Tutorial: How to update tables to follow XOOPS&apos; new naming scheme? - Tutorials

NEWS_PDF_AUTHOR: Mamba

NEWS_PDF_DATE: 2013/3/5 4:20:00

As you might already know, there is an effort to standardize our module development - from using the same module Admin GUI structure, to using the same icons across all modules, from using the same pagination structure for each table, to naming the tables and fields in a consistent way (see this thread). This tutorial will show you how to modify your module so it can rename the tables on the user site, when the user updates the module. This will follow the scheme suggested by alain01 The new table naming scheme is: ***mod_AAA_BBBB*** where AAA is the name of the module, and BBB is the name of the table. For example, when we have in the News module a table called "topics", in the new updated version of News, it will become: mod_news_topics Here are few steps to follow, as used recently in the Pedigree module called "animal": 1) The new version should have the tables defined properly in the SQL file, so new installation have the right tables installed right away 2) In the existing installation the users normally copy files over, and then run "update" in the Admin. Therefore we'll need to add a file with the updates. We'll call it "update_function.php" and will place it in /include folder 3) In order for XOOPS to call this file, we'll add in xoops_version.php file following:

```
 $modversion['onUpdate'] = 'include/update_function.php';
```

4) In that file, we start by adding a function to check if the table that we want to rename, does actually exist. This is done by using a function created by Hervet:

```
 function tableExists($tablename)
{
    global $xoopsDB;
    $result=$xoopsDB->queryF("SHOW TABLES LIKE '$tablename'");
    return($xoopsDB->getRowsNum($result) > 0);
}
```

5) then we add a following function that will be executed when we click on the Update button:

```
 function xoops_module_update_animal()
{
    global $xoopsDB;

    if (tableExists($xoopsDB->prefix('eigenaar'))) {
        $sql    = sprintf(
            'ALTER TABLE ' . $xoopsDB->prefix('eigenaar') . ' RENAME ' . $xoopsDB->prefix(
'mod_pedigree_owner')
        );
        $result = $xoopsDB->queryF($sql);
        if (!$result) {
            echo '' . _AM_PED_UPGRADEFAILED . ' ' . _AM_PED_UPGRADEFAILED2;
            $errors++;
        }
    }
    return TRUE;
}
```

In this code above, we are checking if the "eigenaar" does exist, and if it does, then we're renaming it to "mod_pedigree_owner"'. Of course, this is done for each table that we want to rename. 6) We also have to rename all occurrences of the tables in the code as well. a) as a first step, it's easy to just run search & replace using as part of the search the word "prefiix", so in our example, we'll replace: prefix("eigenaar") with: prefix("mod_pedigree_owner") This is for cases where we call the tables in a conventional way. b) But people are creative, and it might happen that they do it differently, so nothing will save us from testing, and eventually searching for the word "eigenaar" in all files, and then making a judgment call if it is meant as a table and therefore has to be renamed. The new naming scheme will make it easier two things: - to see in phpMyAdmin (or any other database tool) all the tables from a module grouped together. It will also distinguish them from the Core tables. - in the code it will also make it easy to find the tables just by searching "mod_MODULENAME" In the near future, we'll also consolidate names and characteristics of the typical fields in our modules, and provide them as guidelines. When you look at our modules, the same field could be named differently in each module. Let's take "Group ID" - it could be: gid, g_id, group_id, gr_id, etc. And if you are trying to maintain a module from somebody else, we are wasting too much time trying to figure out what a particular field is actually for. If you have improvements for this tutorial, please let us know. **And most importantly: - If you can help us to streamline and standardize module development, we would very much appreciate it. - If you like how a particular module does something and think that other modules should do the same, let us know. - If you see something cool being done by other Open Source Projects that XOOPS could benefit from, please let us know too. Please follow up in t**this thread

As you might already know, there is an effort to standardize our module development - from using the same module Admin GUI structure, to using the same icons across all modules, from using the same pagination structure for each table, to naming the tables and fields in a consistent way (see this thread). This tutorial will show you how to modify your module so it can rename the tables on the user site, when the user updates the module. This will follow the scheme suggested by alain01 The new table naming scheme is: ***mod_AAA_BBBB*** where AAA is the name of the module, and BBB is the name of the table. For example, when we have in the News module a table called "topics", in the new updated version of News, it will become: mod_news_topics Here are few steps to follow, as used recently in the Pedigree module called "animal": 1) The new version should have the tables defined properly in the SQL file, so new installation have the right tables installed right away 2) In the existing installation the users normally copy files over, and then run "update" in the Admin. Therefore we'll need to add a file with the updates. We'll call it "update_function.php" and will place it in /include folder 3) In order for XOOPS to call this file, we'll add in xoops_version.php file following:

```
 $modversion['onUpdate'] = 'include/update_function.php';
```

4) In that file, we start by adding a function to check if the table that we want to rename, does actually exist. This is done by using a function created by Hervet:

```
 function tableExists($tablename)
{
    global $xoopsDB;
    $result=$xoopsDB->queryF("SHOW TABLES LIKE '$tablename'");
    return($xoopsDB->getRowsNum($result) > 0);
}
```

5) then we add a following function that will be executed when we click on the Update button:

```
 function xoops_module_update_animal()
{
    global $xoopsDB;

    if (tableExists($xoopsDB->prefix('eigenaar'))) {
        $sql    = sprintf(
            'ALTER TABLE ' . $xoopsDB->prefix('eigenaar') . ' RENAME ' . $xoopsDB->prefix(
'mod_pedigree_owner')
        );
        $result = $xoopsDB->queryF($sql);
        if (!$result) {
            echo '' . _AM_PED_UPGRADEFAILED . ' ' . _AM_PED_UPGRADEFAILED2;
            $errors++;
        }
    }
    return TRUE;
}
```

In this code above, we are checking if the "eigenaar" does exist, and if it does, then we're renaming it to "mod_pedigree_owner"'. Of course, this is done for each table that we want to rename. 6) We also have to rename all occurrences of the tables in the code as well. a) as a first step, it's easy to just run search & replace using as part of the search the word "prefiix", so in our example, we'll replace: prefix("eigenaar") with: prefix("mod_pedigree_owner") This is for cases where we call the tables in a conventional way. b) But people are creative, and it might happen that they do it differently, so nothing will save us from testing, and eventually searching for the word "eigenaar" in all files, and then making a judgment call if it is meant as a table and therefore has to be renamed. The new naming scheme will make it easier two things: - to see in phpMyAdmin (or any other database tool) all the tables from a module grouped together. It will also distinguish them from the Core tables. - in the code it will also make it easy to find the tables just by searching "mod_MODULENAME" In the near future, we'll also consolidate names and characteristics of the typical fields in our modules, and provide them as guidelines. When you look at our modules, the same field could be named differently in each module. Let's take "Group ID" - it could be: gid, g_id, group_id, gr_id, etc. And if you are trying to maintain a module from somebody else, we are wasting too much time trying to figure out what a particular field is actually for. If you have improvements for this tutorial, please let us know. **And most importantly: - If you can help us to streamline and standardize module development, we would very much appreciate it. - If you like how a particular module does something and think that other modules should do the same, let us know. - If you see something cool being done by other Open Source Projects that XOOPS could benefit from, please let us know too. Please follow up in t**his thread