

## 29. Using ODBC to Connect to a Database

On the server, a database is associated with a DSN (“Data Source Name”).

The Perl package `Win32::ODBC` enables Perl programs to connect to ODBC data sources.

The following statement includes this package:

```
use Win32::ODBC;
```

To connect to the ODBC data source, pass the DSN to the constructor `Win32::ODBC` and apply `new` to this.

This returns a reference to an instance of `Win32::ODBC`.

If the attempt to connect to the data source fails, the result will be empty, which counts as false.

Example:

```
if ( !($Data = new Win32::ODBC( "Products" )) ) {
    - Error-handling code -
}
```

Method

```
Win32::ODBC::Error()
```

returns the error that occurred when an attempt to connect fails.

Method `Close()` of a `Win32::ODBC` object closes the data source – e.g.,

```
$Data->Close();
```

Method `Sql(...)` of a `Win32::ODBC` object takes a SQL query string and issues the query.

If this is successful, then

- the object contains the record set of the result, and
- the method returns an empty reference (which counts as false).

Example:

```
if ( $Data->Sql( $querystring ) ) {
    - Error-handling code -
}
```

Method `Error()` of a `Win32::ODBC` object returns the error that occurred when an attempt to issue a query fails – e.g.,

```
$Data->Error();
```

Method `FetchRow()` accesses one row in the query result.

Since it returns an empty reference when used after the last row has been accessed, it can be used to control a loop that iterates over all rows in the query result.

Example:

```
while ( $Data->FetchRow() ) {
    - Process a row -
}
```

Method `DataHash()` returns the current row as a hash.

The keys are the table fields (columns) and their values are the values for the fields in the current row.

Example:

```
%Data = $Data -> DataHash();
```

We can then use a `foreach` loop to iterate over all the keys.

We can access the values by indexing the hash with the keys.

Example:

```
foreach $key ( keys( %Data ) ) {
    print "<td>$Data{$key}</td>"
}
```

*Example:*

Here a client user constructs a query in a textbox in a form.

When the form is submitted, a Perl program issues the query and sends the results back to the client.

We have an HTML document with a form that contains a textbox element.

The user is expected to enter a query (that makes sense for the database) in the textbox and click the submit button.

The value of the `action` attribute of the form is  
"cgi-bin/data.pl"

Perl file `data.pl` copies the query from the client, connects to the data source in question, and issues the query.

It loops over each row of the query result, constructing a table rendered by the client's browser.

The following is the HTML document.

```
<html>
<head><title>Sample Database Query</title></head>
<body>
  <p>Querying an ODBC database.</p>
  <form method = "POST" action = "cgi-bin/data.pl">
    <input type = "text" name = "QUERY" size = 40
      value = "SELECT * FROM AUTHORS"><br><br>
    <input type = "submit" value = "Send Query">
  </form>
</body>
</html>
```

The following is file data.pl.

(At the end, we use `end_html` from the CGI library instead of the closing HTML tag.)

```
use Win32::ODBC;
use CGI qw/:standard/;

my $querystring = param(QUERY);
$DSN = "Products";

print header;

if (!( $Data = new Win32::ODBC($DSN) ))
{
    print "Error connecting to $DSN\n";
    print "Error: " . Win32::ODBC::Error() . "\n";
    exit;
}

if ( $Data->Sql($querystring) )
{
    print "SQL failed.\n";
    print "Error: " . $Data->Error() . "\n";
    $Data->Close();
    exit;
}

print "<body>";
print "<p> Search Results </p>";

$counter = 0;

print "<table>";
```

Continued next page

## Continued from previous page

```
while($Data->FetchRow())
{
    %Data = $Data->DataHash();
    print "<tr>";

    foreach $key( keys( %Data ) )
    {
        print "<td>$Data{$key}</td>";
    }

    print "</tr>";
    $counter++;
}

print "</table>";
print "<br>Your search yielded <b>$counter</b>",
    " results.";
print end_html;

$Data->Close();
```