ADO.NET

Continued

DataSets & DataAdapters

- SqlDataReader provides stream-based access to the results of database queries
- Read-only, forward-only
- Could be an inconvenience
- ADO.NET provides a set-based access as well as a stream-based access

DataSets & DataAdapters

- DataSet: in-memory database
- SqlDataAdapter: bridge between real database and datasets

DataSet

- DataTable objects are analogous to tables in a database
- DataSet.Tables property lists DataTables
- There are also DataRow and DataColumn objects
- Ideal for capturing the result of database queries and modifying the data
- Useful for caching. How?

DataSets vs. DataReaders

- Which is better?
  - It depends

DataAdapter Classes

- DataSets are mostly initialized from database queries or XML documents
- DataSets do not interact with the database directly
- DataAdapters perform database queries and create DataTables containing the query results
- When working with Microsoft SQL Server, we are concerned with SqlDataAdapter

Example

```
SqlDataAdapter adapter = new SqlDataAdapter
  ("select * from titles",
   "server=localhost;database=pubs;uid=sa;
    pwd=");
DataSet ds = new DataSet();
Adapter.Fill(ds, "Titles");
```


What Happens?

1. Fill opens a connection to the pubs database using the connection string
2. It performs a query using the query string
3. It creates a DataTable named "Titles" in the DataSet
4. It initializes the DataTable with a schema that matches "titles" in database
5. It retrieves all records produced by the query and writes them to the DataTable
6. It closes the connection to the database

DataTable and Friends

- A DataSet is a collection of DataTables

```
foreach(DataTable table in ds.Tables)
 Console.WriteLine(table.TableName);
```


DataTable and Friends

- Individual DataTables can be referenced by name or index
- DataTables contain DataRows

```
DataTable table = ds.Tables[0];
foreach(DataRow row in table.Rows)
 Console.WriteLine(row[0]);
```

- Could also use column name instead of index

DataTables

- You can also enumerate a DataTable's columns

```
DataTable table = ds.Tables[0];
foreach(DataColumn col in table.Columns)
 Console.WriteLine("Name={0}, Type={1}",
  col.ColumnName, colDataType);
```


Adding Rows to a Table

- Perform a query with DataAdapter.Fill, add records to the resulting DataTable, and write the changes to the database

```
SqlDataAdapter adapter = new SqlDataAdapter ("select * from titles",
    "server=localhost;database=pubs;uid=sa;pwd=");

DataSet ds = new DataSet ();
adapter.Fill (ds, "Titles");

DataTable table = ds.Tables["Titles"];
DataRow row = table.NewRow ();

row["title_id"] = "JP1001";
row["title"] = "Programming Microsoft .NET";
row["price"] = "59.99";
row["ytd_sales"] = "1000000";
row["type"] = "business";
row["pubdate"] = "May 2002";

table.Rows.Add (row);
```

Searching Records in a DataTable

- Use Select

```
DataRow[] rows = table.Select ("title_id = 'JP1001'");

DataRow[] rows = table.Select ("price < 10.00");

// Return all rows where "State" equals CA, TN, or WA
DataRow[] rows = table.Select ("state in ('ca', 'tn',
   'wa')");

// Return all rows where "State" begins with CA
DataRow[] rows = table.Select ("state like 'ca*'");
```


Updating Records in a DataTable

```
SqlDataAdapter adapter = new SqlDataAdapter (
    "select * from titles",
    "server=localhost;database=pubs;uid=sa;pwd=");

DataSet ds = new DataSet ();
adapter.Fill (ds, "Titles");

DataRow[] rows = table.Select ("ytd_sales > 10000");
foreach (DataRow row in rows)
    row["price"] = (decimal) row["price"] + 10.00m;
```


Deleting Records from a Table

```
SqlDataAdapter adapter = new SqlDataAdapter (
    "select * from titles",
    "server=localhost;database=pubs;uid=sa;pwd=");

DataSet ds = new DataSet ();
adapter.Fill (ds, "Titles");

DataRow[] rows =
    table.Select (
    "ytd_sales < 10000 OR isnull (ytd_sales, 0) = 0");
foreach (DataRow row in rows)
    row.Delete ();
```


Propagating Changes

- Inserts, updates, and deletes performed on a DataTable do not automatically propagate back to the database


```
SqlDataAdapter adapter =
    new SqlDataAdapter ("select * from titles",
    "server=localhost;database=pubs;uid=sa;pwd=");

SqlCommandBuilder builder = new SqlCommandBuilder (adapter);
DataSet ds = new DataSet ();
adapter.Fill (ds, "Titles");

DataTable table = ds.Tables["Titles"];
DataRow row = table.NewRow ();
row["title_id"] = "JP1001";
row["title"] = "Programming Microsoft .NET";
row["price"] = 59.99m;
row["ytd_sales"] = 1000000;
row["type"] = "business";
row["pubdate"] = new DateTime (2002, 5, 1);
table.Rows.Add (row);

adapter.Update (table);
```


Update Only Changes

- Could specify that only changes be updated

```
// Update the database
DataTable delta = table.GetChanges ();
adapter.Update (delta);
```

- Not necessary
  - Update is smart enough

CommandBuilder

- How does Update physically update the database?
- Executes SQL commands
  - INSERT, UPDATE, DELETE
- Manufactured by SqlCommandBuilder

DataView

```
SqlDataAdapter adapter =
 new SqlDataAdapter ("select * from titles",
 "server=localhost;database=pubs;uid=sa;pwd=");

DataSet ds = new DataSet ();
adapter.Fill (ds, "Titles");

DataView view = new DataView (ds.Tables["Titles"]);
view.Sort = "title ASC";
MyDataGrid.DataSource = view;
MyDataGrid.DataBind ();
```


Summary

- Completed Chapter 12
- Next Time
  - XML
  - Multithreading