Implementing ADO.NET with C#

WHY YOU STILL NEED TO KNOW ADO.NET



Paul D. Sheriff
BUSINESS/IT CONSULTANT PDS CONSULTING
www.pdsa.com psheriff@pdsa.com



Course Goals



Learn why you should know ADO.NET

How to securely connect and interact with databases

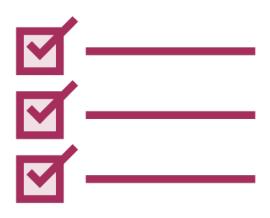
Best ways to retrieve data

Handle exceptions efficiently

Work with disconnected data

Wrapper classes to simply usage of ADO.NET





l assume you...

- Are a .NET programmer
- Are familiar with data access in C#

You want to...

- Understand more about ADO.NET
- Know what ORMs use under the hood
- Simplify ADO.NET usage



Related Pluralsight Course

Enterprise Library Data Access Application Block

By: John Sonmez

Exploring SQL Server Tools and Language Enhancements

By: Leonard Lobel



Modules in This Course





Why You Still Need to Know ADO.NET

- Why ADO.NET is still relevant
- ADO.NET Overview

Connecting and Submitting Queries to a Database

- Creating connections
- Submitting queries
- Transactions





Retrieve Data Quickly using the SqlDataReader

- Looping through cursor
- Create generic List<T> collections
- Using extension methods

Handling ADO.NET Exceptions

- Handling exceptions
- Gather ADO.NET exception info





Disconnected Classes Are Useful

- DataTables and DataViews
- Sorting and Filtering Data
- Multiple result sets

The Building Blocks of the DataTable

- DataRow and DataColumn
- Cloning, copying, selecting rows





Builder Classes Make It Easy to Work with ADO.NET

- Create/break apart connection strings
- Create data modification commands

Making ADO.NET Easy to Use

- Create a data wrapper class
- Simplify your ADO.NET code



Why ADO.NET



Why ADO.NET?

Technology beneath all other data access methods (EF, Dapper, NHibernate, etc.)

Fastest data access method

Don't need to learn other tools or syntax (only C# and SQL)

Simple to understand

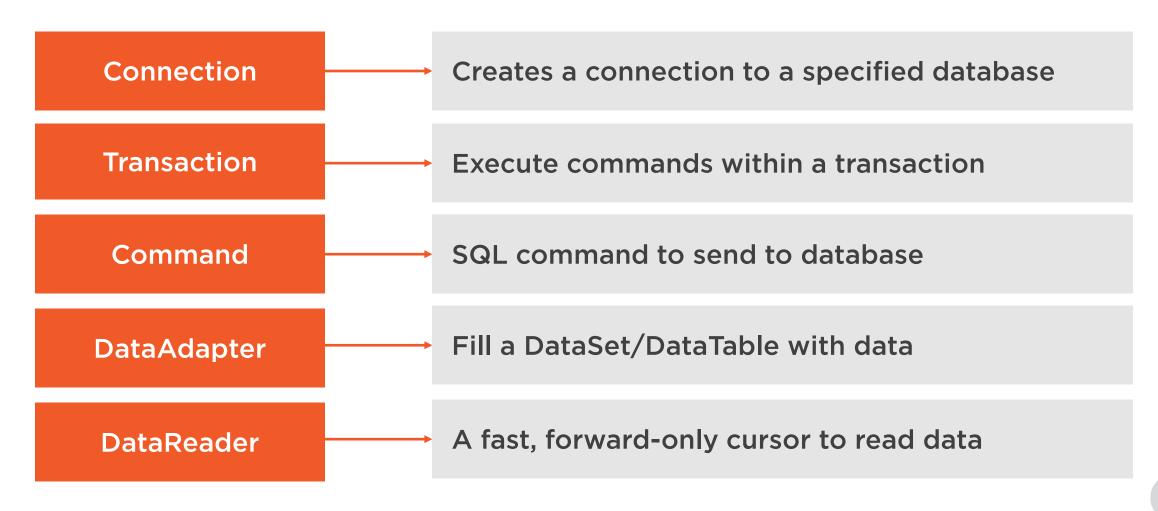
Easy to work with multiple data sources



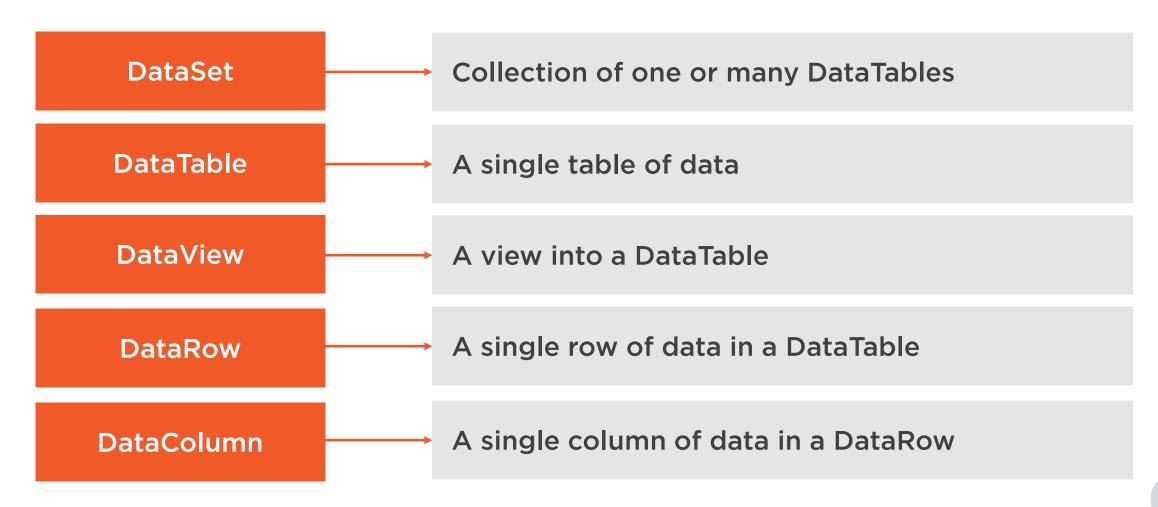
ADO.NET Overview



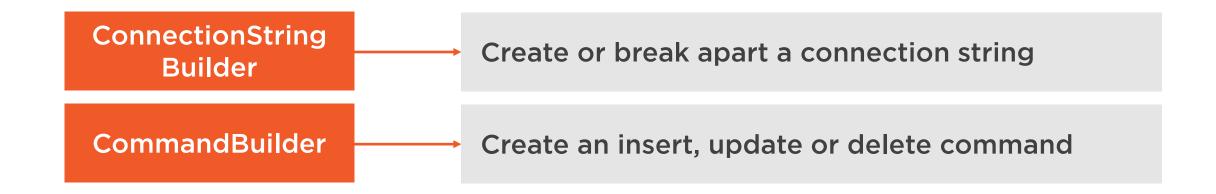
Connected Classes



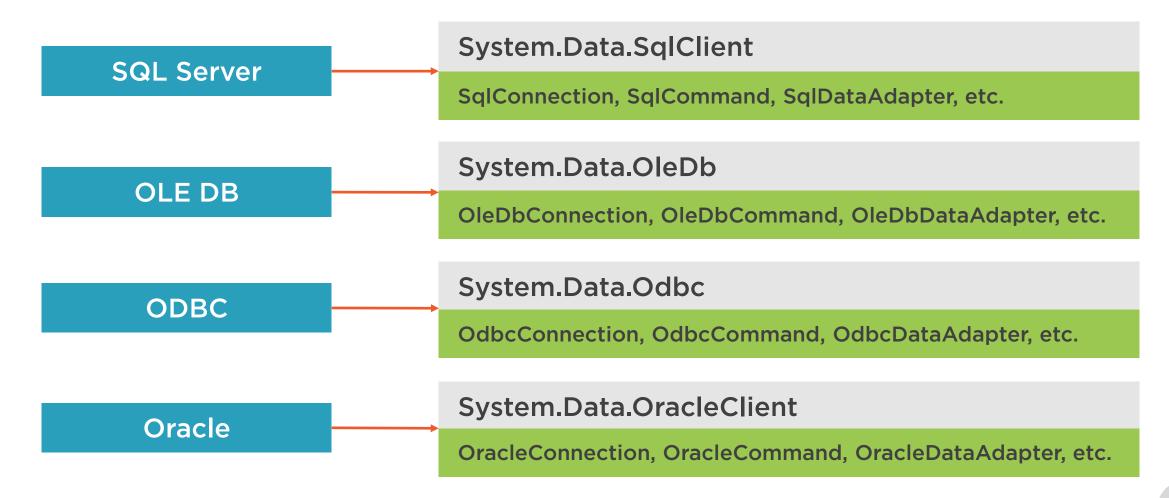
Disconnected Classes



Builder Classes



Providers





ORMs



Object Relational Mapping (ORM)

Examples: EF, Dapper, NHibernate Auto map relational data to objects

Abstracts away the database

Use different language for querying (LINQ)

Often reduces the amount of code you need to write



ORMs Have Benefits

Make development easier

Make development quicker Cut down the amount of code you need to write



ORMs Have Benefits

Automatic mapping of columns to properties

Less SQL to write

Code generation from a database and vice versa



ORMs Have Drawbacks

Slower than ADO.NET

Can send bad SQL (When using LINQ for example)

Can make it hard to call stored procedures



ORMs Have Drawbacks

Sometimes you must learn a new "sub-language" and tooling

Usually heavy configuration and attributes

Can be breaking changes between versions



Demo



ADO.NET in action



Summary



ADO.NET is still very relevant

Supports more data sources than ORMs

Quickest way to process data

Connected and disconnected scenarios





Coming up in the next module...

Creating connections
Submitting queries
Transactions

