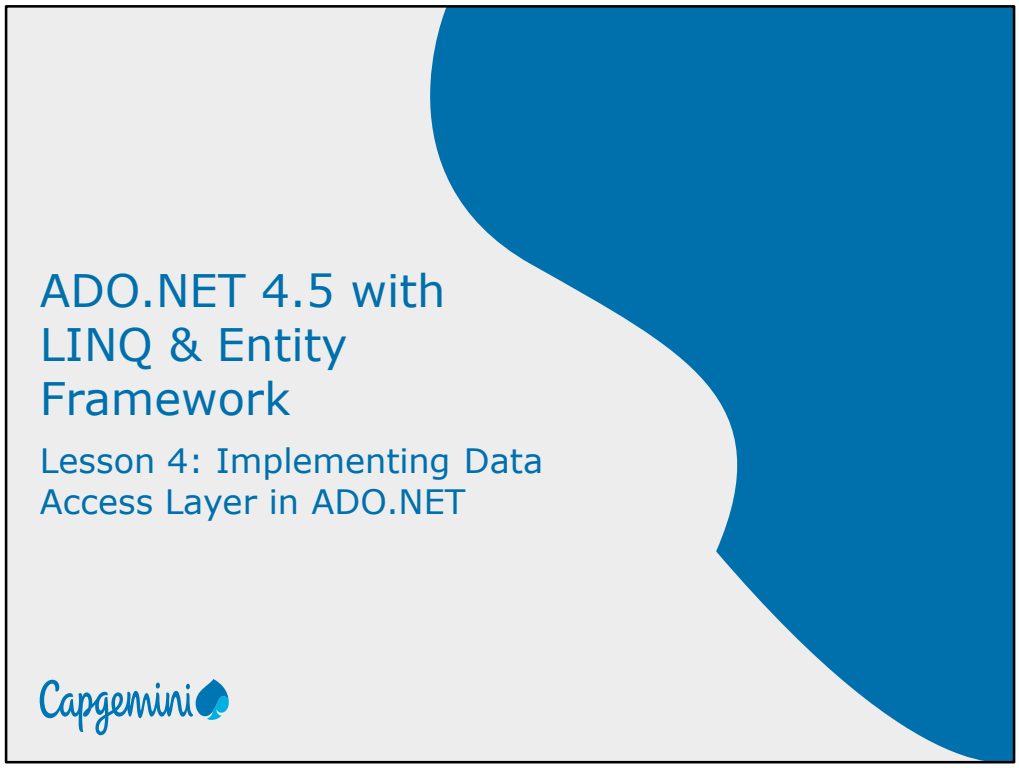
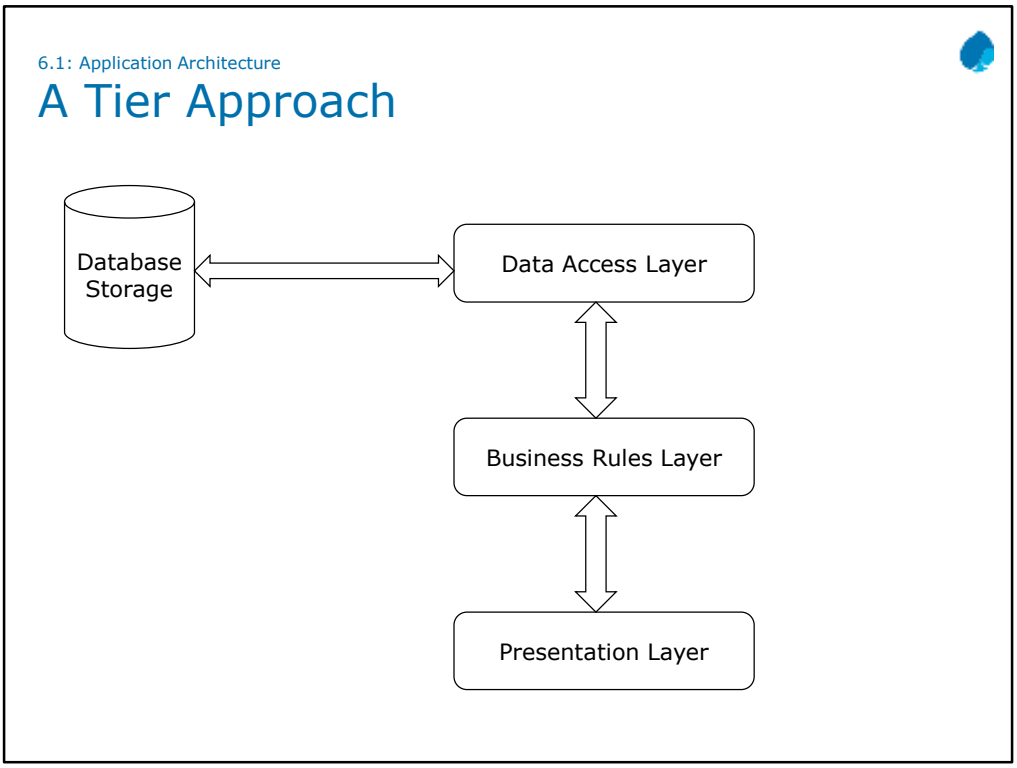

ADO.NET 4.5 with
LINQ & Entity
Framework
Lesson 4: Implementing Data
Access Layer in ADO.NET
Capgemini

Lesson Objectives

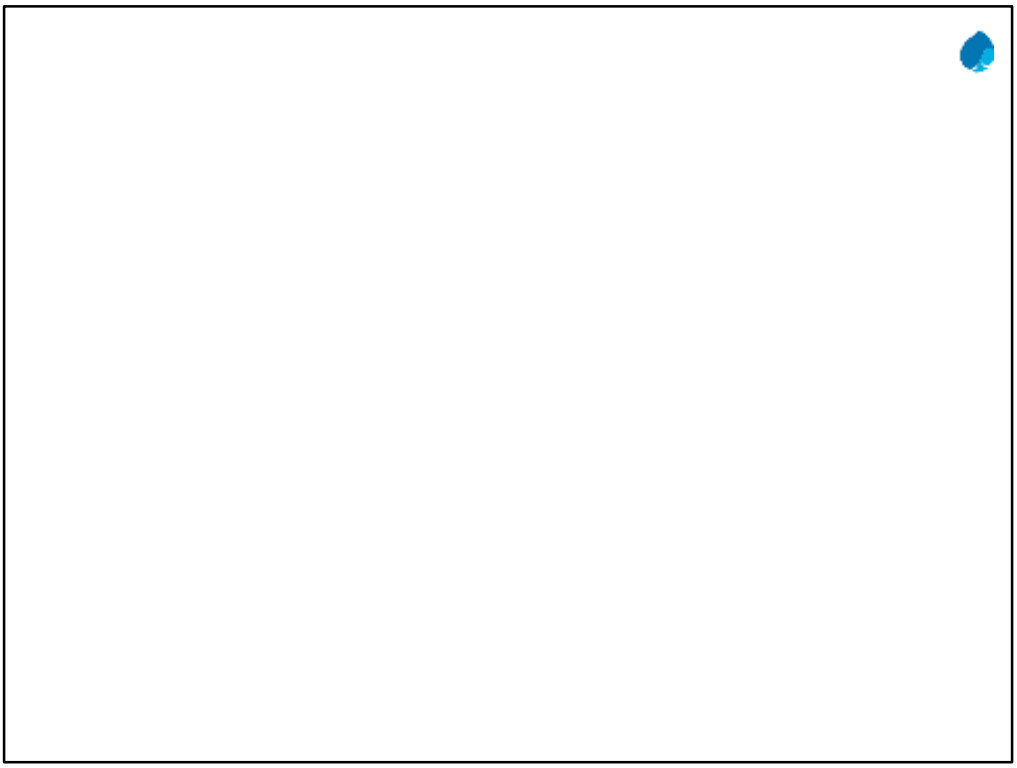
- In this lesson, you will learn:
  - Application Architecture – Tier based architecture
  - Introduction to Data Access Layer
  - DAL – Design considerations
  - Data Access Layer Components
  - Writing code for implementing DAL Components



What is an Application Architecture?

Application architecture is the Organizational Design of an entire software application, including all sub-components and external applications interchanges. A software application is a system designed to automate specific tasks in a logical manner to satisfy a set of requirements. Software applications rely on underlying operating systems and databases to store and perform tasks within the application. The application architecture is the blueprint that defines how the software application will interact with servers and components within the domains of application layers.
There are three main Layers of control within all applications. These are the presentation layer, the business layer, and the data access layer. Each domain within an application has a specific responsibility that, when joined with the other layers, satisfy the underlying business requirements of an application.

Presentation Layer - This is the layer that provides an interface for the end user to your application. The presentation layer is a vitally important part of an application - an inappropriately architected presentation layer can lead to too much complexity, a lack of flexibility and an inefficient and frustrating user experience. It's the layer which represents the interface between the user and the rest of the application.
Business Rules Layer - This is basically where the brains of your application reside; it contains things like the business rules, data manipulation, etc.



Data Access Layer - This layer is where you will write some generic methods to interface with your data. For example, we can write a method for creating and opening a Connection object (internal), and another for creating and using a Command object. This layer, obviously, contains no data business rules or data manipulation/transformation logic. It is merely a reusable interface to the database.

6.2: What is Data Access Layer?

Data Access Layer - An Introduction

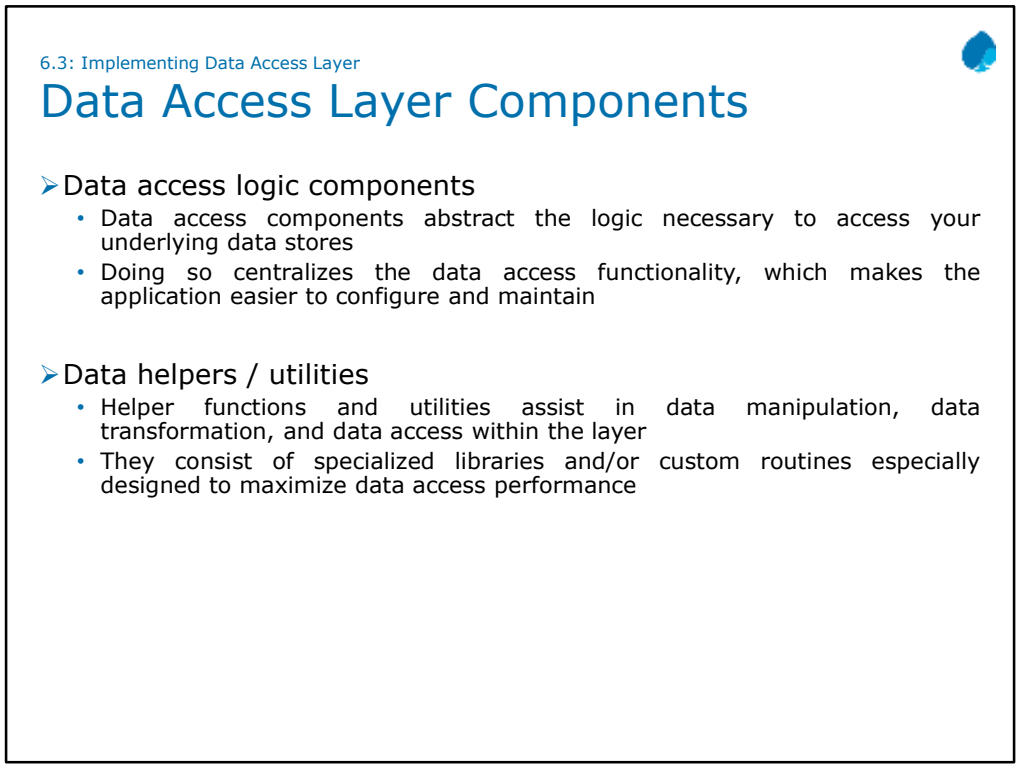
- Implementing data access functionality is a core activity which is performed by most of the developers working with the .NET Framework
- And the data access layers they build an essential part of their applications
- A Data Access Layer is a set of classes and functions for reading from and writing to a database or other data store
- It does not contain any of the business logic for the application nor User Interface elements

Data Access Layer (DAL) – An Introduction

Being an application developer, you have to design many UIs which comprises of Web Pages, Web Forms etc. that perform database CRUD (Create, Read, Update and Delete) operations directly from the code associated with the page. This is a quick but wrong and not so suggested approach that often results in applications that are difficult to maintain, and worse, tend to be inconsistent in their execution of error handling and transactional integrity.

By separating and centralizing code for the activities associated with the specific tasks like data access, you gain the ability to reuse the code, not only within a single project, but across multiple projects, as well. This can greatly simplify maintaining application logic since code for specific tasks is easy to find and changes only need to be made in one place.

In an ideal world, Web pages, or any app for that matter, should be unaware of how the data from a DAL is retrieved. They should only make polite requests to the DAL. It should not matter whether the data comes from Oracle, SQL Server.



Data Access Layer Components

A correct approach to designing the data access layer will reduce development time and assist in maintenance of the data layer after the application is deployed. This part of the lesson briefly outlines an effective design approach for the data layer and the different important components of DAL.

A typical Data Access Layer consists of following main components.

Data access logic component

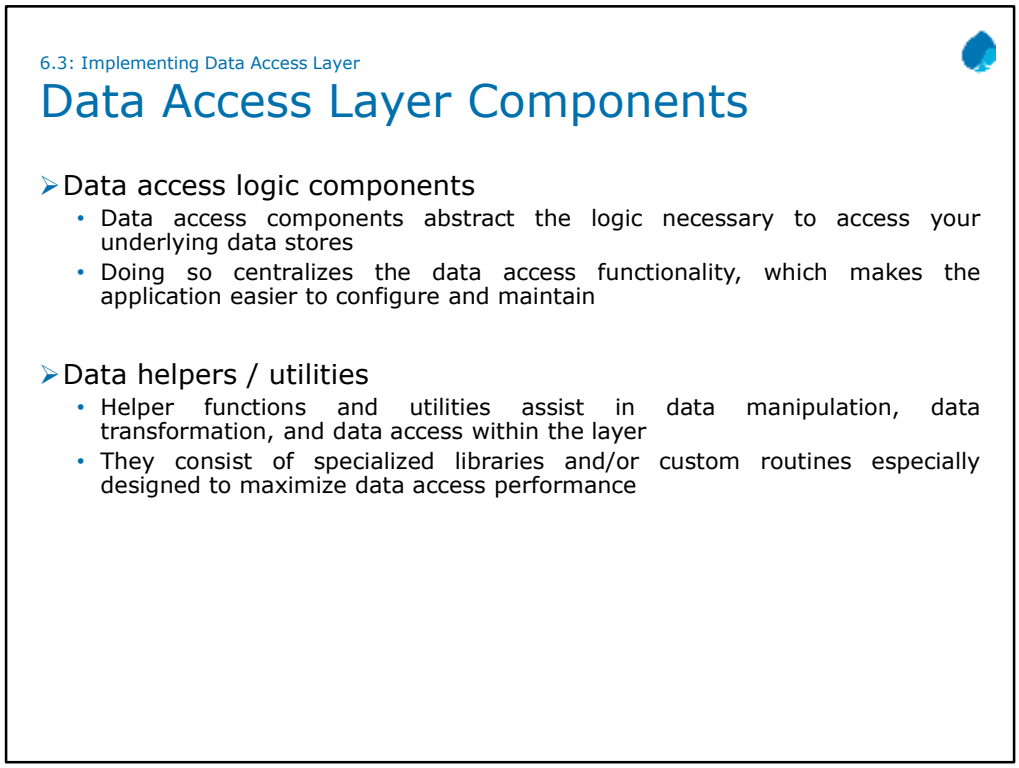
Data access components abstract the logic necessary to access your underlying data stores.

Doing so centralizes the data access functionality, which makes the application easier to configure and maintain.

Data helpers/utilities

Helper functions and utilities assist in data manipulation, data transformation, and data access within the layer.

They consist of specialized libraries and/or custom routines especially designed to maximize data access performance.



Data Access Layer Components

A correct approach to designing the data access layer will reduce development time and assist in maintenance of the data layer after the application is deployed. This part of the lesson briefly outlines an effective design approach for the data layer and the different important components of DAL.

A typical Data Access Layer consists of following main components.
Data access logic component

Data access components abstract the logic necessary to access your underlying data stores.
Doing so centralizes the data access functionality, which makes the application easier to configure and maintain.

Data helpers/utilities

Helper functions and utilities assist in data manipulation, data transformation, and data access within the layer.
They consist of specialized libraries and/or custom routines especially designed to maximize data access performance.

6.3: Implementing Data Access Layer

Data Access Layer Components

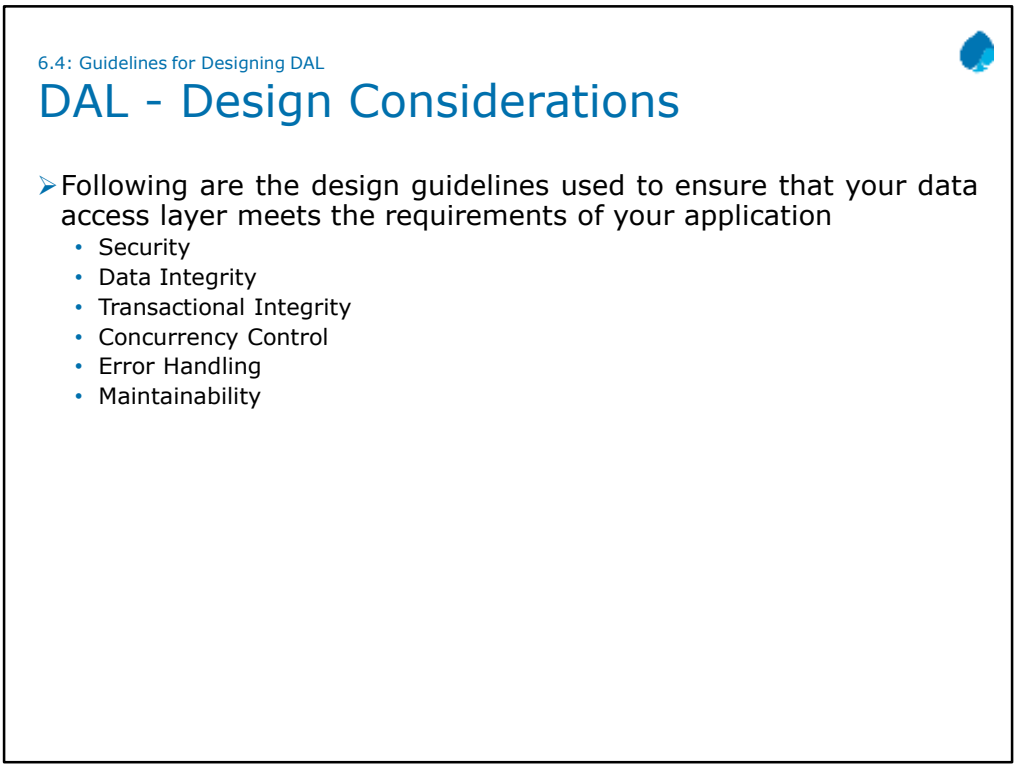
- Customer Management System:

Customer.cs:

```
public class Customer
{
    //Properties to represent individual customer
   //E.g- CustomerID, CustomerName ect
}
```

CustomerOperations.cs:

```
public class CustomerOperations    //Data Access Logic Component
{
  public bool AddCustomerRecord(Customer customerObj)  //Data Helper/Utility
  {
      //Code to add the customer record.
  }
}
```



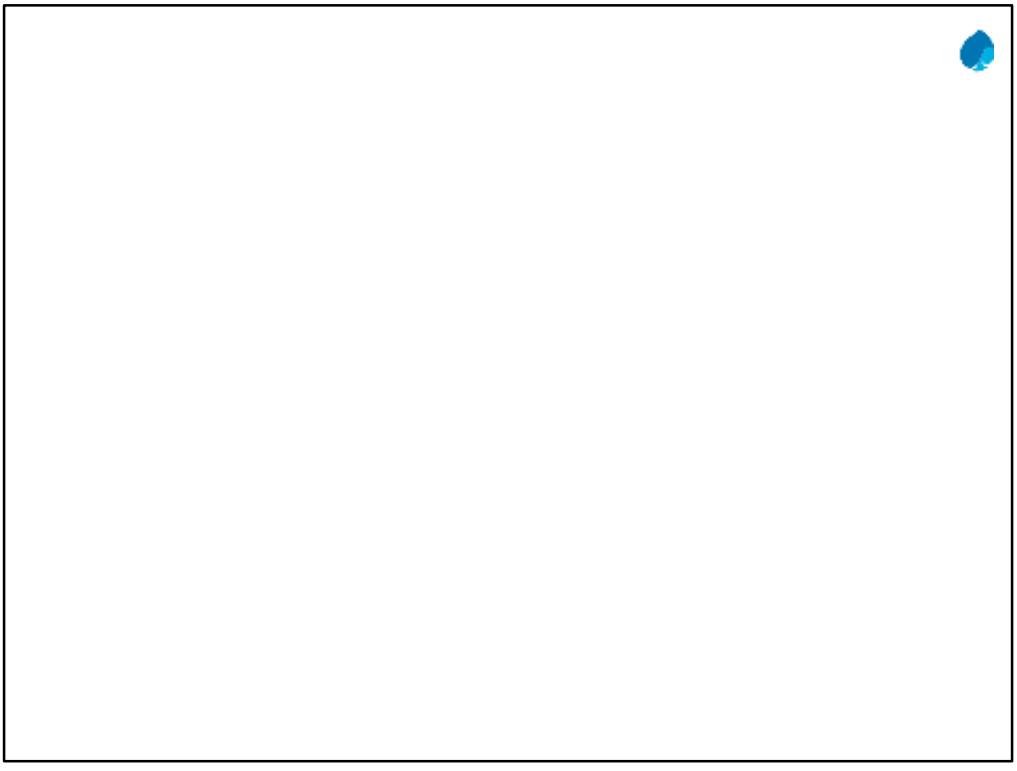
DAL - Design Considerations

The following design guidelines provide information about different aspects of the data access layer that you should consider. Follow these guidelines to ensure that your data access layer meets the requirements of your application, performs efficiently and securely, and is easy to maintain and extend as business requirements change.

Security – The security plays major role in designing DAL. We can design secure data access layer by creating separate database accounts for your applications with no direct permissions on tables within the database. This can be done by providing separate stored procedures for each Select, Update, Insert and Delete action that can be performed on an entity

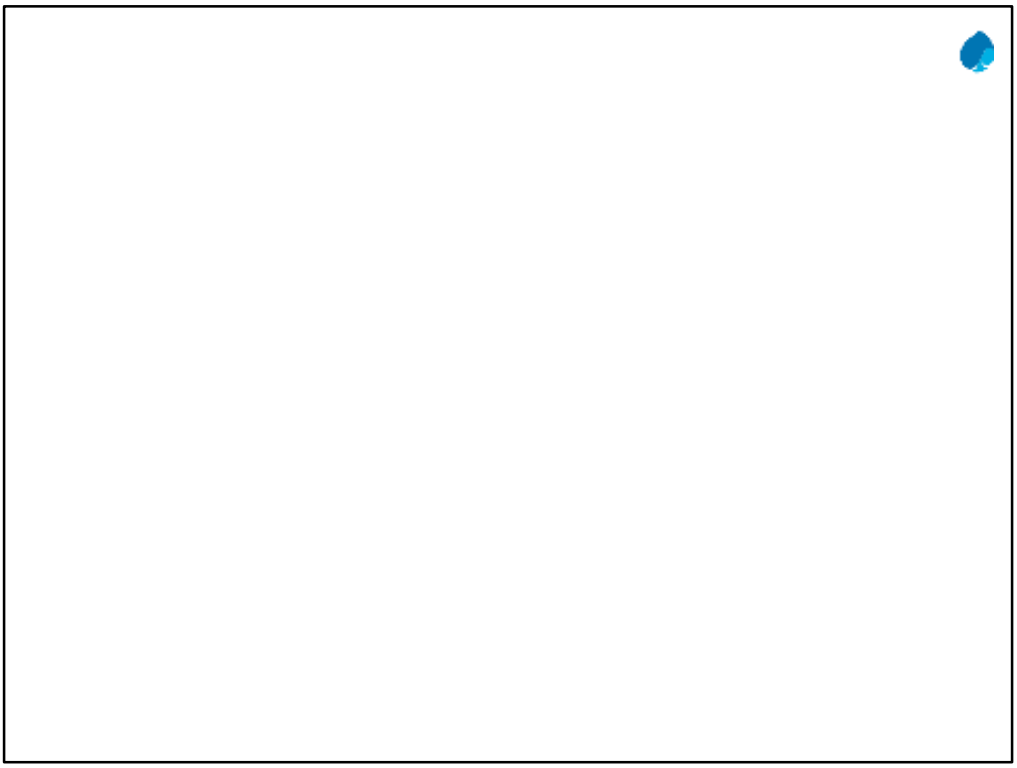
Data Integrity – We always have databases designed with all the necessary constraints to ensure the integrity or correctness of the data. However, we certainly don't want to wait until we submit the data to the database to find out that it doesn't meet the constraints, like referential integrity. Therefore, our DAL logic should be configured to reflect these same constraints and raise errors before we get to the database.

Transaction Integrity - If you are inserting multiple pieces of data in one or more tables, you may need to ensure that all the data is successfully saved. A DAL should support the ability to rollback a series of database transactions if any one of them fails.



Concurrency Control - When many people attempt to modify data in a database at the same time, a system of controls must be implemented so that modifications made by one person do not adversely affect those of another person. This is called concurrency control. We can handle concurrency in three different ways, first, we do nothing, allowing the changes of the last user to overwrite the changes of the first without warning. This is typically known as the "last in wins" scenario. Secondly, you can implement optimistic concurrency control when users do not lock data when they read it. When a user updates data, the system checks to see if another user changed the data after it was read. If another user updated the data, an error is raised. Typically, the user receiving the error rolls back the transaction and starts over. In the third option, you take a pessimistic stand. In this scenario, you predict that it is likely that two users will be vying for the same record so you place a lock on the record when the first person accesses it to prevent any others from accessing it until the first edit is complete.

Error Handling - Since we want our DAL consumers to be unaware of the inner workings of our data retrieval, we certainly don't want unhandled SQL Server exceptions bubbling up to the presentation layer. Instead, the DAL should be providing more useful custom exceptions about the nature of the error. The DAL needs to present these exceptions to the consumer so that the consumer can decide whether it is OK to proceed, try again, or simply give up.



Maintainability – No application code is static or permanent. According to requirements we have to keep on changing application logic/code to meet challenges. Most of the times we needed to change or remove a field in a table and for that to happen we have to search through all our code to find every place that it is being referenced. Such types are changes are always required to be made in our application code. By centralizing your code, you limit the number of places this can happen. Also, by using data objects that provide hard-coded properties that match fields for our entities, the application will not compile if one is removed.

6.5: Advantages Data Access Layer

Advantages of implementing DAL

- It provides most generic way of implementing data access
- It provides encapsulation, centralization and separation of data access code from other application components
- This layer proves higher layer of abstraction
- DAL hides this complexity of the underlying data store from the external world
- It also provides abstraction of Database call

Advantages of implementing DAL

Applications using a data access layer can be either database server dependent or independent. If the data access layer supports multiple database types, the application becomes able to use whatever databases the DAL can talk to.
This layer encapsulates and compartmentalizes all our data access code within nice, clean components, that interact with our database.
The DAL might return a reference to an object complete with its attributes instead of a row of fields from a database table. With this benefit we can create client modules with a higher level of abstraction.
Instead of using commands such as insert, delete, and update to access a specific table in a database, a class and a few stored procedures could be created in the database. The procedures would be called from a method inside the class, which would return an object containing the requested values.
Business logic methods from an application can be mapped to the Data Access Layer. For example, instead of making a query into a database to fetch all users from several tables the application can call a single method from a DAL which abstracts those database calls.

6.6: Data Access Layer

Data Access Layer Example

- Customer Management System:
- Customer.cs:

```
public class Customer  //Entity:  This can be used while communicating with any database
{
    public int CustomerID{get;set;}
    public string CustomerName{get;set;}
    public string CustomerAddress{get;set;}
    public long CustomerContact{get;set;}
}
CustomerOperations.cs:
public class CustomerOperations    //Data Access Logic Component
{
      public bool AddCustomerRecord(Customer customerObj)  //Data Helper/Utility
      {
            //Code to add the customer record.
      }
}
```

In the above example, the developer is working on Customer Management System. The class called Customer has been designed which includes the attributes necessary to represent the customer. The object of this Customer class can be passed to a method that contains the necessary code to connect to the database and perform the CRUD operations. This method may contain the code to connect to SQL Server, Oracle, MS-Access or any other database. From this method, you can call a stored procedure residing on a database server to perform some operation. This relieves you from writing a query to perform a database operation. This method can be called from Business Logic Layer.

Summary

- In this lesson, you have learnt:
  - Understanding Application Architecture – N Tier Architecture
  - Understanding importance of implementing DAL in an application
  - DAL Design considerations
  - Understanding different components involved in DAL

Review Question

- Question 1 What are the different layers of Application?
- Question 2 Which are the two Data Access Layer Components?
- Question 3 From the following list, select the correct guidelines used while designing DAL
  - Data Integrity
  - Event Handling
  - Maintainability
  - Interoperability