Data Access using C#

Visual Studio has computer software components such as ADO and LINQ that can be used by programmers to access data and data services. Learn to use these tools to gain convenient access to each field through strongly-typed properties. Topics: Data Providers, Connections, Commands, Stored Procedures, DataReader, DataSet, Entity Data Model, LINQ to SQL, Entity Framework, LINQ to Entities, and ADO.NET Data Service.

Who should take this course?

This course is designed for developers who know C# and are eager to build data-aware applications using Visual Studio 2010.

Course Objectives

• Use the ADO.NET Object Model to connect to data sources to query and update data.
• Apply LINQ operators and the LINQ API to interact with numerous types of data.
• Implement the Entity Framework and the various modeling strategies used to work with relational data.
• Create WCF Data Services that use the OData Protocol to expose and consume data over the Web or Intranet.

Course Details

• Length: 24 hours
• Format: Classroom
• Prerequisites: SQL: Structured Query Language, C# Programming : 2, Visual Studio experience

The above prerequisites are considered to be the basic skills and knowledge needed prior to taking this class. Instructors will assume your readiness for the class materials and will NOT use class time to discuss prerequisite materials.
Course Contents

Use the ADO.NET Object Model to connect to data sources to query and update data.

- Create connections to access structured and unstructured types of data in both connected and disconnected data sources.
- Apply authentication controls against the data source to establish the identity of the user or process being authenticated.
- Apply data provider specific adapters and commands to implement ADO.NET data providers and data consumers.
- Apply exception handling techniques through provider specific exceptions to control errors in a generic manner.
- Use ADO.NET objects to perform tasks using DDL, DCL, and DML.
- Describe connection pooling and its role in enhancing the performance of executing commands on a database.
- Implement transaction management facilities whenever a client application performs an update, insert, or delete operation on a data source.
- Implement built in change and state tracking mechanisms to control concurrency and resolve data conflicts.

Apply LINQ operators and the LINQ API to interact with numerous types of data.

- Examine the LINQ architecture and language features.
- Apply LINQ queries and updates to ADO.NET DataSets, XML, SQL and Entities through the use of the LINQ API and library of operators.
- Use implicitly typed variables, object initializers, anonymous types, extension methods and lambda expressions to formulate various types of LINQ queries.
- Handle exceptions that might be thrown from inside a LINQ query or update.
Course Contents, continued

Implement the Entity Framework and the various modeling strategies used to work with relational data.

- Describe the architecture used in the development of data-oriented software applications.
- Describe object relational mapping, Entity Data Models, Object Services and the data providers used to support various data sources.
- Implement the Entity Framework development approaches of database first, model first and code first.
- Apply the various ways of querying and updating in the Entity Framework through the use of Entity SQL, Linq to Entities and Native SQL.
- Use self tracking entities to develop N-Tier applications.
- Utilize data annotations and attributes to define data auditing controls and concurrency resolution through State management.

Create WCF Data Services that use the OData Protocol to expose and consume data over the Web or Intranet.

- Describe the purpose and features of WCF Data Services in exposing and retrieving data.
- Use the semantics of representational state transfer (REST) to access and update data from a variety of sources.
- Utilize the OData protocol by applying and building upon web technologies such as HTTP, Atom Publishing Protocol ( AtomPub) and JSON.
- Implement interceptors to apply custom logic to an operation involving requests and updates of data.
- Handle exceptions that may be thrown during an HTTP response service operation.
- Apply security considerations that are specific to developing, deploying, and running WCF Data Services to grant and restrict access to resources.