Developer Guide

Previous topic

Backup

Next topic

Third Party App Integration (REST APIs)

Quick search

Go

Enter search terms or a module, class or function name.

Index

Quick search

Go

Enter search terms or a module, class or function name.

Vtiger CRM » index

Third Party App Integration (REST APIs)

REST APIs (Webservices)

Take advantage of REST APIs exposed over HTTP(s) to push or pull data from Vtiger and integrate with 3rd party applications. You are certainly free to choose the library of your choice to work with these APIs. vtwsclib provides support to work with REST APIs through various programming languages.

Section below covers more details on the APIs:

Request Format

HTTP - GET / POST

application/x-www-form-urlencoded

Response Format

Success

```
{
        success: true,
        result: {
                // ...
        }
}
```

Failure

```
{
        success: false,
        error: {
                message: String,
                code:    String
        }
}
```

Login Operation

Its a two-step process that involves getting the challenge token and exchanging the credentials (username and accessKey). You can find accessKey information under “My Preferences” in the CRM Web UI.

Challenge

```
GET /webservice.php?
operation=getchallenge&username=<USERNAME> HTTP/1.1
```


Previous topic

Developer Guide

Next topic

Develop Extensions For Vtiger

Challenge Response

```
{
        success: true,
        result: {
                token: TOKENSTRING,    // Challenge
token to be used for login.
                serverTime: TIMESTAMP, // Current
Server time
                expireTime: TIMESTAMP  // Time when
token expires
        }
}
```

Login

```
POST /webservice.php HTTP/1.1

operation=login
username=<USERNAME>
accessKey=md5(TOKENSTRING + <ACCESSKEY>) // Note:
accessKey= K here is capitalized.
```

Response

```
{
        success: true,
        result: {
                sessionId:      String, // Unique
Identifier for the session
                userId:         String, // User ID in
CRM
                version:        String, // Webservice
API version
                vtigerVersion:  String  // Version of
CRM
        }
}
```

Logout Operation

```
POST /webservice.php HTTP/1.1

operation=logout
sessionName=sessionId // Obtained through Login
Operation
```

List Types Operation

```
GET /webservice.php?
operation=listtypes&sessionName=sessionId HTTP/1.1
```

Describe Operation

```
GET /webservice.php?
operation=describe&sessionName=sessionId&elementType=<TY
PE> HTTP/1.1
```

Create Operation

```
POST /webservice.php HTTP/1.1

operation=create
sessionName=sessionId        // Obtained through Login
Operation
element=URLENCODED(JSONDATA) // JSONDATA - JSON Map of
(fieldname=fieldvalue)
elementType=<TYPE>           // TYPE - Module Name
```

Retrieve Operation

```
GET /webservice.php?
operation=retrieve&sessionName=sessionId&id=<WEBSERVICE_
ID> HTTP/1.1
```

Update Operation

```
POST /webservice.php HTTP/1.1

operation=update
sessionName=sessionId    // Obtained through Login
Operation
element=URLENCODED(DATA) // DATA - Map of
(fieldname=fieldvalue)
```

Delete Operation

```
POST /webservice.php HTTP/1.1

operation=delete
sessionName=sessionId    // Obtained through Login
Operation
id=<WEBSERVICE_ID>
```

Query Operation

```
GET /webservice.php?
operation=query&sessionName=sessionId&query=<QUERY>
HTTP/1.1
```

<QUERY>

FORMAT:

```
SELECT * | <column_list> | <count(*)>
FROM <object>
[WHERE <conditionals>]
[ORDER BY <column_list>]
[LIMIT [<m>, ] <n>]
```

<column_list>	Comma separated list of field names.
<object>	Type or Module Name.
<conditionals>	<condition_operations> or <in_clauses> or <like_clauses> separated by ‘and’ or ‘or’ operators these are processed from left to

	right. The are no grouping that is bracket operators.
<condition_operations>	<, >, <=, >=, =, !=
<in_clauses>	in ()
<like_clauses>	like 'sqlregex'
<value_list>	a comma separated list of values.
m, n	integer values to specify the offset and limit respectively.

`LIMITATIONS`

- Queries are currently limited to a single object.
- Joins are not supported.
- Query always limits its output to 100 records, Client application can use limit operator to get different records.

Sync Operation

```
GET /webservice.php?
operation=sync&sessionName=sessionId&modifiedTime=<TIMES
TAMP>&elementType=<TYPE> HTTP/1.1
```

Extend Session Operation

```
GET /webservice.php?operation=extendsession HTTP/1.1
```

Backup

Routine backup (archives) of the database and application setup is essential to ensure failover is smooth.

Tip

mysqldump - administration tool is recommended to take dump of database.

zip - vtigercrm folder (specifically storage, user_privileges)

Previous topic

Migration

Next topic

Developer Guide

Quick search

Go

Enter search terms or a module, class or function name.

Documentation

- User Guide
- Administrator Guide
- Developer Guide
- Extensions Store
- Extensions Store Guidelines
- Community Discussion Guidelines
- Contribute to Open Source Project
- Release Notes
- Examples

Next topic

User Guide

Quick search

Go

Enter search terms or a module, class or function name.

Develop Extensions For Vtiger

Previous topic

Third Party App Integration (REST APIs)

Next topic

Extension Types

- When should my module be Entity type?

Porting Extensions

Previous topic

FAQ's

Next topic

Porting Extensions From 5.x.x to 6.0.0

Quick search

Go

Enter search terms or a module, class or function name.

Extensions Store

What is the need for extensions store?

App-Stores in mobile phones and recent desktop OSs have made installing apps as easy as a 1 click action.

In comparison, the process to install an extension in Vtiger CRM is unwieldy.

1. There was no place within product to browse and install extensions. Users had to search on the web and find extensions listed on different sites.
2. For purchasing paid extensions, users had to do it directly from the individual websites of the publishers.
3. The process of downloading the zip file and installing the extension is error-prone.

Extensions Store is designed to bring the simplicity of mobile app-stores to Vtiger users.

What are the benefits of extensions store?

For Users:

1. 1 click install of commercial or free apps
2. Confidence that apps are verified
3. Contact info of publisher is available to user to reach out for support
4. See Ratings and Reviews of extensions

For Developers:

1. Get More users for your free and commercial extensions
2. View who purchased your apps
3. See Monthly and Daily revenue reports

How do i publish an extension to be listed on Store?

Developers can signup and publish extensions from publisher portal (https://marketplace.vtiger.com)

How are payments handled?

Previous topic

Extensions Store Guidelines

Next topic

Extensions Store Guidelines

Administrators should add payment method (major credit cards are accepted) in order to install commercial extensions. Vtiger relays the credit card details to Stripe.com and retains the customer id only (Vtiger only stores the last 4 digits, expiry month & year of the card for reference).

Stripe is a reputed payment processor and confirms to Credit card industry privacy and security guidelines.

What are the fees for listing on the Extensions store?

There is no listing fee. You decide the price of your extension. You get 75% of the revenue.

How will the money be transferred to developer?

We are able to pay via paypal or through direct wire transfer (wiring fees will be deducted). The payment will be made monthly, within 15th of subsequent month.

Will Vtiger deduct Taxes?

We don’t anticipate the need to collect taxes on top of the extension fee, or withhold tax before sending payment to developer. If any taxes are applicable, it will be the responsibility of the developer to pay taxes.

Can I see who installed my extensions?

Yes. Name of the purchased user will be listed on the Publishers portal

Can I push updates to my extensions?

Yes. You can publish the updates via the publishers portal, and users will be alerted to the presence of an updated version of the extension.

Extensions Store Guidelines

1. Naming Guidelines

Note

All Vtiger CRM extensions listed in the market place should follow these naming guidelines.

1.1 Extension names should not use Vtiger in the name, except as a qualifier at the end, i.e. “for Vtiger CRM.”

1.2 Extension names should not include “Edition” in the name, to prevent user confusion with an edition of Vtiger CRM.

1.3 Extension name should not match any core module bundled with Vtiger installation.

1.4 Extension name and identifier should be have unique short prefix that avoids collision with other extensions published.

2. Extension should be compatible with Module Manager

Note

Only module manager extension will be approved. Please follow vtlib documentation for tips on building module manager compatible extensions.

2.1 Extension package should not include files that are malicious.

2.2 Extension should not overwrite any core / other module file during installation / runtime.

2.3 Extension should not change core / other module tables in database.

2.4 Extension should avoid exposing data to third-party services without user-acceptance.

2.5 Extension should avoid using direct database queries while working with records of its own or other modules. (*Server APIs* usage is recommended).

Previous topic

Extensions Store

Next topic

Community Discussion Guidelines

3. Proper Documentation for extension

3.1 Along with extension, you should send documentation for installation and usage which will help Vtiger team to review extension.

3.2 English Demo (with login credentials) of the working extension will speed up the review process.

4. Acceptance

4.1 Extension that does not follow above naming guidelines will not be accepted.

4.2 Extension with critical bugs will be not be accepted.

4.3 Extension without proper documentation will not be accepted.

4.4 Extension without thumbnail will not be accepted.

4.5 Extensions should not copy the functionality and UI of other extensions or services.

5. Extension Thumbnails

5.1 Thumbnail and Banner images can be used for promoting extension and suggest its feature or purpose.

5.2 Thumbnail and Banner images should not used as a medium to advertise company or any other products or services.

5.3 If company name or logo is included in image, it should be smaller than the other text on the image. The company name/logo should not be bigger than size 16px font in Thumbnail, and 22px in Banner.

5.4 Banner image will be used by Vtiger team to promote the extension in Extension Store, and through Promotional Ads inside Vtiger CRM. Vtiger team reserves the right to pick which extensions are promoted.

6. Extension Description

6.1 Description for the extension should describe the features of the extension.

6.2 Description supports Markdown syntax. Use h4 for section headings.

6.3 Description should not have any links to external pages, nor advertise any other products or services.

6.4 Company details should not be explained in the extension description. Use Author section for company details.

7. Other References

7.1 *Developer Guide*

Attention

Important Note about Vtiger trademark

Publishers should also ensure that they are not violating Vtiger trademarks on their website content and website address. Hosting a Country Code top level domain carrying Vtiger in the name (ex: vtiger.in) is a violation of the trademark. Extensions published by violators will not be published until the issue has been addressed. Vtiger reserves the right to take followup action to protect the trademark.

8. Acceptance Criteria

8.1 Publishers should ensure extensions are testing for smooth installations

8.2 Functionality of extensions should be tested for admin and non-admin access.

8.3 Data access and security should ...Using Server APIs is recommended to avoid pit-falls.

Migration

Attention

PLEASE MAKE SURE TO TAKE BACKUP OF EXISTING SETUP - BOTH DATABASE AND SOURCE.

Steps

- Ensure your server configuration meets the pre-requisites.
- Unzip vtigercrm-6ab-7xy-patch.zip into Vtiger CRM Folder.
  - vtiger7.zip and migrate folder will be unpacked.
- Through browser open /migrate path.
  - http://yourserver.tld/vtigercrm/migrate

Follow the instructions provided on the wizard.

Note

On *nix? Enable write-permission for the apache process owner for smooth migration.

Tips

- Always download the latest migration patch pointed from our website.
- Migration script execution is hungry on time and memory. Ensure your PHP configuration does not limit it.
- Make sure you have configured error_reporting well. Wrong configuration can clog the log files.
- Review - existing table Engine type to be in InnoDB - (expect sequential tables ending with _seq).

Administrator Guide

Previous topic

User Guide

Next topic

Installation

Quick search

Go

Enter search terms or a module, class or function name.

User Guide

Please refer to wiki documentation.

Previous topic

Documentation

Next topic

Administrator Guide

Quick search

Go

Enter search terms or a module, class or function name.

Community Discussion Guidelines

Overview

Vtiger hosts mailing lists and discussion forums to enable discussion and encourage public input. All mailing lists are archived online, and viewable by public.

These include but are not limited to the following.

- Vtiger Developers mailing list - vtigercrm-developers@lists.vtigercrm.com
- Vtiger Discussions Forum - discussions.vtiger.com

Subscribing / Unsubscribing

All interested parties can join the discussions at discussions.vtiger.com or subscribe to the mailing list from the Subscription manager

Posting

It is the responsibility of each person to exercise informed caution when posting to Vtiger mailing lists or discussions site. Users should be aware that their messages will be publicly available in perpetuity through publicly accessible mailing list archives and related archiving projects.

- Individuals are solely responsible for the content of their posts.
- Messages must directly pertain to the list topic. Individuals are advised to review the purpose of each mailing list and select the single list that is most appropriate for the nature of their communications. Cross-posting to multiple mailing lists is discouraged and may be considered spam.
- Posts should be made in plain text format in the body of the message. When necessary, HTML message format may be used, and email attachments may be included. Attachments should be provided in a non-proprietary format.
- Individuals should not forward messages to Vtiger mailing lists without the permission of the original author(s).

Messages containing inappropriate content should not be distributed to Vtiger mailing lists. These include posts which are:

- mass-distributed (spam)
- unrelated to the topic of the mailing list
- disrespectful or abusive in tone
- commercial or overtly promotional - Signatures with links will

Previous topic

Extensions Store Guidelines

Next topic

Contribute to Open Source Project

be considered to be promotional.

- Posts should not include email addresses considered private-use-only by their owners.

Those who post messages containing inappropriate content to Vtiger mailing lists may be permanently unsubscribed and restricted from future participation. Spam may be removed from list archives and subject to further action.

Searching and Browsing List Archives

All messages posted to Vtiger mailing lists are archived on lists.vtigercrm.com

Requests to Edit Mailing List Archives

The archives serve as persistently accessible repositories, documenting the day-to-day email communications of the Vtiger community. To preserve the integrity of these historic records, messages posted to Vtiger mailing lists (with the possible exception of identified spam) are not removed or edited, except in cases of extreme need arising from legal threats or risk to physical safety.

Users of Vtiger mailing lists should consider the immediate as well as long-term effects of disclosing personal or sensitive information before posting. Vtiger is unable to consider requests to edit archives in order to change or remove personal contact information, decrease perceived vulnerability to spam, or avoid citation from search engines.

In exceptional cases, where an acute need to edit an Vtiger mailing list archive can be demonstrated in satisfactory detail, requests should be sent to support@vtiger.com for consideration by Vtiger team.

Feedback

Suggestions that improve upon the above guidelines or which seek to add value to the Vtiger mailing list and discussions procedures are welcomed. Please send comments to support@vtiger.com

Contribute to Open Source Project

We respect your valuable time invested in helping us. Here are few different ways you can engage and contribute towards the product. Before you get started, make sure to register account at code.vtiger.com

File Issues

- Ensure the bug is reproducible on the recent release or latest master.
- Report the issue with detailed steps on reproducing / screenshots and dataset at code.vtiger.com

Submit Bug Fixes

- Fork vtigercrm project
- Update the files
- Send merge request to the master
    - If the fix is specific to a report issue, mention the issue id on the merge request subject. (example: Fixes #1234 - subject of the issue ...)
- Vtiger release manager reviews and accepts the merge request to the master

Note

Vtiger release manager will tag the master with the new version name, and makes the downloads available through this tag. The released versions are available at vtigercrm project tags

Develop Extensions

- Read through the *Developer Guide*
- Share it on vtiger Marketplace

Release Notes

Version 7.1.0

- Major version released to enhance stability of Vtiger 7.0 - addresses 200+ key functional issues and knocked off 500+ issues of usability or inconsistent behaviour that were present in earlier versions.

Version 7.0.1

- Minor version released to address blocker issues on Version 7.0.0

Version 7.0.0

Vtiger 7 UI Layout

New layout makes easy, smooth and sensitive navigation throughout the modules.

Private / Public tags

Tags are a way to organize your data in the ways that are more customized to your company. They are the most powerful tools which will differentiate the important records from the other records. Tags help you find records that are otherwise not easily searchable. (read tag-management)

Variable Taxes, Charges and Regions

Tax is a fee charged (“levied”) by a government on a product, income, or activity. After you define the tax rates in the CRM, they will be available for selection in Quotes, Sales Orders, Invoices and Purchase Orders. (read tax-management)

Rollup Comments

You can view all the comments made on the related records directly from the Parent record. For example, In Organization view, comments made on the records of related modules such as Contacts, Cases, etc., is rolled up and a consolidated view is shown on Organization record’s comment tab. (read comments)

Splitting Edit action in Create / Edit action in roles / profiles

Profiles are used to control the user actions on CRM records. In addition,

Previous topic

Contribute to Open Source Project

Next topic

Examples

Quick search

profiles are also used to restrict the users from accessing specific modules, fields, and tools such as import, export, etc. (read profiles)

Comment with attachments

Comments are a form of communication message added on a record. Users can view the comment and reply to it. You can also add files as an attachment while adding a comment to a record. The maximum upload size to the comment is limited to 5 MB. You can also set a workflow to send an email while a comment is added. (read comment-attachments)

Multiple dashboard tabs

Vtiger CRM offers different home page widgets that will provide graphical representation of current status and key performance indicators. As complete analysis of data is displayed on one single screen, you can monitor all decisions and conclusions at a glance. (read dashboard)

Picklist coloring

Picklist coloring is useful to differentiate one value to another value in View point. (read more)

App Menus

Left Menu helps you to place most frequently accessed modules on Module Navigator of a particular App. You can also rearrange them according to your requirements. (read app-menu)

Controlled sharing of Lists

Create lists to find and organize records based on a condition using Custom Filter. (read lists)

Custom Activity Type

Add module related data into widget in Calendar View. (read activity-widget)

Easier list views

Read lists.

Version 6.5.0

Modifications

- Bug fixes and minor enhancements.

Version 6.4.0

Additions

- Multiple Layout support added.

Modifications

- 40 issues fixed.
- SQL Injection and XSS issues addressed.

Version 6.3.0

Additions

- RelatedList API added to Vtiger Web Services.

Modifications

- 90+ issues fixed.

Version 6.2.0

Modifications

- Google Calendar api upgraded to support V3 api
- Security fix for session hijacking
- 200+ issues fixed.

Version 6.1.0

Additions

- New app marketplace (Extension Store)
- Search within ListView
- Mass Edit for Documents
- Workflow for Documents
- Asterisk
- Scheduled Reports
- Click through Charts
- RSS Widget
- Our Sites
- Custom Charts

Modifications

- PHP 5.4, 5.5 support
- MySQLi support

Version 6.0.0

Additions

- Modern UI with Usability in focus.
- Record Summary View.
- Module level dashboards.
- Quick Global Search on record labels.
- Easy to use Layout Editor with Drag and Drop ability.
- Configuring fields in related lists.
- Google Contact & Calendar Synchronization.
- Linking opportunities/tickets to contacts and organizations.
- 'Can Assign Records To' feature in Role.

Modifications

- Nested Condition Grouping in Filter with (AND/OR) now reduced to Flattened mode.
- Report Sharing (all reports are public in 6.0. only accessible records are visible to user)

5.4.0 Features Missing

- Asterisk (coming in 6.1)
- Scheduled Reports (coming in 6.1)
- Click through Charts (coming in 6.1)
- RSS Widget (coming in 6.1)
- Our Sites (coming in 6.1)
- Custom Charts (coming in 6.1)
- Gantt Charts in Projects (coming in 6.1)
- Mail Merge (needs to be addressed through an extension later)
- Chat feature (feature dropped as it is not very user friendly)

Examples

- Hello World
- World Clock
- City Lookup
- An Entity Module Example
- Slideshow
- An Extension Module Example
- Layout i386
- Uninstall Module

Previous topic

Release Notes

Next topic

Hello World

Quick search

Go

Enter search terms or a module, class or function name.

Extension Types

Modules

Module is the functional unit of Vtiger CRM. Each module comprises of Models, Controllers, Views and associated resources that are required to manage the data and build the UI. Module can be categorized as *Entity Module* or *Extension Module* based on the data it manages.

Language Pack

i18n (Internationalization) support for Vtiger is enabled through Language Pack. The application string used both on the server and client side is translated as key-value pairs in the language files (php). Language translators focus on translating the value part in the file for specific language and package for installation.

Layouts & Themes

Layouts and *Themes* control the UI & UX functionality.

Previous topic

Develop Extensions For Vtiger

Next topic

Development Essentials

Development Essentials

Previous topic

Extension Types

Next topic

Entity Module

Vtiger CRM » Developer Guide » Develop Extensions For Vtiger » previous | next | index

Entity Module

Block

Entity record are visually represented as form in Edit View / Detail View. Block comprises of group of Module Field. There can be one or more blocks in the module. Layout Editor provides the ability for administrator to add custom blocks.

Field

Field is the fundamental unit of data management. Module can have one or more fields. However, Entity Field presence is mandatory for the module.

Entity Field

Primary field(s) of the module through which the record is identified by name (a.k.a label) is termed as Entity Field.

Sharing Access

Webservices

Previous topic

Development Essentials

Next topic

Extension Module

Extension Module

Extension module adds additional behavior to the CRM and works with existing data of Entity module or from other sources. Example: Dashboard, Reports, etc... The platform provides API to make it easy to work with Entity module data and defines general structure to follow for the module implementation.

Previous topic

Entity Module

Next topic

Language Pack

Quick search

Enter search terms or a module, class or function name.

Vtiger CRM » Developer Guide » Develop Extensions For Vtiger » Development Essentials »

previous | next | index

Go

Language Pack

To get started use *Console Tool > Create New Language Pack*

Update the php language files created under the `languages/languagecode_countrycode` folder.

Export the language pack using *Language Export* API

Previous topic

Extension Module

Next topic

Layouts

Layouts

Application UI is controlled by layout. CRM administrator can choose the default layout through Configuration Editor (available 6.4.0 onwards).

Previous topic

Language Pack

Next topic

Themes

Themes

Previous topic

Layouts

Next topic

Server APIs

Vtiger CRM » Developer Guide » Develop Extensions For Vtiger » Development Essentials »

previous | next | index

Server APIs

Server API enables data flow control for a module. It is recommended for use when a module needs to access of other module data / state. The Server API will take measure to apply the required Sharing / Profile access restriction as configured for the logged in user. Also triggers configured Event handlers (Workflow etc...) for operations like Save / Delete.

Attention

Using direct database queries is discouraged as developers might induce data-security issues unknowingly and increase risk.

Webservice ID

```
include_once 'include/Webservices/Utils.php';

$crmid = '1'; // ID of Leads record.
$wsid = vtws_getWebserviceEntityId('Leads', $crmid);
```

List Types

```
include_once 'include/Webservices/ModuleTypes.php';

$current_user = CRMEntity::getInstance('Users');
$current_user->retrieveCurrentUserInfoFromFile(1);

try {
        $typeInformation = vtws_listtypes(array(),
$current_user);
        foreach ($typeInformation['types'] as $name) {
                echo sprintf("%s\n", $name);
        }
        foreach ($typeInformation['information'] as
$name => $information) {
                echo sprintf("Name: %s, Label: %s,
SingluarLabel: %s, IsEntity: %s \n",
                        $name, $information['label'],
$information['singular'], ($information['isEntity']?
"yes": "no"));
        }
} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Describe

```
include_once 'include/Webservices/DescribeObject.php';
```

```
try {
        $describe = vtws_describe('Leads',
$current_user);
        foreach ($describe['fields'] as $field) {
                echo sprintf("%s (%s), Mandatory? %s
\n", $field['name'],
                        $field['type']['name'],
($field['mandatory']? "yes": "no"));
        }
} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Create

```
include_once 'include/Webservices/Create.php';
try {
        $data = array (
                'lastname' => 'LNAME',
                'firstname'=> 'FNAME',
                'company'  => 'CNAME',
                'assigned_user_id' => '19x1', //
19=Users Module ID, 1=First user Entity ID
        );
        $lead = vtws_create('Leads', $data,
$current_user);

        print_r($lead);

} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Retrieve

```
include_once 'include/Webservices/Retrieve.php';
try {
        $wsid = vtws_getWebserviceEntityId('Leads',
'CRMID') // Module_Webservice_ID x CRM_ID
        $lead = vtws_retrieve($wsid, $current_user);
        print_r($lead);

} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Revise

```
include_once 'include/Webservices/Revise.php';
try {
        $wsid = vtws_getWebserviceEntityId('Leads',
'CRMID') // Module_Webservice_ID x CRM_ID
        $data = array('firstname' => 'FIRSTNAME', 'id'
=> $wsid);
        $lead = vtws_revise($data, $current_user);
        print_r($lead);

} catch (WebServiceException $ex) {
        echo $ex->getMessage();
```

```
}
```

Query

```
include_once 'include/Webservices/Query.php';
try {
        $q = "SELECT * FROM Leads WHERE lastname LIKE
'%LNAME%'";
        $q = $q . ';'; // NOTE: Make sure to terminate
query with ;
        $records = vtws_query($q, $current_user);
        print_r($records);

} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Delete

```
include_once 'include/Webservices/Delete.php';
try {
        $wsid = vtws_getWebserviceEntityId('Leads',
'CRMID') // Module_Webservice_ID x CRM_ID
        vtws_delete($wsid, $current_user);

} catch (WebServiceException $ex) {
        echo $ex->getMessage();
}
```

Package Structure

Entity Module

```
<MODULENAME>.zip
  manifest.xml
  modules/
    <MODULENAME>/
        <MODULENAME>.php
        views/
        actions/
        models/
  crons/
    MODULENAME.service
  templates/
    .tpl
    resources/

  languages/
    en_us/
      <MODULENAME>.php

  settings/
    views/
    actions/
    models/
    templates/
      .tpl
```

Extension Module

```
<MODULENAME>.zip
  manifest.xml
  modules/
    <MODULENAME>/
        <MODULENAME>.php
        views/
        actions/
        models/

  templates/
    .tpl
    resources/

  languages/
    en_us/
      <MODULENAME>.php
```

Language Pack

```
<en_us>.zip
  manifest.xml
  modules/
     Vtiger.php
```


Previous topic

Server APIs

Next topic

Vtiger Development Library (vtlib)

```
    Accounts.php
    Contacts.php
    Leads.php
    Settings/
      Vtiger.php
```

Layout Pack

```
<LAYOUTNAME>.zip
  manifest.xml
  layouts/
      <NAME>/
          libraries/
              jquery/
                  jquery.min.js
          skins/
              application.js
              application.css
          modules/
              Vtiger/
                  Header.tpl
                  Footer.tpl
                  dashboards/
                      DashBoardPreProcess.tpl
                      DashBoardContents.tpl
              Users/
                  Login.tpl
              Settings/
                  Vtiger/
                      ConfigEditorDetail.tpl
```

Theme Pack

```
<THEMENAME>.zip
  manifest.xml
  vlayout/
    style.css
  otherlayout/
    style.css
```

Bundled Module

Note

Yet to be documented - please be patient.

Vtiger Development Library (vtlib)

Previous topic

Package Structure

Next topic

Module

Module

Class `Vtiger_Module` provides an API to work with vtiger CRM modules.

```
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>	Contains the default fields for the new module
Customtable	vtiger_<MODULENAME>cf	Contains custom fields of the module

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

Previous topic

Vtiger Development Library (vtlib)

Next topic

Module Block

Quick search

Go

Enter search terms or a module, class or function name.

Module Block

Class `Vtiger_Block` provides API to work with a 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.

Previous topic

Module

Next topic

Module Field

Quick search

Go

Enter search terms or a module, class or function name.

Module Field

Class `Vtiger_Field` provides API to work with a 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]

Optional Settings

$fieldInstance->presence	0 – Always Active (Cannot be modified using Layout Editor in 5.1.0) 1 – Mark it In Active (5.1.0 onwards) 2 – Active Property can be modified using Layout Editor (5.1.0 onwards)
$fieldInstance->quickcreate	0 – Enable field in Quick Create Form 1 – Disable field on Quick Create Form
$fieldInstance->masseditable	0 – Permanently Disallow field for mass editing (5.1.0 onwards)

	1 – Allow field for mass editing (5.1.0 onwards) 2 – Disallow field for mass editing (but can be made available using Layout Editor 5.1.0 onwards)

Entity Identifier

One of the mandatory field should be set as entity identifier of module once it is created. This field 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-
>unsetRelatedModules(Array('OtherModule2'));
```

Set MassEdit property

You can make the field available for mass editing use the following ways described below: When creating the field you can set the property:

```
include_once('vtlib/Vtiger/Module.php');
$fieldInstance = new Vtiger_Field();
$fieldInstance->name = 'TestField';
...
...
$blockInstance->addField($fieldInstance);
```

If you have an existing field its property can be updated using the API:

```
$fieldInstance->setMassEditable(value);
```

The value set for masseditable property has the following meaning:

Value	Description
0	Not available for mass edit and this property cannot be controlled by user.
1	Available for mass edit
2	Not available for mass edit but the property can be controlled by user (via Layout Manager etc)

Module Filter

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

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.

Module Related List

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 a 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>	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 an entry in the 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)

Module Sharing Access

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

The example given below describes the way to configure the 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

Previous topic

Module Related List

Next topic

Module Tool

Module Tool

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 to enable and disable the tools for Payslip module

```
include_once('vtlib/Vtiger/Module.php');
$moduleInstance = Vtiger_Module::getInstance('Payslip');
$module->enableTools(Array('Import', 'Export'));
$module->disableTools('Export');
```

Previous topic

Module Sharing Access

Next topic

Module Event

Module Event

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

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

<MODULENAME>	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
		}
	}
}
?>
```

Previous topic

Module Tool

Next topic

Module Webservice

Module Webservice

You will need to invoke the setup API to enable the support for the custom modules.

```
include_once('vtlib/Vtiger/Module.php');
$moduleInstance = Vtiger_Module::getInstance('Payslip');
$moduleInstance->initWebservice();
```

Note

When the module is imported the Webservice initialize API is automatically invoked.

Previous topic

Module Event

Next topic

Module Link

Module Link

You can add custom web link to the module using the following API:

```
include_once('vtlib/Vtiger/Module.php');
$moduleInstance =
Vtiger_Module::getInstance('ModuleName');
$moduleInstance->addLink(<LinkType>, <LinkLabel>,
<LinkURL>);
```

Attention

REVIEW for Vtiger 6.0

LinkType	Type of Link like - DETAILVIEW : Will add a link in the 'More Actions' menu on the Detail View of the record. DETAILVIEWBASIC : Will add a link to the 'Actions' list on the Detail View of the Record. DETAILVIEWWIDGET : Will add a widget on the right hand side of the Detail View of the Record, similar to Tag Cloud widget. LISTVIEW : Will add a link under the 'More Actions' button on the List View of a module. LISTVIEWBASIC : Will add a button on the List View of the module similar to Delete, Mass Edit buttons.
LinkLabel	Label to use for the link when displaying
LinkURL	URL of the link. You can use variables like $variablename$

In module's ListView handler page (modules/Payslip/ListView.php) you will need this piece of code (before the call to $smarty->display()) :

```
include_once('vtlib/Vtiger/Link.php');
$customlink_params = Array('MODULE' => $currentModule,
'ACTION'=>
vtlib_purify($_REQUEST['action']), 'CATEGORY' =>
$category);
$smarty->assign('CUSTOM_LINKS',
Vtiger_Link::getAllByType(getTabid($currentModule),
Array('LISTVIEW', 'LISTVIEWBASIC'),
$customlink_params));
```

In module's DetailView handler page (modules/Payslip/DetailView.php)

you will need this piece of code (before the call to $smarty->display()) :

```
include_once('vtlib/Vtiger/Link.php');
$customlink_params = Array('MODULE' => $currentModule,
'RECORD' => $focus->id, 'ACTION'=>
vtlib_purify($_REQUEST['action']));
$smarty->assign('CUSTOM_LINKS',
Vtiger_Link::getAllByType(getTabid($currentModule),
Array('DETAILVIEW', 'DETAILVIEWBASIC',
'DETAILVIEWWIDGET'), $customlink_params));
```

Note

The $MODULE$, $ACTION$ and $RECORD$ variables in the LinkURL, will be replaced with the values set through DetailView.php The $MODULE$, $ACTION$, $CATEGORY$ variables in the LinkURL, will be replaced with the values set through ListView.php

Special LinkType

Following LinkTypes are treated specially while processing for display:

LinkType	Description
HEADERSCRIPT	The link will be treated as a javascript type and will be imported in the head section of the HTML output page as <script type='text/javascript' src='linkurl'></script>
HEADERCSS	The link will be treated as a CSS type and will be imported in the head section of the HTML output page as <link rel='stylesheet' type='text/css' href='linkurl>
HEADERLINK	You can see these link grouped under More on the top header panel. Useful if you want to provide utility tools like Bookmarklet etc.

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.

Previous topic

Module Link

Next topic

Package Import

Package Import

Tip
Use Module Manager

Previous topic

Package Export

Next topic

Language Export

Language Export

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

```
require_once('vtlib/Vtiger/Module.php');
require_once('vtlib/Vtiger/Language.php');
$language = new Vtiger_Language();
$language->export('<LanguagePrefix>', '<DESTINATION
DIR>', '<ZIPFILE NAME>', <DIRECT DOWNLOAD>);
```

<LanguageCode_CountryCode>	LanguagePrefix (LanguageCode_CountryCode) of Language 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/Module.php');
require_once('vtlib/Vtiger/Language.php');
$language = new Vtiger_Language();
$language->export(
      'en_us',
      'test/vtlib',
      'en_us-Export.zip',
      true
);
```

Note

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

Previous topic
Package Import
Next topic
Console Tool
Quick search
Go
Enter search terms or a module, class or function name.

Console Tool

This CLI (command line interface) tool will assist you to get the skeleton implementation of extensions for Vtiger.

Jump Start

```
php -f vtlib/tools/console.php
```

```
Have a good time. Press CTRL+C to "quit".
Choose the options below:
1. Create New Module.
2. Create New Layout.
3. Create New Language Pack.
4. Create Test Language Pack.
5. Import Module.
6. Update Module.
7. Remove Module.
Enter your choice:
```

Create New Module

```
Enter your choice: 1
>>> MODULE <<<
Enter module name: MySandboxModule
Entity field (Name): name
Creating ...DONE.
```

Create New Layout

Attention

Experimental

Create New Language Pack

```
Enter your choice: 3
>>> LANGUAGE <<<
Enter (languagecode_countrycode): kn_in
Creating ...DONE.
```

Import Module

```
Enter package path: /Downloads/OthModule.zip
Importing ...DONE.
```

Note

php -f vtlib/tools/console.php -- --import=/Downloads/OthModule.zip

Update Module

```
Enter package path: /Downloads/OthModule-v2.zip
Updating ...DONE.
```

Note

php -f vtlib/tools/console.php -- --update=/Downloads/OthModule-v2.zip

Remove Module

```
Enter package path: /Downloads/OthModule.zip
Removing ...DONE.
```

Note

php -f vtlib/tools/console.php -- --remove=OthModule

Hello World

This example will walk you through the steps of developing a simple “Hello World” extension on Vtiger 6 Framework.

Pre-requisites

- Vtiger 6 installed on your server (assuming it would be used for development).
- PHP CLI is setup to be invoked from the command prompt or terminal.
- Gone through module development docs: *Develop Extensions For Vtiger*

Terminology

- <vtigercrm> - root-directory / document root where Vtiger 6 is installed.
- http://<vtigercrm> - web entry access.

Note

You can download “HelloWorld-v1.zip” from here.

Step 1: Create files

Switch to your working directory (say.. Desktop) and create the following files.

```
manifest.xml
modules/HelloWorld/HelloWorld.php
modules/HelloWorld/views/List.php
languages/en_us/HelloWorld.php
templates/List.tpl
```

Step 2: manifest.xml

Edit manifest.xml and fill in basic information required for getting the extension installed in Vtiger.

```
<?xml version="1.0"?>
<module>
        <type>extension</type>
        <name>HelloWorld</name>
        <label>Hello World</label>
        <parent>Tools</parent>
```

```
        <version>1.0</version>
        <dependencies>
                <vtiger_version>6.0.0</vtiger_version>
<vtiger_max_version>6.*</vtiger_max_version>
        </dependencies>
</module>
```

Step 3: Module Class

Update the module class file (modules/HelloWorld/HelloWorld.php).

```
<?php
class HelloWorld {}
```

Note

It is recommended to have this file - although its blank will be useful later.

Step 4: View Class

Edit modules/HelloWorld/views/List.php

```
<?php

class HelloWorld_List_View extends Vtiger_Index_View {

        public function process(Vtiger_Request $request)
{
                $viewer = $this->getViewer($request);
                $viewer->view('List.tpl', $request-
>getModule());
        }

}
```

Note

Default entry for the module from the menu is set to List View. We can customize this for every module.

Step 5: View template

Edit templates/List.tpl

```
<h1>{'Hello World'|vtranslate:$MODULE}</h1>
<h4>{'LBL_WELCOME'|vtranslate:$MODULE}</h4>
```

Step 6: Default i18n

Edit languages/en_us/HelloWorld.php

```
<?php
```

```
$languageStrings = array(
        'Hello World' => 'Hello World!',
        'LBL_WELCOME' => 'A very warm welcome to you'
);
```

Note

Add all the string constants used by the module - required for i18n support.

Step 7: Package files

zip -r HelloWorld-v1.zip *manifest.xml modules/HelloWorld/HelloWorld.php modules/HelloWorld/views/List.php languages/en_us/HelloWorld.php templates/List.tpl*

Refer to *Package Structure*

Step 8: Install in Vtiger

You can import through Module Manager UI.

Note

Importing through CLI script is given below.

```
cd <vtigercrm>
php -f ImportHelloWorld.php
```

ImportHelloWorld.php should use vtlib API to import the package.

```
<?php

require_once 'vtlib/Vtiger/Module.php';
require_once 'vtlib/Vtiger/Package.php';

$Vtiger_Utils_Log = true;

$package = new Vtiger_Package();
$package->import('/path/to/HelloWorld-v1.zip'); // Can be any-where on your server.
```

The module is now installed and ready, you can find it under All > Tools section.

World Clock

This example will walk you through the steps of developing a simple "World Clock" extension on Vtiger 6 Framework. It introduces you to the way of injecting client-side libraries & JSON APIs.

Pre-requisites

- Vtiger 6 installed on your server (assuming it would be used for development).
- PHP CLI is setup to be invoked from the command prompt or terminal.
- Gone through module development docs: *Develop Extensions For Vtiger*

Terminology

- <vtigercrm> - root-directory / document root where Vtiger 6 is installed.
- http://<vtigercrm> - web entry access.

Note

You can download "WorldClock-v1.zip" from here.

Step 1: Create files

Switch to your working directory (say.. Desktop) and create the following files.

```
manifest.xml
modules/WorldClock/WorldClock.php
modules/WorldClock/views/List.php
templates/ListViewContents.tpl
templates/ListViewSidebar.tpl
templates/resources/List.js
templates/resources/lib/
languages/en_us/WorldClock.php
```

Step 2: jQuery Clock Plugin

Download jquery-clock-plugin and unzip into `templates/resources/lib/`

Step 3: manifest.xml

Previous topic

Hello World

Next topic

City Lookup

Edit manifest.xml and fill in basic information required for getting the extension installed in Vtiger.

```
<?xml version="1.0"?>
<module>
        <type>extension</type>
        <name>WorldClock</name>
        <label>World Clock</label>
        <parent>Tools</parent>
        <version>1.0</version>
        <dependencies>
                <vtiger_version>6.0.0</vtiger_version>

<vtiger_max_version>6.*</vtiger_max_version>
        </dependencies>
</module>
```

Step 4: Module Class

Update the module class file (modules/WorldClock/WorldClock.php).

```
<?php
class WorldClock {}
```

Note

It is recommended to have this file - although its blank will be useful later.

Step 5: View Class

Edit modules/WorldClock/views/List.php

```
<?php

class WorldClock_List_View extends Vtiger_Index_View {

        public function process(Vtiger_Request $request)
{
                $viewer = $this->getViewer($request);
                $viewer->view('ListViewContents.tpl',
$request->getModule());
        }

        // Register loading client-scripts for the UI
        public function getHeaderScripts(Vtiger_Request
$request) {
                $allJS =
parent::getHeaderScripts($request);
                $moduleName = $request->getModule();

                // dot.separated paths will be resolved
with respect to active layout
                // example: layouts/vlayout/...
                $jsFileNames = array(
                        "modules.Vtiger.resources.List",

"modules.$moduleName.resources.List"
                );
```

```
                $newJS = $this->checkAndConvertJsScripts($jsFileNames);
                $allJS = array_merge($allJS, $newJS);
                return $allJS;
        }

}
```

Note

Default entry for the module from the menu is set to List View. We can customize this for every module.

Step 6: View template

Edit templates/ListViewContents.tpl

```
<div style="text-align: center;" id="timezone-clocks" class="row-fluid"></div>
```

Edit templates/ListViewSidebar.tpl (Overrides default template)

```
<!-- Direct inclusion of library files with reference to layout -->

<link rel="stylesheet" href="layouts/vlayout/modules/WorldClock/resources/lib/jquery-clock-plugin/css/analog.css" type="text/css" />
<script src="layouts/vlayout/modules/WorldClock/resources/lib/jquery-clock-plugin/js/jquery.clock.js"></script>

<div style="text-align: center">
        <ul id="worldclock-on-sidebar" class="analog" data-tz="{$CURRENT_USER->name}">
          <li class="hour"></li>
          <li class="min"></li>
          <li class="sec"></li>
          <li class="meridiem"></li>
        </ul>
</div>
```

Note

Module specific template `ListViewSidebar.tpl` will be included by framework automatically by preProcess method of Vtiger_List_View.

Step 7: View Javascript

Edit templates/resources/List.js - (`extend Vtiger List jQuery Class`)

```
Vtiger_List_Js.extend('WorldClock_List_Js', {}, {

        // Default function invoked by Vtiger Client Framework on specific View
        registerEvents: function() {
```

```
                var self = this;
                this.initializeSidebarClock(function(){

self.initializeTimezoneClocks();
                });
        },

        initializeSidebarClock: function(next) {
                var offset = (new
Date()).getTimezoneOffset() * -1 / 60.0;
                jQuery('#worldclock-on-
sidebar').clock({offset: offset});
                next();
        },

        initializeTimezoneClocks: function() {
                var self = this;

                jQuery.ajax({
                        dataType: 'jsonp',
                        url:
'http://gomashup.com/json.php?
fds=geo/timezone/locations&jsoncallback=?',
                        success: function(response) {
                                var timezones =
response.result;

self._createClockUIForTimezones(timezones);
                        }
                });
        },

        _createClockUIForTimezones: function(timezones)
{
                var self = this;
                if (timezones.length) {

self._createClockUIForTimezone(timezones.shift(),
function(){

self._createClockUIForTimezones(timezones);
                        });
                }
        },

        _createClockUIForTimezone: function(timezone,
next) {
                var clockUI = jQuery(
                        '<div class="span4
marginLeftZero"><ul class="analog"><li
class="hour"></li><li class="min"></li><li
class="sec"></li><li
class="meridiem"></li></ul><label></label></div>'
                );
                var container = jQuery('#timezone-
clocks');
                jQuery('label',
clockUI).html(timezone.TimeZoneId.replace('-', '/'));
                container.append(clockUI);

                clockUI.clock({offset: timezone.GMT});
                next();
        }
});
```

Step 8: Default i18n

Edit languages/en_us/HelloWorld.php

```
<?php

$languageStrings = array(
        'World Clock' => 'World Clock'
);
```

Note

Add all the string constants used by the module - required for i18n support.

Step 9: Package files

zip -r WorldClock-v1.zip *manifest.xml modules/WorldClock/WorldClock.php modules/WorldClock/views/List.php templates/ListViewContents.tpl templates/ListViewSidebar.tpl templates/resources/List.js templates/resources/lib languages/en_us/WorldClock.php*

Refer to *Package Structure*

Step 10: Install in Vtiger

You can import through Module Manager UI.

The module is now installed and ready, you can find it under All > Tools section.

City Lookup

This example will walk you through the steps of developing a module “City Lookup” extension on Vtiger 6 Framework. It introduces you to the way of manipulating the UI element of other module based on Client side.

Pre-requisites

- Vtiger 6 installed on your server (assuming it would be used for development).
- PHP CLI is setup to be invoked from the command prompt or terminal.
- Gone through module development docs: *Develop Extensions For Vtiger*

Terminology

- <vtigercrm> - root-directory / document root where Vtiger 6 is installed.
- http://<vtigercrm> - web entry access.

Note

You can download “CityLookup-v1.zip” from here.

Step 1: Create files

Switch to your working directory (say.. Desktop) and create the following files.

```
manifest.xml
modules/CityLookup/CityLookup.php
modules/CityLookup/views/List.php
modules/CityLookup/js/CityLookupAutofill.js
languages/en_us/CityLookup.php
```

Step 2: manifest.xml

Edit manifest.xml and fill in basic information required for getting the extension installed in Vtiger.

```
<?xml version="1.0"?>
<module>
        <type>extension</type>
        <name>CityLookup</name>
```


Previous topic

World Clock

Next topic

An Entity Module Example

```
        <label>City Lookup</label>
        <parent>Tools</parent>
        <version>1.0</version>
        <dependencies>
                <vtiger_version>6.0.0</vtiger_version>
<vtiger_max_version>6.*</vtiger_max_version>
        </dependencies>
</module>
```

Step 3: Module Class

Update the module class file (modules/CityLookup/CityLookup.php).

```
<?php

class CityLookup {

        public function vtlib_handler($moduleName,
$eventType) {
                if ($eventType == 'module.postinstall')
{
                        $this-
>_registerLinks($moduleName);
                } else if ($eventType ==
'module.enabled') {
                        $this-
>_registerLinks($moduleName);
                } else if ($eventType ==
'module.disabled') {
                        $this-
>_deregisterLinks($moduleName);
                }
        }

        protected function _registerLinks($moduleName) {
                $thisModuleInstance =
Vtiger_Module::getInstance($moduleName);
                if ($thisModuleInstance) {
                        $thisModuleInstance-
>addLink("HEADERSCRIPT", "City Autofill",
"modules/CityLookup/js/CityLookupAutofill.js");
                }
        }

        protected function _deregisterLinks($moduleName)
{
                $thisModuleInstance =
Vtiger_Module::getInstance($moduleName);
                if ($thisModuleInstance) {
                        $thisModuleInstance-
>deleteLink("HEADERSCRIPT", "City Autofill",
"modules/CityLookup/js/CityLookupAutofill.js");
                }
        }
}
```

Note

vtlib_handler is invoked by ModuleManager with specific $eventType. The javascript is registered / deregistered accordingly.

Step 4: View Class

Edit modules/CityLookup/views/List.php - nothing much just a placeholder in this case.

```
<?php

class CityLookup_List_View extends Vtiger_Index_View {

        public function process(Vtiger_Request $request)
{
                echo "<center><h1>Welcome to
CityLookup<h1>".
                "<h4>The support is enabled through
Javascript for Leads & Contacts!</h4></center>";
        }
}
```

Step 5: Javascript Autofill

Edit modules/CityLookup/js/CityLookupAutofill.js

```
jQuery(function() {

        // Pre-configure list of cities.
        var cities = [
                "New York", "Los Angeles", "Chicago",
"Houston", "Philadelphia",
                "Phoenix", "San Diego", "San Antonio",
"Dallas", "Detroit", "Other"
        ]

        // Enable auto-fill editview / detailview
(ajaxedit)
        var activeModule = app.getModuleName(),
activeView = app.getViewName();
        if (activeView == 'Edit' || activeView ==
'Detail') {
                // For target module
                if (activeModule == 'Leads' ||
activeModule == 'Contacts') {
                        // For target field.
                        var fieldName = 'city';
                        var field =
jQuery("#"+activeModule+"_editView_fieldName_"+fieldName
);
                        field.autocomplete({
                                source: cities
                        });
                }
        }

});
```

Step 8: Default i18n

Edit languages/en_us/CityLookup.php

```
<?php
```

```
$languageStrings = array(
        'City Lookup' => 'City Lookup'
);
```

Note

Add all the string constants used by the module - required for i18n support.

Step 9: Package files

zip -r CityLookup-v1.zip *manifest.xml modules/CityLookup/CityLookup.php modules/CityLookup/views/List.php modules/CityLookup/js/CityLookupAutofill.js languages/en_us/CityLookup.php*

Refer to *Package Structure*

Step 10: Install in Vtiger

You can import through Module Manager UI.

The module is now installed and ready, you can find it under All > Tools section.

An Entity Module Example

Requirements

User should be able to track his or others expenses.

Design

- Create record with amount, date of expenditure.
- User to whom the record belongs could be assumed as the person who did incur an expense.

Implementation

Expense Information			
Summary	InputBox	Expense On	DateInput
Custom Information			
Description	Textarea		

Field Structure

FieldLabel	FieldName	TableName	UIType	Description
Summary	expensesummary	vtiger_expense	2	*Entity Field
Expense On	expenseon	vtiger_expenses	70	Date & Time
Amount	expenseamount	vtiger_expenses	1	Amount of expense incurred.
Assigned To	assigned_user_id	vtiger_crmentity	53	Who made the expense.
Description	description	vtiger_crmentity	19	Details on the expense made.

Getting Started

Create the bootstrap vtlib script in root folder of Vtiger to activate the module entry.

Previous topic

City Lookup

Next topic

Slideshow

```php
<?php
include_once 'vtlib/Vtiger/Module.php';

$Vtiger_Utils_Log = true;

$MODULENAME = 'Expenses';

$moduleInstance =
Vtiger_Module::getInstance($MODULENAME);
if ($moduleInstance ||
file_exists('modules/'.$MODULENAME)) {
        echo "Module already present - choose a
different name.";
} else {
        $moduleInstance = new Vtiger_Module();
        $moduleInstance->name = $MODULENAME;
        $moduleInstance->parent= 'Tools';
        $moduleInstance->save();

        // Schema Setup
        $moduleInstance->initTables();

        // Field Setup
        $block = new Vtiger_Block();
        $block->label = 'LBL_'.
strtoupper($moduleInstance->name) . '_INFORMATION';
        $moduleInstance->addBlock($block);

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

        $field1  = new Vtiger_Field();
        $field1->name = 'summary';
        $field1->label= 'Summary';
        $field1->uitype= 2;
        $field1->column = $field1->name;
        $field1->columntype = 'VARCHAR(255)';
        $field1->typeofdata = 'V~M';
        $block->addField($field1);

        $moduleInstance->setEntityIdentifier($field1);

        $field2  = new Vtiger_Field();
        $field2->name = 'expenseon';
        $field2->label= 'Expense On';
        $field2->uitype= 5;
        $field2->column = $field2->name;
        $field2->columntype = 'Date';
        $field2->typeofdata = 'D~O';
        $block->addField($field2);

        $field3  = new Vtiger_Field();
        $field3->name = 'expenseamount';
        $field3->label= 'Amount';
        $field3->uitype= 71;
        $field3->column = $field3->name;
        $field3->columntype = 'VARCHAR(255)';
        $field3->typeofdata = 'V~M';
        $block->addField($field3);

        $field3  = new Vtiger_Field();
        $field3->name = 'description';
        $field3->label= 'Description';
        $field3->uitype= 19;
        $field3->column = 'description';
        $field3->table = 'vtiger_crmentity';
        $blockcf->addField($field3);
```

```
        // Recommended common fields every Entity module
should have (linked to core table)
        $mfield1 = new Vtiger_Field();
        $mfield1->name = 'assigned_user_id';
        $mfield1->label = 'Assigned To';
        $mfield1->table = 'vtiger_crmentity';
        $mfield1->column = 'smownerid';
        $mfield1->uitype = 53;
        $mfield1->typeofdata = 'V~M';
        $block->addField($mfield1);

        $mfield2 = new Vtiger_Field();
        $mfield2->name = 'createdtime';
        $mfield2->label= 'Created Time';
        $mfield2->table = 'vtiger_crmentity';
        $mfield2->column = 'createdtime';
        $mfield2->uitype = 70;
        $mfield2->typeofdata = 'DT~O';
        $mfield2->displaytype= 2;
        $block->addField($mfield2);

        $mfield3 = new Vtiger_Field();
        $mfield3->name = 'modifiedtime';
        $mfield3->label= 'Modified Time';
        $mfield3->table = 'vtiger_crmentity';
        $mfield3->column = 'modifiedtime';
        $mfield3->uitype = 70;
        $mfield3->typeofdata = 'DT~O';
        $mfield3->displaytype= 2;
        $block->addField($mfield3);

        /* NOTE: Vtiger 7.1.0 onwards */
        $mfield4 = new Vtiger_Field();
        $mfield4->name = 'source';
        $mfield4->label = 'Source';
        $mfield4->table = 'vtiger_crmentity';
        $mfield4->displaytype = 2; // to disable field
in Edit View
        $mfield4->quickcreate = 3;
        $mfield4->masseditable = 0;
        $block->addField($mfield4);

        $mfield5 = new Vtiger_Field();
        $mfield5->name = 'starred';
        $mfield5->label = 'starred';
        $mfield5->table = 'vtiger_crmentity_user_field';
        $mfield5->displaytype = 6;
        $mfield5->uitype = 56;
        $mfield5->typeofdata = 'C~O';
        $mfield5->quickcreate = 3;
        $mfield5->masseditable = 0;
        $block->addField($mfield5);

        $mfield6 = new Vtiger_Field();
        $mfield6->name = 'tags';
        $mfield6->label = 'tags';
        $mfield6->displaytype = 6;
        $mfield6->columntype = 'VARCHAR(1)';
        $mfield6->quickcreate = 3;
        $mfield6->masseditable = 0;
        $block->addField($mfield6);
        /* End 7.1.0 */

        // Filter Setup
        $filter1 = new Vtiger_Filter();
        $filter1->name = 'All';
        $filter1->isdefault = true;
```

```
        $moduleInstance->addFilter($filter1);
        $filter1->addField($field1)->addField($field2,
1)->addField($field3, 2)->addField($mfield1, 3);

        // Sharing Access Setup
        $moduleInstance->setDefaultSharing();

        // Webservice Setup
        $moduleInstance->initWebservice();

        mkdir('modules/'.$MODULENAME);
        echo "OK\n";
}
```

Directory Structure

vtigercrm/

modules/

Expenses/

Expenses.php - `class Expenses`

languages/

en_us/

Expenses.php

Code

Expenses.php

```
<?php

include_once 'modules/Vtiger/CRMEntity.php';

class Expenses extends Vtiger_CRMEntity {
        var $table_name = 'vtiger_expenses';
        var $table_index= 'expensesid';

        var $customFieldTable =
Array('vtiger_expensescf', 'expensesid');

        var $tab_name = Array('vtiger_crmentity',
'vtiger_expenses', 'vtiger_expensescf');

        var $tab_name_index = Array(
                'vtiger_crmentity' => 'crmid',
                'vtiger_expenses' => 'expensesid',
                'vtiger_expensescf'=>'expensesid');

        var $list_fields = Array (
                /* Format: Field Label =>
Array(tablename, columnname) */
                // tablename should not have prefix
'vtiger_'
                'Summary' => Array('expenses',
'summary'),
                'Assigned To' =>
Array('crmentity','smownerid')
```

```
        );
        var $list_fields_name = Array (
                /* Format: Field Label => fieldname */
                'Summary' => 'summary',
                'Assigned To' => 'assigned_user_id',
        );

        // Make the field link to detail view
        var $list_link_field = 'summary';

        // For Popup listview and UI type support
        var $search_fields = Array(
                /* Format: Field Label =>
Array(tablename, columnname) */
                // tablename should not have prefix
'vtiger_'
                'Summary' => Array('expenses',
'summary'),
                'Assigned To' =>
Array('vtiger_crmentity','assigned_user_id'),
        );
        var $search_fields_name = Array (
                /* Format: Field Label => fieldname */
                'Summary' => 'summary',
                'Assigned To' => 'assigned_user_id',
        );

        // For Popup window record selection
        var $popup_fields = Array ('summary');

        // For Alphabetical search
        var $def_basicsearch_col = 'summary';

        // Column value to use on detail view record
text display
        var $def_detailview_recname = 'summary';

        // Used when enabling/disabling the mandatory
fields for the module.
        // Refers to vtiger_field.fieldname values.
        var $mandatory_fields =
Array('summary','assigned_user_id');

        var $default_order_by = 'summary';
        var $default_sort_order='ASC';
}
```

Internationalization

languages/en_us/Expenses.php

Distribution

You can achieve through *Package Export* API

Deployment

Go to Module Manager Import the module.

You can achieve through *Package Import* API

Slideshow

This example will walk you through the use of *Server APIs* (accessing records) of Vtiger 6 Framework.

Note

Server APIs should be used when you are accessing records of any module. It takes care of retrieval with access control (Sharing access & Profile access) setup for the logged in user.

Important

Using direct SQL queries for retrieving records may open-doors for users to gain access to records that they do not have permission to view.

Pre-requisites

- Vtiger 6 installed on your server (assuming it would be used for development).
- PHP CLI is setup to be invoked from the command prompt or terminal.
- Gone through module development docs: *Develop Extensions For Vtiger*

Terminology

- <vtigercrm> - root-directory / document root where Vtiger 6 is installed.
- http://<vtigercrm> - web entry access.

Note

You can download “Slideshow-v1.zip” from here.

Step 1: Create files

Switch to your working directory (say.. Desktop) and create the following files.

```
manifest.xml
modules/Slideshow/Slideshow.php
modules/Slideshow/views/List.php
modules/Slideshow/js/SlideshowShared.js
modules/Slideshow/js/remarkjs.min.js
languages/en_us/Slideshow.php
templates/ListViewContents.tpl
```

Step 2: manifest.xml

Edit manifest.xml and fill in basic information required for getting the extension installed in Vtiger.

```
<?xml version="1.0"?>
<module>
        <type>extension</type>
        <name>Slideshow</name>
        <label>Slideshow</label>
        <parent>Tools</parent>
        <version>1.0</version>
        <dependencies>
                <vtiger_version>6.0.0</vtiger_version>

<vtiger_max_version>6.*</vtiger_max_version>
        </dependencies>
</module>
```

Step 3: Module Class

Update the module class file (modules/Slideshow/Slideshow.php).

```
<?php

class Slideshow {

        public function vtlib_handler($moduleName,
$eventType) {
                if ($eventType == 'module.postinstall')
{
                        $this-
>_registerLinks($moduleName);
                } else if ($eventType ==
'module.enabled') {
                        $this-
>_registerLinks($moduleName);
                } else if ($eventType ==
'module.disabled') {
                        $this-
>_deregisterLinks($moduleName);
                }
        }

        protected function _registerLinks($moduleName) {
                $thisModuleInstance =
Vtiger_Module::getInstance($moduleName);
                if ($thisModuleInstance) {
                        $thisModuleInstance-
>addLink("HEADERSCRIPT", "SlideshowShared",

"modules/Slideshow/js/SlideshowShared.js");

                        $leadsModuleInstance =
Vtiger_Module::getInstance('Leads');
                        $leadsModuleInstance-
>addLink("LISTVIEW", "Slideshow",

'javascript:Slideshow.viewSelected();');

                        $contactsModuleInstance =
```

```
Vtiger_Module::getInstance('Contacts');
                        $contactsModuleInstance-
>addLink("LISTVIEW", "Slideshow",

'javascript:Slideshow.viewSelected();');
                }
        }

        protected function _deregisterLinks($moduleName)
{
                $thisModuleInstance =
Vtiger_Module::getInstance($moduleName);
                if ($thisModuleInstance) {
                        $thisModuleInstance-
>deleteLink("HEADERSCRIPT", "SlideshowShared",

"modules/Slideshow/js/SlideshowShared.js");

                        $leadsModuleInstance =
Vtiger_Module::getInstance('Leads');
                        $leadsModuleInstance-
>deleteLink("LISTVIEW", "Slideshow");

                        $contactsModuleInstance =
Vtiger_Module::getInstance('Contacts');
                        $contactsModuleInstance-
>deleteLink("LISTVIEW", "Slideshow");
                }
        }
}
```

Note

vtlib_handler is invoked by ModuleManager with specific $eventType. The javascript & links on Leads & Contacts module is registered / deregistered accordingly.

Step 4: View Class

Edit modules/Slideshow/views/List.php - showcases retrieval of target module records using *Server APIs* (vtws_query)

```
<?php

/* Include dependency required for using Server API */
include_once 'include/Webservices/Query.php';

class Slideshow_List_View extends Vtiger_Index_View {

        public function process(Vtiger_Request $request)
{

                // Current User Context
                $userContext = vglobal('current_user');
                $viewer = $this->getViewer($request);

                if ($request->has('ids') && $request-
>has('src_module')) {
                        // Was request made for the
slideshow - lets act now.

                        // Prepare Webservices Query
                        $srcModule = $request-
```

```
>get('src_module');
                        $ids = array_map('intval',
$request->get('ids'));
                        $idsuffix =
trim(vtws_getWebserviceEntityId($srcModule, '0'), '0');
                        $wsids = array(); foreach ($ids
as $id) $wsids[] = $idsuffix.$id;

                        $wsquery = sprintf("SELECT *
FROM %s WHERE id IN ('%s');",
                                $srcModule,
implode("','", $wsids) );

                        // Execute the query and fetch
records
                        $records = vtws_query($wsquery,
$userContext);

                        // Pass the records for
rendering the slideshow.
                        $viewer->assign('RECORDS',
$records);
                        $viewer->assign('SRCMODULE',
$srcModule);
                        $viewer->assign('COUNT',
count($records));
                }

                $viewer->view('ListViewContents.tpl',
$request->getModule());
        }
}
```

Step 5: Javascript Hook

Edit modules/Slideshow/js/SlideshowShared.js

```
jQuery.Class('Slideshow', {
        viewSelected: function() {
                var listView    =
Vtiger_List_Js.getInstance();
                var nothingSelected =
listView.checkListRecordSelected();
                if (nothingSelected) {
                        alert('Nothing selected!');
                } else {
                        var selectedIds =
listView.readSelectedIds(false);
                        var excludedIds =
listView.readExcludedIds(false);
                        if (selectedIds.length) {
                                window.open("index.php?
module=Slideshow&view=List&ids="+

JSON.stringify(selectedIds)+"&src_module="+app.getModule
Name());
                        }
                }
        }
}, {});
```

Step 6: Slideshow Renderer

Copy http://gnab.github.io/remark/downloads/remark-latest.min.js to modules/Slideshow/js/remark.min.js

Step 7: View Template

Edit templates/ListViewContents.tpl

```
<br>
{if !$RECORDS}
        <center><h1>Welcome to Slideshow<h1>
        <h4>The support is enabled through Javascript
for Leads & Contacts!</h4></center>
{else}
<script type="text/javascript"
src="modules/Slideshow/js/remark.min.js"></script>
<style type="text/css">
.remark-slide h1 { font-size: 30px; }
.remark-slide em { font-size: 18px; }
#page { display: none; }
</style>

<!-- Prepare remarkjs friendly source -->
<textarea id="source">
class: center, middle
# {$SRCMODULE} ( {$COUNT} )
---
{foreach item=RECORD from=$RECORDS}
Field | Value
---- | ---
{foreach key=FIELDNAME item=FIELDVALUE from=$RECORD}
{if $FIELDVALUE}
{$FIELDNAME} | {$FIELDVALUE}
{/if}
{/foreach}
---
{/foreach}

class: center, middle
# Thank you
</textarea>

<!-- Trigger rendering on the client-side -->
<script type="text/javascript">
        var slideshow = remark.create();
        jQuery(function(){
                jQuery('table').addClass('table table-
bordered');
        });
</script>
{/if}
```

Step 8: Default i18n

Edit languages/en_us/Slideshow.php

```
<?php

$languageStrings = array(
        'Slideshow' => 'Slideshow'
);
```

Note

Add all the string constants used by the module - required for i18n support.

Step 9: Package files

zip -r Slideshow-v1.zip *manifest.xml modules/Slideshow/Slideshow.php modules/Slideshow/views/List.php modules/Slideshow/js/SlideshowShared.js modules/Slideshow/js/remarkjs.min.js languages/en_us/Slideshow.php templates/ListViewContents.tpl*

Refer to *Package Structure*

Step 10: Install in Vtiger

You can import through Module Manager UI.

The module is now installed and ready, you can find it under All > Tools section.

Navigate to Leads / Contacts ListView > Select Desired Records > Click Slideshow under Actions

An Extension Module Example

Requirements

User should be able to add new RSS feed and read the existing feed.

Design

Server Side

- List view of existing RSS feed.
- Save action for new RSS feed.

Client Side

- Enable option to click on the RSS feed and load the items.
- Provide option to add new RSS feed - accept the URL and title.

Implementation

Getting Started

Create the bootstrap vtlib script in root folder of Vtiger to activate the module entry.

```
include_once 'vtlib/Vtiger/Module.php';
$Vtiger_Utils_Log = true;

$MODULENAME = 'MyRss';

$moduleInstance =
Vtiger_Module::getInstance($MODULENAME);
if ($moduleInstance ||
file_exists('modules/'.$MODULENAME)) {
   echo "Module already present - choose a different
name.";
} else {
   $moduleInstance = new Vtiger_Module();
   $moduleInstance->name = $MODULENAME;
   $moduleInstance->parent= 'Tools';
   $moduleInstance->save();

   mkdir('modules/'.$MODULENAME);
   echo "OK\n";
}
```

Schema

```
Vtiger_Utils::CreateTable('vtiger_myrss',
```


Previous topic

Slideshow

Next topic

Layout i386

```
'(id integer not null auto_increment, url varchar(255),
title varchar(50), primary key (id))');
```

Directory Structure

- vtigercrm/
  - modules/
    - MyRss/
      - *schema.xml*
      - *MyRss.php* - `class MyRss`
      - views/
        - *views/List.php* - `class MyRss_List_View`
      - models/
        - *models/Record.php* - `class MyRss_Record_Model`
      - actions/
        - *actions/Save.php* - `class MyRss_Save_Action`
      - layouts/
        - vlayout/
          - modules/
            - MyRss/
              - *SideBar.tpl*
              - *Index.tpl*
              - resources/
                - *resources/MyRss.js* - `jQuery.Class MyRss_List_Js`
              - jquery_rss_min.js

Code

schema.xml

```
<?xml version="1.0"?>
<schema>
    <tables>
```

```
        <table>vtiger_myrss</table>
    </tables>
</schema>
```

MyRss.php

```
<?php

class MyRss {
        function vtlib_handler($moduleName, $eventType)
{
                if ($eventType == 'module.postinstall')
{
                        $data = array('url' =>
'https://www.vtiger.com/blogs/?feed=rss2',
                                        'title' => 'Vtiger
Blogs');

MyRss_Record_Model::create($data);
                }
        }
}
```

views/List.php

```
<?php
class MyRss_List_View extends Vtiger_Index_View {
        // We are overriding the default SideBar UI to
list our feeds.
        public function preProcess(Vtiger_Request
$request, $display = true) {
                $feeds = MyRss_Record_Model::findAll();
                $viewer = $this->getViewer($request);
                $viewer->assign('FEEDS', $feeds);
                return parent::preProcess($request,
$display);
        }

        // Injecting custom javascript resources
        public function getHeaderScripts(Vtiger_Request
$request) {
                $headerScriptInstances =
parent::getHeaderScripts($request);
                $moduleName = $request->getModule();
                $jsFileNames = array(

"modules.$moduleName.resources.jquery_rss_min", // . =
delimiter
                );
                $jsScriptInstances = $this-
>checkAndConvertJsScripts($jsFileNames);
                $headerScriptInstances =
array_merge($headerScriptInstances, $jsScriptInstances);
                return $headerScriptInstances;
        }
}
```

models/Record.php

```
<?php
class MyRss_Record_Model extends Vtiger_Base_Model {
        public function save() {
                $db = PearDatabase::getInstance();
                $db->pquery('INSERT INTO vtiger_myrss
(url, title) VALUES(?,?)', array(
                        $this->get('url'), $this-
>get('title')
                ));
                return $db->getLastInsertID();
        }
        static function create($data) {
                $instance =
self::findWithUrl($data['url']);
                if ($instance) {
                        throw new Exception('Duplicate
Feed');
                }
                $instance = new self($data);
                return $instance->save();
        }
        static function findWithUrl($url) {
                $db = PearDatabase::getInstance();
                $rs = $db->pquery('SELECT * FROM
vtiger_myrss WHERE url=?', array($url));
                return $db->num_rows($rs)? new
self($db->fetch_array($rs)) : NULL;
        }
        static function findAll() {
                $db = PearDatabase::getInstance();
                $instances = array();
                $rs = $db->pquery('SELECT * FROM
vtiger_myrss ORDER BY id DESC', array());
                if ($db->num_rows($rs)) {
                        while ($data = $db-
>fetch_array($rs)) {
                                $instances[] = new
self($data);
                        }
                }
                return $instances;
        }
}
```

actions/Save.php

```
<?php
class MyRss_Save_Action extends Vtiger_Action_Controller
{
        public function checkPermission() {
                return true;
        }
        public function process(Vtiger_Request $request)
{
                $feedurl = $request->get('url');
                if (empty($feedurl)) {
                        throw new Exception('URL cannot
be empty');
                }

                $data = array('url' => $feedurl,
```

```
'title' => $feedurl);
                MyRss_Record_Model::create($data);

                $response = new Vtiger_Response();
                $response->setResult(array('feed' =>
$data));
                return $response;
        }
}
```

SideBar.tpl

```
<div class="quickWidgetContainer">
        <div class="quickWidget">
                <div class="quickWidgetHeader">
                        <h5 class="pull-left">Your RSS
Feeds</h5>
                        <button class="btn pull-right"
id="RssFeedAdd">+</button>
                        <div class="clearfix"></div>
                </div>
                <div class="widgetContainer collapse
in">
                        <div class="row-fluid">
                                <div class="span10">
                                        <ul class="nav
nav-list">

{foreach item=FEED from=$FEEDS}
                                                <li>
                                                    <a
href="{$FEED->get('url')}"

data-feedurl="{$FEED->get('url')}">

{$FEED->get('title')}</a>
                                                </li>

{/foreach}
                                        </ul>
                                </div>
                        </div>
                </div>
        </div>
</div>
```

Index.tpl

```
<div class="listViewPageDiv">
        <div class="listViewTopMenuDiv">
                <div class="listViewActionsDiv">
                        <div id="RssFeedTitle">
                            Learn more about RSS
                            <a
href="http://en.wikipedia.org/wiki/RSS"
target="_blank">here</a>.
                        </div>
                </div>
        </div>
        <div id="RssFeedContainer"
class="listViewEntriesDiv"
```

```
                        >
                &leftarrow; Selected feed content will
be displayed here.
        </div>
</div>
<div id="RssFeedAddFormTpl" style="display: none;">
        <div class="modelContainer">
                <form method="GET"
action="javascript:;" class="RssFeedAddForm">
                        <div class="modal-header">
                                <a class="close" data-
dismiss="modal">x</a>
                                <h3>New RSS Feed</h3>
                        </div>
                        <div class="modal-body">
                                RSS Feed: <input
type="text" name="url" class="validate[required]">
                        </div>
                        <div class="modal-footer">
                                <a class="cancelLink
cancelLinkContainer pull-right" data-dismiss="modal">

{vtranslate('LBL_CANCEL')}</a>
                                <button class="btn btn-
success" type="submit">Add</button>
                        </div>
                </form>
        </div>
</div>
```

resources/MyRss.js

```
jQuery.Class('MyRss_List_Js', {}, {

        registerEvents: function() {
                this.registerFeedClick();
                this.registerFeedAdd();
        },

        registerFeedClick: function() {
                jQuery('[data-
feedurl]').click(function(e){
                        e.preventDefault();
                        var feedAnchor = jQuery(this);
                        var feedurl =
feedAnchor.data('feedurl');

jQuery('#RssFeedTitle').text(feedAnchor.html() + ' - '
+ feedurl);
                        var options = {
                                limit: 50
                        }

jQuery('#RssFeedContainer').empty().rss(feedurl,
options);
                });
        },

        registerFeedAdd: function() {

jQuery('#RssFeedAdd').click(function(e){
                        var formTplClone =
jQuery('#RssFeedAddFormTpl').clone().css({display:
'block'});
```

```
app.showModalWindow(formTplClone, function(){
                                var targetForm =
jQuery('.RssFeedAddForm', formTplClone);

targetForm.validationEngine(app.validationEngineOptions)
;

targetForm.submit(function(){
                                        if
(!targetForm.validationEngine('validate')) {
                                                return;
                                        }
                                        var params =
'module=MyRss&action=Save&';
                                        params +=
jQuery(targetForm).serialize();

AppConnector.request(params).then(function(response){

window.location.reload();

//app.hideModalWindow();
                                        });
                                });
                        });
                });
        }

});
```

Distribution

You can achieve through *Package Export* API

Deployment

Go to Module Manager Import the module.

You can achieve through *Package Import* API

Layout i386

This example will introduce you to custom layout creation for Vtiger 6.

The steps to create a new layout are illustrated through this exercise of building a 1980's type layout (terminal based UI) for managing Leads in the CRM.

Note

Demo setup - click here (credentials: admin/admin)

Pre-requisites

- Vtiger 6.4 should be installed on your server (assuming you are developing the layout for Vtiger 6.4)
- You must be familiar with content covered in module development docs: *Develop Extensions For Vtiger* and *Internals* (*on UI*)

Terminology

- <vtigercrm> - root-directory / document root where Vtiger 6 is installed.
- http://<vtigercrm> - web entry access.

Note

You can download "i386Layout-v1.zip" from here.

Step 1: Create files

Switch to your working directory (say.. Desktop) and create the following files.

```
manifest.xml
layouts/i386/libraries/jquery/jquery.min.js
layouts/i386/libraries/bootstra.386

layouts/i386/skins/application.css
layouts/i386/skins/application.js

layouts/i386/modules/Vtiger/Header.tpl
layouts/i386/modules/Vtiger/Footer.tpl

layouts/i386/modules/Settings/Vtiger/ConfigEditorDetail.
tpl
layouts/i386/modules/Users/Login.tpl

layouts/i386/modules/Vtiger/dashboards/DashBoardPreProce
```

```
ss.tpl
layouts/i386/modules/Vtiger/dashboards/DashBoardContents
.tpl

layouts/i386/modules/Vtiger/ListViewPreProcess.tpl
layouts/i386/modules/Vtiger/ListViewPostProcess.tpl
layouts/i386/modules/Vtiger/ListViewContents.tpl
```

Step 2: manifest.xml

Edit manifest.xml and fill in basic information required for getting the extension installed in Vtiger.

```
<?xml version="1.0"?>
<module>
        <type>layout</type>
        <name>i386</name>
        <label>i386 UI</label>
        <version>1.0</version>
        <dependencies>
                <vtiger_version>6.0.0</vtiger_version>

<vtiger_max_version>6.*</vtiger_max_version>
        </dependencies>
</module>
```

Step 3: Basic Files

Overview of files that need to be implemented for basic layout to work.

File	Description
layouts/i386/libraries/jquery/jquery.min.js	recommended.
layouts/i386/libraries/bootstra.386	bootstra.386-latest-v2.zip for 1980's look and feel.
layouts/i386/skins/application.css	our custom styles.
layouts/i386/skins/application.js	our ui and data interaction implementation.
layouts/i386/modules/Vtiger/Header.tpl	default pre-process template (unless overridden by the view). preamble (<html>...<body> is recommended here)
layouts/i386/modules/Vtiger/Footer.tpl	default post-process template. epilogue

	(...</body></html> is recommended here)
layouts/i386/modules/Settings/Vtiger/ConfigEditorDetail.tpl	required to ensure continuity of work after choosing this layout (post configuration edit save from previous layout).
layouts/i386/modules/Users/Login.tpl	default view presented when user has not logged in.

Step 4: Home View Templates

Post Login when a URL visit is made without specific module/view parameter, (module=Home, view=Dashboard) is defaulted. So it is essential to start implementation with linked templates.

File	Description
layouts/i386/modules/Vtiger/dashboards/DashBoardPreProcess.tpl	pre-process template.
layouts/i386/modules/Vtiger/dashboards/DashBoardContents.tpl	process template.

Step 5: List View Templates

Although, our is focus of this layout is around Leads ListView. We are implementing the template files as generic (fallback) in modules/Vtiger.

File	Description
layouts/i386/modules/Vtiger/ListViewPreProcess.tpl	pre-process template.
layouts/i386/modules/Vtiger/ListViewPostProcess.tpl	process template.
layouts/i386/modules/Vtiger/ListViewContents.tpl	post-process template.

Step 6: Package files

zip -r i386Layout-v1.zip *manifest.xml layouts/*

Refer to *Package Structure*

Step 7: Install in Vtiger

You can import through Module Manager UI.

The layout is now installed and ready, you can set it as default from Configuration Editor

Note

After switching to this layout, in case you would like to revert back - please edit config.inc.php and update $default_layout value to (vlayout)

Uninstall Module

During development you may want to uninstall module, follow the steps below:

Step 1: Delete Module

```
<?php

include_once 'vtlib/Vtiger/Module.php';

$Vtiger_Utils_Log = true;

$module = Vtiger_Module::getInstance('<ModuleName>');
if ($module) $module->delete();
```

Step 2: Delete Folders

Module folders and files needs to be removed manually. Following are the location where its references exists.

```
modules/ModuleName
languages/en_us/ModuleName.php
languages/.../ModuleName.php
layouts/vlayout/modules/ModuleName
cron/ModuleName
```

Step 3: Delete Data

Module stores data in the database-table, you may want to delete / truncate the same.

Previous topic

Layout i386

Internals

UI Control Flow

There are two essential flow that controls UI (view & action).

View - expects the outcome to be HTML / Redirection.

Action - expects the outcome to be JSON (specially useful for Ajax)

Example:

```
index.php?module=Leads&view=List

Leads/views/List.php

If this is not found - the fallback file would be:

Vtiger/views/List.php
```

Similarly:

```
index.php?module=MyModule&action=Save

Leads/actions/Save.php

If this is not found - the fallback file would be:

Vtiger/actions/Save.php
```

UI Request Processing

`main/Web_UI` invokes the appropriate view / action class method in the following order:

`checkPermission($request)`	Handle custom permission control before control proceeds with processing. Authentication & User access to module would have happened before to this call.
`preProcess($request)`	Invocation is suppressed if the request is made via AJAX. Generally this method emits the Header portion of the page.
`process($request)`	Invoked for both normal / ajax request.
`postProcess($request)`	Invocation is suppressed if the request is made via AJAX. Generally this method emits the Footer portion of the page.

Webservice Control Flow

Note

Yet to be documented - please be patient.

Eventing

Note

Yet to be documented - please be patient.

Resource File Paths

`getHeaderScripts` & `getHeaderCss` supports shorthand syntax for specifying the dependent resources to be loaded for the client-side.

Reference Type | **Format** | **Resolves To**

Reference Type	Format	Resolves To
Layout Relative	modules.Vtiger.resources.List	**layouts/<current_layout>**/modules/Vtiger/resources/List
Absolute	~/libraries/fullcalendar/fullcalendar.css	**<vtigercrm>**/libraries/fullcalendar/fullcalendar.css
Remote	http://remote.server.tld/js/function.js	as is.

FAQ’s

When should my module be Entity type?

Every module need not necessarily be an Entity type and the decision should be made wisely by the developer. If any of the following condition are met, then you should build an Entity module.

- If module represents a Business object (like, Account, Contact, Lead, Payment)
- If module requires access control (User/Group Assignment, Sharing Rules, Profile Access).
- If module record has field(s) that is directly / indirectly linked with other Entity module.
- If Workflows, Reports are required for the module.

Previous topic

Internals

Next topic

Porting Extensions

Porting Extensions From 5.x.x to 6.0.0

Entity Module

Vtiger 5.x based entity module that followed Module Manager compatibility (vtlib API) would continue to work on Vtiger 6 without changes.

You will need to review the custom code and refactor them to get it work on the newer platform.

Find more details at - *Package Structure* and *An Entity Module Example*

Extension Module

You need to re-develop the client and server code based on new *Package Structure*.

To get you started on it we have an *An Extension Module Example*...

Control Flow Changes

Refer (UI Control Flow) *Internals*

Code Structure Changes

Refer *Package Structure*

Previous topic

Porting Extensions

Next topic

Extensions Store

Installation

Vtiger CRM is web-application built using PHP.

Pre-requisites

- Apache 2.1+
- MySQL 5.1+
  - storage_engine = InnoDB
  - local_file = On
  - sql_mode = empty (or NO_ENGINE_SUBSTITUTION) for MySQL 5.6+
- PHP 5.2+, 7.0
  - php-imap
  - php-curl
  - php-xml
  - max_memory (min. 256MB)
  - max_execution_time (min. 60 seconds)
  - error_reporting (E_ERROR & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED)
  - display_errors = OFF
  - short_open_tags = OFF
- Hardware: 4 GB RAM, 250 GB Disk (for file attachments)

Procedure

- Get your LAMP, WAMP, MAMP configured that meets the above pre-requisites.
- Unzip Vtiger CRM into web-folder (or document root)
- Make sure to provide writeable access to (Apache process owner) on web-folder.
- Visit http://yourserver.tld/index.php
- The wizard will help you get through next steps of installation.

Previous topic

Administrator Guide

Next topic

Migration