phpScheduleIt Developer Guide

Adding a New Field to a Reservation

Author: Nick Korbel - lqqkout13@users.sourceforge.net
Revision: 2005-11-15

About This Document
The goal of this document is to provide complete documentation on creating and adding one or more new data fields to a reservation in phpScheduleIt. It is assumed in this document that the developer is familiar with PHP, SQL, and HTML. The steps will start from the bottom with adding the field to the database, and work upwards to the HTML frontend. This will work in tandem with the phpScheduleIt developer documentation available for download from SourceForge.net.

Introduction
A common request on the phpScheduleIt forums is to expand the application by adding new fields to the reservation. These fields can be input, display, or metadata needed by the application for a custom use. This document will only describe adding an input field in detail, but will easily adapt to adding other field types. Adding new fields is a fairly simple process, provided the developer is familiar with PHP, SQL, and HTML.

Throughout this document we will be working with a Digital Camera resource. We will add an input field which lets the reservation creator enter a short description of the environment that they will be shooting in. This example will be referenced in the steps below. Code that is changed from the original source (as per phpScheduleIt version 1.1.0) will be highlighted in **`bold and blue`**.

Before you begin - Determine the type of field needed
You must first understand what kind of field you need to add. Will this be a textbox input field such as a summary, a read-only display field such as reservation creation date, or metadata such as reservation recurrence? This decision will also help you determine what type of column this field will be in your database.

Step 1 - Add the new field to the database
Before you write any code, you need to create a place to store the field in the database. This column needs to be created in the 'reservations' table of the phpScheduleIt database. Since phpScheduleIt supports many database backends, exact syntax and procedure for this depend on your database. Please consult your database documentation if you have any questions on inserting a new column into a table.

Example SQL for inserting our column

```
ALTER TABLE reservations ADD COLUMN environment CHAR(50);
```

Explanation
This statement will add a new column which is of type CHAR and 50 characters long named 'environment' to the 'reservations' table of the phpScheduleIt database. This statement assumes that you are operating on the phpScheduleIt database as a user who has permission to alter tables.

Step 2 - Support reading and writing to this column
Now that we have the column to write to and read from, we need to actually support the ability to do so through code. The new column for reservations will automatically be returned from the database when retrieving a reservation, so no changes are needed for that.

phpScheduleIt uses PEAR::DB to abstract the database type, so you may want to reference the PEAR::DB documentation at: http://pear.php.net/manual/en/package.database.db.php. All examples in this step refer to functions in the file **/lib/db/ResDB.class.php**.

Example function for inserting the field into a new reservation

```
function add_res(&$res, $is_parent, $userinfo, $accept_code) {
      $id = $this->get_new_id();

      $values = array (
                        $id,
                        $res->get_machid(),
                        $res->get_scheduleid(),
                        $res->get_start_date(),
                        $res->get_end_date(),
                        $res->get_start(),
                        $res->get_end(),
                        mktime(),
                        null,
                        ($is_parent ? $id : $res->get_parentid()),
                        intval($res->is_blackout),
                        $res->get_pending(),
                        $res->get_summary(),
                        $res->get_environment()
                  );

      $query = 'INSERT INTO ' . $this->get_table('reservations') . '
VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';
      $q = $this->db->prepare($query);
      $result = $this->db->execute($q, $values);
      $this->check_for_error($result);

      // ...
      // the rest of the function does not change
      // when adding a new field
      // ...
}
```

Explanation

The changes to this function are straightforward. We simply add the value from `$res->get_environment()` to the `$values` array. Then, we add an additional question mark to the statement `$query = 'INSERT INTO ' . $this->get_table('reservations') . ' VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';`. When inserting values into a table, the number of question marks must match the number of items in the `$values` array. In addition, the items in the `$values` array must be in the same order as the column names in your 'reservations' table. In our case the new column is the last one in the table, so the value of environment is last in the list.

Example function for updating the field in an existing reservation

```
function mod_res(&$res, $users_to_add, $users_to_remove, $accept_code)
{
      $values = array (
                        $res->get_start_date(),
                        $res->get_end_date(),
                        $res->get_start(),
                        $res->get_end(),
                        mktime(),
                        $res->get_summary(),
                        $res->get_pending(),
                        $res->get_environment(),
                        $res->get_id()
                  );

      $query = 'UPDATE ' . $this->get_table('reservations')
                  . ' SET '
                  . ' start_date=?,'
                  . ' end_date=?,'
                  . ' startTime=?,'
                  . ' endTime=?,'
                  . ' modified=?,'
                  . ' summary=?,'
                  . ' is_pending=?,'
                  . ' environment=?'
                  . ' WHERE resid=?';

      $q = $this->db->prepare($query);
      $result = $this->db->execute($q, $values);
      $this->check_for_error($result);

      // Update the owner of the reservation

      // ...
      // the rest of the function does not change
      // when adding a new field
      // ...
}
```

Explanation

Updating our new field is very similar to inserting it. We have to get the value of the field from the `$res->get_environment()` function and put it in the `$values` array. Next, we add some SQL to assign a value to that column. To do this, we add a comma

after the `is_pending=?,` code, then add the assignment in the next line with the code `. ' environment=?'`. Again, the placement of the value in that list is critical and must match up with the correct number of question marks. In this case it matches up with the 8th question mark, so it is the 8th value in the list.

Step 3 - Add the field to the Reservation class

phpScheduleIt attempts to balance object oriented design principles and code execution speed for an optimal point between ease of code and speed of code. This means that class objects are used when deemed necessary. The idea of a reservation lends itself nicely to object orientation and is represented in **/lib/Reservation.class.php**, where these code changes will be made.

Example for adding the 'environment' property in the Reservation class definition

```
class Reservation {
      var $id           = null;                         //    Properties
      // ...
      // these properties do not change
      // ...
      var $users        = null;                         //
      var $environment  = null;                         //
```

Explanation

This statement just makes a class property named 'environment' available. We will use this to store and retrieve the value of the field.

Example function for getting the value of the property

```
/**
* Returns the environment for this reservation
* @param none
* @return string environment
*/
function get_environment () {
      return $this->environment;
}
```

Explanation

This function returns the value of the 'environment' class property. This function name must match up with the function call we made in the previous step to return the value of the property.

Example for loading the property value from the database

```
function load_by_id() {
      $res = $this->db->get_reservation($this->id);   // Get values
from DB

      if (!$res)        // Quit if reservation doesnt exist
            CmnFns::do_error_box($this->db->get_err());

      $this->start_date = $res['start_date'];
      $this->end_date   = $res['end_date'];
      $this->start      = $res['startTime'];
      $this->end        = $res['endTime'];
      $this->machid     = $res['machid'];
      $this->created    = $res['created'];
      $this->modified   = $res['modified'];
      $this->parentid   = $res['parentid'];
      $this->summary    = htmlspecialchars($res['summary']);
      $this->scheduleid = $res['scheduleid'];
      $this->is_blackout= $res['is_blackout'];
      $this->is_pending = $res['is_pending'];
      $this->environment= $res['environment'];

      $this->users = $this->db->get_res_users($this->id);
      // Store the memberid of the owner
      for ($i = 0; $i < count($this->users); $i++) {
            if ($this->users[$i]['owner'] == 1) {
                  $this->memberid = $this->users[$i]['memberid'];
                  break;
            }
      }
}
```

Explanation

The reservation data is being retrieved from the database in the first line of this function. We then set the value of our 'environment' property to the column that is returned from the database.

Example function for adding a new reservation (phpScheduleIt version 1.0, 1.1)

```
function add_res($machid, $userinfo, $ownerid, $start_date, $end_date,
$start, $end, $repeat, $min, $max, $summary = null, $scheduleid,
$environment = null) {
      $this->machid     = $machid;
      $this->memberid   = $ownerid;
      $this->start_date = $start_date;
      $this->end_date   = $end_date;
      $this->start      = $start;
      $this->end        = $end;
      $this->repeat     = $repeat;
      $this->type       = RES_TYPE_ADD;
      $this->summary    = $summary;
      $this->scheduleid = $scheduleid;
      $this->environment= $environment;

      // ...
      // the rest of the function does not change
      // ...
```

Explanation

This is the function which is called when we want to create a new reservation in the database. We need to change the function header to accept the new field. We also need to add a line which stores that value to the class property. *Note: This code change does not apply to phpScheduleIt version 1.2 and higher.*

Example function for modifying an existing reservation (phpScheduleIt version 1.0, 1.1)

```
function mod_res($ownerid, $userinfo, $removed_users, $start_date,
$end_date, $start, $end, $del, $min, $max, $mod_recur, $summary = null,
$environment = null) {
      $recurs = array();

      $this->load_by_id();                // Load reservation data
      $this->type = RES_TYPE_MODIFY;
      $this->summary = $summary;
      $this->start_date = $start_date;
      $this->end_date = $end_date;
      $this->start = $start;              // Assign new start and end
times
      $this->end   = $end;
      $this->memberid = $ownerid;
      $this->environment = $environment;

      // ...
      // the rest of the function does not change
      // ...
```

Explanation

This function is called when we want to update an existing reservation. The changes here are the same which are needed in the previous example. We need to change the function

header to accept the new field. We also need to add a line which stores that value to the class property. *Note: This code change does not apply to phpScheduleIt version 1.2 and higher.*

Step 4 - Accept the values from the form post

This step will take values that are submitted through the reservation form, validate them, and pass them to the correct Reservation class function. The code changes here apply to the **reserve.php** file.

Example for reading and validating the input

```
// ...
$environment = isset($_POST['environment']) ?
stripslashes(htmlspecialchars($_POST['environment'])) : '';

if ($fn == 'create')
// ...
```

Explanation

This code accepts the 'environment' value from the $_POST array. If the value is set, it strips any slashes that were added, then encodes the html to prevent security issues such as cross site scripting.

Example for assigning the value to the Reservation (phpScheduleIt versions 1.0, 1.1)

```
// ...
$res->add_res($_POST['machid'], $invited_users, $_POST['memberid'],
$start_date, $end_date, $_POST['startTime'], $_POST['endTime'],
$repeat, $minRes, $maxRes, stripslashes($_POST['summary']),
$_POST['scheduleid'], $environment);
// ...
```

Explanation

All we do here is add the variable to the function call. The function header was changed in the last step to accept this new variable.

Example of assigning/modifying the value to the Reservation (phpScheduleIt version 1.2)

```
// ...
$res->memberid    = $_POST['memberid'];
$res->start_date= $start_date;
$res->end_date    = $end_date;
$res->start       = $_POST['startTime'];
$res->end         = $_POST['endTime'];
$res->summary     = stripslashes($_POST['summary']);
$res->allow_participation = (int)isset($_POST['allow_participation']);
$res->allow_anon_participation =
(int)isset($_POST['allow_anon_participation']);
$res->environment = $environment;
// ...
```

Explanation
All we need to do is assign the 'environment' property of the Reservation object before the call to `$res->add_res();`

Example for modifying the value (phpScheduleIt versions 1.0, 1.1)

```
// ...
$res->mod_res($_POST['memberid'], $invited_users, $removed_users,
$start_date, $end_date, $_POST['startTime'], $_POST['endTime'],
isset($_POST['del']), $minRes, $maxRes, isset($_POST['mod_recur']),
str_replace("\n", "", $_POST['summary']), $environment);
// ...
```

Explanation
All we do here is add the variable to the function call. The function header was changed in the last step to accept this new variable.

Step 5 - Display and input the new field
The only thing left to do is make this new field available to the web browser. This step is very open to developer design input, so we will only go into detail about which HTML form elements need to exist; we will only briefly discuss where they can be placed.

All reservation form display functions are in **/templates/reserve.template.php**. They are all called from the `print_res()` function in the Reservation class. If you, as the designer, decide that your new field or fields should be wrapped in their own display function, you will add the call to the function in `print_res()` or the `print_basic_panel()` function in the reservation template file.

In our example, we are going to add a new function called print_environment() to the template file. This will handle display and input for the environment field.

Example for adding the function call

```
function print_basic_panel(&$res, &$rs) {
      // ...
      print_summary($res->summary, $res->type);
      print_environment($res->get_environment(), $res->type);
      // ...
```

Explanation
We need to display the environment field in this function, but we also need to allow a user to enter it (if this is a reservation creation), or edit it (if this is a reservation modification). The reservation type is passed so that we can determine if we should just display the field or if it needs to be editable.

Example of the print_environment() function

```
/**
* Print out the reservation summary or a box to add/edit one
* @param string $environment environment to edit
* @param string $type type of reservation
*/
function print_environment($environment, $type) {
?>
   <table width="100%" border="0" cellspacing="0" cellpadding="1">
    <tr class="tableBorder">
     <td>
      <table width="100%" border="0" cellspacing="1" cellpadding="0">
      <tr>
            <td class="cellColor"><?=translate('Environment')?></td>
            <td class="cellColor" style="text-align: left;">
            <?
            if ($type == RES_TYPE_ADD || $type == RES_TYPE_MODIFY)
                  echo '<input type="text" name="environment"
class="textbox" value="' . $environment . '"/>';
            else
                  echo (!empty($environment) ? $environment : '');
            ?>
            </td>
         </tr>
      </table>
     </td>
    </tr>
   </table>
<?
}
```

Explanation

This function will either allow the user to enter/modify a value for the new field, or it will display it. The logic for which HTML to display is based on the $type variable. If the type is a reservation add or a reservation modify then we display a textbox for entering the value. If it is any other type, we just print it. When we are displaying the text box, we set the value property to the value of the field. This way, if we are modifying a reservation the box will be populated.

The call to `translate('Environment')` uses phpScheduleIt's localization functionality. This assumes that the string 'Environment' is in the translation files. To add it to the translation files, simply add `$strings['Environment'] = 'Environment';` anywhere in the language files that you need. All language files are located in the /langs/ directory. Alternatively, you can leave out the call to the `translate()` function and simply hardcode the string 'Environment' into the HTML.

Summary

That's it! We have successfully added a new field to the reservation that the user can enter and modify and all other users can view. With a little work these steps can be altered to add multiple fields or different types of fields.