vtiger

vtlib – vtiger development library

version 2.0

DISCLAIMER:

The vtlib library development is in progress and subject to change. While we make every effort to make sure modules developed using vtlib will be compatible with future versions of vtiger CRM, some incompatible changes may be required for the next release of vtiger. In which case you will have to re-create your modules with an upgraded version of vtlib for the specific version of vtiger CRM.

Table of Contents

API Version History

Version	Release Date	Highlights
2.0	2008-11-26	* API provided to related modules (related list) * API changes for creating fields, blocks, module * UI type 10 was added for generic popup field
1.4	2008-09-29	* Exporting and Importing was added to Module Manager
1.3	2008-09-01	* Module Manager was added for Administrators to install and enable/disable the new modules developed using vtlib. * API to setup Picklist values
1.2	2008-08-29	* Sharing Access API was added
1.1	2008-08-27	* API's to enable and disable tools like Export, Import were added
1.0	2008-08-19	* Basic API was added to created fields, blocks, module

About vtlib

vtlib is a library to ease new module development for vtiger CRM. vtlib includes APIs to create or modify the backend elements for a module. These APIs help make the necessary changes to the database.

vtlib includes Module Manager which allows new modules to be packaged into zip files that other vtiger CRM installations can easily install and use.

For vtiger 5.0.4, vtlib is available as a download. This package needs to be installed by developers as well as users who intend to install modules developed using vtlib.

vtlib will be integrated in vtiger CRM version 5.1 download.

vtlib Installation

vtlib Installation requires the following steps.

1. vtiger 5.0.4 should be installed.
2. Take backup of your vtiger CRM Source directory.
3. Unpack the vtlib-x.y.zip into your vtiger CRM source directory.

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

vtlib API - Quick Reference

vtlib includes the following APIs that can be used to create new modules. For more details please look at the API docs.

- Vtiger_Module
  - name
  - addBlock()
  - addFilter()
  - initTables()
  - setRelatedList()
  - setDefaultSharing()
  - enableTools()
  - disableTools()
  - save()
- Vtiger_Menu
  - addModule()
- Vtiger_Block
  - label
  - addField()
- Vtiger_Field
  - table
  - column
  - columntype
  - uitype
  - typeofdata
  - setEntityIdentifier()
  - setPicklistValues()
  - setRelatedModules()
- Vtiger_Filter
  - name
  - isdefault
  - addField()
  - addRule()
- Vtiger_Event
  - register()

Creating a new Module

vtlib simplifies creation of new vtiger CRM modules. Developers can use vtlib to develop vtiger CRM modules that add new functionality to vtiger CRM. These modules can then be packaged for easy installation by the Module Manager.

NOTE: In this document we will explain the process of creating a new module by building an example 'Payslip' Module. This example code is included as part of vtlib package, and can be used as a starting point to create new modules. Please refer to the 'Using the example code provided with the vtlib API' section in this document for more information.

The following are important steps that should be followed to get basic working module. The backend involves database level changes for the module, and the frontend covers the UI files.

Backend

Step 1	Create module instance, create database tables, and add it to Menu
Step 2	Add UI blocks for the module.
Step 3	Add fields and associate it to blocks. Set at-least one of the field as entity identifier.
Step 4	Create default list view and additional filters (make sure to create filter name All which is default filter)
Step 5	Create Related List (to show in the ''More information'' tab)
Step 6	Setting Sharing Access Rules
Step 7	Setting Module Tools options (i.e., Import/Export)

FrontEnd

Step 8	Creating Module directory and files

Packaging

Step 9	Packaging

Additional Options

- ➔ Module Templates (to customize Form, List View, and Settings UI)
- ➔ Module Settings (to allow administrators to configure your module)
- ➔ Module Events (only available in vtiger CRM version 5.1)

These steps are explained in detail in the course of this section.

Step 1: Creating Module

Class Vtiger_Module provides an API to work with vtiger CRM modules.
The example given below describes the way of creating module 'Payslip' and associating with Tools menu.

```
include_once('vtlib/Vtiger/Module.php');

$moduleInstance = new Vtiger_Module();
$moduleInstance->name = 'Payslip';
$moduleInstance->save();

$moduleInstance->initTables();

$menuInstance = Vtiger_Menu::getInstance('Tools');
$menuInstance->addModule($moduleInstance);
```

`Vtiger_Module->initTables()` API will initialize (create) the 3 necessary tables a module should have as explained below:

Table	Naming convention	Description
Basetable	vtiger_<MODULENAME>	Used to map module field column by default
Customtable	vtiger_<MODULENAME>cf	Used for mapping custom fields of the module
Grouptable	vtiger_<MODULENAME>grouprel	Used when records are assigned to the group

`Vtiger_Menu->addModule(<ModuleInstance>)` API will create menu item which serves as UI entry point for the module.

Step 2: Creating Block (in UI Form)

Class Vtiger_Block provides API to work with Module block, the container which holds the fields together.

The example given below describes the way of creating new blocks for the module created earlier:

```
include_once('vtlib/Vtiger/Module.php');

$blockInstance = new Vtiger_Block();
$blockInstance->label = 'LBL_PAYSLIP_INFORMATION';
$moduleInstance->addBlock($blockInstance);

$blockInstance2 = new Vtiger_Block();
$blockInstance2->label = 'LBL_CUSTOM_INFORMATION';
$moduleInstance->addBlock($blockInstance2);
```

NOTE: LBL_CUSTOM_INFORMATION block should always be created to support Custom Fields for a module.

Step 3: Adding Fields

Class Vtiger_Field provides API to work with Module field, which are the basic elements that store and display the module record data.

The example given below describes the way of creating new field for the module created earlier:

```
include_once('vtlib/Vtiger/Module.php');

$fieldInstance = new Vtiger_Field();
$fieldInstance->name = 'PayslipName';
$fieldInstance->table = 'vtiger_payslip';
$fieldInstance->column = 'payslipname';
$fieldInstance->columntype = 'VARCHAR(100)';
$fieldInstance->uitype = 2;
$fieldInstance->typeofdata = 'V~M';
$blockInstance->addField($fieldInstance);
```

NOTE: The fieldInstance name is a mandatory value to be set before saving / adding to block.
Other values (if not set) are defaulted as explained below:

$fieldInstance->table	Module's basetable
$fieldInstance->column	$fieldInstance->name in lowercase [The table will be altered by adding the column if not present]
$fieldInstance->columntype	VARCHAR(255)
$fieldInstance->uitype	1
$fieldInstance->typeofdata	V~O
$fieldInstance->label	$fieldInstance->name [Mapping entry should be present in module language file as well]

Entity Identifier

One of the mandatory field should be set as entity identifier of module once it is created. This will be used for showing the details in 'Last Viewed Entries' etc...

```
$moduleInstance->setEntityIdentifier($fieldInstance);
```

Set Picklist Values

If the field is of Picklist type (uitype **15**, 16, 33, 55, 111) then you can configure the initial values using the following API:

```
$fieldInstance->setPicklistValues( Array ('Value1', 'Value2') );
```

Set Related Module

If the field is of Popup select type (uitype=10), you can configure the related modules which could be selected via Popup using the following API:

```
$fieldInstance->setRelatedModules(Array('OtherModule1', 'OtherModule2'));
```

To unset the related module you can use the following API:

```
$fieldInstance->setRelatedModules(Array('OtherModule2'));
```

Step 4: Creating Filters

Class Vtiger_Filter provides API to work with Module's custom view or filter. The list view display is controlled via these filters setup.

The example given below describes the way of creating new filter for the module:

```
include_once('vtlib/Vtiger/Module.php');

$filterInstance = new Vtiger_Filter();
$filterInstance->name = 'All';
$filterInstance->isdefault = true;
$moduleInstance->addFilter($filterInstance);
```

Configure fields

To add fields to the filter you can use the following API:

```
$filterInstance->addField($fieldInstance, $columnIndex);
```

Where $columnIndex (optional) is the order/index at which the field should appear in the list view.

Setup Rules

Once the field is added to filter you can setup rule (condition) for filtering as well using the following API:

```
$filterInstance->addRule($fieldInstance, $comparator, $compareValue,
$columnIndex);
```

Where comparator could be one of the following:

EQUALS
NOT_EQUALS
STARTS_WITH
ENDS_WITH
CONTAINS
DOES_NOT_CONTAINS
LESS_THAN
GREATER_THAN
LESS_OR_EQUAL
GREATER_OR_EQUAL

$compareValue is the value against with the field needs to be compared.

$columnIndex (optional) is the order at which this rule condition should be applied.

Step 5: Related Lists

One module could be associated with multiple records of other module that is displayed under "More Information" tab on Detail View.

The example given below describes the way of creating relation between a Payslip and Accounts module:

```
include_once('vtlib/Vtiger/Module.php');

$moduleInstance = Vtiger_Module::getInstance('Payslip');

$accountsModule = Vtiger_Module::getInstance('Accounts');

$relationLabel  = 'Accounts';

$moduleInstance->setRelatedList(
      $accountsModule, $relationLabel, Array('ADD','SELECT')
);
```

With this you can Add one or more Accounts to Payslip records.

To drop the relation between the modules use the following:

```
$moduleInstance->unsetRelatedList($targetModuleInstance);
```

About setRelatedList API

```
Vtiger_Module->setRelatedList(<TARGET MODULE>[, <HEADER LABEL>, <ALLOWED ACTIONS>,
<CALLBACK FUNCTION NAME>]);
```

<TARGET MODULE>	Module name to which relation is being setup.
<HEADER LABEL>	Optional (default = <TARGET MODULE>) Label to use on the More Information related list view.
<ALLOWED ACTIONS>	Optional ADD or SELECT (default = false) What buttons should be shown in the related list view while adding records.
<CALLBACK FUNCTION NAME>	Optional (default = get_related_list) The function should be defined in the <SOURCE MODULE> class. This should generate the listview entries for displaying.

NOTE:

This API will create vtiger_crmentityrel table to keep track of relation between module records.
Standard modules available in vtiger CRM handles the relation in separate tables and performs the JOIN to fetch data specific to each module.

This is an attempt to achieve generic behavior. You can write custom call back functions to handle related list queries that will meet your requirements.

Limitations
Following limitations apply for the related list APIs

1. Standard module class variables are not set as required by the get_related_list vtlib module API. Case handling should be handled @function vtlib_setup_modulevars in include/utils/VtlibUtils.php

2. get_related_list API added to module class does not handle JOIN
on tables where some modules like (Accounts) store information hence complete details are not fetched in the Related List View.
(Example Sorting on the city field on related list view will fail if dieOnError is true)

3. get_related_list API will provide only adding new records in the related list view, we need to provide other API which can provide SELECT button to populated related records from more information.

Step 6: Sharing Access Rules

Sharing access configuration for the module can be done as shown below:

The example given below describes the way configuring Payslip module as Private

```
include_once('vtlib/Vtiger/Module.php');

$moduleInstance = Vtiger_Module::getInstance('Payslip');

$moduleInstance->setDefaultSharing('Private');
```

The <PERMISSION_TYPE> can be one of the following:

Public_ReadOnly
Public_ReadWrite
Public_ReadWriteDelete
Private

Step 7: Module Tools

Features like Import, Export are termed as module tools. Such tools can enabled or disabled as shown below:

The example given below describes the way enabling and disabling the tools for Payslip module

```
include_once('vtlib/Vtiger/Module.php');

$moduleInstance = Vtiger_Module::getInstance('Payslip');

$module->enableTools(Array('Import', 'Export'));

$module->disableTools('Export');
```

Optional Step: Module Events

Eventing API is supported from vtiger 5.1 onwards (read more here).

To check if your vtiger CRM supports Eventing use the following:

```
include_once('vtlib/Vtiger/Event.php');
boolean Vtiger_Event::isSupported();
```

To register an event for a module, use the following:

```
include_once('vtlib/Vtiger/Event.php');
Vtiger_Event::register('<MODULENAME>', '<EVENTNAME>',
                       '<HANDLERCLASS>', '<HANDLERFILE>');
```

<MODULNAME>	Module for which events should be registered
<EVENTNAME>	vtiger.entity.aftersave vtiger.entity.beforesave
<HANDLERCLASS>	Event handler class, look at the example below
<HANDLERFILE>	File where HANDLERCLASS is defined (should be within vtiger CRM directory)

Example: Registering event callback before and after save.

```
if(Vtiger_Event::hasSupport()) {
      Vtiger_Event::register(
            'Payslip', 'vtiger.entity.aftersave',
            'PayslipHandler', 'modules/Payslip/PayslipHandler.php'
      );

      Vtiger_Event::register(
            'Payslip', 'vtiger.entity.beforesave',
            'PayslipHandler', 'modules/Payslip/PayslipHandler.php'
      );
}
```

modules/Payslip/PayslipHandler.php

```
<?php

class PayslipHandler extends VTEventHandler {

      function handleEvent($eventName, $data) {

            if($eventName == 'vtiger.entity.beforesave') {
                  // Entity is about to be saved, take required action
            }

            if($eventName == 'vtiger.entity.aftersave') {
                  // Entity has been saved, take next action
            }
      }
}

?>
```

Optioal Step: Module Templates

If you would like to customize the list view or have custom Settings page for the module, then you will need to create a Smarty template accordingly. You will need to have some knowledge of Smarty templates usage before proceeding.

Your module specific Smarty template files should be created under `Smarty/templates/modules/<NewModuleName>`.

Use `vtlib_getModuleTemplate($module, $templateName)` API (include/utils/VtlibUtils.php) as:

```
$smarty->display(vtlib_getModuleTemplate($currentModule, 'MyListview.tpl'));
```

Final Completed Script (Backend)

Given below is complete script (vtlib.Test.Create.Module1.php) which creates Payslip module

```
<?php
// Turn on debugging level
$Vtiger_Utils_Log = true;

include_once('vtlib/Vtiger/Menu.php');
include_once('vtlib/Vtiger/Module.php');

// Create module instance and save it first
$module = new Vtiger_Module();
$module->name = 'Payslip';
$module->save();

// Initialize all the tables required
$module->initTables();
/**
 * Creates the following table:
 * vtiger_payslip  (payslipid INTEGER)
 * vtiger_payslipcf(payslipid INTEGER PRIMARY KEY)
 * vtiger_payslipgrouprel((payslipid INTEGER PRIMARY KEY, groupname VARCHAR(100))
 */

// Add the module to the Menu (entry point from UI)
$menu = Vtiger_Menu::getInstance('Tools');
$menu->addModule($module);

// Add the basic module block
$block1 = new Vtiger_Block();
$block1->label = 'LBL_PAYSLIP_INFORMATION';
$module->addBlock($block1);

// Add custom block (required to support Custom Fields)
$block2 = new Vtiger_Block();
$block2->label = 'LBL_CUSTOM_INFORMATION';
$module->addBlock($block2);

/** Create required fields and add to the block */
$field1 = new Vtiger_Field();
$field1->name = 'PayslipName';
$field1->table = $module->basetable;
$field1->column = 'payslipname';
$field1->columntype = 'VARCHAR(255)';
$field1->uitype = 2;
$field1->typeofdata = 'V~M';
$block1->addField($field1); /** Creates the field and adds to block */

// Set at-least one field to identifier of module record
$module->setEntityIdentifier($field1);

$field2 = new Vtiger_Field();
$field2->name = 'PayslipType';
$field2->label = 'Payslip Type';
$field2->columntype = 'VARCHAR(100)';
$field2->uitype = 15;
$field2->typeofdata = 'V~O';// Varchar~Optional
$block1->addField($field2); /** table and column are automatically set */

$field2->setPicklistValues( Array ('Employee', 'Trainee') );

$field3 = new Vtiger_Field();
$field3->name = 'Month';
```

```
$field3->uitype = 23;
$field3->typeofdata = 'D~M'; // Date~Mandatory
$block1->addField($field3);  /** table, column, label, set to default values */

$field4 = new Vtiger_Field();
$field4->name = 'LinkTo';
$field4->label= 'Link To';
$field4->table = 'vtiger_payslip';
$field4->column = 'linkto';
$field4->columntype = 'VARCHAR(100)';
$field4->uitype = 10;
$field4->typeofdata = 'V~O';
$block1->addField($field4);
$field4->setRelatedModules(Array('Contacts'));

/** Common fields that should be in every module, linked to vtiger CRM core table
*/
$field5 = new Vtiger_Field();
$field5->name = 'assigned_user_id';
$field5->label = 'Assigned To';
$field5->table = 'vtiger_crmentity';
$field5->column = 'smownerid';
$field5->uitype = 53;
$field5->typeofdata = 'V~M';
$block1->addField($field5);

$field6 = new Vtiger_Field();
$field6->name = 'CreatedTime';
$field6->label= 'Created Time';
$field6->table = 'vtiger_crmentity';
$field6->column = 'createdtime';
$field6->uitype = 70;
$field6->typeofdata = 'T~O';
$field6->displaytype= 2;
$block1->addField($field6);

$field7 = new Vtiger_Field();
$field7->name = 'ModifiedTime';
$field7->label= 'Modified Time';
$field7->table = 'vtiger_crmentity';
$field7->column = 'modifiedtime';
$field7->uitype = 70;
$field7->typeofdata = 'T~O';
$field7->displaytype= 2;
$block1->addField($field7);
/** END */

// Create default custom filter (mandatory)
$filter1 = new Vtiger_Filter();
$filter1->name = 'All';
$filter1->isdefault = true;
$module->addFilter($filter1);

// Add fields to the filter created
$filter1->addField($field1)->addField($field2, 1)->addField($field5, 2);

// Create one more filter
$filter2 = new Vtiger_Filter();
$filter2->name = 'All2';
$module->addFilter($filter2);

// Add fields to the filter
$filter2->addField($field1);
$filter2->addField($field2, 1);
```

```
// Add rule to the filter field
$filter2->addRule($field1, 'CONTAINS', 'Test');

/** Associate other modules to this module */
$module->setRelatedList(Vtiger_Module::getInstance('Accounts'), 'Accounts',
Array('ADD','SELECT'));

/** Set sharing access of this module */
$module->setDefaultSharing('Private');

/** Enable and Disable available tools */
$module->enableTools(Array('Import', 'Export'));
$module->disableTools('Merge');

?>
```

Step 8: Creating module files (Frontend)

Each new module should have a directory under modules/ folder. To help speed up the module code creation, vtlib comes bundled with skeleton module structure based on the 'PaySlip' module. This code is include in vtlib/ModuleDir folder which can be used as a template for new module that is created. It contains source files that needs to be changed as explained below.

1. Copy ModuleDir contents to newly created modules/<NewModuleName> folder.
2. Rename <NewModuleName>/ModuleFile.php as <NewModuleName>/<NewModuleName>.php (as noted in the table below)
3. Rename <NewModuleName>/ModuleFileAjax.php as <NewModuleName>/<NewModuleName>Ajax.php
4. Rename <NewModuleName>/ModuleFile.js to <NewModuleName>/<NewModuleName>.js
5. Edit <NewModuleName>/<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) Update $def_detailview_recname [*Column name whose value will be used to display DetailView text for a record*]
   k) Rename function ModuleClass to function <NewModuleName> [This is the Constructor Class]

NOTE: Other files under modules/<NewModuleName> need not be changed.

Example ModuleDir	**Purpose**	**File under Payslip**
index.php	Module entry point through Menu	index.php
ModuleFile.php	Module class definition file.	Payslip.php
ModuleFileAjax.php	Base file for ajax actions used under Listview etc...	PayslipAjax.php
ModuleFile.js	Module specific javascript function can be written here	Payslip.js
CallRelatedList.php	More Information Detail view handler	CallRelatedList.php
CustomView.php	Custom view or Filter handler	CustomView.php
Delete.php	Module record deletion handler	Delete.php
DetailView.php	Detail view handler	DetailView.php
DetailViewAjax.php	Detail view ajax edit handler	DetailViewAjax.php
EditView.php	Edit view handler	EditView.php
ExportRecords.php	Module record export handler	ExportRecords.php
Import.php	Module records import handler	Import.php
ListView.php	List view handler	ListView.php
Popup.php	Popup selection handler for this module record	Popup.php
QuickCreate.php	Quick creation handler	QuickCreate.php
Save.php	Module record save handler	Save.php
TagCloud.php	Tag cloud handler	TagCloud.php
updateRelations.php	Related list record handler (save/delete)	updateRelations.php

Step 9: Packaging

Package Export

vtlib provides API to export module as a zip (package) file which can used for importing through Module Manger.

```
require_once('vtlib/Vtiger/Package.php');
require_once('vtlib/Vtiger/Module.php');
$package = new Vtiger_Package();
$package->export('<MODULE Instance>', '<DESTINATION DIR>', '<ZIPFILE NAME>', <DIRECT
DOWNLOAD>);
```

<MODULE Instance>	Vtiger_Module instance to be exported (packaged)
<DESTINATION DIR>	(Optional: Default=test/vtlib) Directory where the zipfile output should be created.
<ZIPFILE NAME>	(Optional: Default=modulename-timestamp.zip) Zipfile name to use for the output file.
<DIRECT DOWNLOAD>	(Optional: Default=false) If true, the zipfile created will be streamed for download and zipfile will be deleted after that.

Example:

```
require_once('vtlib/Vtiger/Package.php');
require_once('vtlib/Vtiger/Module.php');
$package = new Vtiger_Package();
$package->export(
      Vtiger_Module::getInstance('Payslip'),
      'test/vtlib',
      'Payslip-Export.zip',
      true
);
```

NOTE: Please make sure test/vtlib directory exists under vtigercrm root directory and is writeable.

Package Structure

The exported zipfile (package) has the following structure:

```
manifest.xml
modules/
        ModuleName/
                <Module Related Files>
                language/
                        en_us.lang.php
                        <Other language Files>

templates/
       <Smarty templates of the Module>
```

`manifest.xml` has the meta information that will be useful during the import process as shown:

```
<?xml version="1.0"?>
<module>
  <exporttime>YYYY-MM-DD hh:mm:ss</exporttime>
  <name>MODULE NAME</name>
  <label>MODULE LABEL</label>
  <parent>MENU</parent>

  <dependencies>
      <vtiger_version>VTIGER_VERSION_NUMBER</vtiger_version>
  </dependencies>

  <tables>
    <table>
      <name>TABLENAME</name>
      <sql>TABLE SQL</sql>
    </table>
  </tables>

  <blocks>
    <block>
      <label>BLOCK LABEL</label>
      <fields>
        <field>
          <fieldname>payslipname</fieldname>
          <columnname>payslipname</columnname>
          <uitype>UI TYPE</uitype>
          <tablename>TABLE NAME</tablename>
          <generatedtype>GEN TYPE</generatedtype>
          <fieldlabel>FIELD LABEL</fieldlabel>
          <readonly>READONLY</readonly>
          <presence>PRESENCE</presence>
          <selected>SELECTED</selected>
          <maximumlength>MAXLEN</maximumlength>
          <typeofdata>TYPEOFDATA</typeofdata>
          <quickcreate>QUICKCREATE</quickcreate>
          <displaytype>DISPTYPE</displaytype>
          <info_type>INFOTYPE</info_type>
        </field>
      </fields>
    </block>
  </blocks>

  <customviews>
    <customview>
      <viewname>VIEWNAME</viewname>
      <setdefault>0</setdefault>
      <setmetrics>1</setmetrics>
      <fields>
        <field>
          <fieldname>FIELDNAME</fieldname>
          <columnindex>0</columnindex>
         </field>
      </fields>
    </customview>
  </customviews>
  <sharingaccess>
    <default>private</default>
  </sharingaccess>

  <actions>
    <action>
      <name>Export</name>
      <status>enabled</status>
    </action>
    <action>
      <name>Import</name>
      <status>enabled</status>
    </action>
  </actions>
</module>
```

Package Import

You can import module from package (zip) file using the following API

```
require_once('vtlib/Vtiger/Package.php');

$package = new Vtiger_Package();
$package->import(<MODULE ZIPFILE>, <overwrite>);
```

<MODULE ZIPFILE>	Module zipfile.
<overwrite>	(Optional: Default=false) Overwrite the existing module directory if present

The package file should be validated before Importing which can be using the following API

```
require_once('vtlib/Vtiger/Package.php');
$package = new Vtiger_Package();
$package->checkZip(<MODULE ZIPFILE>);
```

<MODULE ZIPFILE>	Module zipfile.

`checkZip` returns **true** if package structure in zipfile is as specified in the section **Package Export** above.

Detecting Module Name being Imported

```
require_once('vtlib/Vtiger/Package.php');
$package = new Vtiger_Package();
$package->getModuleNameFromZip(<MODULE ZIPFILE>);
```

<MODULE ZIPFILE>	Module zipfile.

`getModuleNameFromZip` returns **ModuleName** if checkZip succeeds.

Example:

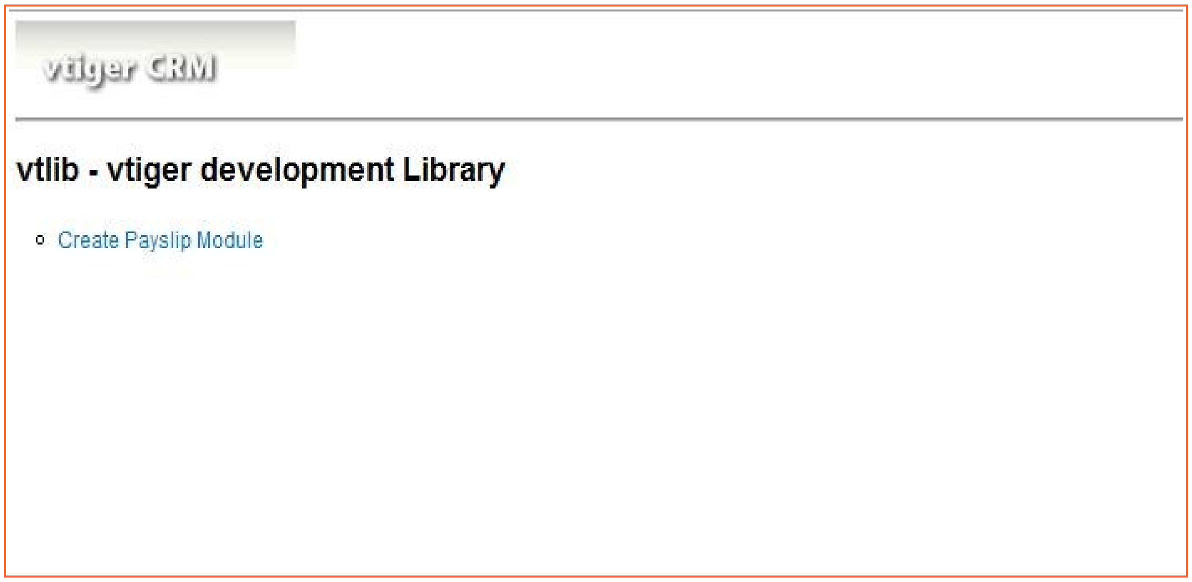
```
require_once('vtlib/Vtiger/Package.php');
require_once('vtlib/Vtiger/Module.php');

$package = new Vtiger_Package();
$module = $package->getModuleNameFromZip('test/vtlib/Payslip.zip');

$module_exists = false;
$module_dir_exists = false;
if($module == null) {
      echo “Module zipfile is not valid!”;
} else if(Vtiger_Module::getId($module)) {
      echo "$module already exists!";
      $module_exists = true;
} else if(is_dir("modules/$module")) {
      echo "$module folder exists! Overwrite?";
      $module_dir_exists = true;
}
if($module_exists == false && $module_dir_exists == false) {
      $package->import('test/vtlib/Payslip.zip');
}
```

Executing Module Creation Script

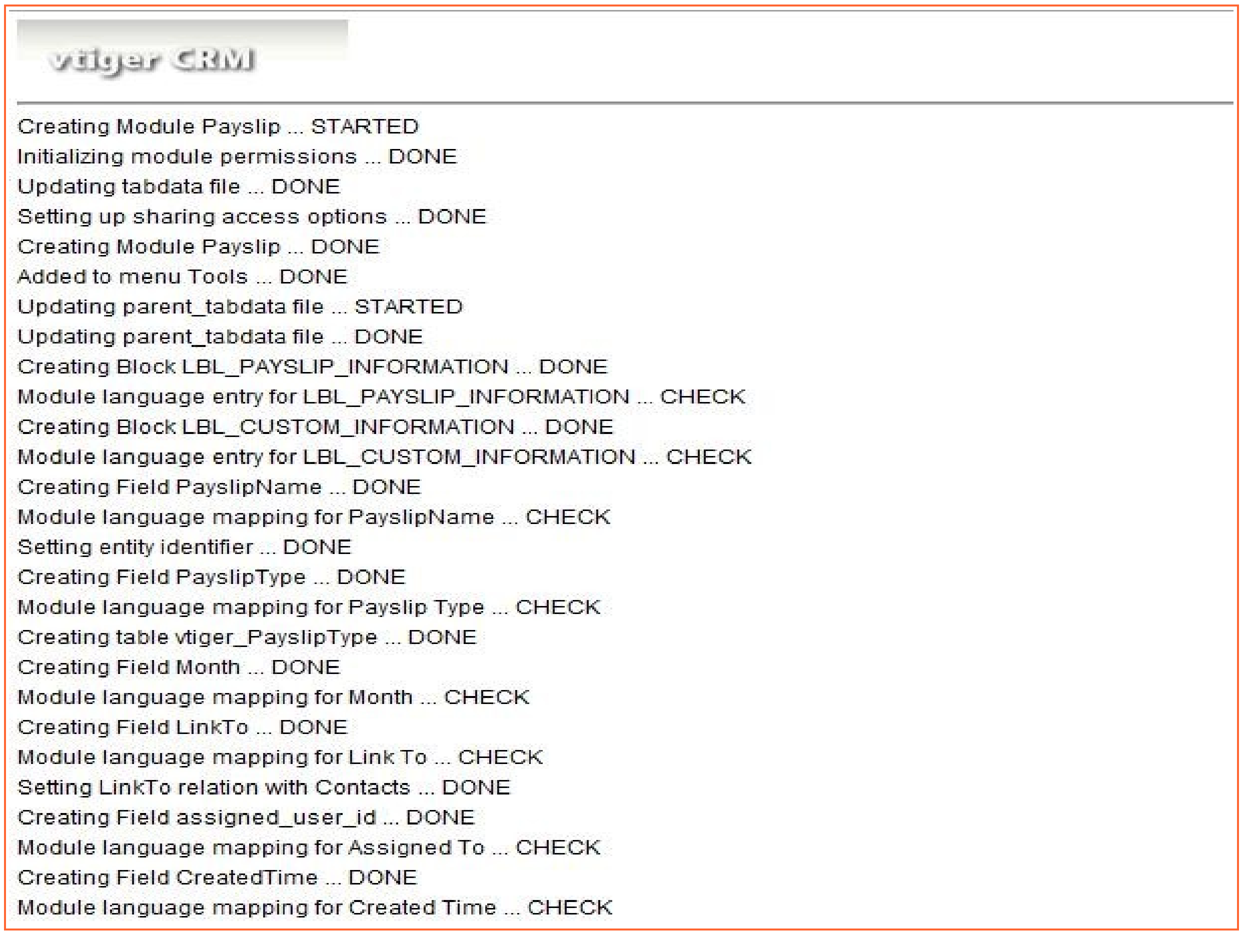
1. To execute the vtlib.Test.Create.Module1.php script, open http://localhost/vtigercrm/vtlib.Test.html



vtlib - vtiger development Library

- Create Payslip Module

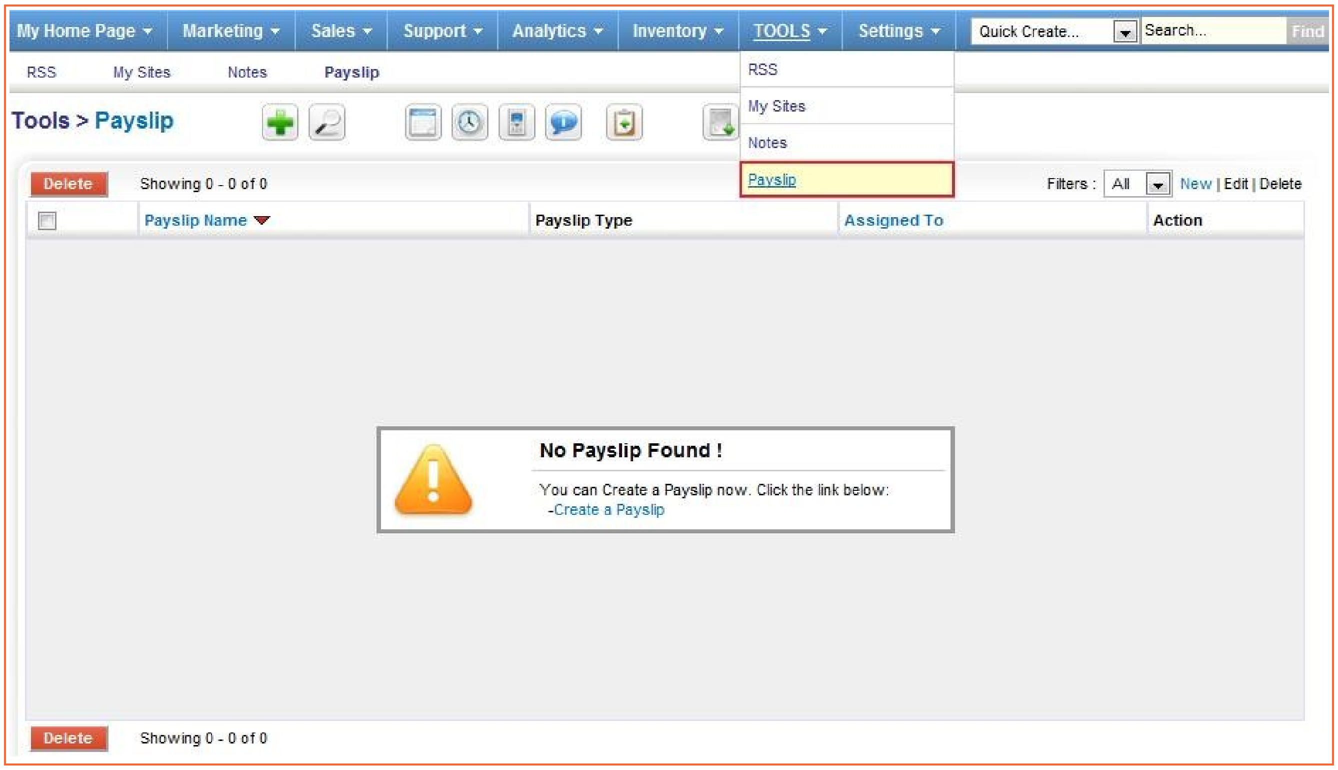
2. Click on Create Payslip Module to test creation of Payslip Module



```
Creating Module Payslip ... STARTED
Initializing module permissions ... DONE
Updating tabdata file ... DONE
Setting up sharing access options ... DONE
Creating Module Payslip ... DONE
Added to menu Tools ... DONE
Updating parent_tabdata file ... STARTED
Updating parent_tabdata file ... DONE
Creating Block LBL_PAYSLIP_INFORMATION ... DONE
Module language entry for LBL_PAYSLIP_INFORMATION ... CHECK
Creating Block LBL_CUSTOM_INFORMATION ... DONE
Module language entry for LBL_CUSTOM_INFORMATION ... CHECK
Creating Field PayslipName ... DONE
Module language mapping for PayslipName ... CHECK
Setting entity identifier ... DONE
Creating Field PayslipType ... DONE
Module language mapping for Payslip Type ... CHECK
Creating table vtiger_PayslipType ... DONE
Creating Field Month ... DONE
Module language mapping for Month ... CHECK
Creating Field LinkTo ... DONE
Module language mapping for Link To ... CHECK
Setting LinkTo relation with Contacts ... DONE
Creating Field assigned_user_id ... DONE
Module language mapping for Assigned To ... CHECK
Creating Field CreatedTime ... DONE
Module language mapping for Created Time ... CHECK
```

New Module Tour

List view



Showing 0 - 0 of 0

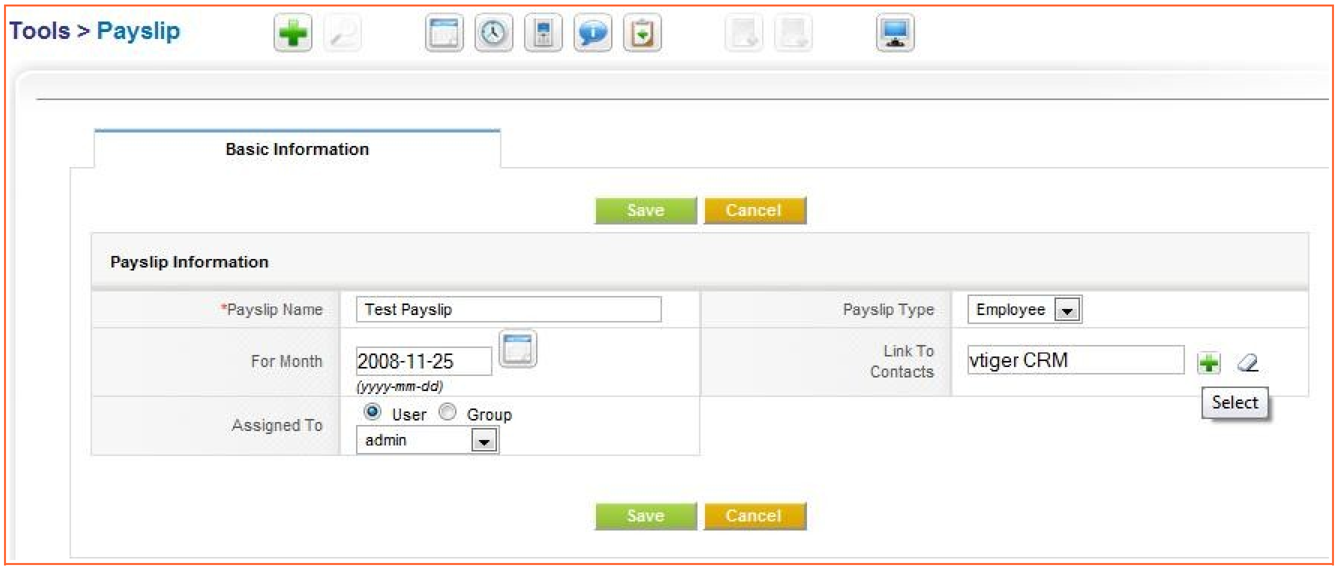
	Payslip Name	Payslip Type	Assigned To	Action

No Payslip Found !

You can Create a Payslip now. Click the link below:
-Create a Payslip

Delete Showing 0 - 0 of 0

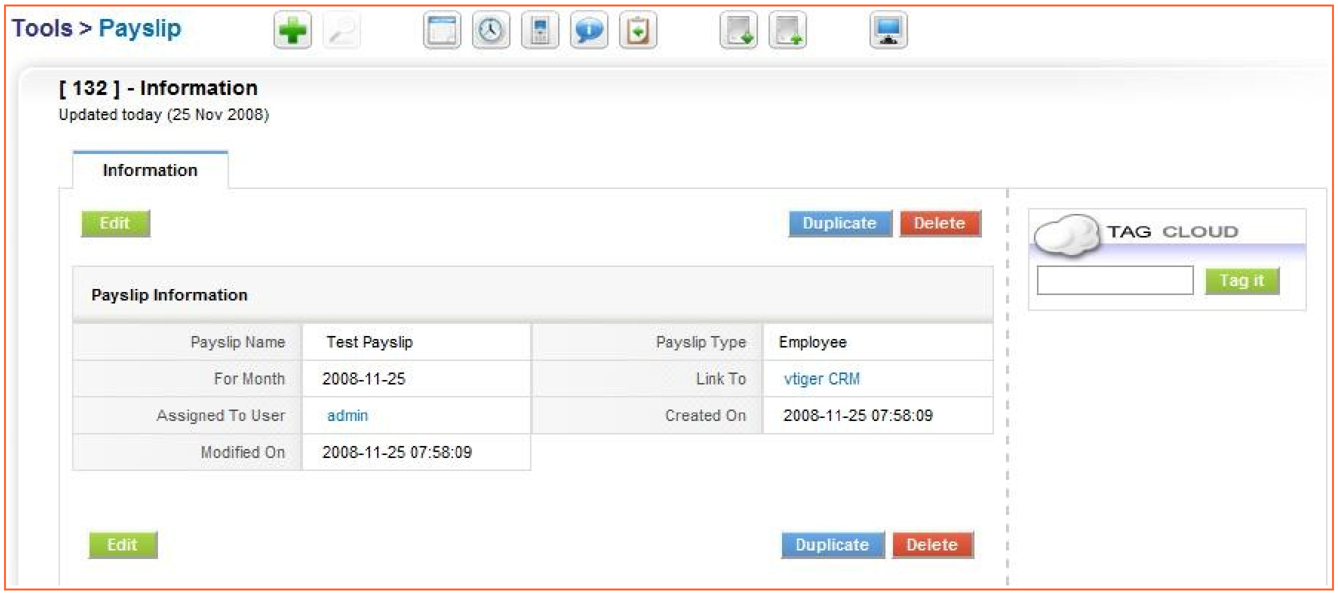
Create view



Payslip Information

*Payslip Name	Test Payslip	Payslip Type	Employee
For Month	2008-11-25 (yyyy-mm-dd)	Link To Contacts	vtiger CRM
Assigned To	[✓ User] Group admin		

Detail view

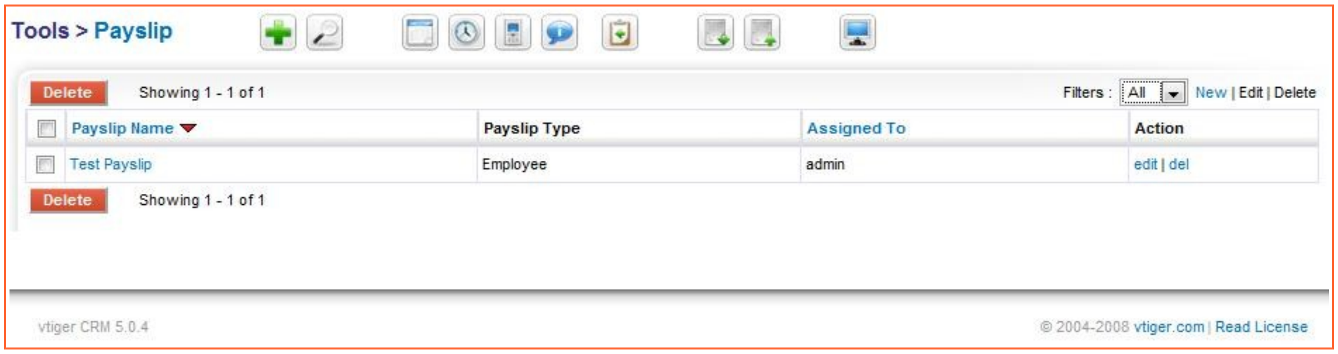


[132] - Information
Updated today (25 Nov 2008)

Payslip Information

Payslip Name	Test Payslip	Payslip Type	Employee
For Month	2008-11-25	Link To	vtiger CRM
Assigned To User	admin	Created On	2008-11-25 07:58:09
Modified On	2008-11-25 07:58:09		

List view

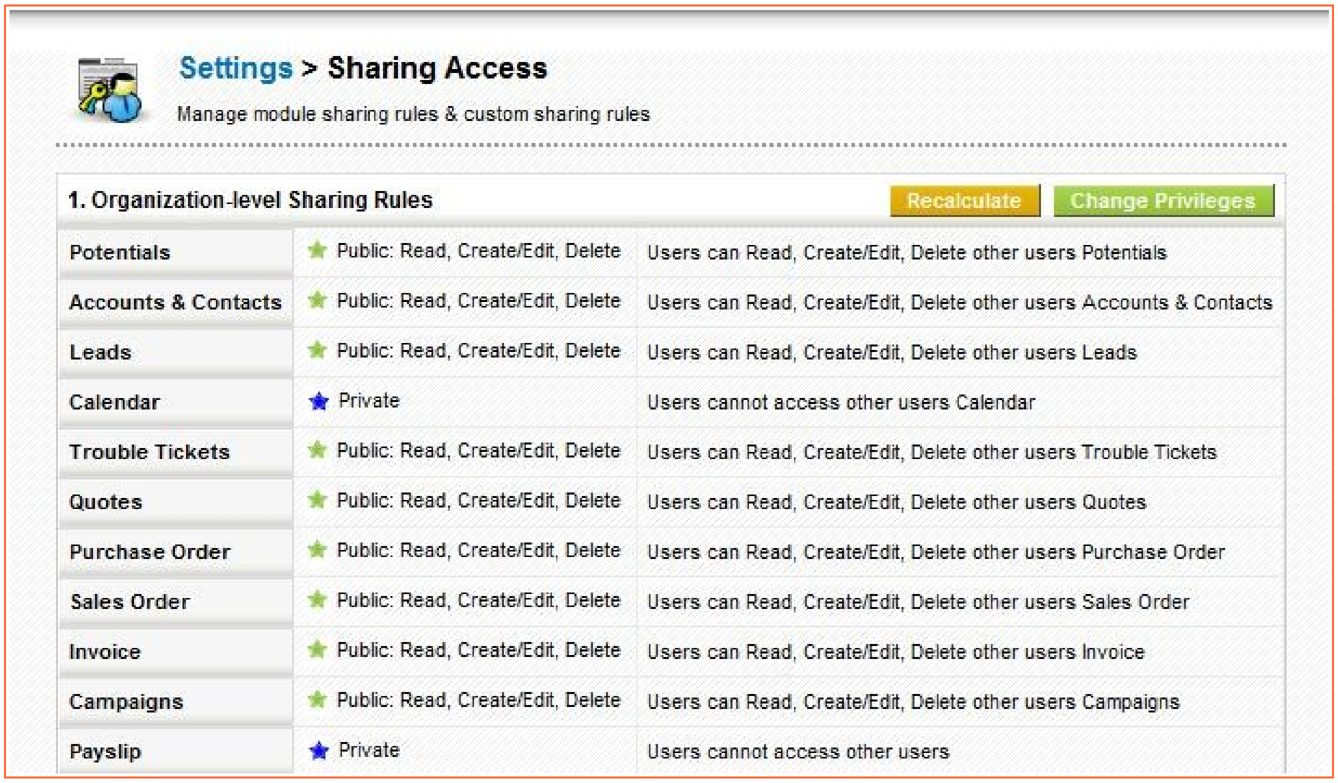


Showing 1 - 1 of 1

Payslip Name	Payslip Type	Assigned To	Action
Test Payslip	Employee	admin	edit | del

Showing 1 - 1 of 1

Sharing Access



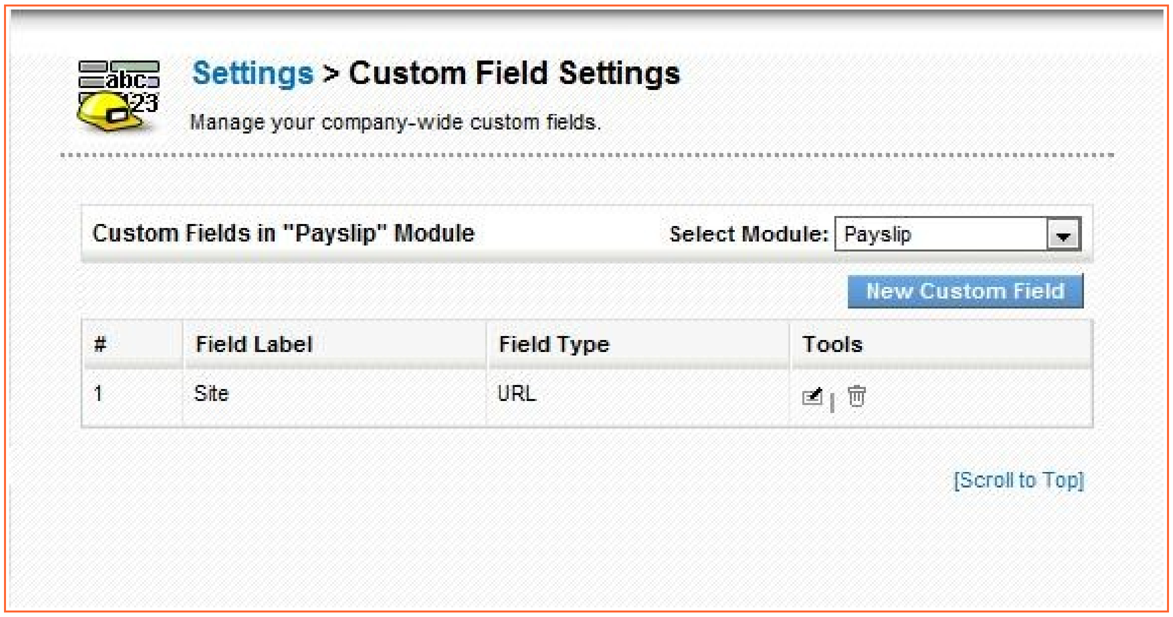
Settings > Sharing Access

Manage module sharing rules & custom sharing rules

1. Organization-level Sharing Rules

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

Custom Fields



Settings > Custom Field Settings

Manage your company-wide custom fields.

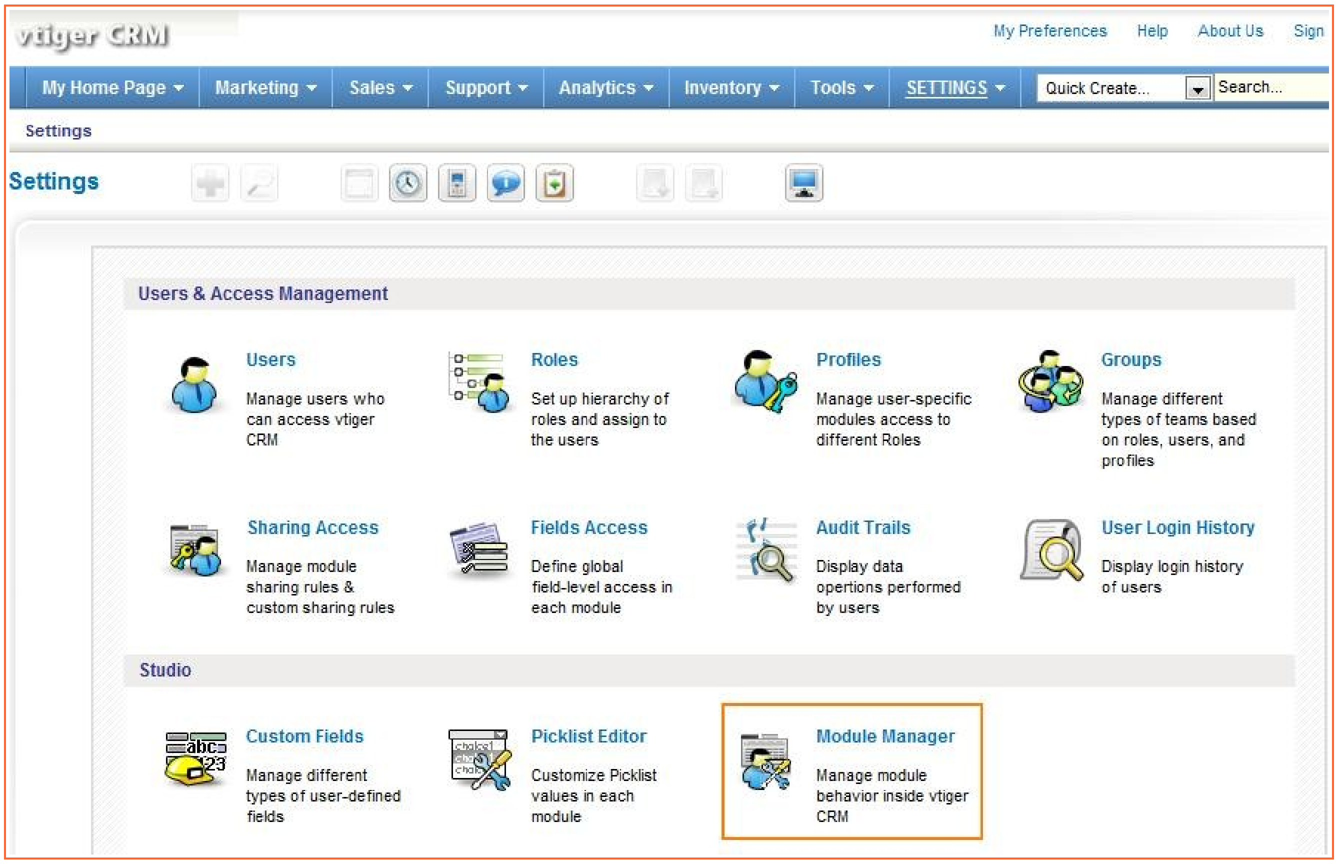
Custom Fields in "Payslip" Module

Select Module: Payslip

#	Field Label	Field Type	Tools
1	Site	URL	

Module Manager

vtlib provides Module Manager configuration tool under Settings. With this you can enable, disable or control settings of vtiger CRM modules. On disabling the module, it won't be shown on the Menu and access is restricted (including administrator).



Users & Access Management

Users
Manage users who can access vtiger CRM

Roles
Set up hierarchy of roles and assign to the users

Profiles
Manage user-specific modules access to different Roles

Groups
Manage different types of teams based on roles, users, and profiles

Sharing Access
Manage module sharing rules & custom sharing rules

Fields Access
Define global field-level access in each module

Audit Trails
Display data opertions performed by users

User Login History
Display login history of users

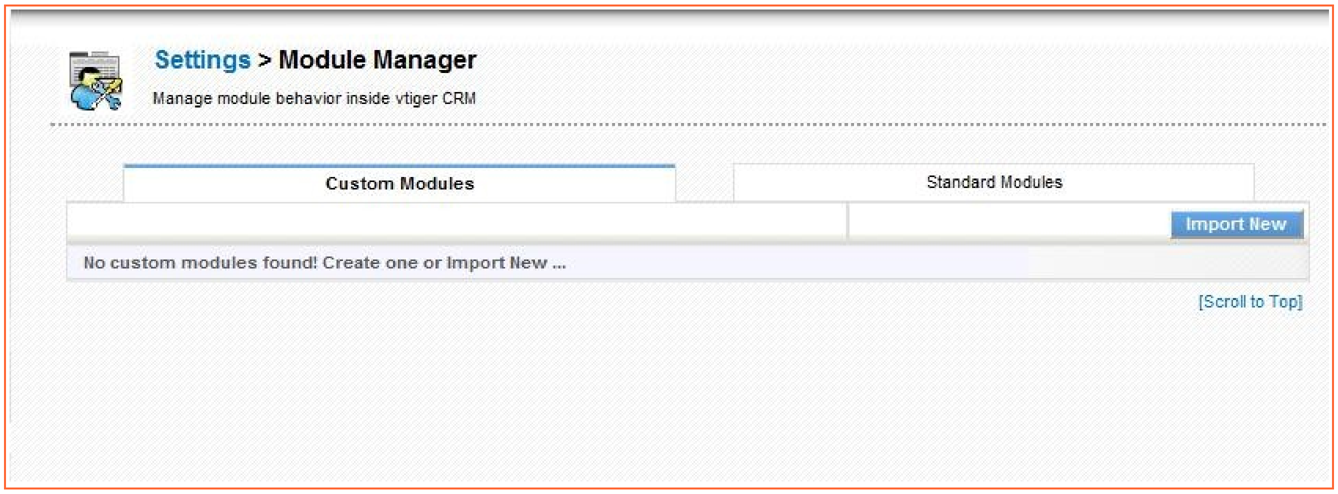
Studio

Custom Fields
Manage different types of user-defined fields

Picklist Editor
Customize Picklist values in each module

Module Manager
Manage module behavior inside vtiger CRM

Modules are categorized as Standard (which are provided by vtiger CRM) and Custom (which you have imported or created)

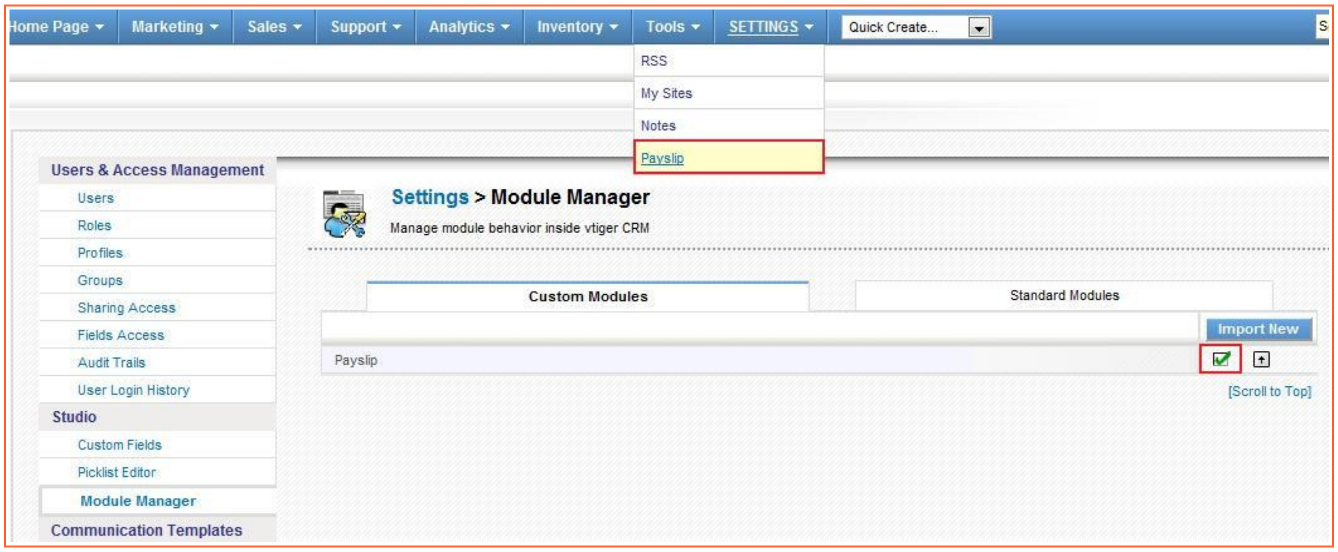


Manage module behavior inside vtiger CRM

No custom modules found! Create one or Import New ...

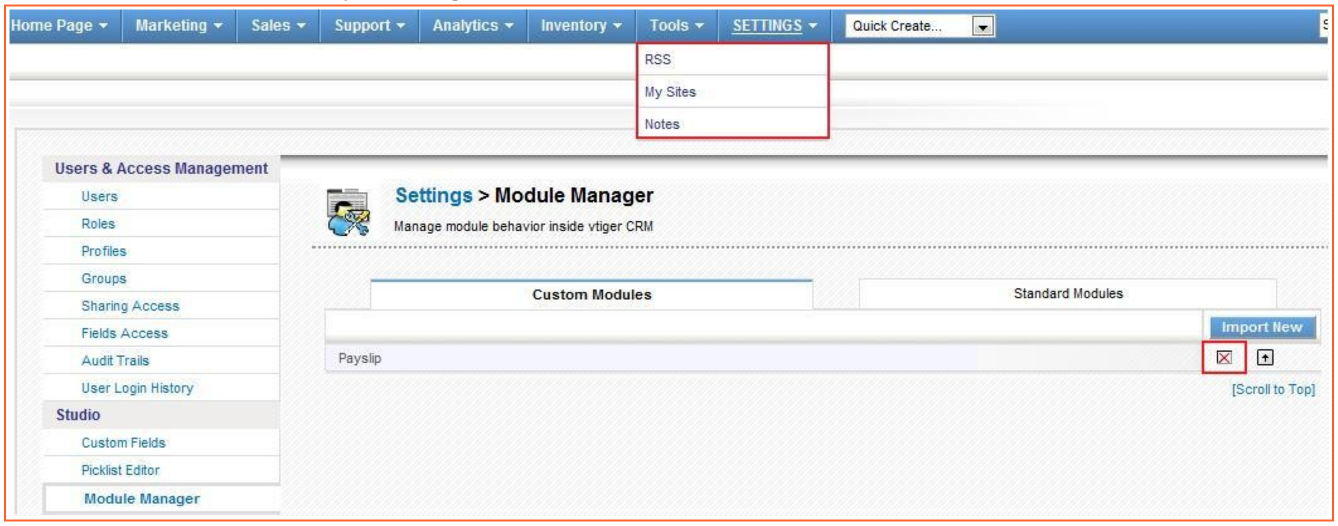
Disabling Module

You can disable module by clicking on the green tick mark icon.



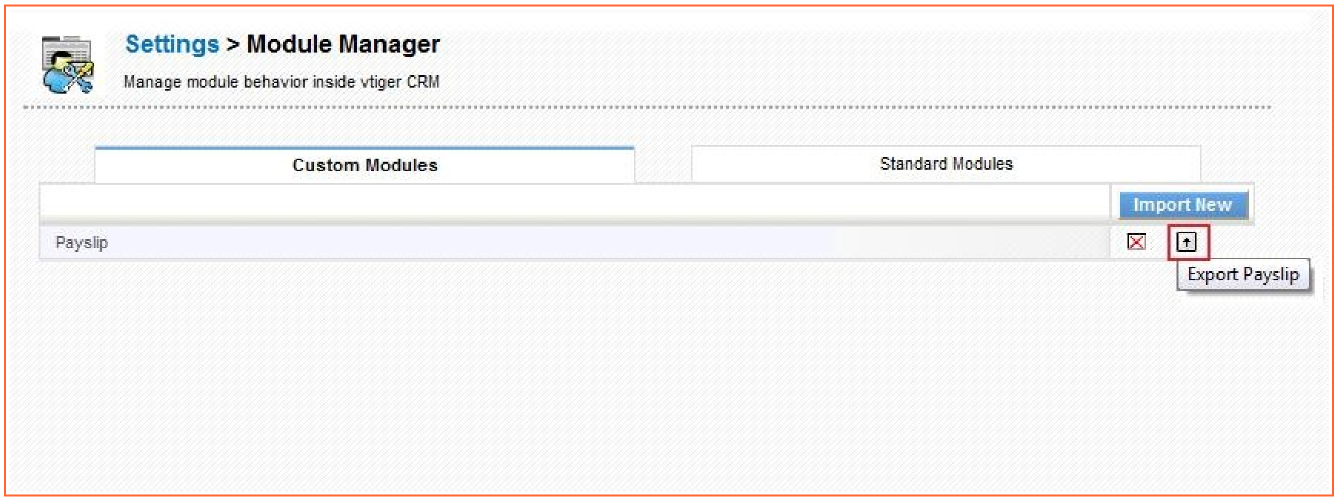
Enabling Module

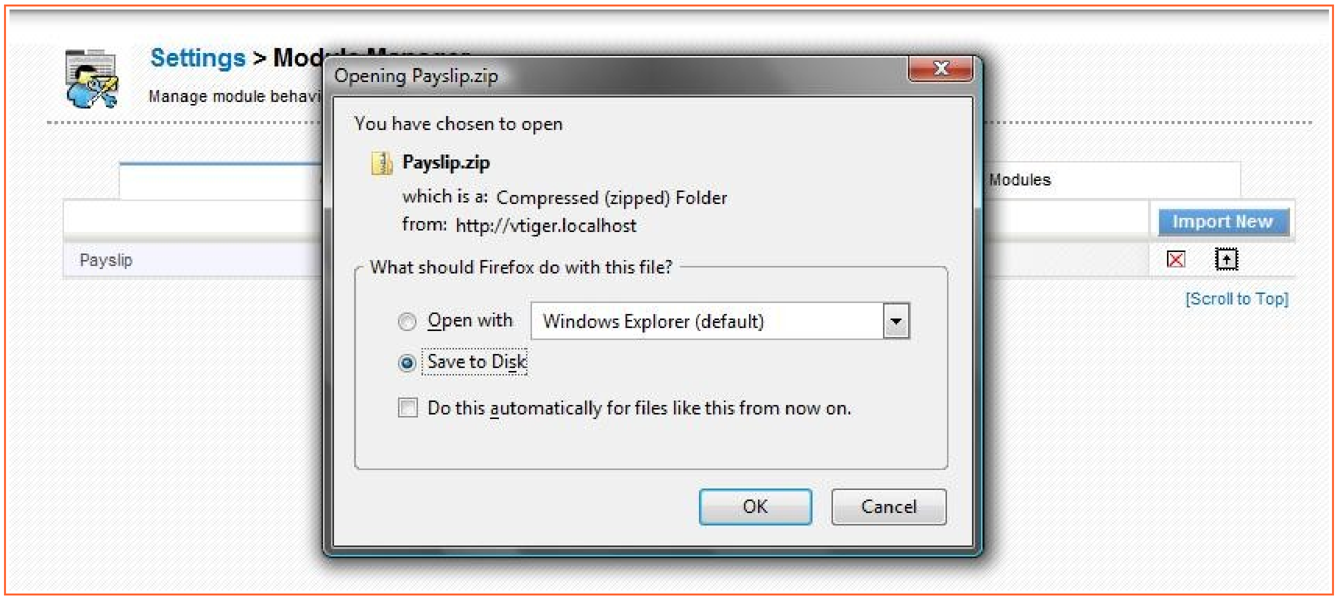
You can enable module by clicking on the red tick mark icon.



Exporting Module

Click on the UP arrow icon in the module manager, which will export the module as a zip file.


Settings > Module Manager
Manage module behavior inside vtiger CRM
Custom Modules
Standard Modules
Import New
Payslip
Export Payslip


Opening Payslip.zip

You have chosen to open

Payslip.zip

which is a: Compressed (zipped) Folder

from: http://vtiger.localhost

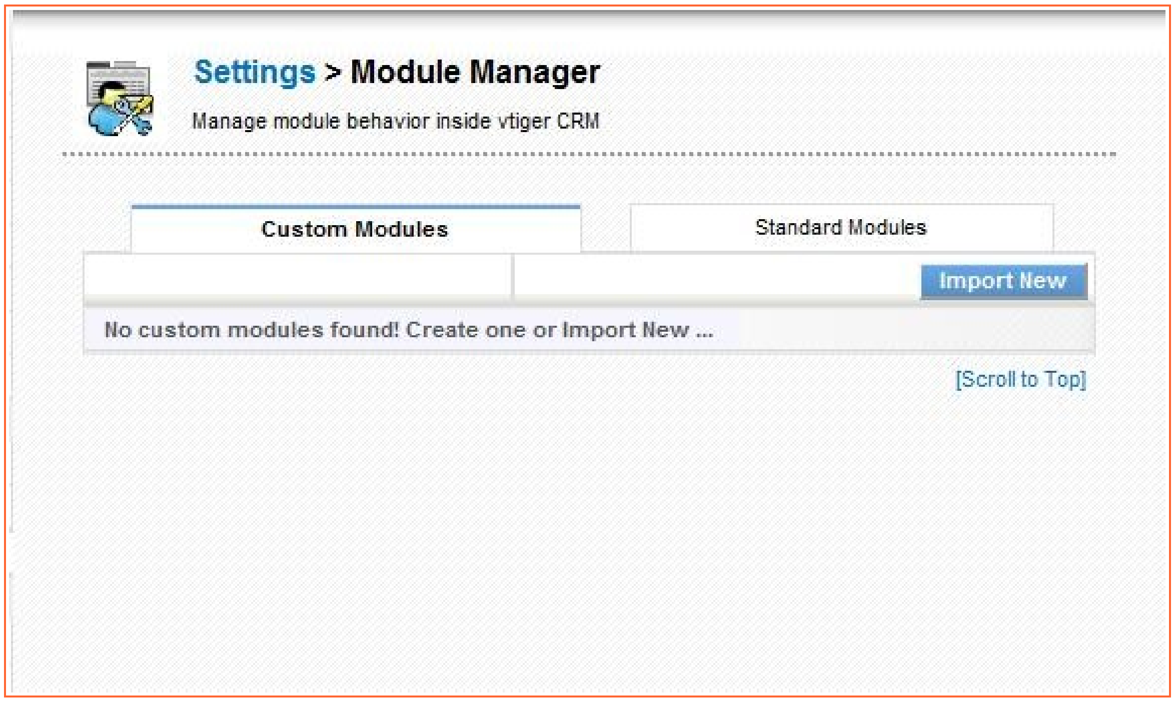
What should Firefox do with this file?

- Open with Windows Explorer (default)
- [✓ Save to Disk]
- Do this automatically for files like this from now on.

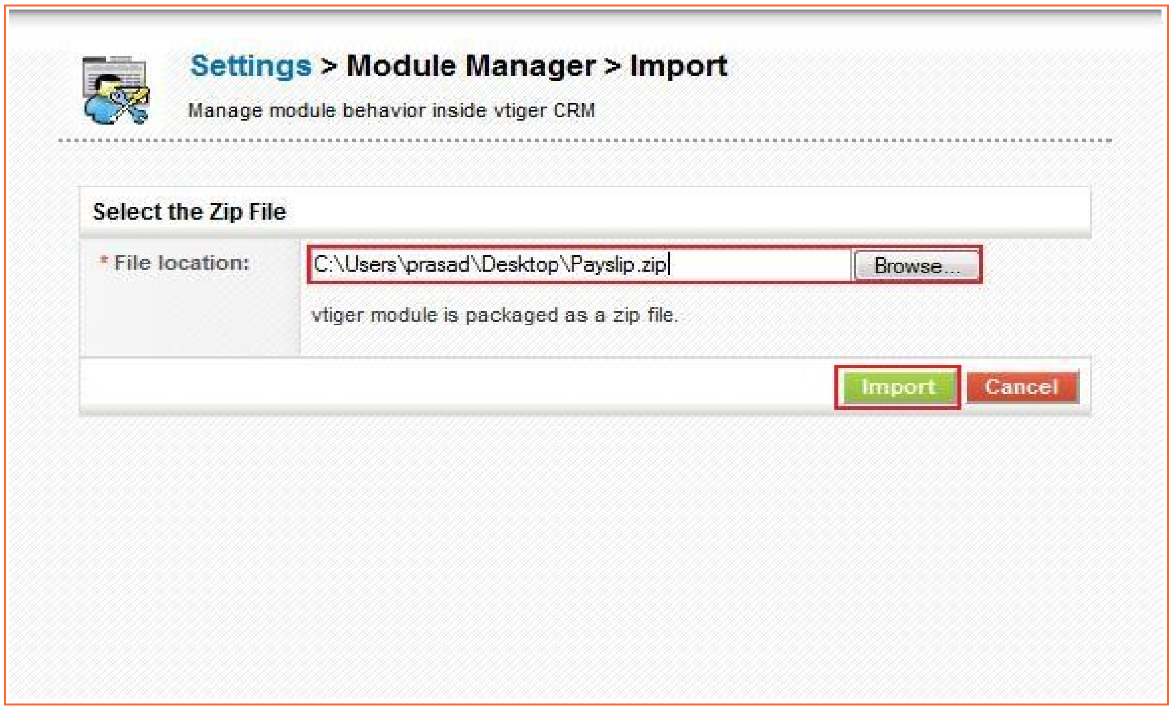
Importing Module

Module manager will let you import new modules. Follow the steps given below:

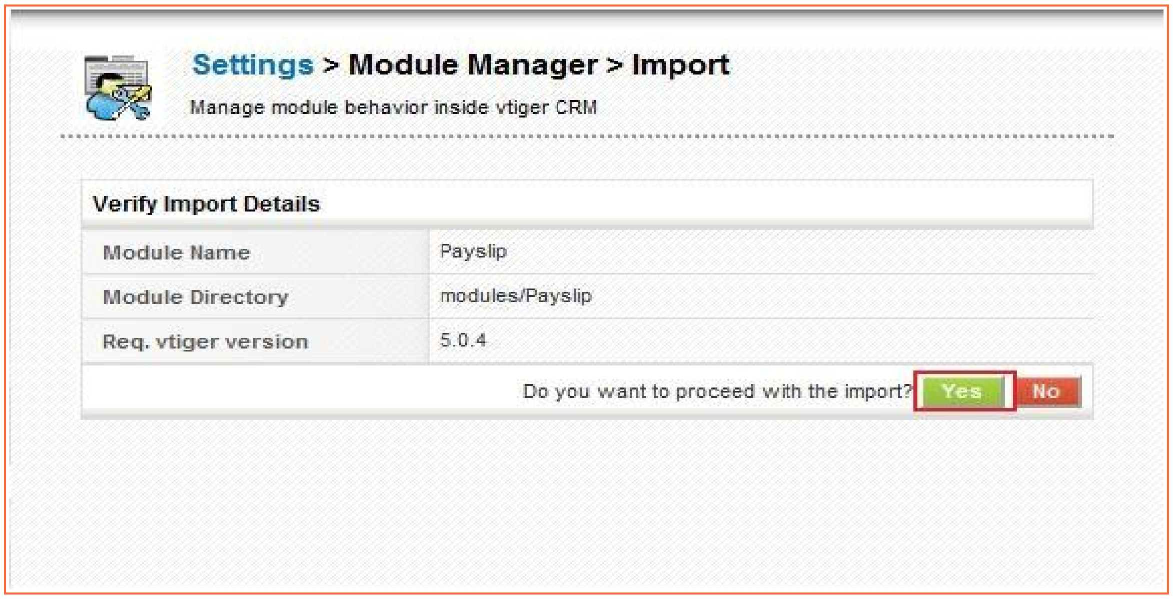
Click on the Import New button


Settings > Module Manager
Manage module behavior inside vtiger CRM
Custom Modules
Standard Modules
Import New
No custom modules found! Create one or Import New ...
[Scroll to Top]

Select the module zip (package) file that was previously exported or created.


Settings > Module Manager > Import
Manage module behavior inside vtiger CRM
Select the Zip File
* File location:
C:\Users\prasad\Desktop\Payslip.zip
Browse...
vtiger module is packaged as a zip file.
Import
Cancel

Verify the import details parsed from zipfile. Click Yes to proceed or No to cancel.

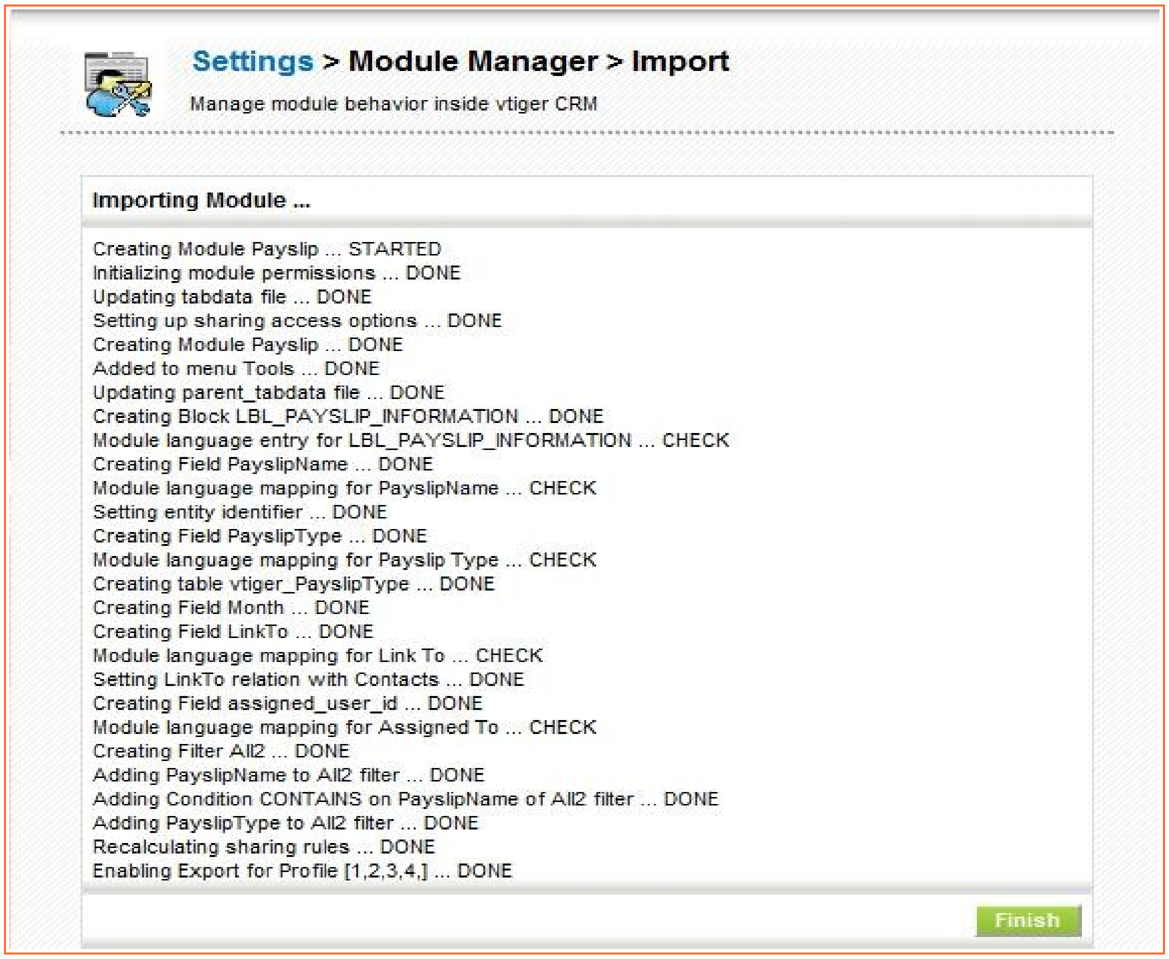

Settings > Module Manager > Import
Manage module behavior inside vtiger CRM

Verify Import Details

Module Name	Payslip
Module Directory	modules/Payslip
Req. vtiger version	5.0.4

Do you want to proceed with the import?

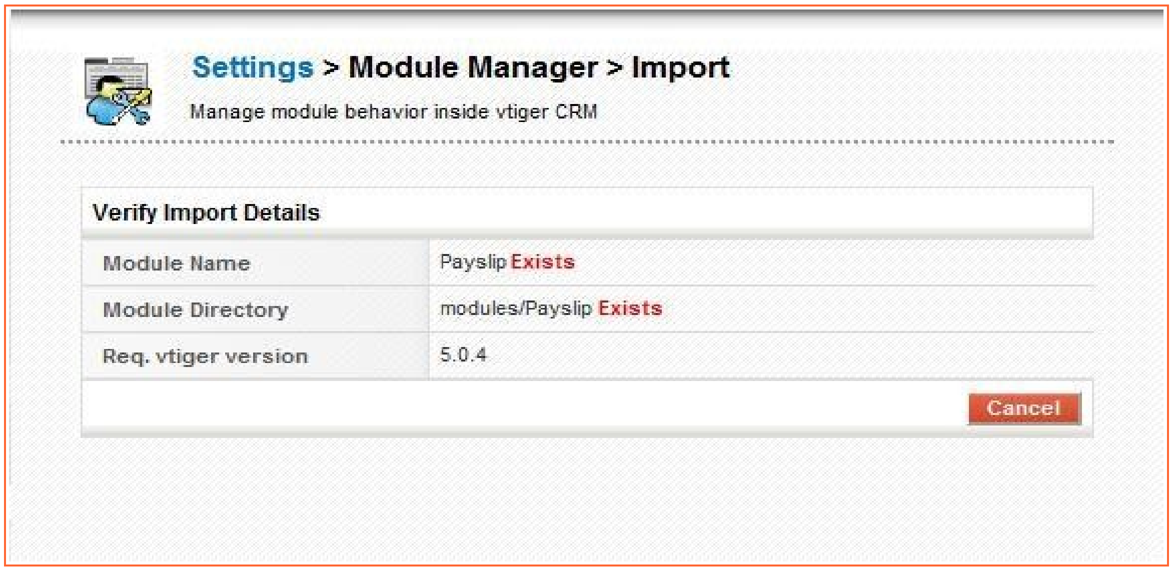
Click on Finish to complete the module import.


Settings > Module Manager > Import
Manage module behavior inside vtiger CRM

Importing Module ...

```
Creating Module Payslip ... STARTED
Initializing module permissions ... DONE
Updating tabdata file ... DONE
Setting up sharing access options ... DONE
Creating Module Payslip ... DONE
Added to menu Tools ... DONE
Updating parent_tabdata file ... DONE
Creating Block LBL_PAYSLIP_INFORMATION ... DONE
Module language entry for LBL_PAYSLIP_INFORMATION ... CHECK
Creating Field PayslipName ... DONE
Module language mapping for PayslipName ... CHECK
Setting entity identifier ... DONE
Creating Field PayslipType ... DONE
Module language mapping for Payslip Type ... CHECK
Creating table vtiger_PayslipType ... DONE
Creating Field Month ... DONE
Creating Field LinkTo ... DONE
Module language mapping for Link To ... CHECK
Setting LinkTo relation with Contacts ... DONE
Creating Field assigned_user_id ... DONE
Module language mapping for Assigned To ... CHECK
Creating Filter All2 ... DONE
Adding PayslipName to All2 filter ... DONE
Adding Condition CONTAINS on PayslipName of All2 filter ... DONE
Adding PayslipType to All2 filter ... DONE
Recalculating sharing rules ... DONE
Enabling Export for Profile [1,2,3,4,] ... DONE
```

NOTE: If you are trying to import module which already exists or directory which is present in the modules folder you will see the following message.



Settings > Module Manager > Import

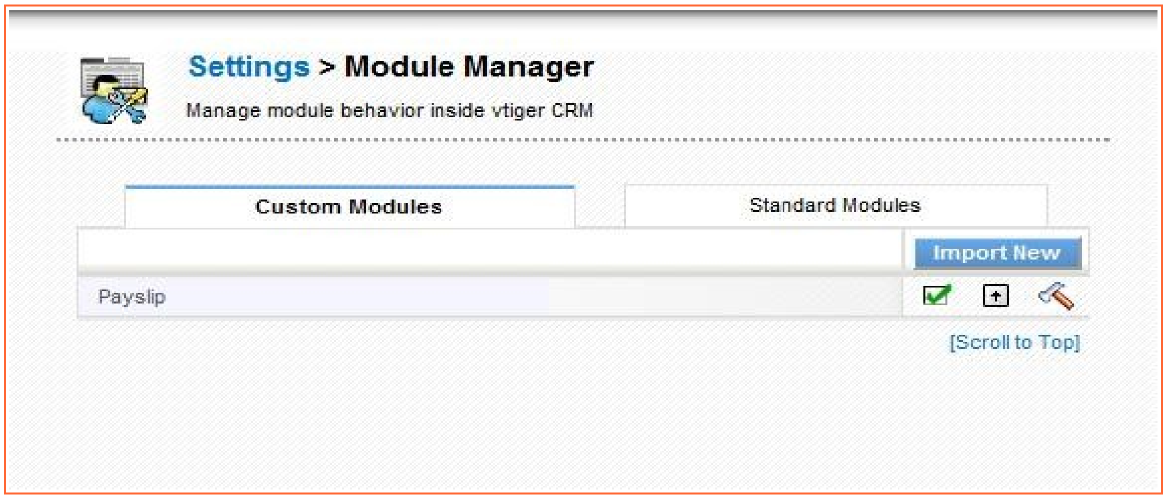
Manage module behavior inside vtiger CRM

Verify Import Details

Module Name	Payslip **Exists**
Module Directory	modules/Payslip **Exists**
Req. vtiger version	5.0.4

Module Specific Settings

A module can have its own specific settings. In such case, Settings.php should be created under the module folder. This file will be invoked (if found) when Settings icon is clicked.


Settings > Module Manager
Manage module behavior inside vtiger CRM
Custom Modules
Standard Modules
Import New
Payslip
[Scroll to Top]

Example: Sample Settings.php for Payslip module

```
<?php

$thisModule = $_REQUEST['formodule'];

require_once('Smarty_setup.php');

$smarty = new vtigerCRM_Smarty();

// Use the module specific template file
// modules/Payslip/MySettings.tpl

$smarty->display(vtlib_getModuleTemplate('Payslip', 'MySettings.tpl'));
?>
```

Appendix 1 - API Changes

vtlib 2.0 contains changes to the APIs previousl provided in 1.x version. The new APIs are more modular and adhere to the OOD model.

We explain the changes below.

Creating Module

Using vtlib 1.x

```
Vtiger_Tab::create('Payslip', 'Payslip', 'Tools');
Vtiger_Utils::CreateTable('vtiger_payslip', '(payslipid integer)');
Vtiger_Utils::CreateTable('vtiger_payslipcf', '(payslipid integer, primary key
(payslipid))');
Vtiger_Utils::CreateTable('vtiger_payslipgrouprel',
      '(payslipid integer, groupname varchar(100), primary key(payslipid))');
```

Using vtlib 2.x

```
$moduleInstance = new Vtiger_Module();
$moduleInstance->name = 'Payslip';
$moduleInstance->save();
$moduleInstance->initTables();

$menuInstance = Vtiger_Menu::getInstance('Tools');
$menuInstance->addModule($moduleInstance);
```

Creating Block

Using vtlib 1.x

```
Vtiger_Block::create('Payslip', 'LBL_PAYSLIP_INFORMATION');
```

Using vtlib 2.x

```
$blockInstance = new Vtiger_Block();
$blockInstance->label = 'LBL_PAYSLIP_INFORMATION';
$moduleInstance->addBlock($blockInstance);
```

Creating Field

Using vtlib 1.x

```
$fieldInstance = new Vtiger_Field();
$fieldInstance-> set('module',        'Payslip')
      -> set('columnname',    'payslipname')
      -> set('tablename',     'vtiger_payslip')
      -> set('columntype',    'varchar(255)')
      -> set('generatedtype', '1')
      -> set('uitype',        2)
      -> set('fieldname',     'payslipname')
      -> set('fieldlabel',    'PayslipName')
      -> set('readonly',       '1')
      -> set('presence',       '0')
      -> set('selected',       '0')
      -> set('maximumlength',  '100')
      -> set('sequence',       null)
      -> set('typeofdata',     'V~M')
      -> set('quickcreate',    '1')
      -> set('block',          null)
      -> set('blocklabel',     'LBL_PAYSLIP_INFORMATION')
      -> set('displaytype',    '1')
      -> set('quickcreatesequence', null)
      -> set('info_type',      'BAS');
$fieldInstance->create();
```

Using vtlib 2.x

```
$fieldInstance = new Vtiger_Field();
$fieldInstance->name = 'PayslipName';
$fieldInstance->table = 'vtiger_payslip';
$fieldInstance->column = 'payslipname';
$fieldInstance->columntype = 'VARCHAR(100)';
$fieldInstance->uitype = 2;
$fieldInstance->typeofdata = 'V~M';
$blockInstance->addField($fieldInstance);
```

Setting Entity Identifier

Using vtlib 1.x

```
$fieldInstance->set('entityidfield', 'payslipid')
              ->set('entityidcolumn', 'payslipid');
$fieldInstance->setEntityIdentifier();
```

Using vtlib 2.x

```
$moduleInstance->setEntityIdentifier($fieldInstance);
```

Set Picklist Values

Using vtlib 1.x

```
$fieldInstance->setupPicklistValues( Array ('Employee', 'Trainee') );
```

Using vtlib 2.x

```
$fieldInstance->setPicklistValues( Array ('Employee', 'Trainee') );
```

Creating Filter

Using vtlib 1.x

```
Vtiger_CustomView::create('Payslip', 'All',true);
$cv = new Vtiger_CustomView('Payslip', 'All');
$cv->addColumn($fieldInstance1)
      ->addColumn($fieldInstance2, 1);
```

Using vtlib 2.x

```
$filterInstance = new Vtiger_Filter();
$filterInstance->name = 'All';
$filterInstance->isdefault = true;
$moduleInstance->addFilter($filterInstance);

$filterInstance->addField($fieldInstance1)->addField($fieldInstance2, 1);
```

Configure Tools

Using vtlib 1.x

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

Using vtlib 2.x

```
$moduleInstance->enableTools(Array('Import', 'Merge'));
$moduleInstance->disableTools('Export');
```

Configure Sharing Access

Using vtlib 1.x

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

Using vtlib 2.x

```
$moduleInstance->setDefaultSharing('Private');
```