

# Kurs-Dokumentation



**Zentrum für Informatik ZFI AG**

**Developing Data Access Solutions with MS  
Visual Studio 2010 (NDAS-0511) -IT Ausbildung  
nach Mass**

<http://www.zfi.ch/NDAS-0511>

Weitere Infos finden Sie unter [www.zfi.ch](http://www.zfi.ch) oder via Adresse:

**Zentrum für Informatik ZFI AG  
Zentralsekretariat  
Technoparkstrasse 1  
CH-8005 Zürich  
Telefon: 044 732 40 00  
Telefax: 044 732 40 09**

**Zürich, Basel, Bern, Zürich, Schweiz**

<b>Titel</b>	<b>Developing Data Access Solutions with MS Visual Studio 2010</b>
<b>Untertitel</b>	<b>effizienter Datenzugriff für .NET-Entwickler mit ADO.NET, LINQ und WCF</b>
<b>Einleitung</b>	<p>Die traditionelle Datenverarbeitung basierte primär auf einem verbindungs-basierten Modell mit zwei Ebenen. Da für die Datenverarbeitung immer mehr auf Architekturen mit mehreren Ebenen zurückgegriffen wird, wird verstärkt mit nicht verbundenen Lösungen gearbeitet, um eine bessere Skalierbarkeit der Anwendungen zu erzielen. ADO.NET stellt konsistenten Zugriff auf Datenquellen wie SQL Server und XML sowie auf Datenquellen bereit, die durch OLE DB und ODBC verfügbar gemacht werden. Verbraucheranwendungen mit Datenfreigabe können mit ADO.NET eine Verbindung mit diesen Datenquellen herstellen und die enthaltenen Daten abrufen, verarbeiten und aktualisieren. ADO.NET trennt den Datenzugriff von der Datenbearbeitung in einzelne Komponenten, die separat oder zusammen verwendet werden können. ADO.NET schliesst .NET Framework-Datenanbieter zum Verbinden mit einer Datenbank, zum Ausführen von Befehlen und zum Abrufen von Ergebnissen ein. Diese Ergebnisse werden entweder direkt verarbeitet oder in einem ADO.NET-DataSet-Objekt platziert, um sie dem Benutzer, kombiniert mit Daten aus mehreren Quellen, bei Bedarf verfügbar zu machen oder um sie an eine andere Ebene zu übergeben. Das DataSet-Objekt kann auch unabhängig von einem .NET Framework-Datenanbieter verwendet werden, um Daten zu verwalten, die für die Anwendung lokal sind oder aus einer XML-Datenquelle beschafft werden. Die ADO.NET-Klassen befinden sich in System.Data.dll und werden in die in System.Xml.dll vorhandenen XML-Klassen integriert. ADO.NET enthält Funktionen für Entwickler von verwaltetem Code. Diese Funktionen sind mit den ADO-Funktionen (ActiveX Data Objects) vergleichbar, die von COM-Entwicklern (Component Object Model) verwendet werden. Wir empfehlen, für den Zugriff auf Daten in Ihren .NET-Anwendungen statt ADO ADO.NET zu verwenden. Unter den heutigen Bedingungen müssen viele Entwickler von Geschäftsanwendungen mit zwei (oder mehr) Programmiersprachen arbeiten: mit einer allgemeinen Programmiersprache für die Geschäftslogik- und die Darstellungsschicht (wie Visual C# oder Visual Basic) und einer Abfragesprache für die Interaktion mit der Datenbank (z. B. Transact-SQL). Der Entwickler muss also mehrerer Sprachen mächtig sein, um seine Arbeit effektiv erledigen zu können. Ausserdem sind dadurch Sprachkonflikte in der Entwicklungsumgebung vorprogrammiert. So ergibt es sich z. B., dass eine Anwendung, die zur Ausführung einer Abfrage von Daten aus einer Datenbank eine Datenzugriffs-API verwendet, die Abfrage als Zeichenfolgenliteral angibt, indem sie Anführungszeichen verwendet. Diese Abfrage ist jedoch für den Compiler nicht lesbar und wird nicht auf Fehler (Syntaxfehler, tatsächliche Existenz der Spalten oder Zeilen, auf die verwiesen wird, usw.) geprüft. Auch der Typ der Abfrageparameter wird nicht geprüft, und es gibt keine IntelliSense-Unterstützung. In der Sprachintegrierten Abfrage (Language-Integrated Query, LINQ) können Entwickler in ihrem Anwendungscode mengenbasierte Abfragen unterbringen, ohne dazu auf eine separate Abfragesprache zurückgreifen zu müssen. Sie können LINQ-Abfragen für die verschiedensten aufzählbaren Datenquellen (also</p>

Datenquellen, die die IEnumerable-Schnittstelle implementieren) schreiben, wie Datenstrukturen, XML-Dokumente, SQL-Datenbanken und DataSet-Objekte im Arbeitsspeicher. Auch wenn diese aufzählbaren Datenquellen auf unterschiedliche Art und Weise implementiert sind, weisen sie doch alle dieselben Syntax- und Sprachkonstrukte auf. Da Abfragen direkt in der Programmiersprache formuliert werden können, benötigen Sie keine andere Abfragesprache, mit der Abfragen als Zeichenfolgenliterals eingebettet werden, die vom Compiler weder gelesen noch geprüft werden können. Durch die Integration von Abfragen in die Programmiersprache werden Visual Studio-Programmierer auch in die Lage versetzt, durch Typ- und Syntaxprüfungen während der Kompilierung sowie IntelliSense-Unterstützung produktiver zu arbeiten. Mit diesen Funktionen wird der für die Beseitigung von Abfragefehlern erforderliche Aufwand beträchtlich reduziert. Das Übertragen von Daten aus SQL-Tabellen in Objekte im Arbeitsspeicher ist häufig nervenaufreibend und fehleranfällig. Der von LINQ to DataSet und LINQ to SQL implementierte LINQ-Anbieter konvertiert die Quelldaten in IEnumerable-basierte Objektaufstellungen. Dem Programmierer werden die Daten stets als IEnumerable-Auflistung angezeigt, gleich ob bei einer Abfrage oder bei einem Update. Für das Schreiben von Abfragen für diese Auflistungen steht uneingeschränkte IntelliSense-Unterstützung zur Verfügung. Es gibt drei separate ADO.NET-Sprachintegrierte Abfrage (Language-Integrated Query, LINQ)-Technologien: LINQ to DataSet, LINQ to SQL und LINQ to Entities. LINQ to DataSet ermöglicht umfangreichere, optimierte Abfragen der DataSet-Daten, mit LINQ to SQL können Sie SQL Server-Datenbankschemas direkt abfragen, und mit LINQ to Entities können Sie ein Entity Data Model abfragen. Dieser ZFI/Microsoft-Kurs wird von einem erfahrenen Microsoft-zertifizierten .NET-Entwickler geleitet.

#### Ihr Nutzen

After completing this course, students will be able to:

- Evaluate a variety of business cases, and then select an appropriate combination of data access technologies and tools most appropriate to each case. Describe the roles of Entity Framework, WCF Data Services, and ADO.NET for building and maintaining applications. Use LINQ on top of these technologies to improve productivity and the quality of their applications.
- Use the tools provided with the Entity Framework to map the conceptual model used by the business logic of an application to the logical data model provided by a database.
- Query an Entity Data Model (EDM) by using common methods such as LINQ to Entities, Entity SQL, and the classes in the EntityClient namespace.
- Perform data modification tasks on data in an EDM.
- Explain the function of the Object Services model implemented by the Entity Framework and the support provided by the Object Services API for addressing the issues faced by enterprise applications that have to handle multiple concurrent users simultaneously accessing the same data .
- Describe best practices for designing and building a scalable, optimized data access layer by using Object Services.

- Customize and extend entities with their own business logic and use advanced mappings to shape the data model to their business and application requirements.
- Reuse existing business classes in a data access layer built by using the Entity Framework.
- Address the architectural issues that can arise when building an n-tier enterprise application by using the Entity Framework.
- Build extensible solutions that can update data in an n-tier enterprise application by using the Entity Framework.
- Access offline data or data that has limited availability in client applications.
- Design, develop, and consume a simple WCF Data Service.
- Use WCF Data Services to update and delete data and to handle multi-user concerns.
- Develop high performance, scalable ADO.NET applications that can query and update data.
- Explain how LINQ to SQL enables development against a logical model which abstracts the low-level details of querying ADO.NET tables and result sets.

#### Voraussetzungen

Before attending this course, students must have: An understanding of the problem-solving techniques that apply to software development, including the following principles of software development: Modern software development models Typical phases of a software development lifecycle Concepts of event-driven programming Concepts of object-oriented programming Creating use-case diagrams Designing and building a user interface Developing a structured application

A basic understanding of the following scripting techniques and some hands-on experience writing scripts: Web scripting techniques Macro scripting techniques Windows scripting techniques

A general understanding of the purpose, function, and features of following .NET Framework topics: Common Language Runtime .NET Framework class library Common Type System Component interoperation Cross-language interoperability Assemblies in the Common Language Runtime Application domains Runtime hosts supported by the .NET Framework

Experience using Visual Studio 2008 in the following task areas: Declaring and initializing typed variables using the Camel case naming convention Using arithmetic, relational, and logical operators in code statements Using branching statements to control code execution Using looping statements to iterate through collections or repeat steps until a specified condition is met Creating classes and methods to establish the basic structure of an application Using methods and events to implement the programming logic of an application Identifying syntax and logic errors Accessing and managing data from a data source

Experience in object oriented design and development as follows: Creating and accessing classes and class properties Creating

and accessing methods and overloaded methods  
 Implementing inheritance, base classes, and abstract classes  
 Declaring, raising, and handling events  
 Responding to and throwing exceptions  
 Implementing interfaces and polymorphism  
 Implementing shared and static members  
 Implementing generics  
 Creating components and class libraries

Experience in N-Tier application design and development as follows:  
 Managing a software development process  
 Controlling input at the user interface level in Windows client and Web applications  
 Debugging, tracing, and profiling .NET applications  
 Monitoring and logging .NET applications  
 Implementing basic testing best practices  
 Performing basic data access tasks with LINQ  
 Basics of LINQ to XML  
 Basics of LINQ to Entities  
 Basics of LINQ to SQL

Implementing basic security best practices in .NET Applications  
 Basics of Code Access Security  
 Basics of Role-Based Security  
 Basics of Cryptography Services

Implementing basic service calls  
 Basics of creating and consuming XML Web Services  
 Basics of creating and consuming WCF Services

Using .NET Configuration Files  
 Deploying .NET Framework Applications using ClickOnce and the MS Installer

Data access experience in Windows client application development as follows:  
 Connect to a data source  
 Implement data binding  
 Implement data validation at the UI layer

Data access experience in Web application development as follows:  
 Connect to a data source  
 Implement dynamic data  
 Implement data validation at the UI layer

#### Teilnehmerkreis

This course is intended for professional .NET software developers who use Microsoft Visual Studio in a team-based, medium-sized to large development environment. They will have experience implementing data access and data binding within their Web and/or Windows client applications and are interested in learning to optimize data access code in their applications by using the Entity Framework, LINQ, and ADO.NET. Members of the audience are experienced users of Microsoft Visual Studio 2008 SP1 or newer releases of the Visual Studio product. The audience has some experience using Visual Studio 2010 for either Windows client or Web application development. Typically, this audience has the following knowledge/experience:

- Experience developing n-tier applications that access various data sources
- Experience implementing data binding within their applications
- Some experience using LINQ and ADO.NET

Unterlagen Folgekurse	<ul style="list-style-type: none"> <li>- A conceptual understanding of the Entity Framework</li> </ul> <p><b>Original Microsoft Unterlagen</b></p> <ul style="list-style-type: none"> <li>- andere Kurse aus dem Bereich Spezialisierung, siehe Bildungsweg-«Application Design for Developers», NADD/10552</li> </ul>
Inhalt	<ul style="list-style-type: none"> <li>- Architecture and Data Access Technologies</li> <li>- Data Access Technologies</li> <li>- Data Access Scenarios</li> </ul> <ul style="list-style-type: none"> <li>- Building Entity Data Models</li> <li>- Introduction to Entity Data Models</li> <li>- Modifying the Entity Data Model</li> <li>- Customizing the Entity Data Model</li> </ul> <ul style="list-style-type: none"> <li>- Querying Entity Data</li> <li>- Retrieving Data by Using LINQ to Entities</li> <li>- Retrieving Data by Using Entity SQL</li> <li>- Retrieving Data by Using EntityClient Provider</li> <li>- Retrieving Data by Using Stored Procedures</li> <li>- Unit Testing Your Data Access Code</li> </ul> <p><b>Creating, Updating, and Deleting Entity Data</b></p> <p><b>Understanding Change Tracking in the Entity Framework</b></p> <p><b>Modifying Data in an Entity Data Model</b></p> <p><b>Handling Multi-User Scenarios by Using Object Services</b></p> <p><b>Handling Concurrency in the Entity Framework</b></p> <p><b>Transactional Support in the Entity Framework</b></p> <p><b>Building Optimized Solutions by Using Object Services</b></p> <p><b>The Stages of Query Execution</b></p> <p><b>Change Tracking and Object Materialization</b></p> <p><b>Using Compiled Queries</b></p> <p><b>Using Design-Time Generated Entity Framework Views</b></p> <p><b>Monitoring Performance</b></p> <p><b>Performing Asynchronous Data Modifications</b></p> <p><b>Customizing Entities and Building Custom Entity Classes</b></p> <p><b>Overriding Generated Classes</b></p> <p><b>Using Templates to Customize Entities</b></p> <p><b>Creating and Using Custom Entity Classes</b></p> <p><b>Using POCO Classes with the Entity Framework</b></p> <p><b>Requirements for POCO Classes</b></p> <p><b>POCO Classes and Lazy Loading</b></p> <p><b>POCO Classes and Change Tracking</b></p> <p><b>Extending Entity Types</b></p> <p><b>Building an N-Tier Solution by Using the Entity Framework</b></p> <p><b>Designing an N-Tier Solution</b></p> <p><b>Defining Operations and Implementing Data Transport Structures</b></p> <p><b>Protecting Data and Operations</b></p> <p><b>Handling Updates in an N-Tier Solution by Using the Entity Framework</b></p> <p><b>Tracking Entities and Persisting Changes</b></p> <p><b>Managing Exceptions in an N-Tier Solution</b></p> <p><b>Building Occasionally Connected Solutions</b></p> <p><b>Offline Data Caching by Using XML</b></p> <p><b>Using the Sync Framework</b></p>

**Querying Data by Using WCF Data Services**  
Introduction to WCF Data Services  
Creating a WCF Data Service  
Consuming a WCF Data Service  
Protecting Data and Operations in a WCF Data Service

**Updating Data by Using WCF Data Services**  
Creating, Updating, and Deleting Data in a WCF Data Service  
Preventing Unauthorized Updates and Improving Performance  
Using WCF Data Services with Nonrelational Data

**Using ADO.NET**  
Retrieving and Modifying Data by Using ADO.NET Commands  
Retrieving and Modifying Data by Using DataSets  
Managing Transactions and Concurrency in Multiuser Scenarios

**Using LINQ to SQL**  
Implementing a Logical Data Model by Using LINQ to SQL  
Managing Performance and Handling Concurrency

**Beitrag**

**Der Teilnehmerbeitrag versteht sich rein netto. Das ZFI ist (gemäss MwSt-Gesetz) nicht Mehrwertsteuerpflichtig und erhebt somit keine MwSt. Bei länger als einen Monat dauernden Lehrgängen ist die Zahlung des Teilnehmerbeitrages in mehreren Raten möglich (pro rata temporis).**

# Bildungsweg Visual Studio .NET

