Ubercart

上周5的时候,决定全面的转型,思量了许久,决定专注于Ubercart的相关研究,包括2次开发,汉化,以及相应的中文特色的模块定制.已经有1周的时间了,以后的全部精力都会投入到与这个模块相关的工作中去.Drupal太大了,就如同Java一样,自己很难把它研究透彻.而且,只有我一个人,很难在Drupal的各个方面,开发出来有用的东西.所以只能舍弃许多与此无关的.

从今天起,开始介绍Ubercart.其实以前我是写过一些相关的文章的.还有就是提供了一些模块的汉化包.从今天以后,本站的许多文章,都会与这个模块相关.

现介绍一下这个模块吧, Ubercart是用来建立电子商务网站的,它是Drupal的一个模块,一个后起之秀吧,现在用它的人不多. Ubercart和Oscommerce有一点点渊源,或者说是里面的许多东西,都是从那里直接继承过来的. Oscommerce像Drupal一样,是自己领域的开源软件的领头羊,我不知道Ubercart的作者是不是Oscommerce的核心开发人员,但是绝对是一个读过Oscommerce源代码,建过n多个Oscommerce相关的网站,扩展过Oscommerce的各种功能的高级程序员了.

有一天,他发现Oscommerce存在这样或者那样的弊端,他同时发现的Drupal的优点,非常认同Drupal的这种内在的机制,而Drupal现有的EC模块(也是很有名的)却不能满足他的实际需要时,他决定在Drupal的基础之上,开发一个购物车模块,来满足他的项目需要,当然这个项目是比较大的.这个购物车模块,就逐步的演化成了Ubercart,供大家使用.开发Ubercart的作者是个德国的程序员,现在有4个核心开发人员,负责Ubercart的开发,升级.

所以说Ubercart是个后起之秀,EC当时已经很流行了. Ubercart现在正在成为Drupal电子商务网站的必选模块,有这种趋势,但是现在还是使用EC的比较多.作为后起之秀,尽管用的人少,也就是市场小,但是却有着后发的优势,那就是它吸取了Oscommerce, Drupal,EC的优点,尽量避免Oscommerce,EC中存在的缺陷.也正是它的这种理念的先进性,使得越来越多的人使用Ubercart进行建站.当然,Ubercart是建立在Drupal基础之上的,也就是只有装了Drupal以后,它才能跑起来.

Oscommerce很流行,但是在国内用的人就比较少.Ubercart不知道能不能在中国也悄然的流行开来.

好了,不多说了,在这个目录下面,我将放置大量的相关文章,如果你也有Ubercart方面的需求,大可以找我.可以一起交流,这方面的使用经验.一起去完善它.另外,本站所有与Ubercart相关的文章,模块,代码,都将以GPL的方式,对于文章可能是署名-非商业性使用-相同方式共享 3.0.授权方式,也就是与Drupal的Ubercart的保持一致.大家可以放心的使用,包括商用.

相关链接: http://zhupou.cn

UTF-8和GBK之间的转换

支付宝给出来的示例代码,里面采用的是GBK编码,使用Zend Studio打开以后中文全是乱码.我先是用写字板打开,然后拷贝到记事本中,然后再拷贝到Zend Studio中,这样就实现了将原有代码拷贝过来.

我是直接使用的原有代码,包括各种函数,都是示例代码中所给的,所以遇到的第一个问题就是乱码问题.我仔细的都了源码,按照计算机执行的顺序,一步步往下读,发现了一个问题:

```
    function charset_encode($input,$_output_charset ,$_input_charset ="GBK" ) {
                    $output = "";
                    if(!isset($_output_charset) )$_output_charset  =
$this->parameter['_input_charset '];
                    if($_input_charset == $_output_charset || $input ==null) {
                            $output = $input;
                    } elseif (function_exists("mb_convert_encoding")){
                            $output =
mb_convert_encoding($input,$_output_charset,$_input_charset);
                    } elseif(function_exists("iconv")) {
                            $output = iconv($_input_charset,$_output_charset,$input);
                    } else die("sorry, you have no libs support for charset change.");
                    return $output;
            }
```

就是这个函数,找到它以后,在最后一句设置了一个断点,使用Zend Studio进行debug，在这里我发现了，我从drupal中读取出来的字符串，经过这个函数处理以后，就变成了乱码了。

这个发现虽然很小，但是找到了问题的所在。我把整个模块文件改造成了GBK编码，传递到支付宝的字符串恢复了正常，但是模块在后台中的配置出现了乱码。但是我发现，在模块中，使用t函数，里面使用英文，然后再翻译成中文的方式可以解决这个后台问题。而在这个时候，我对从drupal中传到这个模块的字符串作了处理，charset_encode，就是使用的这个函数。那么现在整个问题就解决了，我只需要使用GBK编码的形式来编写这个模块就可以了。而对从drupal中传递到这个模块中的数据作一次转换。

但是觉得这种方式，不够地道，因为drupal中，所有的模块都是采用UTF-8的格式，如果我的模块采用GBK的话，有点别扭。

我很快想到了，如果模块本身采用UTF-8格式，我只需要反转一下函数charset_encode就可以了，两者之间是对称的。所以我决定，模块本身仍然采用UTF-8的形式。

尽管没有采用GBK的形式，但是中间的替换过程没有白费，因为它至少证明了问题时可解得，而且找到了解决的办法。

当然，这个时候还学习了mb_convert_encoding和iconv这两个函数，也算知道PHP中编码之间是如何转换的了。

转载请注明出处: http://zhupou.cn

UC_alipay测试（同时寻求帮助）

UC_alipay我已经开发了出来,整个流程已经可以跑通了,这个模块已经花了我将近一周的时间了,但是现在还有两个问题没有解决。

问题1：UTF-8和GBK之间的转换问题,drupal的模块都是采用的UTF-8的编码,我传给支付宝网关的input_charset也设置为了UTF-8，但是我传递到支付宝的汉字编成了乱码。

问题2：通知验证总是失败，支付宝将信息返回来以后，我对这些参数信息规整后，发送到支付宝请求验证这些数据的真伪性，但是每次都是返回失败。验证是使用MD5签名的，我觉得这个问题很有可能是由第一个引起的。当然也不排除其它原因。

其它的接口问题，我已经搞定。现在网上放置的测试模块，对于第一个问题，我传递的都是英文，对于第2个问题，直接设定了验证结果为真。所以可以跑通。

其它已经没有难点了，不知道这两个问题什么时候能够解决，希望能够尽快，代码写的很乱，现在只要求的是功能，因为我基本上采用的是支付宝给出的示例代码，Ubercart的钩子函数都已经搞定了。

如果有人能够帮助解决问题1和问题2的话，那是最好不过了，希望懂得编码转换，MD5签名的人能够参与进来，成为这个模块的共同开发者。

问题2不解决，就会存在安全方面的隐患，当然这个可以通过人工的检查来避免。问题1也需要解决，因为这个是给中国人用的。

Ubercart2.x简体中文汉化包

本简体中文包由chingwen75赞助，感谢他的支持。

下载地址：

http://zhupou.cn/download/ubercart-6.x-2.0-beta1-zh-hans.zip。

使用方法：

下载后，进入界面翻译管理页面，点击导入，导入该PO文件。汉化包只有一个PO文件，而不是drupal6的各个小PO文件的形式。

汉化程度：100%汉化了，不信你看看。有21个字符串有一点点问题，导入不了，但是已经存在于汉化包中了。基本上不影响。

Ubercart的POT文件包含了3159个字符串，所以汉化的工作是非常辛苦的。前前后后花了1个月的时间，它的工作量不亚于drupal6核心的汉化，drupal6有3534个字符串，Ubercart只比它少300+个。但是drupal6汉化的时候已经有了drupal5的汉化包的，虽然很多不能用，但是毕竟还有一部分可用。而Ubercart的基本上没有可用的。

所以我希望使用这个汉化包的人，能够给我捐上10元钱，或者在网站上保留Ubercart支持，zhupou汉化，一个开源的电子商务套件.这样的字段，当然你也可以把它去掉，在后台的配置页面，很简单就可以把这个给去了，当然使用汉化包本身就是对汉化者的支持。所以捐不捐钱，加不加链接都可以的^_^,如果发现里面有错，发现翻译的不贴切的地方，告诉我，在这里。我会逐步完善这个汉化包的。

对于汉化包，今年2009年，我都会完善它的，随着新版本的发布，另外打算建立一个汉化的服务器网站，专门管理这个汉化包，由我专门管理。localization server 这个模块大家应该知道吧，使用它搭建一个翻译站，有人发现更好的翻译了，就在上面贴出来。

本站不打算提供对Ubercart1.x汉化的支持，虽然很多都可以用，如果有人愿意的话，可以自己去整理。

里面有问题，是很正常的，问题很多，翻译的时候，这样的那样的，这些都期望大家共同的努力，来解决这些问题，来完善这个汉化包。在Ubercart页面的页脚中，原有英文版本有指向ubercart.org的链接，我把它翻译成中文的时候，加了zhupou汉化这样的字眼，希望能够把这4个字保留下来，作为对本站的尊重。

最后，为了感谢chingwen75赞助，感谢他的支持，在这里再给它加一个链接。

Ubercart开发指南

这本开发指南是为那些想扩展Ubercart功能,以满足自己需要的模块开发者准备的.模块扩展可能包括新的支付网关,配送服务,产品类型等等.本指南将为大家提供Ubercart核心模块中定义的API,介绍有哪些钩子函数,以及如何使用这些钩子函数.如果有漏掉的章节的话,你可以在文档版面中给我们提供相应的建议.

随着Ubercart的日趋成熟,我们将会为不同的Drupal版本提供不同Ubercart支持.使用下面的指南,来确定你所使用的版本

- Drupal 5 - Ubercart 1.x
- Drupal 6 - Ubercart 2.x （仍处于开发阶段）

你可以在http://api.ubercart.org找到更多的文档,这些都是根据模块文件中的注视自动生成的.

为了将你的模块从一个Ubercart版本升级到下一个版本,你可以参看,Ubercart API 升级说明.

- 代码标准
- ,Ubercart API 升级说明.
- Ubercart 1.x开发指南

- Ubercart 2.x开发指南

开发工具

相关链接: http://zhupou.cn http://www.ubercart.org/docs/developer

Ubercart API 的版本更新

这些页面，为你提供Ubercart API在不同版本之间的变化，以帮助你对你的模块进行更新。有时候，在同一版本的不同的稳定版之间，会有些改动，但是我们会尽量把这种改动减到最小。但是在大的版本之间进行升级时，你需要考虑很多的变动：我们将尽可能的将它们列到这里，同时也欢迎大家通过页面评论来完善和补充这一文档。

- 同一版本的不同稳定版之间的更新
- 将模块从Ubercart 1.x转化到2.x

相关链接： http://zhupou.cn

http://www.ubercart.org/docs/developer/6863/ubercart_api_updates_release

从Ubercart 1.3到1.4的API变动

Index of changes:

变动索引：

1. hook_calculate_tax() was added
2. 添加了hook_calculate_tax()

hook_calculate_tax() was added

添加了hook_calculate_tax()

This hook allows Ubercart to use multiple tax calculation schemes on its orders. The only argument is the entire order object for which the taxes will be calculated.

这个钩子让Ubercart可以对它的订单使用多个税收计算方案。这里面仅有的一个参数就是，所要计算的订单对象/

The hook implementations should return an array of tax line items, which are arrays with the following keys:

钩子函数应该返回一个数组，里面包含了税收行项（line items），里面用到以下键：

- id
- name
- amount
- weight (optional)

Make sure that your module's tax ids are properly namespaced to prevent collisions with other taxes.

确保你模块的税收ids使用了合适的命名空间，以避免与其它的税收冲突。

相关链接： http://zhupou.cn

http://www.ubercart.org/docs/developer/6867/api_changes_ubercart_13_14

从Ubercart 1.5 到1.6的API变动

变动索引：

1. 删除了uc_credit.module中的_array_convert()
2. 在uc_credit.module中，添加了uc_credit_default_gateway()
3. uc_recurring_fee_load()不再使用反序列化的数据字段

删除了uc_credit.module中的_array_convert()

我们把uc_credit.module中没用的函数_array_convert()删除了。它在核心中有点多余。对于那些使用了该函数的第3方模块，你可以把该函数包含到你的模块中，当然，需要改改函数名称，或者你也可以使用其它方式实现同样的功能

向uc_credit.module中添加了uc_credit_default_gateway()

我们向模块中加入了函数uc_credit_default_gateway(),这样就允许模块对于简单的信用卡交易使用默认的支付网关.如果你的模块,需要检查特定支付网关的可用的信用卡交易类型时,这个函数会非常有用.

uc_recurring_fee_load()不再对数据字段反序列化了

周期性收费系统,在对用户收取周期性费用时,允许你将数据存放到一个data列中。当对该列使用加载函数时，它将自动尝试对该列进行反序列化（unserialize），但是在保存该列数据时，却没有进行自动的序列化。对于这种反序列化，如果向数据列中存储一些简单的文本时，在以后的查询时，就会比较麻烦。

解决的方案就是删除了自动的反序列化，对于那些需要序列化的模块，这一功能由它们自己负责。现在，当他们在加载这一列数据时，也要自己进行处理了。

相关链接： http://zhupou.cn

http://www.ubercart.org/docs/developer/6865/api_changes_ubercart_15_16

代码标准

Ubercart核心代码遵守Drupal代码标准。第3方的模块，如果想把模块贡献到我们的网站上的话，必须遵守这些标准。

由于我们还在不断完善中，我们将在今后把其它的需求也列到这里。这些应该更多的告诉你，如何使用Ubercart的 方式来修改购物车的功能，而不是使用其它方式（如果存在一个Ubercart钩子供你使用，你却没有使用，即使你使用的是Drupal的钩子函数，我们也会把它看作一种不好的方式）

Ubercart模块的基本结构如下所示：

1. 文件信息
2. 钩子函数: Drupal, TAPIr, Ubercart
3. 回调函数用于菜单项,表单,表格,结算和订单窗格,行项等等.表单函数应该适用下面的顺序: builder, theme, validate, submit.
4. 模块和帮助函数(特定模块的函数,前缀使用_)

附件是一个可以下载的骨架模块.你在开发模块的时候,可以把它当作起点.只需要修改里面的合适的文本,删除不需要的部分,就可以开始编程了!

在下载包中还包括了一个.info文件。你可以在这里参看一下.info文件的格式：http://drupal.org/node/101009。你将注意到里面的双引号是可选的，但是我觉得最好我们都把它带上。（特别对于一些第3方模块中，里面包含标点符号时，最好加上双引号！）

你还会注意到，没有必要使用umlaut来代替U，使用Übercart是德语的用法。如果大家使用Ubercart的话，也没有关系，我们自己有时也用Ubercart。

附件	大小
skeleton.tar	4 KB

- 文档语法

相关链接：http://zhupou.cn

http://www.ubercart.org/docs/developer/238/coding_standards

同一版本不同稳定版之间的更新

在开发稳定版的同时，我们会尽力把相应的API的更新放到这里。在这些页面所列的修改，就意味着这一变动已经保存在了Bazaar资源库中了，同时也将被以用于下一个稳定版本中去。

- 从Ubercart 1.3到1.4的API变动
- 从Ubercart 1.5到1.6的API变动

相关链接： http://zhupou.cn

http://www.ubercart.org/docs/developer/6864/minor_release_updates

文档语法

当编写在线的代码文档时,很难为代码确定一个统一的标准.我们在这里给出一些简单的规则,供大家使用:

- 函数和钩子名称应使用粗体.例如:hook_perm()
- 文件名称应该使用斜体.例如: uc_store.module
- 变量名称名称应使用粗体.例如: $op
- 对变量类型的引用应该使用斜体.例如: string
- 代码样例应该使用PHP标签,以进行高亮显示.对于输入格式设置为PHP的页面,使用pre标签来显示代码样例：<pre style="border: solid 1px #bbb; padding: 5px;"></pre>
- 菜单标题应该使用粗体和更大字体。例如：Administer（管理） > Site building（站点构建） > Blocks（区块）

对于钩子文档，当你使用表格来显示键值时，这里有一组CSS class集供你使用。（当然，你也可以在别的地方使用这一表格）。将页面的输入格式设为Full HTML，对表格使用类"array_keys"，对于标题行使用"header"，对于键型单元格使用"key"，对于数据类型的单元格使用"type"，对于描述型单元格使用"value"。对于可选的值，在键的名称后面加一个*，并在整个表格的最下面加上一行，里面包含以下内容，这里使用斜体：*键是可选的。对于可选键的描述中，应该给出当被忽略时，所使用的默认值。

相关链接：http://zhupou.cn

http://www.ubercart.org/docs/developer/239/documentation_synatx

hook_add_to_cart

函数hook_add_to_cart()位于uc_cart.module:

```
<?php
  hook_add_to_cart($nid, $qty, $data)
?>
```

描述：

当向购物车中添加商品时，有些模块需要在这里流程中，进行一些逻辑处理。例如，一个库存系统，需要在此时检查库存情况，以防止将已经脱销的商品添加到购物车中。使用这个钩子，开发者可以在添加到购物车这一流程的最后阶段，进行一些逻辑处理，此时已经完整地加载了该商品的信息，并且该商品即将被添加到购物车上。如果该商品由于一些原因，不能被添加到购物车的话，你可以按照下面的描述，简单的返回一个失败信息。当然，也可以使用这个钩子，在商品被添加到购物车的时候，简单的进行一些日常处理。

参数：

- $nid -该商品的节点 ID
- $qty - 该商品添加到购物车中的数量
- $data -数据输租，里面包含了属性和型号调整

返回值：

无论你是出于什么目的，都可以使用这个函数来查看商品是否应该被添加到购物车中。这个函数应该返回一个包含结果数组的数组（这一点是由于Drupal的module_invoke_all()函数的特点决定的。你必须返回一个数组的数组，否则的话其它模块中的数据将被忽略掉。）现在，这个数组有两个键：

- success -当商品可以添加到购物车时返回TRUE，否则返回FALSE；默认为TRUE。
- message -当添加失败时，显示的失败信息；如果忽略的话，Ubercart将显示一个默认的失败信息。
- silent -返回TRUE时，将不会显示任何信息；如果一个模块在添加到购物车时，做了一些其它处理，或者在失败的时候也不想返回错误信息时，就可以用到这个键了。

现在，这个钩子仅用于当商品被添加到购物车时。到了Alpha 7版，可能会有些变动，但是，现在如果你的模块还需要检查购物车查看表单中的数量时，你可以使用hook_form_alter，为该表单添加一个验证函数。该表单的表单ID为uc_cart_view_form.

你在函数中，可以使用uc_cart_get_contents() 来得到客户购物车的当前情况。

例如：

```
<?php
function simple_inventory_add_to_cart($nid, $qty, $data) {
  if ($qty > 1) {
    $result[] = array(
      'success' => FALSE,
      'message' => t('Sorry, you can only add one of those at a time.'),
    );
  }
  return $result;
}
?>
```

相关链接：http://zhupou.cn http://ubercart.org

hook_calculate_tax

函数hook_calculate_tax()位于uc_taxes.module:

```
<?php
  hook_calculate_tax ($order)
?>
```

描述：

为一个订单计算税收行项（line items）

参数：

- $order -进行税收计算的订单。可能是一个完整的订单对象，或者是一个订单 i d 。

返回值：

一个税收行项数组。这个数组的键就是行项 i d s ，所以在所有的税收中，它们必须保证唯一性。每一个行项都是一个关联数组，里面包含了税收的 i d ， 名称， 总额， 和重量。

相关链接: http://zhupou.cn http://ubercart.org

hook_cart_display

函数hook_cart_display()位于uc_cart.module模块中：

```
<?php
  hook_cart_display($item)
?>
```

描述：

产品类型模块决定了哪些节点可以被添加到购物车中。而购物车通过使用这个钩子来决定如何显示这个钩子。对于产品包来说，这一点就非常重要了，因为，即便是它代表了多个项目，它在购物车中可能也就显示为一个单元。

参数：

- $item - 购物车中要显示的项目。

返回值：

一个表单数组，包含了以下元素：

键	#type	#value
nid	value	$item的节点id
module	value	实现了这个钩子，并且与$item所代表的节点对应的模块

remove	checkbox	如果选中的话，那么该项$item将被从购物车中删除。
options	markup	显示额外信息的主体化了的标识文本markup（通常为一个无须列表）
title	markup	$item所要显示的标题
#total	float	$item的价格。注意，这里'#' 意味着这不是一个表单元素，而仅仅是一个保存在表单数组中的值。
data	hidden	序列化了的d $item->data
qty	textfield	购物车中该项的数量。当点击了“更新购物车”时，客户输入的数量将被保存到购物车中。

Example:

```
<?php
function uc_product_cart_display($item){
  $node = node_load($item->nid);
  $element = array();
  $element['nid'] = array('#type' => 'value', '#value' => $node->nid);
  $element['module'] = array('#type' => 'value', '#value' => 'uc_product');
  $element['remove'] = array('#type' => 'checkbox');
  $op_names = '';
  if (module_exists('uc_attribute')){
    $op_names = "<ul class=\"cart-options\">\n";
    foreach ($item->options as $option){
      $op_names .= '<li>'. $option['attribute'] .': '. $option['name'] ."</li>\n";
    }
    $op_names .= "</ul>\n";
  }
  $element['options'] = array('#value' => $op_names);
  $element['title'] = array(
    '#value' => l($node->title, 'node/'. $node->nid),
  );
  $element['#total'] = $item->price * $item->qty;
  $element['data'] = array('#type' => 'hidden', '#value' => serialize($item->data));
  $element['qty'] = array(
    '#type' => 'textfield',
    '#default_value' => $item->qty,
    '#size' => 3,
    '#maxlength' => 3
  );
  return $element;
}
?>
```

相关链接：http://zhupou.cn ，http://ubercart.org

hook_cart_item

函数hook_cart_item()位于uc_cart.module模块中：

```
<?php
  hook_cart_item($op, &$item)
?>
```

描述：

产品在添加到购物车以后，在交易完成以前，被称为项目（ｉｔｅｍｓ）。比如在一家杂货店里面的货架上面，有一批产品，但是在旁边的说明中写道“最多１５项”。在产品的状态转变的多个时期，比如添加到购物车时，保存时，加载时，以及结算时，hook_cart_item()都可对产品项进行处理。

下面是使用这个钩子的理由：在一个正在运营的站点中，产品价格可能会由于属性的调整或者其它原因，进行变动。如果一个用户已经向购物车中添加了一项，而该项的价格如果调整了的话，那么当他们进行结帐时，或者查看购物车时，我们就想为其显示最新的价格和产品型号。也就是说，重要的产品信息保存在购物车中，但是在加载购物车中的项目时，其它模块可根据最新的设置对其进行调整。

参数：

- $op -当前的动作。可能的值有：
  load -当在函数uc_cart_get_contents()中加载购物车时传递的每一项。其它模块可在此时，对购物车中的项目信息进行调整。没有返回值。
  can_ship -如果购物车中，所有的项目都不需要运送的话，浏览购物车时，就传递这个。如果购物车中的产品都不需要运送的话，那么Übercart将会把有关配送方面的信息去掉。当产品可运送时，hook_cart_item检查这个操作时，就会放回TRUE，否则返回FALSE
- $item - 购物车中项目的对象。

返回值：

对于load没有返回值。

对于can_ship可返回TRUE 或者FALSE。

例如：

```
<?php
function uc_manufacturer_cart_item($op, &$item) {
  switch ($op) {
    case 'load':
      $term = array_shift(taxonomy_node_get_terms_by_vocabulary($item->nid,
variable_get('uc_manufacturer_vid', 0)));
      $arg1->manufacturer = $term->name;
      break;
  }
}
?>
```

相关链接：http://zhupou.cn http://ubercart.org

hook_cart_item

函数hook_cart_item()位于uc_cart.module模块中：

```
<?php
  hook_cart_item($op, &$item)
?>
```

描述：

产品在添加到购物车以后，在交易完成以前，被称为项目（ｉｔｅｍｓ）。比如在一家杂货店里面的货架上面，有一批产品，但是在旁边的说明中写道“最多１５项”。在产品的状态转变的多个时期，比如添加到购物车时，保存时，加载时，以及结算时，hook_cart_item()都可对产品项进行处理。

下面是使用这个钩子的理由：在一个正在运营的站点中，产品价格可能会由于属性的调整或者其它原因，进行变动。如果一个用户已经向购物车中添加了一项，而该项的价格如果调整了的话，那么当他们进行结帐时，或者查看购物车时，我们就想为其显示最新的价格和产品型号。也就是说，重要的产品信息保存在购物车中，但是在加载购物车中的项目时，其它模块可根据最新的设置对其进行调整。

参数：

- $op -当前的动作。可能的值有：
  load -当在函数uc_cart_get_contents()中加载购物车时传递的每一项。其它模块可在此时，对购物车中的项目信息进行调整。没有返回值。
  can_ship -如果购物车中，所有的项目都不需要运送的话，浏览购物车时，就传递这个。如果购物车中的产品都不需要运送的话，那么Übercart将会把有关配送方面的信息去掉。当产品可运送时，hook_cart_item检查这个操作时，就会放回TRUE，否则返回FALSE
- $item - 购物车中项目的对象。

返回值：

对于load没有返回值。

对于can_ship可返回TRUE 或者FALSE。

例如：

```
<?php
function uc_manufacturer_cart_item($op, &$item) {
  switch ($op) {
    case 'load':
      $term = array_shift(taxonomy_node_get_terms_by_vocabulary($item->nid,
variable_get('uc_manufacturer_vid', 0)));
      $arg1->manufacturer = $term->name;
      break;
  }
}
```

?>

hook_cart_pane

函数hook_cart_pane()位于uc_cart.module模块中:

```
<?php
  hook_cart_pane($items)
?>
```

描述:

默认的购物车查看页面，显示了一个表格，里面包含了购物车中的内容，以及一些简单的表单特性来管理购物车中的内容。如果你的模块需要向这个页面添加信息的话，你必须使用hook_cart_pane来定义你自己的窗格，这些窗格可以出现在默认信息的上面或者下面。

参数:

$items -当前购物车中的内容。

返回值:

这个函数将返回一个包含窗格数组的数组，窗格数组中的键如下所示:

键	类型	值
id	string	窗格的内部ID，可以使用a-z, 0-9, 和- 或者 _。
title	string	显示给用户的购物车窗格的名字。使用t().
enabled	boolean	窗格的默认启用情况。（默认为TRUE）
weight	integer	窗格的重量，这个决定了它显示的顺序。（默认为0）
body	string	窗格的内容，先是在购物车查看页面中。

如果窗格显示在购物车查看页面的话，那么body（正文）将被显示出来。对于设置页面，正文字段将被忽略。在处理任何查询或者foreach循环时，你最好在你的函数中检查一下NULL参数。

例如:

```
<?php
function uc_cart_cart_pane($items) {
  $panes[] = array(
    'id' => 'cart_form',
    'title' => t('Default cart form'),
    'enabled' => TRUE,
    'weight' => 0,
    'body' => !is_null($items) ? drupal_get_form('uc_cart_view_form', $items) : '',
  );
```

```
  return $panes;
}
?>
```

相关链接: http://zhupou.cn

hook_checkout_pane

函数hook_checkout_pane()位于uc_cart.module模块中：

```
<?php
  hook_checkout_pane()
?>
```

描述：

Ubercart的结算页面，汇集了所有启用了的结算窗格。结算窗格可以用来显示订单信息，从用户那里收集数据，或者与其它窗格进行交互。模块可以使用hook_checkout_pane()来定义结算窗格，通过特定的回调函数来进行逻辑处理和显示窗格。对于每个窗格，里面的一些设置都可以在结算设置页面进行配置，当然，在钩子函数中你需要声明这些设置的默认值。

默认的窗格定义在uc_cart.module模块的uc_cart_checkout_pane()函数中。这些窗格包括，显示购物车内容的窗格，收集用户重要信息的窗格，一个交货地址窗格，一个支付地址窗格，还有订单评论。其它的模块也提供了一些交货和支付方面的窗格。

返回值：

结算窗格数组的数组，使用以下键：

键	类型	值
id	string	结算窗格的内部ID，可以使用a-z，0-9，和- 或者_.
title	string	显示在结算表单的窗格的名称。
desc	string	在管理页面显示的窗格的简短描述。
callback	string	该窗格的回调函数。结算窗格回调函数的更多文档和例子可参看这一页面。
weight	int	窗格的默认重量，这个决定了它在结算表单中显示的顺序。
enabled	bool	可选。窗格的默认启用情况。默认为TRUE。
process	bool	可选。当结算表单提交时，这个窗格是否需要进行处理。默认为TRUE。

collapsible	bool	可选。这个窗格是否显示为可伸缩的字段集。默认为TRUE。

例如：

```
<?php
// Taken from uc_cart_checkout_pane() in uc_cart.module.
function uc_cart_checkout_pane() {
  $panes[] = array(
    'id' => 'cart',
    'callback' => 'uc_checkout_pane_cart',
    'title' => t('Cart Contents'),
    'desc' => t("Display the contents of a customer's shopping cart."),
    'weight' => 1,
    'process' => FALSE,
    'collapsible' => FALSE,
  );
  return $panes;
}
?>
```

还可参看：

结算系统文档-里面包含了结算窗格回调函数的文档。
提示追踪 - 一个基本的模块，演示了如何创建一个结算窗格，保存收集的信息，以及将窗格显示在订单页面。
相关链接：http://zhupou.cn

hook_file_action

函数位于模块*uc_file.module*中：

```
<?php
  hook_file_action($op, $args)
?>
```

描述：

uc_file模块带了一个文件管理器（位于Drupal后台界面Administer » Store administration » Products » View file downloads），它提供了一些基本的功能：删除多个文件和目录，上传单个文件（那些想一次上传多个文件的话，可以直接使用FTP来上传到文件下载目录，而文件管理器能够自动识别该目录下的文件）。使用这个钩子函数，开发者可以为文件管理器添加更多的功能。

参数：

- $op -该钩子正在采用的操作，可用的操作如下所示。

$args -这个参数，会根据op的不同而变化，可能的参数如下所示。

$op的值

'info'

在uc_file模块构建可用的文件动作列表以前调用这个操作。这个操作用来定义新的动作，这些动作将被放到文件动作下拉选择框中。

$args - 无

返回值： 关联数组，里面包含了可能进行的动作。其中键为动作的唯一标识。而值则显示在文件动作下拉框的列表中。

'insert'

当uc_file在文件下载目录发现一个新文件以后，调用该操作。

$args['file_object'] - 最新发现的文件的对象

返回值：无

'form'

当定义的文件动作被选中，并提交到表单时，将会调用这个函数，以显示接下来的表单。由于不管哪个模块定义的文件动作被选中时，都会调用这个函数，所以可以使用$args['action']来定义一个新的表单，或者将其追加到一个已有的表单中。

$args['action'] -正在处理的文件动作， 文件动作定义在hook_file_action($op = 'info')
$args['file_ids'] - n当前动作所处理的所选文件的ids（定义在uc_files表中）

返回值：这个操作将返回一个数组，里面包含了使用drupal表单API定义的Drupal表单元素。

'upload'

当一个文件，通过文件管理器内建的文件上传功能，被上传以后，并被移进文件下载目录，在这个文件被放到uc_files表以前，这个操作op可以对文件进行一些其它的剩余操作。

$args['file_object'] -移进文件下载目录中的文件的对象
$args['form_id'] - 函数的form_id变量
$args['form_values'] -函数的form_values变量

返回值：无

'upload_validate'

通过文件管理器内建的文件上传功能，上传文件以后，为了验证上传的文件，就会调用该操作。此时，文件已经被上传到了PHP的临时目录中。通过本步验证的文件，将被放到文

件下载目录中。

$args['file_object'] - 已经上传到PHP的临时上传目录中的文件的文件对象。

$args['form_id'] -函数的form_id变量
$args['form_values'] -函数的form_values变量

返回值：无

'validate'
当验证文件动作表单时，调用这个操作

$args['form_id'] -函数的form_id变量
$args['form_values'] -函数的form_values变量

返回值：无

'submit'
为了提交文件动作表单，就会调用这个操作

$args['form_id'] -函数的form_id变量
$args['form_values'] -函数的form_values变量

返回值：无

返回值：

这个钩子的返回值，取决于当前进行的操作，可能的返回值，已经在前面给出了。

例如：

```
<?php
/* The following is an example of a hypothetical module that adds an
* image watermark for images sold through a store
*/
function uc_image_watermark_file_action($op, $args)

  switch ($op) {
    case 'info':
      return array('uc_image_watermark_add_mark' => 'Add Watermark');
      break;
    case 'insert':
      //automatically adds watermarks to any new files that are uploaded to the file
download directory
      _add_watermark($args['file_object']->filepath);
      break;
    case 'form':
      if ($args['action'] == 'uc_image_watermark_add_mark') {
        $form['watermark_text'] = array(
          '#type' => 'textfield',
          '#title' => t('Watermark Text'),
        );
```

```
        $form['submit_watermark'] = array(
          '#type' => 'submit',
          '#value' => t('Add Watermark'),
        );
      }
      return $form;
      break;
    case 'upload':
      _add_watermark($args['file_object']->filepath);
      break;
    case 'upload_validate':
      //Given a file path, function checks if file is valid JPEG
      if(!_check_image($args['file_object']->filepath)) {
        form_set_error('upload',t('Uploaded file is not a valid JPEG'));
      }
      break;
    case 'validate':
      if ($args['form_values']['action'] == 'uc_image_watermark_add_mark') {
        if (empty($args['form_values']['watermark_text'])) {
          form_set_error('watermar_text',t('Must fill in text'));
        }
      }
      break;
    case 'submit':
      if ($args['form_values']['action'] == 'uc_image_watermark_add_mark') {
        foreach ($args['form_values']['file_ids'] as $file_id) {
          $filename = db_result(db_query("SELECT filename FROM {uc_files} WHERE fid
= %d",$file_id));
          //Function adds watermark to image
          _add_watermark($filename);
        }
      }
      break;
  }
?>
```

hook_file_transfer_alter

函数hook_file_transfer_alter()位于uc_file.module模块中：

```
<?php
  hook_file_transfer_alter($file_user, $ip, $fid, $file)
?>
```

描述：

对于drupal网店，无论是出于个性化的原因，还是出于版权保护的原因，或者其它原因，可能想要给客户发送定制化的下载文件。使用这个钩子就可以实现这一点。在将文件传递给客户以前，将会调用这个钩子，来对文件进行一些修改，从而实现文件对于客户而言的定制化。实际上，开发者利用这一点，就可以创建一个新的 ，个性化的文件，来传递给相应的客户。

参数：

- $file_user -正在下载文件的用户所对应的对象（比如，一个包含了一个uc_file_users表中一行记录的对象）。

- $ip - 客户下载文件所用的IP地址
- $fid - 正在传递的文件的文件id
- $file -正在传递的文件的路径

返回值：

要传递给客户的新文件的路径

例如：

```
<?php
/* Inside personal_text module - will place personalized text at the end of a file
*/
function personal_text_file_transfer_alter($file_user, $ip, $fid, $file) {
  $file_data = file_get_contents($file)." [insert personalized data]"; //for large
files this might be too memory intensive
  $new_file = tempnam(file_directory_temp(),'tmp');
  file_put_contents($new_file,$file_data);
  return $new_file;
}
?>
```

相关链接：http://zhupou.cn

hook_line_item

函数hook_line_item()位于uc_order.module模块中：

```
<?php
  hook_line_item()
?>
```

描述：

hook_line_item是用来定义行项的，附带在订单的下面。一个行项可以表示收费、费用，以及订单的总计。默认的行项包括了小计和总计行项，税收行项，以及运送行项。还有一个更以本的行项，网店管理员可以使用它来向手工创建的订单添加额外的费用和折扣。模块开发者，可以使用这个钩子为他们的网店定义新的行项。常见的例子，就是优惠券，网店允许自己的客户使用优惠券，这个时候就可以把优惠券实现为一个行项。

一旦在hook_line_item中定义了一个行项，Übercart就会与它进行交互，包括多种不同的方式。最主要的一种方式是，你为这个行项声明一个回调函数。函数的结构如下所示：

```
<?php
function callback_name($op, $arg1) {
  switch ($op) {
    case '...':  // Actual case values described in table below.
```

```
      break;
  }
}
?>
```

当调用这个函数时，$op的值用来声明Übercart期望接受的返回信息的类型。下面列出了目前的所有情况：

<table>
<tr><th>$op</th><th>使用/返回 值</th></tr>
<tr><td>load</td><td>当行项被加载时，没有存储在数据库中的项将通过这个函数来加载进来。$arg1是订单对象。期望的返回值，是一个关联数组，包含了以下键：
<table>
<tr><th>键</th><th>类型</th><th>值</th></tr>
<tr><td>id</td><td>string</td><td>当显示时，用于标识行项的ID</td></tr>
<tr><td>title</td><td>string</td><td>显示在订单上的行项的标题。使用t()。</td></tr>
<tr><td>amount</td><td>float</td><td>显示在订单上的行项的值。</td></tr>
</table>
例如：
<pre>
<?php
  case 'load':
    $lines[] = array(
      'id' => 'subtotal',
      'title' => t('Subtotal'),
      'amount' => uc_order_get_total($arg1, TRUE),
    );
    return $lines;
?>
</pre></td></tr>
<tr><td>display</td><td>对于那些设为仅仅用来显示的行项，当显示订单时，使用这个$op就可以调用它们的回调函数。$arg1是订单对象。期望的返回值，和前面的load的完全一样。</td></tr>
<tr><td>cart-preview</td><td>在客户结算期间，支付窗格将会显示一个预览页面，里面包含了行项和订单。对于页面的初始化加载，每一个行项都回通过这个$op来调用它们的回调函数。通过使用drupal_add_js()来调用下面描述的Javascript函数set_line_item(key, title, value, weight)，从而设置行项的默认值。$arg1是一个用于该行项的产品对象的数组，当然，这些产品对象当前出于购物车中。注意：这个不是必须的。这个仅用于默认的显示。结算页面的其</td></tr>
</table>

它窗格，可以通过在它们自己的Javascript文件中使用set_line_item()来向这个列表中动态的添加项目。这个函数的参数如下所示：

参数	类型	值
key	string	所显示的行项的唯一内部ID
title	string	行项的标题。你可以使用t()来建立这个值，当你的窗格加载时，使用drupal_add_js()可以把它作为一个全局变量添加到页面中。
value	float	行项的值。（用来显示和计算总计）
weight	int	行项的显示顺序。

例如：

```
<?php
  case 'cart-preview':
    $subtotal = 0;
    foreach ($arg1 as $item) {
      $total = ($item->qty) ? $item->qty * $item->price :
$item->price;
      $subtotal += $total;
    }
    drupal_add_js("\$(document).ready( function() {
set_line_item('subtotal', '". t('Subtotal') ."', ". $subtotal .",
-10); } );", 'inline');
    break;
?>
```

返回值：

你的钩子应该返回一个关联数组的数组，就像前面的例子一样。数组中的每一项，都代表着一个单独的行项，它将使用下面的键：

键	类型	值
id	string	行项的内部ID
title	string	在各种界面显示给用户的行项的标题。使用t().
callback	string	行项的回调函数的名称，可用于不同操作。
weight	int	行项在列表中的顺序；重量越小，越靠前。

stored	bool	行项是否保存在数据库中。可通过订单编辑页面修改的行项，都是TRUE。
add_list	bool	行项是否包含在订单编辑页面的“添加一个行项”的下拉选择框中。
calculated	bool	行项的值，是否应该包含在订单总计中。（例如：对于运费行项，返回TRUE，而对于小计行项，则返回FALSE，因为产品价格已经包含进了总额中）
display_only	bool	如果这个行项仅仅用来显示信息，而在其它地方不进行计算的话，返回TRUE，否则返回FALSE。（例如：总计行项仅仅用来在行项列表的底部显示订单的总计，此时就返回TRUE）。

例如：

```
<?php
function uc_order_line_item() {
  $items[] = array(
    'id' => 'generic',
    'title' => t('Empty Line'),
    'weight' => 2,
    'default' => FALSE,
    'stored' => TRUE,
    'add_list' => TRUE,
    'calculated' => TRUE,
    'callback' => 'uc_line_item_generic',
  );

  return $items;
}
?>
```

注意，行项是定义在一个数组中的，这个和Drupal的hook_menu中的菜单项非常类似
相关链接：http://zhupou.cn

hook_order

函数hook_order()位于uc_order.module模块中：

```
<?php
  hook_order($op, &$arg1, $arg2)
?>
```

描述：

在Übercart中，一个订单代表着一个单独的交易。订单是在结算过程中创建的，此时，它们在数据库中的状态为在结帐。当客户完成结算时，订单的状态也会随着进行更新，已反映交易的进行状况。一旦订单创建以后，甚至在订单创建的过程中，其它的模块都可以向订单中添加额外的信息。每当对订单进行某项操作时，都会出发钩子hook_order()，从而让你的模块知道当前发生了什么，而你的模块根据当前情况，进行一些相应的处理。

参数：

- $op -当前进行的动作
- &$arg1 -这个是订单对象，或者指向订单的引用，下面有进一步的说明。
- $arg2 -这个变量将在下面的表格中说明

$op	用途
new	当一个订单创建时调用。$arg1 是对新订单对象的引用，所以模块可以在订单的创建时期添加或者修改订单。
save	当订单对象被保存时，这个钩子将触发这个操作，以让其它模块保存相应的信息。$arg1为订单对象的引用
load	当订单加载时，在订单和产品数据从数据库中加载以后，调用。$arg1为订单对象的引用，所以其它模块可以在订单对象加载时对其进行修改。
submit	当销售正在完成时，并且客户点击了结算页面的提交订单按钮，这个钩子将触发这个操作。这给了其它模块一个机会，来决定订单是否允许被提交。一个示例是，当订单提交时，信用卡模块试图处理支付，如果支付失败的话，将返回一个失败信息。 为了阻止订单提交的通过，你必须返回一个包含错误信息的数组，其格式如下所示： <?php return array(array('pass' => FALSE, 'message' => t('We were unable to process your credit card.'))); ?>
can_update	在订单的状态改变以前调用，从而确定订单是否可被更新。$arg1是订单对象，带有旧的订单状态ID（$arg1->order_status），而$arg2就是新的订单状态ID。如果由于某些原因，可以返回FALSE来停止订单的状态更新。
update	当订单状态更新时调用。$arg1是订单对象，带有旧的订单状态ID（$arg1->order_status），而$arg2就是新的订单状态ID。
can_delete	在订单被删除以前调用，以确认当前订单可被删除。返回FALSE可以阻止删除的发生。（例如，当订单已经受到付款时，支付模块默认返回FALSE）

返回值：

没有返回值

示例：

```
<?php
function uc_payment_order($op, &$arg1, $arg2) {
  switch ($op) {
    case 'save':
      // Do something to save payment info!
      break;
  }
}
?>
```

hook_order_pane

函数hook_order_pane()位于uc_order.module模块中：

```
<?php
  hook_order_pane()
?>
```

描述：

这个钩子是用来向订单查看和管理界面添加窗格的。默认的窗格包含了地址的显示和编辑区域，产品，评论，等等。当开发者需要向订单显示或者修改一些自定义的数据时，就可以使用这个钩子来完成。例如，使用了一个自定义的结算窗格，来查找客户期望的交货日期的Drupal网店，可以创建一个相应的订单窗格，来将数据显示到订单界面上。

在钩子hook_order_pane()中，可以定义一些新的订单窗格，并为它们提供一点信息。对于这个钩子可以定义订单窗格的哪些部分，可参看下面的返回值部分。

订单窗格的要点是它的回调函数（这个也在这个钩子中声明）。回调函数负责处理在哪些页面显示哪些内容，以及可以操作哪些数据。这些都超出了这个API页面的范围，为了了解回调函数中可以包含什么内容，你可以点击这里。

返回值：

函数应该返回一个关联数组，里面包含了你模块定义的订单窗格。这些数组包含了下面的键/值对：

键/	类型	值
id	string	订单窗格的内部ID
callback	string	订单窗格回调函数的名称，在各种操作中调用
title	string	

desc	string	
class	string	
weight	integer	
show	array	

示例：

```
<?php
/**
* Implementation of hook_order_pane().
*/
function uc_payment_order_pane() {
  $panes[] = array(
    'id' => 'payment',
    'callback' => 'uc_order_pane_payment',
    'title' => t('Payment'),
    'desc' => t('Specify and collect payment for an order.'),
    'class' => 'pos-left',
    'weight' => 4,
    'show' => array('view', 'edit', 'customer'),
  );
  return $panes;
}
?>
```

hook_payment_gateway

函数hook_payment_gateway()位于uc_payment.module模块中：

```
<?php
  hook_payment_gateway()
?>
```

描述：

支付网关是整个支付系统中的重要部分。它们在你的Ubercart和支付服务之间搭了一个桥梁，支付网关将会对你收到的支付信息进行处理。常见的支付服务的例子有PayPal 和Authorize.net。每个支付网关，都会使用一种不同的方式，让你把你从客户那里收集到的信息安全的提交过去，以完成支付。根据支付的成功和失败，它们还会返回相应的信息。

hook_payment_gateway()定义了支付网关，它让Ubercart知道哪些可用，使用的是什么支付方法，以及如何使用它们处理支付。

在支付网关数组中，你可以使用方法ID来引用方法。网关可以处理的每一个支付方法，都应该有一个键，它的值为一个函数的名字，在处理支付时，支付系统将会调用这个函数。（比如uc_payment.module模块中的uc_payment_process()函数）

每个回调函数，都应该能够接受同样的参数，其中$order_i代表着要付款的订单的ID，$amount为要支付的金额，而$data是传递给这个函数的一些其它的必要的数据，可以通过调用uc_payment_process()函数来传递这些数据。这个回调函数应该负责与网关服务的所有交互，并记录合适的日志信息。

在示例的信用卡网关模块中，仅仅假定了支付成功完成。它接着将信息记录到订单的管理评论中。最后，它准备一个网关回调的返回值，一个使用了以下键值的数组：

键	类型	值
success	bool	支付成功时，返回TRUE，否则返回FALSE
comment	string	在支付表格中，留给支付的评论。Use t().
message	string	这里显示给用户的消息,是通过普通的Drupal消息系统实现的.如果在结算过程中,付款是自动完成的话,那么这个消息就不再显示.使用t().
uid*	int	当前用户的ID,将被记录到支付表中.默认为0.
data*	mixed	你想要传递过来的各种数据,这些数据将被存储到支付表中.数组在被插入数据库以前将被自动的序列化.
log_payment*	bool	如果订单的支付被记录下来的话,那么返回TURE,否则返回FALSE;默认为TRUE.返回FALSE的一个示例情况是,使用了信用卡支付,现在还没有从里面扣钱.这样的支付还没有完成,也就不应该为订单记录这一支付,但是我们仍然想知道信用卡是不是有效的.
*键是可选的		

在你的回调函数中,你可以使用任何你创建的其它函数,或者和支付网关的web服务进行交互.有一点非常有用,那就是你收集的和保存了的支付信息,可以在订单对象的支付详细数组中查看.为了在你的回调函数中加在订单,可以这样:

```
<?php
  $order = uc_order_load($order_id);
  // Now you can access the payment details with $order->payment_details.
  drupal_set_message('<pre>'. print_r($order->payment_details, TRUE) .'</pre>');
?>
```

支付方法的注释文档,能够帮你了解该方法保存了哪些支付详细信息.这对于信用卡支付方法非常有用,因为你需要访问信用卡的信息,以处理你的订单!

返回值:

返回一个支付网关的数组,这个数组使用了以下键:

键	类型	值
id	string	支付网关的内部ID,使用a-z, 0-9, 和- 或者 _.
title	string	显示给用户的支付网关的名字.使用t().
description	string	支付网关的简洁描述.
settings	string	这是一个返回设置数组的函数的名字,数组里面包含的是用于网关的表单元素.
credit_txn_types	array	处理信用卡支付的支付网关可以使用这个可选键.它的值为一个常量数组,用来说明网关可以提供哪些信用卡交易类型.可用的常量有UC_CREDIT_AUTH_ONLY, UC_CREDIT_PRIOR_AUTH, UC_CREDIT_AUTH_CAPTURE, UC_CREDIT_REFERENCE_TXN;其中默认为UC_CREDIT_AUTH_CAPTURE,它将自动授权并且立即支付.

示例:

```
<?php
function test_gateway_payment_gateway() {
  $gateways[] = array(
    'id' => 'test_gateway',
    'title' => t('Test Gateway'),
    'description' => t('Process credit card payments through the Test Gateway.'),
    'credit' => 'test_gateway_charge',
  );
  return $gateways;
}
?>
```

在下面的例子中,显示的是测试网关的信用卡支付方法的回调函数:

```
<?php
function test_gateway_charge($order_id, $amount, $data) {
  global $user;
```

```
  $message = t('Credit card charged: !amount', array('!amount' =>
uc_currency_format($amount)));
  uc_order_comment_save($order_id, $user->uid, $message, 'admin');

  $result = array(
    'success' => TRUE,
    'comment' => t('Card charged, resolution code: 0022548315'),
    'message' => t('Credit card payment processed successfully.'),
    'uid' => $user->uid,
  );
  return $result;
}
?>
```

相关链接： http://ubercart.org http://zhupou.cn

hook_shipping_method

函数hook_shipping_method()位于uc_quote.module模块中：

```
<?php
  hook_shipping_method()
?>
```

描述：

运送报价控制模块，uc_quote，通过实现这个钩子来得到一个特定结构的方法数组。

运送方法和类型的重量和启用的旗帜，可以在网店配置下面的运送报价设置页面中进行设置。变量的键为运送方法的ids。方法的报价（quote）和运送（ship）数组都是可选的。

- type --报价和运送类型就是产品运送类型的ids，这些方法将应用于这些产品运送类型。Type还可以是'order'（订单），这意味着报价应用于整个订单，而不再考虑它的产品的运送类型。当报价方法是基于客户的地理位置，而不是他们购买的产品时，就会用到这一点。
- callback --当一个运送报价被请求时，uc_quote调用这个函数。它的参数是一个产品数组和一个订单详细（运送地址）的数组。返回值为一个数组，里面包含了为accessorials（辅助）数组中的每一项返回的费率和返回的错误信息（如果存在的话）。
- accessorials --这个数组代表了在运送方面可供客户选择的不同选项。回调函数应该为accessorials中的每一项生成一个报价，并通过数组将其返回。在这里，drupal_to_js()就非常有用了。

```
<?php
return array(
  '03' => array('rate' => 15.75, 'format' => uc_currency_format(15.75)
'option_label' => t('UPS Ground'),
                'error' => 'Additional handling charge automatically applied.'),
  '14' => array('error' => 'Invalid package type.'),
```

```
  '59' => array('rate' => 26.03, 'format' => uc_currency_format(26.03),
'option_label' => t('UPS 2nd Day Air A.M.'))
);
?>
```

- pkg_types --运送方法可以处理的包裹类型的列表。这应该是一个关联数组，可用作select（选择）表单元素的#options。推荐编写一个函数来输出这个数组，这样就不需要仅仅为包裹类型准备一个查找方法了

返回值：

一个数组，里面包含了在uc_quote和uc_shipping中可用的运送方法

示例：

```
<?php
function uc_ups_shipping_method(){
  $methods = array();

  $enabled = variable_get('uc_quote_enabled', array('ups' => true));
  $weight = variable_get('uc_quote_method_weight', array('ups' => 0));
  $methods['ups'] = array(
    'id' => 'ups',
    'title' => t('UPS'),
    'enabled' => $enabled['ups'],
    'weight' => $weight['ups'],
    'quote' => array(
      'type' => 'small package',
      'callback' => 'uc_ups_quote',
      'accessorials' => array(
        '03' => t('UPS Ground'),
        '11' => t('UPS Standard'),
        '01' => t('UPS Next Day Air'),
        '13' => t('UPS Next Day Air Saver'),
        '14' => t('UPS Next Day Early A.M.'),
        '02' => t('UPS 2nd Day Air'),
        '59' => t('UPS 2nd Day Air A.M.'),
        '12' => t('UPS 3-Day Select'),
      ),
    ),
    'ship' => array(
      'type' => 'small package',
      'callback' => 'uc_ups_fulfill_order',
      'pkg_types' => array(
        '01' => t('UPS Letter'),
        '02' => t('Customer Supplied Package'),
        '03' => t('Tube'),
        '04' => t('PAK'),
        '21' => t('UPS Express Box'),
        '24' => t('UPS 25KG Box'),
        '25' => t('UPS 10KG Box'),
        '30' => t('Pallet'),
      ),
    ),
  );
```

```
  return $methods;
}
?>
```

hook_shipping_type

函数hook_shipping_type()位于uc_quote.module模块中：

```
<?php
  hook_shipping_type()
?>
```

描述：

这个钩子该模块所要处理的运送类型。这些类型分别有一个内部名字'id'，和一个外部名字'title'。对于单独的产品，制造商，以及整个网店目录，都可以为其设置运送类型。运送模块应该在使用运送类型ids的时候，要小心，因为其它的类似模块也可能使用相同的名字（比如，FedEx 和UPS都有“小包裹”运送类型）。对于那些与运送无关的模块，则不需要实现这个钩子。

返回值：

一个数组，里面包含了运送类型，键为运送类型的内部名字

示例：

```
<?php
function uc_ups_shipping_type(){
  $weight = variable_get('uc_quote_type_weight', array('small_package' => 0));

  $types = array();
  $types['small_package'] = array(
    'id' => 'small_package',
    'title' => t('Small Packages'),
    'weight' => $weight['small_package'],
  );

  return $types;
}
?>
```

hook_store_status

函数hook_store_status()位于uc_store.module模块中：

```
<?php
  hook_store_status()
?>
```

描述：

这个钩子是用来向主网店管理界面上的网店状态表格中添加项目的。每一项对应表格中的一行，里面包含了状态图标，标题，描述。这些项目用来为模块创建的购物车组件提供特殊的指示，通知，或者指标。通过图标，网店店长就能够一眼看出你模块的关键组件是否正常工作。

例如，如果启用了目录模块，但是目录模块却找不到目录分类词汇表，那么它就会在这里显示一个错误信息，来警告网店店长。

网店状态项目定一在一个数组中，使用了下面的键：

- status -根据信息分别显示ok（正常），warning（警告），或者error（错误）
- title -状态信息的标题，或者模块定义的标题
- desc -描述；可以是任何信息，包括指向页面的链接，以及处理该问题的表单。

返回值：

一个包含了网店状态数组的数组，网店状态数组的定义如上所示。

示例：

```
<?php
// From uc_credit.module:
function uc_credit_store_status() {
  if ($key = uc_credit_encryption_key()) {
    $statuses[] = array(
      'status' => 'ok',
      'title' => t('Credit card encryption'),
      'desc' => t('Credit card data in the database is currently being encrypted.'),
    );
  }
  return $statuses;
}
?>
```

hook_uc_message

函数hook_uc_message()位于uc_store.module模块中：

```
<?php
  hook_uc_message()
?>
```

描述：

有很多例子中，Ubercart模块都会用到可配置的区块文本。这通常是一些默认消息，比如新订单的电子邮件模板。如果使用普通的默认值设置方式的话，你可能需要在模块中的不同位置，（当你希望使用变量，或者使用默认值来显示设置表单时），多次拷贝和粘贴大段的文本。为了

降低这种混乱，我们引入了这个钩子。使用这个钩子，你就可以把你的消息放到一个位置，而在其它需要的地方（或者其它模块）使用函数uc_get_message()来调用相应的文本就可以了。

这个函数非常简单，它没有参数，返回值为一个基本的关联数组，其中键为消息的IDs，而值为默认的消息。当你使用uc_get_message()时，使用你这里设置的消息ID来引用你想要的消息

注意：当使用t()时，你不能向里面传递一个串联的字符串！所以我们的例子中，没有使用换行符，尽管它的宽度超过了80个字符。使用串联就会影响字符串的翻译。

返回值：

一个关于消息的数组。

示例：

```
<?php
function uc_cart_uc_message() {
  $messages['configurable_message_example'] = t('This block of text represents a
configurable message such as a set of instructions or an e-mail template.  Using
hook_uc_message to handle the default values for these is so easy even your grandma
can do it!');

  return $messages;
}
?>
```

theme_uc_cart_block_content

函数theme_uc_cart_block_content()位于uc_cart.module模块中

```
<?php
  theme_uc_cart_block_content()
?>
```

描述：

这个函数负责购物车区块的主题化。默认情况下，这是购物车中的项目列表，每个项目都链接到它对应的产品页面。它还包括“查看购物车”和“结算”链接。

返回值：

一个字符串，包含了购物车区块内容的HTML输出。

示例：

```
<?php
// The default function from uc_cart.module.
function theme_uc_cart_block_content() {
  global $user;

  if (variable_get('uc_cart_show_help_text', FALSE)) {
    $output = '<span class="cart-help-text">'
            . variable_get('uc_cart_help_text', t('Click title to display cart
```

```
contents.'))
            .'</span>';
  }

  $output .= '<div id="block-cart-contents">';

  $items = uc_cart_get_contents();

  $item_count = 0;
  if (!empty($items)) {
    $output .= '<table class="cart-block-table">'
             .'<tbody class="cart-block-tbody">';
    foreach ($items as $item) {
      $output .= '<tr class="cart-block-item"><td class="cart-block-item-qty">'.
$item->qty .'x</td>'
              . '<td class="cart-block-item-title">'. l($item->title, 'node/'.
$item->nid) .'</td>'
              . '<td class="cart-block-item-price">'.
uc_currency_format($item->price) .'</td></tr>';
      if (is_array($item->data['attributes']) && !empty($item->data['attributes']))
{
        $display_item = module_invoke($item->module, 'cart_display', $item);
        $output .= '<tr><td colspan="3">'. $display_item['options']['#value']
.'</td></tr>';
      }
      $total += ($item->price) * $item->qty;
      $item_count += $item->qty;
    }

    $output .= '</tbody></table>';
  }
  else {
    $output .= '<p>'. t('There are no products in your shopping cart.') .'</p>';
  }

  $output .= '</div>';

  $item_text = format_plural($item_count, '1 Item', '@count Items');
  $view = '('. l(t('View cart'), 'cart', array('rel' => 'nofollow')) .')';
  if (variable_get('uc_checkout_enabled', TRUE)) {
    $checkout = ' ('. l(t('Checkout'), 'cart/checkout', array('rel' => 'nofollow'))
.')';
  }
  $output .= '<table class="cart-block-summary-table"><tbody class="cart-block-
summary-tbody">'
           .'<tr class="cart-block-summary-tr"><td class="cart-block-summary-
items">'
           . $item_text .'</td><td class="cart-block-summary-total">'
           .'<strong>'. t('Total:') .'</strong> '. uc_currency_format($total)
.'</td></tr>';
  if ($item_count > 0) {
    $output .= '<tr><td colspan="2" class="cart-block-summary-checkout">'. $view .
$checkout .'</td></tr>';
  }
  $output .= '</tbody></table>';

  return $output;
}
```

?>

theme_uc_cart_block_title

函数theme_uc_cart_block_title()位于uc_cart.module模块中：

```
<?php
  theme_uc_cart_block_title($cart_image, $uc_cart_path, $arrow_up_image)
?>
```

描述：

这个函数负责购物车区块标题栏的显示。你可以使用它来覆写该函数的部分功能，如果通过区块配置页面修改标题的话，那么会同时删除图片和Javascript功能，许多人都希望能够保留这些功能。

参数：

- $cart_image - 购物车图标的URI
- $uc_cart_path - 购物车模块目录的基路径
- $arrow_up_image - 向上箭头图片的URI

返回值：

返回一个字符串，包含了购物车区块标题的HTML输出。

示例：

```
<?php
// The default function from uc_cart.module.

function theme_uc_cart_block_title($cart_image, $uc_cart_path, $arrow_up_image) {
  $output = l('<img src="'. $cart_image .'" id="block-cart-title-image" alt="" />',
'cart', NULL, NULL, NULL, FALSE, TRUE)
            .'<span class="block-cart-title-bar" id="block-cart-title-bar-text"
onclick="cart_toggle(\''. $uc_cart_path .'\');">'
            .'<span id="block-cart-title">'. t('Shopping Cart') .'</span></span>'
            .'<span class="block-cart-title-bar" id="block-cart-title-bar-arrow"
onclick="cart_toggle(\''. $uc_cart_path .'\');">'
            .'<img id="block-cart-title-arrow" "src="'. $arrow_up_image .'"
alt="[]" title="'. t('Expand cart block.') .'" /></span>';

  return $output;
}
```

相关联接： http://ubercart.org http://zhupou.cn

theme_uc_cart_checkout_review

函数theme_uc_cart_checkout_review()位于uc_cart.module模块中：

```
<?php
```

```
  theme_uc_cart_checkout_review($help, $panes, $form)
?>
```

描述：

当你提交结算表单以后，进行检查时，你看到的页面，它使用的drupal主题函数就是这个。它简单的将多有的东西都放到了一个表格中，而没有使用过多的classes 和IDs进行控制.

我(Ryan)非常欢迎大家能够帮我改进Ubercart中这部分的主题函数。

参数：

- $help -一个包含了订单检查页面中帮助信息的字符串。
- $panes -. 一个关联数组，用于为每个结算窗格向订单检查页面添加信息。键就是窗格的标题，而值为一个为该窗格返回的数据或者数据的数组。
- $form -表单的HTML版本，默认包含了显示在检查页面底部的“回退”和“提交订单”按钮。

返回值：

一个字符串，包含了显示在订单检查页面的HTML输出。

例如：

```
<?php
// Default function from uc_cart.module.
function theme_uc_cart_checkout_review($help, $panes, $form) {
  drupal_add_css(drupal_get_path('module', 'uc_cart') .'/uc_cart.css');

  $output = '<div>'. check_markup(variable_get('uc_checkout_review_instructions',
uc_get_message('review_instructions')),
                      variable_get('uc_checkout_review_instructions_format', 3),
                      FALSE) . '</div><table class="order-review-table">';

  foreach ($panes as $title => $data) {
    $output .= '<tr class="pane-title-row"><td colspan="2">'. $title
               .'</td></tr>';
    if (is_array($data)) {
      foreach ($data as $row) {
        if (is_array($row)) {
          if (isset($row['border'])) {
            $border = ' class="row-border-'. $row['border'] .'"';
          }
          else {
            $border = '';
          }
          $output .= '<tr valign="top"'. $border .'><td class="title-col" '
                     .'nowrap>'. $row['title'] .':</td><td class="data-col">'
                    . $row['data'] .'</td></tr>';
        }
        else {
          $output .= '<tr valign="top"><td colspan="2">'. $row .'</td></tr>';
        }
      }
```

```
    }
    else {
      $output .= '<tr valign="top"><td colspan="2">'. $data .'</td></tr>';
    }
  }

  $output .= '<tr class="review-button-row"><td colspan="2">'. $form
           .'</td></tr></table>';

  return $output;
}
?>
```

相关联接 : http://ubercart.org http://zhupou.cn

theme_uc_catalog_browse

函数theme_uc_catalog_browse()位于uc_catalog.module模块中：

```
<?php
  theme_uc_catalog_browse($tid = 0)
?>
```

描述：

负责目录页面的格式，用来显示类别图片，子类别，和产品。

参数：

- $tid -要显示的类别的术语id。默认为0，这对应于目录的顶级。

返回值：

主题化了的HTML字符串，里面包括类别图片，图片，指向子类别的链接，以及该类别下面的产品。

例如：

```
<?php
uc_catalog_menu($may_cache) {
  $items[] = array(
    'path' => variable_get('uc_catalog_url', 'catalog'),
    'access' => TRUE,
    'title' => variable_get('uc_catalog_name', t('Catalog')),
    'callback' => 'theme',
    'callback arguments' => array('uc_catalog_browse'),
    'type' => MENU_NORMAL_ITEM,
  );

  return $items;
}
```

```
?>
```

theme_uc_empty_cart

函数theme_uc_empty_cart()位于uc_cart.module模块中：

```
<?php
  theme_uc_empty_cart()
?>
```

描述：

当用户访问购物车页面时，购物车中还没有商品的情况下，所显示的信息，由这个函数负责。

返回值：
一个字符串，包含了空购物车页面的HTML输出。

例如：

```
<?php
// Default function from uc_cart.module
function theme_uc_empty_cart() {
  return '<p>'. t('There are no products in your shopping cart.') . '</p>';
}
?>
```

相关链接: http://ubercart.org http://zhupou.cn

theme_uc_file_downloads_token

函数theme_uc_file_downloads_token()位于uc_file.module模块中：

```
<?php
  function theme_uc_file_downloads_token($file_downloads)
?>
```

描述：

这个主题函数负责为[file-downloads]令牌创建文件下载文本，该令牌可用于文件下载的邮件通知中。对于那些想覆写默认文本（包含文件下载链接和链接旁边的文本）的开发者，就可通过覆写这个主题函数来完成。

参数：

$file_downloads --一个数组，里面包含了与订单相关的文件下载对象（uc_file_users表中的记录）

返回值：
构成文件下载信息的HTML

示例：

```
<?php
    $output = '';
    foreach ($file_downloads as $file_download) {
        $filename = basename(db_result(db_query("SELECT filename FROM {uc_files}
WHERE `fid` = %d",$file_download->fid)));
        $download_url = url('download/'.$file_download->fid.'
/'.$file_download->key,NULL,NULL,TRUE);
        $output .= '<a href="'.$download_url.'">'.$download_url.'</a>'."\n";
    }
    return $output;
?>
```

See also:
还可参看：

http://www.ubercart.org/issue/2271/uc_file_description_text_filedownload...

相关链接： http://ubercart.org http://zhupou.cn

theme_uc_product_add_to_cart

函数theme_uc_product_add_to_cart()位于uc_product.module模块中：

```
<?php
   theme_uc_product_add_to_cart($node)
?>
```

描述：

将添加到购物车表单封装在了一个 标签中，该标签的类为"add_to_cart".

参数：

$node -当添加到购物车表单提交时，向购物车中添加的产品节点对象。

返回值：

一个包含了添加到购物车表单的HTML。

示例：

```
<?php
  $node = node_load($nid);
  $output .= theme('uc_product_add_to_cart', $node);
```

theme_uc_product_display_price

函数theme_uc_product_display_price()位于uc_product.module模块中：

```
<?php
   theme_uc_product_display_price($price)
?>
```

描述：

把价格包装在一个类为"display_price"的 中。

参数：

$price -要显示的数值

返回值：

一个包含了格式化价格的HTML字符串。

示例：

```
<?php
  $output .= theme('uc_product_display_price', $node->sell_price);
?>
```

同一版本不同稳定版之间的更新

在开发稳定版的同时，我们会尽力把相应的API的更新放到这里。在这些页面所列的修改，就意味着这一变动已经保存在了Bazaar资源库中了，同时也将被以用于下一个稳定版本中去。

- 从Ubercart 1.3到1.4的API变动
- 从Ubercart 1.5到1.6的API变动

相关链接： http://zhupou.cn
http://www.ubercart.org/docs/developer/6864/minor_release_updates

hook_download_authorize

函数hook_download_authorize()位于uc_file.module模块中：

```
<?php
  hook_download_authorize($user, $file_download)
?>
```

描述:

uc_file模块默认实现了3个下载方面的限制：下载的IP地址数量，下载的次数，以及过期日期。如果开发者还想增加更多的限制的话，就可以通过这个钩子函数来实现。当前面提到的3个限制被选中以后，uc_file模块将检查这个钩子的实现。

参数:

- $user -请求下载的Drupal用户对象
- $file_download - 定义的文件下载对象，它对应于uc_file_users表中的一行记录，该表用于记录哪些用户可以下载哪些资源。

返回值：

如果用户可以下载所请求的文件的话，就返回TRUE，否则返回FALSE。如果返回FALSE的话，你需要在drupal中设置一个错误消息（使用drupal_set_message(t($message_text),'error')），来通知客户问题出在了哪里。

例如：

```
<?php
function module_name_download_authorize($user, $file_download) {
  if (!$user->status) {
    drupal_set_message(t("This account has been banned and can't download files anymore. "),'error');
    return FALSE;
  }
  else {
    return TRUE;
  }
}
?>
```

hook_payment_method

函数hook_payment_method()位于uc_payment.module模块中:

```
<?php
  hook_payment_method()
?>
```

描述:

支付方法收钱的不同方式.默认的, Übercart支持支票、信用卡、和普通付款方式。支付方法显示在结算页面或者订单管理页面，它们从用户那里收集到不同的信息，用来处理和追踪支付。

返回值:

支付方法数组.

示例:

```
<?php
function uc_payment_payment_method() {
  $methods[] = array(
    'id' => 'check',
    'name' => t('Check'),
    'title' => t('Check or Money Order'),
    'desc' => t('Pay by mailing a check or money order.'),
    'callback' => 'uc_payment_method_check',
    'weight' => 1,
    'checkout' => TRUE,
  );
  return $methods;
}
?>
```