

Общая информация

Данное руководство описывает процесс создания модулей для RegSubmitter, начиная с версии 1.1.0. В ветке 1.0.x модульная система не поддерживала разделение модулей на пакеты, и поэтому не все нижеизложенные инструкции будут применимы к старым версиям. В особенности это касается структуры файлов.

Структура файлов

Основные файлы RegSubmitter располагаются в 6 папках:

- `config` — содержит файлы настроек RegSubmitter.
- `handlers` — скрипты-обработчики различных режимов работы.
- `includes` — базовые библиотеки, необходимые для работы программы.
- `modules` — все файлы, относящиеся к модулям RegSubmitter.
- `template` — html шаблоны страниц.
- `tmp` — временные файлы, т. ч. cookies и картинки captcha.

В дальнейшем нас будут интересовать файлы в каталоге `modules`. Каждый пакет модулей находится в отдельной подпапке `modules`, при чем имя этой папки используется в качестве системного идентификатора пакета и должно состоять только из латинских букв и цифр. В каждой папке пакета должен находиться файл с расширением `.php`, имя которого должно совпадать с именем папки модулей, со следующим содержимым:

Листинг 1:

```
<?php
$cat_name='Русские';
?>
```

Содержимое `$cat_name` будет выведено на главной странице как название пакета сервисов.



В папке пакета располагаются папки модулей. Так же, как и с пакетами, имя папки модуля должно совпадать с именем `.php`-файла, содержащего модуль. Например, если папка модуля называется `Memori`, то файл, должен называться `Memori.php`, при чем регистр символов обязательно должен совпадать. Так же в папке модуля должна находиться favicon сервиса, с которым работает модуль.

Листинг 2 (пример итоговой файловой структуры):

```
/modules/
/modules/Rus/
/modules/Rus/Rus.php
/modules/Rus/Memori/
/modules/Rus/Memori/Memori.php
/modules/Rus/Memori/memori.png
```

Код модуля.

RegSubmitter работает в кодировке UTF-8 и это нужно учитывать при написании модулей. Сами скрипты так же должны быть написаны в кодировке UTF-8 (большинство современных РНР редакторов поддерживают ее). Ниже мы рассмотрим процесс создания модуля, работающего с сервисом Bookmarks.ru (название взято из головы и отношения к реальному сайту bookmarks.ru не имеет ☺), который работает в кодировке cp1251.

Каждый модуль является классом-наследником системного класса Module и его имя должно совпадать с именем папки модуля. Важно: чтобы ваш модуль мог работать вместе с другими модулями, необходимо, чтобы он имел уникальное имя класса и \$id (о нем будет сказано чуть позже).

Листинг 3 (Объявление класса):

```
<?php
class Bookmarks extends Module
{
```

Далее нам необходимо задать несколько параметров модуля:

- \$name — название сервиса, будет отображаться пользователю.
- \$id — системный идентификатор модуля, должен быть уникальным, иначе модуль будет конфликтовать с другими. Рекомендуется делать его таким же, как и имя класса, но это не обязательно.
- \$url — адрес сервиса. Будет использоваться всюду, где заголовок сервиса является ссылкой.
- \$icon — имя файла favicon сервиса. Этот файл должен располагаться в той же папке, что и модуль. Обратите внимание, что не нужно указывать полный путь к картинке, а только ее имя.
- \$notice — примечание к сервису, которое будет отображено на финальном шаге регистрации. В частности, в нем можно указать, требует ли сервис активации по email.

Листинг 4 (Параметры модуля):

```
var $name = 'Bookmarks.ru';
var $id = 'Bookmarks';
var $url = 'http://bookmarks.ru';
var $icon = 'favicon.ico';
var $notice = 'Сервис требует активации аккаунта по e-mail.
Проверьте ваш почтовый ящик на предмет писем со ссылками для
активации.';
```

Для работы модуля необходимо определить два метода класса: GetForm() и SubmitReg(). Первый отвечает за загрузку captcha, а второй — за собственно регистрацию.

Рассмотрим метод GetForm(). Параметры ему никакие не передаются, а возвращать он должен либо null, либо массив с информацией о сессии и капче. Null нужно возвращать в случае, если сервис не требует ввода капчи. Разберемся с ситуацией, когда она нужна.

В возвращаемом массиве должен присутствовать как минимум один элемент — 'image_file'. Это путь к картинке капчи. Поскольку картинки должны быть сохранены в папке tmp, то путь должен иметь вид tmp/image_93ea908a80.gif, где «93ea908a80» - уникальное случайное число. Второй стандартный, хотя и не обязательный, элемент —

'cookie_file'. Он содержит путь к файлу cookies (который тоже для каждой итерации должен быть уникален), чтобы при осуществлении регистрации мы использовали те же данные сессии. Так же там могут присутствовать и любые другие элементы, которые потом будут переданы функции SubmitReg().

Для осуществления операций с сетью используется класс Curl_HTTP_Client, который детально будет описан ниже.

Листинг 5 (Пример реализации функции GetForm()):

```
function GetForm()
{
    // Создаем HTTP клиент
    $curl = new Curl_HTTP_Client();

    // Устанавливаем User-Agent браузера
    $curl->set_user_agent("Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1)");
    // Генерируем временный файл для хранения cookies
    $cookies_file = tempnam('./tmp/', 'cookie_');
    $curl->store_cookies($cookies_file);

    // Скачиваем картинку капчи
    $image = $curl->fetch_url('http://bookmarks.ru/captcha.php');
    $curl->close();

    // Генерируем имя файла картинки и сохраняем.
    $image_file = './tmp/image_'.md5($cookies_file).'.gif';
    file_put_contents($image_file, $image);

    // Возвращаем параметры.
    return array(
        'image_file' => $image_file,
        'cookies_file' => $cookies_file,
    );
}
```

Теперь обратимся к методу SubmitReg(). Ему передаются три параметра:

- \$account — строка с информацией о том, какой аккаунт регистрировать, в формате логин:пароль:email.
- \$captcha_data — массив, который вернула функция GetForm()
- \$captcha — текст капчи, введенный пользователем.

В качестве возвращаемого значения метода должно быть null, если регистрация прошла успешно, или массив сообщений об ошибках в противном случае.

Листинг 6 (Пример реализации функции SubmitReg()):

```
function SubmitReg($account, $captcha_data, $captcha)
{
    // Формируем массив с данными аккаунта
    $account = explode(':', convert_encoding($account, 'utf-8',
'cp1251'));

    // Создаем HTTP клиент
    $curl = new Curl_HTTP_Client();
    $curl->set_user_agent("Mozilla/4.0 (compatible; MSIE 6.0;
```

```
Windows NT 5.1)");
    $curl->store_cookies($captcha_data['cookies_file']);

    // Подготавливаем данные формы
    $post_data = array(
        'login'           => $account[0],
        'email'           => $account[2],
        'password'        => $account[1],
        'password_confirm' => $account[1],
        'captcha'         => $captcha,
    );

    //Отправляем запрос и получаем ответ.
    $html_data = $curl-
>send_post_data("http://bookmarks.ru/register.php", $post_data);

    // Закрываем сессию curl
    $curl->close();

    // Удаляем временные файлы.
    @unlink($captcha_data['image_file']);
    @unlink($captcha_data['cookies_file']);

    // Перекодируем ответ в UTF-8
    $html_data = convert_encoding($html_data, 'cp1251', 'utf-8');

    //Проверяем, не произошла ли ошибка при регистрации
    if(strpos($html_data, 'На указанный Вами электронный адрес'))
    {
        // Ошибка не произошла
        return null;
    }
    else
    {
        // Произошла ошибка, вылавливаем из полученной страницы
сообщения об ошибках и возвращаем их как массив.
        preg_match_all('#<p class="error">([^\<]+)</p>#mu',
$html_data, $matches);
        return $matches[1];
    }
}
```

На этом создание модуля можно считать законченным ☺. Полный код модуля находится в конце руководства, в приложении №3.

Приложение 1. Описание класса Curl_HTTP_Client.

Класс Curl_HTTP_Client предоставляет широкие возможности для работы с протоколом HTTP, описывать которые можно очень долго, поэтому я затрону лишь наиболее полезные методы.

- set_user_agent(string \$agent) — устанавливает заголовок User-Agent равным \$agent.
- store_cookies(string \$file) — указывает, что все cookies должны браться из файла \$file и записываться в него же.
- string fetch_url(string \$url) — запросить адрес \$url методом GET.

Возвращает заголовки и тело ответа HTTP сервера.

- `string send_post_data(string $url, mixed $args)` — запросить адрес `$url` методом POST с параметрами запроса `$args`. `$args` может быть ассоциативным массивом вида `array('param_name' => 'param_value')` или строкой вида `param1_name=param1_value¶m2_name=param2_value`. Возвращает заголовки и тело ответа HTTP сервера.
- `int get_http_response_code()` - возвращает код ответа http сервера (200, если запрос успешен).

Приложение 2. Дополнительные функции.

`string convert_encoding($string, $from, $to)` — преобразует строку `$string` из кодировки `$from` в `$to`. Для преобразования кодировок рекомендуется использовать именно ее, поскольку она работает как с модулем `mbstring`, так и с `iconv`, что обеспечивает большую переносимость кода.

`dbg(mixed $var)` — полезно использовать эту функцию в процессе отладки для просмотра содержимого переменной `$var`. Она распечатывает его при помощи встроенной функции `print_r()` и выводит результат внизу страницы. Эта функция работает корректно, даже если модуль вызывается через AJAX (как на последней стадии процесса регистрации), поэтому удобнее использовать ее, а не встроенные. Обратите внимание, что если не установлена константа `DEBUG`, то функция ничего не выведет!

Приложение 3. Полный код модуля.

Листинг 7 (Полный пример модуля):

```
<?php
class Bookmarks extends Module
{
    var $name = 'Bookmarks.ru';
    var $id = 'Bookmarks';
    var $url = 'http://bookmarks.ru';
    var $icon = 'favicon.ico';
    var $notice = 'Сервис требует активации аккаунта по e-mail.
    Проверьте ваш почтовый ящик на предмет писем со ссылками для
    активации.';

    function GetForm()
    {
        // Создаем HTTP клиент
        $curl = new Curl_HTTP_Client();

        // Устанавливаем User-Agent браузера
        $curl->set_user_agent("Mozilla/4.0 (compatible; MSIE 6.0;
    Windows NT 5.1)");
        // Генерируем временный файл для хранения cookies
        $cookies_file = tempnam('./tmp/', 'cookie_');
        $curl->store_cookies($cookies_file);

        // Скачиваем картинку капчи
```

```
        $image = $curl-
>fetch_url('http://bookmarks.ru/captcha.php');
        $curl->close();

        // Генерируем имя файла картинки и сохраняем.
        $image_file = './tmp/image_'.md5($cookies_file).'.gif';
        file_put_contents($image_file, $image);

        //Возвращаем параметры.
        return array(
            'image_file'          => $image_file,
            'cookies_file'       => $cookies_file,
        );
    }

    function SubmitReg($account, $captcha_data, $captcha)
    {
        // Формируем массив с данными аккаунта
        $account = explode(':', convert_encoding($account, 'utf-8',
'cp1251'));

        // Создаем HTTP клиент
        $curl = new Curl_HTTP_Client();
        $curl->set_user_agent("Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1)");
        $curl->store_cookies($captcha_data['cookies_file']);

        // Подготавливаем данные формы
        $post_data = array(
            'login'                => $account[0],
            'email'                => $account[2],
            'password'             => $account[1],
            'password_confirm'     => $account[1],
            'captcha'              => $captcha,
        );

        //Отправляем запрос и получаем ответ.
        $html_data = $curl-
>send_post_data("http://bookmarks.ru/register.php", $post_data);

        // Закрываем сессию curl
        $curl->close();

        // Удаляем временные файлы.
        @unlink($captcha_data['image_file']);
        @unlink($captcha_data['cookies_file']);

        // Перекодируем ответ в UTF-8
        $html_data = convert_encoding($html_data, 'cp1251',
'utf-8');

        //Проверяем, не произошла ли ошибка при регистрации
        if(strpos($html_data, 'На указанный Вами электронный
адрес'))
        {
            // Ошибка не произошла
            return null;
        }
    }
}
```

```
        else
        {
            // Произошла ошибка, вылавливаем из полученной
            // страницы сообщения об ошибках и возвращаем их как массив.
            preg_match_all('#<p class="error">([^\<]+)</p>#mu',
            $html_data, $matches);
            return $matches[1];
        }
    }
}
?>
```

Changelog.

- 1.0.0 — Первая версия руководства, актуальна на момент написания версия — 1.1.0.
22.01.2009

Лицензирование.

Запрещается распространение самостоятельно созданных модулей для RegSubmitter на коммерческой или некоммерческой основе без договоренности с разработчиком RegSubmitter. Нарушение может караться вплоть до лишения автора данных модулей лицензии на RegSubmitter.