**Создание модуля контроля в Magento.**

В этой статье мы постараемся объяснить, как можно создать модуль Magento, который будет фиксировать информацию от сторонних источников (например, внешнего веб-сайта или новостной рассылки по электронной почте) и включать HTML скрипты для проверки успешности преобразований на странице.

Мы будем рассматривать следующие темы:

* Сбор данных об идентификаторах ресурса (URI);
* Сохранение и извлечение значений из куки-файлов;
* Проверка успешности добавления пользовательского контента на страницу;
* Добавление модуля конфигурации в панель администрирования Magento;
* Создание общедоступного модуля без закодированных установок.

Как всегда, этот модуль будет создан таким образом, что ни один из основных файлов не будет изменен. Это позволит сохранить его портативность и основные преимущества характеристик Magento.

### ПЕРЕД ТЕМ КАК МЫ НАЧНЕМ.

Это руководство предполагает, что вы уже знакомы с [основами создания модуля Magento](http://coding.smashingmagazine.com/2012/03/01/basics-creating-magento-module/), и ознакомлены с вопросами, затронутыми в [нашей предыдущей статье о Компоновке Magento](http://coding.smashingmagazine.com/2012/11/30/introducing-magento-layout/). Так что может быть стоит немного остановиться на данной информации, для тех читателей, которые еще не проработали весь материал.

Мы будем добавлять любые специфические параметры веб-сайта, как настраиваемые установки в панель администрирования Magento. Как следствие, наш модуль будет представлять собой «объединение модулей» код которого будет принадлежать файлу app/code/community. Идея заключается в том, что мы можем поместить это объединение модулей в любой объект Magento, и никакие модификации кода не будут нуждаться в полной работоспособности.



Для начала, создайте следующее структурное дерево с пустыми файлами, которые можно будет заполнить позднее.

app (приложение)

 - code (код)

 - community (объединение)

 - SmashingMagazine

 - Affiliate (структурная единица)

 - Block (блок)

 - Conversion.php

 - Model (модель)

 - Observer.php

 - etc (и так далее)

 - config.xml

 - design (проект)

 - frontend (внешний интерфейс)

 - base (основная часть)

 - default (установки по умолчанию)

 - layout (компоновка)

 - smashingmagazine\_affiliate.xml

 - template (шаблон)

 - smashingmagazine\_affiliate

 - conversion.phtml

 - etc (и так далее)

 - modules (модули)

 - SmashingMagazine\_Affiliate.xml

Мы уже можем организовать заполнение информацией файлаSmashingMagazine\_Affiliate.xml:

<?xml version="1.0" encoding="UTF-8"?>

<config>

 <modules>

 <SmashingMagazine\_Affiliate>

 <active>true</active>

 <codePool>community</codePool>

 </SmashingMagazine\_Affiliate>

 </modules>

</config>

### СБОР ДАННЫХ О СТРУКТУРНЫХ НАПРАВЛЕНИЯХ.

Первой задачей отслеживания структурных процессов, является сбор данных о направлениях, которые обычно предоставлены в виде $\_GET параметров URI из сторонних веб-сайтов или новостных рассылок. Напримерsmashingmagazine.com/?utm\_source=some\_affiliate\_id где some\_affiliateявляется идентификатором дочернего подразделения, которое мы будем награждать, если клиент совершит приобретение.

#### Определение событий блока наблюдений.

Мы хотим, чтобы наши дочерние подразделения имели возможность ссылаться на любую страницу нашего сайта, для того, чтобы клиенты могли перейти непосредственно к интересующему их продукту или услуге. Поэтому необходимо организовать метод проверки для  $\_GET параметров на каждой странице. Как уже было принято, мы не будем изменять код в основных файлах Magento, так как хотим сохранить модуль портативным, с максимально возможным уровнем базовых функций. Мы будем использовать команду event observer.

Существует одно событие, которое обрабатывается на каждой страницеcontroller\_front\_init\_before, поэтому давайте создадим собственный файлconfig.xml с функцией observer для этого события.

<?xml version="1.0" encoding="UTF-8"?>

<config>

 <modules>

 <SmashingMagazine\_Affiliate>

 <version>0.0.1</version>

 </SmashingMagazine\_Affiliate>

 </modules>

 <global>

 <models>

 <smashingmagazine\_affiliate>

 <class>SmashingMagazine\_Affiliate\_Model</class>

 </smashingmagazine\_affiliate>

 </models>

 <events>

 <controller\_front\_init\_before>

 <observers>

 <smashingmagazine\_affiliate>

 <class>smashingmagazine\_affiliate/observer</class>

 <method>captureReferral</method>

 <type>singleton</type>

 </smashingmagazine\_affiliate >

 </observers>

 </controller\_front\_init\_before>

 </events>

 </global>

</config>

Теперь при каждой загрузке страницы, будет вызываться наша процедураSmashingMagazine\_Affiliate\_Model\_Observer::captureReferral(), давайте добавим в файл Observer.php следующее содержимое:

<?php

class SmashingMagazine\_Affiliate\_Model\_Observer

{

public function captureReferral(Varien\_Event\_Observer $observer)

 {

// here we add the logic to capture the referring affiliate ID

 }

}

#### Сбор дочерних идентификаторов.

Событие controller\_front\_init\_before осуществляется из основной процедуры Magento Mage\_Core\_Controller\_Varien\_Front::init(), и включает в себя ссылку на тот же класс, что и в следующем фрагменте кода из файлаapp/code/core/Mage/Core/Controller/Varien/Front.php:

Mage::dispatchEvent(

 'controller\_front\_init\_before',

 array('front' => $this)

);

Можно получить доступ к экземпляру Mage\_Core\_Controller\_Varien\_Front, включив следующий код в событие event observer:

<?php

class SmashingMagazine\_Affiliate\_Model\_Observer

{

public function captureReferral(Varien\_Event\_Observer $observer)

 {

 $frontController = $observer->getEvent()->getFront();

 }

}

Команда controller отвечает за синтаксический анализ URI. Теперь у нас есть доступ к соответствующему объекту, и мы должны определить существует ли наш $\_GET параметр:

<?php

class SmashingMagazine\_Affiliate\_Model\_Observer

{

public function captureReferral(Varien\_Event\_Observer $observer)

 {

 $frontController = $observer->getEvent()->getFront();

 $utmSource = $frontController->getRequest()

 ->getParam('utm\_source', false);

 if ($utmSource) {

 // here we will save the referrer affiliate ID

 }

 }

}

В приведенном выше коде, мы вызываем getRequest(), чтобы получитьMage\_Core\_Controller\_Request\_Http например, от controller, который будет содержать всю требуемую информацию о текущем запросе URI, включая$\_POST или $\_GET параметры. Нас интересует utm\_source в параметре $\_GET, и мы можем получить его значение из процедуры getParam().

Переменная $utmSource теперь будет содержать ссылку на дочерний идентификатор.

#### Сохранение дочернего идентификатора.

Теперь нам нужно сохранить значение переменной $utmSource, для того, чтобы мы могли вознаградить свой филиал за совершение клиентом покупки. Есть несколько способов сделать это, но в этом уроке мы будем использовать простую переменную $\_COOKIE. Magento имеет встроенную функциональность для работы с cookie , так что все будет очень просто:

<?php

class SmashingMagazine\_Affiliate\_Model\_Observer

{

const COOKIE\_KEY\_SOURCE = 'smashingmagazine\_affiliate\_source';

public function captureReferral(Varien\_Event\_Observer $observer)

 {

 $frontController = $observer->getEvent()->getFront();

 $utmSource = $frontController->getRequest()

 ->getParam('utm\_source', false);

 if ($utmSource) {

 Mage::getModel('core/cookie')->set(

 self::COOKIE\_KEY\_SOURCE,

 $utmSource,

 $this->\_getCookieLifetime()

 );

 }

 }

 protected function \_getCookieLifetime()

 {

 $days = 30;

 // convert to seconds

 return (int)86400 \* $days;

 }

}

Мы инициализировали новую постоянную COOKIE\_KEY\_SOURCE, потому что мы будет ссылаться на это же значение из другого класса. При необходимости вызова значения из разных мест, появляется удобная возможность использовать постоянную. Таким образом, мы сводим работу над улучшением внутренней структуры программного кода (рефакторинг) к минимуму, ведь уменьшается число значений, которые когда-нибудь понадобится изменить. Мы также инициализировали новый защищенный метод, \_getCookieLifetime(), для установления величины времени жизниcookie строго на 30 дней.

### УВЕДОМЛЕНИЕ ОБ ИЗМЕНЕНИЯХ.

На данный момент, вся информация от партнерского идентификатора хранится в файле cookie. Давайте предположим, что клиент добавил некоторые продукты в корзину, завершил процесс проверки и сейчас находится на странице оплаты. Это случай, когда большинство партнерских схем требуют определенного уведомления, что справочные данные из каталога преобразованы в заказ. Как правило, это достигается путем активации фрагментов JavaScript кода или крошечного изображения в нижней части страницы.

Есть, конечно, и другие способы обновления партнерских модулей, например, через серверные вызовы API, которые могут быть сделаны с помощью команды event observers. Но в этом уроке мы будем использовать макет Magento для того, чтобы отобразить пользовательский выбор в нижней части страницы.

Мы должны обновить файл config.xml чтобы обслуживать blocks и обновить макет:

<?xml version="1.0" encoding="UTF-8"?>

<config>

 <modules>

 <SmashingMagazine\_Affiliate>

 <version>0.0.1</version>

 </SmashingMagazine\_Affiliate>

 </modules>

 <global>

 <blocks>

 <smashingmagazine\_affiliate>

 <class>SmashingMagazine\_Affiliate\_Block</class>

 </smashingmagazine\_affiliate>

 </blocks>

 <models>

 <smashingmagazine\_affiliate>

 <class>SmashingMagazine\_Affiliate\_Model</class>

 </smashingmagazine\_affiliate>

 </models>

 <events>

 <controller\_front\_init\_before>

 <observers>

 <smashingmagazine\_affiliate>

 <class>smashingmagazine\_affiliate/observer</class>

 <method>captureReferral</method>

 <type>singleton</type>

 </smashingmagazine\_affiliate >

 </observers>

 </controller\_front\_init\_before>

 </events>

 </global>

 <frontend>

 <layout>

 <updates>

 <smashingmagazine\_affiliate

 module="SmashingMagazine\_Affiliate">

 <file>smashingmagazine\_affiliate.xml</file>

 </smashingmagazine\_affiliate>

 </updates>

 </layout>

 </frontend>

</config>

#### Добавление компоновки нового файла.

Мы заинтересованы в успешном выполнении командыonepage\_checkout\_success, так как именно эта команда отображает успешность процедуры оплаты. Давайте создадим новый макет файла:

<?xml version="1.0" encoding="UTF-8"?>

<layout>

 <checkout\_onepage\_success>

 <reference name="before\_body\_end">

 <block type="smashingmagazine\_affiliate/conversion"

 name="smashingmagazine\_affiliate\_conversion"

 template="smashingmagazine\_affiliate/conversion.phtml" />

 </reference>

 </checkout\_onepage\_success>

</layout>

#### Добавление шаблона нового файла.

Содержимое файла conversion.phtml будет зависеть от требований конкретного партнерского модуля. В данном примере мы будем использовать общую форму, которая включает в себя одно пиксельное изображение с динамическим отображением информации о торговце и партнерском идентификаторе:

<img

src="http://media.smashingmagazine.com/themes/smashingv4/images/logo.png

 ?merchant\_id=<?php echo $this->getMerchantId() ?>

 &affiliate\_id=<?php echo $this->getAffiliateId() ?>"

 width="1"

 height="1"

/>

#### Создание пользовательского блока.

Теперь нам нужно создать новый файл block для заполнения динамических данных в нашем шаблоне:

<?php

class SmashingMagazine\_Affiliate\_Block\_Conversion

 extends Mage\_Core\_Block\_Template

{

 public function getMerchantId()

 {

 return '12345';

 }

 public function getAffiliateId()

 {

 return Mage::getModel('core/cookie')->get(

 SmashingMagazine\_Affiliate\_Model\_Observer::COOKIE\_KEY\_SOURCE

 );

 }

}

### ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ В ПАНЕЛЬ АДМИНИСТРИРО … [ЕСЛИ ВЫ ХОТИТЕ ПРОЧИТАТЬ ПОЛНОСТЬЮ СТАТЬЮ, ПОСЕТИТЕ САЙТ НАШИХ СПОНСОРОВ](http://bot.kz/sponsor.php?p=22428&ret=http%3A%2F%2Fbot.kz%2F2013%2F02%2F22428)