Accessing the Personalized Schema

for User Tables and Client Programs

A ***Delightful Labor*** White Paper
January 30th , 2015

Table of Contents

Overview

This white paper describes the concepts behind the personalized tables used in ***Delightful Labor*** and tools that can be used to access the schema elements associated with these tables.

This paper is intended for developers who wish to extend the function of ***Delightful Labor*** by creating custom reports or features based on personalized table content, or those who wish to have an understanding of the mechanics behind the personalized table implementation of ***Delightful Labor***.

This paper refers to versions of ***Delightful Labor*** released on or after January 30th, 2015.

It is assumed that the reader has familiarity with php, mySQL, object-oriented programming, and the codeIgniter framework. That being said, Be Not Afraid!

Personalized Table Overview

Personalized tables greatly extend the functionality of ***Delightful Labor.*** Administrators can create and personalize tables (with a variety of field types) and attach these tables to parent tables. Parent tables may be Clients, Client Locations, People, Businesses, Donations, Sponsorships, Volunteers, and Users.

Tables may be defined as single entry or multi-entry. If single entry, the table allows only one entry per parent table record. This essentially extends the fields of the parent record. A multi-entry table allows 0, 1, or more entries per parent record. For example, if you wished to add demographics information to the client table, you would define the table as single entry (since the information extends the base client record), but if you wished to add a table for staff member training, you would use a multi-table (since a staff member may have multiple training sessions).

The personalized tables are managed by the following mySQL database tables:

- **`uf_tables`** – the master table for managing personalized tables, this table contains:
  - the name of the personalized table (both the external and internal name)
  - the attachment type (for example, Client, Sponsorship, etc)
  - flag to indicate single entry or multi-entry table type
  - information linking this table to custom validation code
  - information for generating alerts if the personalized table entry has not been written
  - additional fields to manage personalized tables
- **`uf_fields`** – information about the fields contained in the personalized tables. This table contains:
  - foreign key to the parent table **`uf_tables`**
  - internal and external field name
  - field type (checkbox, date, drop-down list, etc)
  - sort index
  - default values
  - pre-fill flag (fields can optionally be pre-filled from the most recent entry for a given parent record)
  - flag to indicate if the field is required
  - additional fields to administer this table
- **`uf_ddl`** – drop-down list values. This table contains:
  - foreign key to the parent field (table **`uf_fields`**)

    - list entry name (defined by the user)
    - sort index
    - "retired" flag; when list entries are "removed" by the user, the system does not physically remove the entry (to prevent orphaned references), but simply flags it as "retired"
- **`uf_ddl_multi`** – this table links the user's selections from a multi-select drop-down list. The table contains:
    - the foreign key to the field (table **`uf_fields`**)
    - the foreign key to the personalized table (table **`uf_tables`**)
    - the foreign key to the data table (table **`uf_xxxxxx`**, see below)
    - the foreign key to the drop-down list (table **`uf_ddl`**)
- **`uf_logs`** – log entries for the field type "log". Note that this field type may be deprecated in the future, since the introduction of the multi-record personalized table provides the same functionality).

When a new personalized table is created, a new database table is also created, with a name in the form of
`uf_xxxxxx`
where `xxxxxx` is a six-digit number corresponding to the keyID of the personalized table's entry in the **`uf_tables`** table.

Client Program Overview

Client programs are managed as a special case implementation of personalized tables. When a user creates a client program, the ***Delightful Labor*** system makes an entry in the table **`cprograms`** and creates two multi-record personalized tables: one for enrollment, and one for attendance/contact.

The personalized tables contain additional fields to support the client program. The enrollment table contains:

- enrollment start and stop dates
- a flag to indicate if the client is currently enrolled

The attendance table contains:

- a foreign key to the associated enrollment record
- the attendance date (a standard field for all attendance records)
- a duration field (a standard field for all attendance records)
- a case notes field (a standard field for all attendance records)

The top-level client program table is **`cprograms`**. It contains the following fields:

- the user's program name and description
- the program start and end dates
- the vocabulary used for enrollment and attendance (for example, some programs may record "client contact" rather than "attendance"
- enrollment and attendance table IDs (these are foreign key to the table **`uf_tables`**)
- custom validation information (***Delightful Labor*** users may provide custom php code to validate data entry for the enrollment and attendance forms)
- an activity list ID (foreign key to the table **`uf_ddl`** for managing drop-down list activity entries in the attendance records).

The mschema Model

Overview

Kurt Vonnegut once famously said, “Ilium, New York, is divided into three parts,” and the same can be said for codeIgniter, the framework used by ***Delightful Labor***. Most of ***Delightful Labor***'s code can be grouped into one of three categories:

- **controllers** – code that manages the events associated with processing a web page
- **models** – class modules that generally provide an interface to the database and manage data to and from other parts of the program
- **views** – the code that generates the HTML code that is displayed as a web page

Delightful Labor provides a model, called **`mschema`**, that provides information about the personalized tables to the software developer.

The associated file is located at

```
application/models/personalization/muser_schema.php
```

The model can be added to a controller via the statement

```
$this->load->model('personalization/muser_schema', 'cUFSchema');
```

(note: the programmer can choose any variable name for the model; for convenience the variable `cUFSchema` is used here.)

When loading personalized table schema information, the model **`mschema`** loads an array called

```
$this->cUFSchema->schema
```

The array is indexed by `tableID`; for example, `$this->cUFSchema->schema[87]` corresponds to the schema for personalized table `uf_000087`.

The variable `$this->cUFSchema->lNumTables` contains the number of tables loaded.

By default, the fields associated with a personalized table are included in `$this->cUFSchema->schema`. This can be disabled by setting

```
$this->cUFSchema->bLoadFields = false;
```

Methods

Table Utilities

Load a Single Table Via Table ID

`loadUFSchemaSingleTable($lTableID)`

This method loads the schema for a specified table ID.

Parameters

- `$lTableID` (input / integer) - the key ID of the table to be loaded

Sample calling sequence:

```
function example01(){
//------------------------------------------------------------------------
// load a single personalized table via tableID
//------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaSingleTable(83);
}
```

Sample output:

This example uses the personalized table "Staff Training" described below.

```
$this->cUFSchema->schema
Array
(
    [83] => stdClass Object

(
    [lTableID] => 83
    [strUserTableName] => Staff Training
    [strDataTableName] => uf_000083
    [enumAttachType] => user
    [strDescription] =>
    [bHidden] =>
    [bMultiEntry] => 1
    [bReadOnly] =>
    [bCollapsibleHeadings] => 1
    [bCollapseDefaultHide] => 1
    [strVerificationModule] =>
    [strVModEntryPoint] =>
    [bAlertIfNoEntry] =>
    [strAlertMsg] =>
    [strFieldPrefix] => uf000083
    [strDataTableKeyID] => uf000083_lKeyID
    [strDataTableFID] => uf000083_lForeignKey
    [lNumFields] => 4
    [fields] => Array
        (
            [0] => stdClass Object
                (
                    [lFieldID] => 1841
                    [lSortIDX] => 1
                    [strFieldNameInternal] => uf000083_001841
                    [strFieldNameUser] => Date of Training
                    [strFieldNotes] =>
                    [enumFieldType] => Date
                    [bRequired] => 1
                    [bConfigured] => 0
```

```
                [bCheckDef] => 0
                [curDef] => 0.00
                [strTxtDef] =>
                [lDef] => -1
                [lDDLDefault] =>
                [lCurrencyACO] => 1
                [bHidden] => 0
            )

        [1] => stdClass Object
            (
                [lFieldID] => 1842
                [lSortIDX] => 2
                [strFieldNameInternal] => uf000083_001842
                [strFieldNameUser] => Subject
                [strFieldNotes] =>
                [enumFieldType] => DDL
                [bRequired] => 1
                [bConfigured] => 1
                [bCheckDef] => 0
                [curDef] => 0.00
                [strTxtDef] =>
                [lDef] => -1
                [lDDLDefault] =>
                [lCurrencyACO] => 1
                [bHidden] => 0
            )

        [2] => stdClass Object
            (
                [lFieldID] => 1843
                [lSortIDX] => 3
                [strFieldNameInternal] => uf000083_001843
                [strFieldNameUser] => Notes
                [strFieldNotes] =>
                [enumFieldType] => TextLong
                [bRequired] => 0
                [bConfigured] => 0
                [bCheckDef] => 0
                [curDef] => 0.00
                [strTxtDef] =>
                [lDef] => -1
                [lDDLDefault] =>
                [lCurrencyACO] => 1
                [bHidden] => 0
            )

        [3] => stdClass Object
            (
                [lFieldID] => 1844
                [lSortIDX] => 4
                [strFieldNameInternal] => uf000083_001844
                [strFieldNameUser] => Duration
                [strFieldNotes] =>
                [enumFieldType] => DDL
                [bRequired] => 1
                [bConfigured] => 1
                [bCheckDef] => 0
                [curDef] => 0.00
                [strTxtDef] =>
                [lDef] => -1
                [lDDLDefault] =>
                [lCurrencyACO] => 1
                [bHidden] => 0
            )
    )
)
```

Field Notes

Most fields are self-explanatory, but here are some additional details:

- `lTableID` - foreign key to the table `uf_tables`
- `enumAttachType` - indicates the parent table type. Context types are defined in file `application/helpers/dl_config_helper.php`
- `bMultiEntry` - if true (i.e. 1), this table allows multiple entries per parent table record; otherwise a single record is allowed.
- `strVerificationModule` - if custom verification has been created for this personalized table, this is the file name
- `strVModEntryPoint` - if custom verification has been created for this personalized table, this is the entry point name
- `lFieldID` - foreign key to the table `uf_fields`
- `enumFieldType` - indicates the field type. Field types are defined in file `application/helpers/dl_config_helper.php`

Loading All Tables for a Given Parent Table Type

```
loadUFSchemaViaAttachType($enumAttachType, $bShowHidden=true,
                 $bIncludePerms=false)
```

This routine loads all the tables associated with a given parent table type. For example, you could load all personalized Client tables with this call.

Parameters

- `$enumAttachType` (input / enumerated) - the parent table attachment type. The attachment context constants are defined in `application/helpers/dl_config_helper.php`
- `$bShowHidden` (input / boolean) - if true, include tables that have been "hidden" by the user
- `$bIncludePerms` (input / boolean) - if true, include table permission information. The information is the permission groups that have been assigned to the table. ***Delightful Labor*** requires users either be administrators or members of all the assigned permission groups to grant access to a given personalized table.

Sample calling sequence:

```
function example02(){
//---------------------------------------------------------------------
// load all client personalized tables; include hidden tables and
// permission information
//---------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachType(CENUM_CONTEXT_CLIENT, true, true);
}
```

Sample output:

The output is similar to that described above, with the addition of table permission information for each table:

```
[lNumPerms] => 2
[perms] => Array
    (
        [32] => stdClass Object
            (
                [strGroupName] => Client Intake (general)
                [strSafeGroupName] => Client Intake (general)
```

```
                    [lGroupChildID] => 32
                    [lGroupID] => 9
                    [lForeignID] => 1
                    [enumSubGroup] => personalizedTable
                )

            [5464] => stdClass Object
                (
                    [strGroupName] => Health-Related Programs
                    [strSafeGroupName] => Health-Related Programs
                    [lGroupChildID] => 5464
                    [lGroupID] => 8
                    [lForeignID] => 1
                    [enumSubGroup] => personalizedTable
                )
        )

    [lNumConsolidated] => 2
    [cperms] => Array
        (
            [9] => stdClass Object
                (
                    [strGroupName] => Client Intake (general)
                    [strSafeGroupName] => Client Intake (general)
                    [enumSubGroup] => personalizedTable
                )

            [8] => stdClass Object
                (
                    [strGroupName] => Health-Related Programs
                    [strSafeGroupName] => Health-Related Programs
                    [enumSubGroup] => personalizedTable
                )
        )

    [bAllowAccess] => 1
)
```

- `strSafeGroupName` - the group name, pre-filtered for HTML display
- `bAllowAccess` - Boolean value that indicates if the current user has access to this table

Loading a Table Based on Name/Parent Table Type

```
loadUFSchemaViaAttachTypeUserTabName($enumAttachType, $strUserTabName,
                 &$lTableID, $bFreakIfNotFound = false)
```

Loads a personalized table based on the attachment type and the user's table name. User-defined personalized table names must be unique within an attachment type, but the same table name may be used for different attachment types. For example, there can not be two client tables named "Demographics", but it is permissible to have a client table and a people table named "Demographics".

Parameters

- `$enumAttachType` (input / enumerated) - the parent table attachment type. The attachment context constants are defined in file `application/helpers/dl_config_helper.php`
- `$strUserTabName` (input / string) - table name assigned by the user
- `$lTableID` (output / integer) - the table ID of the loaded table
- `$bFreakIfNotFound` (input / boolean) - if true and the requested table is not found, stop execution of the program. If the caller has developer permissions, extensive debug information is displayed.

Sample calling sequence:

```
function example03(){
//---------------------------------------------------------------------
// load all client personalized tables; include hidden tables and
// permission information
//---------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(CENUM_CONTEXT_CLIENT,
          'Demographics', $lTableID, true);
}
```

Generic Table Load

`loadUFSchema($bIncludePerms=false)`

This generic table schema loaded allows you to provide your own SQL where clause.

Parameters

- `$bIncludePerms` (input / boolean) - if true, include table permission information. The information is the permission groups that have been assigned to the table. ***Delightful Labor*** requires users either be administrators or members of all the assigned permission groups to grant access to a given personalized table.

Sample calling sequence:

To set your own SQL where clause, set the variable `$this->(class name)->sqlWhereExtra`.

```
function example04(){
//---------------------------------------------------------------------
// load all hidden personalized tables
//---------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->sqlWhereExtra = ' AND pft_bHidden ';    // load all hidden tables
   $this->cUFSchema->loadUFSchema();
}
```

Table Information as Scalars

```
tableEssentialsViaTableID($lTableID,
              &$strUserTableName, &$strDataTableName,  &$enumAttachType,
              &$strFieldPrefix,   &$strDataTableKeyID, &$strDataTableFID)
```

This utility routine returns key table information as scalars. The table must be previously loaded by the routines described above.

Parameters

- `$lTableID` (input / integer) - table ID of the table of interest
- `$strUserTableName` (output / string) - the user-specified table name
- `$strDataTableName` (output / string) - the internal table name
- `$enumAttachType` (output / enumerated) - the parent table context type
- `$strFieldPrefix` (output / string) - the string used as a prefix for fields in the table

- `$strDataTableKeyID` (output) - the field name of the keyID of the table
- `$strDataTableFID` (output) - the field name of the foreign ID (ID of parent table)

Sample calling sequence:

```
function test05(){
//-------------------------------------------------------------------------------
// retrieve scalar info for a personalized table
//-------------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);
   $this->cUFSchema->tableEssentialsViaTableID($lTableID,
               $strUserTableName, $strDataTableName,  $enumAttachType,
               $strFieldPrefix,   $strDataTableKeyID, $strDataTableFID);
}
```

Sample output:

In this example,

`$strUserTableName` == "Staff Training"
`$strDataTableName` == "uf_000083"
`$enumAttachType` == "user" (corresponding to constant `CENUM_CONTEXT_USER`)
`$strFieldPrefix` == "uf000083"
`$strDataTableKeyID` == "uf000083_lKeyID"
`$strDataTableFID` == "uf000083_lForeignKey"

Field Utilities

The following utilities can help the developer work with specific fields within a table. These routines require that the referenced tables be previously loaded (by the routines described above).

Locating a Field Within a Given Table by User Name

`lFieldIdxViaUserFieldName($lTableID, $strUFieldName, $bFreakOutIfNotFound = false)`

This routine provides the index for a specified field with the given table.

Parameters

- `$lTableID` (input / integer) - table ID of the table of interest
- `$strUFieldName` (input / string) - the field name defined by the user
- `$bFreakIfNotFound` (input / boolean)- if true and the requested field is not found, stop execution of the program. If the caller has developer permissions, extensive debug information is displayed.

Sample calling sequence:

```
function test04(){
//-----------------------------------------------------------------------------
// find the array index for field "Age Range" in client table "Demographics"
//-----------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(CENUM_CONTEXT_CLIENT,
                              'Demographics', $lTableID, true);
   $lIDX_AgeRange =
          $this->cUFSchema->lFieldIdxViaUserFieldName($lTableID,
                        'Age Range', true);
}
```

Sample output:

In this example,

```
$lIDX_AgeRange == 2
```

Which corresponds with `$this->cUFSchema->schema[1]->fields[2]:`

```
[2] => stdClass Object
    (
        [lFieldID] => 2
        [lSortIDX] => 2
        [strFieldNameInternal] => uf000001_000002
        [strFieldNameUser] => Age Range
        [strFieldNotes] =>
        [enumFieldType] => DDL
        [bRequired] => 0
        [bConfigured] => 1
        [bCheckDef] => 0
        [curDef] => 0.00
        [strTxtDef] =>
        [lDef] => -1
        [lDDLDefault] =>
        [lCurrencyACO] => 1
        [bHidden] => 0
    )
```

Locating a Field Via Field ID

`lFieldIdxViaFieldID($lTableID, $lFieldID, $bFreakOutIfNotFound = false)`

This routine is used to load the field index (the index within
`$this->cUFSchema->schema[x]->fields`
based on the field ID (the field's key ID from table `uf_fields`).

Parameters

- `$lTableID` (input / integer) - table ID of the table of interest
- `$lFieldID` (input / integer) - field ID (foreign Key to table **`uf_fields`**) of the field of interest
- `$bFreakIfNotFound` (input / boolean)- if true and the requested field is not found, stop execution of the program. If the caller has developer permissions, extensive debug information is displayed.

Sample calling sequence:

```
function test06(){
//-------------------------------------------------------------------------
// find the array index for field ID 1844 ("Duration") in client table "Staff Training"
//-------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                           CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);
   $lIDX_Duration = $this->cUFSchema->lFieldIdxViaFieldID($lTableID, 1844, true);
}
```

Sample output:

In this example,
`$lIDX_Duration == 3`
Corresponding to:

```
$this->cUFSchema->schema[83]->[fields] => Array
    (
        [3] => stdClass Object
            (
                [lFieldID] => 1844
                [lSortIDX] => 4
                [strFieldNameInternal] => uf000083_001844
                [strFieldNameUser] => Duration
                [strFieldNotes] =>
                [enumFieldType] => DDL
                [bRequired] => 1
                [bConfigured] => 1
                [bCheckDef] => 0
                [curDef] => 0.00
                [strTxtDef] =>
                [lDef] => -1
                [lDDLDefault] =>
                [lCurrencyACO] => 1
                [bHidden] => 0
            )
```

Field Information as Scalars

```
fieldsEssentialsViaFieldIDX($lTableID, $lFieldIDX,
                  &$lFieldID, &$lSortIDX, &$strFNInternal, &$strFNUser,
                  &$enumFType, &$strFieldNotes)
```

This routine provides scalar information about a field within a given pre-loaded personalized table. The caller specifies the field by the array index within the table schema's field section.

Parameters

- `$lTableID` (input / integer) - table ID of the table of interest
- `$lFieldIDX` (input / string) - the index within the field array of the specified table
- `$lFieldID` (output / integer) - the field ID (foreign key to table **`uf_fields`**)
- `$lSortIDX` (output / integer) - the user-specified sort index (order in which fields are displayed)
- `$strFNInternal` (output / string) - the internal field name
- `$strFNUser` (output / string) - the user-specified field name
- `$enumFType` (output / enumerated) - the field type. The field type constants are defined in file `application/helpers/dl_config_helper.php`
- `$strFieldNotes` (output / string)

Sample calling sequence:

```
function test07(){
//---------------------------------------------------------------------------
// field info for field ID 1844 ("Duration") in client table "Staff Training"
//---------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);
   $lIDX_Duration = $this->cUFSchema->lFieldIdxViaFieldID($lTableID, 1844, true);
   $this->cUFSchema->fieldsEssentialsViaFieldIDX($lTableID, $lIDX_Duration,
               $lFieldID, $lSortIDX, $strFNInternal, $strFNUser,
               $enumFType, $strFieldNotes);
}
```

Sample output:

In this example,

```
$lFieldID      == 1844
$lSortIDX      == 4
$strFNInternal == "uf000083_001844"
$strFNUser     == "Duration"
$enumFType     == DDL
$strFieldNotes == ""
```

Loading the Table Schema with Drop-Down List Values

`loadDDLValues($lTableID)`

This routine loads the drop-down information for all drop-down and multi-select drop-down fields for the specified table. If the tableID is null, drop-down information is loaded for all tables. The table schema information must be pre-loaded before calling this routine.

Parameters

- `$lTableID` (input / integer) - table ID of the table of interest. If null, drop-down list information is provided for all loaded tables

Sample calling sequence:

```
function test08(){
//---------------------------------------------------------------------------
// drop-down info for client table "Staff Training"
//---------------------------------------------------------------------------
   $this->load->model('personalization/muser_schema', 'cUFSchema');
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                       CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);
   $this->cUFSchema->loadDDLValues($lTableID);
}
```

Sample output:

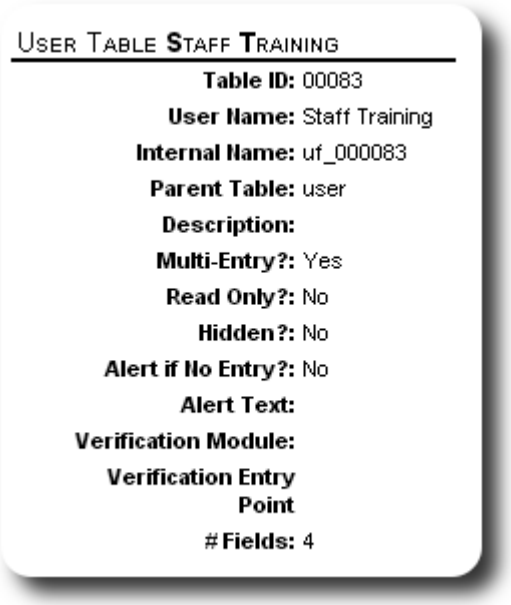
In this example, the array `$this->cUFSchema->schema[83]->fields was updated` with:

```
    [1] => stdClass Object
        (
            [lFieldID] => 1842
            [lSortIDX] => 2
         ...
            [lNumDDL] => 8
            [ddlInfo] => Array
                (
                    [0] => stdClass Object
                        (
                            [lKeyID] => 2656
                            [lSortIDX] => 1
                            [strDDLEntry] => Cleaning the Microwave for Beginners
                        )
                    [1] => stdClass Object
                        (
                            [lKeyID] => 2655
                            [lSortIDX] => 2
                            [strDDLEntry] => CPR and First Aid Training
                        )
                    [2] => stdClass Object
                        (
                            [lKeyID] => 2515
                            [lSortIDX] => 3
                            [strDDLEntry] => Delightful Labor - general
                        )
                 ...
                    [7] => stdClass Object
                        (
                            [lKeyID] => 2658
                            [lSortIDX] => 8
                            [strDDLEntry] => Go Freedonia's Phone System - Part 3
                        )
                )
        )
```

Examples

An Example: Accessing the Fields in a Personalized Table

For the examples, lets say we have a personalized table to track staff member training. The parent table is Users, and the table is configured as:



User Table Staff Training

Table ID: 00083
User Name: Staff Training
Internal Name: uf_000083
Parent Table: user
Description:
Multi-Entry?: Yes
Read Only?: No
Hidden?: No
Alert if No Entry?: No
Alert Text:
Verification Module:
Verification Entry Point
 # Fields: 4

For the purpose of these examples, the key, foreign ID, and internal names are not important. As will be shown, the `mschema` model resolves all these field values and names.

The fields are defined as:

Fields in *"Staff Training"* (Users / Staff Members table)

	Field ID		Field Name	Type	Internal Name	Required?	Prefill?	Default
1	01841	✕	**Date of Training**	Date	`uf000083_001841`	Yes	No	n/a
2	01842	✕	**Subject**	Drop-down list 8 items	`uf000083_001842`	Yes	No	n/a
3	01843	✕	**Notes**	Text (long)	`uf000083_001843`	No	No	n/a
4	01844	✕	**Duration**	Drop-down list 21 items	`uf000083_001844`	Yes	No	n/a

The drop-down lists have the following entries:

Entries in Drop-down List:
Staff Training: Subject

Entry ID	Entry
02656	Cleaning the Microwave for Beginners
02655	CPR and First Aid Training
02515	Delightful Labor - general
02652	Freedonia's Non-Profit Regulations
02653	Go Freedonia New Hire Orientation
02654	Go Freedonia's Phone System - Part 1
02657	Go Freedonia's Phone System - Part 2
02658	Go Freedonia's Phone System - Part 3

Entries in Drop-down List:
Staff Training: Duration

Entry ID	Entry
02521	- none -
02522	15 minutes
02523	30 minutes
02524	45 minutes
02525	1 hour
02526	1 hour 15 minutes
02527	1 hour 30 minutes
02528	1 hour 45 minutes
02529	2 hours
02530	2 hours 15 minutes
02531	2 hours 30 minutes
02532	2 hours 45 minutes
02533	3 hours
02534	3 hours 15 minutes
02535	3 hours 30 minutes
02536	3 hours 45 minutes
02537	4 hours
02538	5 hours
02539	6 hours
02540	7 hours
02541	8 hours

An Example: Creating a Simple Report

The following example creates a simple report of the users who have received training for a given year. It is assumed that the personalized user table "Staff Training" has been configured as shown above. Note that the key IDs and foreign keys will be resolved automatically by they report.

Files

Controller: `application/controllers/admin/dev/training_report.php`
Model: `application/models/admin/dev/mtraining.php`
View: `application/views/admin/dev/training_by_year_view.php`
Sample URL: `delightful/index.php/admin/dev/training_report/reportByYear/2013`

The Controller File

```
<?php
/*-------------------------------------------------------------------------
// Delightful Labor              copyright (c) 2015 Database Austin
// This software is provided under the GPL.
// Please see http://www.gnu.org/copyleft/gpl.html for details.
-------------------------------------------------------------------------*/
class training_report extends CI_Controller {

   function __construct(){
      parent::__construct();
      session_start();
      setGlobals($this);
   }

   function reportByYear($lYear){
   //-------------------------------------------------------------------------
   //
   //-------------------------------------------------------------------------
      $displayData = array();
      $displayData['js'] = '';

      $lYear = $displayData['lYear'] = (int)$lYear;

         // load models
      $this->load->model('personalization/muser_schema', 'cUFSchema');
      $this->load->model('admin/dev/mtraining',          'cTraining');

         // load training summary for the specified year
      $this->cTraining->loadUserTrainingByYear(
                           $lYear, $displayData['lNumUsers'], $displayData['training']);

         // breadcrumbs
      $displayData['pageTitle']      = anchor('main/menu/admin', 'Admin', 'class="breadcrumb"')
                              .' | Staff Training By Year';

      $displayData['title']          = CS_PROGNAME.' | Reports';
      $displayData['nav']            = $this->mnav_brain_jar->navData();

         // load the view for this report
      $displayData['mainTemplate']   = 'admin/dev/training_by_year_view';

         // make variables available to the view
      $this->load->vars($displayData);

         // load the template, which will then load this report's view
      $this->load->view('template');
   }
}
```

The Model File

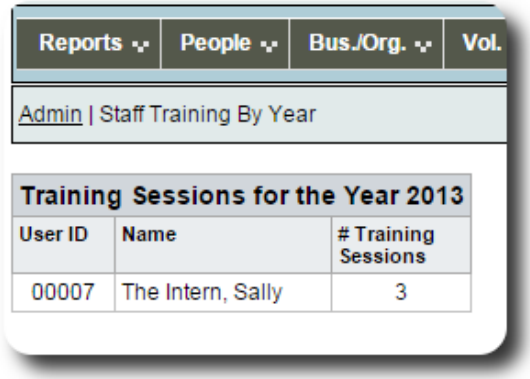
```
<?php
/*--------------------------------------------------------------------
// Delightful Labor               copyright (c) 2015 Database Austin
// This software is provided under the GPL.
// Please see http://www.gnu.org/copyleft/gpl.html for details.
--------------------------------------------------------------------*/

class mtraining extends CI_Model{

   function __construct() {
   //-------------------------------------------------------------------------
   // constructor
   //-------------------------------------------------------------------------
      parent::__construct();
   }

   function loadUserTrainingByYear($lYear, &$lNumUsers, &$training){
   //-------------------------------------------------------------------------
   // return info about users who received training during the
   // specified year
   //-------------------------------------------------------------------------
      $training = array();

         // load the personalized user table "Staff Training"
      $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                     CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);

         // short-cut to our table of interest
      $staff = &$this->cUFSchema->schema[$lTableID];

         // field info for date of training
      $lIDX_DateOfTraining = $this->cUFSchema->lFieldIdxViaUserFieldName(
                           $lTableID, 'Date of Training', true);
      $strFN_DateOfTraining = $staff->fields[$lIDX_DateOfTraining]->strFieldNameInternal;

         // field name for "retired", or deleted record flag
      $strFN_Retired = $staff->strFieldPrefix.'_bRetired';

         /* this sequel statement resolves to
               SELECT COUNT(*) AS lNumRecs, us_lKeyID, us_strFirstName, us_strLastName
               FROM uf_000083 INNER JOIN admin_users ON uf000083_lForeignKey=us_lKeyID
               WHERE YEAR(uf000083_001841)=2013 AND NOT uf000083_bRetired
               ORDER BY us_strLastName, us_strFirstName;
         */
      $sqlStr =
        "SELECT COUNT(*) AS lNumTraining, us_lKeyID, us_strFirstName, us_strLastName
         FROM $staff->strDataTableName
            INNER JOIN admin_users ON $staff->strDataTableFID=us_lKeyID
         WHERE YEAR($strFN_DateOfTraining)=$lYear
            AND NOT $strFN_Retired
         ORDER BY us_strLastName, us_strFirstName;";

      $query = $this->db->query($sqlStr);
      $lNumUsers = $query->num_rows();
      if ($lNumUsers > 0) {
         $idx = 0;
         foreach ($query->result() as $row){
            $training[$idx] = new stdClass;
            $training[$idx]->userID       = (int)$row->us_lKeyID;
            $training[$idx]->lNumTraining = (int)$row->lNumTraining;
            $training[$idx]->strFName     = $row->us_strFirstName;
            $training[$idx]->strLName     = $row->us_strLastName;
            ++$idx;
         }
      }
   }
}
```

The View File

```
<?php
/*-------------------------------------------------------------------------
// Delightful Labor                copyright (c) 2015 Database Austin
// This software is provided under the GPL.
// Please see http://www.gnu.org/copyleft/gpl.html for details.
-------------------------------------------------------------------------*/

   if ($lNumUsers == 0){
      echoT('There were no training sessions for year '.$lYear.'<br><br>');
      return;
   }

      // write the table headings
   echoT(
      '<br>
         <table class="enpRpt">
         <tr>
            <td colspan="3" class="enpRptTitle">
               Training Sessions for the Year '.$lYear.'
            </td>
         </tr>');

   echoT('
       <tr>
          <td class="enpRptLabel">
             User ID
          </td>
          <td class="enpRptLabel">
             Name
          </td>
          <td class="enpRptLabel">
             # Training<br>Sessions
          </td>
       </tr>');

   foreach ($training as $tr){
      echoT('
       <tr>
          <td class="enpRpt" style="text-align: center;">'
             .str_pad($tr->userID, 5, '0', STR_PAD_LEFT).'
          </td>
          <td class="enpRpt">'
             .htmlspecialchars($tr->strLName.', '.$tr->strFName).'
          </td>
          <td class="enpRpt" style="text-align: center;">'
             .number_format($tr->lNumTraining).'
          </td>
       </tr>');
   }
   echoT('</table>');
```

Sample Output

And Sally again leads the way!



An Example: Working with Drop-Down List Entries

For this example, let's find all the staff members who have completed the course "Cleaning the Microwave for Beginners "

Files

Controller: `application/controllers/admin/dev/training_uwave.php`
Model: `application/models/admin/dev/mtraining.php`
View: `application/views/admin/dev/training_by_class_view.php`
Sample URL: `delightful/index.php/admin/dev/training_uwave/reportByMicroWave`

The Controller File

```
<?php
/*-------------------------------------------------------------------------
// Delightful Labor              copyright (c) 2015 Database Austin
// This software is provided under the GPL.
// Please see http://www.gnu.org/copyleft/gpl.html for details.
-------------------------------------------------------------------------*/
class training_uwave extends CI_Controller {

   function __construct(){
      parent::__construct();
      session_start();
      setGlobals($this);
   }

   function reportByMicroWave(){
   //-------------------------------------------------------------------------
   //
   //-------------------------------------------------------------------------
      $displayData = array();
      $displayData['js'] = '';

         // load models
      $this->load->model('personalization/muser_schema', 'cUFSchema');
      $this->load->model('admin/dev/mtraining',          'cTraining');

         // users trained in the ways of the microwave
      $strCourseName = $displayData['strCourseName'] = 'Cleaning the Microwave for Beginners';
      $this->cTraining->loadUserTrainingDDLValue(
                  $strCourseName,
                  $displayData['lNumUsers'], $displayData['training']);

         // breadcrumbs
      $displayData['pageTitle']      = anchor('main/menu/admin', 'Admin', 'class="breadcrumb"')
                              .' | Staff Training / Cleaning the Microwave';

      $displayData['title']          = CS_PROGNAME.' | Reports';
      $displayData['nav']            = $this->mnav_brain_jar->navData();

         // load the view for this report
      $displayData['mainTemplate']   = 'admin/dev/training_by_class_view';

         // make variables available to the view
      $this->load->vars($displayData);

         // load the template, which will then load this report's view
      $this->load->view('template');
   }
}
```

The Model Routines

The same model header as the previous example are used here. The following is the new method in this example:

```
function loadUserTrainingDDLValue($strCourseName, &$lNumUsers, &$training){
//---------------------------------------------------------------------
// return info about users who received training during the
// specified year
//---------------------------------------------------------------------
   $training = array();

      // load the personalized user table "Staff Training"
   $this->cUFSchema->loadUFSchemaViaAttachTypeUserTabName(
                  CENUM_CONTEXT_USER, 'Staff Training', $lTableID, true);

      // short-cut to our table of interest
   $staff = &$this->cUFSchema->schema[$lTableID];

      // field info for date of training
   $lIDX_DateOfTraining = $this->cUFSchema->lFieldIdxViaUserFieldName(
                              $lTableID, 'Date of Training', true);
   $strFN_DateOfTraining = $staff->fields[$lIDX_DateOfTraining]->strFieldNameInternal;

      // field info for subject
   $lIDX_Subject = $this->cUFSchema->lFieldIdxViaUserFieldName($lTableID, 'Subject', true);
   $strFN_Subject = $staff->fields[$lIDX_Subject]->strFieldNameInternal;

      // field info for duration
   $lIDX_Duration = $this->cUFSchema->lFieldIdxViaUserFieldName($lTableID, 'Duration', true);
   $strFN_Duration = $staff->fields[$lIDX_Duration]->strFieldNameInternal;

      // field info for notes
   $lIDX_Notes = $this->cUFSchema->lFieldIdxViaUserFieldName($lTableID, 'Notes', true);
   $strFN_Notes = $staff->fields[$lIDX_Notes]->strFieldNameInternal;

      // load the Drop-down list entries
   $this->cUFSchema->loadDDLValues($lTableID);

      // field name for "retired", or deleted record flag
   $strFN_Retired = $staff->strFieldPrefix.'_bRetired';

      // find the foreign key associated with the caller's course name
   $lDDL_FID = $this->lID_Via_Field_Value($staff->fields[$lIDX_Subject], $strCourseName);


      /* this sequel statement resolves to
            SELECT
               us_lKeyID, us_strFirstName, us_strLastName,
               UNIX_TIMESTAMP(uf000083_001841) AS dteTraining,
               uf000083_001843 AS strNotes,
               uduration.ufddl_strDDLEntry AS strDuration
            FROM uf_000083
               INNER JOIN admin_users ON uf000083_lForeignKey=us_lKeyID
               INNER JOIN uf_ddl AS uduration ON uduration.ufddl_lKeyID=uf000083_001844
            WHERE uf000083_001842=2656
               AND NOT uf000083_bRetired
            ORDER BY us_strLastName, us_strFirstName, uf000083_001841;
      */
   $sqlStr =
     "SELECT us_lKeyID, us_strFirstName, us_strLastName,
          UNIX_TIMESTAMP($strFN_DateOfTraining) AS dteTraining,
          $strFN_Notes AS strNotes,
          uduration.ufddl_strDDLEntry AS strDuration
      FROM $staff->strDataTableName
         INNER JOIN admin_users ON $staff->strDataTableFID=us_lKeyID
         INNER JOIN uf_ddl AS uduration ON uduration.ufddl_lKeyID=$strFN_Duration
      WHERE $strFN_Subject=$lDDL_FID
         AND NOT $strFN_Retired
```

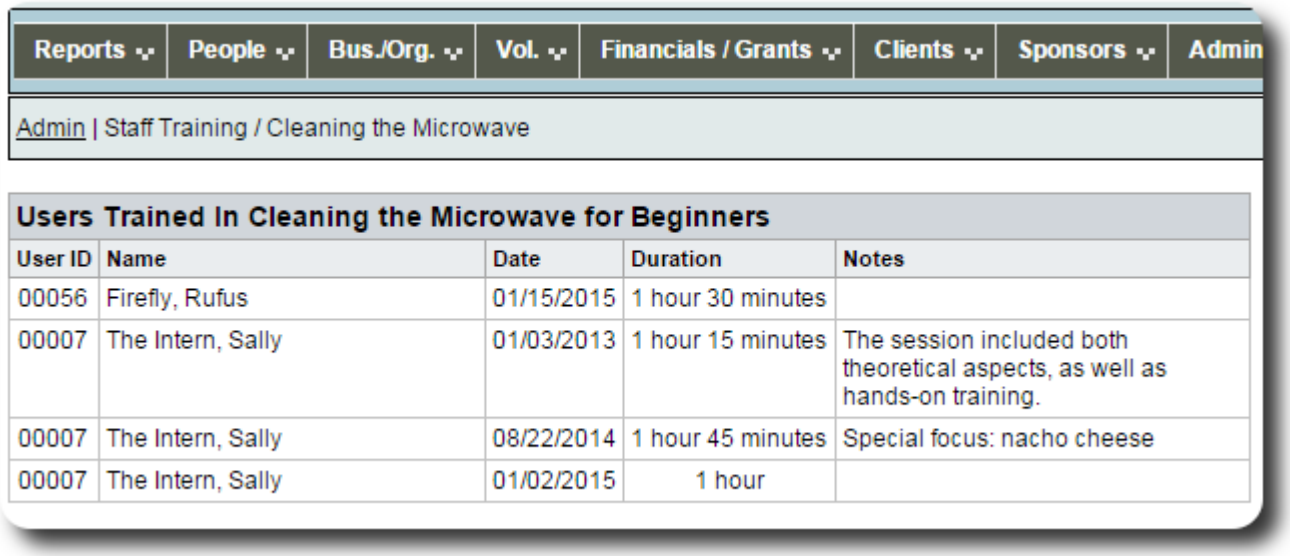
```
      ORDER BY us_strLastName, us_strFirstName, $strFN_DateOfTraining;";

   $query = $this->db->query($sqlStr);
   $lNumUsers = $query->num_rows();
   if ($lNumUsers > 0) {
      $idx = 0;
      foreach ($query->result() as $row){
         $training[$idx] = new stdClass;
         $training[$idx]->userID       = (int)$row->us_lKeyID;
         $training[$idx]->strDuration  = $row->strDuration;
         $training[$idx]->strNotes     = $row->strNotes;
         $training[$idx]->dteTraining  = (int)$row->dteTraining;
         $training[$idx]->strFName     = $row->us_strFirstName;
         $training[$idx]->strLName     = $row->us_strLastName;
         ++$idx;
      }
   }
}

function lID_Via_Field_Value($field, $strCourseName){
//---------------------------------------------------------------------------
//
//---------------------------------------------------------------------------
   $lReturn = null;

   for ($idx = 0; $idx < $field->lNumDDL; ++$idx){
      if ($field->ddlInfo[$idx]->strDDLEntry == $strCourseName){
         $lReturn = $field->ddlInfo[$idx]->lKeyID;
         break;
      }
   }
   return($lReturn);
}
```

The View File

```
<?php
/*-------------------------------------------------------------------------
// Delightful Labor                copyright (c) 2015 Database Austin
// This software is provided under the GPL.
// Please see http://www.gnu.org/copyleft/gpl.html for details.
-------------------------------------------------------------------------*/

   $strSafeClass = htmlspecialchars($strCourseName);
   if ($lNumUsers == 0){
      echoT('There were no training sessions for class '.$strCourseName.'<br><br>');
      return;
   }

      // write the table headings
   echoT(
      '<br>
         <table class="enpRpt">
         <tr>
            <td colspan="5" class="enpRptTitle">
               Users Trained In '.$strCourseName.'
            </td>
         </tr>');

   echoT('
       <tr>
          <td class="enpRptLabel">
             User ID
          </td>
          <td class="enpRptLabel">
             Name
          </td>
          <td class="enpRptLabel">
             Date
          </td>
          <td class="enpRptLabel">
             Duration
          </td>
          <td class="enpRptLabel">
             Notes
          </td>
       </tr>');

   foreach ($training as $tr){
      echoT('
       <tr>
          <td class="enpRpt" style="text-align: center;">'
             .str_pad($tr->userID, 5, '0', STR_PAD_LEFT).'
          </td>
          <td class="enpRpt" style="width: 140pt;">'
             .htmlspecialchars($tr->strLName.', '.$tr->strFName).'
          </td>
          <td class="enpRpt" style="text-align: center;">'
             .date('m/d/Y', $tr->dteTraining).'
          </td>
          <td class="enpRpt" style="text-align: center;">'
             .$tr->strDuration.'
          </td>
          <td class="enpRpt" style="width: 150pt;">'
             .nl2br(htmlspecialchars($tr->strNotes)).'
          </td>
       </tr>');
   }
   echoT('</table>');
```

Sample Output

And yet again, Sally paves the way....

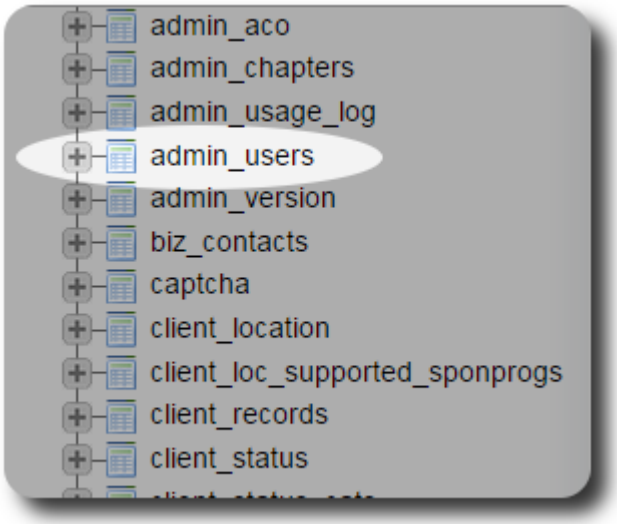


Appendix

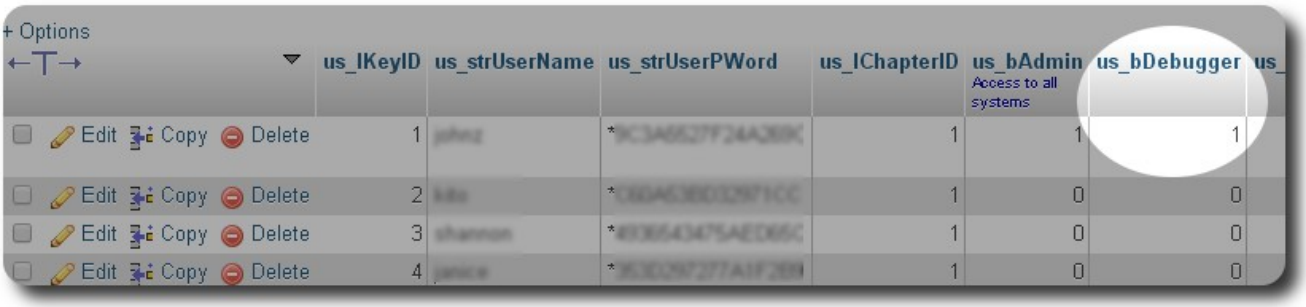
Setting the Developer Mode

Developers may find it useful to set the "developer mode" for their ***Delightful Labor*** account. When set, additional features become available within ***Delightful Labor***.

To set the developer mode via phpMyAdmin, navigate to the **`admin_users`** table:



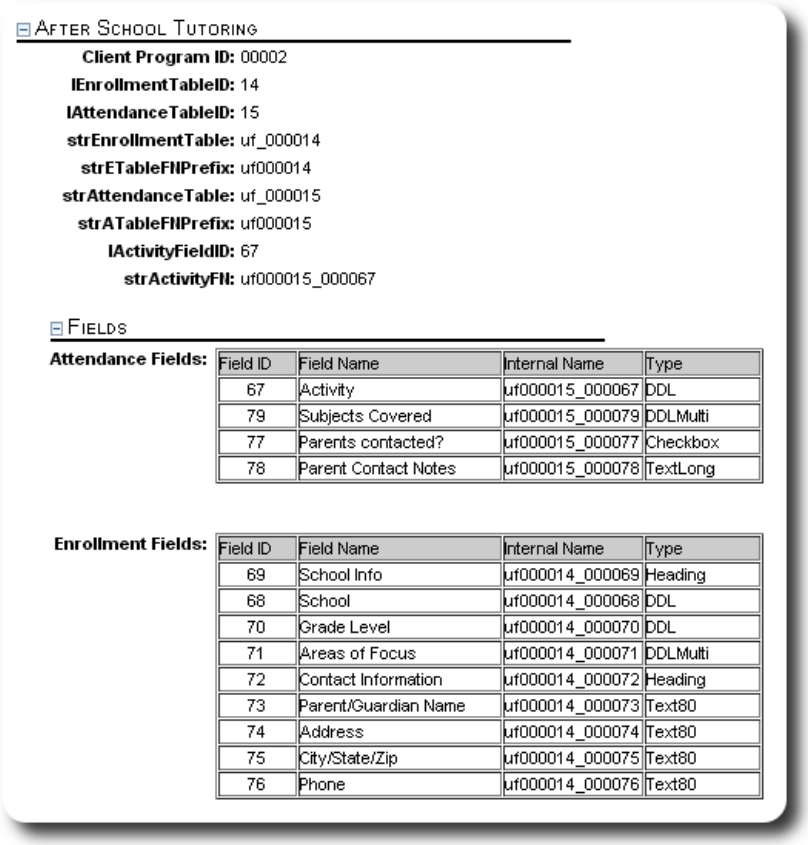
For the account of interest, set the field **`us_bDebugger`** to 1:



If you are currently logged into Delightful Labor, log out, then log back in. You will see some additional features highlighted by this icon: 

For example, when viewing client programs, you will now see:

Clicking on the debug icon will provide the following:



AFTER SCHOOL TUTORING

Client Program ID: 00002
lEnrollmentTableID: 14
lAttendanceTableID: 15
strEnrollmentTable: uf_000014
strETableFNPrefix: uf000014
strAttendanceTable: uf_000015
strATableFNPrefix: uf000015
lActivityFieldID: 67
strActivityFN: uf000015_000067

FIELDS

Attendance Fields:

Field ID	Field Name	Internal Name	Type
67	Activity	uf000015_000067	DDL
79	Subjects Covered	uf000015_000079	DDLMulti
77	Parents contacted?	uf000015_000077	Checkbox
78	Parent Contact Notes	uf000015_000078	TextLong

Enrollment Fields:

Field ID	Field Name	Internal Name	Type
69	School Info	uf000014_000069	Heading
68	School	uf000014_000068	DDL
70	Grade Level	uf000014_000070	DDL
71	Areas of Focus	uf000014_000071	DDLMulti
72	Contact Information	uf000014_000072	Heading
73	Parent/Guardian Name	uf000014_000073	Text80
74	Address	uf000014_000074	Text80
75	City/State/Zip	uf000014_000075	Text80
76	Phone	uf000014_000076	Text80

Please keep in mind that some of the developer/debugger features may not be fully functional! In addition, database tables used in development-only features may be rebuilt when the functions are officially released.

About Delightful Labor

Delightful Labor is a free, open source project created by Database Austin. ***Delightful Labor*** helps non-profits manage their contacts, donations, sponsorships, client programs, silent auctions, and volunteers.

Delightful Labor is available from SourceForge at
`https://sourceforge.net/projects/delightfullabor`

The ***Delightful Labor*** user's guide is available at
`http://www.delightfullabor.com/`