# Database Programming

Chapter 4 (Part 1)

# What Are You Going To Learn?

- At the end of this lesson, you will be able to:
  - Explain data-driven web page.
  - Use ASP.NET database namespaces and custom-built objects (ADO.NET)
  - Discuss the issues in data-driven page
  - Apply parameterized query method.
  - Create, retrieve, update and delete (CRUD) records on database.

# Data-Driven Web Page (Advantages)

**Web pages using databases and other sources of data** to turn static web pages into dynamic data-driven web pages.

| | |
|---|---|
| Maintainability | • easier to maintain data and keep it up-to-date. |
| Reusability | • information in databases can easily be backed up and reused as required. |

# Data-Driven Web Page(Advantages)

| | |
|---|---|
| Data context | • database allows to define relationship and rules for the data in the database. |
| Quality and timeliness of content | • databases are optimized for the storage and retrieval of data. They allow you to use and update information on live Web site almost in real time. |

TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

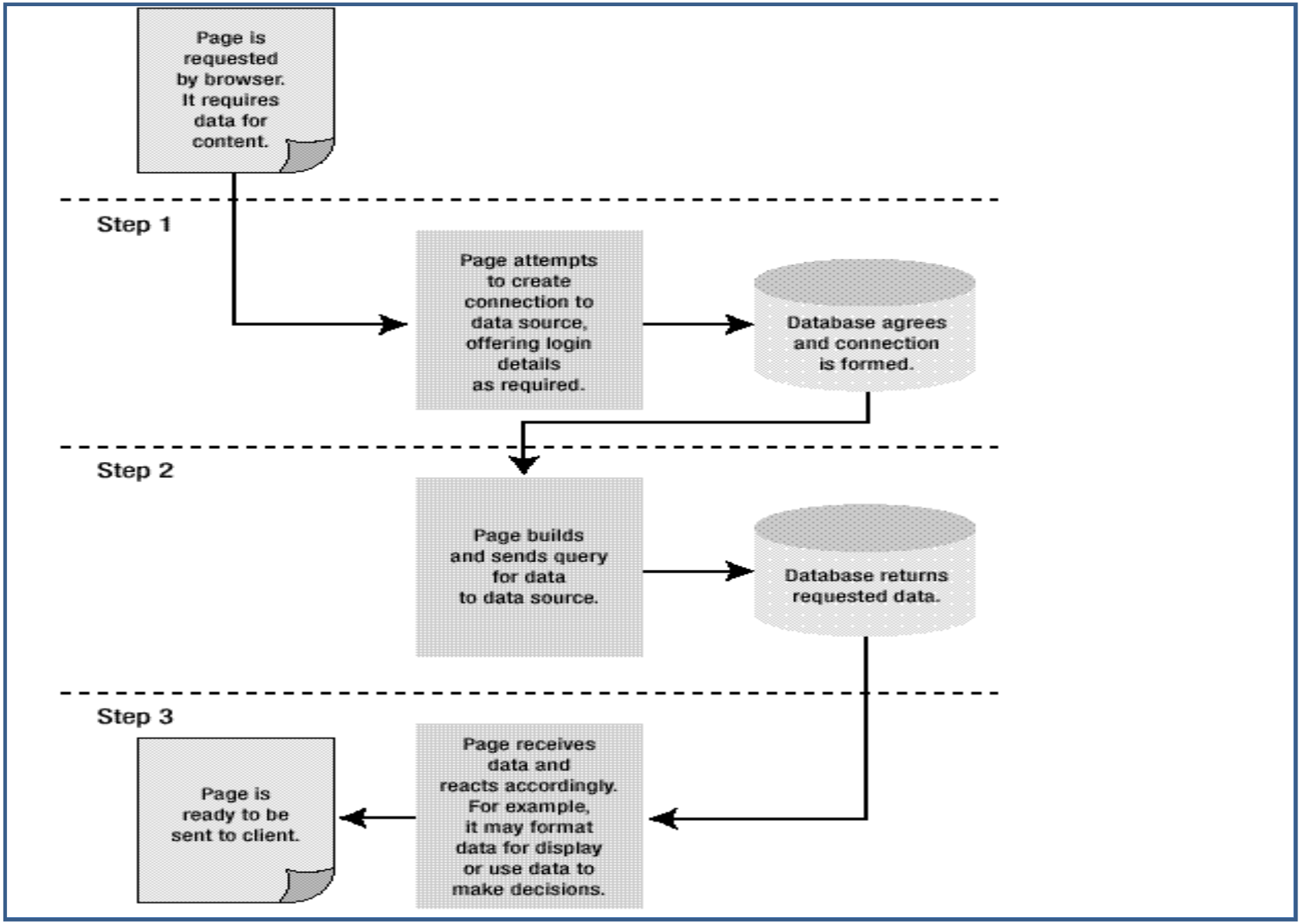
# Disadvantages

| | |
|---|---|
| Development time | • it takes a little more time to write code to access the database. |
| Dependency on the database | • The whole Web site will fail if the database fail for some reason. |

# Disadvantages

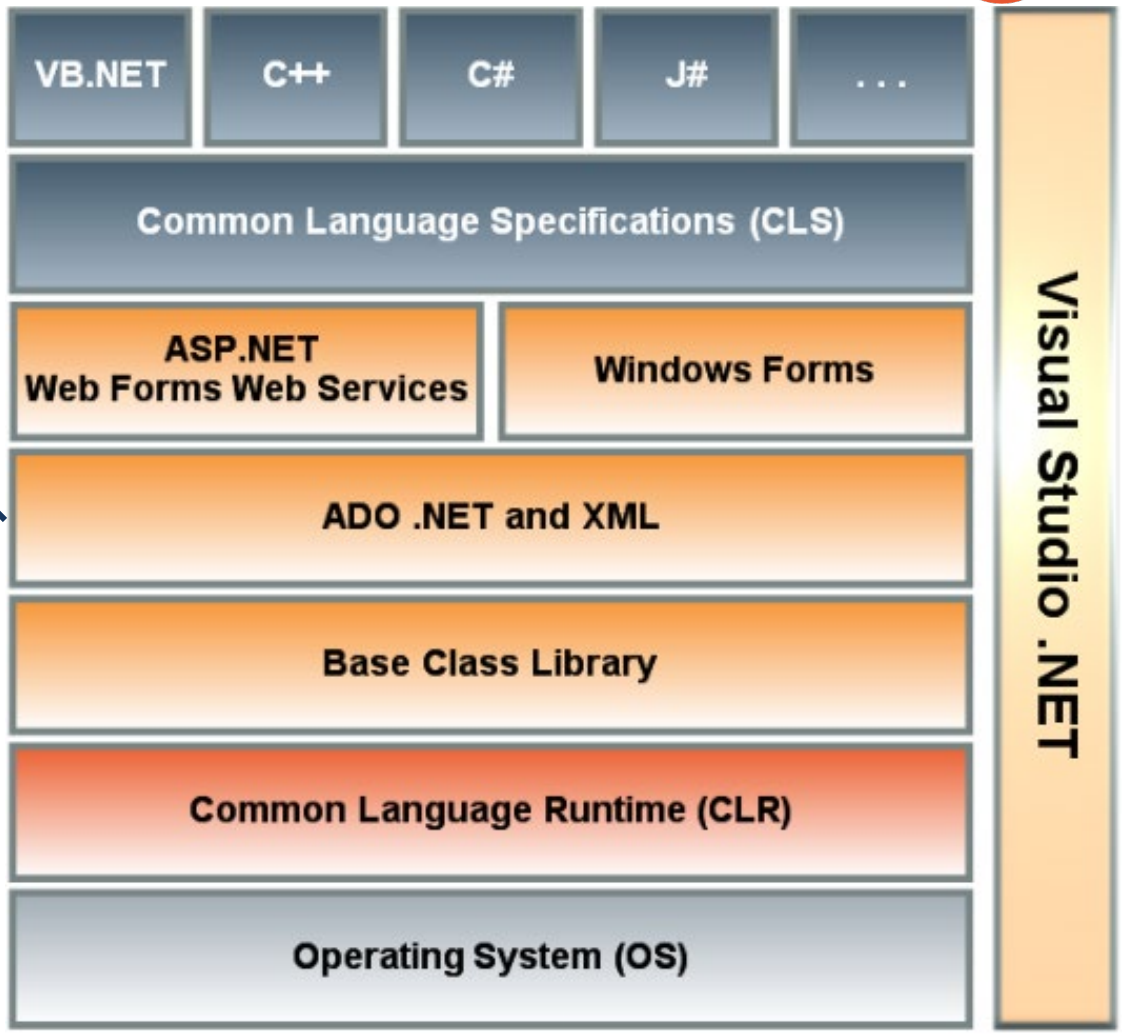
| | |
|---|---|
| Database round-trip | • The round-trip between client and server means a slight reduction in performance |
| Cost | • Full enterprise-level database solution are quite expensive. |

# How Does The Web Site Get The Data?



# ADO.NET – ASP.NET Data Objects

ADO.NET is a part of the base class library that is included with the Microsoft .NET Framework.



# ADO.NET – ASP.NET Data Objects

- A set of computer software components that can be used by programmers to access data and data services.
- commonly used by programmers to access and modify data stored in relational database systems, though it can also be used to access data in non-relational sources.
- ADO.NET Data Services has been renamed as WCF data services in 2009 (aka Astoria).

# ADO.NET – Namespaces

- The .NET framework contains several namespaces with dozens of classes devoted to database access.
- Microsoft has created separate namespaces that are optimized for working with different data providers (different types of databases).

# ADO.NET - Namespaces

## System.Data.SqlClient:

- Contains classes for connecting to Microsoft SQL Server version 7.0 or higher

## System.Data.OleDb:

- Contains classes for connecting to a data source that has an OLE DB provider (such as Ms Access).

# ADO.NET - Namespaces

## System.Data.Odbc:

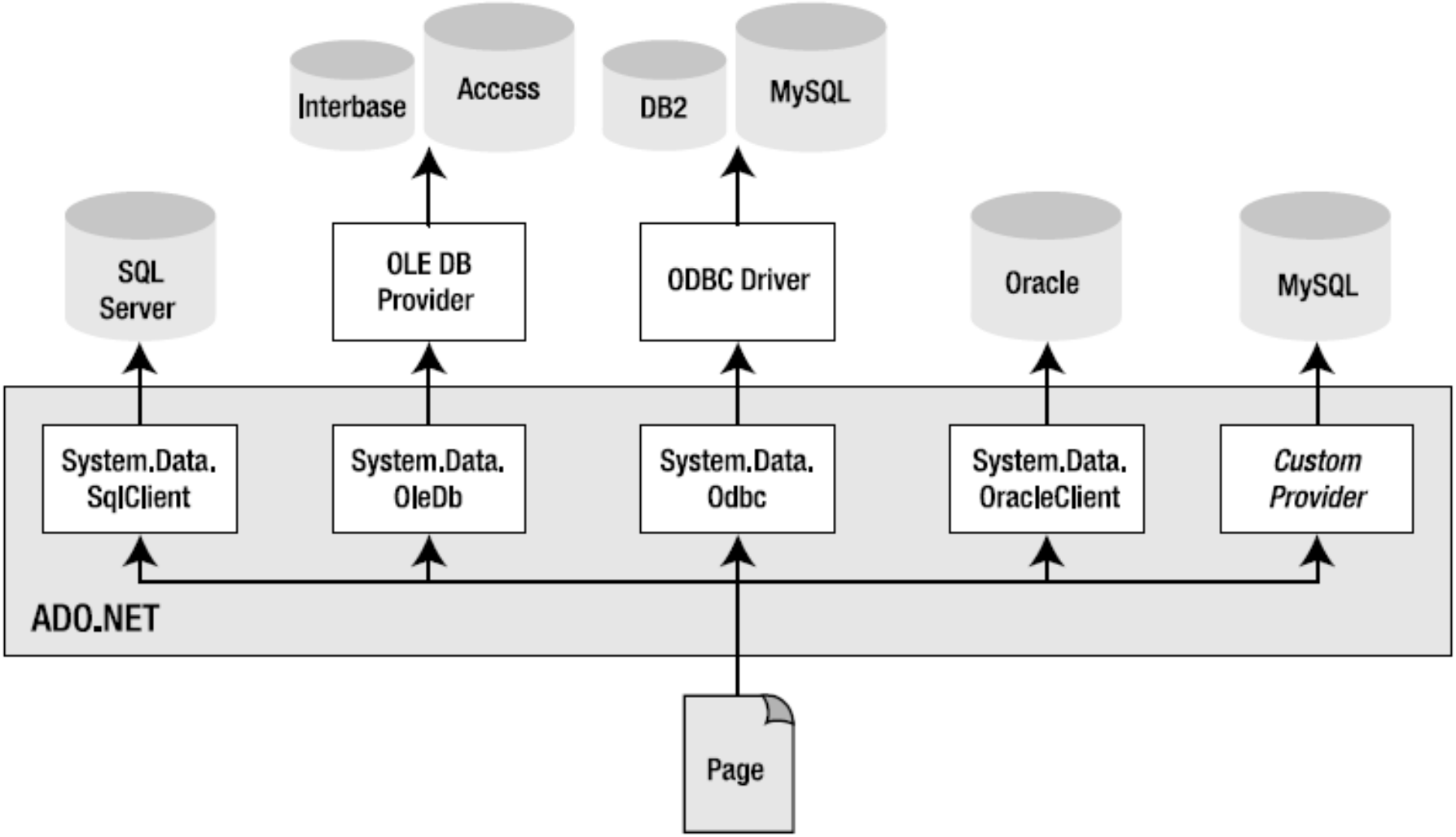
- for a data source that has an ODBC driver.

## System.Data.OracleClient:

- for Oracle database server.

## System.Data.SqlServerCe:

- for SQL Server CE.




*Using data providers means you have easy, optimized access to all databases.*

# ADO.NET - Common Data Classes

Each data source has its own set of provider objects, but they each have a common set of utility classes as follow:

## Connection object:

- Provides a connection used to communicate with the data source.

## Command object:

- Used to perform some action on the data source, such as reading, updating, or deleting relational data.

## DataAdapter object:

- A bridge used to transfer data between a data source and a DataSet object

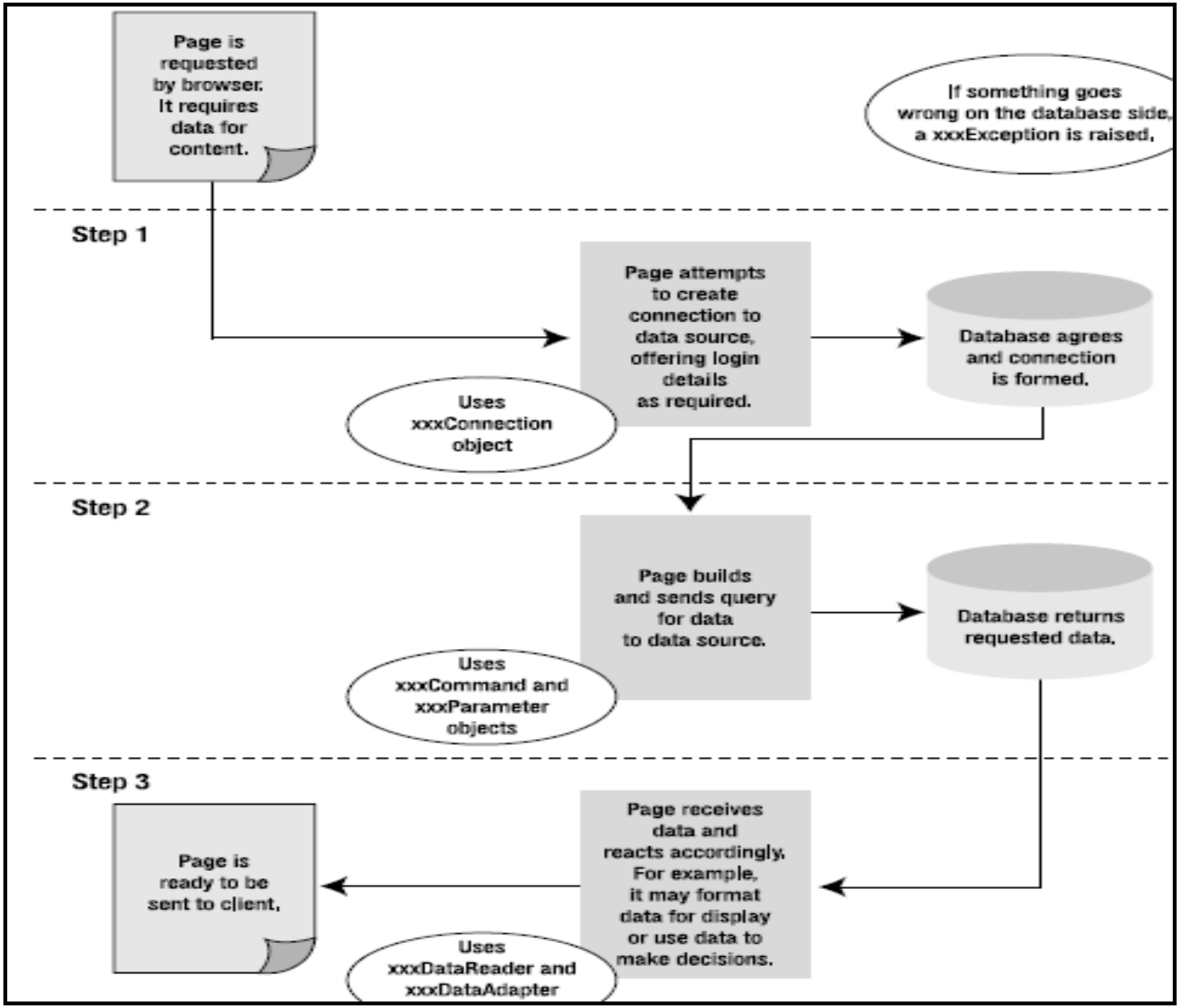
# ADO.NET - Common Data Classes

## DataReader object:

- Used to efficiently process a large list of results one record at a time. It allows records to be accessed in a read-only, forward-only mode, i.e., records have to be accessed in sequential order; they can neither be randomly accessed nor can a record which has been processed previously be accessed again.

## Parameter object:

- Describes a single parameter to a command.


Page is requested by browser. It requires data for content.
If something goes wrong on the database side, a xxxException is raised.
Step 1
Page attempts to create connection to data source, offering login details as required.
Database agrees and connection is formed.
Uses xxxConnection object
Step 2
Page builds and sends query for data to data source.
Database returns requested data.
Uses xxxCommand and xxxParameter objects
Step 3
Page receives data and reacts accordingly. For example, it may format data for display or use data to make decisions.
Page is ready to be sent to client.
Uses xxxDataReader and xxxDataAdapter

# Question

Identify the data objects (connection, command, Parameter, DataReader) that are needed by:

1. A Search engine page.
2. A login page.
3. A page that allows you to add a new product's details.
4. A page that allows you to edit your personal details.

# Which Data Type to Use?

| SQL 2005 Data Type | Description |
|---|---|
| bit | Stores Boolean values in a 0 / 1 format |
| char / nchar | Contains fixed-length text. When you store text shorter than the defined length, the text is padded with spaces. The nchar stores the data in Unicode format, which allows you to store data for many foreign languages. |
| datetime | Stores a date and a time. |

- Example:
  - Use *bit* to store 0/1 (e.g. 0=absent; 1=present)
  - Use *char* to store student's registration number
  - Use *nchar* to store Chinese character in fixed length

# Which Data Type to Use?

| | |
|---|---|
| decimal | Allows you to store large, fractional numbers. |
| float | Allows you to store large, fractional numbers. |
| image | Allows you to store large binary objects such as files. Although the name suggests that you can only use it to store images, this is not the case. You can use it to store any kind of document or other binary object. |

- *Decimal* storage size is 17 bytes. Maximum precision and scale goes up to - $10^{38}$ +1 through $10^{38}$ - 1. Used for fixed precision and scaled numeric data. E.g. decimal(5,2) stores 123 as 123.00 (precision of 5, scale of 2)
- Default *float* precision is 15, storage size is 8 bytes.
- Image stores variable-length binary data from 0 through $2^{31}$-1 (2,147,483,647). Will be obsolete in the future (avoid using this)

# Which Data Type to Use?

| Data type | Range | Storage |
|---|---|---|
| **bigint** | -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807) | 8 Bytes |
| **int** | -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647) | 4 Bytes |
| **smallint** | -2^15 (-32,768) to 2^15-1 (32,767) | 2 Bytes |
| **tinyint** | 0 to 255 | 1 Byte |

# Which Data Type to Use?

| | |
|---|---|
| `tinyint` | Used to store integer numbers ranging from 0 to 255. (1 byte) |
| `smallint` | Used to store integer numbers ranging from -32,768 to 32,767. (2 bytes) |
| `int` | Used to store integer numbers ranging from -2,147,483,648 to 2,147,483,647 (4 bytes) |
| `bigint` | Used to store large integer numbers ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. (8 bytes) |

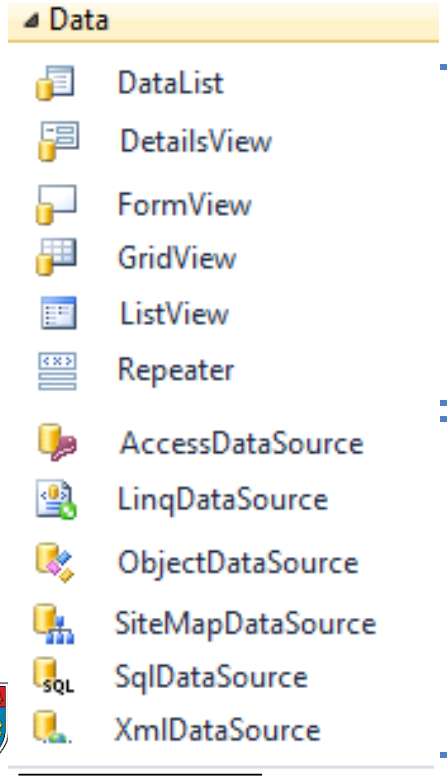
What data type would you use to store Bill Gate's net worth (income) in 2014?

# Which Data Type to Use ?

- Selecting the appropriate data type for your application will help your database to function more correctly.
  - Too large of a data type : wasting space
  - Too small of a data type: artificial ceiling
  - incorrect type : required data type conversion
  - incorrect type : makes reporting more difficult
    - Example: Zip code, money

# Connecting to Database in ASP.NET

- A simple way is to use data source controls, which allow you to encapsulate data access in a control that you can configure with connection and query information.




Namespace: System.Web.UI.WebControls

data bound controls

data source controls

# Data Source Control

- collection of Web controls designed to provide a declarative approach to accessing and modifying data.
- enable rich capabilities for retrieving and modifying data, including querying, sorting, paging, filtering, updating, deleting, and inserting.
- you can work with data without having to write a lick of data access code.

# Data Source Control

- The built-in data source controls
  - SqlDataSource
  - AccessDataSource
  - ObjectDataSource
  - XmlDataSource
  - SiteMapDataSource
  - LinqDataSource
  - EntityDataSource (not in syllabus)

# SqlDataSource Control

The "Sql" in the control name does not refer to Microsoft SQL Server, but rather the SQL syntax for querying **relational databases.**

- SqlDataSource control can be used to access not only Microsoft SQL Server databases, but Microsoft Access databases, Oracle databases and basically any OLE-DB or ODBC-compliant data store.

# Connecting to Database in ASP.NET

- Alternatively, we can code to access the database for customized requirements (which cannot be fulfilled by the Data Access Web Controls.

```
private static void ReadOrderData(string connectionString)
{
        string queryString = "SELECT OrderID, CustomerID FROM
        dbo.Orders;";
        using (SqlConnection connection = new      SqlConnection(
connectionString))
        {
        SqlCommand command = new SqlCommand( queryString, connection);
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
```

# System.Data.SqlClient

The `System.Data.SqlClient` namespace include the following classes :

## SqlConnection:

- Represents an open database connection to a database.

## SqlCommand:

- to execute a SQL statement against a SQL Server database.

# System.Data.SqlClient (cont..)

## SqlParameter:

- Pass parameter values to a SQL command. Parameters are commonly used to limit the number of row retrieved by a Select statement

## SqlDataReader:

- To create a data reader object, which provides an efficient way to read the rows in a result set resulted by a database query

## SqlDataAdapter:

- to provide a link between a database and dataset

# Common Database Task

- In this section, you will learn to perform common database tasks using the ADO.NET's data objects:
  - Create & open a database connection.
  - **C:** Create/Add new database records.
  - **R:** Retrieve & display database records.
  - **U:** Update existing database records.
  - **D:** Delete database records.

# Configuring `Web.config`

placed inside the <Configuration> tag

```
<?xml version="1.0"?>
<configuration>
<connectionStrings>
      <add name=" WebConfigConString"
connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\
Authors.mdf;Integrated Security=True“
providerName="System.Data.SqlClient" />
</connectionStrings>
<system.web>
</system.web>
</configuration>
```

placed before the <system.web> tag

# Storing *ConnectionString* in *web.config* File

- you can secure the connection string using encryption.

<u>Question:</u>

- Discuss another advantage of storing the connection string in the `web.config` file.

# Reference: connectionstrings.com

www.connectionstrings.com/sql-server/

Developers number one | Connection Strings reference | Knowledge Base | Q & A forums | log in | About | Thank You | Search | enter keywords . . .

## SQL Server connection strings

ConnectionStrings.com

**.NET libraries**

- .NET Framework Data Provider for SQL Server
- Context Connection

**OLE DB providers**

- SQL Server Native Client 11.0 OLE DB Provider
- SQL Server Native Client 10.0 OLE DB Provider
- SQL Native Client 9.0 OLE DB Provider
- Microsoft OLE DB Provider for SQL Server
- SQLXML 4.0 OLEDB Provider
- SQLXML 3.0 OLEDB Provider

**ODBC drivers**

- SQL Server Native Client 11.0 ODBC Driver
- SQL Server Native Client 10.0 ODBC Driver
- SQL Native Client 9.0 ODBC Driver
- Microsoft SQL Server ODBC Driver

**Wrappers and others**

- MSDataShape
- .NET Framework Data Provider for OLE DB
- .NET Framework Data Provider for ODBC



### .NET Framework Data Provider for SQL Server

#### Standard Security

```
Server=myServerAddress; Database=myDataBase; User Id=myUsername;
Password=myPassword;
```

### Connect

SQL Server ×89
SQL Server 2012 ×38
SQL Server 2008 ×33
SQL Server 2005 ×51

# DEMO

Creating New Database Table

Using SQL Server Express

# Retrieve & Display Database Records

- 4 steps to retrieve database records:
  1. Create & open a database connection.
  2. Create SQL **Select statement** and **SqlCommand** object.
  3. Execute the Command to retrieve and store the records in the memory (with DataReader or variable)
  4. **Display** the results of the query.
  5. Close the database connection

# Retrieve & Display Database Records

- Step 1: Create & open a database connection

- First, include the namespace

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

- Method 1: Place the connection string on **Web Form**

```
SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\
Authors.mdf;Integrated Security=True");
```

# Step 1: Create & open a database connection

- Method 2: The connection string is in the **web.config**

Web Form

```
string strCon = ConfigurationManager.ConnectionStrings["
WebConfigConString "].ConnectionString;
SqlConnection con = new SqlConnection(strCon);
```

web.config

```
<?xml version="1.0"?>
<configuration>
<add name="WebConfigConString" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\A
uthors.mdf;Integrated Security=True"/>
<system.web>
</system.web>
</configuration>
```

# Step 1: Create & open a database connection

- To open a database connection
- Syntax:

```
connectionName.Open();
```

- Example

```
con.Open(); //con is created in Step 1
```

- Similarly, to close a database connection

```
con.Close();
```

# Retrieve & Display Database Records

- Step 2: use the **`Select`** SQL statement to query the records from the connected database.
- Syntax (just in case you have forgotten about it ☺ ):

```
Select column1, column2…
From table1, table2….
Where search_condition
```

- Example

```
string strSelect = "Select fname, lname
From Students where lname = 'Wong'";
```

# Retrieve & Display Database Records

- Step 2: Create an **SqlCommand** object

```
SqlCommand cmdXXX =
new SqlCommand(SQL_statement,
connection_object);
```

The SQL Select Statement created in Step 2

the connection object created in Step 1

- Example

```
SqlCommand cmdSelect=
new SqlCommand(strSelect, con);
```

# Retrieve & Display Database Records

- Step 3: Execute Command to retrieve records
- **Method 1: 1 or more records are returned**
- Use **SqlDataReader** object (to temporary store the records that retrieved by the Command object).
- Syntax:

The command object is created in Step 2

```
SqlDataReader dtrXXX = commandName.ExecuteReader();
```

- Example

```
SqlDataReader dtrStud= cmdSelect.ExecuteReader();
```

# DataReader

- It represents forward-only stream of database records.
- It processes ONE single record at a time.
- Once you pass a records, there is no going back.
- It does not have a property that returns a count of records.

# DataReader – Read()

- Use `Read()` method to fetch the next record in the stream.
- To display all the records returned from a query, use loop to Read the datareader until the end of the stream.

```
SqlDataReader dtrStud= cmdSelect.ExecuteReader();
while (dtrStud.Read())
{
    Response.Write(dtrStud["lname"] + dtrStud["fname");
 }
```

# DataReader - HasRows

- Checking whether any row (of records) is returned .
- Unlike the Read() method, it does not advance the DataReader to the next row

```
if(dtrStud.HasRows)
{
    while (dtrStud.Read())
    {
        Response.Write(dtrStud["lname"] + dtrStud["fname");
    }
}
else
Response.Write ("No records retrieved");
```

# Retrieve & Display Database Records

- Step 3: Execute Command to retrieve records
- **Method 2: Only 1 data is returned**
- Using a **variable** to store the data is sufficient.
  - Example: Retrieve a user's password.
  - Retrieve a single value concern aggregate functions:
    - SQL Server Express supports several aggregate functions in a SQL statement, such as count(*), sum(field), avg( ), min( ), max( ).

# Retrieve & Display Database Records

- use the **ExecuteScalar ( )** method (from SqlCommand class). This method returns *object* type.
- ExecuteScalar ( ) returns the value of the first column of the first row returned by a query.

```
string strCountProd = "Select count(*) From Product";
SqlCommand cmdCount = new SqlCommand(strCountProd, con);
int count= (int)cmdCount.ExecuteScalar();
```

# Question

- Can you forecast the potential problem in this code?

*ExecuteScalar( )* returns the value of the first column of the first row returned by a query.

```
string strGetPW = "Select * From Students
                        where username='Eric Tan' ";
SqlCommand cmdGetPW = new SqlCommand(strGetPW, con);
object  getPW= cmdGetPW.ExecuteScalar();
string password;
if(getPW != null)
        password = getPW.ToString();
else
        Response.Write("Invalid username");
```

# DEMO

Retrieving Database Record

# Question

- What are the general procedures to retrieve database record(s)?
- What are the differences between `ExecuteReader()` and `ExecuteScalar()` methods?
- What execute statement to use?
  - To retrieve the price when the product code is entered.
  - To retrieve the shops nearby based on user's current location
  - To retrieve the top 10 movies in this week

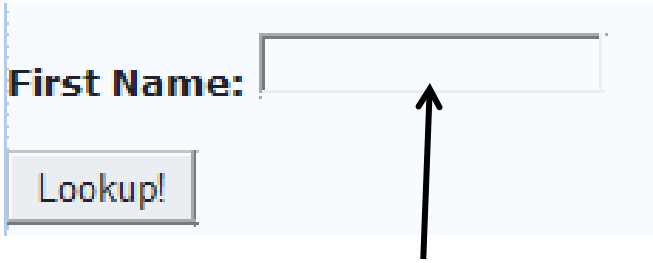
# Retrieving Information Based on User's Input

- 2 methods:
  1. Using string concatenation.
  2. Using the Parameterized query method

# Retrieving Information Based on User's Input

- **Method 1: Using String Concatenation Method**
  - the SQL Select statement is concatenate with value that pass from the server control.

```
strSelect = "Select lastname From
Author Where firstname='" +
txtFirstname.Text + "'";
```




txtFirstname

# Retrieving Information Based on User's Input

- **Method 2: Using Parameterized Query Method**
- Pass parameters with SqlParameter (from SqlCommand class)
- Syntax

*Commandobj*.Parameters.AddWithValue ("*@Param*", *value*);

- Example

```
strSelect = "Select * from Student Where fname=@firstname";
SqlCommand cmdSelect = new SqlCommand(strSelect, con);
CmdSelect.Parameters.AddWithValue("@firstname", "Alex");
```

# Using Parameterized Query Method

- We do not need to specify the data type of the parameter, which it is automatically inferred from the value assigned to the parameter.

```
CmdSelect.Parameters.AddWithValue("@firstname", "Fred");
```

- In the example above, the value Fred is a *String*, the data type *varchar* is inferred.

# Using Parameterized Query Method

- Advantages:

1. Using parameterized query makes tasks for doing query much easy and simple
2. Your query is more readable
3. Preventing SQL Injection attacks

# DEMO

SQL Injection

# Issues in Database Programming: SQL Injection

- is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.
- The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.

# Issues in Database Programming: SQL Injection

effective measures that can be adopted to prevent SQL injection attacks:

| | |
|---|---|
| Prevent unauthorized access | • limit the permissions that are granted to the database user account that the application uses. |
| Avoid disclosing technical errors | • Avoid displaying the actual database errors or messages to the end users. |

# Issues in Database Programming: SQL Injection

| | |
|---|---|
| Validate user input | • stripping off the potentially malicious characters; Check for known **good data** by validating for type, length, format, and range. |
| Use parameterized query | • Always use parameterized SQL queries and stored procedures rather than string concatenation |

# SQL Injection and Parameterized Query

Parameterized query is considered the best defense against SQL injection

| | |
|---|---|
| Provide type checking and length validation | • provide type checking and length validation.<br>• Values outside of the range trigger an exception. |
| The input is treated as literal value | • SQL Server does not treat the input value as executable code. |

# DEMO

Using Parameterized Query

# Next Week

- CUD (Create, Update and Delete) data from the database
- Advanced Data Handling