Subject: Re: BRenBot Plugins
Posted by danpaul88 on Sat, 05 Nov 2011 17:35:29 GMT
View Forum Message <> Reply to Message

That's not unusual, its just because the version of the XML parser used in BRenBot uses a deprecated feature in Perl, nothing I can do about it until they release an updated version of the parser which doesn't use the deprecated feature. It shouldn't prevent anything from working.

As for changing the database... its probably just a case of swapping out

```
$dbh = DBI->connect("dbi:SQLite:dbname=brenbot.dat","","");
```

for

```
$dbh = DBI->connect("dbi:<db_of_choice>:dbname=<db_path>","<username>","<password >");
```

Documentation;
Quote:connect

```
  $dbh = DBI->connect($data_source, $username, $password)
            or die $DBI::errstr;
  $dbh = DBI->connect($data_source, $username, $password, \%attr)
            or die $DBI::errstr;
```

Establishes a database connection, or session, to the requested $data_source. Returns a database handle object if the connection succeeds. Use $dbh->disconnect to terminate the connection.

If the connect fails (see below), it returns undef and sets both $DBI::err and $DBI::errstr. (It does not explicitly set $!.) You should generally test the return status of connect and print $DBI::errstr if it has failed.

Multiple simultaneous connections to multiple databases through multiple drivers can be made via the DBI. Simply make one connect call for each database and keep a copy of each returned database handle.

The $data_source value must begin with "dbi:driver_name:". The driver_name specifies the driver that will be used to make the connection. (Letter case is significant.)

As a convenience, if the $data_source parameter is undefined or empty, the DBI will substitute the value of the environment variable DBI_DSN. If just the driver_name part is empty (i.e., the $data_source prefix is "dbi::"), the environment variable DBI_DRIVER is used. If neither variable is set, then connect dies.

Examples of $data_source values are:

```
  dbi:DriverName:database_name
  dbi:DriverName:database_name@hostname:port
  dbi:DriverName:database=database_name;host=hostname;port=port
```

There is no standard for the text following the driver name. Each driver is free to use whatever syntax it wants. The only requirement the DBI makes is that all the information is supplied in a single string. You must consult the documentation for the drivers you are using for a description of the syntax they require.

It is recommended that drivers support the ODBC style, shown in the last example above. It is also recommended that that they support the three common names 'host', 'port', and 'database' (plus 'db' as an alias for database). This simplifies automatic construction of basic DSNs: "dbi:$driver:database=$db;host=$host;port=$port". Drivers should aim to 'do something reasonable' when given a DSN in this form, but if any part is meaningless for that driver (such as 'port' for Informix) it should generate an error if that part is not empty.

If the environment variable DBI_AUTOPROXY is defined (and the driver in $data_source is not "Proxy") then the connect request will automatically be changed to:

```
  $ENV{DBI_AUTOPROXY};dsn=$data_source
```

DBI_AUTOPROXY is typically set as "dbi:Proxy:hostname=...;port=...". If $ENV{DBI_AUTOPROXY} doesn't begin with 'dbi:' then "dbi:Proxy:" will be prepended to it first. See the DBD::Proxy documentation for more details.

If $username or $password are undefined (rather than just empty), then the DBI will substitute the values of the DBI_USER and DBI_PASS environment variables, respectively. The DBI will warn if the environment variables are not defined. However, the everyday use of these environment variables is not recommended for security reasons. The mechanism is primarily intended to simplify testing. See below for alternative way to specify the username and password.

DBI->connect automatically installs the driver if it has not been installed yet. Driver installation either returns a valid driver handle, or it dies with an error message that includes the string "install_driver" and the underlying problem. So DBI->connect will die on a driver installation failure and will only return undef on a connect failure, in which case $DBI::errstr will hold the error message. Use eval { ... } if you need to catch the "install_driver" error.

The $data_source argument (with the "dbi:...:" prefix removed) and the $username and $password arguments are then passed to the driver for processing. The DBI does not define any interpretation for the contents of these fields. The driver is free to interpret the $data_source, $username, and $password fields in any way, and supply whatever defaults are appropriate for the engine being accessed. (Oracle, for example, uses the ORACLE_SID and TWO_TASK environment variables if no $data_source is specified.)

The AutoCommit and PrintError attributes for each connection default to "on". (See "AutoCommit" and "PrintError" for more information.) However, it is strongly recommended that you explicitly define AutoCommit rather than rely on the default. The PrintWarn attribute defaults to on if $^W is true, i.e., perl is running with warnings enabled.

The \%attr parameter can be used to alter the default settings of PrintError, RaiseError, AutoCommit, and other attributes. For example:

```
  $dbh = DBI->connect($data_source, $user, $pass, {
        PrintError => 0,
        AutoCommit => 0
  });
```

The username and password can also be specified using the attributes Username and Password, in which case they take precedence over the $username and $password parameters.

You can also define connection attribute values within the $data_source parameter. For example:

```
  dbi:DriverName(PrintWarn=>1,PrintError=>0,Taint=>1):...
```

Individual attributes values specified in this way take precedence over any conflicting values specified via the \%attr parameter to connect.

The dbi_connect_method attribute can be used to specify which driver method should be called to establish the connection. The only useful values are 'connect', 'connect_cached', or some specialized case like 'Apache::DBI::connect' (which is automatically the default when running within Apache).

Where possible, each session ($dbh) is independent from the transactions in other sessions. This is useful when you need to hold cursors open across transactions--for example, if you use one session for your long lifespan cursors (typically read-only) and another for your short update transactions.

For compatibility with old DBI scripts, the driver can be specified by passing its name as the fourth argument to connect (instead of \%attr):

```
  $dbh = DBI->connect($data_source, $user, $pass, $driver);
```

In this "old-style" form of connect, the $data_source should not start with "dbi:driver_name:". (If it does, the embedded driver_name will be ignored). Also note that in this older form of connect, the $dbh->{AutoCommit} attribute is undefined, the $dbh->{PrintError} attribute is off, and the old DBI_DBNAME environment variable is checked if DBI_DSN is not defined. Beware that this "old-style" connect will soon be withdrawn in a future version of DBI.

From http://search.cpan.org/~timb/DBI-1.616/DBI.pm#connect