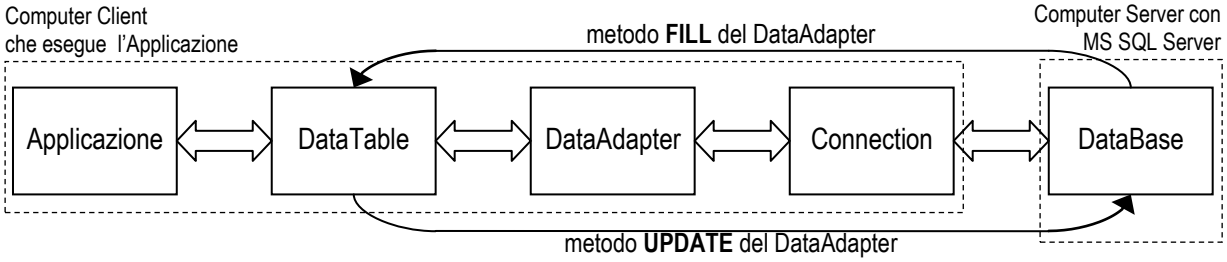
ACCESSO A UN DATABASE MS SQL SERVER DA CODICE VISUAL C#

L'Architettura ADO.NET per l'Accesso a DataBase di MS SQL Server

ADO.NET (*Active Data Objects .NET*) è il nome con il quale si indicano l'insieme di strumenti creati da Microsoft per consentire ad una *Applicazione Windows (o a una Applicazione Web)* di accedere ai dati presenti in un *DataBase*.

- ADO.NET è un insieme di Classi appartenenti al **.NET Framework di Microsoft** e appositamente create per accedere ai *DataBase* gestiti da diversi "*motori server SQL*" quali *MicroSoft SQL Server, MicroSoft Access, MySQL, Oracle* e altri.
- Noi utilizzeremo ADO.NET per accedere a **DataBase di MicroSoft SQL Server** da **Applicazioni Windows o Web** scritte in linguaggio **C#**.

Per accedere ai dati di un DataBase, l'Architettura ADO.NET offerta da MicroSoft opera secondo il seguente schema:



La **Connection** *stabilisce una connessione* fra l'Applicazione e il DataBase sul Server.

Con il **DataAdapter** si *specifica il comando SQL* per recuperare i dati desiderati dal DataBase.

- Il *DataAdapter* rende disponibile il **metodo FILL** che: … *invia il comando SQL al Server;* … il Server *esegue il comando*, genera i dati richiesti e li trasmette al DataAdapter; … infine, il DataAdapter *memorizza i dati ottenuti nel DataTable*.

Il **DataTable** immagazzina i dati recuperati dal DataBase e li rende accessibili "in RAM" all'Applicazione.

Se l'Applicazione modifica i dati nel DataTable, tali modifiche possono facilmente essere "scaricate" sul Server utilizzando il **metodo UPDATE** del DataAdapter.

La classe SqlConnection e la Stringa di Connessione

Per utilizzare le Classi di ADO.NET utili al nostro scopo, è necessario attivare i due *NameSpace* (ossia due "librerie" di classi) chiamati **System.Data** e **System.Data.SqlClient**.

Per **Attivare un NameSpace** si utilizza la direttiva **using** da posizionare all'inizio del codice.

- Ciò significa che, tutte le tue *pagine di codice* (form, classi, ecc.) che accedono a dati di un DataBase dovranno iniziare così:

```
using System.Data
using System.Data.SqlClient
```

La classe **SqlConnection** è la classe *specifica* per stabilire connessioni con DB gestiti su Server con *MS SQL Server*.

Il **Costruttore della classe SqlConnection** necessita di un *argomento di tipo String* detto **Stringa di Connessione**, che specifica tutte le informazioni necessarie a stabilire una connessione con il DataBase sul Server.

- Un tipico esempio di creazione di un oggetto SqlConnection potrebbe essere il seguente:

```
SqlConnection cn = new SqlConnection(
        "server=srvitis\\sqlexpress;database=scuola;user id=sa;password=abacus" );
```

Questa istruzione definisce una *connessione* con un *server* di nome **srvitis**, per l'*istanza* **sqlexpress** di MS SQL Server, al *DataBase* di nome **scuola** e, infine, che si desidera accreditarsi con l'account: *user id* pari a **sa** e *password* pari a **abacus**. Si noti la presenza di una "doppia back-slash" (****), che viene interpretata da C# come un'unica back-slash (****).

La **Sintassi della Stringa di Connessione** è la seguente:

"parametro=valore; parametro=valore; parametro=valore;…"

Alcuni tipici *Parametri* della *Stringa di Connessione* sono (ne esistono molti altri):

server: specifica il *Nome del Server* sul quale è attivo MS SQL Server e, dopo una *backslash* **"\"** (in C# ne vanno inserite due perché le interpreta come una sola), *l'Istanza di MS SQL Server* che contiene il DataBase desiderato.

✋ Tale server deve essere *raggiungibile tramite rete* dal Client che esegue l'applicazione. Un caso limite è l'uso della versione di SQL Server che si usa per lo Sviluppo delle Applicazioni, denominata **localdb**, la cui istanza è **MSSQLLocalDB**: in tal caso lo stesso PC fa *sia da Server che da Client* e la *Stringa di Connessione* è del tipo:

"server=(localdb)\\MSSQLLocalDB;database=scuola;user id=sa;password=abacus"

database: specifica il *Nome del DataBase* al quale si desidera accedere.

Generalmente, per accedere a un DB su MS SQL Server, è necessario "accreditarsi" con i seguenti parametri:

user id: specifica il *Nome Utente* dell'account con il quale si desidera accreditarsi.

password: specifica la *Password* dell'account con il quale si desidera accreditarsi.

✋ Nei nostri esempi utilizzeremo, per semplicità, l'account speciale **sa** (*sql administrator*) che dà accesso a *tutte le funzionalità* di MS SQL Server. La password che abbiamo scelto per tale account è **abacus**.

La Connessione con il Server viene *automaticamente* "*aperta*" solo *al momento dell'esecuzione del metodo FILL* (o del *metodo UPDATE*) del DataAdapter e, terminate le azioni previste dal FILL (o UPDATE), viene *automaticamente chiusa*.

La *Creazione di un Oggetto di classe SqlConnection è necessaria* prima di poter utilizzare le altre classi ADO.NET.

La classe SqlDataAdapter e il Comando SQL di tipo SELECT

La classe **SqlDataAdapter** è la classe *specifica* per *inviare Comandi SQL di Selezione* a DB gestiti su Server con *MS SQL Server* e *gestire le Tabelle di Dati* restituite dal Server stesso.

SqlDataAdapter esige un **Comando SQL di tipo SELECT**, che consente di "*selezionare*" (ossia scegliere) quali dati si desidera ottenere, da quali tabelle del DB, in che ordine, con quale "filtro", eccetera.

☞ Esempi di Comandi SQL di tipo SELECT sono i seguenti:

SELECT * FROM Alunni … tutti i campi e tutti i record della tabella Alunni
SELECT Nome, Cognome FROM Alunni … solo Nome e Cognome di tutti i record della tabella Alunni

SELECT * FROM Alunni
WHERE Nome="Antonio" … tutti i campi ma solo i record degli alunni che hanno per nome "Antonio"

SELECT Cognome, Nome FROM Alunni
ORDER BY Cognome … solo Cognome e Nome di tutti gli alunni, ma ordinati per Cognome

Nota che, il risultato dell'esecuzione di un Comando SELECT è comunque e sempre una *tabella di dati.*

Il **Costruttore della classe SqlDataAdapter** necessita di un *primo parametro di tipo String* che contiene il **Comando SQL di tipo SELECT** e di un *secondo parametro che indica l'oggetto SqlConnection* per connettersi al DataBase.

☞ Un tipico esempio di *creazione di un oggetto SqlDataAdapter* potrebbe essere il seguente:

```
SqlConnection cn = new SqlConnection("server=(localdb)\\MSSQLLocalDB;database=scuola;user id=sa;password=abacus");
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Alunni", cn);
```

Questa istruzione crea l'oggetto **da** di classe SqlDataAdapter e lo prepara per inviare al Server, il Comando SQL di selezione **SELECT * FROM Alunni**, sfruttando la connessione **cn** definita nell'istruzione precedente.

L'invio del Comando SQL al Server avviene solo *quando si esegue il metodo FILL* dell'oggetto DataAdapter.

La classe DataTable e il metodo FILL del DataAdapter

La classe **DataTable**, come già studiato in precedenza, consente di *memorizzare e gestire in RAM dati in forma tabellare*, specificando nome e caratteristiche dei campi (**insieme Columns**) e permettendo un accesso diretto ai record tramite un indice (**insieme Rows**).

Il **Costruttore della classe DataTable** non necessita di alcun parametro obbligatorio.

☞ Per creare un oggetto di classe DataTable basta quindi scrivere:

```
DataTable dt = new DataTable();
```

La classe *SqlDataAdapter* dispone del **metodo FILL** che, quando eseguito provoca le seguenti azioni:

- il DataAdapter *apre la Connessione con il Server* e *invia ad esso il Comando SQL* di tipo SELECT.
- Il motore di MS SQL Server riceve ed *esegue il Comando SQL* sul DataBase, *generando la Tabella di Dati* richiesta.
- Il motore di MS SQL Server *invia al Client la Tabella di Dati* generata dal Comando SQL.
- Il DataAdapter memorizza la Tabella di Dati ricevuta all'interno di un *oggetto di classe DataTable*.

Il *metodo FILL* necessita, come parametro, *dell'oggetto DataTable* in cui memorizzare la tabella restituita dal Server.

☞ Per caricare in un DataTable di nome *dt*, tutti i campi e tutti i record della *tabella Alunni* del DB, dovrai scrivere:

```
SqlConnection cn = new SqlConnection("server=(localdb)\\MSSQLLocalDB;database=scuola;user id=sa;password=abacus")
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Alunni", cn);
DataTable dt = new DataTable();
Da.Fill (dt)
```

La *Struttura del DataTable*, ossia la definizione delle sue Colonne (**Insieme Columns**) viene *automaticamente creata dal DataAdapter* in base alla Tabella di Dati proveniente dal Server.

Anche l'**Insieme Rows** del DataTable viene *automaticamente riempito dal DataAdapter* con tutti i record presenti nella Tabella di Dati proveniente dal Server.

Accedere o Modificare i Record in un oggetto DataTable

Per **Accedere ad uno specifico Record** di un DataTable, si utilizza l'**insieme Rows**, indicando l'**Indice del Record** desiderato:

<*oggetto-datatable*>**.Rows[**<*indice*>**]**

✋ Non dimenticare che *il primo record ha indice* **0** e che *l'ultimo record ha indice* **dt.Rows.Count-1**

Per **Accedere ad uno specifico Campo** di uno specifico Record di un DataTable, si utilizza la **proprietà Item** della Riga indicando una **Stringa con il Nome del Campo**:

<*oggetto-datatable*>**.Rows[**<*indice*>**].Item[**<*nome-campo*>**]**

o nella *forma abbreviata*, che omette ".Item":

<*oggetto-datatable*>**. Rows[**<*indice*>**][**<*nome-campo*>**]**

✋ L'accesso ad un campo può avvenire, ovviamente, sia in *lettura* che in *scrittura*. il campo può essere usato *come una variabile*: può essere "letto", per utilizzare il suo valore, oppure vi si può "assegnare" un dato, modificandone il valore.

☞ Ad esempio, per visualizzare in un MessageBox il Nome dell'alunno registrato nel record in 6° posizione, dovrai scrivere:

MessageBox.Show(**dt.Rows[5].Item["Nome"]**) oppure MessageBox.Show (**dt.Rows[5]["Nome"]**)

Per modificare il Cognome dell'alunno registrato nel primo record e impostarlo a "Rossi", dovrai scrivere

dt.Rows[0].Item["Cognome"] = "Rossi" oppure **dt.Rows[0]["Cognome"]** = "Rossi"

Query Parametriche (comandi SELECT con uso di Parametri)

La maggior parte delle volte, un comando Sql di tipo **SELECT**, deve essere eseguito in base ad uno o più **Parametri**.

☞ Ad esempio, si desidera visualizzare, anziché TUTTI gli alunni, solo quelli che hanno un *determinato nome* (Es. 'Antonio').

In tal caso, la tecnica corretta per impostare il comando Sql nel DataAdapter, prevede di **indicare esplicitamente il parametro tramite una "variabile sql"**, caratterizzata dalla notazione: **@<identificatore>**.

☞ Riprendendo l'esempio precedente, il comando Sql dovrebbe essere il seguente:

```
SELECT * FROM Alunni WHERE Nome = @NomeDaCercare
```

dove la notazione ***@NomeDaCercare*** indica il ***Parametro*** *(variabile sql)*, il cui valore sarà confrontato con il campo Nome allo scopo di filtrare gli alunni desiderati. Quindi il DataAdapter dovrà essere impostato come segue:

```
SqlDataAdapter da = new SqlDataAdapter (
    "SELECT * FROM Alunni WHERE Nome = @NomeDaCercare", cn);
```

Oltre a indicare esplicitamente il *Parametro* nel comando Sql, è necessario **informare il DataAdapter dell'esistenza del parametro** stesso, specificandone **Identificatore, Tipo** ed eventualmente, **Dimensione**:

<*oggetto-DataAdapter*>**.SelectCommand.Parameters.Add (**
"<*identificatore-parametro*>**"**, **SqlDbType.**<*tipo-sql*>**,** <*dimensione*>)

Come si vede, è necessario accedere all'oggetto *"interno"* **SelectCommand** e, quindi, al suo insieme *"interno"* **Parameters**, al quale si *aggiunge* (metodo **Add**) la definizione del nuovo parametro.

☞ Nel nostro solito esempio, si ha:

```
SqlDataAdapter da = new SqlDataAdapter ( "SELECT * FROM Alunni WHERE Nome = @NomeDaCercare", cn);
da.SelectCommand.Parameters.Add ( "@NomeDaCercare", SqlDbType.NChar, 50 );
```

indicando *l'identificatore* **("@NomeDaCercare")** e il *Tipo* **(SqlDbType.NChar)**.

✋ Si noti come **SqlDbType** è una *enumerazione* che, dopo aver digitato il punto, elenca automaticamente tutti i possibili tipi disponibili in MS Sql Server.

Infine, per **Attribuire un Valore al Parametro**, si accede al parametro appena definito nell'insieme *Parameters* e si imposta la sua **proprietà Value**:

<*oggetto-DataAdapter*>**.SelectCommand.Parameters["**<*identificatore-parametro*>**"].Value =** <*valore*>

☞ Procedendo nel nostro esempio:

```
SqlDataAdapter da = new SqlDataAdapter ( "SELECT * FROM Alunni WHERE Nome = @NomeDaCercare", cn);
da.SelectCommand.Parameters.Add ( "@NomeDaCercare", SqlDbType.NChar, 50 );
da.SelectCommand.Parameters["@NomeDaCercare"].Value = txtNomeRichiesto.Text;
```

ipotizzando che l'utente abbia specificato il nome da filtrare in una apposita TextBox chiamata *txtNomeRichiesto.*

A questo punto è possibile richiamare il *metodo Fill del DataAdapter* che invierà al Server Sql il comando SELECT e la definizione del Parametro: il Server eseguirà la query *"sostituendo"* al parametro il valore specificato.

☞ Completiamo l'esempio:

```
SqlDataAdapter da = new SqlDataAdapter ( "SELECT * FROM Alunni WHERE Nome = @NomeDaCercare", cn);
da.SelectCommand.Parameters.Add ( "@NomeDaCercare", SqlDbType.NChar, 50 );
da.SelectCommand.Parameters["@NomeDaCercare"].Value = txtNomeRichiesto.Text;
DataTable dt = new DataTable();
da.Fill (dt);
```

La tecnica esposta per le *Query Parametriche* è preferibile alla costruzione del comando sql *includendo "brutalmente" il valore del parametro* nel comando stesso. Così facendo, infatti, un dato "esterno" (il *valore* del parametro) viene a costituire parte integrante del comando sql, diventando così "eseguibile" ed esponendo la query al cosiddetto **SQL INJECTION** (https://it.wikipedia.org/wiki/SQL_injection): questa "debolezza" consente a un malintenzionato di inserire, nella query, parte di comandi indesiderati e malevoli, semplicemente specificando valori particolari per il parametro.