищу данных и отключаться от него.
Command	**DbCommand**	**IDbCommand**	Представляет SQL-запрос или хранимую процедуру.
DataReader	**DbDataReader**	**IDataReader, IDataRecord**	Предоставляет доступ к данным, предназначенным только для чтения
DataAdapter	**DbDataAdapter**	**IDataAdapter, IDbDataAdapter**	Передает наборы данных между вызывающим процессом и хранилищем данных. Адаптеры данных содержат подключение и набор из четырех внутренних объектов команд для выборки, вставки, изменения и удаления информации в хранилище данных
Parameter	**DbParameter**	**IDataParameter, IDbDataParameter**	Представляет именованный параметр в параметризированном запросе

Пространства ADO.NET

ADO.NET

Основным пространством имен **ADO.NET** является **System.Data**, которое содержит набор классов и пространств, реализующих работу с **ADO.NET**

System.Data.Common	Содержит базовые абстрактные классы (например, такие как `DbConnection`, `DbCommand`, `DbDataAdapter`, и так далее), которые переопределяются в специфических подпространствах
System.Data.OleDb	Используется для подключения к данным при помощи технологии OLE DB и работе с ними.
System.Data.Odbc	Используется для подключения к данным при помощи технологии ODBC и работы с ними.
System.Data.OracleClient	Используется для работы с базами данных Oracle.
System.Data.SqlClient **System.Data.Sql**	Используются для работы с базами данных SQL Server.

Поставщик данных	Пространство имен	Сборка
OLE DB	`System.Data.OleDb`	`System.Data.dll`
Microsoft SQL Server	`System.Data.SqlClient`	`System.Data.dll`
Microsoft SQL Server Mobile	`System.Data.SqlServerCe`	`System.Data.SqlServerCe.dll`
ODBC	`System.Data.Odbc`	`System.Data.dll`

Модели работы ADO.NET

ADO.NET

Библиотеки ADO.NET можно применять тремя концептуально различными способами:

- в подключенном режиме,
- в автономном режиме
- с помощью технологии Entity Framework.

Подключенный уровень - база явно подключается к соответствующему хранилищу данных и отключается от него. При таком способе использования ADO.NET обычно происходит взаимодействие с хранилищем данных с помощью объектов подключения, объектов команд и объектов чтения данных.

Автономный уровень - позволяет работать с набором объектов, который представляет на стороне клиента **копию внешних данных**. При получении данных с помощью объекта адаптера данных подключение **открывается** и **закрывается автоматически**. Получив данные, вызывающий код может просматривать и обрабатывать **их без затрат на сетевой трафик**. А если нужно занести изменения в хранилище данных, то адаптер данных задействуется для обновления данных — при этом подключение открывается заново для проведения обновлений в базе, а затем сразу же закрывается.

Entity Framework позволяет взаимодействовать с реляционной базой данных с помощью объектов на стороне клиента, которые скрывают конкретные низкоуровневые особенности СУБД.

Пример создания подключения через ODBC

```
OdbcConnection connection = new OdbcConnection();
try
{
 connection.ConnectionString = @"Driver={Microsoft Access Driver
                                        (*.mdb, *.accdb)};Dbq=D:\simple.accdb";
 // Открываем соединение
 connection.Open();
 // Создаём объект для исполнения запроса
 OdbcCommand command = new OdbcCommand("SELECT * FROM PEOPLE", connection);
 // Исполняем запрос и сохраняем ссылку на объект результата
 OdbcDataReader reader = command.ExecuteReader();
 while (reader.Read() != false)
  {
   // Для обращения к столбцу используется индексатор
   // Возможен доступ как по имени столбца так и по индексу
   Console.WriteLine("{0,-10} {1,-10} {2,-10}",
                      reader["id"], reader["firstname"], reader["lastname"]);
   // Console.WriteLine("{0,-10} {1,-10} {2,-10}",
                     reader[0], reader[1], reader[2]);
  }
}
catch (Exception ex)
{  Console.WriteLine(ex.Message);  }
finally
{ connection.Close();              }
```

Пример создания подключения через OLE DB

```
OleDbConnection connection = new OleDbConnection();
try
{
 connection.ConnectionString =
     "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=DB/sample.mdb";
 // Открываем соединение
 connection.Open();
 // Создаём объект для исполнения запроса
 OleDbCommand command = new OleDbCommand("SELECT * FROM PEOPLE", connection);
 // Исполняем запрос и сохраняем ссылку на объект результата
 OleDbDataReader reader = command.ExecuteReader();
 while (reader.Read() != false)
  {
   // Для обращения к столбцу используется индексатор
   // Возможен доступ как по имени столбца так и по индексу
   Console.WriteLine("{0,-10} {1,-10} {2,-10}",
                     reader["id"], reader["firstname"], reader["lastname"]);
   // Console.WriteLine("{0,-10} {1,-10} {2,-10}",
                     reader[0], reader[1], reader[2]);
  }
}
catch (Exception ex)
{ Console.WriteLine(ex.Message);}
finally
{connection.Close();            }
```