Phi Consulting

.Net Training

ADO.Net (C#)

Agenda

- What is ADO.Net
- Overview of ADO.Net
- ADO.Net vs. ADO
- ADO.Net Architecture
- Databinding

Phi Consulting

What is ADO.Net?

- ADO.NET is a data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways
- The ADO.NET stack has two major parts: providers and services
- ADO.NET "providers" are the components that know how to talk to specific data stores (SQL Server, Oracle, etc.) - All providers surface a unified API on top of which other layers can be built
- ADO.NET also includes services built on top of the providers that are designed to facilitate writing applications

Φ Phi Consulting

Overview of ADO.Net

- ADO.NET separates data access from data manipulation into discrete components that can be used separately or in tandem
- The ADO.NET classes are found in System.Data.dll, and are integrated with the XML classes found in System.Xml.dll
- ADO.NET is an evolution of ADO that provides better platform interoperability and scalable data access
- It is a generic tool belt for raw database access

© 2008 Phi Consulting

Phi Consulting

ADO.Net vs. ADO

- ADO uses a single object, the Recordset, as a common representation for working with all types of data
- Recordset is used for working with a forward-only stream of results from a database, scrolling through data held on a server, or scrolling through a set of cached results
- Changes made to data may be applied immediately to the database, or applied as a batch using optimistic search and update operations
- You specify the desired functionality when you create the Recordset, and the behavior of the resulting Recordset can vary greatly depending on the requested properties

Phi Consulting

ADO.Net vs. ADO

- Because ADO uses a single object that can behave in many different ways, it enables you to keep the object model of your applications very simple
- The trade-off is the difficulty in writing common, predictable, and optimized code because the behavior, performance, and semantics exhibited by that single object can vary greatly

© 2008 Phi Consulting Proprietary and Confidential

ADO.Net vs. ADO

- Rather than using a single object to perform a number of tasks, ADO.NET factors specific functionality into explicit objects that are optimized for each type of task
- The **DataReader** provides fast, forward-only, read-only access to query results
- The **DataSet**, provides an in-memory relational representation of data
- The **DataAdapter**, provides a bridge between the **DataSet** and the data source

Phi Consulting

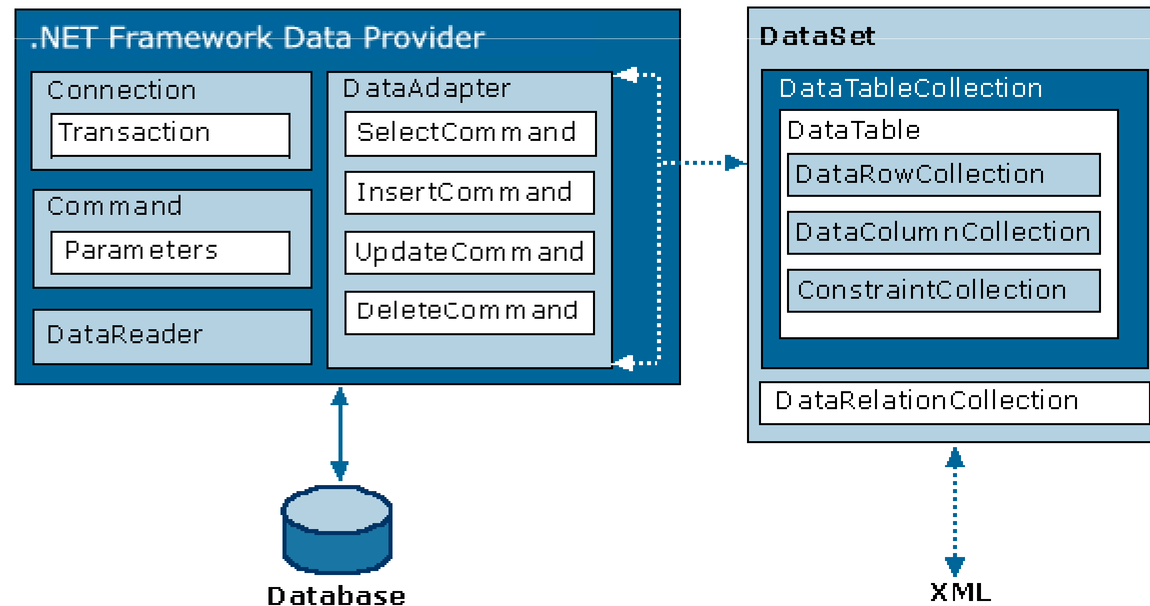
ADO.Net Applications

- There are three main types of tasks when retrieving data from a database:
  - Forward-only, read-only DataStreams
    - For this type of application a DataReader is applicable
  - Disconnected access to data, with batch updates
    - For this type of application a DataSet is applicable
  - Returning a single value
    - For this type of application the Command object supports a method called ExecuteScalar which will return a single value without the need for another object to be created – in ADO you would have to create a RecordSet, read through the results, fetch a single value, then close the RecordSet

© 2008 Phi Consulting

ADO.Net Architecture

- The ADO.Net framework includes the following major components
- Each provider (SQL Server, Oracle, etc.) provides database specific implementations for these objects



© 2008 Phi Consulting

Connection

- Connection objects are Provider specific (SQL Server, Oracle, OLEDB, ODBC, etc.);
- All inherit from System.Data.Common.DbConnection
- Connection objects take in a connection string as a parameter and can open and close database connections to their respective providers
- They are used as the basis for Command objects to operate

Connection

- For valid connection strings use a tool or see www.connectionstrings.com
- E.g.:

```
SqlConnection conn = new SqlConnection("Server=(local); Database=Northwind;
    Trusted_Connection=true;");
conn.Open();
conn.Close();
conn.Dispose();
```

- Tips:
  - Open connections as late as possible
  - Close them as soon as possible
  - Dispose of the connection object as soon as you close it
  - .Net takes advantage of connection pooling

© 2008 Phi Consulting Proprietary and Confidential

Connection Pooling

- Happens behind the scenes
- When a connection is first opened, a connection pool is created based on an exact matching with the connection string in the connection
- Each connection pool is associated with a distinct connection string
- When a new connection is opened, if the connection string is not an exact match to an existing pool, a new pool is created
- Connections are pooled per process, per application domain, per connection string and when using integrated security, per Windows identity

Phi Consulting

Command

- Represents a Transact-SQL statement or stored procedure to execute against a database
- Provider specific (SqlCommand, OracleCommand, OleDbCommand, etc.)
- All inherit from System.Data.Common.DbCommand
- Command objects take as inputs a DbConnection object (provider specific), a command type (text, stored proc) and the command text itself (sql statement or stored proc name)

Command

- E.g.:

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = conn; //assign connection object
cmd.CommandType = CommandType.Text; //CommandType.StoredProcedure
cmd.CommandText = "select * from Orders"; //up_GetOrders
```

- You can optionally include a Transaction object
- Once the command is prepared you can then execute the command
  - cmd.ExecuteNonQuery();
  - cmd.ExecuteReader();
  - cmd.ExecuteScalar();
  - cmd.ExecuteXmlReader();

Phi Consulting

Command

- ExecuteNonQuery()
  - This method is optimized for changing the data in a database without using a DataSet by executing UPDATE, INSERT, or DELETE statements
  - Although it no rows, any output parameters or return values mapped to parameters are populated with data
  - For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command
  - For all other types of statements the return value is -1
  - If a rollback occurs, the return value is also -1

Phi Consulting

Command

- ExecuteScalar()
  - Executes the query, and returns the first column of the first row in the result set returned by the query (Additional columns or rows are ignored)
  - Use the ExecuteScalar method to retrieve a single value (for example, an aggregate value) from a database
  - This requires less code than using the ExecuteReader method, and then performing the operations that you need to generate the single value using the data returned by a SqlDataReader

Phi Consulting

Command

- ExecuteScalar() example:

```
cmd.CommandText = "SELECT COUNT(*) FROM dbo.region";
Int32 count = (Int32) cmd.ExecuteScalar();
```

- Also useful after an Insert to retrieve Identity column

```
cmd.CommandText = “INSERT INTO region (name) VALUES (‘Arizona’);
    SELECT @@IDENTITY";
Int32 regionId = (Int32) cmd.ExecuteScalar();
```

- *Note – there is a more effective way to return an Identity using a stored procedure and output parameters with ExecuteNonQuery()*

Φ Phi Consulting

Command

- ExecuteXmlReader()
  - Sends the CommandText to the Connection and builds an XmlReader object
  - This method is optimized for retrieving XML from the database
  - The **CommandText** property ordinarily specifies a Transact-SQL statement with a valid FOR XML clause
  - However, **CommandText** can also specify a statement that returns **ntext** or **nvarchar** data that contains valid XML, or the contents of a column defined with the **xml** data type (Sql Server 2005)

© 2008 Phi Consulting Proprietary and Confidential

Phi Consulting

Command

- ExecuteReader()
  - This method is optimized for returning a result set
  - It returns a DataReader object for the specific provider (SqlDataReader, OracleDataReader, etc.)
  - If you want to fetch rows of data, this is the method for you
    - E.g. SqlDataReader reader = cmd.ExecuteReader();

DataReader

- Provides a way of reading a forward-only stream of rows from a database
- While the **DataReader** is being used, the associated Connection is busy serving the **DataReader**, and no other operations can be performed on the **Connection** other than closing it
- This is the case until the Close method of the **SqlDataReader** is called
- For example, you cannot retrieve output parameters until after you call **Close**

© 2008 Phi Consulting

Phi Consulting

DataReader

- IsClosed and RecordsAffected are the only properties that you can call after the **DataReader** is closed
- Although the **RecordsAffected** property may be accessed while the **DataReader** exists, always call **Close** before returning the value of **RecordsAffected** to guarantee an accurate return value
- DataReaders are CONNECTED objects – while they are open the connection object is open too
- When the connection is closed the DataReader cannot read and must close too

DataReader

- To read from a DataReader iterate through the rows one by one by calling Read()

```
SqlDataReader reader = cmd.ExecuteReader();
// Call Read before accessing data.
while (reader.Read())
{
    for (int i = 0; i < reader.FieldCount; i++)
    {
        object data = reader.GetValue(i);
    }
}
// Call Close when done reading.
reader.Close();
```

- DataReaders have a HasRows property and a FieldCount property to help the developer

© 2008 Phi Consulting

DataAdapter

- Represents a set of SQL commands and a database connection that are used to fill the DataSet and update the data source
- The **DataAdapter** serves as a bridge between a **DataSet** and a data source for retrieving and saving data
- The **DataAdapter** provides this bridge by mapping Fill, which changes the data in the **DataSet** to match the data in the data source, and Update, which changes the data in the data source to match the data in the **DataSet**
- The DataAdapter is also Provider specific (there's a SqlDataAdapter, OracleDataAdapter, etc.)

Phi Consulting

DataAdapter

- A DataAdapter includes a Connection object and a collection of Command objects:
  – SelectCommand
  – InsertCommand
  – UpdateCommand
  – DeleteCommand
- The DataAdapter works with the various command objects to mediate data access for the DataSet
- It also includes a TableMappings collection to keep track of database table names vs. DataTable names
- Although still supported in .Net 2.0, there is a more modern construct that is preferable – the TableAdapter

Phi Consulting

TableAdapter

- TableAdapters provide communication between your application and a database
- It connects to a database, executes queries or stored procedures, and either returns a new data table populated with the returned data or fills an existing DataTable with the returned data
- TableAdapters are also used to send updated data from your application back to the database
- You can think of a TableAdapter as a DataAdapter with a built-in connection object and the ability to contain multiple queries

© 2008 Phi Consulting

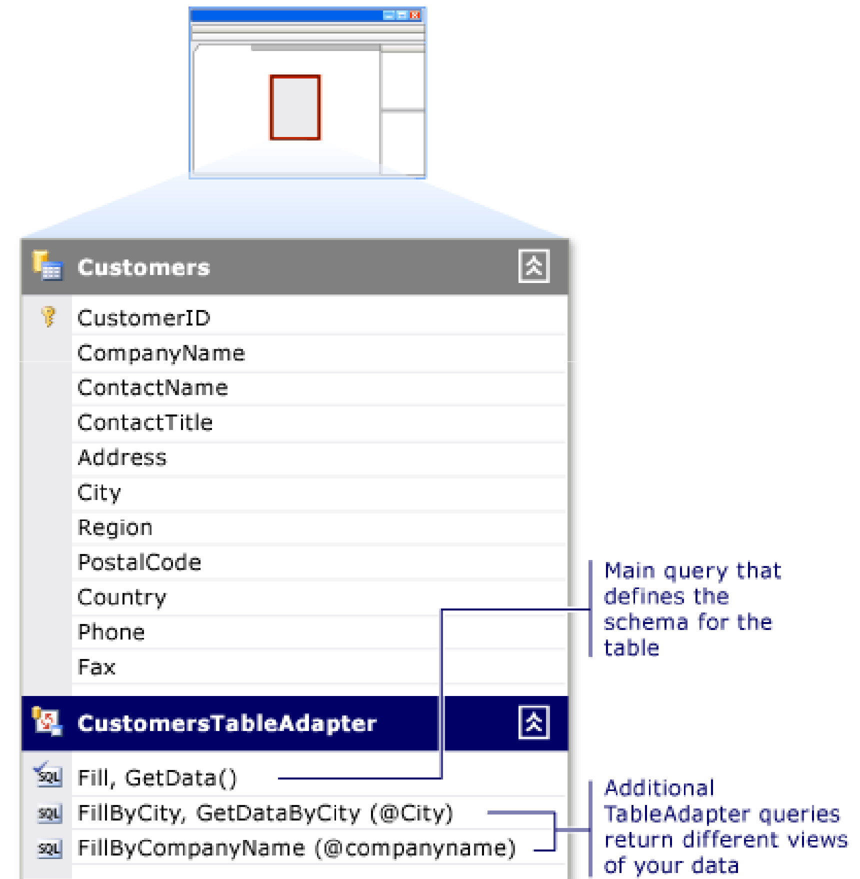
Phi Consulting

TableAdapter

- Each query added to a TableAdapter is exposed as a public method that is simply called like any other method or function on an object
- DataAdapters are limited to single Select, Insert, Update and Delete commands
- The TableAdapter is more versatile

TableAdapter

- In addition to the standard functionality of a **DataAdapter**, TableAdapters provide additional typed methods that encapsulate queries that share a common schema with the associated typed **DataTable**
- In other words, you can have as many queries as you want on a TableAdapter as long as they return data that conforms to the same schema



© 2008 Phi Consulting

Phi Consulting

TableAdapter

- TableAdapters are designer-generated components that improve upon the functionality of **DataAdapters**
- TableAdapters typically contain Fill and Update methods to fetch and update data in a database
- TableAdapters are created with the **Dataset Designer** inside of strongly typed datasets
- You can create TableAdapters during creation of a new dataset with the Data Source Configuration Wizard
- You can also create TableAdapters in existing datasets with the TableAdapter Configuration Wizard or by dragging database objects from **Server Explorer** onto the **Dataset Designer**

© 2008 Phi Consulting

Phi Consulting

TableAdapter

- While TableAdapters are designed with the **Dataset Designer**, the TableAdapter classes generated are not generated as nested classes of the DataSet
- They are located in a separate namespace specific to each dataset
  - For example, if you have a dataset named NorthwindDataSet, the TableAdapters associated with the **DataTable**s in the NorthwindDataSet would be in the NorthwindDataSetTableAdapters namespace

Phi Consulting

TableAdapter

- To access a particular TableAdapter programmatically, you must declare a new instance of the TableAdapter. For example:

```
NorthwindDataSet northwindDataSet = new NorthwindDataSet();

NorthwindDataSetTableAdapters.CustomersTableAdapter customersTableAdapter =
    new NorthwindDataSetTableAdapters.CustomersTableAdapter();

customersTableAdapter.Fill(northwindDataSet.Customers);
```

Phi Consulting

TableAdapter

- Changing the schema:
  - When creating a TableAdapter, the initial query or stored procedure is used to define the schema of the TableAdapter's associated **DataTable**
  - You execute this initial query or stored procedure by calling the TableAdapter's main Fill method (which fills the TableAdapter's associated **DataTable**)
  - Any changes made to the TableAdapter's main query are reflected in the schema of the associated data table
  - For example, removing a column from the main query removes the column from the associated data table
  - If any additional queries on the TableAdapter use SQL statements returning columns that are not in the main query, then the designer will attempt to synchronize the column changes between the main query and any additional queries

© 2008 Phi Consulting Proprietary and Confidential

Phi Consulting

TableAdapter

- TableAdapters come closer that DataAdapters to fulfilling the promise of true business object
- But they fall short on multiple counts when compared to true business objects
  - There's no place on DataSets for business rules – these often end up mired in presentation layer code
  - Business objects have better encapsulation and extensibility than their DataSet counterparts - Code reuse lowers project costs
  - Business Objects are easier to regression test

Phi Consulting

DataSet

- Represents an in-memory cache of data
- It is disconnected in nature
- A DataAdapter or a TableAdapter is typically used to populate a DataSet
- Can have a collection of DataTables and DataViews inside it
- DataTables can be related via DataRelations
- You can also enforce data integrity in the **DataSet** by using the UniqueConstraint and ForeignKeyConstraint objects

Phi Consulting

DataSet

- A DataSet can be serialized and deserialized to and from XML
- A DataSet also keeps track of changes (inserts, deletes, updates) and you can fetch separate collections for the changed values or the original values
- The DataSet is in essence an in-memory, disconnected, representation of a database or portion of a database that can be operated on in batch process fashion

© 2008 Phi Consulting Proprietary and Confidential

DataSet

- DataSets can be Typed or Untyped
- Untyped DataSets are a generic collection of DataTables, whose names must be referred to with strings
- Columns must also be referred with strings
- This is prone to error and database changes will only be detected as runtime errors
- Typed DataSets are generally preferable because tables, columns are strongly named at design time – less prone to errors

DataSets – putting it all together

- See demo on creating a generic DataSet through code
- See demo on creating a Typed DataSet through the designer

Phi Consulting

Data Binding

- Once you have the DataSet/DataTables you can Databind to presentation elements
- There are two kinds of Databinding
- Simple Databinding – binding to simple controls (TextBox, single select ComboBox ,etc.)
- Complex Databinding – binding to complex controls (ListBox, grids, etc.)

Phi Consulting

What is Data Binding?

- The process or method for configuring controls on a form or Web page to fetch data from or write data to a data source such as a database, XML file, and so on
- You basically bind a data source to a UI element or elements and changes in one are affected in the other
- Through a series of events (known as the Model-View-Controller pattern or MVC) the UI element and the data store notify eachother

Phi Consulting

Data Binding

- Data binding works slightly differently whether you are binding to a Windows UI or Web UI
- Most web server-side controls can be databound
- There are exceptions (Panels, Literals, Buttons, TreeViews etc.)
- Databinding works differently in 2.0 and 1.x
- We will focus primarily on databinding to ASP.Net 2.0
- Besides binding to DataSets/DataTables we can also bind to Business Objects – these must implement certain interfaces to allow the data binding to work
- The later is a complex subject and we will not cover it

Data Binding

- See data binding demos

© 2008 Phi Consulting Proprietary and Confidential

Phi Consulting

Business Objects

- DataSets, TableAdapters, Databinding and UI Wizards are a great development tool
- But for the serious programmer they soon begin to break down as you encounter their inherent limitations
- The main reason is lack of support for business rules
- Real world applications have business rules that go above and beyond what wizards can do

© 2008 Phi Consulting Proprietary and Confidential

Business Objects

- Let's say you have web site that sells memberships – say it's a Match.com membership that lasts 90 days from date of payment
- Let's say you joined on 6/1/2007
- But instead of paying right then by CC online you decided to pay by phone and you did so 3 days later (6/4/2007)
- The web page will set your Order Date to 6/1/2007 with a payment status of "Unpaid"
- But 3 days later you pay by phone – you have lost 3 days of membership…
- Shouldn't the Order Date be 6/4/2007?

Phi Consulting

Business Objects

- This is where a Business Object can come in handy
- You can write complex rules that are separate from the Data Access Layer code
- So when the admin updates the order with the payment information a business rule can trigger that can detect when the order changes from “Unpaid” to “Paid by Phone” and then sets the Order Date to the date this event occurred

Phi Consulting

Business Objects

- DataSets and TableAdapters will not help you do this
- You often end up writing that code in the Presentation Layer – where it doesn't belong

What you need is a Business Layer

Phi Consulting

Business Objects

- Business Objects are easy to write
  - Create properties that mimic table data and load with DataReaders
  - They are a little harder to make databindable
  - You have to implement many interfaces
- Fortunately there are lots of code-generators out there that make the task easier for you
- Many claim faster performance that DataSets (some claim as much as 30x)

That is the subject of **O**bject **R**elational **M**apping

© 2008 Phi Consulting