Teaching Office Database Developers Guide

Table of Contents

caret.

JISC

1 Useful Tools for Developing TODB

Please refer to the Installation Guide for help on your Apache/MySQL/PHP installation.

1.1 Editing PHP

Beside using a text editor, you may find HTML-Kit http://www.htmlkit.com/ useful. If you are familiar with the Eclipse IDE you may wish to use a version of this configured for PHP http://www.eclipse.org/pdt/ .

Check the Apache error log file for help pin-pointing PHP errors. You should take care not to use an editor such as Notepad on Windows that might introduce ^Ms at the end of lines. (See Troubleshooting section in Installation Guide for more help on this).

1.2 Working with MySQL

MySQL Query Browser provides a useful interface for working with the database. This seems to have been replaced by the SQL Development feature of MySQL workbench 5.2 http://dev.mysql.com/downloads/workbench/5.2.html .

Do not use Notepad (Wordpad is OK) for editing scripts as it adds spurious escape characters.

1.3 Other Web Development Tools

1.3.1 Linux commands for Windows

You may wish to download and install 'UnxUtils' (http://unxutils.sourceforge.net/) which provides many useful Unix/Linux commands for use on Windows. You will need to add to your system path; search for "Unxutils Installing" for more help.

1.3.2 Firebug and Web Developer

Firebug https://addons.mozilla.org/en-US/firefox/addon/1843 is an add on for Firefox web-browser that let's you view and edit CSS, HTML and Javascript in a web page. Web Developer https://addons.mozilla.org/en-US/firefox/addon/60 compliments the use of Firebug with an extension menu and toolbar. These are of particular use should you need to debug the pop-up box used when editing.

1.4 Browsers

You should make sure that changes to TODB, especially involving the pop-up box, are tested with the different browsers that your users may be using. Please refer to the Troubleshooting Guide in the Installation Document for help with some potential issues.

2 Pages in the TODB

2.1 Introduction

Each page on the TODB effectively represents an application or operation in theTODB. Each application or operation could be:

- A Report (e.g. view_faculty_summary.php)
- A facility for viewing and editing a table (e.g. view_people.php, view_jobs.php, view_units.php)
- A facility for performing an administrative operation (e.g. view_new_year.php)
- A combination of these (e.g. view_ujobs.php, a Cambridge University Engineering Department-specific report that supports editing of records).

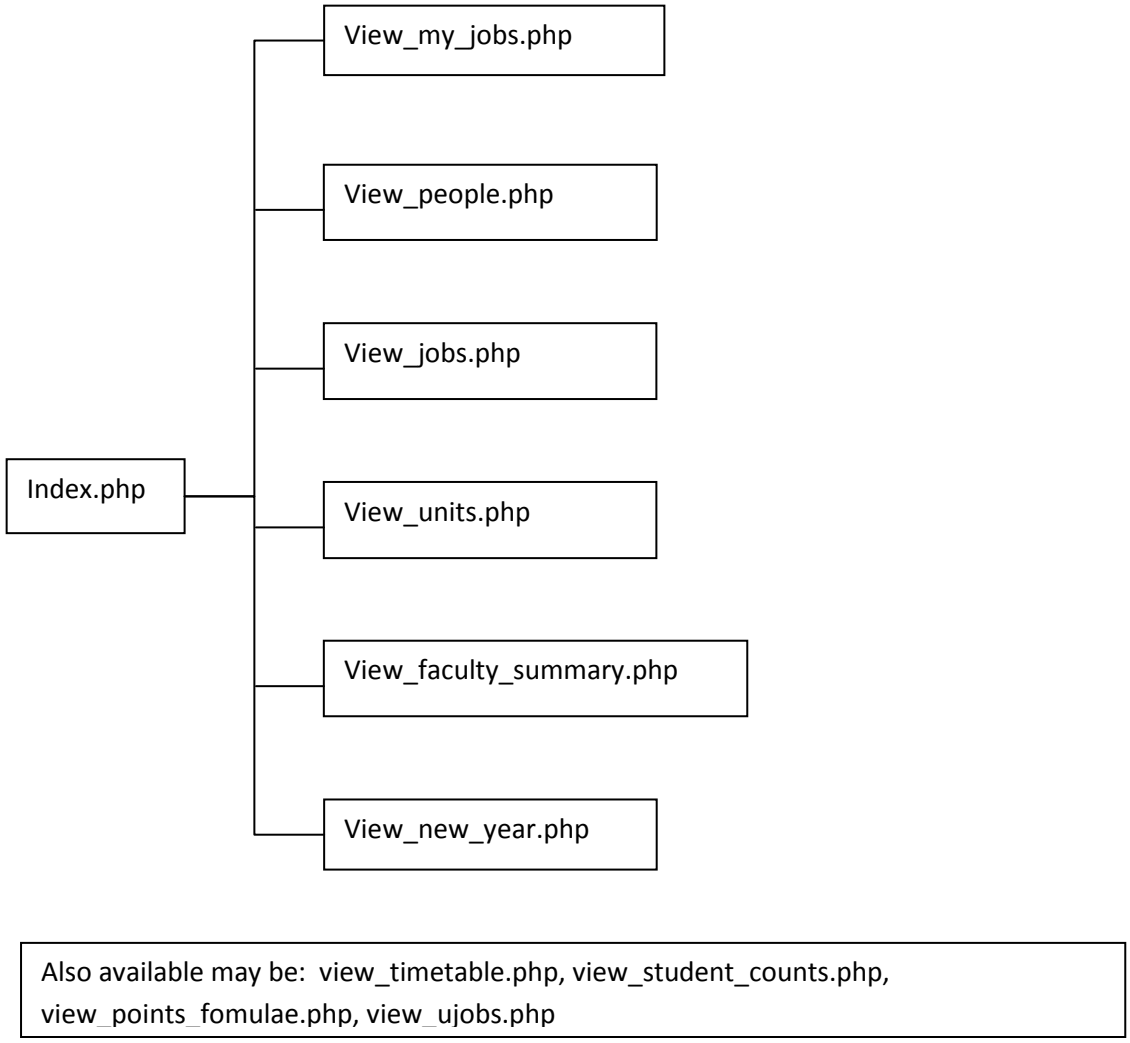

Figure 2-1 Hierarchy of TODB pages

2.2 Page Conventions

2.2.1 Page Language

The pages are written in PHP. Formatting is via CSS. The general principle is keep the code and operation of the system as simple as possible.

2.2.2 Database

Data storage is provided by a MySQL database; connection details are clearly defined in `dbconnect.inc` and `config/db.inc`. The variables $dbread and $dbwrite provide read and write access to the database. `$db_metadata` provides access to the MySQL `information_schema` for accessing table properties. These variables become available if `config/db.inc` is included in the page.

2.2.3 Table Locking

A slightly crude locking system has been developed. This uses a table Editlocks_<year> which holds locks for Jobs, People and Units for each year. The user id is inserted in the lock table and a lock flag set when someone successfully selects edit mode. If the lock is not available (someone else is updating this data) they are told and a lock is not taken out. There is an option to override an existing lock when the lock flag will be unset, the user then needs to click on the 'Edit' button again to take out their own lock. The lock flag is unset when the user clicks on 'Finish Edit'.

2.2.4 Admin, Flux and Ordinary users

Admin users are defined in config/config.inc and are allowed to edit data. Flux users can view but not change data; they can view data even if it has been marked as 'in flux'. Ordinary users can only view data, and only that not marked as in flux. Apache must be configured to allow users access to the system.

2.2.5 Edit Mode

It is understood, in general, that in order to change data, it is necessary to click on the Edit button provided in common2.inc to go into 'Edit Mode'.

2.2.6 CSV Mode

This is used to specify whether output should be to an on screen table ($csvmode = 0) or as a Comma Separated Variable list that can be read by Excel ($csvmode = 2). As far as this developer can tell, $csvmode = 1 is an old setting that is no longer used.

2.3 Creating a New TODB Page

A lot of code has already been written for facilitating easier, faster and more consistent development of new TODB features.

2.3.1 Basic Page Template

The following constitutes a basic page template and is explained more fully in the next section:

```
<?php

require('config/config.inc');
require('config/db.inc');
require('useful.inc');
require('config/years.inc');
require('auth.inc');
require('config/jobs.inc');
require('config/people.inc');
require('locks.inc');

require('common1.inc');
require('config/header.inc');
require('config/top.inc');

$tablename = '?????';
require('common2.inc');

// content goes here

require('config/footer.inc');

?>
```

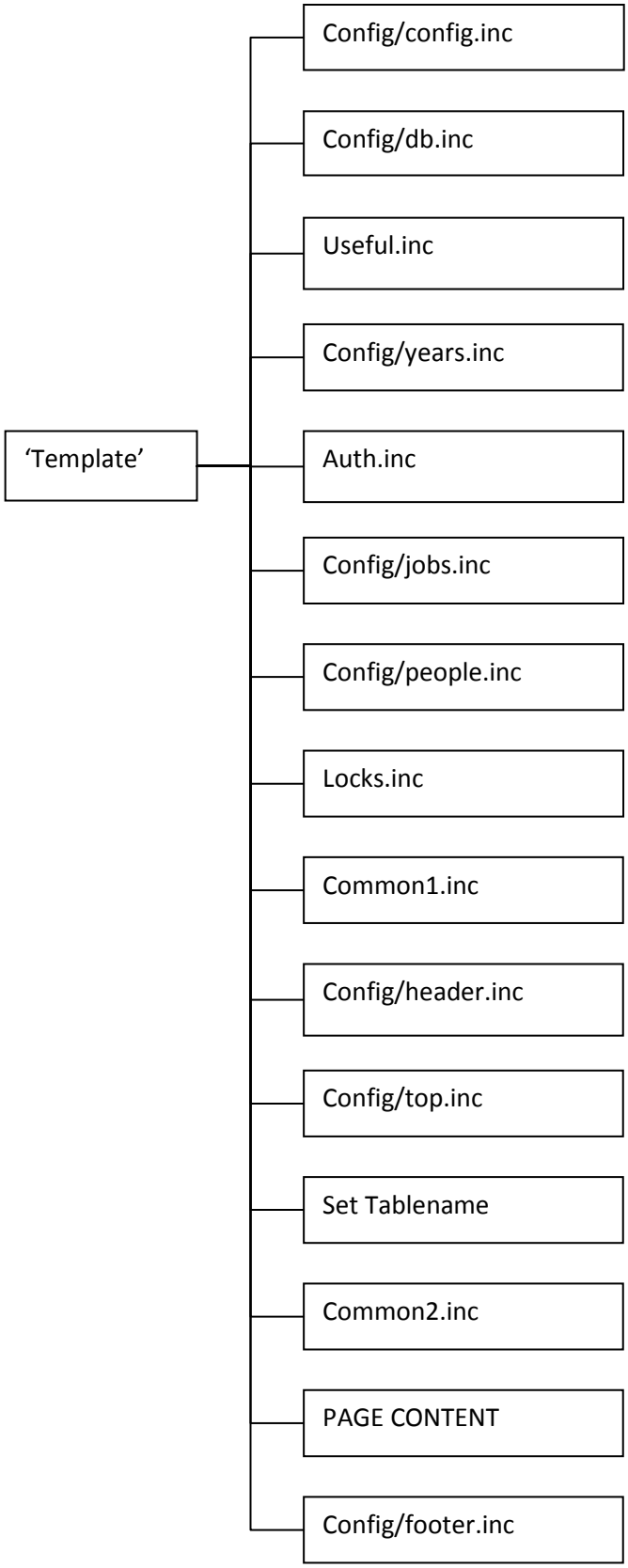

Figure 2-2 TODB 'Template' code

2.3.2 Outline Functionality

File	Purpose
Config/config.inc	Setups the following : Admin, Flux and Demo user arrays Email Address Text fields for Screen Divisions array Jobs Filtering arrays CSV directory locations Points calculator settings Applications accessable by Generic Operations Location of any generated list of all allowed users
Config/db.inc	Sets up the database connection information (db username and password).
Useful.inc	Various Functions: see 6.1.1
config/years.inc	Get years information from db.
auth.inc	Authenticate user and see if they are 'special'
config/jobs.inc	Controls columns and display order for jobs
config/people.inc	Controls columns for people
locks.inc	Locking functions: getLock () releaseLock() checkLock()
common1.inc	Go HOME if required Find out which year we are dealing with Check demo and flux users are following rules See if we are Editing and set/unset locks appropriately *CLIENT: Successful clicking of the Edit button uses Javascript to put table into selection mode and generate pop-up box for editing when a line is selected.* If Add/Update: UPDATE database if Id exists (unless duplicate required) ADD if new Id or duplicate If PurgeDeleted: DELETE those marked as deleted
config/header.inc	<START OF THE OUTPUT PROCESS> CSS Style settings
config/top.inc	Header banner for the page
Set $tablename	People/Faculty_summary etc - controls which table type is displayed
common2.inc	Generate the line with the HOME button with appropriate Edit button

Content	Generate any filtering buttons Handle POSTs appropriately – e.g. Filter, CSV button Form the SQL Query depending on criteria from POSTs Display the Table: Do the SQL Query <table>_Table.inc Display the results using ShowGeneralTable()
config/footer.inc	Footer banner for the page <END OF THE OUTPUT PROCESS>

3 Valid Years

Data in the TODB is organised by year; separate tables are held for earch year. Available years on the system and their associated information are provided by the valid_years array. This is made available by including years.inc. This was previously 'hard-coded'. It has since been replaced with softer system which reads this information from the `config_years` table.

Valid years are created, edited and removed using the '`Admin: modify years configurations`' link on the main page of the standard TODB site. This link is only visible to admin users. The link refers to the PHP page 'view_new_year.php' which was originally used to create new years of data only. It now also allows the user to remove years, set flux years and current year.

id	yearval	description	notinflux	yearorder
10	2009_10	current_year	1	
11	2008_09	Oct 2008 - 2009	1	
13	2009_10	Oct 2009-2010	1	
15	2006_07	Oct 2006-2007	1	
16	2007_08	Oct 2007-2008	1	
17	2010_11	Oct 2010 - 2011	0	

Figure 3-1 Example Config Years data

You will see from the above example that there is a double entry for the current year (2009-10) and (2010-11) is in flux.

The 'yearval' column is used to name the tables and identify year data to the system, (e.g. '2009_10'). The longer 'description' (e.g. 'Oct 2009 to 2010) is used to more verbosely inform the user which year is under examination – e.g. in page headers.

4 Changing the Types displayed in the Faculty Summary View

Each job can have an activity type associated with it. This might be lecturing, examining, some form of administration (e.g. sitting on a committee), taking leave/sabbatical, conducting classes and seminars, and so on. These are also represented with letter codes, e.g. 'L', 'E', 'A', etc.

Types can be amended through the configuration file following a process set out in the Installation and Configuration Guide. Such changes are then automatically reflected in the view jobs facility which will add a new type to the list of type filters. To reflect a new type in the Faculty Summary view, small changes are needed in the PHP script for this view. Although the query is quite complex the changes needed are relatively simple provided you follow the symmetry.

The example below illustrates adding a column for a new Outreach type.

1. Edit view_faculty_summary.php,. Look for the following comment:
   ```
   // COLUMNS TO DISPLAY
   ```

2. Immediately below this comment, add the new column(s) to the array which holds the tooltips for the display.

```
// Build an array to store the tooltips for each heading
$titles = array(  'Unique name of person',
                  'Hours spent lecturing',
                  'Hours spent in classes',
                  'Hours spent in practicals',
                  'Hours spent in web-based teaching',
                  'TOTAL hours spent on Teaching (Lecturing, Classes,
Projects, Web-based)',
                  'Sessions spent lecturing',
                  'Sessions spent in classes',
                  'Sessions spent in practicals',
                  'Sessions spent in web-based teaching',
                  'TOTAL sessions spent on Teaching (Lecturing, Classes,
Projects, Web-based)',
                  'Count of Projects jobs',
                  'Count of Examining jobs',
                  'Count of Safety jobs',
                  'Count of Administration jobs',
                  'Count of Outreach jobs'
                  );
```

3. Now add the new column(s) to the array which holds the column names for the display.

```
// This is the list of columns to select, as well as what will be used in
the order by section:
$select_order = array(  'Faculty'         ,
                        '`Lectures (hrs)`',
                        '`Classes (hrs)`'    ,
                        '`Practicals (hrs)`',
                        '`Web Teach. (hrs)`',
                        '`TOTAL Teaching (hrs)`'     ,
                        '`Lectures (sessions)`',
                        '`Classes (sessions)`'    ,
                        '`Practicals (sessions)`',
```

```
                        ``Web Teach. (sessions)``,
                        ``TOTAL Teaching (sessions)``     ,
                        ``Projects (count)`` ,
                        ``Examining (count)`` ,
                        ``Safety (count)``  ,
                        ``Admin (count)``,
                        ``Outreach (count)``
                        );
```

4. Now add what these fields should contain which comes from a subquery (see 6).

```
// what to select?
$what_to_select = array('p.uname',
                        'ifnull(round(subqueryL.hours, 1), 0)',
                        'ifnull(round(subqueryC.hours, 1), 0)',
                        'ifnull(round(subqueryP.hours, 1), 0)',
                        'ifnull(round(subqueryW.hours, 1), 0)',
                        'ifnull(round(totals.hours, 1), 0)',
                        'ifnull(round(subqueryL.sessions, 0), 0)',
                        'ifnull(round(subqueryC.sessions, 0), 0)',
                        'ifnull(round(subqueryP.sessions, 0), 0)',
                        'ifnull(round(subqueryW.sessions, 0), 0)',
                        'ifnull(round(totals.sessions, 0), 0)',
                        'ifnull(round(subqueryR.numjobs, 0), 0)',
                        'ifnull(round(subqueryE.numjobs, 0), 0)',
                        'ifnull(round(subqueryS.numjobs, 0), 0)',
                        'ifnull(round(subqueryA.numjobs, 0), 0)',
                        'ifnull(round(subqueryO.numjobs, 0), 0)'
                        );
```

Note: The round() function controls how many decimal places the value is rounded to and the ifnull() will display 0 if the value is empty.

5. Finally add the columns to the FROM clause:

```
// build the FROM clause

// grouping job type as follows:
// L - Lectures
// C - JC, PBL, Discuss
// P - Projects
// S - Safety
// E - Examining
// A - Administration
// O - Outreach
// W - Web-based Teaching

$all_from = " people_$yearval as p ". /*inner joins follow "*/
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval where type in ('L','C','P','W') and
deleted = false group by uname) as totals on p.uname = totals.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='L' and deleted = false
group by uname) as subqueryL on totals.uname = subqueryL.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='P' and deleted = false
group by uname) as subqueryP on totals.uname = subqueryP.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='C' and deleted = false
group by uname) as subqueryC on totals.uname = subqueryC.uname  ".
```

```
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='R' and deleted = false
group by uname) as subqueryR on totals.uname = subqueryR.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='S' and deleted = false
group by uname) as subqueryS on totals.uname = subqueryS.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='E' and deleted = false
group by uname) as subqueryE on totals.uname = subqueryE.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='A' and deleted = false
group by uname) as subqueryA on totals.uname = subqueryA.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='O' and deleted = false
group by uname) as subqueryO on totals.uname = subqueryO.uname  ".
" left join (select uname, sum(hours) as hours, sum(sessions) as sessions,
count(*) as numjobs from jobs_$yearval  where type='W' and deleted = false
group by uname) as subqueryW on totals.uname = subqueryW.uname  ";
```

6. It is a good idea to double check all the line endings at this point (commas and semicolons) !

7. If there is a total column, you may (depending on requirements) need to add the new type to this column. This example shows type 'W' being added into the total column.

```
$all_from = " people_$yearval as p ". /*inner joins follow "*/
   " left join (select uname, sum(hours) as hours, sum(sessions) as
   sessions, count(*) as numjobs from jobs_$yearval where type in
   ('L','C','P','W')...
```

Accompanying change for tooltips arrary:

```
$titles = array(  'Unique name of person',
                  'Hours spent lecturing',
                  'Hours spent in classes',
                  'Hours spent in practicals',
                  'Hours spent in web-based teaching',
                  'TOTAL hours spent on Teaching (Lecturing, Classes,
Projects, Web-based)',...
```

8. Now save the file and test out the changes.

9. If things don't work:
   - Check all the line endings again
   - Check your spellings

Note on the SQL Query :
These segments of the query are used to form the overall query which is of the form:
SELECT p.uname, subqueryL.hours .... subqueryL.sessions ..
FROM people_<year> as p,
LEFT JOIN (SELECT uname, sum(hours) as hours, sum(sessions) as sessions from jobs_<year>
WHERE type = 'L' GROUP by uname)
AS subqueryL on p.uname = subqueryL.uname
LEFT JOIN …

The Left Joins are used to form the appropriate totals grouped for each type that are needed for each line of the result set.

5 Miscellaneous Code Changes

The following possible improvements may need some changes to implement. These are detailed below:

5.1 Saturday Lectures

This is only applicable to departments that use the Timetabling feature.

The assumed setting is for Monday to Friday Lectures which makes sure there are no entries for Saturdays on the Timetable. Saturday lectures can be added with a two line code change:

1. Open `timetable_functions.inc` in a suitable editor.
2. Search for the following array:

```
// days of week on which lectures are given
$days_of_week = array('M', 'Tu', 'W', 'Th', 'F');
```

3. Add 'S' for Saturday:

```
$days_of_week = array('M', 'Tu', 'W', 'Th', 'F', 'S');
```

4. Search for the following code block, which links possible variant entries with the standard short day abbreviation:

```
function ReFormatTimeslot($input)
{
// ::::::::::::::::::::::::::::::::::::::::::::::::::::::
// day map lists
$day_map = array(
      'M' => array('M', 'MON', 'MDY', 'MDAY', 'MONDAY', 'MODAY'),
      'Tu' => array('T', 'TU', 'TUE', 'TUES', 'TUESDAY'),
      'W' => array('W', 'WED', 'WDN', 'WE', 'WEDS', 'WEDNESDAY', 'WDY'),
      'Th' => array('TH', 'THS', 'THURS', 'THDAY', 'THURSDAY', 'THRUSDAY'),
      'F' => array('F', 'FRI', 'FDAY', 'FRIDAY'),
      'S' => array('S', 'SAT', 'SDAY', 'SATURDAY'),
        );
```

5. Make sure it includes an entry for 'S'.

5.2 Correct Groups

This only applies where you are using both subject Groups and course Units together with Units which may contain jobs which have different groups.

[Note: Engineering was the only department during the lifetime of this project that this applied to. No change will be needed in the Generic version of TODB.]

In this case the 'Subj Group' column on View Units would be entered as a list of groups used for jobs in this unit. The column entitled 'Correct Groups' is calculated to include any groups for jobs in this course unit which are not listed in the 'Subj Group' column.

e.g. There is a Unit ESD with Subject Groups 'A' and 'D' (Subj Group entered as 'AD') . Clicking on ESD to see all the constituent jobs also shows jobs of type 'E'. The 'Correct Groups' column would display 'E'.

To make this work in other departments you will need to make changes to `view_units.php`, `table_functions.inc` and `config/units.inc` as shown below:

1. Open `view_units.php` in a suitable editor

2. Amend the Where clauses to ignore deleted jobs

```
if ($_POST['update'] == 'All') {
  $unit_query_where = " WHERE ((units_".$yearval.".deleted = FALSE) &&
(jobs_".$yearval.".deleted = FALSE))";
  $querydesc = 'All';
}
elseif ($_POST['update'] == 'Deleted') {
  $unit_query_where = " WHERE ((units_".$yearval.".deleted = TRUE) &&
(jobs_".$yearval.".deleted = FALSE))";
  $querydesc = 'Deleted';
}
/* MJ, Dec 08.  Was:
elseif (strpbrk($_POST['update'], 'ABCDEFGM')) {
   $unit_query_where = " WHERE ((units_".$yearval.".deleted  = FALSE) &&
instr(sgrps, '".
      $_POST['update'].
            "'))";
now:
*/
elseif (strpbrk($_POST['update'], $allgroups)) {
   $unit_query_where = " WHERE ((units_".$yearval.".deleted  = FALSE) &&
(jobs_".$yearval.".deleted = FALSE) && instr(sgrps, '".
      $_POST['update'].
            "'))";
      $querydesc = $querydesc = "for subject group ".$_POST['update'];

}
else {
  $unit_query_where = " WHERE ((units_".$yearval.".deleted = FALSE) &&
(jobs_".$yearval.".deleted = FALSE))";
}
```

3. Replace the unit query as shown:

```
/*  Each Unit has associated Group(s)/Division(s)
 *  The Unit may have jobs associated with it that are associated with
Groups that are not already linked to the unit
 *  We want to find these and show them in another column - othergroups
 *  The GROUP_CONCAT is making a list (with no separator - \ are Escape
chars) of any prgroups for the associated jobs
 *  that are not already in sgrps.  The LEFT join uses straight equality as
an earlier use of INSTR was picking up 5R13, 5R14 along with 5R1.
 */

    $unit_query = "SELECT units_".$yearval.".*,
                        (GROUP_CONCAT(DISTINCT (if ((LOCATE(prgroup, sgrps)
= 0), prgroup, \"\")) SEPARATOR \"\")) AS othergroups
                FROM units_".$yearval."
                LEFT JOIN jobs_".$yearval."
```

```
                ON (jobs_".$yearval.".paper = units_".$yearval.".uname)"
              .$unit_query_where."
              GROUP BY id
              ORDER BY ordering";
```

4. Open `table_functions.inc` in a suitable editor:

5. Search for `$column_comments` array and add the following below to set tooltips for derived columns:

```
$column_comments = array();
    $tt_query = "select column_name, column_comment from columns where
table_name = '".$tablename."_".$yearval."' and table_schema =
'$database_name';";
    $tt_res = mysql_query($tt_query, $db_metadata);
    if (!$tt_res) error_log("Error getting column comments: table:
$tablename; year: $yearval; schema: $database_name;Query is:# $tt_query
#".mysql_error($db_metadata));
    else
    {
        while ($row = mysql_fetch_array($tt_res))
        {
            $column_comments[$row[0]] = $row[1];
        }
        mysql_free_result($tt_res);

        // Add in comments for columns derived not stored AEC 25 03 10
        $column_comments[othergroups] = "Subject Groups relating to jobs in
this Unit that are not included in the Subj Group entry";
        $column_comments[sum] = "Total points from jobs allocated to this
person";
       }
```

Note: this also provides a tooltip for the Points column on View People

6. Open `config/units.inc`

7. Rename 'ignorewronggroups' to 'othergroups' for the administrator view, remove this field from pop-up box and adjust pop-up box widths to match:

```
$unititems = array ("id", "uname", "course", "ordering", "sgrps", "name",
"assessmode", "running", "global", "note", "updatetime");

// $adminunitcols is a list of all the columns that should be shown to an
// administrative user in edit mode

$adminunitcols = array ("uname", "course", "ordering", "sgrps",
"othergroups", "name", "assessmode", "running", "global", "note",
"updatetime");

$updateableadminunitcols = array ("uname", "course", "ordering", "sgrps",
"name", "assessmode", "running", "global", "note");

// $adminunitcolshdr are the strings for the header line in edit mode

$adminunitcolshdr = array ("Paper/Unit", "Course", "Order", "Subj Grps",
"Other Groups", "Full Name", "Mode", "Running", "Global Interest", "Note",
"Last updated");
```

```
$updateableadminunitcolshdr  = array ("Paper/Unit", "Course", "Order",
"Subj Grps", "Full Name", "Mode", "Running", "Global Interest", "Note");

// $adminunitcolwidths are the widths for the fields in the unit editing
form

$adminunitcolwidths = array (6, 5, 5, 8, 40, 8, 1, 1, 30);
```

This provides a more meaningful column header.

6 List of Code in TODB

The greyed out items are things that seem to be non-essential.

Item	Used By	Comment
Calculator-1-16x16.png	Addjob.php Job_popup.php	Calculator image used when adding points in pop-up
Calculator-1-32x32.png	Addjob.php Job_popup.php	Calculator image used when adding points in pop-up
ViewGeneric.php	Index.php	Form to display Generic Operations on index page
ac_files directory		Library javascript functions for autocomplete (ac) used in jobs popup for name autocompletion
addjob.php	View_jobs_java.inc	Run to gather and submit job data
addperson.php	View_people_java.inc	Run to gather and submit people data
addunit.php	View_units_java.inc	Run to gather and submit unit data
auth.inc	Part of 'template'	Authenticate user and see if they are 'special'
build_jobs_select.php	View_jobs.php	Run to do Special Filter stuff
checkbox_all_js.inc		NOT USED

common1.inc	Part of 'template'	Performs any actions from previous POST: Go HOME if required Find out which year we are dealing with Check demo and flux users are following rules See if we are Editing and set/unset locks appropriately *CLIENT: Successful clicking of the Edit button uses Javascript to put table into selection mode and generate pop-up box for editing when a line is selected.* If Add/Update: UPDATE database if Id exists (unless duplicate required) ADD if new Id or duplicate If PurgeDeleted: DELETE those marked as deleted
common2.inc	Part of 'template'	Generate the line with the HOME button with appropriate Edit button
config/ config.inc db.inc db_tmp.inc cbnew.inc footer.inc header.inc index.inc jobs.inc new_years.inc people.inc top.inc units.inc years.inc	Lots	Configuration files Main one Database stuff. The installation script dbsetup.php will create its own version of this file and overwrite the original Copy of db.inc – for backup purposes Not used ? Page footer CSS Styles Grid of links Settings for jobs Controls data copied to a new year Settings for people Page header Settings for units Accesses years
config_eng.inc	REDUNDANT	
csv_jobs.php	view_jobs_jbs.php view_quotas.php	No longer used
csv_jobs_new.inc	view_jobs_jbs.php	No longer used
csv_people.php	Was used by view_ people	No longer used
csvhdr.inc	csv_jobs_new.inc csv_jobs.php csv_people.php	No longer used
csvhdr_redir.inc	Was in job_table.inc	No longer used

dbconnect.inc	Db.inc	Functions to access the database
dbsetup.php	Installation process	Creates the database and tables See Installation Guide
images	DIRECTORY for images	e.g. University Banner
index.php	VITAL	Drives the 'front page'
job_popup.php	view_jobs_java.inc	Used to render the contents of the job pop-up box
job_table.inc	Lots	Used pretty much everywhere for jobs as 'glue' before using generic ShowGeneralTable function.
js_PointsCalc.php	addjob.php job_popup.php	Used to run the Points Calculator form.
locks.inc	Lots	Locking functions
notauthd.php	Common1.inc	Only now appears to be used for demo_users. Tells demo user they can only view Jobs
notyet.php	Common1.inc	Tells user they can't see data as it is 'in flux'
other_duties.php	View_People Optional	Used to lookup duties in other department
people_table.inc	View_people View_quotas	Used for people as 'glue' before using generic ShowGeneralTable function.
person_popup.php	View_people_java	Used to render the contents of the people pop-up box
points_breakdown.inc	View_people.php	Displays Points Breakdown after table
popup_form_state.inc	<table>_popup.php Add<table>.php	Its purpose is to hoover up the state of the main form within the page that called the popup, so that the popup can stash it in its own form, entirely as hidden inputs, for resubmission to the main page
requirements.php	DEVELOPMENT TOOL	1. Read a series of files, looking for all required includes, and displays the heirarchy of includes. 2. It can also recursively search these for a particular included file, in order to show which files include it.
resize_popup.inc	<table>_popup.php	Javascript to resize pop-up box
scroll_java.inc	<table>_popup.php	Browser tailored functions to get x and y scroll positions
simple_table_io.inc	View_points_formulae.php View_student_counts.php	THESE ARE USED FOR ENGINEERING AND NOT MANY OTHER DEPARTMENTS Contains functions to allow the programmer to display a table for view/editing in a simple HTML kind of way, by configuring a PageDisplayConfig object, and passing it as a parameter to the ViewPage function.

SQL/	SQL scripts	
Create_TODB_tables.sql Create_TODB_tables_ 2009_10_data.sql Create_TODB_users.sql	Dbsetup – automatic DB setup Manual DB Setup only	To create empty tables Can be used to create tables with test data Adds database users.
static_csv	DIRECTORY FOR CSV output	
styles	DIRECTORY FOR CSS	
table_functions.inc	Lots	SEE SECTION 6.1.2
test_csv.php	Installation Process	Tests csv directory is writable
test_html.html	Installation Process	TODB Test Wizard links to other tests
test_js.html	Installation Process	Tests Javascript works
test_mysql.php	Installation Process	Tests MySQL with PHP
test_php.php	Installation Process	Tests PHP installation
test_tables.php	Installation Process	Tests MySQL tables
timetable_functions.inc	View_people.php View_timetable.php Optional	Functions for producing Timetabling Displays
TODB_Raven_User.txt	Common1.inc. Optional	May be used to hold a list used by Apache to authenticate users. Where used this list is updated whenever people are added or removed from TODB database.
todo.php	Trunk only	Just a TODO List
unit_popup.php	View_units_java.inc	Used to render the contents of the units pop-up box
units_job_table.inc	View_units.php	Used to show Units in Table Actually sets up units query then uses Job_table.inc to do the display
units_table.inc	View_units.php	Used for units as 'glue' before using generic ShowGeneralTable function.
useful.inc	Part of 'template'	Various functions see 6.1.1
variable_listing.php	DEVELOPMENT TOOL	This software parses a PHP file and identifies all variables used:
view_faculty_summary.php	Linked from Index. Optional	Produces totals by job type
view_jobs.php	Linked from Index	Used to view and edit jobs
view_jobs_java.inc	Lots	Javascript for popup editing of jobs
view_jobs_jbs.php	NOT USED	
view_jobs_special.inc	View_jobs.php	Builds the query and associated description for Special Filter options
view_my_jobs.php	Linked from Index and from View_people, other_duties.php	'Glue' before using view_people.php to display jobs for this person and supply forms for them to confirm or send a message to the Teaching Office
view_new_year.php	Linked from Index	Add and remove years. Set current year. Set/unset year in flux.
view_people.php	Linked from Index	Used to view and edit people
view_people_java.inc	Lots	Javascript for popup editing of people
view_points_formulae.php	Linked from Index. Optional	

view_quotas.php	Linked from Index. Optional	
view_related_jobs.php	Table_functions.decorate()	Not used
view_student_counts.php	Linked from Index. Optional	View and edit student counts. Form to allow them to be pasted in
view_summary.php	Linked from Index. Optional	View Teaching Allocation Summary
view_timetable.php	Index.php Optional	Special page developed for Vet School
view_ujobs.php	Linked from Index. Optional	View Jobs by Unit
view_ujobs_printable.php	View_ujobs.php. Optional	A printable version of the above
view_units.php	Linked from Index	Used to view and manage units
view_units_java.inc	Lots	Javascript for popup editing of units
viewmyjobs.inc	View_people.php other_duties.php	Forms for user to confirm or send a message to the Teaching Office

6.1.1 Functions in Useful.inc

Function	**Purpose**
post_exists($key)	Test if the key value is set in the POST form data
post_match($key, $match)	Test if the key value is set in the POST form data and matches $match
post_nomatch($key, $match)	Test if the key value is set in the POST form data and doesn't match $match
post_ncmp($key, $match, $len)	Test if the key value is set in the POST form data and matches $len characters of $match
get_exists($key)	Test if the key value is set in the GET form data
qfix($dat)	?Fix a SQL query by adding slashes before characters that need to be escaped
crs2eng($crsid)	Engineering Specific
ShowTable($conn, $query, $classname)	This function takes an SQL query and shows an HTML table of it. More recent code uses ShowGeneralTable()
GetFormulaeList($yearval)	Get appropriate list of formulae for year from DB
InsertCSVApparatus()	A function to display a CSV button and also respond if it was pressed.
CloseCSVFile()	Close CSV file and give link to file
WriteQueryToCSVFile($resultset, $csv_file_handle)	Take a MySQL resultset and write it to the specified file handle
GetArrayValue($array_arr, $element)	Get the array value for the passed element

6.1.2 Functions in TableFunctions.inc

Function	**Purpose**
ShowGeneralTable ($isadminuser,	Generic Function for displaying a table of information (replaces job_table, people_table, etc)

$adminwantstoedit, $firstheader, $admin_col_hdrs, $result, $admin_cols, $items, $showemail, $cols_hdrs, $csvmode, $csv_file_handle, $cols, $button_cols, $show_quotas, &$linecount, &$points_sum, &$quota_sum)	
editable_entry($line, $item, $array_items, $showemail)	Sets line to editable mode
decorate($line, $item, $csvmode)	Determines if particular entries get highlighted, boldified, italicised, or whatever, based on column and sometimes on content
per_person_count($line, $numcols)	Prints a summary including past year's data