

vtlib – vtiger development library
version 1.2

Table of Contents

About	3
Requirements	3
Applying Customization	3
Testing Module Creation	4
API Usage	5
Creating Tab (Menu Item)	5
Creating Block	6
Creating Field	7
Creating Custom View	9
Enable/Disable Action	10
Sharing Access Setting	11
Custom Fields Setup	12
Creating New Module	13
Creating Payslip Module	14
Module Entry Point	15
Default List view	15
Create view	16
Detail view	16
List view	16
Export – Import Tool	17
Sharing Access Setting	17
Custom Field Addition	18

About

vtlib aims at providing easier development environment for vtiger. In this version API's are provided which lets you create module easily for vtiger.

Requirements

vtiger 5.0.4 should be installed.

NOTE: Source directory in this document refers, vtiger CRM source installation.
If you have used vtiger CRM bundled installation like, .exe or .bin, it will be located in
<vtigercrm_install_directory>\apache\htdocs\vtigerCRM

Applying Customization

1. Take backup of your vtiger CRM Source directory.
2. Unpack the vtlib-x.y.zip into your vtiger CRM source directory.

Files and Folder:

Existing file changes		
include/utlis	export.php	Hooks added to support record export for modules.
	UserInfoUtil.php	Added function to obtain custom module names for controlling Sharing Access.
modules/Import	ImportStep1.php	Hooks added to support record import into modules.
	ImportStep2.php	
	ImportStep3.php	
	ImportSteplast.php	
	ImportStepundo.php	
	UsersLastImport.php	
modules/Users	CreateUserPrivilegeFile.php	Hooks added to save the sharing access and user privileges for custom modules.
modules/Settings	OrgSharingDetailView.php	Hooks added to provide custom module sharing access.
Smarty/templates	Buttons_List.tpl	Hook provided to enable import and export buttons for custom modules.
	Buttons_List1.tpl	

New folder/file additions

modules/Payslip	Example modules for testing vtlib scripts.
modules/EmailScanner	

Test Scripts

vtlib.Test.html	Provides link to execute the following scripts
vtlib.Test.Create.Menu.Item.php	Example to create menu item

vtlib.Test.Create.Module.Blocks.php	Example to show module creation and block for it.
vtlib.Test.html	Provides link to execute the following scripts
vtlib.Test.Create.Module.Fields.php	Example to show module creation and fields for it.
vtlib.Test.Create.Module1.php	<p>Payslip Module Creation (modules/Payslip dir is already packaged)</p> <p>Make the following entry in include/language/en_us.lang.php after script execution</p> <pre>\$app_strings = Array('Payslip' => 'PaySlip',</pre>
vtlib.Test.Create.Module2.php	<p>EmailScanner Module Creation (modules/EmailScanner dir is already packaged)</p> <p>Make the following entry in include/language/en_us.lang.php after script execution</p> <pre>\$app_strings = Array('EmailScanner' => 'Email Scanner',</pre>

Testing Module Creation

1. After unpacking the vtlib zip file, open <http://localhost/vtigercrm/vtlib.Test.html>
2. Click on [Create Payslip Module](#) to test creation of Payslip Module
3. Click on [Create EmailScanner Module](#) to test creation of EmailScanner Module

NOTE: When you create module as said above, you will see blank links added under Tools menu. Kindly edit include/language/en_us.lang.php and update \$app_strings as shown below:

```
$app_strings = Array (
'Payslip' => 'PaySlip',
'EmailScanner' => 'EMAIL SCANNER',
...

```

After this you will be able to see **Tools » Payslip** and **Tools » EMAIL SCANNER**

API Usage

Given below are examples on how to use the API's provided by vtlib.

Creating Tab (Menu Item)

```
include_once('vtlib/vtiger/Tab.php');  
Vtiger_Tab::create('<NEWMODULE>', '<NEWMODULE_LABEL>', '<MENUNAME>');
```

This API will create new menu item which serves as entry point for the module.

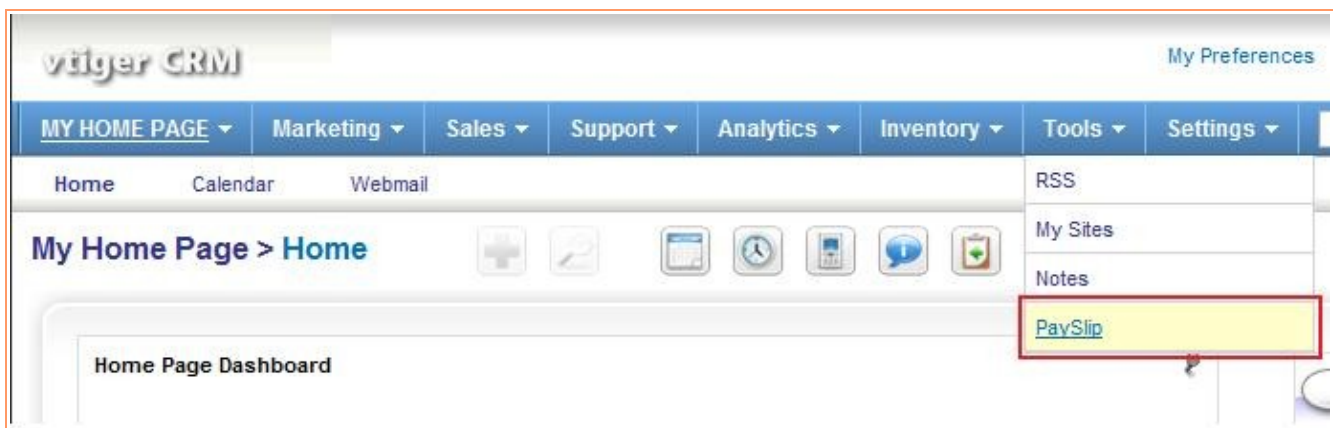
Example:

```
Vtiger_Tab::create('Payslip', 'Payslip', 'Tools');
```



NOTE: You will need to put the <NEWMODULE_LABEL> mapping in the language file (include/language/en_us.lang.php) to see the affect.

```
$app_strings = Array (  
'Payslip' => 'PaySlip',  
...
```



Creating Block

```
include_once('vtlib/vtiger/Block.php');  
vtiger_Block::create('<MODULE>', '<BLOCK_LABEL>');
```

This API will create block for the specified <MODULE>.

Example:

```
vtiger_Block::create('Leads', 'LBL_VTLIB_INFORMATION');
```

NOTE: You will need to put the <BLOCK_LABEL> mapping in the module language file (modules/<MODULE>/language/en_us.lang.php) to see the affect.

For the above example, add the mapping in modules/Leads/language/en_us.lang.php

```
$mod_strings = Array (  
'LBL_VTLIB_INFORMATION' => 'vtlib Information',  
...  
);
```

NOTE2: The block will not be displayed until we associate one field to it.

Creating Field

```
include_once('vtlib/Vtiger/Field.php');
$field1 = new Vtiger_Field();

// Populate Field Information
$field1->set('module',      '<MODULE>')
    -> set('columnname',    '<TABLE COLUMN NAME>')
    -> set('tablename',     '<TABLE NAME>')
    -> set('columnname',    '<TABLE COLUMN TYPE>')
    -> set('generatedtype', '<STANDARD (=1) OR CUSTOM (=2) TYPE>')
    -> set('uitype',        '<UI TYPE>')
    -> set('fieldname',     '<FIELD NAME>')
    -> set('fieldlabel',    '<FIELD LABEL ON UI>') // Need language file entry
    -> set('readonly',     '<READONLY (=0) orelse (1)>')
    -> set('presence',     '0')
    -> set('selected',      '0')
    -> set('maxlength',    '<MAXIMUM DATA LENGTH (=100)>')
    -> set('sequence',     null) // will be set dynamically.
    -> set('typeofdata',   '<TYPE OF DATA>')
    -> set('quickcreate',   '<AVAILABLE IN QUICK CREATE FORM(=0) orelse(=1)>')
    -> set('block',        null) // Set blockid and make blocklabel null
    -> set('blocklabel',   '<BLOCK LABEL to which field needs to attached>')
    -> set('displaytype',   '<DISPLAY TYPE>')
    -> set('quickcreatesequence', null) // will be set dynamically
    -> set('info_type',    '<TYPE OF INFORMATION (BAS=Basic,ADV=Advanced)>');

// Create Field
$field1->create();
```

The above API will let you create vtiger field and associate it to the module block. You will have to provide all the information described above.

NOTE: During vtiger field creation, the columnname existence is checked in the tablename specified. If it does not exist then tablename is altered and columnname (column type) is created.

You can create table before creating field & provide the tablename, columnname to use this table.

```
include_once('vtlib/Vtiger/Utils.php');
Vtiger_Utils::CreateTable('<TABLENAME>', '(<COLUMNNAME COLUMNTYPE>');
```

This API will let us create table with tablename and given column conditions.

Example: To create a new field and associate to custom block in Leads module.

```
include_once('vtlib/Vtiger/Field.php');
$field1 = new Vtiger_Field();
$field1-> set('module',    'Leads')
    -> set('columnname',    'vtlib_leads') // New column
    -> set('tablename',     'vtiger_leaddetails') // Existing table
    -> set('columnname',    'varchar(255)')
    -> set('generatedtype', '1')
    -> set('uitype',        2)
    -> set('fieldname',     'vtlibldname')
    -> set('fieldlabel',    'vtlibLeadName')
    // Entry needed in modules/Leads/language/en_us.lang.php
    -> set('readonly',     '1')
```

```
-> set('presence', '0')
-> set('selected', '0')
-> set('maxlength', '100')
-> set('sequence', null)
-> set('typeofdata', 'V~M')
-> set('quickcreate', '1')
-> set('block', null)
-> set('blocklabel', 'LBL_VTLIB_INFORMATION')
-> set('displaytype', '1')
-> set('quickcreatesequence', null)
-> set('info_type', 'BAS');
```

```
$field1->create();
```

After executing this script, make sure to check the mapping in `modules/Leads/language/en_us.lang.php` for fieldlabel (vtlibLeadName) used above.

If you have associated the field with a new table then you will need to update:
modules/Leads/Leads.php \$tab_name and \$tab_name_index Arrays

```
var $tab_name = Array( ..., '<NEW_TABLENAME>');
var $tab_name_index = Array( ..., '<NEW_TABLENAME>' => '<ID_COLUMN>');
```


Creating Custom View

```
include_once('vtlib/Vtiger/CustomView.php');  
Vtiger_CustomView::create('<MODULE>', '<FILTER NAME>');
```

This API will let you create custom view (filter) for the required module. Once the filter is created you need to associate module fields to it. The module field instance need to be populated as explained in previous section.

```
$cv = new Vtiger_CustomView('<MODULE>', '<FILTER NAME>');  
$cv->addColumn($Vtiger_Field_Instance);
```

Example:

Create custom view for Leads module and associating the field created in last section.

```
include_once('vtlib/Vtiger/CustomView.php');  
include_once('vtlib/Vtiger/Field.php');  
  
Vtiger_CustomView::create('Leads', 'vtlibFilter');  
  
$cv = new Vtiger_CustomView('Leads', 'vtlibFilter');  
$field1-> set('module', 'Leads')  
-> set('columnname', 'vtlib_leads')  
-> set('tablename', 'vtiger_leaddetails')  
-> set('fieldname', 'vtlibldname')  
-> set('fieldlabel', 'vtlibLeadName')  
-> set('typeofdata', 'V~M');  
  
$cv->addColumn($field1);
```

Enable/Disable Action

Import, Export and Merge are treated as Actions for a module. To enable or disable them, you can use the API given below.

```
include_once('vtlib/Vtiger/Module.php');  
Vtiger_Module::disableAction('<MODULE>', '<Action>');  
Vtiger_Module::enableAction('<MODULE>', '<Action>');
```

Example:

To Disable Import and Enable Export for Leads module.

```
Vtiger_Module::disableAction('Leads', 'Import');  
Vtiger_Module::enableAction('Leads', 'Export');
```

NOTE: This will enable or disable the action (tool) for all the existing profiles.

Sharing Access Setting

```
include_once('vtlib/Vtiger/Module.php');  
Vtiger_Module::setDefaultSharingAccess('<MODULE>', '<PERMISSION_TYPE>');
```

Use this API to set default sharing access for the <MODULE>.

The <PERMISSION_TYPE> can be one of the following:

Public_ReadOnly
Public_ReadWrite
Public_ReadWriteDelete
Private

Example:

```
Vtiger_Module::setDefaultSharingAccess('Leads', 'Private');
```

Custom Fields Setup

To enable Custom Fields for the module you will need to follow the steps given below:

1. The module should have the block labeled LBL_CUSTOM_INFORMATION
Look at the section [Creating Block](#)
2. Create table with name vtiger_<modulename>cf (should have primary key column) to save the custom field data.

Example: If your module name is Payslip, then custom field table to be created should be named vtiger_payslipcf with a primary key column integer type in it.

```
include_once('vtlib/vtiger/Module.php');  
vtiger_Utils::CreateTable('vtiger_payslipcf', '(payslipid integer, primary  
key (payslipid))');
```

3. Update \$tab_name and \$tab_name_index Array in the module class with the custom field table information.

```
var $tab_name = Array('vtiger_crmentity', 'vtiger_payslip',  
    'vtiger_payslipcf');  
  
var $tab_name_index = Array(  
    'vtiger_crmentity' => 'crmid',  
    'vtiger_payslip'   => 'payslipid',  
    'vtiger_payslipcf' => 'payslipid');
```

4. Update \$customFieldTable variable entry in the module class with custom field tablename and the primary key column name.

```
var $customFieldTable = Array('vtiger_payslipcf', 'payslipid');
```

Creating New Module

This can be broken down into two steps:

1. Database setup
2. Module directory setup

Database setup

Using the above API's explained above you will be able to create new modules for vtiger. The following are important steps that should be followed to get basic working module.

- a) Create tab (entry point for module on UI)
- b) Create required module table, module group table.
- c) Create blocks for the module.
- d) Create fields and associate it to blocks.
Set atleast one of the field as entity identifier.
- e) Create custom view with required fields (make sure to create filter name All which is treated as special default filter)

NOTE: Please look at `vtlib.Test.Create.Module1.php` file to understand the steps explained above.

Module directory setup

vtlib directory contains skeleton ModuleDir which can be used as a template for new module that is created. It contains source files that needs to be changed accordingly.

1. Rename ModuleDir to `<NewModuleName>`
2. Rename ModuleDir/ModuleFile.php to `<NewModuleName>.php`
3. Rename ModuleDir/ModuleFileAjax.php to `<NewModuleName>Ajax.php`
4. Edit `<NewModuleName>.php`
 - a) Rename Class ModuleClass to `<NewModuleName>`
 - b) Update `$table_name` and `$table_index` (Module table name and table index column)
 - c) Update `$groupTable`
 - d) Update `$tab_name`, `$tab_name_index`
 - e) Update `$list_fields`, `$list_fields_name`, `$sortby_fields`
 - f) Update `$detailview_links`
 - g) Update `$default_order_by`, `$default_sort_order`
 - h) Update `$required_fields`
 - i) Update `$customFieldTable`
 - j) Rename function ModuleClass to function `<NewModuleName>` [This is the Constructor Class]

Creating Payslip Module

When Create Payslip Module link is clicked from <http://localhost/vtigercrm/vtlib.Test.html> you will see the following:

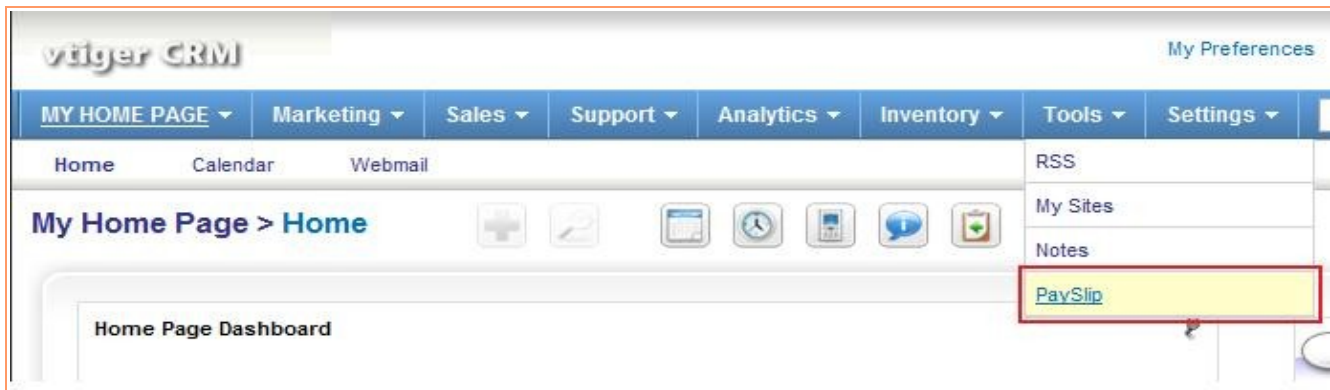
```
Creating new item (30) under Tools ... DONE
Creating new menu item Payslip (Payslip) ... DONE
Check Language Mapping entry for Payslip ... TODO
Creating block Payslip (LBL_PAYSLIP_INFORMATION) ... DONE
Check Module Language Mapping entry for LBL_PAYSLIP_INFORMATION ... TODO
Adding PayslipName to Payslip ... DONE
Check Module Language Mapping entry for PayslipName ... TODO
Adding Month to Payslip ... DONE
Check Module Language Mapping entry for Month ... TODO
Adding Assigned To to Payslip ... DONE
Check Module Language Mapping entry for Assigned To ... TODO
Adding Created Time to Payslip ... DONE
Check Module Language Mapping entry for Created Time ... TODO
Adding Modified Time to Payslip ... DONE
Check Module Language Mapping entry for Modified Time ... TODO
Creating CustomView(All) for Payslip ... DONE
Adding PayslipName to Payslip CustomView(38) ... DONE
Adding Month to Payslip CustomView(38) ... DONE
Adding Assigned To to Payslip CustomView(38) ... DONE
Adding Modified Time to Payslip CustomView(38) ... DONE
Creating CustomView(All2) for Payslip ... DONE
Adding PayslipName to Payslip CustomView(39) ... DONE
Adding Assigned To to Payslip CustomView(39) ... DONE
Adding Modified Time to Payslip CustomView(39) ... DONE
Disabling Action Import for module Payslip for Profile 1 ... DONE
Disabling Action Import for module Payslip for Profile 2 ... DONE
Disabling Action Import for module Payslip for Profile 3 ... DONE
Disabling Action Import for module Payslip for Profile 4 ... DONE
Enabling Action Export for module Payslip for Profile 1 ... DONE
Enabling Action Export for module Payslip for Profile 2 ... DONE
Enabling Action Export for module Payslip for Profile 3 ... DONE
Enabling Action Export for module Payslip for Profile 4 ... DONE
```

NOTE: Kindly edit include/language/en_us.lang.php and update \$app_strings as shown below:

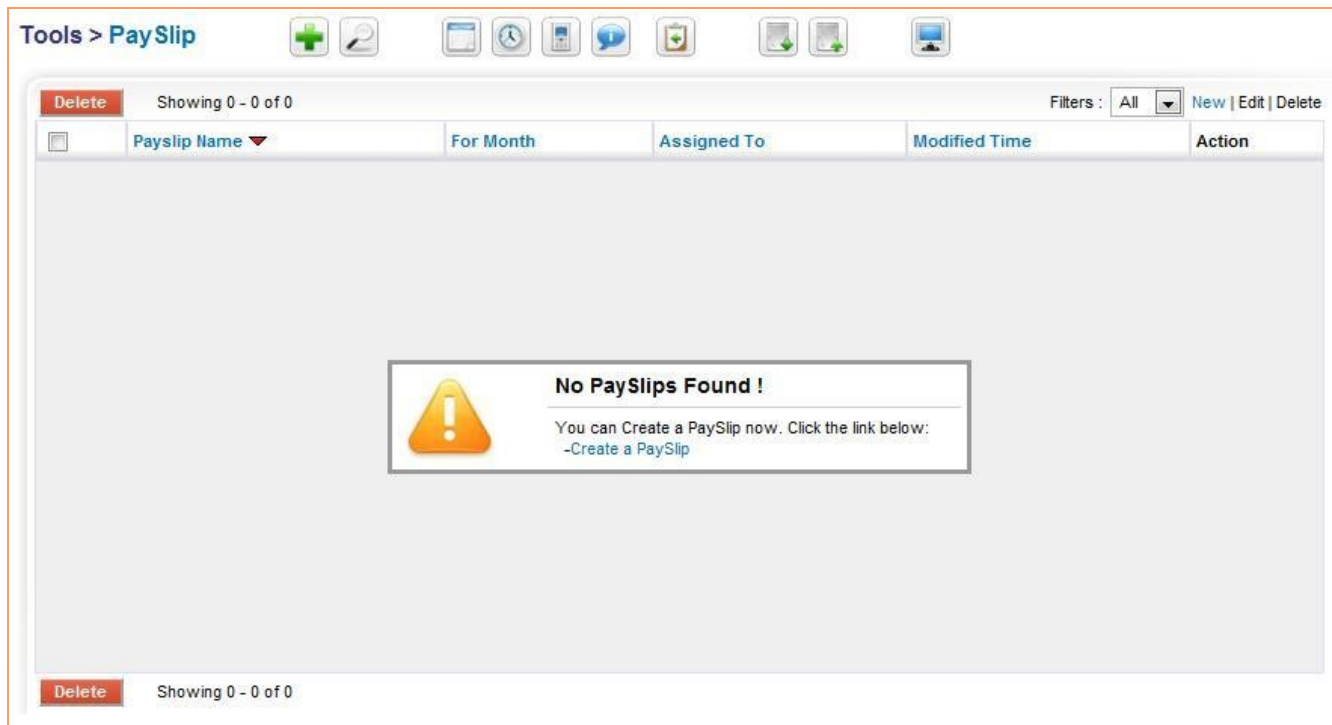
```
$app_strings = Array (
'Payslip' => 'Payslip'
...
```

Module Entry Point

The module link appears under Tools Menu.



Default List view



Create view

Tools > PaySlip

Basic Information

Save Cancel

Payslip Information

*Payslip Name	<input type="text" value="Test Payslip"/>	For Month	<input type="text" value="2008-08-01"/> <small>(yyyy-mm-dd)</small>
Assigned To	<input checked="" type="radio"/> User <input type="radio"/> Group <input type="text" value="admin"/>		

Save Cancel

Detail view

Tools > PaySlip

[132] Test Payslip - PaySlip Information
Updated today (06 Aug 2008)

PaySlip Information

Edit Duplicate Delete

PaySlip Information

Payslip Name	Test Payslip	For Month	2008-08-01
Assigned To User	admin	Created On	2008-08-06 14:54:03
Modified On	2008-08-06 14:54:03		

TAG CLOUD

Tag it

List view

Tools > PaySlip

Delete Showing 1 - 1 of 1

Filters : All2 New | Edit | Delete

Payslip Name	Modified Time	Assigned To	Action
<input type="checkbox"/> Test Payslip	2008-08-06 14:54:03	admin	edit del


Delete Showing 1 - 1 of 1

Export – Import Tool

Tools > PaySlip

Showing 0 - 0 of 0

Filters: All New | Edit | Delete

<input type="checkbox"/>	PaySlip Name ▼	For Month	Assigned To	Modified Time	Action
<div>No PaySlips Found !<p>You can Create a PaySlip now. Click the link below: -Create a PaySlip</p></div>					

Showing 0 - 0 of 0

Sharing Access Setting

Settings > Sharing Access


Manage module sharing rules & custom sharing rules

1. Organization-level Sharing Rules

Recalculate Change Privileges

Potentials	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Potentials
Accounts & Contacts	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Accounts & Contacts
Leads	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Leads
Calendar	★ Private	Users cannot access other users Calendar
Trouble Tickets	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Trouble Tickets
Quotes	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Quotes
Purchase Order	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Purchase Order
Sales Order	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Sales Order
Invoice	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Invoice
Campaigns	★ Public: Read, Create/Edit, Delete	Users can Read, Create/Edit, Delete other users Campaigns
PaySlip	★ Private	Users cannot access other users PaySlip

Custom Field Addition





Settings > Custom Field Settings

Manage your company-wide custom fields.

Custom Fields in "PaySlip" Module

Select Module: PaySlip

New Custom Field

#	Field Label	Field Type	Tools
1	Site	URL	 

[Scroll to Top]