Introduction

The Nozicle web components access scripts running on a web server. The scripts can be written in any web scripting language although the scripts supplied are written in PHP.

The TNozicleDataset component is a TDataset descendant with NO dependency on the BDE. It communicates with the scripts db_mysql.php and db_mysql_update.php for data access. As it is a TDataset descendant, all data aware controls can be used, including quick reports. TNozicleDataset allows you to run SQL against a database hosted on the web and return the results as a dataset. Essentially it allows you to use a standard web server as a middle tier. The only requirements of the server are that it is able to run PHP scripts and MySQL is installed. No special server software or DLLs are required. Thus the webserver is platform independent. In fact the demo databases hosted at www.nozicle.com/demodb are running on RedHat Linux using Apache.

The TNozicleServerScript is used to pass arbitrary parameters to a script and receive the results. The parameters are passed as an http 'POST'. A stringlist is returned with the results.

Both TNozicleDataset and TNozicleServerScript access a web site via the TNozicleWebConnection component. This component hold properties such as site URL, login, password, proxy parameters etc.

Hosting Requirements

The complete requirements for a webserver hosting a database for access by the Nozicle Web Components are:

1. Standard web server such as Apache (www.apache.org) or Microsoft iis (installed with windows).
2. PHP (www.php.net). The PHP scripts are included in the distribution of the Nozicle components. These can be rewritten in any other scripting language providing the output is identical to the supplied scripts.
3. MySQL (www.mysql.com). The supplied PHP scripts assume access to MySQL. It is reasonably straight forward to modify the scripts to access any other database (Firebird/Interbase, Oracle, MS SQL Server etc).

Installation

1. Unzip the package to a directory.
2. Open the 'DPK' file in the 'source' directory for your version of Delphi (e.g "Nozicle_WebAccess_D5.dpk" for Delphi 5).
3. Click on 'Install'.

TNozicleWebConnection

The TNozicleWebConnection component contains properties such as database name, login, password etc. There is normally one TNozicleWebConnection per database. TNozicleDataSet components connect to the database via the TNozicleWebConnection component in the same way that Delphi TTable and TQuery connect to a database with the TDatabase componet.

Properties:

URL: **string** — The URL of the website where the PHP scripts are located, e.g. www.yourcompanyname.com.

UserLogin: **string** — The user's login name. Set the user table and fields in the script 'user_vars.inc'. The use of UserLogin and UserPassword is optional, but you will need to edit the PHP scripts if you decide not to use them.

UserPassword: **string** — The user's password. See UserLogin above. The UserPassword is not sent to the webserver with each request – it is used a a key to encrypt data.

SelectScriptName: **string** — Name and location of the of the PHP script for processing SQL 'Select' statements. Normally 'db_mysql.php'. E.g. If the scripts are located in a directory 'scripts' off the web server root directory this would take the format 'scripts/db_mysql.php'.

UpdateScriptName: **string** — Name and location of the PHP script for processing SQL 'Update' and 'Insert' statements. Normally 'db_mysql_update.php'. See SelectScriptName above.

Compress: **Boolean** — Compress the dataset returned to the client after executing an SQL 'Select' statement. Normally 'True'.

Key: **string** — Secret key used by the application to encrypt requests sent to the web server. You need to ensure that the key is the same in the file 'user_vars.inc' on the web server.

ProxyServer: **string** — URL of your proxy server (if using a proxy server)

ProxyPort: **integer** — Port of proxy server (e.g. 8080)

ProxyUserName: **string** — User login name for your proxy server

ProxyPassword: **string** — User password for your proxy server

TNozicleDataSet

TNozicleDataSet is a descendant of the Delphi TDataSet component, thus the usual methods of TDataSet can be used with TNozicleDataSet (e.g. FieldByName, RecordCount, First, Next, Last, Append, Edit, etc).

TNozicleDataSet has a property 'DoSQL'. If this is 'True' the SQL statement in the 'SQL' property will be executed on the server. If 'False' then an SQL statement is automatically constructed from the 'TableFields', 'TableName', 'RowOrder' and 'Filter' properties. The statement constructed is 'Select <TableFields> from <TableName> where <Filter> order by <RowOrder>'.

Properties:

SQL: **string** — A string containing the SQL to execute. The property 'DoSQL' must be 'True' for this to be used.

Filter: **string** — Filter used when 'DoSQL' is set to 'False'. This is the equivalent of the 'Where' clause in SQL.

TableName: **string** — Tablename when DoSQL is set to 'False'.

RowOrder: **string** — Order of the fields when DoSQL is set to 'False'.

TableFields: **string** — Fields to select when DoSQL is set to 'False'. Normally '*'.

DoSQL: **Boolean** — 'True' if an SQL statement is used in the 'SQL' property (similar to Delphi TQuery). 'False' if SQL is automatically constructed from 'TableFields', 'TableName', 'RowOrder' and 'Filter' (similar to Delphi TTable).

IndexField: **string** — Name of the field to use an primary index. This is a unique ID that is used for SQL 'update' and 'insert' statements generated by the component when Append or Edit is called. Use in conjunction with 'GUIDIndex'. If IndexField is not set then the database will not be able to be updated.

NozicleWebConnection: **TNozicleWebConnection** — Name of the TNozicleWebConnection component to use. Can be on another Form or DataModule (the name of the Form or DataModule unit must be in the 'uses' clause).

ErrorMessage: **string** — Error messages returned by the web server.

ErrorCode: **integer** — Error number of any error generated by the web server..

GUIDIndex: **Boolean** — 'True' if you want the system to automatically generate GUID's for use as Primary index ID's. If 'True' the property 'IndexField' must be set. When a row is appended (by calling Append) a GUID is automatically generated for the row.

DownloadTime: **integer** — Time in milliseconds that was taken between sending a request to the web server and receiving the full dataset.

TNozicleDataSet (cont)

DataSetSize: **integer** — Size in bytes of the raw data (i.e. compressed) received after completing a SQL 'Select' statement.

DeleteKeyWord: **string** — SQL keyword used in SQL statements. Eg if DeleteKeyWord is defined as **__del__** then the SQL statement to use in the Dataset is '__del__ from user where user_id = "1"' which is interpreted as 'delete from user where user_id = "1"'. The keyword is configured on the server in the user_vars.inc script. The DeleteKeyword property is part of the security (together with TNozicleWebConnection.Key and the username/password) as an outside user must know the keyword, the application key and a username to run delete SQLs.

Methods:

PostToWeb:

Commits the data to the database. If the 'Post' method is called the row being edited or appended is posted to memory. The changes are not saved to the database until the 'PostToWeb' method has been called. You can add or change as many rows in a dataset as desired before calling 'PostToWeb'. Thus a 'PostToWeb' call can update many rows in the database with a single call.

Because data is accessed via a web server, the connection is stateless meaning that the TNozicleDataSet is not permanently connected to the database. A new connection is made with each 'Select' (i.e set Active := True) and with each update/insert ('PostToWeb') in the same way that a web browser creates a new connection with the web server each time a page is requested.

TNozicleServerScript

This component allows you to send data to a script on your web server and receive the response. The script can be written in any language. The TNozicleServerScript sends parameters as a HTTP 'POST'. Examples of TNozicleServerScript in action are given in *Tutorial 3: TNozicleServerScript and email* and *Tutorial 4: TNozicleServerScript and SOAP (Web Services).* One of the uses of this component is to access Web Services (SOAP) without need for installing specialised client software, components or libraries on the end users machine.

Properties:

Params: **TStringList** — Contains the parameters to pass to your web server script. These are passed as a 'POST'. The format is <param name>=<param value>.

ReturnMessage: **string** — The text that is returned from the script after execution. May contain error messages if the script did not execute correctly.

ReturnInteger: **integer** — An optional numeric value that may be returned from your script.

SendAuthParams: **Boolean** — If TRUE then the user_name in the linked TNozicleWebConnection is sent to the script. This would be used to authenticate the user.

ScriptName: **string** — The name of the script on the server, including the directory if the script is located in a directory off the web server root directory. E.g. 'scripts/email.php' (note that there is no leading '/').

Methods:

Execute:

Executes the script. That is, send the POST to the script and receive the results.

Appendix: Installation of Apache Web Server with PHP and MySQL.

Nozicle does not offer official support for Apache, PHP and MySQL, however the following notes may be useful for installation in the Windows environment for development/testing purposes. If you received the Nozicle Development Environment on a CD, Apache, PHP and MySQL installation files will be on the CD.

Step 1: Download latest version of Apache from www.apache.org.
Step 2: Install Apache, following the installation Wizard.
Step 3: Test to see if the installation was successful by starting your browser and entering The URL 'http://localhost'.
Step 4: Download latest version of PHP from www.php.net.
Step 5: PHP is normally downloaded as a zip file. Extract the contents to **c:\php**.
Step 6: Copy php.ini to **c:\windows**.
Step 7: Edit c:\windows\php.ini: extension_dir = c:\php\extensions.
Step 8: Edit c:\windows\php.ini: uncomment 'extension=php_zlib.dll'.
Step 9: Copy contents of **c:\php\dlls** to **c:\windows\system32**.
Step 10: Copy **c:\php\php4ts.dll** to **c:\windows\system32**.
Step 11: Edit Apache configuration file **c:\program files\apache\cont\httpd.conf:**
Add line **LoadModule php4_module c:/php/sapi/php4apache.dll** at end of 'LoadModule' section.
Add line **AddModule mod_php4.c** at end of 'AddModule' section.
Add line **AddType application/x-httpd-php .php .phtml .php3** in 'AddType' module.
Note: There are no explicit 'LoadModule', 'AddModule' or 'AddType' sections in the configuration file, however most of these statements appear together in the same section of the file.
Step 12: Restart the Apache service. Select services from **Start → Settings → Control Panel → Administrative Tools**. Stop then Start (or restart) the service 'Apache'.
Step 13: Check the PHP installation by creating the following file:

```
<html><body>
<? phpinfo(); ?>
</body></html>
```

save as **test1.php** in **c:\program files\apache\htdocs**. Run using your web browser
with the URL **http://localhost/test1.php.** You should see detailed information about your system.
Step 14: Download latest version of MySQL from www.mysql.com. Install to directory **c:\mysql**.
Step 15: Install MySQL as an NT service: Open a command prompt window (**Start → Run,** enter **cmd** then click **Ok**).
Type **cd c:\mysql\bin**
Type **mysqld-nt --install**
Step 16: Optional: Change the MySQL root password:
Type **mysqladmin –u root password <new-password>**