Table of contents

Tutorial for Database interfacing

This is a tutorial for working examples for WikiPluginsDb. The code has been developed by YannickMajoros.

There are eight plugins that make up the functionality of the WikiPluginsDb, which can be Downloaded here. A basic knowledge of SQL is needed.

Things to note:

1. Wiki plugins are processed back-to-front by Tiki, meaning what is located at the bottom of the page is executed first. This means all of your database requests {REQUEST} should be at the bottom of the page.
2. Although plugins are **processed** or executed in reverse order by Tiki, they do **display** in the right order. Where you place the plugin objects on the page is where they appear
3. If you get a blank page, maybe you came across the Tiki adodb bug with dsn's. Check YannickMajoros post about it on the dev list for an explanation and a quick fix.

Working with these examples

1. To install the plugins to your tiki wiki site, uncompress the package to this directory lib/wiki-plugins/
2. In your MySQL database create a table called *Students* with the following fields:
    1. Id (autoincrement)
    2. First_Name
    3. Last_Name
3. Add some data into the Student so there is something to display
4. Create a dsn to tell the plugins how to connect to your database(s). The plugins can use multiple databases. If the name of your DSN is *default*, you won't have to specify it. To create a new DSN expand the 'Admin menu' and select 'DSN'. Create a DSN called "default???
5. When a DSN is created it produces a permission in the name of tiki_p_DNS_(what you named your DNS). Set the permissions in Admin menu->Group for the users that will have access to view the page.
6. In order for this plugin to work correctly you must disable the cache, goto Admin->Wiki->Cache wiki pages (global), set to 0.
7. Create a wiki page and paste the example and press save.

Examples

Display all data from a table:

```
{DATATABLE()}{DATATABLE}

{REQUEST()}
SELECT * FROM Students
{REQUEST}
```

Comments:

- Your SQL statements are executed using the REQUEST plugin, notice it is at the bottom of the page.
- In this example the REQUEST plugin has no db parameter, so it will take the default DSN (=dsn named *default*).
- The DATATABLE plugin generates a table from the SQL statement found in the REQUEST plugin
- When the DATATABLE is used without any parameters it will displays the data from the first 'default' request

---

Display data from a table with nested links

```
{DATATABLE(title=>1)}
__Id.__|__First name__|__Last name__|__''Links''__
*Id|*First_Name|*Last_Name|LINK:More info:StudentInfo:Id=*Id
{DATATABLE}

{REQUEST()}
SELECT Id, First_Name, Last_Name FROM Students
{REQUEST}
```

Comments:

- The *DATATABLE* plugin has the *title* parameter set, so the first row is the title row.
- Rows are separated by | , as are Tiki tables.
- The second line contains the template for the remaining rows. This relies on normal wiki formatting such as links. Database field names have a * before them (*First_Name)
- The SQL statement must specify each field name to be returned
- The link in the Links column is intended to provide more information on the student selected, see 'Using the navigator' example.

Adding new records

```
{DATATABLE(title=>1)}
__Id.__|__First name__|__Last name__|__''Links''__
*Id|*First_Name|*Last_Name|LINK:More info:StudentInfo:Id=*Id
{DATATABLE}

{FORM(update=>Create)}
__First name:__ {FIELD(name=>First_Name)}{FIELD}
__Last name:__ {FIELD(name=>Last_Name)}{FIELD}
{FORM}

{REQUEST(cursor=>1)}
SELECT Id, First_Name, Last_Name FROM Students
{REQUEST}

{REQUEST(id=>new_student,cond=>update)}
INSERT IGNORE INTO Students SET First_Name='?First_Name', Last_Name='?Last_Name'
{REQUEST}
```

Comments:

- A FIELD can be used to display or edit data. A FORM can create a submit button which will perform an action involving the encapsulated FIELD's.
- The *update* parameter of the *FORM* plugin is simply some text for the *Submit* button, which will display automatically unless you specify it.
- The *cond=>update* parameter on the *REQUEST* field tells that this REQUEST will only be executed if an *update* parameter has been given to the page, which is automatically the case when you submit a form using *{FORM}* (unless you specify otherwise with the *noupdate* parameter in *FORM*)
- The *id* parameter is the name of the request, which you could use in *FIELD* or *DATATABLE* with their *from* parameter. If you don't specify a name, unnamed requests will be numbered, beginning with 0 (which will be the default request).
- If you set the cursor=>1 on *REQUEST*, the data table will display navigation buttons for moving between result pages.

Counting records in a table

```
There are __{FIELD(edit=>0,from=>Row_Count, src=>Record_Count)}{FIELD}__ Rows in this
table

{REQUEST(id=>Row_Count)}
SELECT COUNT(*) Record_Count FROM Students
{REQUEST}
```

Comments:

- This will display the number of rows in the table
- Notice how the FIELD statement is mixed in with the wiki formatting

---

Using the Navigator

```
{FORM(id=>Form_Name, from=>Request_Name)}
__Text Box:__   {FIELD(name=>Field_Name1,from=>Request_Name,src=>First_Name)}{FIELD}
__Check Box:__{FIELD(type=>chk,src=>Last_Name,size=>150,from=> Last _Name)}{FIELD}
__Comment box: __  {FIELD(type=>memo, src=>Last_Name, from=>Request_Name)}{FIELD}
__Select: __ {FIELD(type=>select,src=>First_Name, from=> Request_Name,choices=>Student,
ch_field=>Student.First_Name, nopos=>1)}{FIELD}
{FORM}

{NAVIGATOR(from=>Request_Name, key=>Id, table=>Student)}{NAVIGATOR}

{REQUEST(id=Request_Name)}
SELECT Id, First_Name, Last_Name from Students where Id='?Id'
{REQUEST}
```

Comments:

- The NAVIGATOR will product the following error 'Navigator: position unknown..' at the end of the URL you will need to pass &Id=1 now the Navigator will work and the fields will populate.
- The check box will be ticked if there is data in the field, if the field is null then the check box will be empty
- nopos= if the table contains a column 'Pos' it can be used to order the options. Set this value to 1 if you do not have this column in the table.
- choices=DNS name/SQL table name. if useing default as the DNS use just the SQL table name.
- ch_field= field name in the sql statement to return the results from (Table_Name.Feild_Name)
- noadmin= removes the admin link on combobox's, default is active

---

More

There is much, much more to it. You could let the plugins make backups of your updates and add history into another table automatically, simply by setting flags. A lot of the features have not yet been documented.