Modules

This document describes how to implement Modules. A Module allows packages and services to be created. They define the fields requested when adding or editing packages and services, as well as handle all communication with remote servers to provision said packages and services.

> **Use the Extension Generator**
>
> As of Blesta 4.12 we've included a useful tool to help developers get started and save time. Blesta's Extension Generator can be used to generate many the files necessary for a module and will create basic code with an option to include comments to help you understand each part of the code.

Getting Started with Modules

For the purpose of this manual, the module name we'll refer to is **my_module**. Your module name will, of course, differ.

> **Module names must be unique**
>
> A user will not be able to install two module with the same name.

File Structure

Modules are fully contained within their named module directory and placed in the */installpath/components/modules/* directory. Each module is only required to contain a single class, representing the module, but the following is the recommended structure:

- */modules/*
  - *my_module/*
    - *views/*
    - *language/*
    - **my_module.php**

Install Logic

If your module requires any code to execute when installed, place that logic in your module's **install()** method.

/modules/my_module/my_module.php

```
<?php
class MyModule extends Module {

    ...

    public function install() {
        #
        # TODO: Place installation logic here
        #
    }
}
?>
```

Uninstall Logic

If your module required code to install, it likely requires code to uninstall. Place that logic in your module's **uninstall()** method.

There are two parameters passed to the uninstall() method. The first (**$module_id**) is the ID of the module being uninstalled. Because a module can be installed independently under different companies you may need to perform uninstall logic for a single module instance. The second parameter (**$last_instance**) identifies whether or not this is the last instance of this module in the system. If **true**, be sure to remove any remaining remnants of the module.

/modules/my_module/my_module.php

```
<?php
class MyModule extends Module {

    ...

    public function uninstall($module_id, $last_instance) {
        #
        # TODO: Place uninstallation logic here
        #
    }
}
?>
```

Upgrade Logic

When the version of your code differs from that recorded within Blesta a user may initiate an upgrade, which will invoke your module's **upgrade()** method.

The **$current_version** is the version currently installed. That is, the version that will be upgraded from.

/modules/my_module/my_module.php

```
<?php
class MyModule extends Module {

    ...

    public function upgrade($current_version) {
        #
        # TODO: Place upgrade logic here
        #
    }

}
?>
```

Error Passing

Blesta facilitates error message passing through the use of the **Input** component. Simply load the Input component into your module and set any errors you encounter using **Input::setErrors()**. The setErrors() method accepts an array of errors. This can be a multi-dimensional array (in the case of multiple errors triggered for the same input or criteria) or a single dimensional array. The first dimension should be the name of the input that triggered the error. The second dimension, if necessary, is used to identify the type of error encountered.

/module/my_module/my_module.php

```
<?php
class MyModule extends Module {

    public function __construct() {
        Loader::loadComponents($this, array("Input"));
    }

    ...

    public function upgrade($current_version) {
        // Ensure new version is greater than installed version
        if (version_compare($this->version, $current_version) < 0) {
            $this->Input->setErrors(array(
                'version' => array(
                    'invalid' => "Downgrades are not allowed."
                )
            ));
            return;
        }
    }
}
?>
```

Module Methods

Now that we've looked at some of the basic of creating a module, let's take a look at all of the required methods each module must implement: Module Methods.

Registrar Modules

As of Blesta 5.1 we introduced a new type of module, the registrar module. Registrar modules are utilized by the Domain Manager plugin, which include specific functions for domain management. Registrar modules implement all the functions as standard modules in addition to the `Blesta\Core\Util\Modules\Registrar` interface.

/module/my_module/my_module.php

```
<?php
use Blesta\Core\Util\Modules\Registrar;

class Namesilo extends Module implements Registrar
{
        ...

    public function getDomainContacts($domain, $module_row_id = null)
    {
                #
                # TODO: Get the contacts of the domain
                #
    }

    public function setDomainContacts($domain, array $vars = [], $module_row_id = null)
    {
                #
                # TODO: Set the contacts of the domain
                #
    }

    public function checkTransferAvailability($domain, $module_row_id = null)
    {
                // Check if the domain can be transferred
        return !$this->checkAvailability($domain, $module_row_id);
    }

        public function checkAvailability($domain, $module_row_id = null)
```

```
{
            #
            # TODO: Check if the domain is available
            #
    }

    public function getExpirationDate($domain, $format = 'Y-m-d H:i:s', $module_row_id = null)
{
            #
            # TODO: Get the expiration date of the domain
            #
    }

public function getDomainInfo($domain, $module_row_id = null)
{
            #
            # TODO: Get the domain information
            #
}

public function getDomainIsLocked($domain, $module_row_id = null)
{
            #
            # TODO: Check if the domain is locked
            #
}

public function getDomainNameservers($domain, $module_row_id = null)
{
            #
            # TODO: Get the domain name server
            #
}

public function setDomainNameservers($domain, $module_row_id = null, array $vars = [])
{
            #
            # TODO: Set the domain name server
            #
}

public function setNameserverIps(array $vars = [], $module_row_id = null)
{
            #
            # TODO: Set the domain name server IP addresses
            #
}

public function lockDomain($domain, $module_row_id = null)
{
            #
            # TODO: Lock the domain from being transferred
            #
}

public function unlockDomain($domain, $module_row_id = null)
{
            #
            # TODO: Unlock the domain to allow transfers
            #
}

public function registerDomain($domain, $module_row_id = null, array $vars = [])
{
            #
            # TODO: Register a new domain with the registrar
            #
}

public function transferDomain($domain, $module_row_id = null, array $vars = [])
{
```

```
                #
                # TODO: Transfer a domain to another registrar
                #
    }

    public function resendTransferEmail($domain, $module_row_id = null)
    {
                #
                # TODO: Re-send the transfer confirmation email
                #
    }

    public function sendEppEmail($domain, $module_row_id = null)
    {
                #
                # TODO: Send an email with the EPP code
                #
    }

    public function updateEppCode($domain, $epp_code, $module_row_id = null, array $vars = [])
    {
                #
                # TODO: Update the EPP code
                #
    }

    public function renewDomain($domain, $module_row_id = null, array $vars = [])
    {
                #
                # TODO: Renew the domain for the given amount of years
                #
    }

    public function restoreDomain($domain, $module_row_id = null, array $vars = [])
    {
                #
                # TODO: Restore a suspended domain
                #
    }

    public function getTldPricing($module_row_id = null)
    {
                // Returns an array containing the pricing for each tld
        return [
            '.com' => [
                'USD' => [
                    1 => ['register' => 10, 'transfer' => 10, 'renew' => 10],
                    2 => ['register' => 20, 'transfer' => 20, 'renew' => 20]
                ]
            ]
        ];
    }

        public function getTlds($module_row_id = null)
    {
                // Returns a list of the TLDs supported by the module
                return [
                        '.com',
                        '.net',
                        '.org'
                ];
        }
        ...
}
```