Wednesday, November 4, 2009

* Recently downloaded trial verison of PhpED
* Copied in source form Chapter 27 of book PHP and MySQL Web Development (purple book)
* Cloned and updated script PWDCreate18.sql for these adjustments:
  + New database = PWD2009
  + Added database file = users
  + Added logic to create user profile to access data
* Started populating PHP project with objects:
  + CSS sheet
  + Sample HTML page
  + Database\_functions.php

Thursday, November 5, 2009

* Messed around some more with the utility that creates tables in HTML; very cool
* Decided to see what it looks like up on the site… FTPed the project up to atonllc.com – realized we need the SQL setup!
* Spent some time getting ghte database setup on the atonllc site
* Aligned database names and user ids so that they would be the same in both places – and I could use one file of config info to supply IDs and passwords, and database names, etc…
* But the locic is working and easily uploaded to the aton website…

Tuesday, Nov 10

* Built on research and testing from yesterday…
* Modified the template for creating DB-Forms (standard.phpt in the “plugins” folder) to accommodate behavior I want to see in the PWD application. Main objectives:
  + Regenerate anytime – reduce reliance on updating after code generation
  + Add in PWD header and footer logic
  + Eliminate the page shown to “confirm” adds/inserts/deletes
* Works great!!

Wednesday, Nov 11

* Finished off logic to add buttons for existing from the insert and update functions – as well as to leave and go back to the main menu
* Loaded everything up on the atonllc website and messed around with it a bit
* Recognized that the PC based mysql application always creates tables with lower case table names… but the unix favor does not – it permits mixed case…

Thursday, Nov 12

* Worked on table creation:
  + Found how to sort the tables
  + Found how to capitalize the table name
* Created routine to delete and re-create the MySQL database…

Friday, November 13, 2009

* Needed to separate the tasks of dropping and creating the database from the drop and create of the tables; I cannot create a script on the “hosted” server that will drop/create the schema (database) because that is an admin function that must be done in the GIU. As a result, I needed to code in logic to drop all of the tables individually (since they must be dropped in order – foreign keys, etc..
* Reworked header and footer and added log-out and “home” options on footer; moved the user id they are logged in as down to the footer…
* Started messing with logic to send forgotten password
* Did logic to more gracefully deal with trying to add a record that is already in the database – and is setup as unique; give button to send to main table
* Realized that the program was searching for a dictionary… searched for a list of words via google, downloaded a text file (this is common on Unix – and is stored in /usr/dict/words)

Monday, November 16, 2009

* Started to spit out config files to be unique by computer name
* Started to deal with the task of assigning a variable name for the location of the dictionary…

Wednesday. November 18, 2009

* Worked through extracting my code from the DBFORMS standard template – so I do not wind up overlaying it again….
* Added new table = valid\_emails – the list of emails for people who CAN create an account.
* Updated database routine to drop/add new table; also new function to add an administrator email so that they can login and start assembling the site when this is initialized.
* Added logic to access the email table when registering… if no email address present – no registration.
* Made first pass at adding maintenance routines for “valid emails” and “users” – mainly maintenanance functions

Thursday, November 19, 2009

* Changed “User” file to be keyed by an integer – not the user ID itself…
* Added a field for “Admin?” function
* Updated database initialization to add record to user file to permit administration
* Trimmed back the foreign values in the “Racer” table; dropped “car number”, “race”, and “heat” – seeking simplicity; dropped most of the indexes over racer… it seemed to be overly heave with unique keys where they did not belong…
* Figured out how to add a drop down box on the Racer Update panel so that we can pick the den that the racer belongs to… very cool!!

Friday, November 20, 2009

* Added more dropdowns – to eht usre panel and the den panel so that we can control input
* On user panel, hid the entry of the password
* In user table, if a new record created, assign a password behind the scenes as the same as the email address

Saturday, November 21, 2009

* Planned out architecture and screens for rest of project
* Reworked the Race table

Sunday, November 22, 2009

* Coded Race table
* Coded Race\_registration table
* Cut “heat” table down to 4 lanes (simplify!)
* Cut “Race\_heat down to 4 lanes, 4 places

Monday, November 23, 2009

* Did a lot of work to get the “race” maintenance program going
* Started experimentation with how to plug in race-registration; I think I need to:
  + Create a new function in db\_utils that I can pass a “where” clause into… then I could limit the screen for race-registration to the race that I stuffed into the session variable…

Tuesday, November 24, 2009

* Modified the db\_utils.inc functions to add a “where” clause to give more flexibility – and help me select only records I need in some of these functions…
* Tested and fixed new “select” functionality.. seems to work OK

Wednesday/Thu November 25-26

* Finished up the race and race-registration pair of programs

Friday, November 27

* Finished up first draft of the heat and the heat header updates
* Pulled in data from the Java programs and converted to PHP program to populate the database

Saturday, November 28, 2009

* Hammered through the creation of the race\_heat (score entry) routine – but have not yet tested it beyond just hitting the button to go into it
* Updated the “init” routine to create all of the dens – as well as den 1 kids.
* Next up: test out creating a race and entierng the racers and the score…

Monday, November 30, 2009

* Work on Race table (race maintenance) routine
* Add drop down to restrict selections on “den level”
* Move assignment of “den level” array into constants include – to make the same for any places we need to show den levels.
* To do: keep working on the logic to recognize that the race is now “started” and punch out the race\_heat records….

Tuesday, December 1, 2009

* Started to look over the logic in the race\_heat table – focusing on the record edit page (we should never be creating records in this routine when this is done).
* Realized that the values going into the race\_heat file are car numbers – and they SHOULD be racer #’s… I fixed this and now the file is being updated properly…

Thursday, December 4, 2008

* Cleaned up the race\_heat maintenance program; made it work and show racer names, etc.
* Left to be done at this point:
  + Edit the panel input; need to ensure that the same racer not specified twice; need to make sure every racer specified once.

Sunday, December 06, 2009

* Hammered out auto-refreshing race heat results page; also created flash to second page with racer scores

Monday, December 07, 2009

* Cleaned up log in page and started identifying repair needed and doing bug bashing

Tuesday, December 8, 2009

* Issues list

Wednesday, December 9, 2009

* Issues list

Thursday, December 10, 2009

* Issues list – Race table improvements

Friday, December 11, 2009

* Add grade (school grade) to den table;
* Changed the sort of dens on the drop down for a new racer – now organized by school grade then den #
* Fixed problems with registration when a valid email address is

Saturday, December 12, 2009

* Lots more bug bashing
* Recover from overwriting the table\_racer source

Sunday, December 13, 2009

Monday, December 21, 2009

* Changed race\_registration:
  + to make racer “display only” unless the race is in a “registering” status. Avoided some mean bugs here..
  + To make car number “display only unless the race is in a “registering” status
  + Unless user is an “administrator” type, ALWAYS hide the points earned and place in race values

Tuesday, December 22, 2009

* Corrected problem with auto-refresh on “registration display”
* Added ability to restrict race display by “den level” (checkbox selections)
* Fixed bug created yesterday (!)
* Changed the radio button on the work with races panel to “stick” to the last race action was performed on.
* Added display of total number of heats on both of the panels in the auto-refresh sequence.
* Added text in footer to remind user they are an administrator
* Moved the “words file right into the PWD directory to eliminate the hassle of locating it in a directory… the password reset function is now working…

Wednesday, 12/23

* Added radio button to change sort on race registration screen to permit sort by car number or place-in-race
* Added javascript option to print the race registration page!
* Updated logic to update the place-in-race value in the race\_registration file

Thursday, 12/24

* Added logic to update the race file every heat that completes with the top three scorers so far in the race (recognizing we could have “ties” that are not showing up…
* Added display of race leaders in both auto-refresh pages…

Saturday, December 26, 2009

* Added logic to be able to turn auto-refresh on/off
* Added logic to control interval between auto-refreshes
* Finished up the two auto-refreshing panels till I thought they were good to go
* Added in logic to fully populate the database

Friday, January 1, 2010

* Hammered in the logic needed to store the link between the heat and the video… also updated the auto-refresh stuff to cycle in the video of the most recent heat…
* Fixed the problems with creating users and updating users – so now a password may be assigned on the spot.
* Also fixed the logic that would show “additional pages” on subfiles that were really already subsetted by race…
* Also added “print” on the footer – so always there
* Added logic to reverse order of columns on heat maint program so facilitate printing the listings for the race organizers…
* Made the heat maintenance more readable and added color on lines

Wednesday, Jan 6th

* Added new panel to show “on-deck” for main audience
* Updated the existing panels to show the number of heats so far
* Added a “prefix” in the race definition that will precede each car number

TO DO:

* Visit this site to sort out how to embed videos: http://www.walkernews.net/2008/08/17/windows-media-player-how-to-embed-wmv-file-in-html-code/
* Fix the “add user” so that you can assign a password when you create the user
* Add a “quit” button on the table maintenance to return to the main menu
* Add a “quit” button on the insert/update panels to permit escape
* Make the main menu look more like a main menu
* Find a way to create database on ATONLLC easier than I have been doing… perhaps a php script?
* Figure out how to sort den maint by den number; we will need sorting as we go forward…
* Merge security between old website stuff and new table logic… get it together!
* Make sure that the password is required to access all pages
* Add logic to look against a list of email addresses that are permitted – not ANYONE can sign up…
* If a duplicate is encountered, we loose our footers and the users cannot “get back!”
* Wrap up dealing with emailing lost password
  + get\_random\_word – source of problems… disctionary look up
  + settle the location of the “words” file and add that to the file of options;
* Clean up login page
* Create separate config.inc.php file for different servers…
* Figure out whatelse may have been overwritten- the utility functions associated
* Assign a field to the user file – “administrator; show admin functions only to users that are admins
* Search and replace all references to “bookmark”
* Re-biuld the “User” maintenance table
* Hide the admin functions if the user does not have “admin = Y in their “User” record
* Change the sort of the dens showing up in the drop down on the racer table. Is it getting them in arrival sequence? It should be sorting them by “den level” then den #...
* Make the drop down show the correct value when looking at the “update” mode for a record in the racer table.
* Can we make the racer table re-sort when records are added – they seem to just go to to the bottom… this may be true for ALL tables… (probably need to find all occurrences of table load… )
* Hide the racer ID on the racer table
* Race maintenance work
  + Logic to prevent incorrect status changes (closed to registering, etc)
    - Changes should not matter much… each time you change to “Starting, it will punch out new race\_heat records – but only if none exist already…
  + Add den level to race file as a new field
  + Modify drop downs to only pick racers registered to this race
  + Make sure that description is non blank
  + OK if there is no racer for “places” – assign a null value to database fields
  + OK if there is no heat assigned – assign a null value to database fields
  + If race defined – but no heat assigned ye t- do not permit access to registration
  + If any racers registered, do not permit the change to the heat assignment
  + On delete of a race, verify – then purge all lower level records (race\_registrations and race\_heats)
* Race registration maintenance
  + Improve look of main table
    - Get rid of key
    - Replace screen name
    - Add at heading the race that is being registered
    - Show racer name rather than foreign key
    - Sort panel by car# (may already be done)
    - Spread out page a little
    - DO NOT PERMIT INSERT AFTER MAXIMUM NUMBER OF CARS HAVE BEEN REGISTERED
      * The insert panel already will not let you – there is no number assigned; not perfect, but gets the job done…
    - Do not permit racers to be removed – or car numbers changed if race is in progress…
      * Racer table
      * Race registration
  + On insert
    - Only allow selection of the race we are working on – in fact make it output only
    - Limit selection of racers to those in the den level in the race (may be phase 2)
    - Drop down selection of car numbers?
    - Don’t let same racer get registered more than once…
  + On update
    - Same as insert
* Do not allow user to hit button for updating race results of there are no records; better yet – do not permit into race results if there are no records…
* In Race Maintenance: If we switch from “Started” to “registering” – we wind up with race\_heat records stranded there… perhaps:
  + If there is any scoring – do not let them switch from Started to Registering
  + If there is no scoring, delete the race\_heat records
* Why is there a “111111111 ” in the top of the screen when you go from registering to Started in race maintenance?
* Modify programs that exit to stop at the w/w races table – rather than ejecting to the menu…
* Race heat update (record results)
  + Ensure that each racer is entered (no blanks)
  + Ensure that no racer is specified twice
  + Use results to go out and update the race\_registration file with total scores (probably do a full recalc of racer scores on each update action
* Show some help
* Show some recent bug fixes
* Any functions we want to limit to a single user?
  + How do we set a flag and then follow it to ensure user didn’t walk away?
* Improve the table for showing the heat race results
  + Create constants for the table setup and implement so we can change the look and feel of the table on the fly – perhaps even the color!!
  + Setup the refresh rate to be controlled by the constants file – or slow down to every 10 or 15 seconds
  + Remove top part of table header??
* Prevent maint of race\_registration table while race in progress (unless an admin?). This will prevent data corruption or competition with job that does the update of the values…
* Capitalize all lettersin in EMAIL address wehne entered into email maintance; add logic to capitalize entries when doing scan on new member sign up (in order to avoid “case” problems…
* Why doesn’t the member sign up work when there is a valid email address in the file? Need debug…
* Figure out what can and cannot be updated when race is in a “complete” status
* Add logic to assign “complete” status when last race heat is run; also assign 1/2/3 place racers at that time.
* Prevent change to den level in race maintenance if there are any scouts registered to race.
* Add logic to update racer scores when heat is complete (or is that there?
* Limit selection of racers for 1/2/3 place in race maintenance – to those in the right den level; prevent entry if a den level not assigned.
* Permit no assignment to Heat or 1/2/3 place racers in race maintenance.
* Default “next” car number when doing an insert in the race\_registration file; if a new record hide points earned and “place in race”
* When “exit” from race\_registration, return to W/W Races
* Prevent access to “Race Results” if the race is not in a “started” status
* When exit from “Race results,” return to W/W Races
* Add logic to replace hard coded race leaders in the “Race Results” panel
* Prevent race delete if any racers are registered
* Prevent delete of registered racer if there are any heats run.
* Add confirmation screen when you hit the “initialize database” option
* Create display of all past race winners to display
* Display for race-head
* On panel where scores are entered – move radio button for “update” down to first record that needs to be updated
* Prevent den deletes if any racers refer to sen – or if specified on any races
* Prevent racer deletes if any race registration refer to them.
* Find out why I was getting the strange message on the auto-refresh pafge – related to quarty limit; was the second refresh page; somehow cam up on both local and remote servers at the same time…

**Bug/Enhancement list:**

Den maintenance

* More edits on “den level”
* Not resorting by den # if you change a den number

Change password

* Is showing menu under it!

Registration

* Might be a problem if you are already signed in… perhaps force session closed if they go to this page… force them to log off

User maintenance

* Hide password in record updates
* Ensure that the user ID has no spaces and is less than a certain length…
* Send an email to the user welcoming them to the website with their password = to their email address

FUTURE:

* Show a list of other people on the website? Follow up on the logic for handling the sessions via the database; Maybe both??
* Create a table that will keep track of bugs & fixes
* Add a java program that will read the serial port and update a file
  + Or: http://serial-com-port-monitor.findmysoft.com/
  + Or: http://www.giveawayoftheday.com/freeware/windows/xp-serial
  + Search for serial port monitor for Windows XP

Table CASE tool notes

|  |
| --- |
| Den table:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Add this line:  require\_once "constants.inc.php";  After create, locate and change this line to hide key field:  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'IdDen', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  After create, locate and change this line to sort by den number:  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, “Den\_number” );  Locate these three lines:  <table border="0">  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then insert afterwards:  if ($fld[FLD\_DATABASE] == 'Den\_level') {  // Be sure to have dtaabase value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // following no longer needed – part of constants include file  ~~// Assemble an array of den levels to show in drop down selection box~~  ~~$denLevel['Tiger'] = 'Tiger';~~  ~~$denLevel['Wolf'] = 'Wolf';~~  ~~$denLevel['Bear'] = 'Bear';~~  ~~$denLevel['Webelos I'] = 'Webelos I';~~  ~~$denLevel['Webelos II'] = 'Webelos II';~~  ~~$denLevel['Sibling I'] = 'Sibling I';~~  ~~$denLevel['Sibling II'] = 'Sibling II';~~  // Punch out start of HTML  echo "<tr><td>Den level</td>";  echo "<td><select name = 'f4'>\n";  // Output each individual drop down entry  foreach ($denLevel as $key => $choice) {  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry shoudl show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  } else {  Will also need a closing bracket after CASE construct |
|  |
|  |

|  |
| --- |
| User table:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  After create, locate and change this line to hide key field:  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idUser', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'passwd', FLD\_DISPLAY\_SZ => 100,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 100,  Hide password: Replace as follows:  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 100,  FLD\_INPUT => true, FLD\_INPUT\_TYPE => 'text',  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 100,  FLD\_INPUT => true, FLD\_INPUT\_TYPE => 'hidden',  After create, locate and change this line to sort by user id:  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, ‘username’);  Add logic to show dropdown for admin flag; locate these three lines:  <table border="0">  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then insert this logic afterwards:  // If this is the "administrator?" field, show as a "Yes" or "No" drop down box  if ($fld[FLD\_DATABASE] == 'admin') {  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Assemble an array of options for whether user is an admin  $adminOptions['N'] = 'Not an administrator';  $adminOptions['Y'] = 'Is an administrator';  // Punch out start of HTML  echo "<tr><td>Administrator?</td>";  echo "<td><select name = 'f3'>\n";  // Output each individual drop down entry  foreach ($adminOptions as $key => $choice) {  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry shoudl show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  } else {  A closing bracket will also be needed after the case statement below  Handle automatic assignment of password to new users (same as email address)  Locate this line in code:  case 'i2':  Add this logic right afterwards:  // If password value is a temp value, replace it with the encrypted version of their email address  if ($\_POST['f2'] == 'tempvalue') {  $\_POST['f2'] = sha1($\_POST['f4']) ;  }  Locate this pair of lines:  case "hidden":  echo "<td><input name=\"$fkey\" type=\"$fld[FLD\_INPUT\_TYPE]\" value=\"$val\" /></td></tr>";  Replace with these:  case "hidden":  if ($fld[FLD\_DATABASE] == 'passwd') {  echo "<td><input name=\"$fkey\" type=\"$fld[FLD\_INPUT\_TYPE]\" value=\"tempvalue\" /></td></tr>";  } else {  echo "<td><input name=\"$fkey\" type=\"$fld[FLD\_INPUT\_TYPE]\" value=\"$val\" /></td></tr>";  } |
|  |
|  |

|  |
| --- |
| Racer table:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  After create, locate and change this line to hide key field & den field (foreign key):  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer ID', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Den', FLD\_DISPLAY\_SZ => 7,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  After create, locate and change this line to sort by last name then first name:  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, 'last\_name,first\_name');  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, ‘username’);  Locate these three lines:  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  if ($fld[FLD\_DATABASE] == 'Den\_IdDen') {  Then add in this logic:  if ($fld[FLD\_DATABASE] == 'Den\_IdDen') {  // Be sure to have dtaabase value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);    // Assemble an array of dens tro show in drop down selection box  $sql = "SELECT \* FROM den";  $result = $dbconn->db\_get\_all\_rows($sql);  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  $denArray[$value[IdDen]] = $value[Den\_name];  }  }  // Punch out start of HTML  echo "<tr><td>Den</td>";  echo "<td><select name = 'f3'>\n";    // Output each individual drop down entry  foreach ($denArray as $key => $choice) {  echo "<option value='$key' ";    // If the entry is the same as the existing database value  // indicate this entry shoudl show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  } else {  Will also need a closing bracket after CASE construct |
|  |
|  |

|  |
| --- |
| Race table:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Add this line  require\_once "constants.inc.php";  Locate this set of code:  function errorHandler() {  ?>  <form method="LINK" ACTION="<?php "table\_".$table.".php"?>">  <input type="submit" value="Go back">  </form>  <?php  }  And then add this right after it:  /\* Function errorHandler2  \* Purpose: send user a message and return themn to the main table  \*  \*  \*/  function errorHandler2($errorMessage) {  ?>  <body>  <h5><?php echo ($errorMessage); ?></h5>  </body>  <form method="LINK" ACTION="<?php "table\_".$table.".php"?>">  <input type="submit" value="Go back">  </form>  <?php  }  After create, locate and change this line to hide fields:  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRace', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => '# of Tracks', FLD\_DISPLAY\_SZ => 5,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE =>false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 5,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idHeat\_Header', FLD\_DISPLAY\_SZ => 7,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRacer1', FLD\_DISPLAY\_SZ => 7,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRacer2', FLD\_DISPLAY\_SZ => 7,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f7' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRacer3', FLD\_DISPLAY\_SZ => 7,  'f7' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  Add this set of lines (only because it was easier to do this than to regenerate and update routine):  'f8' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Den level', FLD\_DISPLAY\_SZ => 20,  FLD\_INPUT => true, FLD\_INPUT\_TYPE => 'text',  FLD\_INPUT\_SZ => 20, FLD\_INPUT\_MAXLEN => 20, FLD\_INPUT\_DFLT => '',  FLD\_INPUT\_NOTEMPTY => true, FLD\_INPUT\_VALIDATION => '',  FLD\_DATABASE => 'Den\_level'),  Change the value of “input validation from Numeric to nothing for these fields:  FLD\_INPUT\_NOTEMPTY => true, FLD\_INPUT\_VALIDATION => '',  idHeat\_Header. idRacer1, idRacer2, idRacer3  locate and change this line to sort by race description  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, ‘Description’);  Locate this line:  <?php // INPUT  And input this code block:  // Assemble a list of possible racers that are already registered for this race  $racerArray[0] = "Not assigned";  $sql = "SELECT First\_name, Last\_name, idRacer FROM racer, race\_registration WHERE ((rrRace =".$row[f0].") and (idRacer = rrRacer)) order by Last\_name, First\_name";  $result = $dbconn->db\_get\_all\_rows($sql);  if ($result) {  foreach($result as $key => $value) {  $racerArray[$value[idRacer]] = $value[First\_name]." ".$value[Last\_name];  }  }  Then right after this line:  foreach($fielddef as $fkey=>$fld) {  Then add in this logic:  switch ($fld[FLD\_DATABASE]) {  // Show a dropdown box of possible race statuses (from "constants" include file)  case 'Race\_status':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Race status</td>";  echo "<td><select name = 'f2'>\n";  // Output each individual drop down entry  foreach ($raceStatus as $key => $choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  // Save current status  $currentRaceStatus = $val;  break;  case 'Tracks':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td># of Tracks</td>";  echo "<td><select name = 'f3'>\n";  // Output each individual drop down entry  foreach ($trackSizes as $key => $choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'idHeat\_Header':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Heat assignment</td>";  echo "<td><select name = 'f4'>\n";  // Assemble a list of possible heat assignments  $sql = "SELECT \* from heat\_header order by description";  $result = $dbconn->db\_get\_all\_rows($sql);  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  $heatHeaderArray[$value[idHeat\_Header]] = $value[Description];  }  }  // Need option for "not assigned" (zero in database field)  echo "<option value = 0";  if (($val == NULL) or ($val == 0)) {  echo " selected";  }  echo " >Not assigned</option>\n";  // Output each individual drop down entry  foreach ($heatHeaderArray as $heatKey => $heatChoice){  echo "<option value='$heatKey' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $heatKey) {  echo " selected";  }  echo " >$heatChoice</option>\n";  }  echo "<select></td><tr>";  break;  case 'idRacer1':  // If current race status not "Registering," build a list of all racers in the race  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>1st Place racer</td>";  echo "<td><select name = 'f5'>\n";  // Output each individual drop down entry  foreach ($racerArray as $key => $Choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == NULL)  $val = 0;  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'idRacer2':  // If current race status not "Registering," build a list of all racers in the race  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>2nd Place racer</td>";  echo "<td><select name = 'f6'>\n";  // Output each individual drop down entry  foreach ($racerArray as $key => $Choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == NULL)  $val = 0;  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'idRacer3':  // If current race status not "Registering," build a list of all racers in the race  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>3rd Place racer</td>";  echo "<td><select name = 'f7'>\n";  // Output each individual drop down entry  foreach ($racerArray as $key => $Choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == NULL)  $val = 0;  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'Den\_level':  // Be sure to have dtaabase value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Den level</td>";  echo "<td><select name = 'f8'>\n";  // Output each individual drop down entry  foreach ($denLevel as $key => $choice) {  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  break;  default:  Will also need a closing bracket after CASE construct  Locate these three lines:  <input type="button" value="insert" onclick="document.forms.ActionForm.action='?mode=i'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="delete" onclick="document.forms.ActionForm.action='?mode=d'; document.forms.ActionForm.submit()" />  .. and replace with these 5 lines  <input type="button" value="insert" onclick="document.forms.ActionForm.action='?mode=i'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="delete" onclick="document.forms.ActionForm.action='?mode=d'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="register" onclick="document.forms.ActionForm.action='?mode=r'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="enter scores" onclick="document.forms.ActionForm.action='?mode=e'; document.forms.ActionForm.submit()" /> &nbsp;  <input type="button" value="heat results" onclick="document.forms.ActionForm.action='?mode=h'; document.forms.ActionForm.submit()" />  Locate this line and make changes:  if (($mode != 's') && ($mode != 'i') && ($mode != 'u') && ($mode != 'd')) {  if (($mode != 's') && ($mode != 'i') && ($mode != 'u') && ($mode != 'd')&&( $mode != 'r')&&( $mode != 'e')) {  Locate this line:  switch ($mode) {  and add these lines right after it:  case 'r':  // fetch the record the user selected; assign to a session variable  $row = dbu\_fetch\_by\_key($fielddef, $scheme, $table, $dbconn, $\_POST, $keys);  $\_SESSION['raceToRegister'] = $row['f0'];  // Check to validate that the race has a heat\_header assigned; if not - error  if ($row['f4'] == 0) {  errorHandler2('Registration may not start until a heat is assigned to the race. Try again later!');  break;  }  // run the race registration script  echo '<meta http-equiv="refresh" content="0; URL=table\_race\_registration.php">';  case 'e':  // fetch the record the user selected; assign to a session variable  $row = dbu\_fetch\_by\_key($fielddef, $scheme, $table, $dbconn, $\_POST, $keys);  $\_SESSION['raceToEnterResults'] = $row['f0'];  // run the race scoring script  echo '<meta http-equiv="refresh" content="0; URL=table\_race\_heat.php">';  case 'h':  // fetch the record the user selected; assign to a session variable  $row = dbu\_fetch\_by\_key($fielddef, $scheme, $table, $dbconn, $\_POST, $keys);  $\_SESSION['raceToEnterResults'] = $row['f0'];  // run the heat results display script  echo '<meta http-equiv="refresh" content="0; URL=table\_race\_heat\_display.php">';  Locate these lines:  case 'u2':  $rslt = dbu\_handle\_update($fielddef, $scheme, $table, $dbconn, $\_POST, $keys);  if ($rslt) {  $show\_message = true;  $message = sprintf(MSG\_UPDATED, $start);  } else {  dbu\_handle\_error($dbconn->db\_lasterror());  }  Add this code afterward:  // If race is "starting" make sure we have race\_heat records ready to go  if ($HTTP\_POST\_VARS['f2'] == 'Started') {  // See if race\_heat records already exist  $raceHeatSql = "SELECT \* from race\_heat where idRace=".$HTTP\_POST\_VARS['RKEY'];  $raceHeatResult = $dbconn->db\_get\_all\_rows($raceHeatSql);  //$countRH = sizeof($raceHeatResult);  if (sizeof($raceHeatResult) == 0) {  // If records do not exist, get array of race heat template records  $heatSql = "SELECT \* from heat where Heat\_Header\_idHeat\_Header =".$HTTP\_POST\_VARS['f4']." order by Number";  $heatResult = $dbconn->db\_get\_all\_rows($heatSql);  // read through array of race heat template records - and punch out  // a race\_heat record for each  if (!$heatResult) {  echo "falure </br>";  } else {  $i=1;  foreach($heatResult as $key => $value) {  //echo "Car:".$value[Car\_Lane1];  // go get the record from the "race\_registration" file that has the same car#;  // we will use the racer id# we find here to write out in the new record  // Get racer id for car in lane #1  $rrSql = "SELECT rrRacer from race\_registration where (rrRace=".$HTTP\_POST\_VARS['RKEY'].") and (rrCarNumber=".$value[Car\_Lane1].")";  $rrResult = $dbconn->db\_get\_all\_rows($rrSql);  if (!rrResult) {  echo "failure </br>";  } else {  foreach($rrResult as $rrKey => $rrValue) {  $lane1Racer = $rrValue['rrRacer'];  }  }  // Get racer id for car in lane #2  $rrSql = "SELECT rrRacer from race\_registration where (rrRace=".$HTTP\_POST\_VARS['RKEY'].") and (rrCarNumber=".$value[Car\_Lane2].")";  $rrResult = $dbconn->db\_get\_all\_rows($rrSql);  if (!rrResult) {  echo "failure </br>";  } else {  foreach($rrResult as $rrKey => $rrValue) {  $lane2Racer = $rrValue['rrRacer'];  }  }  // Get racer id for car in lane #3  $rrSql = "SELECT rrRacer from race\_registration where (rrRace=".$HTTP\_POST\_VARS['RKEY'].") and (rrCarNumber=".$value[Car\_Lane3].")";  $rrResult = $dbconn->db\_get\_all\_rows($rrSql);  if (!rrResult) {  echo "failure </br>";  } else {  foreach($rrResult as $rrKey => $rrValue) {  $lane3Racer = $rrValue['rrRacer'];  }  }  // Get racer id for car in lane #4  $rrSql = "SELECT rrRacer from race\_registration where (rrRace=".$HTTP\_POST\_VARS['RKEY'].") and (rrCarNumber=".$value[Car\_Lane4].")";  $rrResult = $dbconn->db\_get\_all\_rows($rrSql);  if (!rrResult) {  echo "failure </br>";  } else {  foreach($rrResult as $rrKey => $rrValue) {  $lane4Racer = $rrValue['rrRacer'];  }  }  $raceHeatSql = "INSERT into race\_heat values(0,".$HTTP\_POST\_VARS['RKEY'].",".$i++.",".$lane1Racer.",".$lane2Racer.",".$lane3Racer.",".$lane4Racer.",NULL,NULL,NULL,NULL)";  $raceHeatInsert = $dbconn->db\_query($raceHeatSql);  echo($raceHeatInsert);  // $heatHeaderArray[$value[idHeat\_Header]] = $value[Description];  }  }  }  } |

|  |
| --- |
| Race Registration table:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Replace this line with the next:  $panel\_heading = ucfirst($table." Maintenance");  $panel\_heading = ucfirst(“Race Registration");  Locate this line:  $dbconn->db\_connect($config['dbhost'], $config['dblogin'], $config['dbpass'], $config['dbname'], $config['dbport']);  Then add these lines afterward:  // Get description for race identified by session variable $\_SESSION['raceToRegister']  $sql = "SELECT description FROM race where idRace = ".$\_SESSION['raceToRegister'];  $result = $dbconn->db\_get\_all\_rows($sql);  // should only be one record returned...  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  $raceDescription = $value['Description'];  }  }  echo "<h3>Registering race: <FONT COLOR=\"ff0000\">\*\*\* ".$raceDescription." \*\*\*</FONT></h3>";  After create, locate and change these lines to hide key field & race id field (foreign key):  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRace\_registration', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Race', FLD\_DISPLAY\_SZ => 7,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  Change these field sizes:  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer', FLD\_DISPLAY\_SZ => 27,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car number', FLD\_DISPLAY\_SZ => 17,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Points earned', FLD\_DISPLAY\_SZ => 17,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place in race', FLD\_DISPLAY\_SZ => 17,  locate and change this line to sort by car# and select by the race that is in the session variable ‘raceToRegister’  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, ‘rrCarNumber’, "rrRace= ".$\_SESSION['raceToRegister');  +3 other spots to replace this same line of code – find them!  Locate these two lines:  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then add in this logic:    switch ($fld[FLD\_DATABASE]) {  // If this is the race field - hide it; if no value, assgn the session value for the race  case 'rrRace':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Assign session race value if needed  if (($val == NULL) or ($val == 0))  $val = $\_SESSION['raceToRegister'];  // Punch out HTML  echo "<tr><td></td>";  echo "<td><input name = 'f1' type=hidden value=\" $val\" /></td></tr>";  break;  // If this is the "Racer" field, show a drop down list of racers in the den level that  // are eligible for tihs race  case 'rrRacer':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Racer registering</td>";  echo "<td><select name = 'f2'>\n";  // Get the Den\_level associated with the race  $sql = "SELECT Den\_level FROM race WHERE idRace =".$\_SESSION['raceToRegister'];  $rslt = $dbconn->db\_get\_one\_row($sql);  if (!$rslt) {  echo "falure </br>";  } else {  // Assemble a list of possible racers  // $sql = "SELECT \* from racer order by Last\_name";  $sql = "SELECT First\_name, Last\_name, idRacer FROM racer, den WHERE (IdDen = Den\_IdDen) and (Den\_level = '".$rslt['Den\_level']."') order by Last\_name, First\_name";  $result = $dbconn->db\_get\_all\_rows($sql);  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  // Is racer already registered? Only show drop down of racers not yet registered for this race  $sql = "SELECT idRace\_registration FROM race\_registration WHERE (rrRace=".$\_SESSION['raceToRegister'].") and (rrRacer=".$value[idRacer].")";  $rslt = $dbconn->db\_get\_one\_row($sql);  if (!$rslt) {  $racerArray[$value[idRacer]] = $value[First\_name]." ".$value[Last\_name];  }  }  }  }  // Output each individual drop down entry  foreach ($racerArray as $key => $Choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  default:  Will also need a closing bracket after CASE construct  Locate this chunk of code:  if ($fld[FLD\_INPUT]) {  echo "<tr><td>$fld[FLD\_DISPLAY]</td>";  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  Insert this afterwards:  // Initialize new field to simpliyf entry  if ($val == NULL)  $val = 0;  Find this chunk of code:  <?php // DATA  $checked = ' checked="checked"';  $i = 0;  foreach($rows as $row) {  $bk = $i++ % 2 ? "" : ' class="sublight"';  echo "<tr$bk><td><input type='radio'$checked name='".RKEY."' value='".htmlentities($row[RKEY], ENT\_QUOTES, $config['encoding'])."' /></td>";  foreach ($fielddef as $fkey=>$fld) {  if ($fld[FLD\_VISIBLE]) {  $value = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  … then insert after it:  // Show racer name - not racer foriegn key  if ($fld[FLD\_DATABASE] == 'rrRacer') {  // Get description for racer identified fforign key "rrRacer"  $racerSql = "SELECT First\_name, Last\_name FROM racer where idRacer =".$value;  $racerResult = $dbconn->db\_get\_all\_rows($racerSql);  // should only be one record returned...  if (!$racerResult) {  echo "falure </br>";  } else {  foreach($racerResult as $racerKey => $racerValue) {  $value = $racerValue['First\_name']." ".$racerValue['Last\_name'];  }  }  } |
|  |
|  |

Heat Header table

|  |
| --- |
| Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Add this line  require\_once "constants.inc.php";  Replace this line:  $panel\_heading = ucfirst($table." Maintenance");  $panel\_heading = ucfirst("Race Heat Template Maintenance");  After create, locate and change this line to hide fields:  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Key', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  locate and change this line to sort by Heat Header description  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, ‘Description’);  +3 other locations that needs to be replaced…  Locate these two lines:  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then add in this logic:  switch ($fld[FLD\_DATABASE]) {  // Show a dropdown box of possible # of tracks (from "constants" include file)  case 'Lanes':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td># of Lanes</td>";  echo "<td><select name = 'f4'>\n";  // Output each individual drop down entry  foreach ($trackSizes as $key => $choice){  echo "<option value='$key' ";  // If the entry is the same as the existing database value  // indicate this entry should show (selected)  // (used for existing database records)  if ($val == $key) {  echo " selected";  }  echo " >$choice</option>\n";  }  echo "<select></td><tr>";  break;  default:  Will also need a closing bracket after CASE construct  Locate this chunk of code:  if ($fld[FLD\_INPUT]) {  echo "<tr><td>$fld[FLD\_DISPLAY]</td>";  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  And add this in afterwards:  // If a new database field and no value assign, defualt to zero to simplify entry  if ($val == NULL) {  if ($fld['FLD\_INPUT\_VALIDATION'] == 'Numeric') {  $val = 0;  } else {  $val = ' ';  }  }  Locate these three lines:  <input type="button" value="insert" onclick="document.forms.ActionForm.action='?mode=i'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="delete" onclick="document.forms.ActionForm.action='?mode=d'; document.forms.ActionForm.submit()" />  .. and replace with these 4 lines  <input type="button" value="insert" onclick="document.forms.ActionForm.action='?mode=i'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="delete" onclick="document.forms.ActionForm.action='?mode=d'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="heats" onclick="document.forms.ActionForm.action='?mode=r'; document.forms.ActionForm.submit()" />  Locate this line and make changes:  if (($mode != 's') && ($mode != 'i') && ($mode != 'u') && ($mode != 'd')) {  if (($mode != 's') && ($mode != 'i') && ($mode != 'u') && ($mode != 'd')&&( $mode != 'r')) {  Locate this line:  switch ($mode) {  and add these lines right after it:  case 'r':  // fetch the record the user selected; assign to a session variable  $row = dbu\_fetch\_by\_key($fielddef, $scheme, $table, $dbconn, $\_POST, $keys);  $\_SESSION['raceToRegister'] = $row['f0'];  // run the race registration script  echo '<meta http-equiv="refresh" content="0; URL=table\_race\_registration.php">'; |

|  |
| --- |
| Table\_heat:  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Replace this line with the next:  $panel\_heading = ucfirst($table." Maintenance");  $panel\_heading = ucfirst(“Detail Heat Maintenance");  FLD\_DATABASE => 'Number'),  FLD\_DATABASE => 'Heat#'),  FLD\_DATABASE => 'Car\_Lane1'),  FLD\_DATABASE => 'Car in Lane 1'),  FLD\_DATABASE => 'Car\_Lane2'),  FLD\_DATABASE => 'Car in Lane 2'),  FLD\_DATABASE => 'Car\_Lane3'),  FLD\_DATABASE => 'Car in Lane 3'),  FLD\_DATABASE => 'Car\_Lane4'),  FLD\_DATABASE => 'Car in Lane 4'),  Locate this line:  $dbconn->db\_connect($config['dbhost'], $config['dblogin'], $config['dbpass'], $config['dbname'], $config['dbport']);    Then add these lines afterward:  // Get description for Heat Header identified by session variable $\_SESSION['heatHeaderToEdit']  $sql = "SELECT description FROM heat\_header where idHeat\_Header = ".$\_SESSION['heatHeaderToEdit'];  $result = $dbconn->db\_get\_all\_rows($sql);  // should only be one record returned...  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  $HHDescription = $value['Description'];  }  }  echo "<h3>Race Template: <FONT COLOR=\"ff0000\">\*\*\* ".$HHDescription." \*\*\*</FONT></h3>";  After create, locate and change these lines to hide key field & race id field (foreign key):  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idHeat', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Heat\_Header\_idHeat\_Header', FLD\_DISPLAY\_SZ => 50,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => ‘', FLD\_DISPLAY\_SZ => 50,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Number', FLD\_DISPLAY\_SZ => 7,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Heat#', FLD\_DISPLAY\_SZ => 20,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car\_Lane1', FLD\_DISPLAY\_SZ => 10,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car in Lane 1', FLD\_DISPLAY\_SZ => 20,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car\_Lane2', FLD\_DISPLAY\_SZ => 10,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car in Lane 2', FLD\_DISPLAY\_SZ => 20,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car\_Lane3', FLD\_DISPLAY\_SZ => 10,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car in Lane 3', FLD\_DISPLAY\_SZ => 20,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car\_Lane4', FLD\_DISPLAY\_SZ => 10,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Car in Lane 4', FLD\_DISPLAY\_SZ => 20,  locate and change this line to sort by heat# and select by the heat header that is in the session variable ‘heatHeaderToEdit’  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, 'Number', "Heat\_Header\_idHeat\_Header=".$\_SESSION['heatHeaderToEdit']);  +3 other spots to replace this same line of code – find them!  Locate these two lines:  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then add in this logic:  switch ($fld[FLD\_DATABASE]) {  // If this is the Heat Header id field - hide it; if no value, assgn the session value  case 'Heat\_Header\_idHeat\_Header':  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Assign session race value if needed  if (($val == NULL) or ($val == 0))  $val = $\_SESSION['heatHeaderToEdit'];  // Punch out HTML  echo "<tr><td></td>";  echo "<td><input name = 'f1' type=hidden value=\" $val\" /></td></tr>";  break;  default:  Will also need a closing bracket after CASE construct  Locate this chunk of code:  if ($fld[FLD\_INPUT]) {  echo "<tr><td>$fld[FLD\_DISPLAY]</td>";  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  Insert this afterwards:  // Initialize new field to simplify entry  if ($val == NULL)  $val = 0;  Find this chunk of code:  <?php // DATA  $checked = ' checked="checked"';  $i = 0;  foreach($rows as $row) {  $bk = $i++ % 2 ? "" : ' class="sublight"';  echo "<tr$bk><td><input type='radio'$checked name='".RKEY."' value='".htmlentities($row[RKEY], ENT\_QUOTES, $config['encoding'])."' /></td>";  foreach ($fielddef as $fkey=>$fld) {  if ($fld[FLD\_VISIBLE]) {  $value = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  … then insert after it:  // Show heat header (race template) name - not foriegn key  if ($fld[FLD\_DATABASE] == 'Heat\_Header\_idHeat\_Header') {  // Get description for heat header  $hhSql = "SELECT Description FROM heat\_header where idHeat\_Header =".$value;  $hhResult = $dbconn->db\_get\_all\_rows($hhSql);  // should only be one record returned...  if (!$hhResult) {  echo "falure </br>";  } else {  foreach($hhResult as $hhKey => $hhValue) {  $value = $hhValue['Description'];  }  }  }  Find this chunk of logic:  <?php } else if ($show\_data) { ?>  <form name="ActionForm" method="post" action="">  <table cellpadding="1" cellspacing="0" border="0" bgcolor="#ababab"><tr><td>  <table cellpadding="0" cellspacing="1" border="0" class="datatable">  Change the last line to:  <table cellpadding="0" cellspacing="1" border="0" class="datatable" align="center"> |

|  |
| --- |
| Table\_race\_heat: (capture race results)  Replace this line with next:  require\_once "config.inc.php";  require\_once $\_SERVER['COMPUTERNAME'] . ".config.inc.php";  Replace this line with the next:  $panel\_heading = ucfirst($table." Maintenance");  $panel\_heading = ucfirst(“Enter Race Results");  Locate this line:  $dbconn->db\_connect($config['dbhost'], $config['dblogin'], $config['dbpass'], $config['dbname'], $config['dbport']);  Then add these lines afterward:  // Get description for race identified by session variable $\_SESSION['raceToEnterResults']  $sql = "SELECT Description FROM race where idRace = ".$\_SESSION['raceToEnterResults'];  $result = $dbconn->db\_get\_all\_rows($sql);  // should only be one record returned...  if (!$result) {  echo "falure </br>";  } else {  foreach($result as $key => $value) {  $raceDescription = $value['Description'];  }  }  echo "<h3>Enter results for race: <FONT COLOR=\"ff0000\">\*\*\* ".$raceDescription." \*\*\*</FONT></h3>";  After create, locate and change these lines to hide key field & racer id fields (foreign key) and change names:  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRace\_Heat', FLD\_DISPLAY\_SZ => 7,  'f0' => array(FLD\_ID => true, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'idRace', FLD\_DISPLAY\_SZ => 7,  'f1' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Number', FLD\_DISPLAY\_SZ => 7,  'f2' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Heat #', FLD\_DISPLAY\_SZ => 7,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer\_lane1', FLD\_DISPLAY\_SZ => 7,  'f3' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer\_lane2', FLD\_DISPLAY\_SZ => 7,  'f4' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer\_lane3', FLD\_DISPLAY\_SZ => 7,  'f5' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Racer\_lane4', FLD\_DISPLAY\_SZ => 7,  'f6' => array(FLD\_ID => false, FLD\_VISIBLE => false, FLD\_DISPLAY => '', FLD\_DISPLAY\_SZ => 7,  'f7' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place\_1', FLD\_DISPLAY\_SZ => 20,  'f7' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place 1', FLD\_DISPLAY\_SZ => 20,  'f8' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place\_2', FLD\_DISPLAY\_SZ => 20,  'f8' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place 2', FLD\_DISPLAY\_SZ => 20,  'f9' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place\_3', FLD\_DISPLAY\_SZ => 20,  'f9' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place 3', FLD\_DISPLAY\_SZ => 20,  'f10' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place\_4', FLD\_DISPLAY\_SZ => 20,  'f10' => array(FLD\_ID => false, FLD\_VISIBLE => true, FLD\_DISPLAY => 'Place 4', FLD\_DISPLAY\_SZ => 20,  locate and change this line to sort by heat sequence # and select by the race that is in the session variable ‘raceToEnterResults’  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager);  $rows = dbu\_handle\_select($fielddef, $scheme, $table, $dbconn, $keys, $start, $config['rows\_per\_page'], $config['pager\_items'], $pager, 'Number', "idRace= ".$\_SESSION['raceToEnterResults']);  +3 other spots to replace this same line of code – find them!  Locate these two lines:  <?php // INPUT  foreach($fielddef as $fkey=>$fld) {  Then add in this logic after the <?php // INPUT:  // Assemble a list of racer names in this heat  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f3];  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane1RacerName = $value['First\_name']." ".$value['Last\_name'];  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f4];  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane2RacerName = $value['First\_name']." ".$value['Last\_name'];  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f5];  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane3RacerName = $value['First\_name']." ".$value['Last\_name'];  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f6];  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane4RacerName = $value['First\_name']." ".$value['Last\_name'];  // Assemble a list of racer car numbers in this heat  $sql = "SELECT rrCarNumber from race\_registration where (rrRacer=".$row[f3]." and rrRace=".$\_SESSION['raceToEnterResults'].")";  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane1CarNumber = $value['rrCarNumber'];  $sql = "SELECT rrCarNumber from race\_registration where (rrRacer=".$row[f4]." and rrRace=".$\_SESSION['raceToEnterResults'].")";  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane2CarNumber = $value['rrCarNumber'];  $sql = "SELECT rrCarNumber from race\_registration where (rrRacer=".$row[f5]." and rrRace=".$\_SESSION['raceToEnterResults'].")";  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane3CarNumber = $value['rrCarNumber'];  $sql = "SELECT rrCarNumber from race\_registration where (rrRacer=".$row[f6]." and rrRace=".$\_SESSION['raceToEnterResults'].")";  $result = $dbconn->db\_get\_all\_rows($sql);  foreach($result as $key => $value)  $lane4CarNumber = $value['rrCarNumber'];  // Create an array of racers for drop down boxes  $racerChoice[0] = "";  $racerChoice[$row[f3]] = "Car# ".$lane1CarNumber." - ".$lane1RacerName;  $racerChoice[$row[f4]] = "Car# ".$lane2CarNumber." - ".$lane2RacerName;  $racerChoice[$row[f5]] = "Car# ".$lane3CarNumber." - ".$lane3RacerName;  $racerChoice[$row[f6]] = "Car# ".$lane4CarNumber." - ".$lane4RacerName;  Add this after the line: foreach($fielddef as $fkey=>$fld) {  switch ($fld[FLD\_DATABASE]) {  case 'idRace':  // If this is the race field - hide it; if no value, assign the session value for the race  // Be sure to have database value ready  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Assign session race value if needed  if (($val == NULL) or ($val == 0))  $val = $\_SESSION['raceToEnterResults'];  // Punch out HTML  echo "<tr><td></td>";  echo "<td><input name = 'f1' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Number':  // Show heat sequence number (output only)  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  echo "<tr><td>Race heat number:</td>";  echo "<td>$val</td><td><input name = 'f2' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Racer\_lane1':  // Show racer name for lane 1 (output only)  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  echo "<tr><td>Racer - Lane #1:</td>";  echo "<td>$lane1RacerName</td><td>Car# $lane1CarNumber</td><td><input name = 'f3' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Racer\_lane2':  // Show racer name for lane 2 (output only)  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  echo "<tr><td>Racer - Lane #2:</td>";  echo "<td>$lane2RacerName</td><td>Car# $lane2CarNumber</td><td><input name = 'f4' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Racer\_lane3':  // Show racer name for lane 3 (output only)  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  echo "<tr><td>Racer - Lane #3:</td>";  echo "<td>$lane3RacerName</td><td>Car# $lane3CarNumber</td><td><input name = 'f5' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Racer\_lane4':  // Show racer name for lane 4 (output only)  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  echo "<tr><td>Racer - Lane #4:</td>";  echo "<td>$lane4RacerName</td><td>Car# $lane4CarNumber</td><td><input name = 'f6' type=hidden value=\" $val\" /></td></tr>";  break;  case 'Place\_1':  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>First place racer:</td><td><select name = 'f7'>\n";  // Output each individual drop down entry using array built above for choices  foreach ($racerChoice as $key => $Choice){  echo "<option value='$key' ";  // indicate this entry should show (selected) if same as database value  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'Place\_2':  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Second place racer:</td><td><select name = 'f8'>\n";  // Output each individual drop down entry using array built above for choices  foreach ($racerChoice as $key => $Choice){  echo "<option value='$key' ";  // indicate this entry should show (selected) if same as database value  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'Place\_3':  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Third place racer:</td><td><select name = 'f9'>\n";  // Output each individual drop down entry using array built above for choices  foreach ($racerChoice as $key => $Choice){  echo "<option value='$key' ";  // indicate this entry should show (selected) if same as database value  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  case 'Place\_4':  $val = htmlentities($row[$fkey], ENT\_QUOTES, $config['encoding']);  // Punch out start of HTML  echo "<tr><td>Fourth place racer:</td><td><select name = 'f10'>\n";  // Output each individual drop down entry using array built above for choices  foreach ($racerChoice as $key => $Choice){  echo "<option value='$key' ";  // indicate this entry should show (selected) if same as database value  if ($val == $key) {  echo " selected";  }  echo " >$Choice</option>\n";  }  echo "<select></td><tr>";  break;  default:  Will also need a closing bracket after CASE construct  Locate this chunk of code:  <?php // DATA  $checked = ' checked="checked"';  $i = 0;  foreach($rows as $row) {  $bk = $i++ % 2 ? "" : ' class="sublight"';  echo "<tr$bk><td><input type='radio'$checked name='".RKEY."' value='".htmlentities($row[RKEY], ENT\_QUOTES, $config['encoding'])."' /></td>";  foreach ($fielddef as $fkey=>$fld) {  And add this afterwards:  switch ($fld[FLD\_DATABASE]) {  case 'Place\_1';  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f7];  $result = $dbconn->db\_get\_all\_rows($sql);  if ($result) {  foreach($result as $key => $value)  $lane1RacerName = $value['First\_name']." ".$value['Last\_name'];  } else {  $lane1RacerName = "";  }  echo "<td>$lane1RacerName</td>";  break;  case 'Place\_2';  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f8];  $result = $dbconn->db\_get\_all\_rows($sql);  if ($result) {  foreach($result as $key => $value)  $lane2RacerName = $value['First\_name']." ".$value['Last\_name'];  } else {  $lane2RacerName = "";  }  echo "<td>$lane2RacerName</td>";  break;  case 'Place\_3';  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f9];  $result = $dbconn->db\_get\_all\_rows($sql);  if ($result) {  foreach($result as $key => $value)  $lane3RacerName = $value['First\_name']." ".$value['Last\_name'];  } else {  $lane3RacerName = "";  }  echo "<td>$lane3RacerName</td>";  break;  case 'Place\_4';  $sql = "SELECT Last\_name, First\_name from racer where idRacer=".$row[f10];  $result = $dbconn->db\_get\_all\_rows($sql);  if ($result) {  foreach($result as $key => $value)  $lane4RacerName = $value['First\_name']." ".$value['Last\_name'];  } else {  $lane4RacerName = "";  }  echo "<td>$lane4RacerName</td>";  break;  default:  will also require a closing bracket after the “if” construct that follows  Now locate these three lines:  <input type="button" value="insert" onclick="document.forms.ActionForm.action='?mode=i'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" />&nbsp;  <input type="button" value="delete" onclick="document.forms.ActionForm.action='?mode=d'; document.forms.ActionForm.submit()" />  Replace them with this one line (drop the insert and delete options)  <input type="button" value="update" onclick="document.forms.ActionForm.action='?mode=u'; document.forms.ActionForm.submit()" /> |