PV178: Programming for .NET Framework

Accessing Relational Databases with ADO.NET

Vojtěch Forejt, forejt@fi.muni.cz
Martin Osovský, osovsky@ics.muni.cz

Faculty of Informatics and **Institute of Computer Science**
Masaryk University

March 26, 2009

ADO.NET vs. ADO

- ADO.NET is successor of ADO (ActiveX Data Objects)
- ADO
  - Works mainly "online"
  - Basic structure is RecordSet representing one table
  - Unsatisfactory abstraction from physical data model
  - Loss of relation between data
- ADO.NET
  - Designed for "offline" work
  - Basic structure is DataSet representing one or more tables and relations between them

Namespaces

- `System.Data` – classes used for data access
- `System.Data.(OleDb|Oracle|Odbc|SqlClient)`, `MySql.Data.MySqlClient...` for data providers
- `System.Data.SqlTypes`, `SystemData.Sql` – specific classes for Microsoft SQL Server

Classes of data providers

- Derived from common base class, implementing common interface
- E.g. `(Sql|Oledb|...|MySql)Connection` implement `IDbConnection` and derive from `DbConnection`, allow to connect to database

Database Connection

- `<provider>Connection`
- represent database connection
- connection is determined by connection string given in constructor or via `ConnectionString` property
  - example:
    "server=pat.fi.muni.cz;user id=mysql;database=maindb"
- connection is opened and closed using `Open` and `Close` methods.

Database Commands

- `<provider>Command`
- represent command passed to database (mostly SQL query)
- properties `Connecion` and `CommandText` (may be set in constructor)
- `CommandType` property – `Text` or `StoredProcedure` (or `TableDirect`)

Database Commands – Execution

- `ExecuteNonQuery` – executes command and returns number of affected rows
- `ExecuteScalar` – executes and returns one value of type `Object`
- `ExecuteReader` – executes and returns `IDataReader`

Database Commands – Parameters

- property `Parameters` allows to pass parameters to stored procetude or to include them in SQL queries.
- every provider has `<provider>Parameter` class
- parameter properties
  - `ParameterName` – unique in the collection
  - `Direction` – input/output (for stored procedures)
  - `Value`
  - `DbType`, `<provider>Type` – type of parameter in database, given by `DbType` and `<provider>Type` enums.

Example

- `ElementaryDbExample`

Transactions

- used when some changes to database are done together
- connection's method `BeginTransaction` returns `<provider>Transaction`
- methods `Commit` and `Rollback`
- command's `Transaction` property
- property `IsolationLevel` – visibility of changes

Example

- `TransactionExample`

<provider>DataReader Class

- sequential access to data
- data read row by row, step to next row using `Read` method
- two indexers
  - integer – order of column
  - string – names of column
- for common types, one may use `Get<type>` instead of indexer.

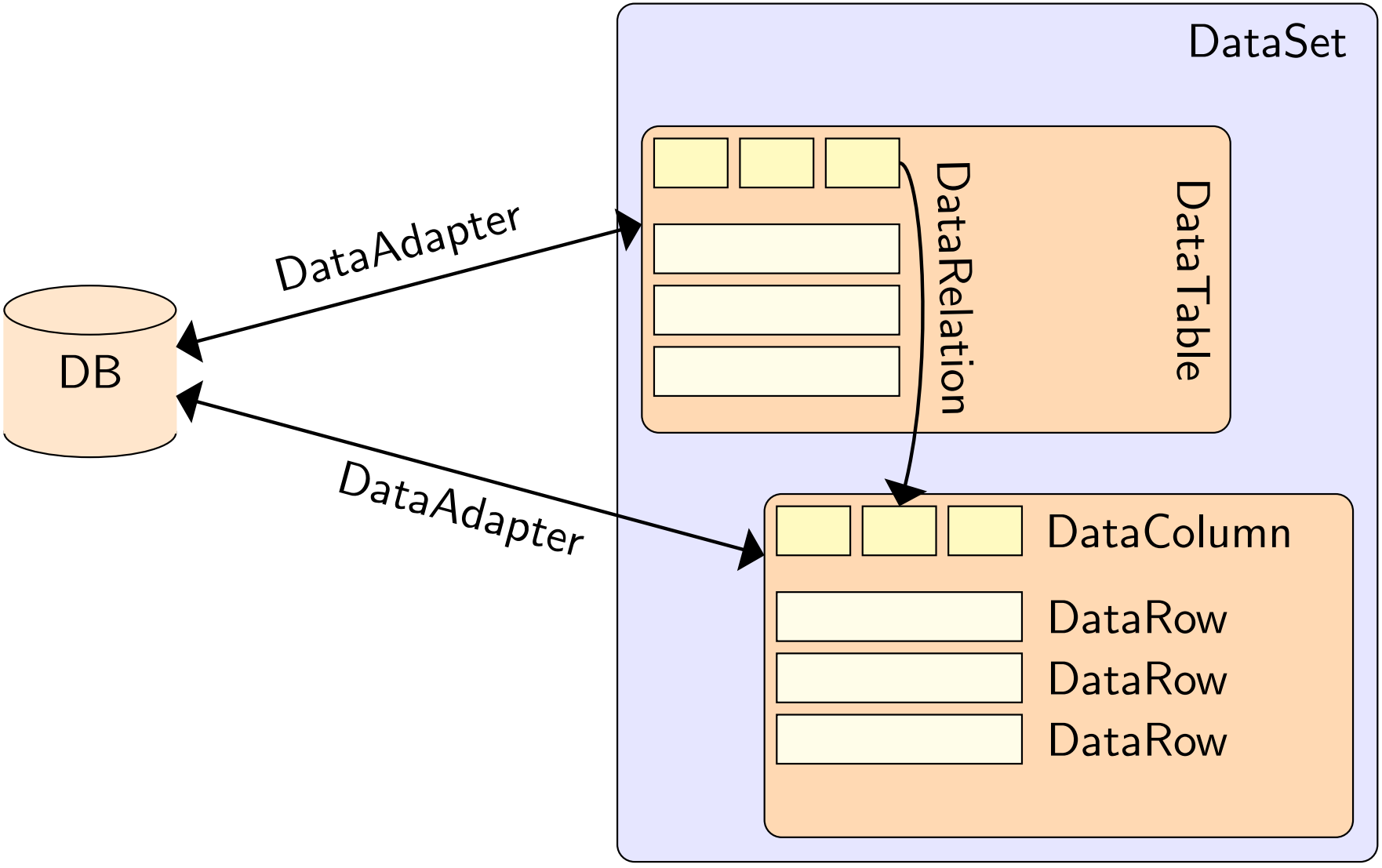
Example

- `DataReaderExample`

ADO.NET and RAD

- Rich suport for RAD in Visual Studio
- Many classes can be generated automatically.
- WinForms controls can be bound with data in desing time

Overview


DataSet
DataRelation
DataTable
DataAdapter
DB
DataAdapter
DataColumn
DataRow
DataRow
DataRow

`System.Windows.Forms.BindingSource` Class

- Provides a data source for a form
- Binds data sources with controls using their `DataBindings` property
- Supports quite complex operations with data (sorting, filtering)

DataSet Class

- Class representing online data container
- Not dependent on database, may e.g. store XML data
- Contains objects for tables, their columns, row and relations. Each of object represented by separate class (`DataTable`, `DataRow`,...)

`DataTable` Class

- object representing table containing data
- contains collection of `DataColumns`
- may contain one or more `DataRows`
- may contain constraints on columns and primary key information

DataColumn Class

- identified by name
- properties
  - `ColumnName`
  - `AutoIncrement` – automatic generation of numeric value
  - `DataType` – one of several .NET types (`int`, `double`, `TimeSpan`, `String`...), cannot be changed after table is filled with data
  - `DefaultValue`

DataRow

- class representing data in dable
- editing is started using `BeginEdit`, ended using `EndEdit` or `CancelEdit`. In between, constraint control is suspended
- data are stored "offline", thus must be versioned
  - `Original` – value retrieved from external source
  - `Current` – last "valid" value assigned
  - `Proposed` – last (potentionally "invalid") value assigned between calling `BeginEdit` and `EndEdit` or `CancelEdit`
  - `Default`

DataRow cont.

- (six) indexers, one or two parameters
  - first parameter is either `DataColumn`, `int` or `String`
  - second (optional) parameter is `DataRowVersion`.
- `RowState` property
  - `Added`, `Deleted`, `Detached`, `Modified`, `Unchanged`
- `AcceptChanges` method changes `RowState` to `Unchanged` and version to `Original`.

Example

- DataTableExample.cs

DataRelation

- Adds a named relation between two collection of columns in two tables of dataset
- foreign-key – primary-key
- methods `GetChildRows` and `GetParentRows` of `DataRow` may be used to navigate using these relations.

Constraints

- Constraints on values in columns
- represented using objects
  - `UniqueConstraint` – every value in column must be unique
  - `ForeignKeyConstraint` – restriction on two (collections of) columns.
- applied only if DataSet's `EnforceConstraint` property is `true`

Constraints cont.

- property `(Update|Delete)Rule` for rules of updates and deletion of rows
  - `Cascade` – changes all child rows
  - `None`
  - `SetDefault` – child columns with FK that does not exist get default value
  - `SetNull`

DataSet Schemas

- may be generated in 3 ways
  - automatically from data source
  - manually in code
  - from XML schema

Filling DataSet

- after setting DataSet schema, it is filled with data
  - using `<provider>DataAdapter`
  - from XML file
  - manually by adding and editing rows

DataAdapter class

- provides connection between `DataSet` and databese
- properties `SelectCommand`, `UpdateCommand`, `InsertCommand`, `DeleteCommand`
- methods
  - `Fill` – uses `SelectCommand` to fill dataset from database.
  - `Update` – calls the other three commands so that data are stored to the database.