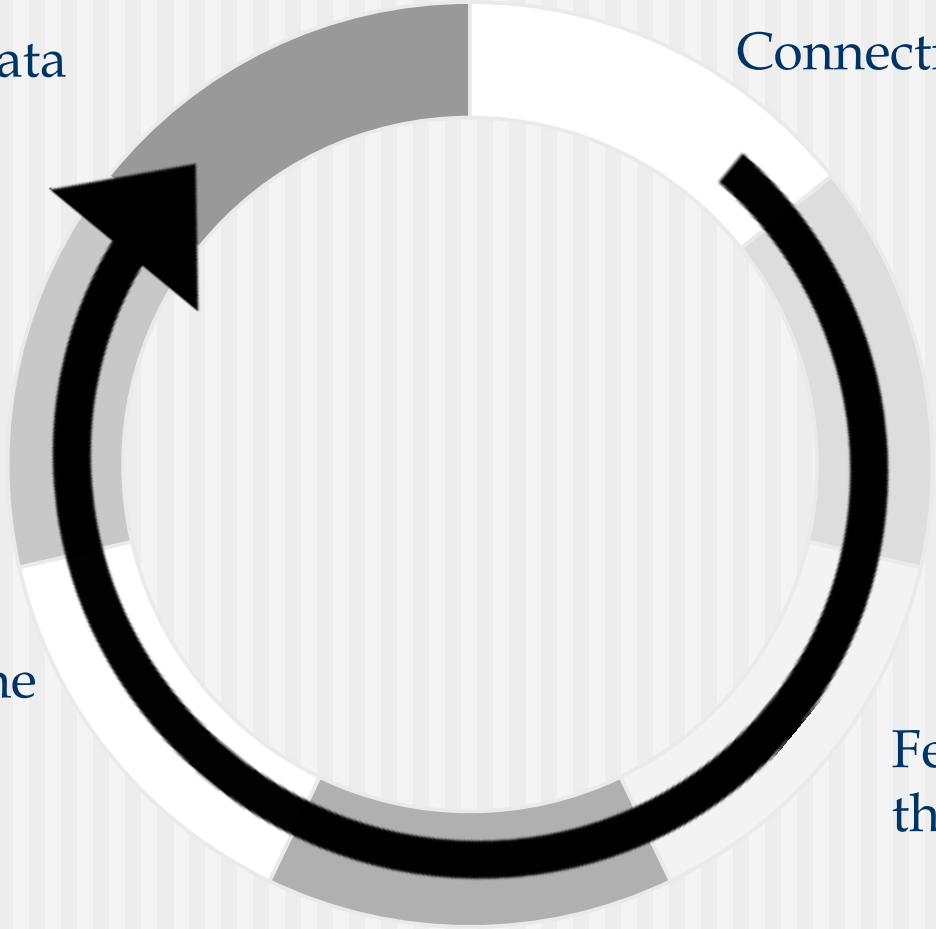
Seminar 1

ADO.NET

The Data Cycle



The Data Cycle

- connecting to data: establish a two-way communication between the application and the database server (*TableAdapter*, *ObjectContext*)
- preparing the app to receive data: when using a disconnected data model there are specific objects that temporarily store data (*datasets*, *entities*, *LINQ to SQL* objects)
- fetching data: execute queries and stored procedures (*TableAdapters*, *LINQ to Entities*, direct connection between entities and stored procedures)

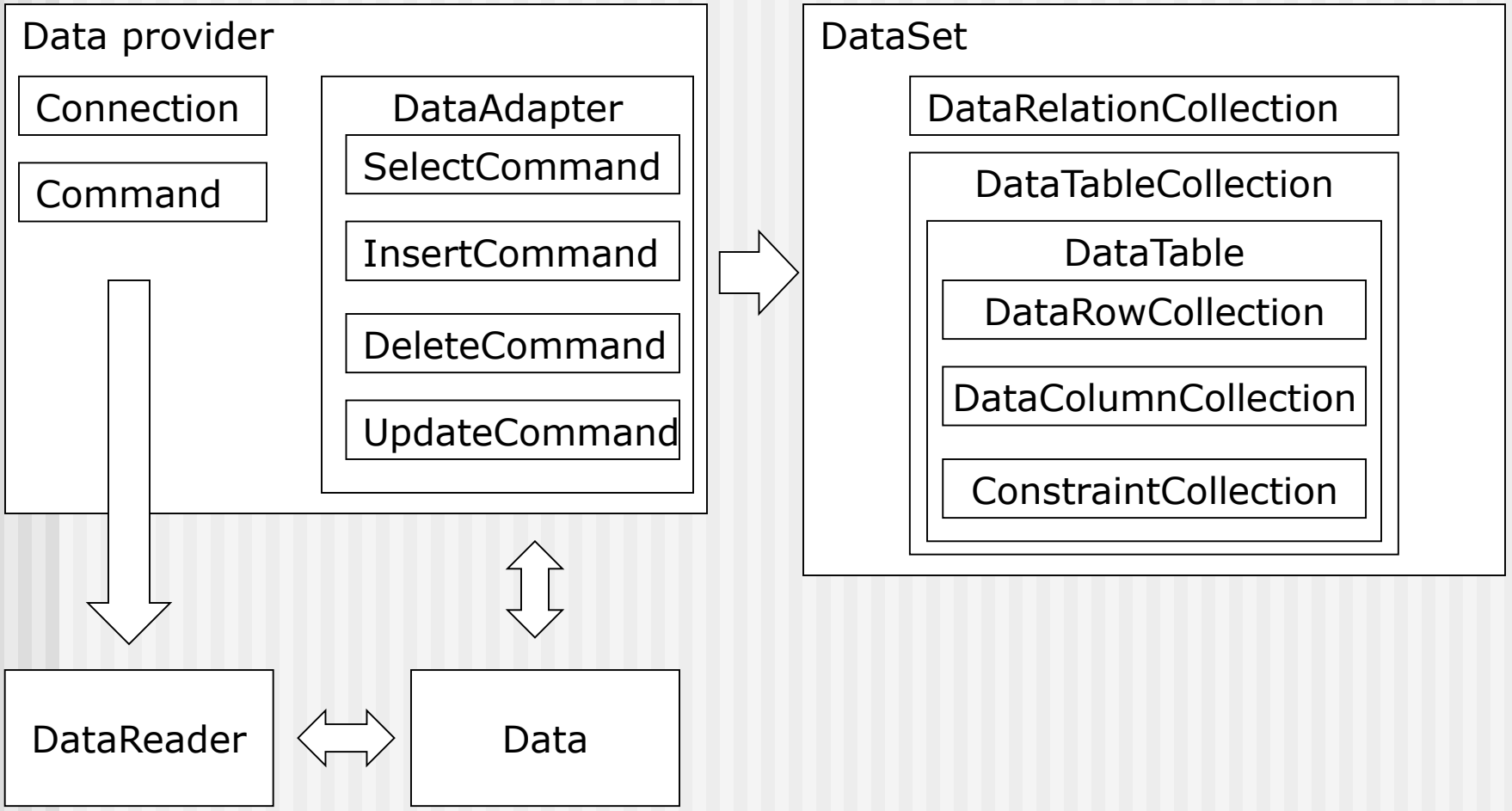
The Data Cycle

- displaying data: data-bound controls
- editing and validating data: add / modify / delete records; verify if the new values meet the requirements of the application
- saving data: *TableAdapterManager*, *SaveChanges*

Data Models

- typed / untyped datasets
- conceptual model based on the *Entity Data Model*; it can be used by the *Entity Framework* or *WCF Data Services*
- *LINQ to SQL* classes

ADO.NET


Data provider
Connection
Command
DataAdapter
SelectCommand
InsertCommand
DeleteCommand
UpdateCommand
DataSet
DataRelationCollection
DataTableCollection
DataTable
DataRowCollection
DataColumnCollection
ConstraintCollection
DataReader
Data

DataSet

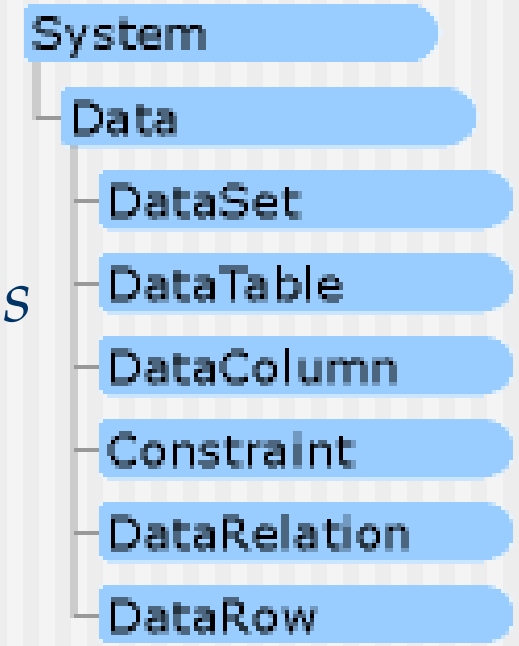
- object containing data tables that can temporarily store the data used in the application
- typed / untyped
- local in-memory cache
- also works when the application disconnects from the database
- has a similar structure to a relational database (tables, rows, columns, constraints, relationships)

DataSet

- properties:
  - *Tables (DataTableCollection)* – collection of tables in the DataSet
  - *Relations (DataRelationCollection)* – collection of relations (child / parent tables)
- methods:
  - *Clear*() – clears all rows in all tables
  - *HasChanges*() – indicates whether there are new / deleted / modified rows

DataSet

- the *DataTable* class – properties
  - *Rows (DataRowCollection), Columns (DataColumnCollection), ChildRelations* and *ParentRelations (DataRelationCollection)*, etc
- the *DataRow* class - the *RowState* property (possible values: *Added, Deleted, Modified, Unchanged*)



SqlConnection

- represents a connection to a SQL Server database
- cannot be inherited
- if the SqlConnection goes out of scope, it is not closed => must ensure the connection is always closed (e.g., *Close)*
- properties:
  - *ConnectionString* - string used to open a SQL Server database (http://www.connectionstrings.com/)
  - *ConnectionTimeout* - time to wait to establish a connection before terminating the attempt and generating an error

SqlConnection

- methods:
  - *Open*()
  - *Close*()
- if a *SqlException* is generated, the *SqlConnection* remains open when the severity level <= 19

SqlCommand

- represents a Transact-SQL statement or stored procedure to be executed on a SQL Server database
- cannot be inherited
- properties:
  - *CommandText*
  - *CommandTimeout*
- methods:
  - *ExecuteNonQuery* – returns the number of affected rows

SqlCommand

- methods:
  - *ExecuteScalar* – returns the first column of the first row in the answer set
  - *ExecuteReader* – builds a SqlDataReader

SqlDataReader

- reads a forward-only stream of rows from a SQL Server database

SqlDataAdapter

- bridge between a DataSet and SQL Server to obtain data and save changes back to the database
- a set of commands and a database connection
- properties:
  - *UpdateCommand* - statement / stored procedure used to update records in the data source
  - *InsertCommand* - statement / stored procedure used when inserting records into the data source
  - *DeleteCommand* - statement / stored procedure used to delete records

SqlDataAdapter

- methods:
  - *Fill(DataSet, String)* - adds or refreshes rows in the DataSet object to match those in the data source (2nd param – name of table)
  - *Update(DataSet, String)* – changes the values in the database by executing INSERT / UPDATE / DELETE statements for every added / modified / removed row

Console

- it represents the standard input, output, error streams for console applications
- properties:
  - *WindowLeft*, *WindowTop*, *WindowHeight*, *WindowWidth*, *BackgroundColor*, *Title*, etc
- methods:
  - *Write(….), WriteLine(….)*
  - *Read(), ReadLine(), ReadKey()*