Handout: Connecting C# to SQL Server Using Dapper

Introduction

Lecturer: Tim Corey
Title: How to Connect C# to SQL (the easy way)
Source: YouTube Video
Views: 1,088,660 (as of the last update)
Publish Date: February 28, 2017
Duration: Approximately 80 minutes

Objective:
The primary goal is to simplify the process of connecting a C# application to a SQL Server database. The aim is to make data access one of the least concerning aspects of application development.

Data Access Options (00:30)

Various methods can be utilized to establish a connection between a C# application and a SQL Server database:

- **Traditional ADO.NET**: This is a direct connection using C#, but it can be quite complex due to the configuration required.
- **Entity Framework (EF)**: An Object-Relational Mapper (ORM) that automates database connectivity. While powerful, EF can sometimes be a "black box" that obscures the data access process, which may lead to complications, especially in production environments.
- **Dapper**: An object mapper for .NET created by the team behind Stack Overflow. It represents a middle ground between ADO.NET and EF, providing a balance of simplicity and control, which is why it's the preferred choice for this tutorial.

Why Choose Dapper? (01:30)

Dapper offers the following advantages:

- It's a straightforward ORM that doesn't over-complicate data access.
- It allows developers to maintain control over the data access process.
- Being an open-source tool supported by Stack Overflow, it's robust, heavily tested, and suitable for production use.

Setting Up the Project (02:30)

The project setup includes:

- Creating a new Windows Forms Application named `SQLDataAccessDemo`.
- Developing a `Dashboard` form to serve as the interface for data interactions.

Database Overview (03:30)

The sample database comprises a `People` table with the following columns: ID, FirstName, LastName, EmailAddress, PhoneNumber. It also includes stored procedures for data retrieval and insertion.

Building the Data Model (04:30)

The data model is built around a `Person` class that reflects the structure of the `People` table in the database. The class includes properties for ID, FirstName, LastName, EmailAddress, and PhoneNumber.

Connection String Helper (05:30)

A static helper class is created to facilitate retrieving the connection string from `App.config` using `System.Configuration`.

Establishing Database Connection (06:00)

- Dapper is added to the project via NuGet packages.
- An `IDbConnection` is established using Dapper to create a connection to SQL Server.
- The `query` method is used to retrieve data, while the `execute` method is used for data insertion.

Retrieving Data with Dapper (06:30)

A method is developed to get people by last name using a stored procedure, which securely retrieves data and maps it to the `Person` class, returning the data as a list.

Displaying Retrieved Data (07:00)

The retrieved list of people is bound to a `ListBox` control on the form. The interface allows users to search by last name and displays the results in the list.

Inserting Data into the Database (07:30)

A new method is created to insert a person into the database. User input from the form is mapped to a new `Person` instance, and a stored procedure executes the insert operation.

User Interface for Data Insertion (08:00)

The UI includes text fields for FirstName, LastName, EmailAddress, and PhoneNumber. A button triggers the insertion method, and the input fields are cleared after successful data insertion.

Verifying Data Insertion (08:30)

The process of adding new records is demonstrated by searching for the inserted last names and observing the updated list to confirm successful inserts.

Best Practices and Recommendations (09:00)

Best practices include:

- Preferring stored procedures over raw SQL queries for security and maintainability.
- Using the `Execute` method for operations that do not return data.
- Encapsulating data access logic within a separate class library for a cleaner codebase.

Why Avoid Entity Framework and Direct ADO.NET (09:30)

- Entity Framework can be complex and may cause issues in production.
- Direct ADO.NET is inherently complex and prone to misconfiguration.

Additional Resources (10:00)

For further exploration and advanced usage:

- Tim Corey's blog for source code and further reading: IAmTimCorey.com
- Online database connection strings reference: ConnectionStrings.com
- Dapper documentation for custom mappings and advanced features.

Concluding Thoughts (10:30)

The tutorial underscores the ease and power of Dapper in making data access straightforward. Viewers are encouraged to practice and delve deeper into Dapper's capabilities.

Contact and Community Engagement (11:00)

Tim Corey invites viewers to:

- Join his mailing list for exclusive content and discounts.
- Engage with the community by asking questions and sharing thoughts in the comments.

Next Steps (11:30)

A preview of an upcoming video series on C# development is provided. Viewers are encouraged to subscribe to the channel and join the mailing list for the latest updates.

[Note: The handout above follows the structure and content of Tim Corey's video on connecting C# to SQL Server using Dapper. The notes serve as an outline for a comprehensive handout to accompany the video tutorial.]