

vtiger 5.x Developer Manual

Prasad

prasad@vtiger.com



www.vtiger.com

Revision History

Version 0.1	October 16, 2007
Version 0.2	October 20, 2007
Version 0.3	October 28, 2007

References

- PHP <http://php.net>
<http://en.wikipedia.org/wiki/PHP>
- Smarty <http://smarty.php.net/crashcourse.php>
<http://en.wikipedia.org/wiki/Smarty>
- SQL <http://en.wikipedia.org/wiki/SQL>
<http://www.w3schools.com/sql/default.asp>
<http://in.php.net/mysql>

Source Organization

vtiger source is moderately structured. Some of the folder and file organization details are described below:

File or Folder	Description
install	Contains file required for installation of product
install.php	It is the entry point for installation. It includes files from install folder depending on the condition (or steps). Both install folder and install.php is renamed after successful installation.
adodb	Contains adodb 3 rd party library files.
cache	Cache directory (configurable during installation) holds temporary files like home page dashboard chart (generated by GD library)
cron	Contains files which calls php scripts offline, the shell scripts can be used by crond (scheduler) with suitable modifications
data	Base class CRMEntity and Tracker file is present in this directory. CRMEntity is extended by our module files.
include	Contains several include files and 3 rd party library.
logs	Logging directory, the logs are created as defined in log4php.properties.
modules	Contains vtiger's module files.
schema	Contains DatabaseSchema.xml which is the snapshot of database structure, it is used by adodb during installation to create the required tables and indexes.
Smarty	Contains Smarty library files and all the required templates to generate the UI is present in the templates directory. Smarty compiles the template files (.tpl) to (.tpl.php) at runtime into the folder templates_c directory.
soap	Contains files which exposes API for our plug-ins.
user_privileges	This directory is used to store the user_privilege and sharing_access computation as .php file. It will also contains files based on the settings enabled in vtiger. These files are created dynamically.

Application Entry Point

The basic entry point to the application are handled through the following files:

index.php

All the actions within vtiger application passes through this file.
This file basically contains the logic of dispatching or including other required files for a particular action.

vtigerservice.php

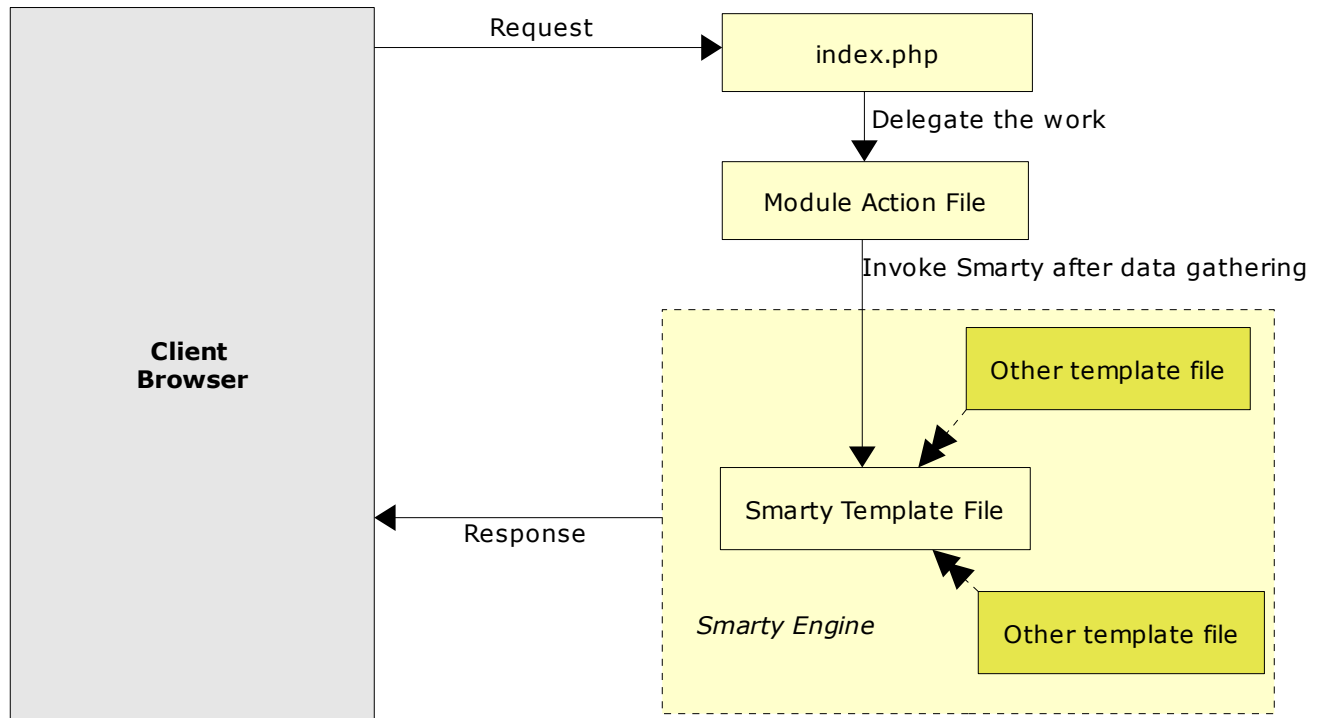
This handles the request from the soap plug-ins (Customer Portal, Firefox, Thunderbird ...)
It has the logic to delegate the task by file inclusion in (soap directory) based on the requested task.

Popup.php

The window pop-up that is shown within vtiger sends request through Popup.php

Control flow overview

Here is the visual representation of the basic control flow that is involved in serving the response to the request from the application.



The client request goes into the index.php which delegates the work to the module action file. The module action file will gather the required data and calls the Smarty template file. The Smarty template file in turn can include several other template files to generate the UI.

NOTE: Smarty templates also does some special case handling while generating the UI. However all the data access logic are within the module action file.

Understanding code base

Let us understand how the UI is generated when you click on the 'Trouble Tickets'.

Ticket ID	Title	Related To	Status	Priority	Assigned To	Action
118	How to automatically add a lead from a w...	Miller Maria	Closed	Normal	admin	edit del
117	Import Error CSV Leads	Taylor Dorothy	Open	Normal	admin	edit del
116	Export Output query	Wilson Susan	Wait For Response	Normal	admin	edit del
115	Individual Customization -Menu and RSS	Brown Elizabeth	In Progress	Normal	admin	edit del
114	Upload Attachment problem	Johnson Patricia	Open	Normal	admin	edit del

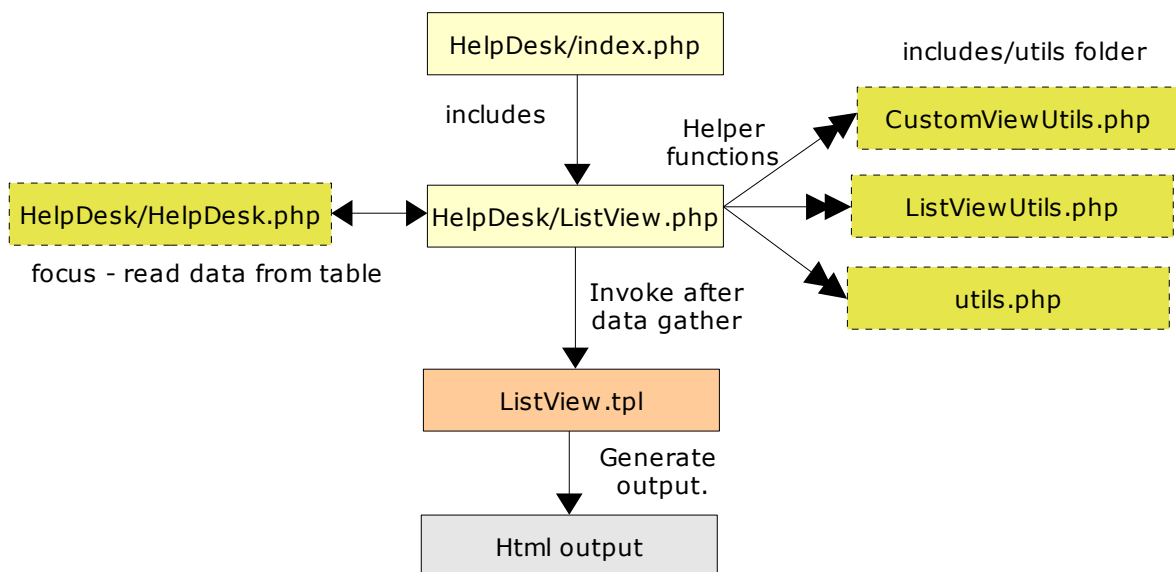
When click on 'Trouble Tickets' under Support menu. The following url is invoked:

```
index.php?module=HelpDesk&action=index&parenttab=Support
```

As already discussed **index.php** is the entry point to most of the actions within vtiger application, the delegation of work from **index.php** is based on the **module** and **action** parameter input.

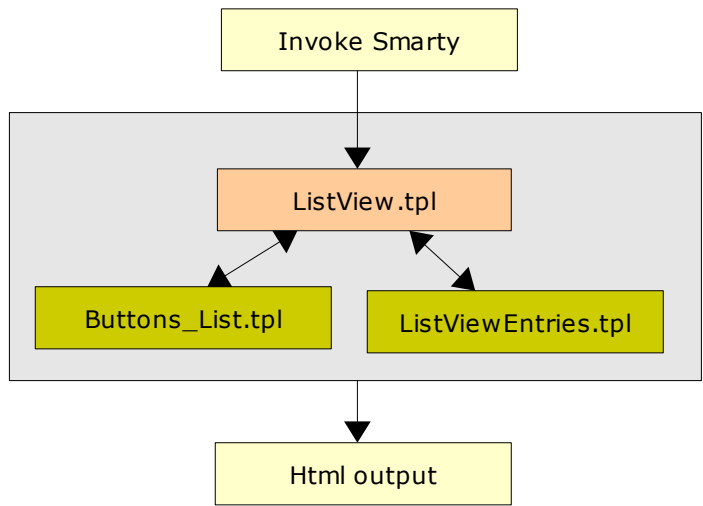
index.php includes the output generated by **module/action.php** generally, expect few special cases like export and ajax actions where the delegation of work slightly defers.

In the current case, **modules/HelpDesk/index.php** file output is included. Given below is the pictorial representation of final output from this file.



ListView.tpl file is present in the smarty/templates directory. Smarty_setup.php file defines vtigerCRM_Smarty class that extends Smarty class. It is the place where Smarty template directory is initialized.

NOTE: Smarty will compile the template file into php file and stores in the compile directory (Smarty/templates_c). If the changes you made doesn't not reflect try to clear the file in compile directory.



The UI generation handled by Buttons_List.tpl and ListViewEntries.tpl is shown below.

The screenshot shows the vtiger CRM 5 interface. The main content area displays a list of tickets under the heading "Support > Trouble Tickets". The list is rendered using the ListViewEntries.tpl template, and the buttons for "Delete" and "Change Owner" are rendered using Buttons_List.tpl. The list contains 5 tickets, with the first one highlighted.

Ticket ID	Title	Related To	Status	Priority	Assigned To	Action
118	How to automatically add a lead from a w...	Miller Maria	Closed	Normal	admin	edit del
117	Import Error CSV Leads		Open	Normal	admin	edit del
116	Export Output query	Wilson Susan	Wait For Response	Normal	admin	edit del
115	Individual Customization -Menu and RSS	Brown Elizabeth	In Progress	Normal	admin	edit del
114	Upload Attachment problem	Johnson Patricia	Open	Normal	admin	edit del

Localization support

vtiger has been localized to nearly 20 different languages. It is important to understand how the strings are organized to support easy translation.

File	Description
include/language/en_us.lang.php	Defines the language strings used commonly across the application \$app_strings
modules/<module>/language/en_us.lang.php	Defines module specific strings \$mod_strings
.js (Javascript)	Several javascript also contains strings that are to translated by language pack

The language pack maintainer (or owner) will bundle only the translated language files (and Javascript files) with the same directory structure.

At the time of sign-in the available languages are shown for selection. The key used to identify the languages (from config.inc.php) are used to include the required language file for translating the strings.

NOTE:

If you are going to develop a module or customize application make sure you add entries in the language file.

While invoking smarty template the following are the general language variable assignments:

1. \$APP to \$app_strings defined in include/language/<sign-in language key>.lang.php
2. \$MOD to \$mod_strings defined in include/modules/<modulename>/language/<sign-in language key>.lang.php

To make other languages available during signin, you have to first unpack the language pack into root directory and then update \$languages variable in config.inc.php

By default en_US (English) language files exists in the product and in config.inc.php

Example: To add German language in config.inc.php

```
$languages = Array('en_us'=>'US English', 'de_de' =>'German');
```

Customizing Menu Label

Our aim is to change the menu label 'Support' to 'Customer Support' as shown in the figure:



Let us try to understand the way in which we can approach the changes knowing very little about the vtiger code.

index.php delegates the work of generating header portion of application (*based on theme selected*) to header.php file.

By default, bluelagoon theme is selected (during signin) so the file which handles header generation in our example is: **theme/bluelagoon/header.php**

theme/bluelagoon/header.php gathers the required data and invokes **Header.tpl**

In smarty/templates/Header.tpl, the following code exist

```
<!-- header - master tabs -->
<td class=tabSeparator></td>
{foreach key=maintabs item=detail from=$HEADERS}
    {if $maintabs ne $CATEGORY}
        <td class="tabUnSelected" ... nowrap><a
href="index.php?module={ $detail[0] }&action=index&parenttab={ $maintabs }">{ $APP[ $maintabs ] }</a></td>
    ...

```

Look at **\$APP[\$maintabs]** we understand the string is getting translated from the application language file. \$maintabs is each key item of the **\$HEADERS** array.

\$HEADERS was assigned in Header.php as:

```
$header_array = getHeaderArray();
$smarty->assign("HEADERS", $header_array);
```

getHeaderArray() is in include/utils/CommonUtils.php

In this function, we are reading the information from the following files:

```
include('parent_tabdata.php');
include('tabdata.php')
```

NOTE: This file is created during the time of installation, to improve the overall performance of the application. The menu and sub-menu details are stored in the table vtiger_parenttab and vtiger_tab tables. Instead of reading the information each time the main page loads from the database, the parent_tabdata.php and tabdata.php are included.

Looking at the vtiger_parenttab table, we find:

parenttabid	parenttab_label	sequence	visible
1	My Home Page	1	0

2	Marketing	2	0
3	Sales	3	0
4	Support	4	0
5	Analytics	5	0
6	Inventory	6	0
7	Tools	7	0
8	Settings	8	0

getHeaderArray() creates a result array with parenttab_label as the key (based on the visibility, sequence and user privilege). *The menu item is taken into account only if has subtabs.*

So \$maintabs in Header.tpl refers to parenttab_label, to achieve the required customization we need to update \$APP['support'] = 'Customer Support'. This can be done by updating the entry in application language file include/language/en_us.lang.php

The following the change:

```
$app_strings = array(
...
'support' => 'Customer Support',
...);
```

Instead of:

```
$app_strings = array(
...
'support' => 'Support',
...);
```

The impact of this change will occur at all places where this language key was referred to. You can see the new label is appears in the UI generated by Buttons_List.tpl (for ListView) as well.



Exercise:

1. Change the 'Customer Support' to 'HelpDesk'.

Customizing Sub-menu Label

Our aim is to change the sub-menu label 'FAQ' to 'Knowledge Base' as shown in the figure:



[This customization is similar to menu label expect few minor difference].

In Smarty/templates/Header.tpl we see:

```
<!-- - level 2 tabs starts-->
<TABLE border=0 cellspacing=0 cellpadding=2 width=100% class="level2Bg" >
<tr>
  <td>
    <table border=0 cellspacing=0 cellpadding=0>
      <tr>
        {foreach key=maintabs item=detail from=$HEADERS}
          {if $maintabs eq $CATEGORY}
            {foreach key=number item=module from=$detail}
              {if $module eq $MODULE_NAME}
                <td class="level2SelTab" nowrap><a
href="index.php?module={$module}&action=index&parenttab={$maintabs}">{$APP[$module]}</a></td>
              </td>
            </foreach>
          </if>
        </foreach>
      </tr>
    </table>
  </td>
  ...
</tr>
</table>
```

Looking getHeaderArray() function again, we arrive at the following conclusion:

Each of the submenu is tied to the module, the information is available in vtiger_tab table and

vtiger_tab information:

tabid	id
name	Module name
presence	UI presence, 0 - default
tabsequence	Overall display order
tablabel	Sub-menu label, used as key in language file
modifiedby	Last modified user
modifiedtime	Time of last modification
customized	
ownedby	Userid who owns this record

The FAQ tab record has the following information:

tabid	name	presence	tabsequence	tablabel	modifiedby	modifiedtime	customized	Ownedby
15	Faq	0	14	Faq	NULL	NULL	0	1

NOTE:

The relation between the menu and sub menu (parenttab and tab) is obtained from vtiger_parenttabrel. Naming convention for relational table:
Table connecting two other table vtiger_first and vtiger_second is named as vtiger_firstsecondrel.

So \$module in Header.tpl refers to tablabel, to achieve the required customization we need to update \$APP['Faq'] = 'Knowledge Base'. This can be done by updating the entry in application language file include/language/en_us.lang.php

The following is the change:

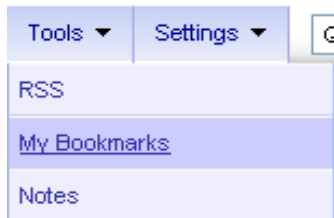
```
$app_strings = array(
...
'Faq' => 'Knowledge Base',
...);
```

Instead of:

```
$app_strings = array(
...
'Faq' => 'FAQ',
...);
```

Exercise:

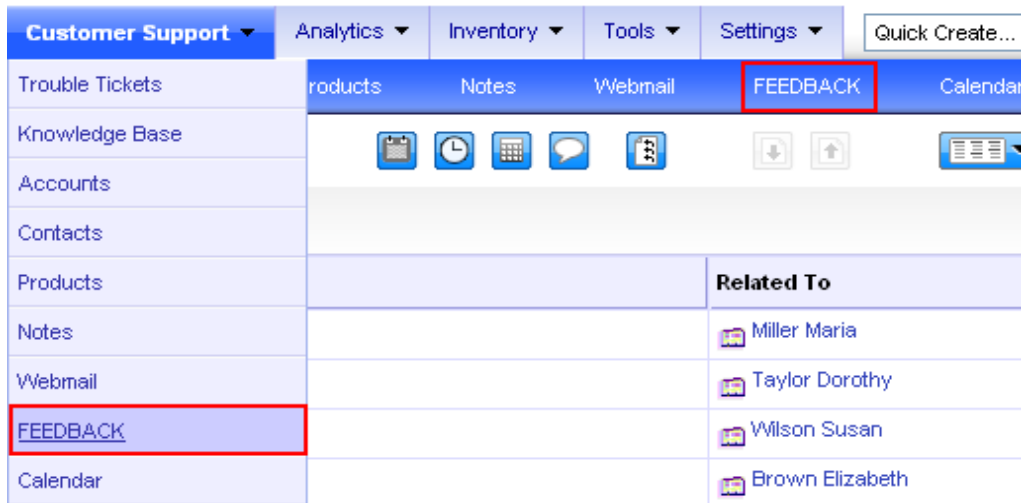
1. Change the 'My Sites' to 'My Bookmarks' under Tools menu, as shown in the figure:



HINT: The link to 'My Sites' is `index.php?module=Portal&action=index&parenttab=Tools`

Adding Sub-menu Item

We want to add a 'FEEDBACK' menu item under our 'Customer Support' as shown in the figure:



To achieve this we have written this script (tut-addsubtab.php), to execute this you have to copy it to vtigercrm root folder.

```
<?php
require_once('config.php');
require_once('include/database/PearDatabase.php');
require_once('include/utils/utils.php');

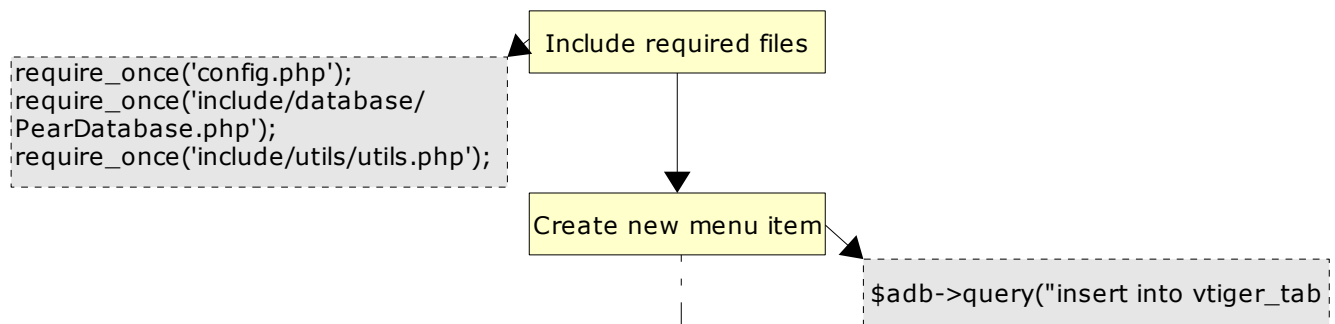
// $adb is defined in PearDatabase.php

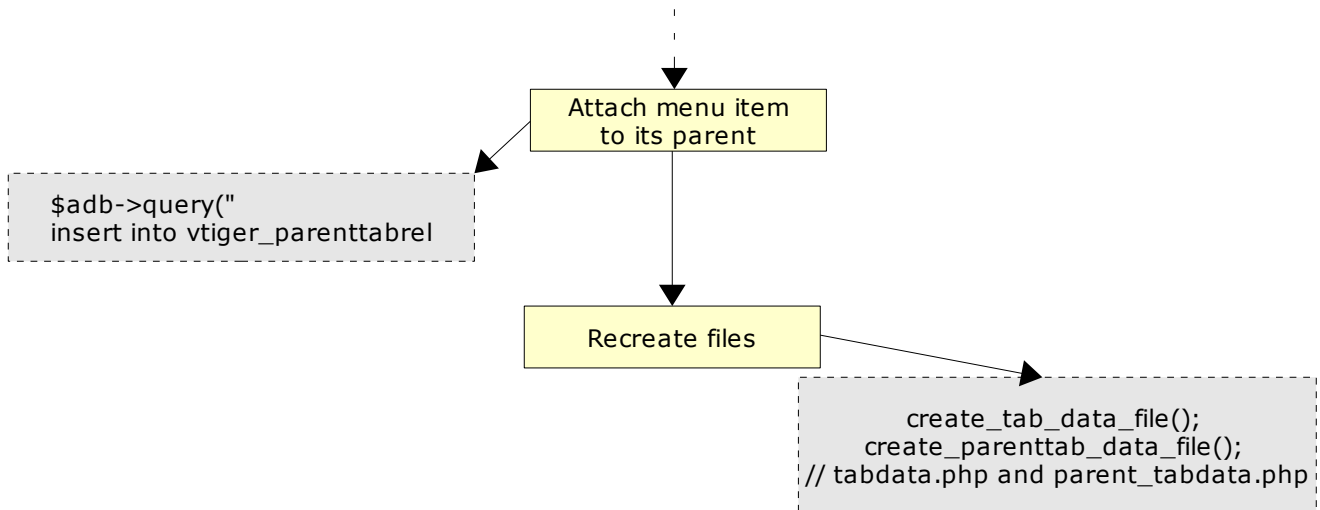
// vtiger_tab (tabid, name, presence, tabsequence, tablabel,
//             modifiedby, modifiedtime, customized, ownedby);
$adb->query("insert into vtiger_tab values(30, 'Feedback', 0, 27, 'Feedback', null, null,
0, 1)");

// attach the tab to parent tab
// vtiger_parenttabrel (parenttabid, tabid, sequence)
$adb->query("insert into vtiger_parenttabrel values(4, 30, 7)");

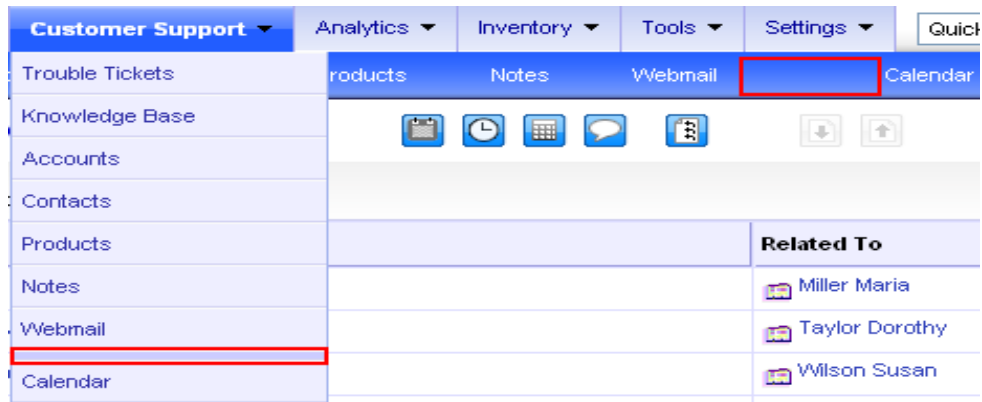
create_tab_data_file();
create_parenttab_data_file();
?>
```

Given below is the pictorial description of the above script:





The result of the above changes is shown in the figure.



One final step to make the menu item appear is to add the tablabel mapping in the language file.

The following entry needs to be added in `include/language/en_us.lang.php`

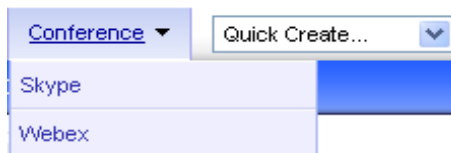
```

$app_strings = array(
    ...
    'Feedback' => 'FEEDBACK',
    ...);
  
```

NOTE: If the mapping already exist in language file, there is no need of duplication.

Exercise:

1. Add a new menu 'Conference' with the following items:
 - a) Skype
 - b) Webex



NOTE: Menu is not shown if it no items is associated with it.

UI Views Introduction

Following are the different UI views we need know:

1. List View
2. Create | Edit View
3. Detail View
4. Quick Create View
5. Popup View

List View

The module records are listed in the tabular format based on the filter (custom view) selected (or set as default). This is the default view for most of the modules as shown in the figure below.

Customer Support > Trouble Tickets Button_List.tpl

Showing 1 - 5 of 5 **ListView.tpl** Filters: All New | Edit | Delete

Ticket ID	Title	Related To	Status	Priority	Assigned To	Action
118	How to automatically add a lead from a w...	Miller Maria	Closed	Normal	admin	edit del
117	Import Error CSV Leads	Taylor Dorothy	Open	Normal	admin	edit del
116	Export Output query	Johnson Patricia	Wait For Response	Normal	admin	edit del
115	Individual Customization -Menu and RSS	Brown Elizabeth	In Progress	Normal	admin	edit del
114	Upload Attachment problem	Johnson Patricia	Open	Normal	admin	edit del

Customer Support > Trouble Tickets Create Mail Merge templates

Create View

When a new record for a module is created this create view is shown. Create view can be further break down into Basic Information and Advanced Information tabs which are displayed based on the module. Each of the tabs further has the blocks. The blocks contains the actual fields of inputs.

Customer Support > Trouble Tickets Buttons_List1.tpl

Creating New Ticket Create Ticket...

Basic Information CreateView.tpl

Ticket Information BAS - Basic Block

Assigned To: User Group
 admin Contacts File Folder

Priority: Low DisplayFields.tpl Product Name: File Folder

Severity: Minor Status: Open

Category: Big Problem Attachment: Browse...

*Title:

Description Information BAS - Basic Block

Description: DisplayFields.tpl

Edit View

When a module record is under edit, Edit View is displayed. The Edit View is similar to Create View with few modification, it can contain more blocks, the input fields are pre-populated with the data.

Customer Support > Trouble Tickets

[118] How to automatically add a lead from a web form to VTiger - Editing Ticket Information
Updated today (20 Oct 2007)

Buttons_List1.tpl

salesEditView.tpl

Save Cancel

Ticket Information Block Information

Assigned To	<input checked="" type="radio"/> User <input type="radio"/> Group admin	Contacts	Miller Maria
Priority	Normal	Product Name	Vtiger 50 Users Pack
Severity	Minor	Status	Closed
Category	Other Problem	Attachment	Browse...
*Title	How to automatically add a lead from a web form to VTiger		

Description Information Block Information

Description	DisplayFields.tpl
-------------	-------------------

Solution Information Block Information

Detail View

When individual record is displayed or after a new record is created the detail view is shown. Based on the edit permission the field editing is allowed. If SinglePane detail view is not set, then the blocks are grouped under basic module information and more information tabs.

Customer Support > Trouble Tickets

[118] How to automatically add a lead from a web form to VTiger - Trouble Tickets Information
Updated today (20 Oct 2007)

Buttons_List1.tpl

More Information

DetailView.tpl

Edit Convert As FAQ Duplicate Delete

Ticket Information Block Information

Assigned To	User admin	Contact Name	Miller Maria
Priority	Normal	Product Name	Vtiger 50 Users Pack
Severity	Minor	Status	Closed
Category	Other Problem	Created Time	2007-10-20 18:11:24
Modified Time	2007-10-20 18:11:24	Attachment	
Title	How to automatically add a lead from a web form to VTiger		

Description Information Block Information

Description	DetailViewUL.tpl
-------------	------------------

Solution Information Block Information

Solution	DetailViewUL.tpl
----------	------------------

Comment Information Block Information

TAG CLOUD
vtiger_50usr Tag it

Select template to Mail Merge:
Create Mail Merge templates

DetailView.tpl

Quick Create View

Many modules support quick create. Using this minimal (required) information can be entered instead of entering all the information in the CreateView. Shown below is a simple quick create form to create new ticket.

The screenshot shows a web interface for creating a ticket. At the top, there is a navigation bar with dropdown menus for 'Inventory', 'Tools', and 'Settings'. A dropdown menu is open, showing 'New Ticket' selected. Below this is a form titled 'Create Ticket' with a 'Quick Create' label. The form has a 'Title' field (marked with an asterisk), a 'Description' field, an 'Attachment' field with a 'Browse...' button, and a 'Priority' dropdown menu set to 'Low'. At the bottom of the form are 'Save' and 'Cancel' buttons.

Popup View

Popup view is displayed for input selection. For example, when creating the quote the product list is shown in popup view and the product can be selected as shown in the figure below:

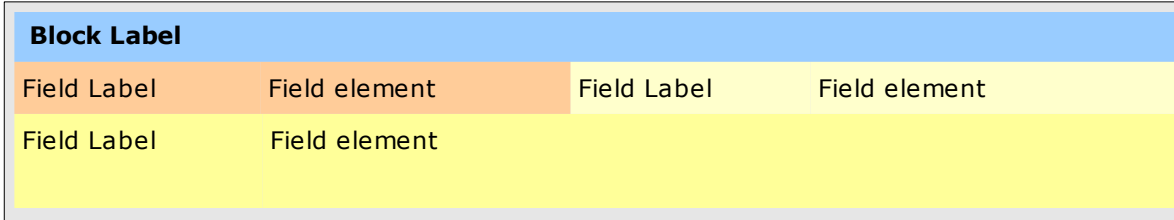
The screenshot shows a 'Products' popup window. At the top, there is a search bar with a magnifying glass icon and the text 'Search Basic mode'. To the right of the search bar is a dropdown menu for 'In Product Name' and a 'Search Now' button. Below the search bar is a row of letters from A to Z. The main content area is titled 'PopupContents.tpl' and shows a table of products. The table has three columns: 'Product Name', 'Part Number', and 'Unit Price'. The table contains 10 rows of product data. The text 'Showing 1 - 10 of 10' is displayed at the bottom right of the table.

Product Name	Part Number	Unit Price
Vtiger Single User Pack	001	...
Vtiger 5 Users Pack	002	...
Vtiger 10 Users Pack	003	...
Vtiger 25 Users Pack	023	...
Vtiger 50 Users Pack	005	...
Double Panel See-thru Clipboard	sg-106	...
abcd1234	1324356	...
Cd-R CD Recordable	sg-108	...
Sharp - Plain Paper Fax	sg-119	...
Brother Ink Jet Cartridge	sg-125	...

UI Block and Field

The UI generation for various forms is driven by database and hence customizing it is very simple. Understanding the form structure is important to achieve required level of field customization.

In the previous section we learnt about various views. Each view (except ListView) has the following basic structure.



The field which comprises of label and the ui (input) element is enclosed under the block. The block can also have the label. Each block has several rows split into two columns. The fields are layed out in each column based on the sequence number as shown in the above figure.

In this section we shall understand how to add a new block, field and customize it. For code refer script (tut-addnewblockfield.php).

Add block with field

We want to add 'Version Information' block for the Trouble Tickets module, with a single 'Version' text field, as shown in the figure:

*Title Test Field and Block Addition

Description Information

Description

Version Information

Version 0.3

Save Cancel

To achieve this we need entries in vtiger_block and vtiger_field tables as shown below:

```
// Get tabid to use
$tabres = $adb->query("select tabid from vtiger_tab where name='HelpDesk'");
$tabid = $adb->query_result($tabres, 0, 'tabid');

// Get max blockid in use
$blockres = $adb->query("select max(blockid) as maxblockid from vtiger_blocks");
$blockid = $adb->query_result($blockres, 0, 'maxblockid');

$adb->query("insert into vtiger_blocks values(" . ($blockid+1) . ", $tabid,
'LBL_VERSION_INFORMATION', 4, 0, 0, 0, 0, 0)");
```

```
$fieldid = $adb->getUniqueId("vtiger_field"); // use this instead of max(fieldid)
$adb->query("insert into vtiger_field values($tabid, $fieldid, 'version',
'vtiger_troubletickets', 2, 1, 'prodversion', 'Version', 1, 0, 0, 100, 1, ".$blockid+1).",
1, 'V~0', 1, null, 'BAS')");
```

vtiger_blocks information:

blockid	id for the block
tabid	module to which the block is associated (vtiger_tab)
blocklabel	Label to display as block title (used as key in modules/module/language/en_us.lang.php – language file)
sequence	Determines the placement order on the page (blocks are stacked one below the other)
show_title	0 – show title, 1 – hide title
visible	0 – show field, 1 – hide field
createview	0 – show in create view, 1 – hide in create view
editview	0 – show in edit view, 1 – hide in edit view
detailview	0 – show in detail view, 1 – hide in detail view

vtiger_field information:

tabid	module to which the field is associated (vtiger_tab)
fieldid	id for the field, use <code>adb->getUniqueId("vtiger_field")</code>
columnname	Name of the table name where values are stored
tablename	Name of the table for this field
generatedtype	
uitype	Refer UI Type information below
fieldname	Name to identify this field, used in special cases like. (vtiger_<fieldname> is taken as the picklist value table)
fieldlabel	Label to display on the UI (for most modules this is the key for module string in the file: modules/module/language/en_us.lang.php)
readonly	1 – allow edit, 0 – disallow edit (could be handled by uitype)
presence	1
selected	1
maxlength	Length associated with this field element
sequence	Order of field placement inside the block
block	blockid to which this field is associated
displaytype	1 – show field inside block, 0 – hide field inside block
typeofdata	Refer TypeOfData information below
quickcreate	0 – show field in quickcreate view, 1 – hide field in quickcreate view
quickcreatesequence	Order of field placement inside quickcreate view
info_type	BAS – basic information, ADV – advanced information

When adding the new field we have associated it with tablename=vtiger_troubletickets and columnname=version, this means the value of this field should be stored in this column.

To achieve this we execute this statement:

```
// Add the table column  
$adb->query("alter table vtiger_troubletickets add column version varchar(10)");
```

Now when a new trouble ticket is created the detail view of the ticket also shows the version information as shown below.

Title		Test Field and Block Information	
Description Information			
Description			
Version Information			
Version		0.3	

NOTE: The block is not displayed if it does not have any fields associated with it or all the fields of the block are hidden.

Field Access Control

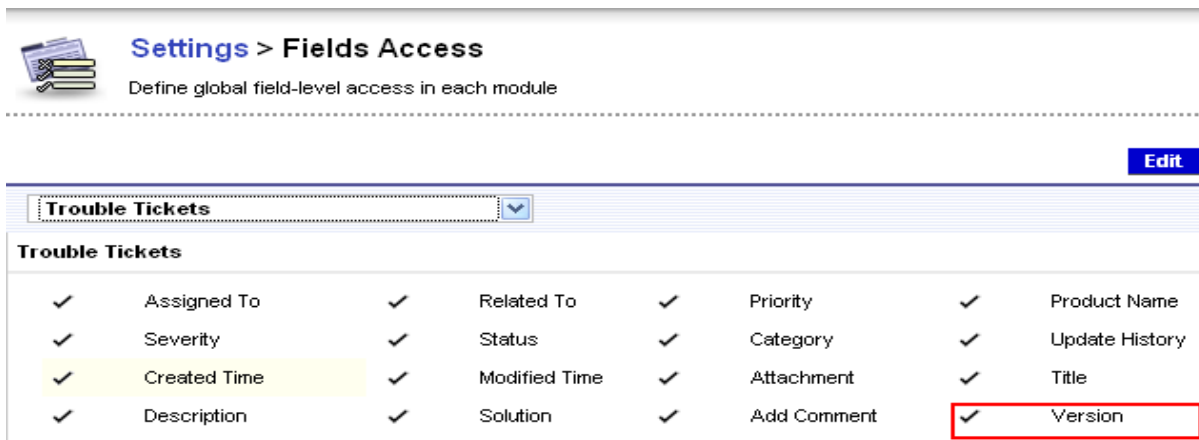
vtiger_def_org_field table stores the information which controls the field access across the organization. With this administration can control the display of the field of the module through the Settings -> Field Access page.

To achieve this we have to execute this statement:

```
$adb->query("insert into vtiger_def_org_field values($tabid, $fieldid, 0, 1)");
```

vtiger_def_org_field information:

tabid	module to which the field is associated (vtiger_tab)
fieldid	field id
visible	0 – show, 1 – hide [for non-admin]
readonly	0 – true, 1 – false



Settings > Fields Access
Define global field-level access in each module

[Edit](#)

Trouble Tickets

Trouble Tickets

<input checked="" type="checkbox"/>	Assigned To	<input checked="" type="checkbox"/>	Related To	<input checked="" type="checkbox"/>	Priority	<input checked="" type="checkbox"/>	Product Name
<input checked="" type="checkbox"/>	Severity	<input checked="" type="checkbox"/>	Status	<input checked="" type="checkbox"/>	Category	<input checked="" type="checkbox"/>	Update History
<input checked="" type="checkbox"/>	Created Time	<input checked="" type="checkbox"/>	Modified Time	<input checked="" type="checkbox"/>	Attachment	<input checked="" type="checkbox"/>	Title
<input checked="" type="checkbox"/>	Description	<input checked="" type="checkbox"/>	Solution	<input checked="" type="checkbox"/>	Add Comment	<input checked="" type="checkbox"/>	Version

Profile Level Sharing Access:

vtiger_profile table stores the information about the field access for each profile. To enable field administration through Settings -> Profile UI we need to add suitable entry into this table.

When a new field is added we can enable its access to all the profile. Later its access can be restricted selectively for each profile through the Settings -> Profile UI.


The following snippet of code enables access to new field to all the profile:

```
$profileres = $adb->query("select profileid from vtiger_profile");
$profilecnt = $adb->num_rows($profileres);
for($profileidx=0; $profileidx < $profilecnt; $profileidx++) {
    $profileid = $adb->query_result($profileres, $profileidx, "profileid");
    $adb->query("insert into vtiger_profile2field values ($profileid, $tabid,
$fieldid, 0, 1)");
}
```

vtiger_profile information:

profileid	id of the profile
tabid	module to which the field is associated (vtiger_tab)
fieldid	field id
visible	0 – show, 1 – hide
readonly	0 – true, 1 – false

Set Privileges for each Module

modules to be shown	Edit Permissions			Fields & Tools Settings
	Create/Edit	View	Delete	
<input checked="" type="checkbox"/> Dashboard				
<input checked="" type="checkbox"/> Trouble Tickets	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/> Assigned To	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Priority
<input checked="" type="checkbox"/> Product Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Status
<input checked="" type="checkbox"/> Category	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Created Time
<input checked="" type="checkbox"/> Modified Time	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Title
<input checked="" type="checkbox"/> Description	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Add Comment
<input checked="" type="checkbox"/> Version				
<input checked="" type="checkbox"/> Merge				

TypeOfData information:

This information is used field validation (when form is submitted) from the client.

It can have more than two parts separated by ~. The general format is as follows:

DataType~ValuePresence~OtherRulesForValidation

DataType options:

C	Boolean
D	Date
DT	DateTime
E	Email
I	Integer
N	Number
P	Password
T	Time
V	Varchar

ValuePresence options:

O	Optional
M	Mandatory

Example:

1. V~M – The field is of type varchar and mandatory.
2. D~O~OTH~GE~support_start_date~Support Start Date

[Refer: include/js/general.js – function formValidate for usage understanding]

UI Type information:

UI type determines the HTML element generation for the different view:

1	Text box
2	Text box, mandatory entry
3	Text box with Inheritance
4	Text box with Inheritance, mandatory entry
5	Date
6	Date, default to currenttime
7	Number box
8	Non editable text box
9	Percent
11	Phone
12	OrgUnit pickbox
13	EMail
14	Organization multiselect pickbox
15	Picklist
16	Picklist, mandatory entry
17	URL
18	URL with Inheritance
19	Textarea with colspan=2
20	Textarea with colspan=2, mandatory entry
21	Textarea
22	Textarea, mandatory entry
23	Date
24	Textarea, mandatory entry
30	Time left
31	Text box, Extending Inheritance
32	Text box, Extending Inheritance, mandatory entry
33	Multi-select combo box
34	Large multi-select combo box
50	Popup select box for account, mandatory entry potentials module.
51	Popup select box for account and contact addresses
52	Picklist for username entries
53	User picklist
54	Group picklist
55	Salutation type picklist
56	Checkbox
57	Contacts popup select box

58	Campaign popup select box
59	Product non-editable capture, popup picklist
61	Attachments, file selection box
62	Names out of entities popup picklist
63	Duration minutes picklist
64	Calendardate
66	Names out of entities popup picklist
67	Names out of entities
68	Names out of entities popup picklist
69	Products attachments
70	Date
71	Currency
72	Currency, mandatory entry
73	Popup select box for Accounts, mandatory entry
75	Vendor name
76	Potential popup picklist
77	Picklist for secondary username entries
78	Quote popup picklist
79	Purchase order name
80	Sales order popup picklist
81	Vendor name, mandatory entry
83	Tax in Inventory
85	Skype
90	
98	Role name popup picklist, mandatory entry
99	Password, mandatory entry
101	User capture popup picklist
103	Text box
104	EMail, mandatory entry
105	User image
106	Text box, mandatory entry
107	Company logo
111	Non editable picklist
115	Non editable picklist
116	Currency in user details
156	Admin toggle, checkbox
357	Email, Popup picklist

[Mandatory ui types show * mark beside the field label. Smarty/templates/DisplayField.tpl and include/utils/EditViewUtils.php file gives more information on how the uitype is used to control field display.]

Making the field mandatory

When a field is made mandatory its input value cannot be empty during save. Let us change the newly added text field (version) for Trouble Tickets previously as mandatory.

The script (tut-makefieldmandatory.php) shown below is written to achieve this effect:

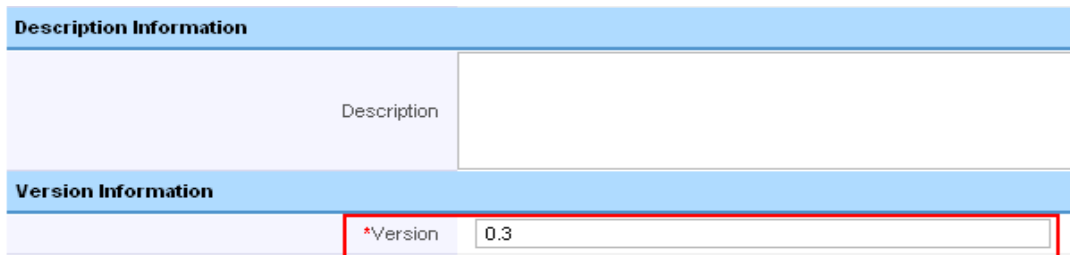
```
// Get tabid to use
$tabres = $adb->query("select tabid from vtiger_tab where name='HelpDesk'");
$tabid = $adb->query_result($tabres, 0, 'tabid');

// Get fieldid to work with
$fieldres = $adb->query("select fieldid from vtiger_field where tabid=$tabid and
columnname='version' and tablename='vtiger_troubletickets'");
$fieldid = $adb->query_result($fieldres, 0, 'fieldid');

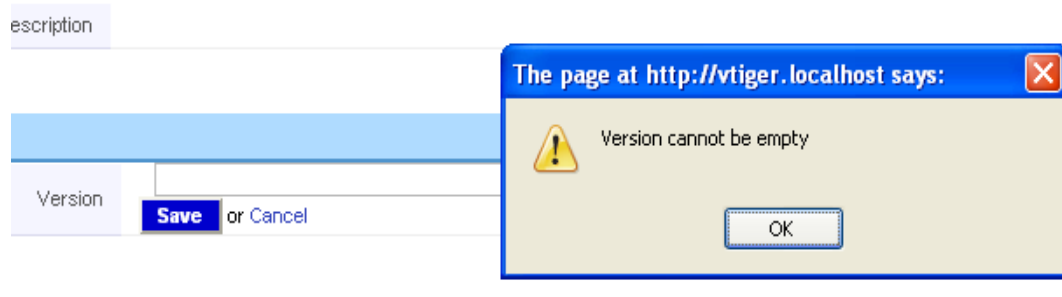
if($fieldid != '') $adb->query("update vtiger_field set uitype=2, typeofdata='V~M' where
tabid=$tabid and fieldid=$fieldid");
```

As you can see, we have set the UI Type to be 2 (to show * on UI) and typeofdata as V~M.

After this script is executed, you can see the changes when creating new ticket or updating the version field of existing ticket.




The screenshot shows a form with two sections: "Description Information" and "Version Information". The "Description" field is a large text area. The "Version" field is a smaller text input field containing the value "0.3". The "Version" label is followed by an asterisk (*), indicating it is a mandatory field. The entire "Version" field area is highlighted with a red border.



The screenshot shows the same form as above, but the "Version" field is now empty. A dialog box is overlaid on the form, titled "The page at http://vtiger.localhost says:". The dialog box contains a yellow warning icon and the text "Version cannot be empty". There is an "OK" button at the bottom of the dialog box. The "Save" button is visible on the form.

Exercise:

1. Add 'Updated On' field to 'Version Information' block of Trouble Tickets module.
 - The value should be saved in 'updatedon' column in vtiger_troubletickets table.
 - The default value should be '0000-00-00 00:00:00' if no value is selected from user.
 - Field Access control should be available for this.
 - Profile level access should be available for this field in all existing profiles.

Version Information	
*Version	<input type="text"/>
Updated On	<input type="text"/>  <i>(yyyy-mm-dd)</i>

Version Information	
Version	0.3
Updated On	2007-10-29

Display field in default filter

The default filter for the modules is 'All', this cannot be customized through UI. Let us now try to add our new field 'version' created previously to be shown in this filter along with the other fields.

The filter information are stored in vtiger_customview and vtiger_cvcolumnlist tables.

vtiger_customview information:

cvid	id of the custom view
viewname	Name of the custom view (filter)
setdefault	0 – no, 1 – yes [mark this filter as default for the module]
setmetrics	0 – no, 1 – yes [display on the home page the data count]
entitytype	Module name to which the custom view is associated

vtiger_cvcolumnlist information:

cvid	id of the custom view (from table vtiger_customview)
columnindex	Position at which this column should appear in the list view
columnname	Colon separated column information, has the following format: tablename:columnname:fieldname:ModuleName_ListViewDisplayLabel:DataType Example: vtiger_troubletickets:parent_id:parent_id:HelpDesk_Related_to:I In list view the column heading will be Related To

The following snippet of code will add the version field to 'All' filter of Trouble Tickets module:

```
// Get max columnindex in use for the cv
$cvcolres = $adb->query("select max(columnindex) as maxcolumnindex from vtiger_cvcolumnlist
where cvid=$cvid");
$cvcolidx = $adb->query_result($cvcolres, 0, 'maxcolumnindex');
$adb->query("insert into vtiger_cvcolumnlist values($cvid, " . ($cvcolidx+1) . ",
'vtiger_troubletickets:version:prodversion:HelpDesk_version:V')");
```

Customer Support > Trouble Tickets

<input type="checkbox"/>	Ticket ID ▲	Title	Related To	Status	Priority	Assigned To	version	Action
<input type="checkbox"/>	129	Test Field and Block Information		Open	Low	admin	0.3	edit del
<input type="checkbox"/>	118	How to automatically add a lead from a w...	Wilson Susan	Closed	Normal	admin		edit del
<input type="checkbox"/>	117	Import Error CSV Leads	Johnson Patricia	Open	Normal	admin		edit del
<input type="checkbox"/>	116	Export Output query	Taylor Dorothy	Wait For Response	Normal	admin		edit del
<input type="checkbox"/>	115	Individual Customization -Menu and RSS	Moore Margaret	In Progress	Normal	admin		edit del
<input type="checkbox"/>	114	Upload Attachment problem	Davis Jennifer	Open	Normal	admin		edit del

Configurable PickList Field

Having version field as the select box is more preferable than a text field. vtiger lets you configure the select box, this is called pick list. The PickList Editor under Settings allows the configuration of values for each module field which are picklist fields (15, 16, 33, 111)

In this section we are going to understand the conversion of text field version to picklist field version.

To identify the initial need you can look at modules/Settings/PickList.php which handles picklist value configuration.

The basic assumption behind the picklist field are as follows:

1. The UI type is one of these values (15, 16, 33, 111)
2. The picklist values are stored in the table vtiger_<fieldname> where fieldname is the value entered in vtiger_field for the field.
3. vtiger_<fieldname> should be created before using picklist editor and should have the following basic structure:

id	Auto increment column to keep uniqueness of value
<fieldname>	Column name should be same as fieldname column used in vtiger_field
sortorderid	int default 0 column – to store sorting order of values
presence	int default 1 column – control value edit using picklist editor.

NOTE: The number and the order of column is very essential item in the picklist field table. When the value is saved from the picklist editor insert statements sends only 4 values without columnname. If you have any other columns in this table make sure to have proper default values for the other columns.

The following code snippet will convert the previously added version text field to mandatory picklist field. (Refer tut-makefieldpicklist.php for complete code)


```
// Get tabid to use
$tabres = $adb->query("select tabid from vtiger_tab where name='HelpDesk'");
$tabid = $adb->query_result($tabres, 0, 'tabid');

// Get fieldid to work with
$fieldres = $adb->query("select fieldid, fieldname from vtiger_field where tabid=$tabid and
columnname='version' and tablename='vtiger_troubletickets'");
$fieldid = $adb->query_result($fieldres, 0, 'fieldid');
$fieldname = $adb->query_result($fieldres, 0, 'fieldname');

if($fieldid != '') {
    $adb->query("update vtiger_field set uitype=16, typeofdata='V~M' where tabid=$tabid
and fieldid=$fieldid");
}

$adb->query("create table vtiger_{$fieldname} (id int auto_increment, {$fieldname} varchar(25),
sortorderid int default 0, presence int default 1, primary key(id))");
```

After the field is converted to picklist you can see it under picklist editor when Trouble Tickets module is selected as shown in the figure below:

 **Settings > Picklist Editor**
 Customize Picklist values in each module

1. Select Module

Select CRM Module:

2. Picklists Available in Trouble Tickets

Priority	Edit	Severity	Edit
Low		Minor	
Normal		Major	
High		Feature	
Urgent		Critical	

Category	Edit	Version	Edit
Big Problem			
Small Problem			
Other Problem			

The values can be configured by editing, as shown below:

Version **Edit Pick List - Version** [X]

Type the entries one - by - one below and click the Save button to save the list.

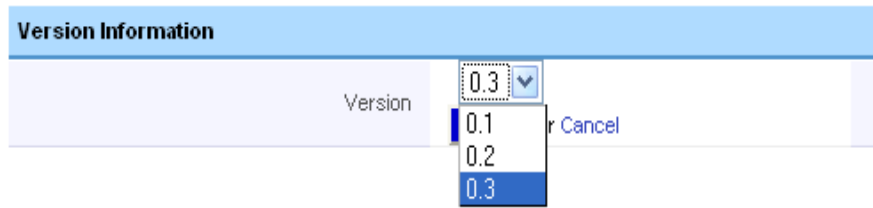
0.1
0.2
0.3

Save **Cancel**

These values are stored in vtiger_prodversion table that was created us.

Version	Edit
0.1	
0.2	
0.3	

The picklist version field is shown in detailview, create/edit view of the Trouble Ticket module.



Exercise:

1. Add mandatory picklist field 'Release' to 'Version Information' block of 'Trouble Tickets'
 - o The input value should be stored in release column of vtiger_troubletickets table
 - o Field Access and Profile Level Access should be available for this column.
 - o The picklist values should be (Stable, RC, Validation)
 - o The Stable value should not be editable in the picklist editor.

