WebGUI Shop Guide 7.7

by Kristi McCombs, JT Smith

Printed in the United States of America by Lulu.com.

Published by Plain Black Corporation, 1360 Regent St, #145, Madison, WI, 53715

Editors:	Kristi McCombs JT Smith Doug Bell
Cover Design:	Darci Gibson
Printing History:	October 2009: Second Edition WebGUI Version 7.7

Preface

The *WebGUI Shop Guide* walks you through setting up Shop on your WebGUI site from the points of view of both the content manager and developer. Setting up Shop through the Admin Console, creating Shop related assets by which to display products for sale, and customizing the inner-workings of Shop are covered in these pages to help you create a shop you can call your own. This version of *WebGUI Shop Guide* is intended for WebGUI users in version 7.7 and higher.

WebGUI Shop allows users to sell online merchandise, sell subscription services, and sell access to and manage events. New assets have been created to provide content managers increased flexibility in how products are displayed and managed. Shopkeepers can give shoppers in-store credit and provide coupons. Shop includes the assets Shelf, Product, Flat Discount Coupon, Subscription,Donation, ThingyRecord. and a full Event Management System.

WebGUI Shop's API provides developers the ability to customize and add functionality to Shop with the same ease and flexibility to which WebGUI developers are accustomed. In these pages you'll be introduced to a glimpse of what is possible through writing custom SKUs (an asset subclass) to provide users a dynamic shopping experience.

Table of Contents

Welcome to Shop

In WeGUI 7.5 the old WebGUI commerce system was torn down to the studs and completely built back up again. User documentation for the new system, WebGUI Shop, is now housed in its own guide, *WebGUI Shop Guide*. *WebGUI Shop Guide* is an all-encompassing guide to setting up Shop on your WebGUI site.

Administrative functions in Shop include:

- setting up payment gateways
- managing transactions
- managing tax rates
- setting up shipping methods
- assigning in-Shop credit
- managing vendors

Many of the features in WebGUI's old commerce system have been recreated as assets in Shop. These include:

- Shelf: a Shop-specific container asset that allows you to create shelves within your shop on which to place products for sale, just like a physical shelf in a traditional store. In addition, products and shelves can be assigned metadata keywords, which allows you to populate your shelves with products based on keyword associations.

Product, Donation, Subscription, Flat Discount Coupon, Ad Sales, Thingy Record and the EMS Ticket, Badge, Ribbon and Token are all subclasses of WebGUI::Asset::Sku

- Product: this asset is used to create a product for sale. Once created, variants of a product can be created to sell different versions, such as different sizes and colors, of a specific product. The Product asset will keep track of what is in stock, and when a variant is out of stock, it will no longer be available for sale. Specific shipping methods and tax overrides can be applied to individual Product assets. Product images and attachments may be included with the asset, and products can be related to each other as a means of directing shoppers to other products of interest.

- Subscriptions: subscriptions may be purchased and users assigned to the appropriate subscription group upon that purchase. Batches of subscription codes can be generated so users can access subscription content without making a purchase, helpful for organizations purchasing a single subscription for multiple users.

- Flat Discount Coupon: places a coupon on the site that users add to their carts. Shop recognizes that a user has placed a coupon in the cart, and will not allow more than one to be applied to the transaction. The coupon can be configured to take a flat monetary amount or a percentage of the total cost off the final price. It may also be set up to require a total purchase amount in order to activate.

- Donation: the Donation asset allows users to make a charitable donation through the site.

- Ad Sales: allow you to sell advertising space on your website.

- Thingy Record: allows you to sell a record in a Thing within a Thingy asset. This is useful for setting up environments like a classifieds area in which people can purchase an advertisement to display in the classifieds.

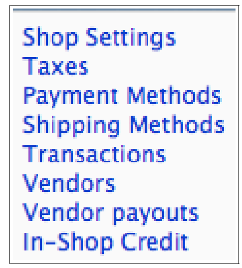
- Event Manager: a full event management system from which event access can be sold and managed. Users purchase an event badge that corresponds to a specific event to attend. Within that event tickets to individual presentations or performances can be purchased, and users can purchase tokens to “spend” like currency. Ribbons can also be purchased to provide discounts on related items. Event managers can access individual attendee registration information, manage individual events, and check people in on the day of the event.

In addition to all this, the Shopping Cart has been redesigned to be more user-friendly. Users enter shipping and billing addresses, and Shop will store those addresses for the user to select from again. Items are easily removed or increased in number, and totals updated. A number of macros are also available to customize how users access their carts, including a mini cart view that allows users to see how many items are in their carts and the total purchase price while they shop.

Chapters near the end of this book are devoted to developers, and provide examples of how to take advantage of WebGUI Shop's API to enhance the shopping experience for your users.

Set Up Shop

You must set up Shop in order to begin using e-commerce on your WebGUI site. This chapter discusses all of the administrative functions available through the Shop interface. Shop is accessed through the Admin Console. Either click on the Shop icon in the Admin Console located down the left hand side of the screen in the inline editing view, or in the Admin Console accessed through the Asset Manager editing view.



Upon entering Shop you will see the initial Shop Settings screen. On the far right hand side of the Shop Settings screen are a number of blue links. Use these links to navigate between the different screens used to configure commerce on the site.

Shop Settings

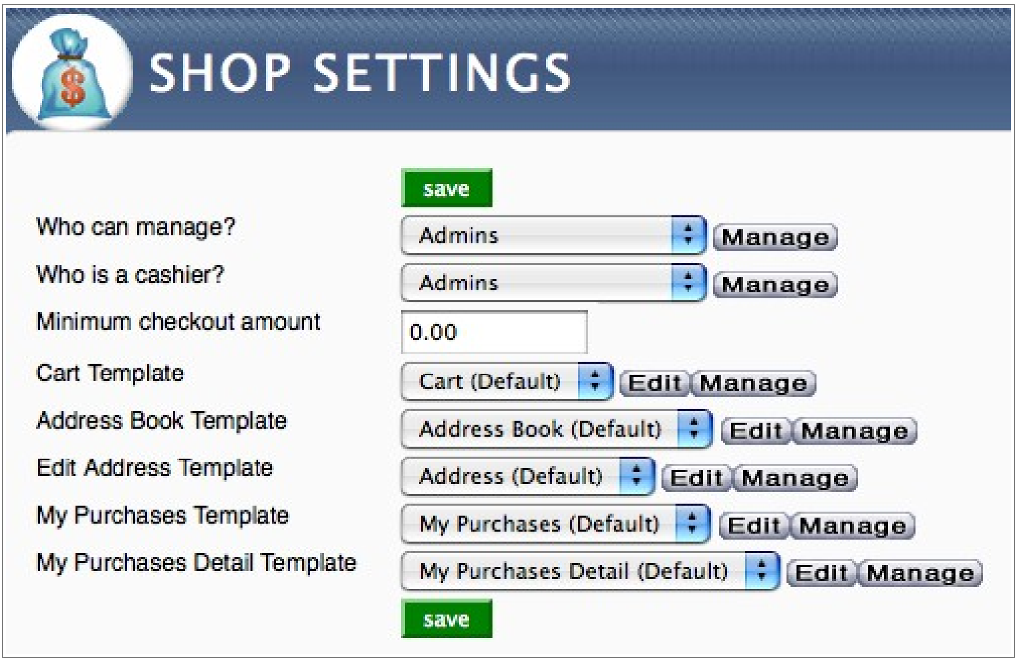
The Shop Settings screen contains is the default screen upon entering WebGUI Shop.

Who can manage?: controls the group of users allowed to manage commerce settings on the site.

Who is a cashier?: allows you to select a group of users who are allowed to make purchases on behalf of another site user by typing that user's email address into the cart.

Minimum checkout amount: allows you to set a minimum purchase amount that must be made in order to check out and complete a transaction.

The remaining fields are related to templates.



Next to each template is an Edit and Manage button. Clicking the Edit button will open the template, where you can edit the template itself. Clicking on the Manage button will direct you to the asset manager view of the template asset.

Cart Template: this template controls the appearance of the shopping cart.

Address Book Template: this template controls the appearance of the address book. Addresses are entered for shipping and billing information during the checkout process.

Edit Address Template: determines the appearance of the screen used for editing an address book entry. Addresses are entered for shipping and billing information during the checkout process.

My Purchases Template: determines the appearance of a user's order history, the screen in which all transactions are listed. Users can access this history through their Update Account Information screen and clicking on My Purchases. This is the template used to display that screen.

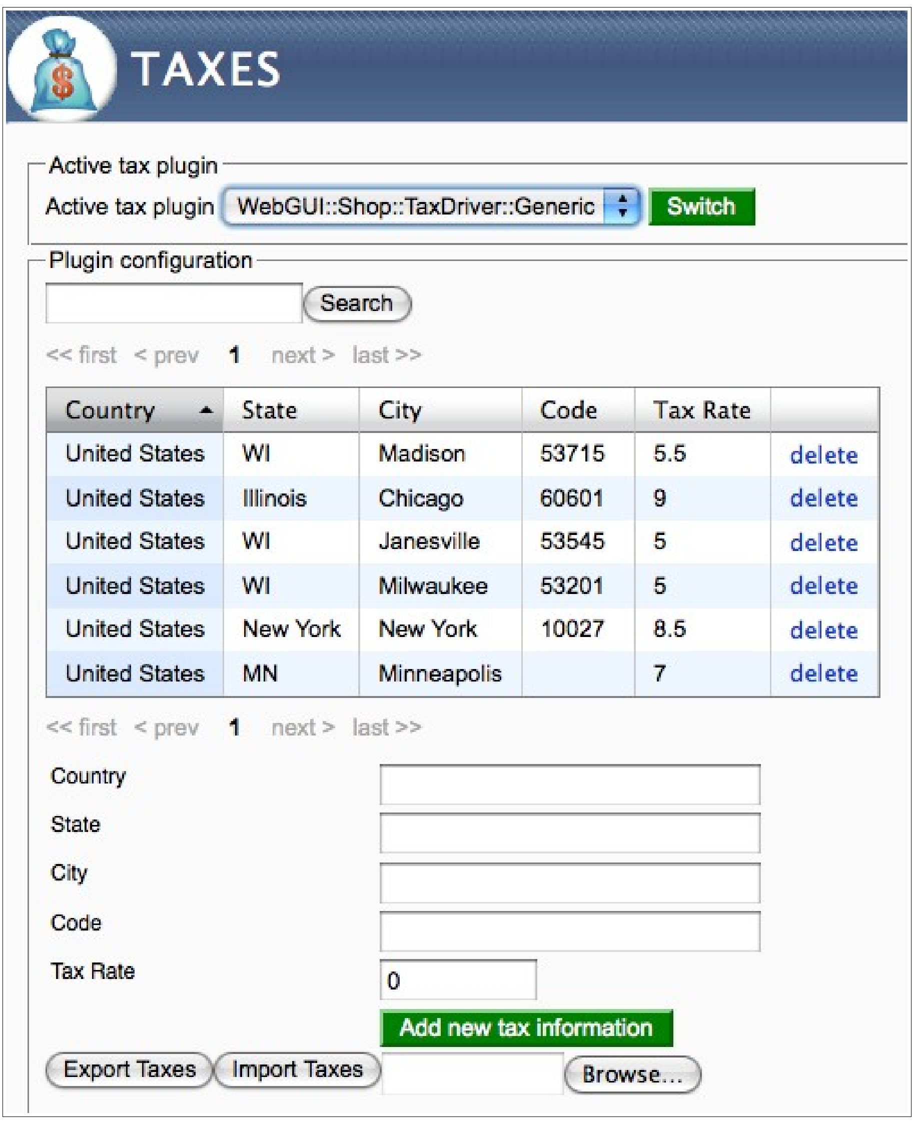
My Purchases Detail Template: in a user's order history, or My Purchases screen, a user may select to view an individual transaction by clicking on its order number. This controls the appearance of the order details they see.

Taxes

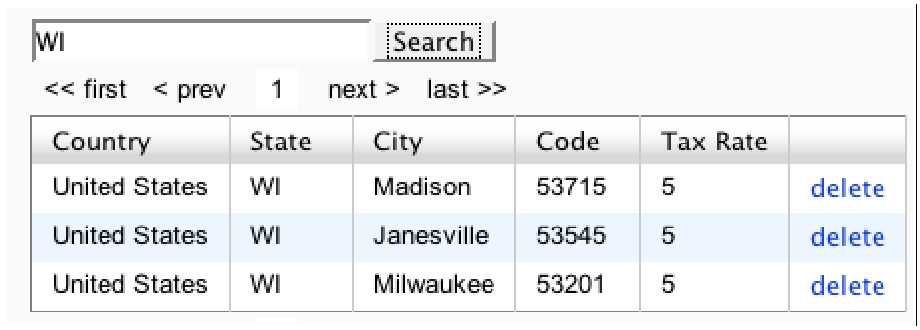
The Taxes screen allows you to input information related to taxes in a specific location. You have the option of manually entering tax information for a given locale, or importing a tax table to load a lot of tax data at once. In addition, the tax table you create in WebGUI Shop can be exported as a .csv file for use in an external application.

At the top of the Taxes screen is an “Active tax plugin” field. Shop's tax system is pluggable, which allows you to build your own tax rules. By default, a Generic and European Union tax driver are available.

Generic Tax Driver



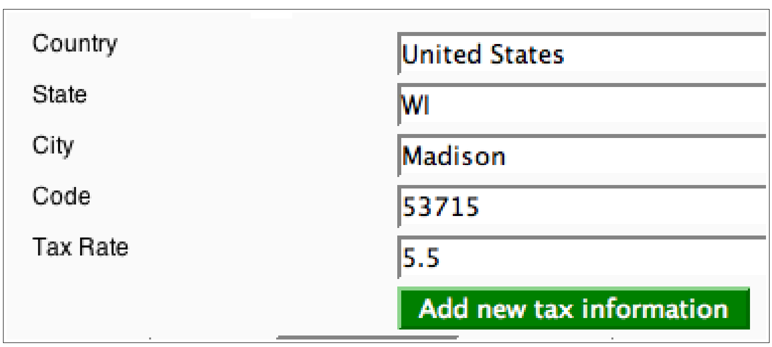
The first field in the Tax screen is Search. The Search field allows you to search this screen for specific tax information in the event that this screen has become heavily populated. For example, if you enter WI in the Search field and click the Search button, tax information for only those entries with WI entered for the state would be returned, as seen below.


WI Search

<< first < prev 1 next > last >>

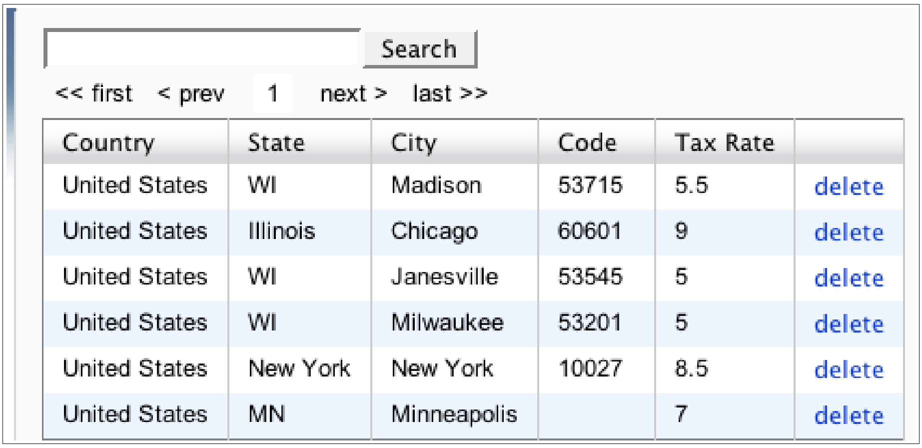
Country	State	City	Code	Tax Rate	
United States	WI	Madison	53715	5	delete
United States	WI	Janesville	53545	5	delete
United States	WI	Milwaukee	53201	5	delete

The form located towards the bottom of the screen is used to input new tax information. This form allows you to enter a specific location and its applicable tax rate.



In the first four fields, enter the Country, State, City and (Zip/Postal)Code in which this tax rate applies. In the “Tax Rate” field, enter the applicable tax rate for this location. You can enter a tax rate for an entire state by leaving the City and Code field blank. Only fill in the Country, State, and Tax Rate fields.

Upon clicking the green “Add new tax information” button the data will be loaded, and will appear at the top of the Taxes screen. Each entry will have a delete link to its right by which to remove the entry.


Search

<< first < prev 1 next > last >>

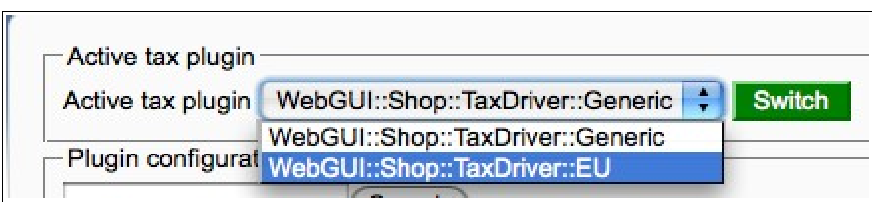
Country	State	City	Code	Tax Rate	
United States	WI	Madison	53715	5.5	delete
United States	Illinois	Chicago	60601	9	delete
United States	WI	Janesville	53545	5	delete
United States	WI	Milwaukee	53201	5	delete
United States	New York	New York	10027	8.5	delete
United States	MN	Minneapolis		7	delete

By default, entries are listed in the order in which they are added. To rearrange entries in a descending/ascending order, for example to list them alphabetically by state, click on the headings of each column.

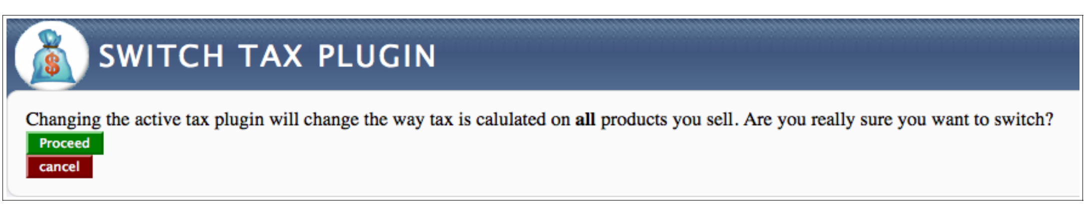
Near the bottom of the tax screen is an import and export field. This allows you to import tax information from an external database, as well as export the data you have input into the Taxes screen, as a .csv file.

Switch Tax Drivers

To switch to a different tax driver, select the “Active tax plugin” you want to switch to from the dropdown menu.



Then, click the Switch button. A confirmation screen will appear.

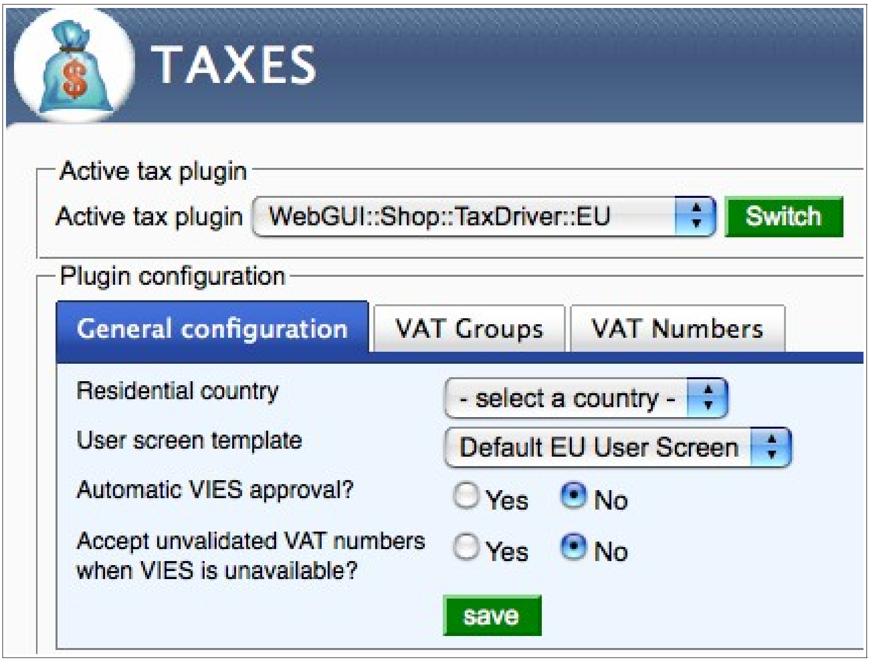


Confirm the change by clicking on the Proceed button.

European Union Tax Driver

The European Tax Driver contains three tabs.

General Configuration



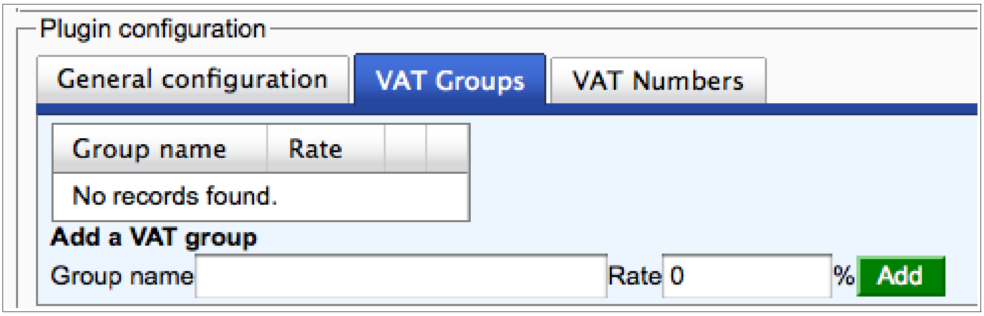
Residential Country: a dropdown menu in which you select the country where you shop resides. If your country does not appear, it is not a member of the European Union, and you should not use this tax driver.

User screen template: the template for the user screen in which users enter their VAT numbers.

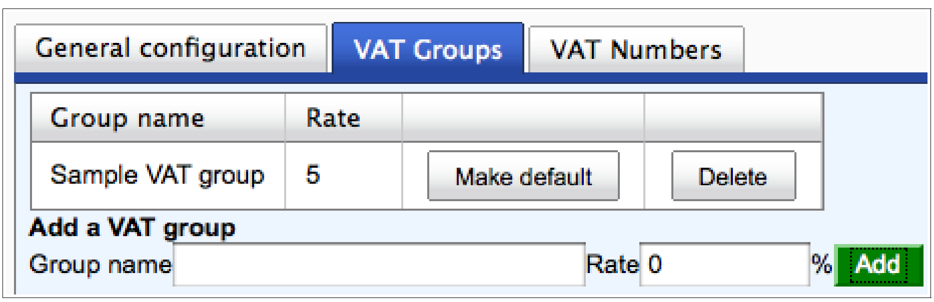
Automatic VIES approval?: if set to Yes, VAT numbers that are validated through the VIES service are directly usable by your customers. If set to No, only VAT numbers that have been explicitly approved by you can be used.

Accept non-validated VAT numbers when VIES is unavailable?: if one of the VIES member states' databases is temporarily unavailable, or the connection to VIES fails, VAT numbers cannot be checked through this service. If you set this field to Yes, VAT numbers will still be able to be used. The format of the VAT number is always checked, regardless of the availability of VIES.

VAT Groups



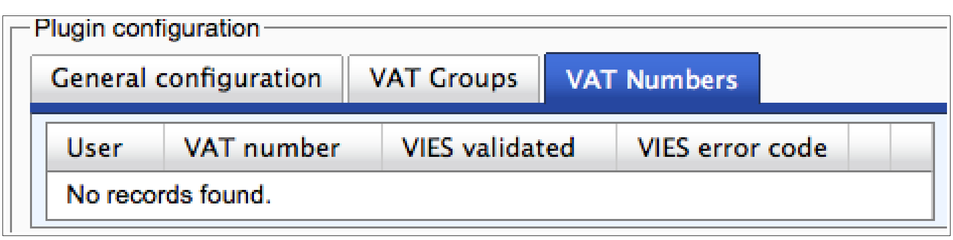
Enter the VAT group name in the "Group name" field, and the tax rate in the "Rate" field. Then click the Add button.



Once added, it will appear in the "Group Name" area. From this point you can select an added VAT group to be the default by clicking on the "Make default" button, or you can click on a group's "Delete" button to delete it.

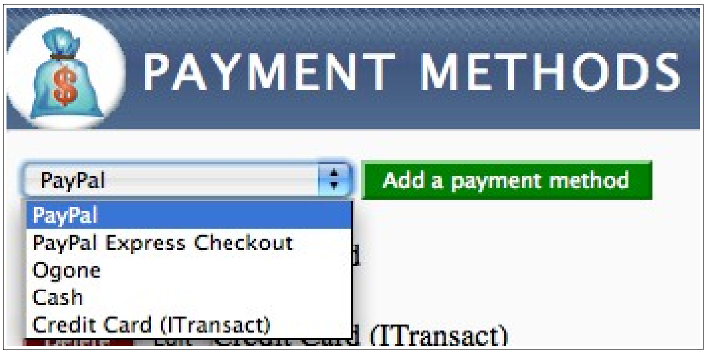
VAT Numbers

This tab contains information regarding individual user's VAT numbers and VIES validation.



Payment Methods

The Payment Methods screen is where you set up and enable acceptable payment methods for your shop.

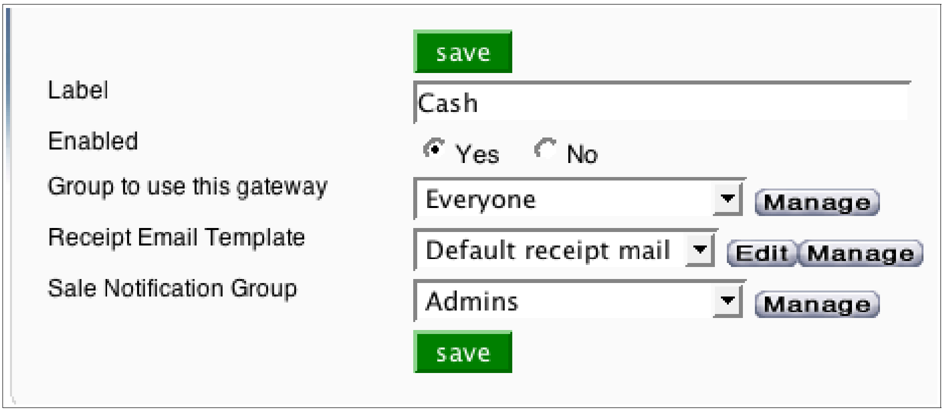


By default, WebGUI Shop contains payment methods for Cash, Credit Card (iTransact), Paypal and Ogone. To add a new payment method, set the drop down menu to Cash, Credit Card (iTransact©), Paypal or Ogone and click on the green "add payment method" button. To view an existing payment method, click the edit button to the left of each payment method. This will open the Payment Methods edit screen in which you enter information related to this method.

It is important to note that not all payment methods are capable of handling recurring transactions, such as in the case of a Subscription. For example, iTransact© can, but Cash can not. This will remain true as more payment gateways are developed for Shop.

Cash

The screen below shows the Cash payment method.



Label: this is the name for this payment method as it is displayed on the screen to the user. If you want a the shopper to see something other than "Cash," enter a different label here.

Enable: in this example, you can see that the Cash payment method is enabled because the Enabled field is set to Yes. In order for this payment method to be made available to shoppers, you need to enable it.

Group to use this gateway: the "Group to use this gateway" field indicates the group of users who will have this payment method available to them during checkout.

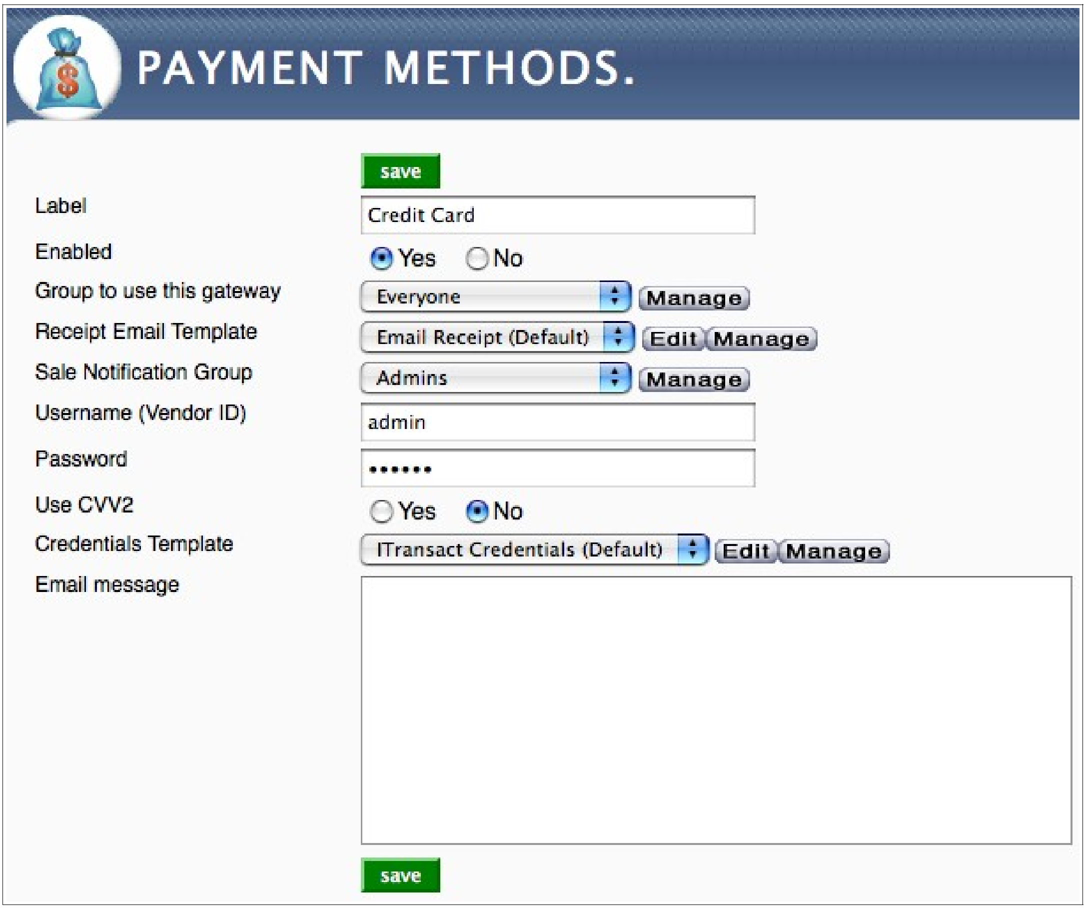
Receipt Email Template: the template used to generate an email that is sent to a user upon completing a purchase. This email confirms a purchase and acts as the user's receipt.

Sale Notification Group: the group of users who will be notified when a sale is made on the website. WebGUI will send an email to their WebGUI Inboxes (accessed through the Admin Console) and/or to an email address associated with their user profiles. This email will alert the users of this group that a sale has been made and provide a transaction number for reference in the Shop Transaction screen.

When all fields in this screen are complete, click the green save button to save the settings.

Credit Card

The default credit card payment gateway used by WebGUI Shop is iTransact©. You are not required to use this payment gateway, but it is provided for your convenience and the examples here refer to it. The add/edit payment method for credit cards contains a number of instructions. Simply follow the steps in order to gather the information necessary for completing this screen.



Label: the label is the name of the payment method as it will be displayed on the screen to the user.

Enable: the payment method must be enabled, by setting this field to Yes, in order to be used on the site.

Group to use this gateway: this is the group of users to which this payment method will be made available during checkout.

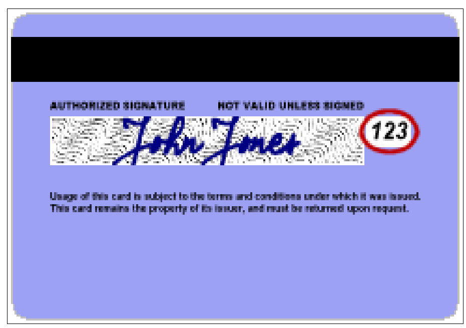
Receipt Email Template: the template used to generate an email that is sent to a user upon completing a purchase. This email confirms a purchase and acts as the user's receipt.

Sale Notification Group: the group of users who will be notified when a sale is made on the website. WebGUI will send an email to their WebGUI Inbox (accessed through the Admin Console) and/or to an email address associated with their user profile. This email will alert the users of this group that a sale has been made and provide a transaction number for reference in the Shop Transaction screen.

Username (Vendor ID): when you have acquired a merchant account, which is discussed later in this section, you will establish a vendor ID to enter in this field.

Password: when your merchant account is created you will establish a password to enter in this field.

Use CVV2: the cvv2 (Card Verification Value) is a 3 or 4 digit numerical code that is printed on the credit card or signature strip on the back of a credit card. Supplying this code at the time of an online purchase helps ensure that the customer has the physical credit card in his/her possession, and is a security measure to help reduce credit card fraud. Indicate if you would like to require this code at the time of purchase.



Credentials Template: the template used to display the form in which users will enter their billing and credit card information.

Email Message: enter a message in this field that will be emailed to users after they have completed a credit card purchase on your site.

At the bottom of the screen are the steps to complete in order to begin using the iTransact© credit card payment gateway in WebGUI Shop.

Setting up your ecommerce site is as easy as these few steps:

Step 1: Get A Merchant Account
Register for a merchant account now to get started processing online transactions.

Step 2: Set Up Your Merchant Account Info
See the information toward the bottom of this page to set up your merchant account info.

Step 3: Get An SSL Certificate
Get an SSL Certificate from CompleteSSL.

Step 4: Install The Certificate
Contact your hosting provider to install your certificate or install it yourself.

Step 5: Enable IP Address
For added security the system will not allow just anyone to post requests to the merchant account. We have to tell the merchant account what the IP address of our site (or sites) is. To do this go to your virtual terminal and log in. Go to Account Settings > Fraud Control > and click on the "IP Filter Settings" link. There enter the IP address of your server Set the status to Active and set the module to XML, then hit go. Contact your system administrator for your server IP address. You'll also need to submit a support ticket to let iTransact know that you wish to enable the XML API.

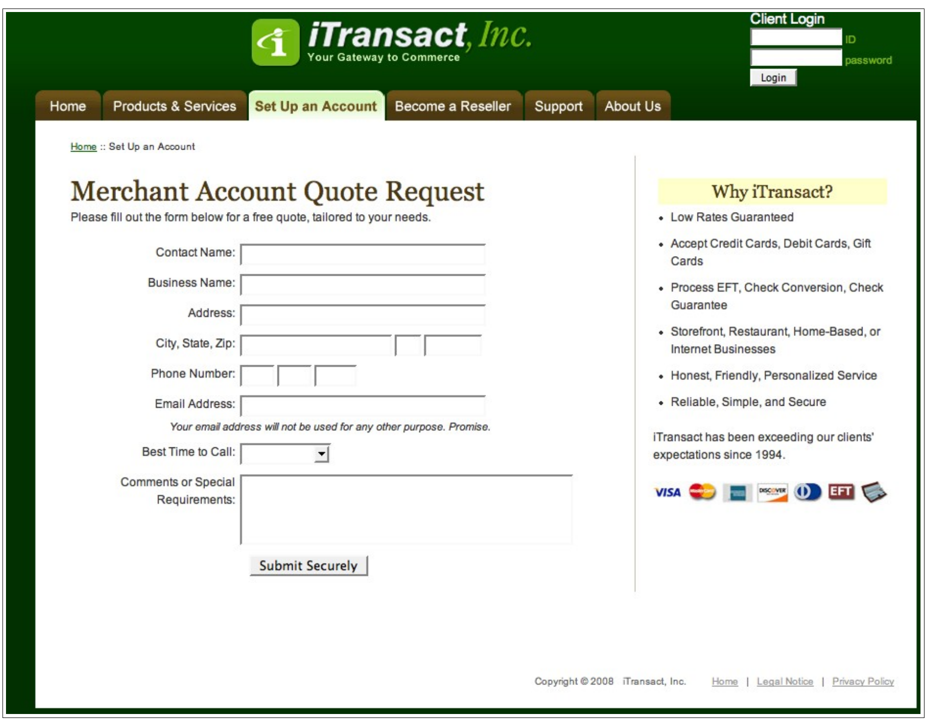
Step 6: Enable The Commerce System
Set the enabled field to "Yes" in your WebGUI commerce settings.

Step 7: Optionally Accept American Express, Discover, and Diners Club
By default you'll only be able to accept MasterCard and Visa. If you want to accept others you'll need to follow these steps:

1. Call the credit card vendor to apply:
   - American Express: (800) 528-5200
   - Discover: (800) 347-2000
   - Diners Club: (800) 525-7376
2. Submit the account numbers that you get from those companies in a support ticket. to get them registered with your merchant account.
3. Go to your virtual terminal and enable these cards under your Account settings.

Step 1: Get a Merchant Account

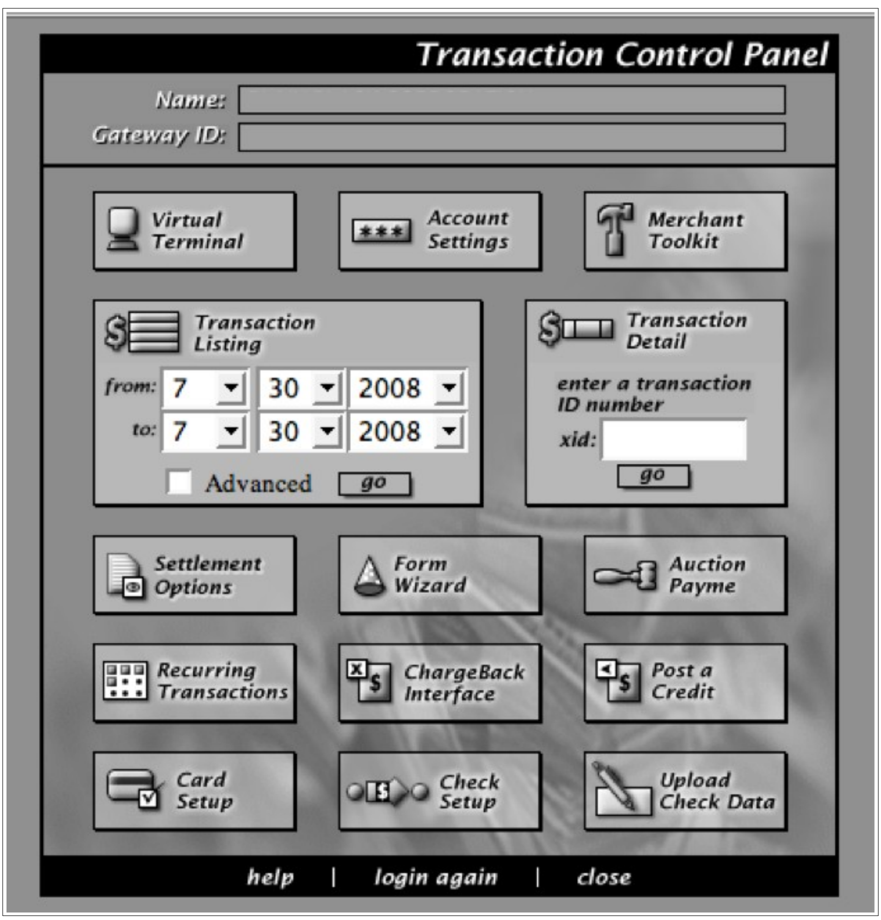
Below this step a link is provided to direct you to the iTransact© website. You will be provided a form to complete to begin the process of setting up your merchant account.



Once your merchant account is established, you will be provided a vendor ID and password to enter in the Payment Methods screen, as well as to use on the iTransact© site.

Step 2: Set Up Your Merchant Account Information

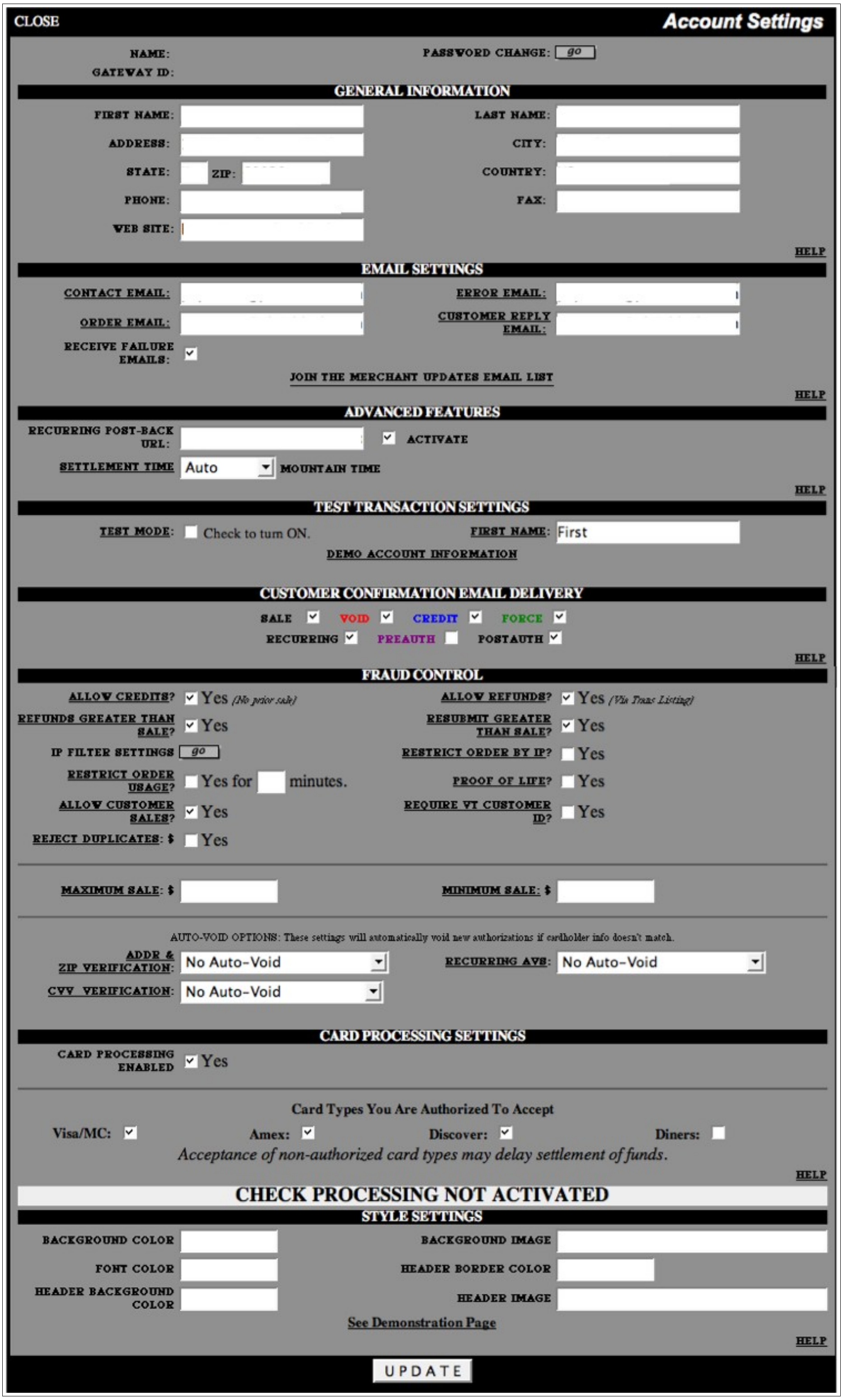
Once you have established a Merchant Account with iTransact©, you will be able to log in to the iTransact site and access a Transaction Control panel to set up your account details. iTransact has a great deal of built-in documentation to help you along the way. Many of the fields in the system will open a pop-up window displaying help documentation when clicked upon, and some screens will have a help link right on them. The Merchant Toolkit icon opens a page with a great deal of helpful information, including links to iTransact documentation.



At the bottom of the control panel is a “help” link. Clicking on the help link will open a screen that defines what each icon in the Transaction Control Panel opens.

Account Settings

The Account Settings screen is where you enter information specific to your account. Each of the fields on this screen has help documentation available. Simply click on a field name and a box will pop up providing you with more information about what needs to be entered. In addition, there are Help links located on the right hand side of the screen for each major section.


CLOSE
Account Settings
NAME:
PASSWORD CHANGE: go
GATEWAY ID:
GENERAL INFORMATION
FIRST NAME:
LAST NAME:
ADDRESS:
CITY:
STATE:
ZIP:
COUNTRY:
PHONE:
FAX:
WEB SITE:
HELP
EMAIL SETTINGS
CONTACT EMAIL:
ERROR EMAIL:
ORDER EMAIL:
CUSTOMER REPLY EMAIL:
RECEIVE FAILURE EMAILS:
JOIN THE MERCHANT UPDATES EMAIL LIST
HELP
ADVANCED FEATURES
RECURRING POST-BACK URL:
ACTIVATE
SETTLEMENT TIME
Auto
MOUNTAIN TIME
HELP
TEST TRANSACTION SETTINGS
TEST MODE:
Check to turn ON.
FIRST NAME:
First
DEMO ACCOUNT INFORMATION
CUSTOMER CONFIRMATION EMAIL DELIVERY
SALE
VOID
CREDIT
FORCE
RECURRING
PREAUTH
POSTAUTH
HELP
FRAUD CONTROL
ALLOW CREDITS? Yes (No prior sale)
ALLOW REFUNDS? Yes (Via Trans Listing)
REFUNDS GREATER THAN SALE? Yes
RESUBMIT GREATER THAN SALE? Yes
IP FILTER SETTINGS go
RESTRICT ORDER BY IP? Yes
RESTRICT ORDER USAGE? Yes for minutes.
PROOF OF LIFE? Yes
ALLOW CUSTOMER SALES? Yes
REQUIRE VT CUSTOMER ID? Yes
REJECT DUPLICATES: $ Yes
MAXIMUM SALE: $
MINIMUM SALE: $
AUTO-VOID OPTIONS: These settings will automatically void new authorizations if cardholder info doesn't match.
ADDR & ZIP VERIFICATION:
No Auto-Void
RECURRING AVS:
No Auto-Void
CVV VERIFICATION:
No Auto-Void
CARD PROCESSING SETTINGS
CARD PROCESSING ENABLED
Yes
Card Types You Are Authorized To Accept
Visa/MC:
Amex:
Discover:
Diners:
Acceptance of non-authorized card types may delay settlement of funds.
HELP
CHECK PROCESSING NOT ACTIVATED
STYLE SETTINGS
BACKGROUND COLOR
BACKGROUND IMAGE
FONT COLOR
HEADER BORDER COLOR
HEADER BACKGROUND COLOR
HEADER IMAGE
See Demonstration Page
HELP
UPDATE

Near the bottom of the WebGUI Shop Payment Methods screen for iTransact are further instructions for setting up your merchant account. This provides helpful recipes for setting up recurring transactions on your site. You will want this to correspond to what is entered in the Recurring Transactions screen in iTransact, accessed through the icon of that name.

This plugin expects that you set up the following recipe's in your iTransact account. Be very careful to enter the recipe names exactly as given below.

weekly -> 7 days
biweekly -> 14 days
fourweekly -> 28 days
monthly -> 30 days
quarterly -> 91 days
halfyearly -> 182 days
yearly -> 365 days

Please note that some of these recipe's are only roughly correct. They don't 'fit' exactly in a whole year. Below the affected recipe's are given together with their difference on a year's basis.

- **monthly** (differs 5 days each year, 6 days each leap year)
- **quarterly** (differs 1 day each year, 2 days each leap year)
- **halfyearly** (differs 1 day each year, 2 days each leap year)
- **yearly** (differs 1 day each leap year)

Also set the 'RECURRING POST-BACK URL' field in the Account Settings part of the virtual terminal to:
https://betatraining.plainblack.com/?shop=pay;method=do;do=processRecurringTransactionPostback;paymentGatewayId=WIJ5WuhvEvZr5mKyvPeqcQ

Step 3: Get An SSL Certificate

This step includes a link to CompleteSSL©. This is a company that Plain Black has established a partnership with in order to pass along some savings to you. Upon clicking on this link you will be taken directly to a page that lists a number of options for Plain Black customers. On this screen you will be able to see the discounted prices available, and will be provided a link from which to make your purchase. You are not required to purchase an SSL certificate through this provider, but the option is provided for your convenience.

Step 4: Install the Certificate

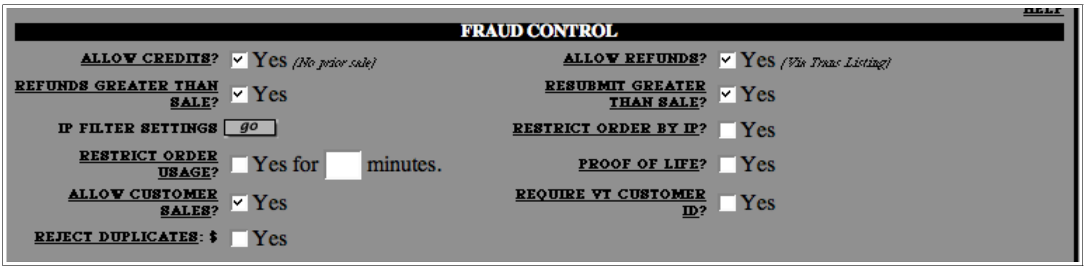
Once you have purchased an SSL certificate you need to have it installed. CompleteSSL© offers this service for free upon request, or you can contact your hosting provider.

Step 5: Enable IP Address

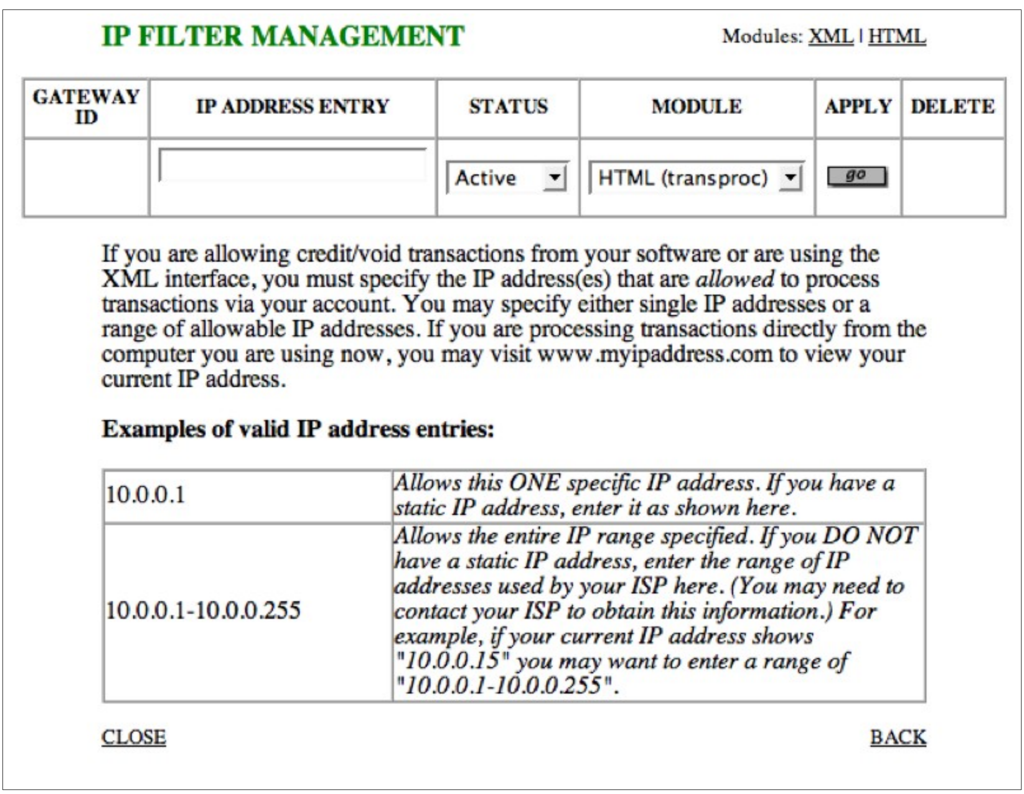
There are some specific instructions regarding how to do this included on the screen.

Step 5: Enable IP Address
For added security the system will not allow just anyone to post requests to the merchant account. We have to tell the merchant account what the IP address of our site (or sites) is. To do this go to your virtual terminal and log in. Go to Account Settings > Fraud Control > and click on the "IP Filter Settings" link. There enter the IP address of your server Set the status to Active and set the module to XML, then hit go. Contact your system administrator for your server IP address. You'll also need to submit a support ticket to let iTransact know that you wish to enable the XML API.

Basically, you need to log in to iTransact© with the ID and password established when you set up your merchant account. Once you login, you need to access the Transaction Control Panel, and in that control panel you need to click on the Account Settings icon. Scroll down to the Fraud Control field and click the go button next to the IP Filter Settings field.



In the IP Filter Management screen, you need to add the IP Address from which you will be working.



You need to set its status to Active and set the module to XML. You will need to contact iTransact© to let them know you want to enable the XML

API. The Payment Methods screen contains a link in the text of the instructions to do this.

Step 6: Enable the Commerce System

Double check that the Enable field near the top of this screen is set to Yes.

Step 7: Optionally Accept American Express, Discover, and Diners Club

These credit card types are not accepted by default. Step 7 includes some instructions to enable these in Shop. These instructions include the telephone numbers of each credit card, a link from which to submit the account numbers received from these companies, and instructions to enable these credit card payment types through the iTransact Transaction Control Panel.

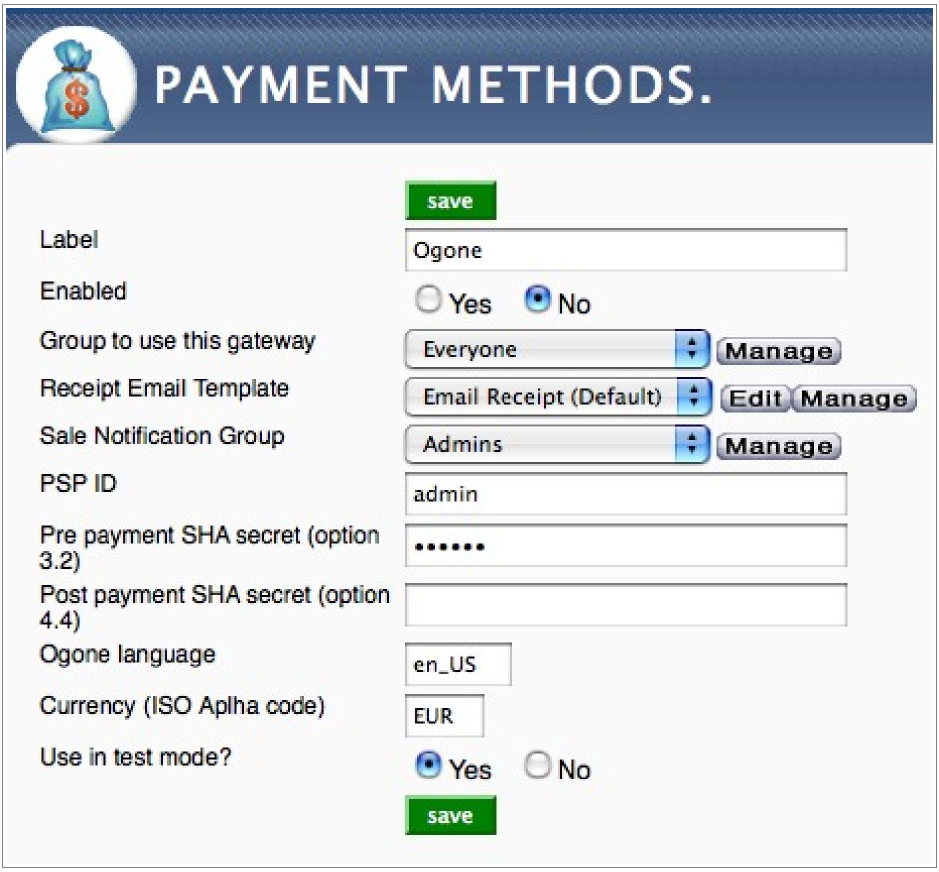
When the Payment Methods screen for setting up credit card transactions is complete, click the green save button. Once everything is set up, you will be provided a green button on the iTransact payment method screen in WebGUI Shop that will take you directly to the iTransact site where you an access the Transaction Control Panel.

Ogone

Ogone is used to support local credit card processing in Europe. There are some additional instructions at the bottom of this screen to assist you in setting up an Ogone account.

The first field is the “Label” field, in which you will see Ogone. You can edit this field to display something different in the Payment Methods screen.

In order to use the Ogone payment method, you must set the “Enabled” field to Yes.



Group to use this gateway: select a group of users who are allowed to use this payment gateway.

Receipt Email Template: the template used to display the email that is sent to users to confirm their purchase.

Sale Notification Group: this group of users will receive a notification via email and/or their WebGUI Inbox informing them when a sale takes place.

PSP ID: your Ogone username.

Pre payment SHA secret (option 4.4): the passphrase you set in section 3.2 of the Technical Information page of the Ogone interface.

Post payment SHA secret (option 4.4): the passphrase you set in section 4.4 of the Technical Information page of the Ogone interface.

Ogone language: the locale string for the language that should be used to display the Ogone interface to the user.

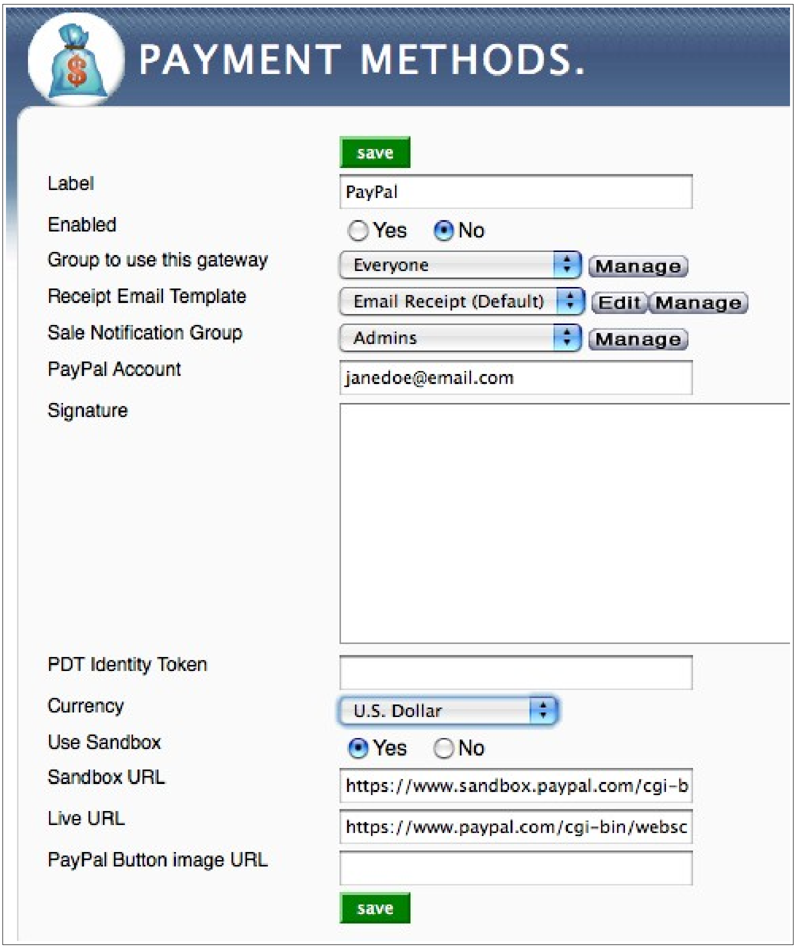
Currency (ISO Alpha code): the currency in which the payment is to be made. You can see some common examples in the hover help.

Use in test mode?: if set to Yes, all payment requests will be directed to Ogone's test environment. This allows you to make sure everything is correct before going live. No actual payments will be completed while this is set to Yes, so make sure you set it to No when you have completed testing.

PayPal and PayPal Express

Before setting up the PayPal© payment method, you will need to set up a Business Account through PayPal's website. You can do so by going to the PayPal site and entering the Business tab.

Once you have set up your business account through PayPal, select PayPal in Shop's Payment Methods screen.



Label: enter the label that you would like to appear for this payment method on your site.

Enabled: the payment method must be enabled in order for it to be used on your site.

Group to use this gateway: select a group of users who are allowed to use this payment gateway.

Receipt Email Template: the template used to display the email that is sent to users to confirm their purchase.

Sale Notification Group: this group of users will receive a notification via email and/or their WebGUI Inbox informing them when a sale takes place.

PayPal Account: enter the email address associated with your business PayPal account.

Signature: the account signature for your PayPal account.

PDT Identity Token: the identity token listed under the Payment Data Transfer radio button in your website payment preference on PayPal.

Currency: from the dropdown menu, select the currency used to complete transactions in your PayPal account.

Use Sandbox: set to yes if you want to use the PayPal development (not production) environment. This is recommended for testing. When you are ready to use this payment method in a production environment, remember to return to this screen and switch this field to no.

Sandbox URL: if you are using a sandbox, this is the URL to post to.

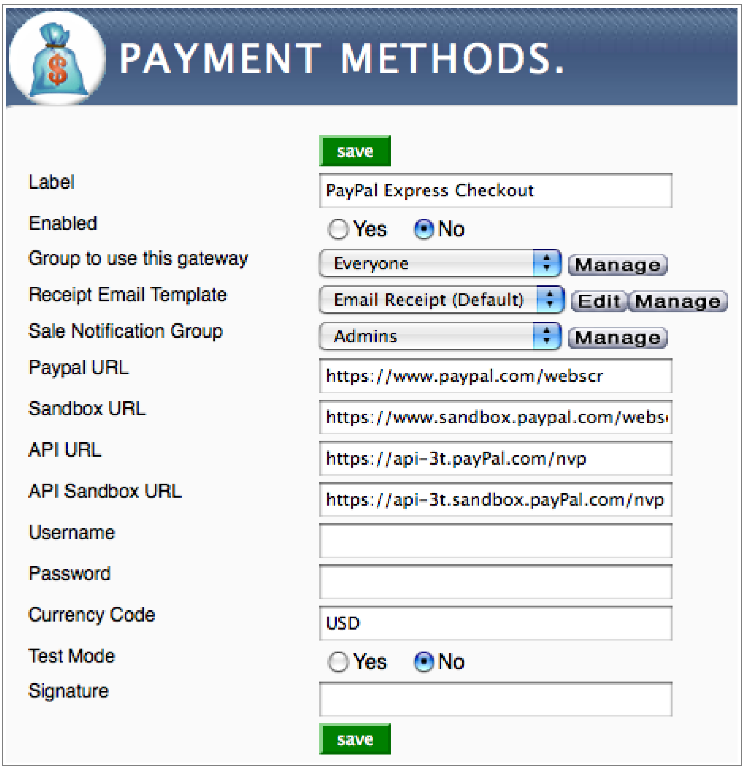
Live URL: when you are not using the sandbox, this is the live URL to post to.

PayPal Button image URL: enter the URL of the PayPal image you would like to use for this checkout option.

PayPal Express

You can also set up the PayPal Express payment method. PayPal Express streamlines the checkout process and keeps the site user on the merchant's website after making a purchase.

Select PayPal Express Checkout in the Payment Methods screen to configure this method.



Label: enter the label that you would like to appear for this payment method on your site.

Enabled: the payment method must be enabled in order for it to be used on your site.

Group to use this gateway: select a group of users who are allowed to use this payment gateway.

Receipt Email Template: the template used to display the email that is sent to users to confirm their purchase.

Sale Notification Group: this group of users will receive a notification via email and/or their WebGUI Inbox informing them when a sale takes place.

Paypal URL: URL to use when redirecting site users to PayPal.

Sandbox URL: the URL used to redirect while in test mode.

API URL: the base URL for PayPal's NVP API.

API Sandbox URL: URL for PayPal API in test mode.

Username: the username associated with the PayPal account being used for this payment method.

Password: the password associated with the PayPal account being used for this payment method.

Currency Code: the PayPal currency code to use for this payment method.

Test Mode: if set to Yes, the site will use PayPal's sandbox. When you move to production on the live site, make sure you switch this field back to no.

Signature: signature for PayPal credentials.

Shipping Methods

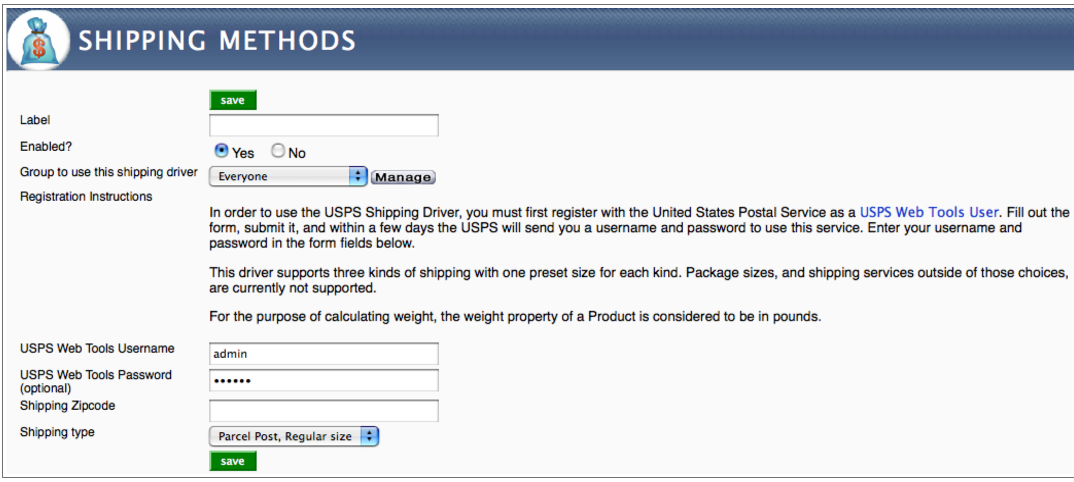
The Shipping Methods screen in where you establish shipping methods and charges associated with those methods.

The U.S. Postal Service (USPS) and Flat Rate shipping methods are included with a standard install of WebGUI.

Select a shipping method from the dropdown, and then select the “Add Shipping Method” button.

United States Postal Service

Allows you to ship via the USPS and use the shipping costs associated with the USPS.



Label: enter a label for this shipping method. This is the label site users will see. Enter something clear and easy to understand, and limit it to under 100 characters.

Enabled?: in order for this shipping method to be available to users on your site, you must set this field to Yes.

Group to use this shipping driver: select the group of users allowed to use this shipping driver.

Registration Instructions: follow the instructions in this field to register with the United States Postal Service as a USPS Web Tools User. Carefully follow these instructions and read the field in its entirety.

USPS Web Tools Username: the username you established when registering with USPS as a Web Tools User.

USPS Web Tools Password: the password you established when registering with the USPS as a Web Tools User.

Shipping Zipcode: the postal/zip code of the location you are shipping from.

Shipping type: select from the list of options. If you wish to provide multiple shipping types, create a USPS payment driver for each shipping type you would like to provide.

Flat Rate

Flat Rate shipping will allow you to configure shipping costs based on a flat fee, percentage of price, percentage of weight, or a flat price per item.

Label: this is the label, or name, of this Shipping Driver. This should be a short, clear label restricted to 100 characters or less. This is the name of the shipping driver as the user will see it on the site.

Enabled?: indicate if this shipping driver is enabled by selecting Yes or No. In order to be used, it must be set to Yes.

Group to use this shipping driver: this is the group of users allowed to select this shipping method during the checkout process.

Flat Fee: if you are going to charge a flat fee for shipping, enter the amount here. This is a standard shipping price that will be applied to a purchase, regardless of the physical size or number of items included in it.

Percentage of Price: if you want to base shipping costs on a percentage of the total purchase amount in the cart, enter the percentage amount here. For example, if you enter 5, every purchase will have 5% of the total amount in the cart added on to it to cover shipping costs. A cart that contains $120 dollars worth of products will cost $126. A cart that contains

$500 of products will have a total cost of $525.

Percentage of Weight: if you want to base shipping costs on a percentage of the total weight of all products in the cart, enter the percentage amount here.

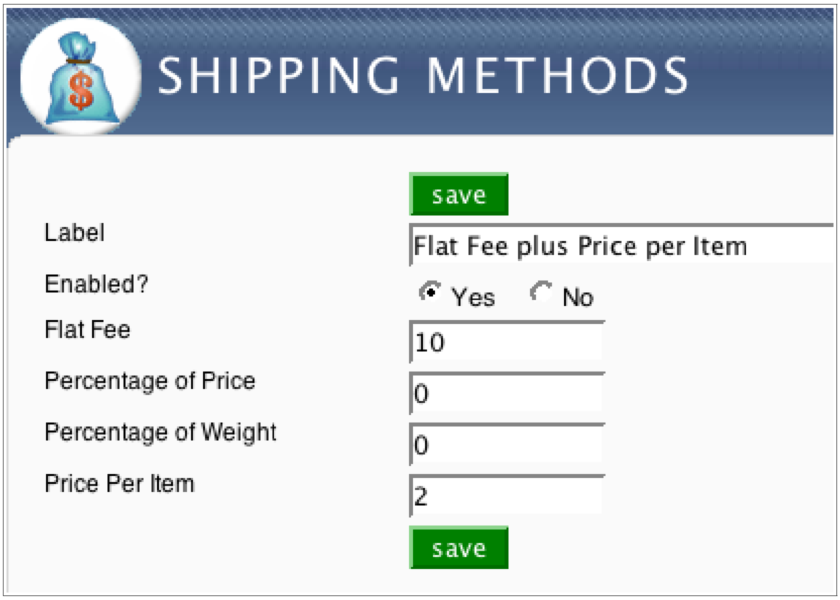
Price Per Item: if you want to charge a set amount to each item placed in the cart, enter that amount here. So, if you enter 2, and a customer places 10 items in the cart, the user will be charged $20 in addition to the total cost of all items.

When all fields are complete, click the green save button to add this shipping method.

The shipping method below shows an example of how you could charge a flat shipping amount to apply to all purchases, as well as an additional charge based on weight. This might be helpful in circumstances where some of your items are very heavy and cost more to ship.

If an item weighed 100 pounds, the normal $15 flat fee would be added to the purchase price, and an additional 10%, or $10 dollars, would be added based on the item's weight. The total amount for that item would then be $125 plus any taxes.

In the example below, you can see that shipping costs are being based on a flat fee and a price per item.



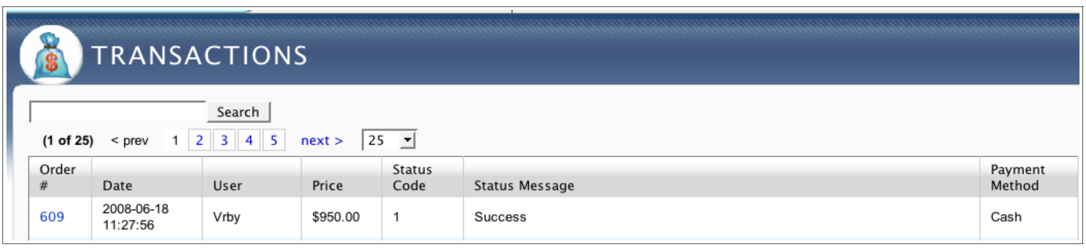
If a shopper had five items in her cart, she would be charged these shipping costs plus the prices for the items. Say the items cost $75 total. Her final cost would be $75, plus the flat fee of $10, and an additional $10 ($2 for each item in the cart), for a total cost of $95 plus any taxes.

Or, you could charge a shopper a percentage of the total cost in the cart in addition to a price per item.

In this scenario, the shopper might have eight items in her cart, for a total cost of $180. The shopper will be charged the $180, plus $37 (15% of 180) and an additional $8 ($1for each item in the cart), for a total of $215 plus any taxes.

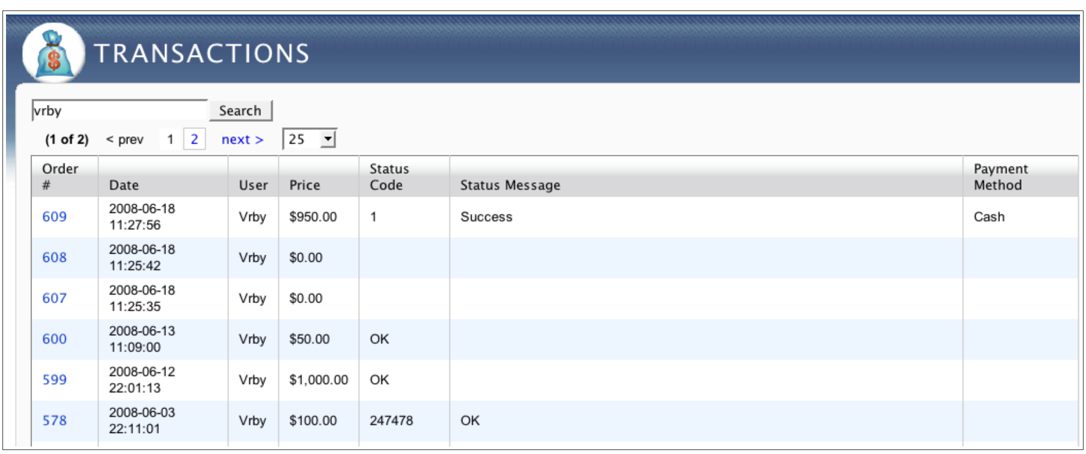
Transactions

The Transactions screen contains a list of all Shop transactions that have taken place on the site. This screen is normally not visited until after the Shop, along with its products, has been completely set up.



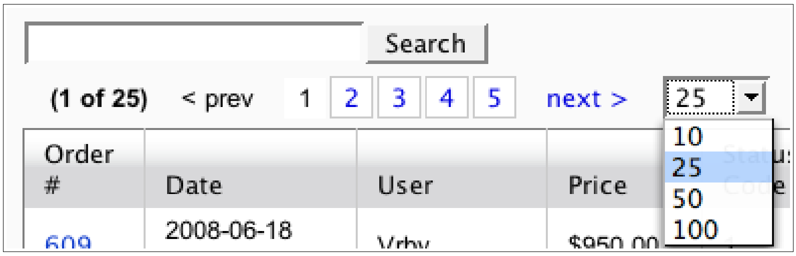
Order #	Date	User	Price	Status Code	Status Message	Payment Method
609	2008-06-18 11:27:56	Vrby	$950.00	1	Success	Cash

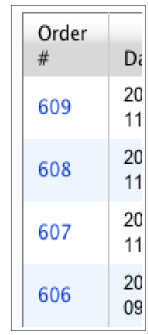
At the top of the screen is a Search field. This allows you to search the Transactions screen for all transactions by a given user, as seen below.



Order #	Date	User	Price	Status Code	Status Message	Payment Method
609	2008-06-18 11:27:56	Vrby	$950.00	1	Success	Cash
608	2008-06-18 11:25:42	Vrby	$0.00			
607	2008-06-18 11:25:35	Vrby	$0.00			
600	2008-06-13 11:09:00	Vrby	$50.00	OK		
599	2008-06-12 22:01:13	Vrby	$1,000.00	OK		
578	2008-06-03 22:11:01	Vrby	$100.00	247478	OK	

Below the Search field are links to navigate between pages of transaction entries. Each page displays twenty-five (25) transactions. The dropdown box can be used to determine how many transactions you want to view on the page.





The Order # column contains the order number that WebGUI Shop assigns to each transaction. This is the human readable number assigned to each transaction in the system for reference. Clicking on an order number will direct you to the Transaction Details.

To the right of the Order # column are the Date, User, and Price columns. This allows you to see the date the transaction took place, the username of the user who completed the transaction, and the total cost of the transaction.

Date	User	Price
2008-06-18 11:27:56	Vrby	$950.00

The fifth column from the left is the Status Code column. The Status Code is a code that is sent back from the payment gateway that indicates success or failure.

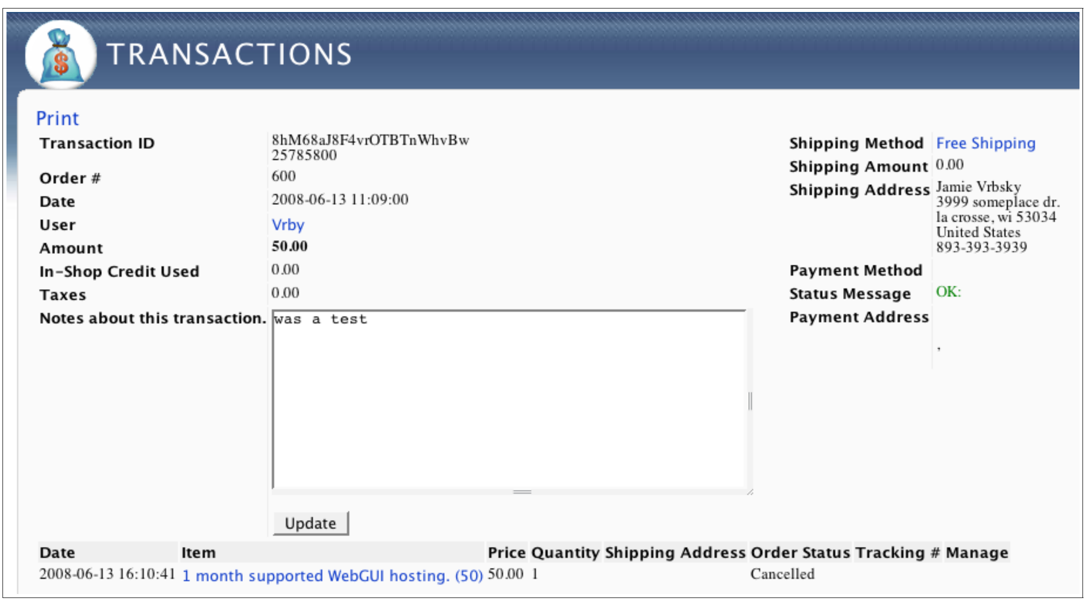
Status Code	Status Message	Payment Method
1	Success	Cash
OK		Credit Card
ERROR	Code: NBE001 Your credit card was declined by the credit card processing network. Please use another card and resubmit your transaction. Category: PROCESSOR_ERROR	Credit Card

The Status Message column contains messages relating to the status of the transaction. For example, if there is a problem with a credit card transaction, a message will display indicating the problem. If a transaction is successful you might see a success message or the field might be left blank.

The column on the far right of the screen, Payment Method, indicates what payment method was used to complete the transaction.

Transaction Details

Each transaction in Shop is assigned an order number, which is displayed in the Order # column on the left of the Transactions screen. Clicking on an order number will display the transaction details, as shown below.



Print: at the top of the details screen is a link from which to print this screen.

Transaction ID: below the print link is a Transaction ID. This is a GUID (globally unique identifier) assigned to this transaction and can not be edited. The first 22 characters of the Transaction ID refer to the unique ID assigned this transaction by WebGUI. The second portion is assigned by the payment gateway (such as iTransact) for their reference.

Order #: the human readable order number assigned to this transaction is listed in the Order # field. This is the order number referenced in the email sent to the group of users assigned to manage Shop.

Date: the date this transaction was last updated. This could be the date of the purchase, the date of shipping, the date a refund took place or any date corresponding to recent activity in this Transaction.

User: the username of the person who made the transaction.

Amount: the total amount of the transaction.

In-Shop Credit Used: if any In-Shop Credit was applied to this transaction it will be indicated in the In-Shop Credit Used field. Sometimes when you return an item to a store, they give you in-store credit to use towards another purchase in that store, as opposed to just refunding the money. Shop provides this same functionality.

Taxes: this field contains the amount of tax applied to this transaction.

Notes about this transaction: this is an area in which special notations about this transaction can be placed for your reference.

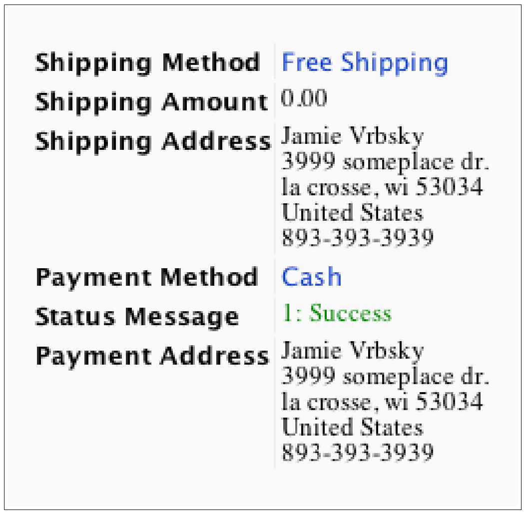
At the bottom of screen is a list of the items that were purchased with this transaction.

Date	Item	Price	Quantity	Shipping Address	Order Status	Tracking #	Manage
2008-06-18 16:27:56	WUC 2008 Badge (Early Bird)	500.00	1		Shipped		Update Refund
2008-06-18 16:27:56	WUC 2008 Badge (Early Bird)	500.00	1		Shipped		Update Refund

The date column contains the date this item was purchased. The Item column contains a list of items placed in the cart for this transaction. These items will be hyperlinked. Upon clicking on an item you will be directed to a page view of the item as it appears on the site.

The next column to the right, Price, indicates the price of each item purchased in this transaction, and to the right of that the Quantity of each item purchased. Order Status contains a dropdown menu that indicates the status of this transaction, such as item shipped, not shipped or backordered. There is also a field for a Tracking Number (#) to associate with each item. If a user has decided to return an item, the Refund button to the far right of the screen can be used. This operates just like returning an item at the store: the user will be refunded the amount of the item, in-Shop credit will be applied to the user's account, and the item will be placed back on the Shelf for others to purchase. Likewise, if this is a badge being purchased from an Event Management System, it will make that badge available for another person to purchase.

On the right hand side of the transaction detail screen are a number of fields related to shipping.



Shipping Method: indicates the shipping driver/method applied to this transaction. This is hyperlinked. Clicking on the shipping driver directs you to the Shipping Method screen in which this method was configured.

Shipping Amount: if an additional cost was applied to this transaction to cover shipping costs, the amount would be indicated here.

Shipping Address: this is the physical address to which items purchased in this transaction were shipped.

Payment Method: this indicates the method of payment used to complete this transaction. Clicking on the payment method will direct you to the Payment Methods screen in which this payment method was configured.

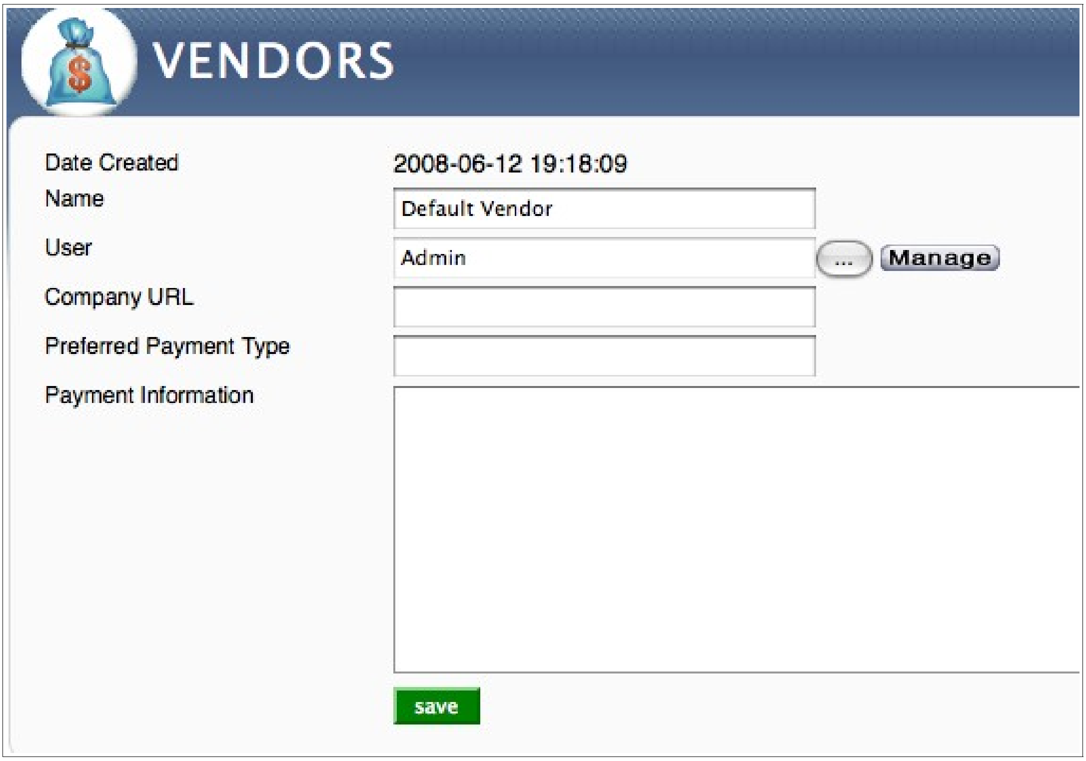
Status Message: this indicates the status of this transaction: was this transaction completed successfully? This corresponds to the Status Code and Status Message fields in the main Transaction screen, and represents a message returned by the payment gateway.

Payment Address: this is the billing address for this transaction. For example, a credit card transaction's billing address may be different than the transaction's shipping address.

Vendors

The Vendor screen allows you to keep track of vendors used in your Shop. When Products are added to Shop, you can link each product up with a Vendor that exists in this screen. Upon initially entering this screen you will see a Default Vendor already exists.

You can add a new vendor by clicking on the green “Add a Vendor” button, or edit an existing vendor by clicking its Edit button. Either way, the Vendors add/edit screen will open.



Date Created: a non-editable field that displays when this vendor was created.

Name: the name of this vendor.

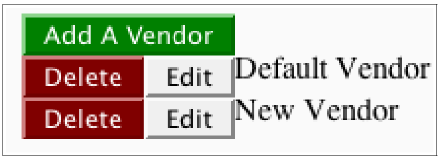
User: the username of this vendor. Clicking the "..." button will open a screen in which you can search for and select a vendor.

Company URL: the URL of the company associated with this vendor.

Preferred Payment Type: enter this vendor's preferred method of payment.

Payment Information: enter specific information related to this vendor's preferred method of payment.

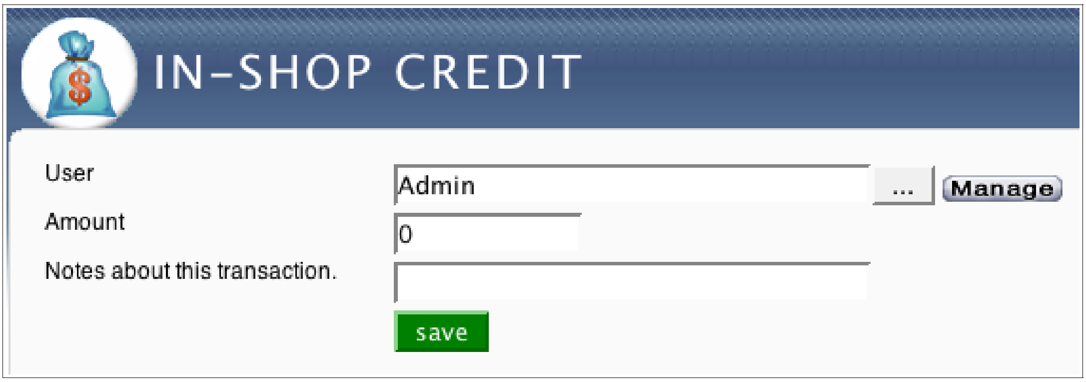
The new vendor will be added to the list on the main Vendors screen. You can use the red Delete button to delete a vendor.



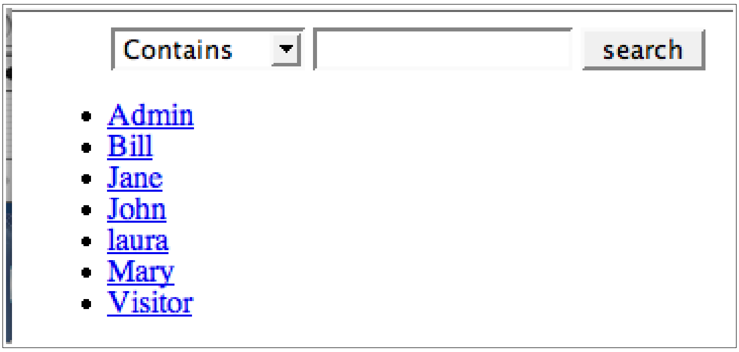
Now, when a new SKU asset is added to Shop, the asset can be linked to the vendor that supplies it.

In-Shop Credit

The In-Shop Credit screen allows you to assign in-shop credit to a specific user. This is similar to in-store credit at a department store. The customer returns an item, and instead of refunding the amount, the customer instead is given that amount to use as in-shop credit. Or, in-shop credit can be assigned to a customer on an as-needed basis for special considerations.



User: select the user who will be given this in-shop credit. The in-shop credit will be made available to the user in the user's cart. To select a user, click on the gray box (...) to the right of the field. This will open a menu of users, along with a search field, from which to select.



Simply click on a username to load the user into the User field.

Amount: enter the amount of in-shop credit this user will receive. Entering a positive value will add in-Shop credit to the user's account. Entering a negative value (ie. -20) will detract existing in-Shop credit.

Notes about this transaction: this is a field in which to enter a brief note to yourself regarding why this in-shop credit was given. For example, “returned an item to the shop.”

When finished, click save. At the top of the screen a brief message will display indicating the amount of the in-shop credit assigned to the user, along with an indication of the total amount of in-shop credit this user currently possesses (in the case that the user has in-shop credit from another time that has gone unused, it will be added to the current amount).

When the user views his/her cart, the In-Shop Credit field will show the amount of credit available, and indicate that it is being subtracted from the total purchase amount. In-Shop credit will automatically be deducted from the total purchase amount. The shopper will be responsible for the remaining balance.

If the user has more in-shop credit than the total purchase amount, the remaining balance will remain in the user's account to use on a future purchase.

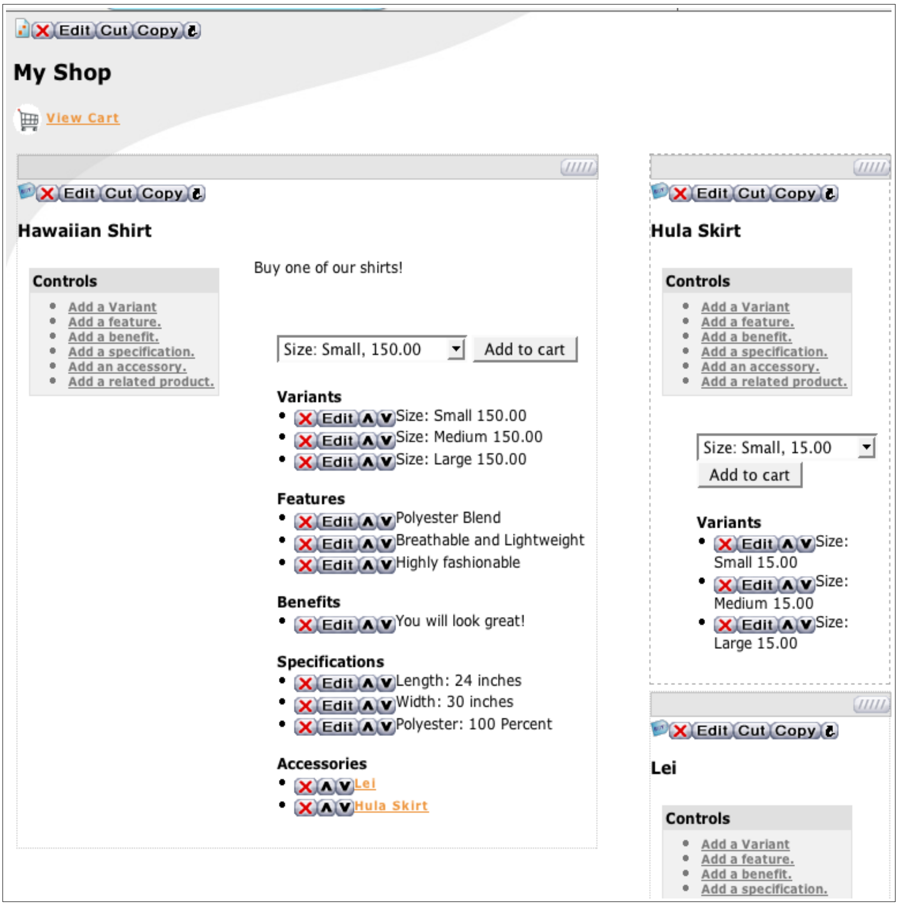
Create Your Shop Front

Once WebGUI Shop is configured by a site administrator through the Admin Console, it's time to create a shop front. After all, people can't shop if they don't have a store to shop at.

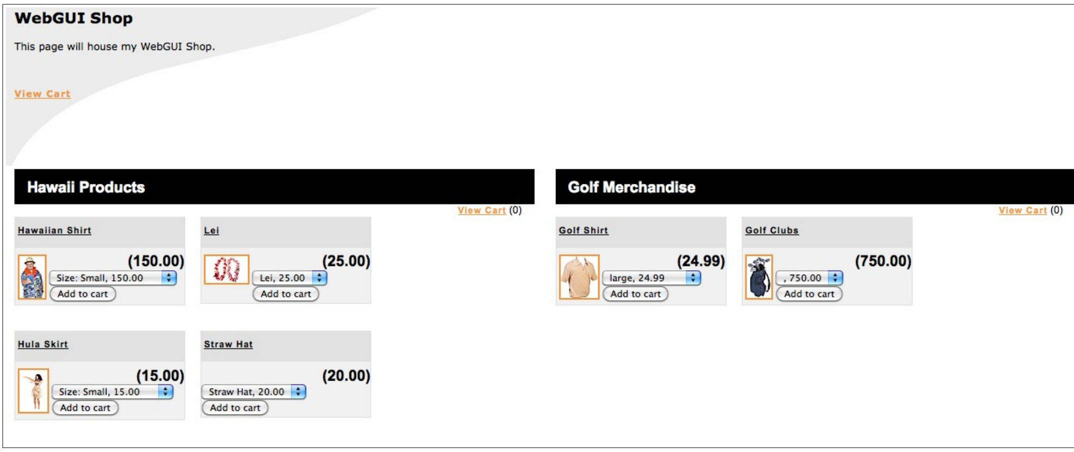
You'll want to create a container asset to house the shop. Container assets are found in the New Content menu of the Admin Bar, and are located at the top of that menu. You first add a container asset to the site, and then you fill up that asset with regular assets. Some container assets you may be familiar with include the Page Layout asset, used to create a new page on a site, and the Folder asset. WebGUI Shop introduces a new container asset called Shelf, which will be discussed in its own chapter.

It is recommended that the ^ViewCart; and/or ^MiniCart; macros be built into the site/shop design to make it easy for users to check out from anywhere in your Shop. See the section on these macros earlier in this chapter to learn more about them.

There are a couple of options to consider when organizing your shop. If you are selling a fairly small number of Products, or other SKU assets, on your site, you could probably get away with simply creating a Page Layout and adding Products directly to it, as illustrated below. In this scenario, your shop is in one location. Users visit the shop page, and add items to their carts directly from it.



However, if you are selling many Products of a wide variety, you might want to organize your Products into categories. To do this, you use the Shelf asset. Just like in a physical store, you can create a Shelf, and place Products directly on it. Shelf will display the Product asset's title/name, an image if there is one associated with it, and the Product price. Users click on a Product on a Shelf to be directed to the actual Product asset where it can be added to the user's cart, or the user can add a product directly to the cart from the shelf. In the example below, a Page Layout was created. Then different Shelf assets were added to it to create categories of related Products. Alternatively, you could create a Shelf as a parent asset and access the Shelf directly from the site navigation.

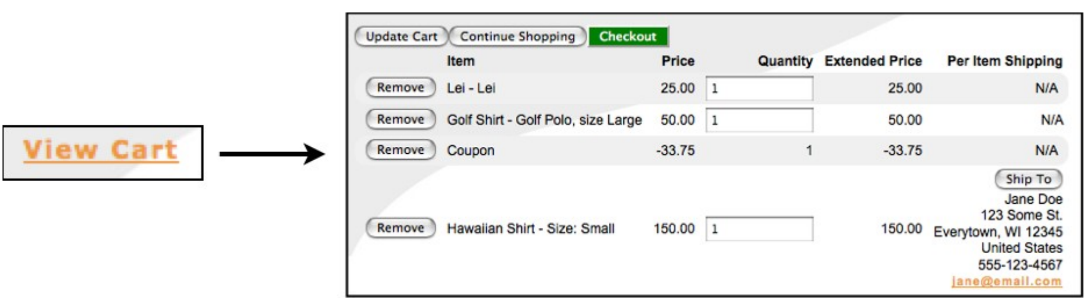


Shop Macros

Before going further, it's important to understand what the Shop related macros accomplish. These macros should be built into your Shop style. This will ensure that they appear to the user from anywhere in your Shop.

^ViewCart;

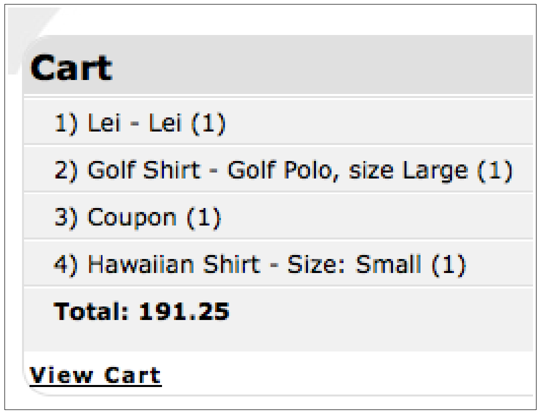
This macro will display a small cart icon next to a “View Cart” link. Users can click on this to be directed to the full cart view.



The ^ViewCart; macro is included in some SKU asset templates by default.

^MiniCart;

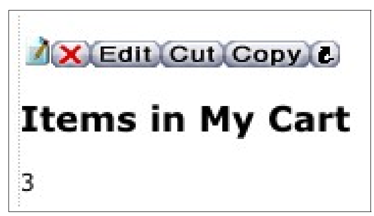
The ^MiniCart; macro places a miniature cart directly on the body of the page. When the asset is added to the style, the macro displays a mini version of the full cart view. Included in the mini cart view are each item, its quantity currently in the user's cart, and the total purchase price. There is also a link to direct the user to the full cart view so the user can complete the purchase.



Use this macro if you want users to always be able to see what is currently in their carts, without leaving their current location on the site.

^CartItemCount;

The ^CartItemCount; macro simply displays the number of items in a user's shopping cart. In the example to the right, the user has three items in the cart.



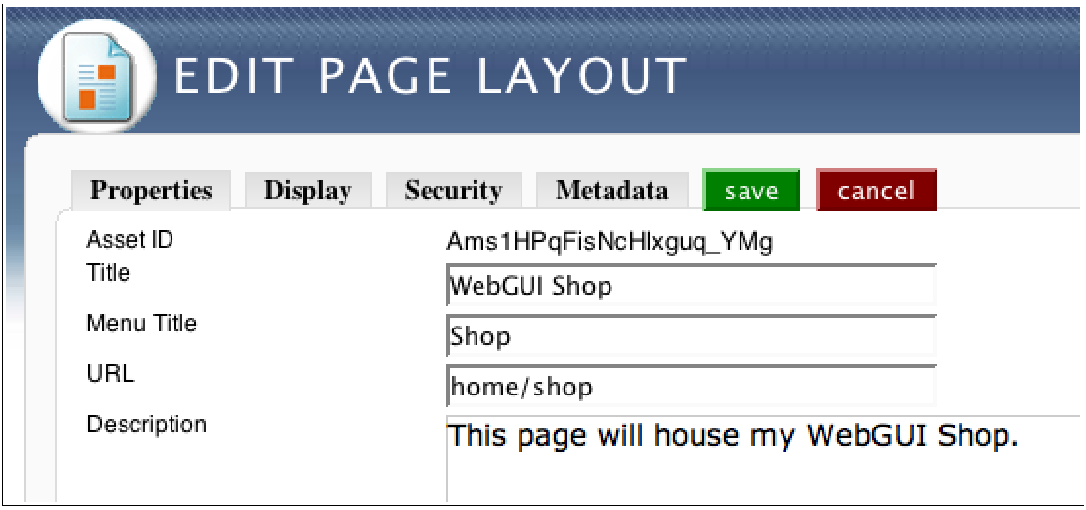
The true intent of ^CartItemCount; is to be combined with ^ViewCart; so that the user sees a link to the cart along with an indication of how many items are currently in the cart. To do so, the following would be entered: ^ViewCart;(^CartItemCount;) This generates (included in Shelf by default):

Set up Your Premises

Create a container asset to house your Shop. This first example uses a Page Layout. Shelf will be covered in its own section.

1. Under the New Content tab, click on “Page Layout.” The “Add Page

Layout" screen will open.

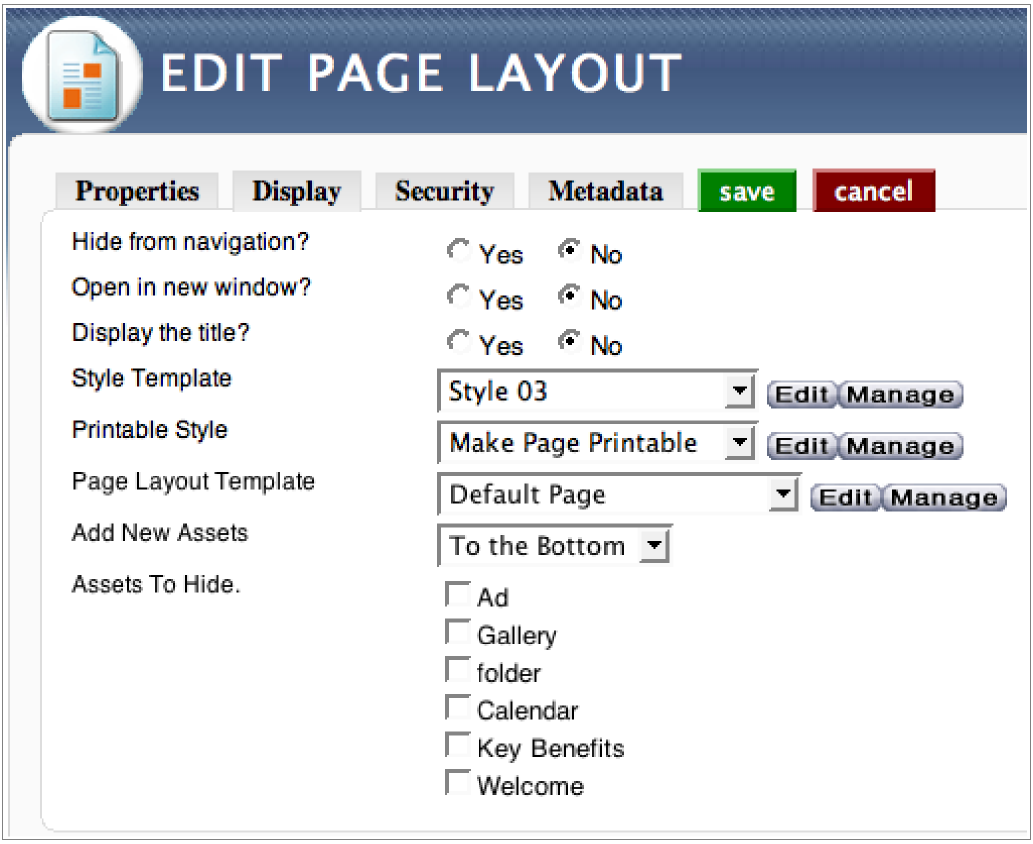


2. In the "Title" box near the top of the screen give your shop a title.

3. The "Menu Title" is the title of this page, or your shop, as it will appear in the site navigation. The URL field is the URL of this asset, or page. You can leave the Menu Title and URL fields blank and WebGUI will fill them in for you, or you can enter them yourself.

4. In the "Description" field you can enter content that will be displayed as static content at the top of this page, regardless of any other content placed on the page.

5. In the Display tab you can set display options for this page.



A. The “Hide from navigation?” field determines if this page's menu title will appear in the site navigation.

B. The “Open in new window?” field determines if this page will be opened in a new browser window..

C. The “Display the title?” field determines if the title of this page will be displayed on the body of the page to the user.

D. The “Style Template” field allows you to select a style to wrap this page in from the dropdown menu. If you have a special style template to apply to your shop, select it here. A page layout's style will override the style of any regular asset that is placed on it.

E. The “Printable Style” is the style template used for the printable version of this page.

F. The “Page Layout Template” determines the placement of regular assets on the page. This will be represented by some shaded gray boxes.

G. The "Add New Assets" field allows you to determine if you want new assets, such as Products, to be added to the page at the bottom of the page or at the top of the page. Assets can then be dragged and dropped into new positions.

H. The "Assets to Hide" field will only contain options if you are editing an existing Page Layout asset that already is filled with content. It contains a checkbox list of assets this page contains. You can check an asset you want hidden from view. When you are ready for it to display again, simply uncheck it. If you are creating a brand new page to house your shop, this field will be blank. In regards to Shop, you might wish to hide Products that are currently not available in your shop, but will be again at a later time. For example, seasonal items that you only sell during certain months of the year. By hiding them, users won't be able to add them to their carts to purchase, but the Product asset will still exist and you can still access it and manage it through the asset manager. When it's time to sell that Product again, simply make it visible again.

6. The Security tab allows you to set viewing and editing privileges for this page.

A. The "Owner" will always have full editing and viewing privileges for this asset.

B. The "Who can view?" fields allows you to select the group of users allowed to view content displayed on this page.

C. The “Who can edit?” field determines the group of users allowed to edit this page layout asset.

7. Metadata allows you to attach arbitrary fields to your content for search engine friendliness, profiling, or simply to add extra display options. The standard metadata tab is shown below. To add metadata fields, use Content Profiling in the Admin Console.

8. Click “save” at the top of the screen. This will bring you back to the new page you just added.
   You will see the title at the top, followed by the text you entered in the description field. If you entered the ViewCart macro you also see the cart icon next to the View Cart link.

Once a page has been created to house your Shop, it's time to create Products for people to buy, and Shelf assets to help keep your Shop tidy.

Shelf

If your Shop contains relatively few products, and if those products all relate to one another, it's fine to just add Product assets directly to a Page Layout. For larger Shops, you may want to categorize your Products. Shelf allows you to create a "shelf" in your shop to house Products that relate to one another. It is a container asset, so you will add a Shelf, and then fill it with Products to sell. A Shelf asset will display on a Page Layout, or it can be created as a parent asset to be accessed directly from the site navigation if you wish.

You don't have to add a Shelf to a Page Layout. You can add it directly to the site from any location. Like other container assets, it will appear one level lower in the site navigation from the current location.

In addition to placing assets directly on the Shelf, Shelf can also be populated using Metadata keywords. Any SKU asset on the site (Product, Donation, Subscription, Flat Discount Coupon) can be assigned a keyword in its Metadata tab. If a matching keyword is entered in the Keyword field of a Shelf asset's Metadata tab, the Shelf asset will pull in the SKU asset to display on the Shelf. When users click on an item they will be directed to the original asset. This is covered more later.

1. Select Shelf from the New Content menu to open the Add/Edit Shelf screen.

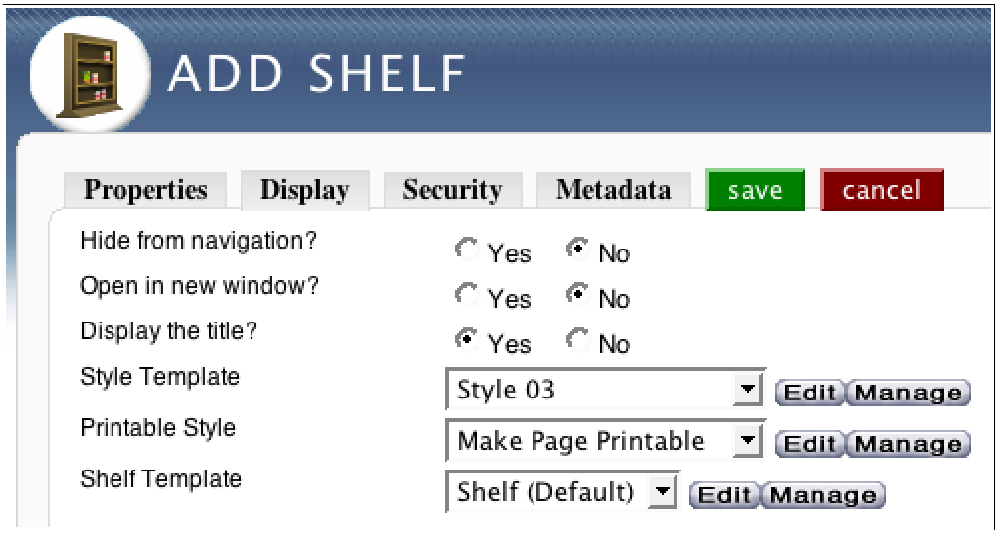
2. In the "Title" field enter a title for this Shelf.

3. The “Menu Title” field is the title of this Shelf as it appears in the site navigation. If you leave this field blank, WebGUI will just use the title entered in the “Title” field for this field as well.

4. The URL field is the URL of this asset. You can enter a URL, or leave it blank and WebGUI will create one for you.

5. In the “Description” field you can enter text related to what will be displayed on this Shelf.

6. Set your display settings in the Display tab.



A. If “Hide from navigation?” is set to Yes, the menu title of this Shelf will be hidden in the site navigation. Because Shelf is a container asset, the default setting for this field is No, meaning the menu title of this Shelf will appear in the site navigation. If you don't want that, set this field to Yes.

B. “Open in new window?” determines if this asset will open in a new browser window when selected from the site navigation.

C. “Display the title?” determines if the title of this asset will be displayed on the body of the web page.

D. The “Style Template” is the style template used to view this asset, similar to applying a style template to a Page Layout asset. If you have a custom style template for your Shelf asset, this is where you select it.

E. The “Printable Style” is the template used to display a printer-friendly version of this asset.

F. The “Shelf Template” is the template used to display this Shelf asset.

7. The Security tab allows you to determine who can view this Shelf on the site, and who can edit this Shelf asset. The Owner can always do both.

8. The Metadata tab is the standard metadata screen common amongst most WebGUI assets. Shelf will allow you to use the Keywords field in the Metadata tab to display Products or other SKU assets from throughout the site that contain matching keywords.

   For example, a number of Product assets are created on the site. Many of the Products are related to Hawaii, so in their Metadata tabs each related Product asset is assigned the Keyword “hawaii”.

   Then, a Shelf asset is created on which to store all Hawaii related products. So, the Shelf asset is named Hawaii, and the same keyword, hawaii, is entered in the Keyword field in the Shelf asset's Metadata tab.

Upon saving the Shelf, the asset will display all the Product assets (or any other SKU asset on the site) with the matching keyword "hawaii" associated with them.

If you manually add a SKU asset to Shelf, it will become a child of the Shelf asset. Pulling in SKU assets for display on a Shelf asset via a Metadata keyword does not create child assets. Users will be directed to the actual SKU asset upon selecting it in Shelf.

This will allow you to populate the Shelf asset using keyword associations, as opposed to manually adding Product assets to the Shelf asset. In this way, you can create any number of SKU's (Products, Subscriptions, Donations, Flat Discount Coupons) and place those assets anywhere on the site. Then, create categories within your Shop using the Shelf asset, and populate the Shelf assets representing each category based on keyword association. This can result in some overlapping in the event that two categories have identical keyword associations. For example, a Hawaiian shirt might have the keywords hawaii and shirt. This might result in it being displayed on two Shelf assets: one with the keyword hawaii, and one that displays clothing items with a keyword of shirt.

You might also use the keyword associations to your advantage when creating Flat Discount Coupons. A single Flat Discount Coupon asset can be created for your entire Shop ("shop today and save X amount"), but displayed on multiple Shelf assets. Because it is a single coupon asset, WebGUI will still only allow it to be applied to the total purchase one time.

9. Click save to add your Shelf to the site.

SKU assets include Product, Donation, Subscription, Flat Discount Coupon, and the EMS Ticket, Badge, Ribbon and Token.

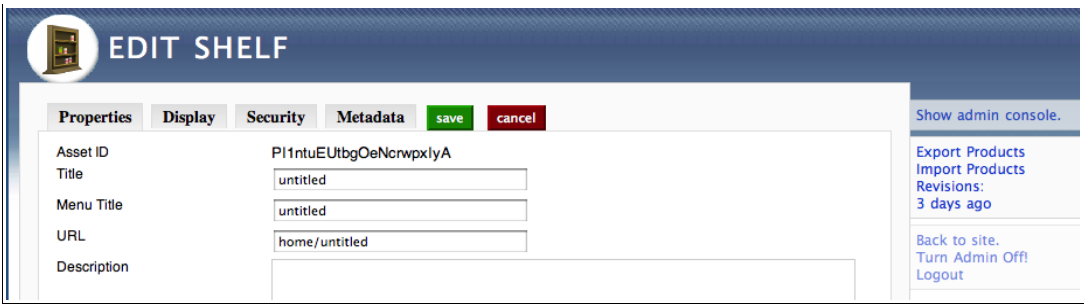
Once you create a Shelf, you can add SKU assets, such as Product or Donation, to it, or use the Keyword field in the Metadata tab to pull in SKU assets with matching keyword associations. The image above shows a Shelf asset with three Product assets added to it. If the user clicks on the Hawaiian Shirt product name, or the product's image, the user will be directed to the Product asset for that item. From there, the user can see variants of the product, learn more about the product and add the product to the cart. Alternatively, the user can select the variant of the Product and add it to the cart directly from the Shelf.

The Shelf asset contains a View Cart link in the default template, located in the upper right hand corner of the asset. This link will display the number of items currently in the user's cart. Clicking the View Cart link will direct the user to his/her shopping cart.

Import and Export Products

WebGUI Shop allows you to export data from existing Product assets from your shop to your computer in a .csv (comma separated value) file for use in an external inventory control system or to import into another WebGUI Shop.

To do so, edit a Shelf asset that contains Product assets as children. On the far right hand side of the Shelf asset add/edit screen is an “Export Products” link.



Clicking the Export Products link will download a .csv file to your computer containing data from the Product assets contained on this Shelf. WebGUI will open a page displaying this data upon clicking on the Export Products link.

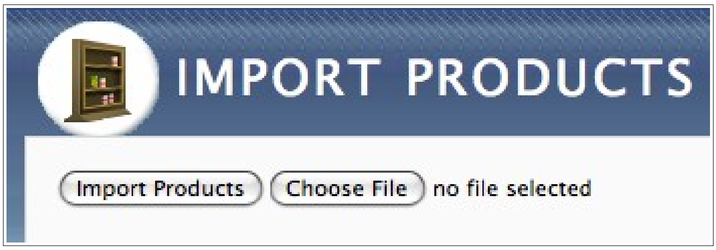
mastersku	title	sku	shortdescriptio	price	weight	quantity
y1sEVu5hmGzv	Lei		Lei	25	0	0
y1sEVu5hmGzv	Lei		Plumeria Lei	25	0	10
y1sEVu5hmGzv	Lei		White Ginger	30	0	10
kvG2hl3CvjCyls	Hula Skirt		Size: Small	15	0	10
kvG2hl3CvjCyls	Hula Skirt		Size: Medium	15	0	10
kvG2hl3CvjCyls	Hula Skirt		Size: Large	15	0	10
TShirt	Hawaiian Shirt		Size: Small	150	0	54
TShirt	Hawaiian Shirt		Size: Medium	150	0	32
TShirt	Hawaiian Shirt		Size: Large	150	0	82
y1sEVu5hmGzv	Lei		Lei	25	0	0
kvG2hl3CvjCyls	Hula Skirt		Size: Small	15	0	10
kvG2hl3CvjCyls	Hula Skirt		Size: Medium	15	0	10
kvG2hl3CvjCyls	Hula Skirt		Size: Large	15	0	10
4QzH-wU6zpXI	Golf Shirt		Golf Polo, size l	50	0	10
gBJ_R54yxq1Q	Golf Clubs		Full Set	875	0	0
gBJ_R54yxq1Q	Golf Clubs		Pitch Wedge	45	0	0
gBJ_R54yxq1Q	Golf Clubs		Triangular Drive	30	0	0
TShirt	Hawaiian Shirt (copy)		Size: Small	150	0	54
TShirt	Hawaiian Shirt (copy)		Size: Medium	150	0	33
TShirt	Hawaiian Shirt (copy)		Size: Large	150	0	82
TShirt	Hawaiian Shirt		Size: Small	150	0	53
TShirt	Hawaiian Shirt		Size: Medium	150	0	33
TShirt	Hawaiian Shirt		Size: Large	150	0	82
kXEzw1lVX9lcK	Straw Hat		Straw Hat	20	0	35

The file can then be imported into an external spreadsheet application to manage product inventory, as seen above.

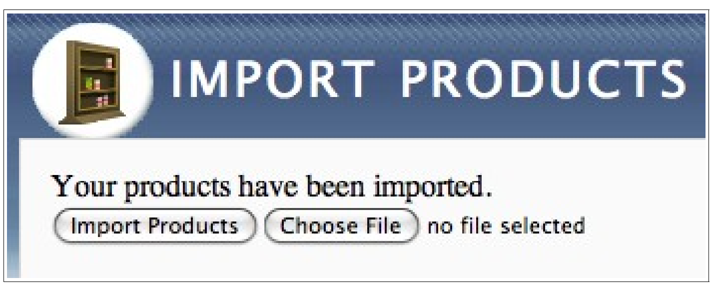
The Import option allows you to import many Products from an external .csv file, as an alternative to creating many Product assets. In some cases, you might be managing product inventory in an external application, and this would allow you to import Product data back into WebGUI Shop. If

WebGUI finds a Product asset in the uploaded file, it will update its data on the site based on the data in the .csv file (quantity, price, title etc).

To import Products, edit a Shelf asset, and click on the “Import Products” link on the far right hand side of the Shelf add/edit screen. This will open the Import Products screen.



Click on the Choose File button, select the .csv file from your computer, and upload it. A message on the screen will indicate that the products have been imported.



WebGUI will import the products directly to the Shelf asset, so Shelf will be the parent asset and the Products will be contained under Shelf. To view the Products that you have imported, simply click the Back to Site link on the far right hand side of the screen. You will be returned to the Shelf asset on which the Products are now displayed.

Add Products

Create products to sell in your Shop by using the Product asset, located in the New Content menu.

1. Select Product from the New Content menu to open the Add Product screen.

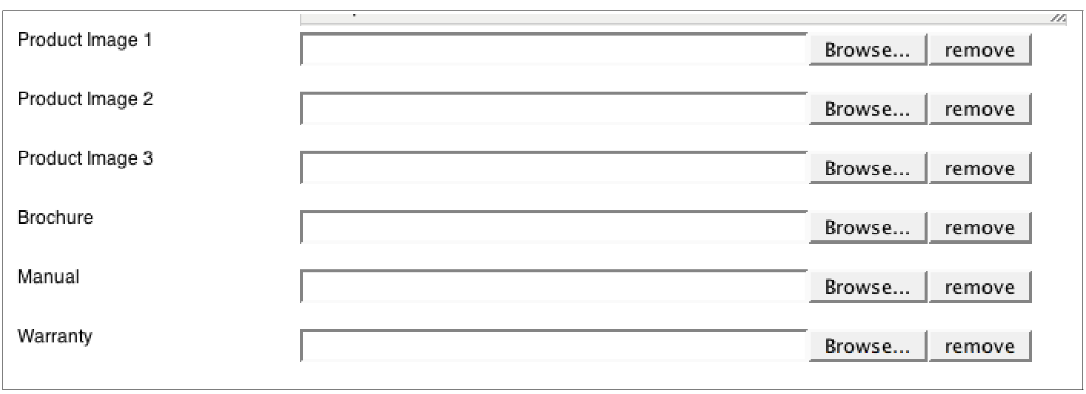
2. Give you product a name in the “Title” field.

3. The “Menu Title” is the title of this product as it would appear in the site navigation. You can enter a menu title, or you can leave this field blank and WebGUI will automatically use the Title for this field as well.

4. The “URL” is the URL for this specific Product asset. You can specify a URL, separating words with a dash or underscore, or you can leave this field blank and WebGUI will generate a simple URL for you.

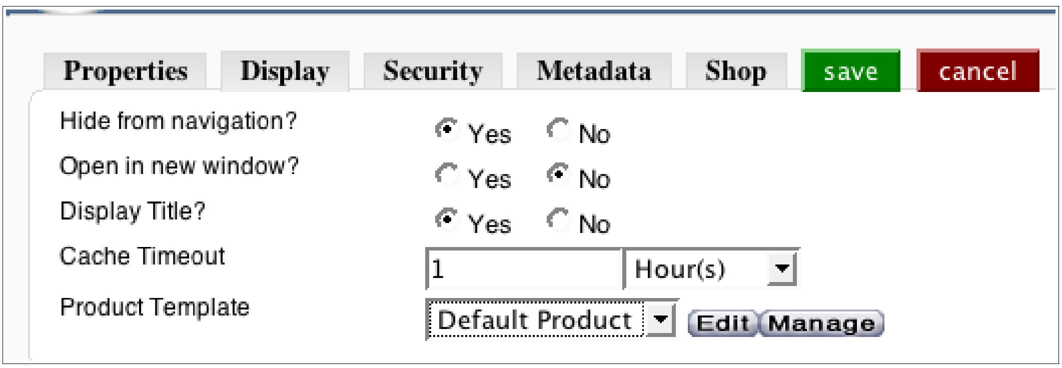
You can also add the ^ViewCart; (and/or ^MiniCart;) macro to the Description field of a Product asset. This will place a View Cart icon/link within the body of a Product asset. If you only have one Product for sale on a page, this is a good option. This might also be useful if you have an article placed on your site, outside of the Shop, and you want to allow people to purchase a Product directly from the same page as the article. Another idea is to use ^AssetProxy; to pull the Product asset from the Shop into the page on which the article is displayed.

5. In the “Description” field you can enter a description of the product you're selling.
6. The “Thank You Message” field is used to enter a message the user will receive upon purchasing this product. A user adds the Product to their cart via the Add to Cart button. Once this is done, the thank you message is displayed along with a link to access the shopping cart.
7. Near the bottom of the screen are a number of fields that can be used to upload files related to this product. Click the Browse button to the right of these fields to locate an image, brochure, manual, or warranty file on your workstation to upload. To remove a file from a field, click the remove button.



If you are editing an existing Product asset, the file names will appear in these fields. To delete an existing file from one of these fields, click on the red X next to the file, and confirm that you wish to delete it in the “Delete this file?” field by clicking on Yes or No.

8. Set the display settings in the Display tab.



A. The “Hide from navigation?” field lets you determine if the menu title of this asset will appear in the site navigation. The default is Yes, meaning the menu title will not appear.

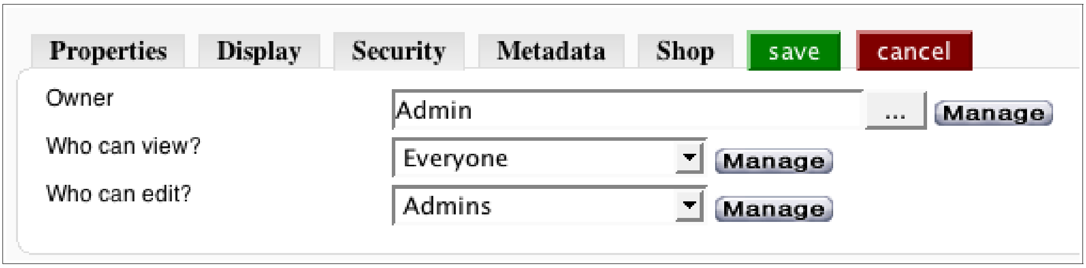
B. If the “Open in new window?” field is set to Yes, this asset will open in a new browser window when it is clicked on in the site navigation.

C. “Display the title?” lets you determine if you want the title entered in the Title field of the Properties tab displayed on the page.

D. The “Cache Timeout” field determines the amount of time to pass before the cached view of this asset is cleared. Since all users will view this asset the same way, the cache timeout can be set at a long length of time to increase performance.

E. The “Product Template” field is used to select a template in which to view this product. If you have a special template to use for Products, this is where you would select it.

9. Set this Product's security settings in the Security tab.



A. The “Owner” of an asset is a user who always has editing and viewing privileges. Usually, this is the user who created the asset. To select a new Owner, click on the gray box to the right of the field.

B. The “Who can view?” field determines the group of users allowed to see this asset on the site.

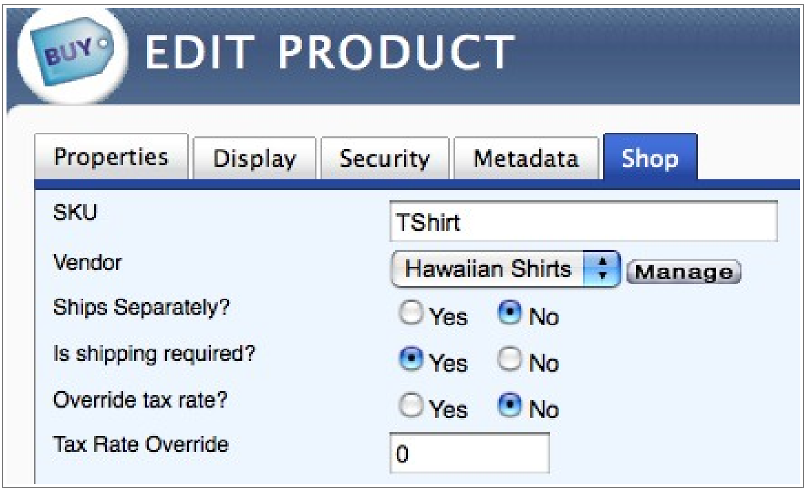
C. The “Who can edit?” field determines the group of users who have editing rights for this asset.

10. If necessary, apply Metadata in the Metadata tab. This is the same metadata tab common amongst most assets. You can learn more about its fields in the instructions for creating a Page Layout earlier in this section.

    Entering an identical metadata keyword in a Product asset and Shelf asset will allow you to pull in a Product from anywhere on the site for display on Shelf. See the section on Shelf for more information.

11. The Shop tab contains settings specific to this Product asset. This screen's appearance is different depending on the Tax Driver Shop is using.

Generic Tax Driver



A. The “SKU” field contains an inventory code (stock keeping unit) generated for this Product. You can erase the random code generated by WebGUI and enter your own value. For example, you could just enter Tshirt. This is the base SKU for all variants of this product.

B. In the Shop Settings, a site administrator was able to add Vendors. The “Vendor “ field is where you can select the Vendor for this Product from the dropdown menu.

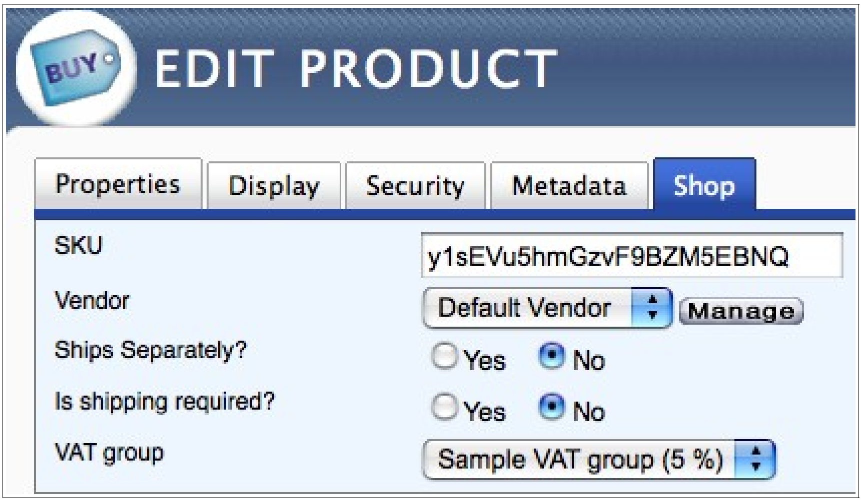
C. If “Ships Separately?” is set to Yes, this Product will incur additional shipping costs separate from the rest of the order.

D. If “Is shipping required” is set to No, this Product will not incur shipping costs at checkout.

E. The “Override tax rate?” field allows you to override the default tax rate that was set in the Shop Settings. This field corresponds to the following field, “Tax Rate Override.” This field might be used in areas where no tax is applied for life-essential products, like food.

F. If the “Tax Rate Override?” field is set to Yes, you should enter the new percentage used to calculate tax for this product in the “Tax Rate Override?” field.

European Union Tax Driver



A. The “SKU” field contains an inventory code (stock keeping unit) generated for this Product. You can erase the random code generated by WebGUI and enter your own value. For example, you could just enter Tshirt. This is the base SKU for all variants of this product.

B. In the Shop Settings, a site administrator was able to add Vendors. The “Vendor “ field is where you can select the Vendor for this Product from the dropdown menu.

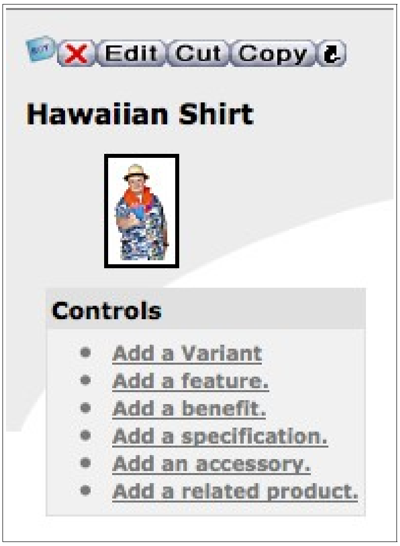
C. If “Ships Separately?” is set to Yes, this Product will incur additional shipping costs separate from the rest of the order.

D. If “Is shipping required” is set to No, this Product will not incur shipping costs at checkout.

E. If using the European Union tax driver, you will need to select a VAT group from the “VAT group” field.

12. Click save to add your product to the shop.

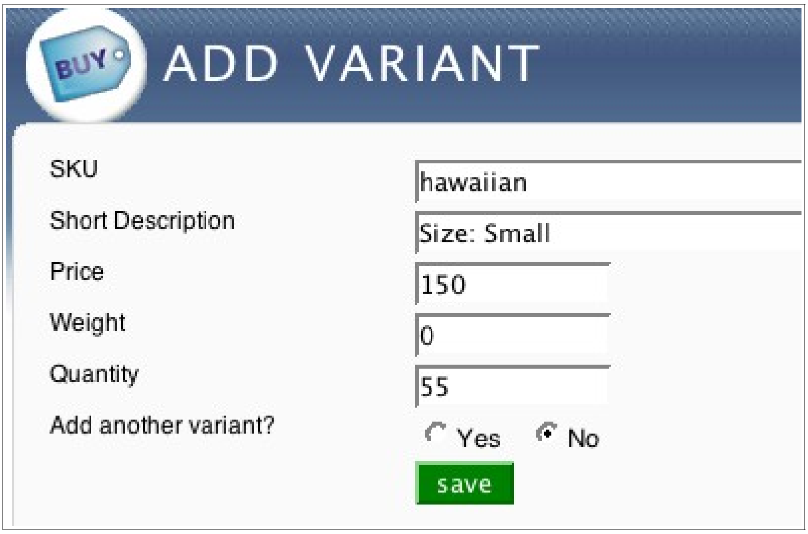
Once a Product is created, a Controls menu will appear inside the Product asset. This menu will only be visible while in Admin mode. The Controls Menu contains options for further customizing your Product.



Add a Variant

Once a Product asset is added to your Shop, you will be able to set its price and SKU from the Controls menu. This is the screen in which the price and first product variant SKU is established. After you have initially completed this screen, the link will disappear from the Controls menu for this product and you will see “Add a Variant” instead.

Clicking on the “Set a Product Price and SKU” link opens the “Add Variant” screen. Product variants allow you to create different versions of the same Product. For example, the Product may be a shirt, but you will want to sell different sizes or colors of that shirt. Product variants allow you to do that.



- SKU: enter a SKU (stock keeping unit) for this product variant. This will be appended to the base SKU established at the time of the Product asset creation.

- Short Description: a place to enter a very brief description of this product variant. Text entered in this field will appear as the description of this product variant on the screen to the user, so normally this is a descriptive phrase alerting the user to what s/he is purchasing. When a shopper views the Product on the site, a dropdown menu is made available from which to select different variants of the Product. This is the text that appears in that dropdown menu.

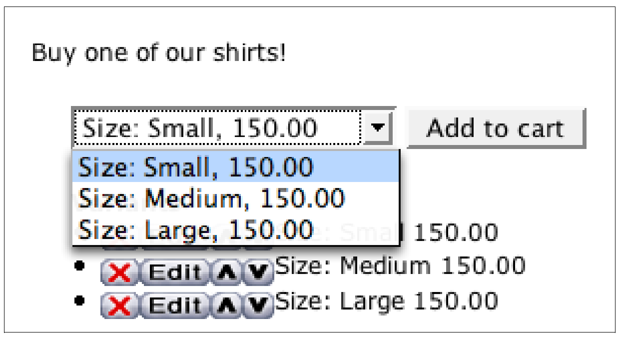
- Price: the price of this Product variant.

- Weight: this is the weight of this Product variant. If you are charging shipping costs based on weight, you would want to enter a Product variant's weight here.

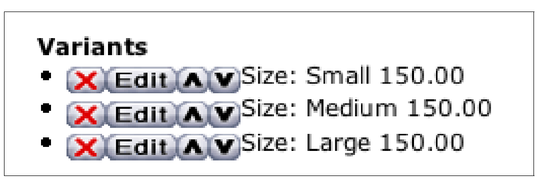
- Quantity: this is the quantity of this Product variant in stock. WebGUI Shop will keep track of how many items of this variant are in stock, and will alter this quantity accordingly. When it is out of stock, this variant will not be available to add to the cart. When all variants are out of stock, an out of stock message will appear for this Product asset. If you have a unique item in your shop, this field is very helpful. It allows you to tell WebGUI that only 1 of this kind exists, and when a user places it in his/her cart, Shop will know that no more are available.

- Add another variant?: if set to Yes, upon clicking Save another Product Variant screen will open, and you can create another variant of this product. If set to No, you will be directed back to the site.

Upon returning to the page/Shelf, the product variant will display in a dropdown menu next to the Add to Cart button. Only variants of this product that are currently in stock will be made available as a selection.



In Admin Mode, each variant will be listed in the asset as well.



Each variant will have an asset toolbar. Use the red X to delete a variant, the Edit button to edit the properties of a variant, or the up and down arrows to rearrange the order in which variants are displayed.

Add a Feature

The “Add a feature” link in the Product asset's Controls menu allows you to display product features within the Product asset. In this way, you can point out additional information of interest to your shoppers. For example, if you're selling shirts, you might want to point out what material they're made of, what country they're manufactured in, or any other feature that is unique to the product.

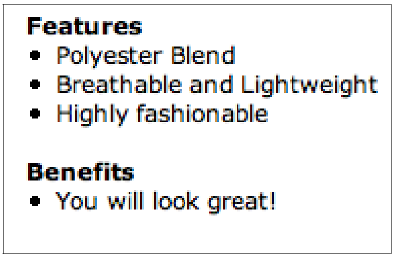
Click on the “Add a feature” link in the Product asset's Controls menu to open the Product Feature Add/Edit screen.

In the "Feature" field, simply enter the feature you want to point out. Then, in the "Add another feature?" field, indicate if you would like to continue adding features by setting the field to Yes, or simply return to the site by selecting No. Click save to add the feature.

Once all features are added, you will see them listed under the Features heading in the Product asset.

In Admin mode, each feature will have an asset toolbar. Click the red X to delete the feature, the Edit button to edit a feature's text, or use the up and down arrows to arrange the order in which features are displayed.

When Admin mode is turned off, each feature is listed in a bulleted list.



Add a Benefit

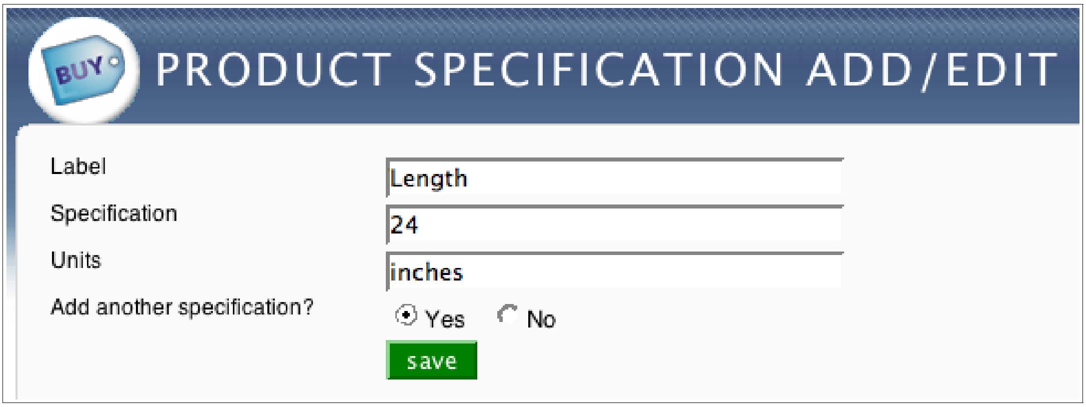
The “Add a benefit” feature is similar to adding product features. It allows you to display the benefits of using this product. Clicking on the “Add a benefit” link in the Product asset's Controls menu opens the “Product Benefit Add/Edit” screen.

In the “Benefit” field, enter a benefit associated with this product. Then, in the “Add another benefit?” field, indicate if you would like to continue adding benefits by setting the field to Yes, or simply return to the site by selecting No. Click save to add the benefit.

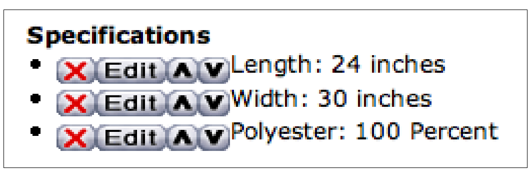
In Admin mode, each benefit entered is displayed in list under the Benefits heading. Each benefit will have its own asset toolbar. Click the red X to delete the benefit, the Edit button to edit a benefit's text, or use the up and down arrows to arrange the order in which benefits are displayed.

Add a Specification

The “Add a Specification” link allows you to display a list of product specifications. For example, often furniture pieces will have specifications listed informing the user of the furniture dimensions. Clothing might have specifications informing people of what materials they are made of, or measurements. Click on the “Add a Specification” link to open the “Product Specification Add/Edit” screen.



In the “Label” field, enter the label of this specification. This is basically the name of the specification as it appears on the page to the user. In the “Specification” field enter the value of this specification, and in the “Units” field enter the units used to measure this specification. In the example above, you can see that a length specification is being added. The length of the product is 24 inches. Then, in the “Add another specification?” field, indicate if you would like to continue adding product specifications by setting the field to Yes, or simply return to the site by selecting No. Click save to add the product specification.



In the Product asset, each specification has its own asset toolbar (when in Admin mode). Use the red X to delete a specification. The Edit button allows you to edit a specification, and the up and down arrows allow you rearrange the order in which the specifications are displayed.

Add an Accessory

The “Add an Accessory” link in a Product asset's Controls menu allows you to associate the product with other existing products on the site. For example, when a user is looking at a shirt, the user can also see other existing products related to it available for purchase in your shop.

Click on the “Add an Accessory” link in the Product asset's Controls menu to open the “Product Accessory Add/Edit” screen.

The “Accessory” field contains a dropdown menu. This menu lists all the Product assets that exist on your site. Select which product you want to display as an accessory to the current Product asset. Other Product assets can only be made an accessory to this Product once; so, when you select a product from this field, and save it, it will no longer appear as an option in this dropdown menu.

In the “Add another accessory?” field, indicate if you would like to add another accessory to associate with this product by setting the field to Yes, or select No to return to the site. Click save to add the accessory.

Each accessory will be listed, and will be linked to the Product asset for that accessory. Users can click on an accessory's name, view that accessory's Product asset, and purchase it. In Admin mode, each accessory will have an asset toolbar. Use the red X to remove an accessory, or the up and down arrows to rearrange the order in which the accessories are listed.

Add a Related Product

Adding a related product is similar to adding a product accessory. It allows you to link users to products related to the one they are viewing. For example, if you are selling a golf shirt, you might want to link users to golf clubs or tees to purchase through your shop.

Clicking on the “Add a related product” link in a Product asset's Controls menu opens the “Product Related Add/Edit” screen.

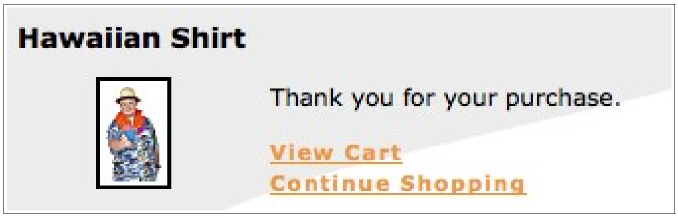
In the “Related Product” field you can select an existing Product asset to relate to the current Product asset. A product can only be selected once, and will no longer appear as an option once it has been chosen. In the “Add another related product?” field indicate if you would like to add another related product by selecting Yes, or simply return to the site by selecting No. Click save to add the related product.

In the Product asset, the related Product will be listed in the Related Products field, and linked to the related Product's asset, where the user can purchase it. If more than one related Product is listed, each will have an asset toolbar from which to remove it, or rearrange the order in which it is displayed.

Below is a Product asset with all items from the Controls menu complete.

Hovering your mouse over the thumbnail image of the product will display a larger image.

When a user visits a Product asset and adds a Product to the cart, the thank you message will display, along with a link to View Cart.

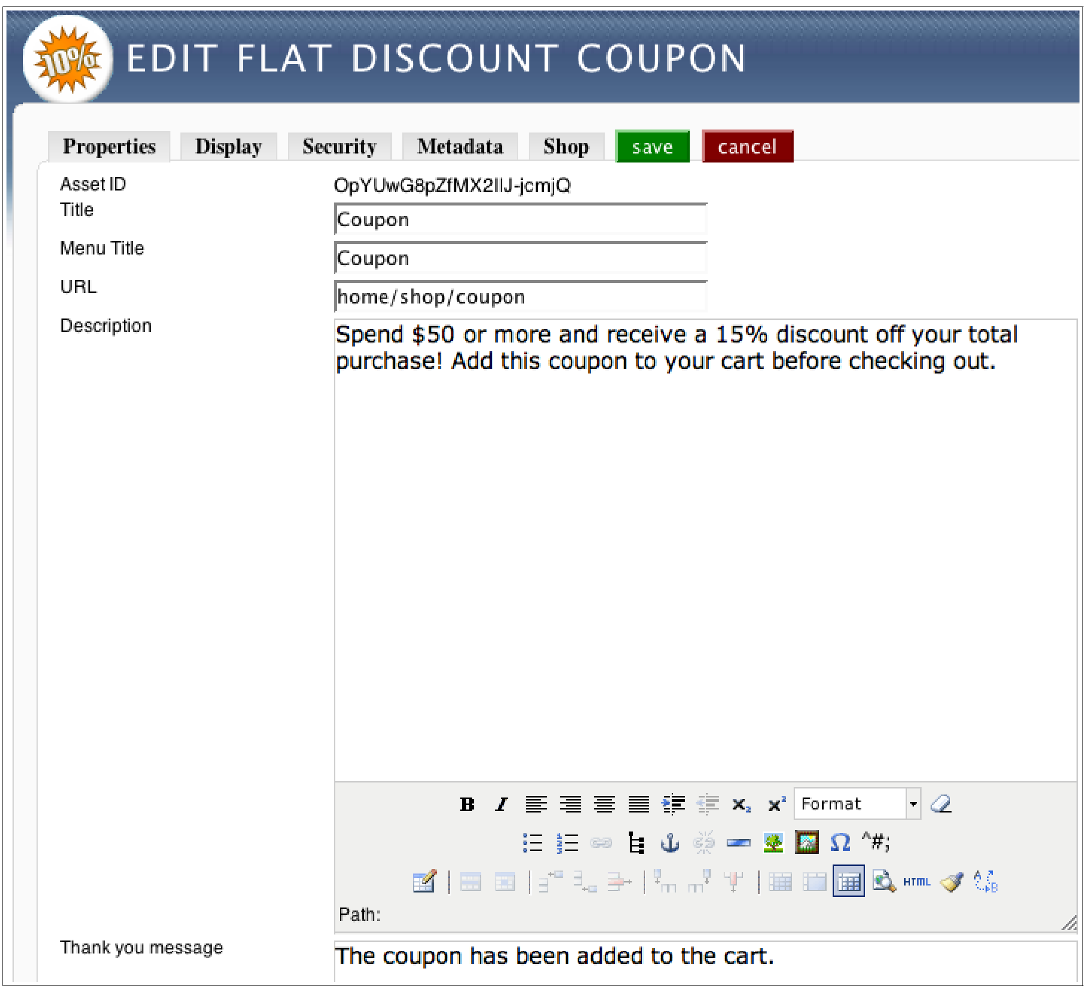


The user may proceed to the cart to check out, or click on Continue Shopping. Clicking on Continue Shopping directs the user back to the Product (or other SKU) asset that was just added to the cart.

Flat Discount Coupons

The Flat Discount Coupon allows you to place a coupon on the page that users can add directly to their carts. You specify the amount of the discount, and whether the discount will be based on a percentage or flat monetary value. The Flat Discount Coupon asset will also let you specify a base amount that must be spent before the coupon can be applied.

1. Select Flat Discount Coupon from the New Content menu to open the Add/Edit Flat Discount Coupon Screen.



2. In the “Title” field enter a title for this coupon. This could be just Coupon, or something like “Save Now” or “Special Offer.”

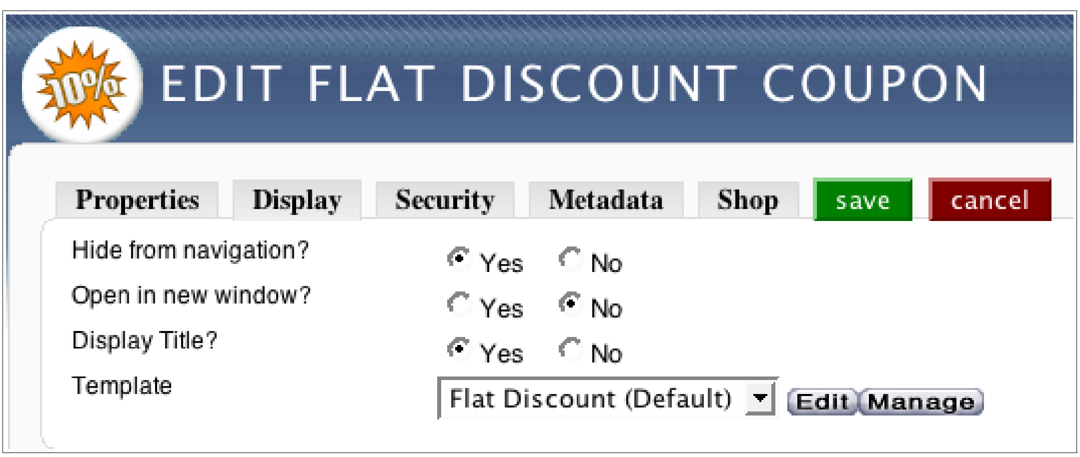
3. The “Menu Title” is the title of this asset as it appears in the site navigation. You can leave this blank and WebGUI will use the title entered in the Title field for this field.

4. The “URL” field is a field in which you can enter a URL for this asset. Again, you can leave this blank and WebGUI will create a URL for you. Because this coupon is an asset, and has its own URL, you can create hyperlinks to it. For example, you may wish to advertise this as a special promotion. You can send a promotional email out that links people to this coupon so they can take advantage of the savings.

5. In the “Description” field enter text that will display to the user. This could be an explanation of how to use this coupon, special terms of this coupon, or any other related information you want the reader to know. You can also upload an image to use with this coupon through the WebGUI Collateral Image Manager, located in the gray area of the Rich Editor.

6. The “Thank you message” field displays the message users will receive upon adding this coupon to their carts.

7. Configure the coupon's display settings in the Display tab:



A. If the “Hide from navigation?” field is set to Yes, the asset will not appear in the site navigation.

B. If the “Open in new window?” field is set to Yes, the asset will open in a new browser window if it is viewed directly.

C. The “Display Title?” field determines if the title of this asset will appear on the body of the page.

D. The “Template” field is the template used for displaying this asset. If you have a custom template to display this coupon you will select it here.

8. The Security tab is the standard security tab in which you can select the group of users who can edit this asset and view it on the site. The Owner can always do both.

9. The Metadata tab is the standard metadata tab common amongst most assets. If you want to assign metadata to this asset do so here. Like other SKU assets, you can associate a coupon with a Shelf asset using metadata keywords. This will allow you to place a coupon anywhere on the site, and have it pulled onto a Shelf asset for display with other SKU assets (such as related Product assets).

10. The Shop tab contains vital fields for configuring this coupon.

A. The “SKU” field displays the stock keeping unit for this item. The coupon is treated as an item in the cart, so therefore, a SKU is assigned. WebGUI will generate a SKU for you, or you can enter your own.

B. The “Vendor” field allows you to specify a specific vendor to get credit for selling this item.

C. If “Ships Separately?” is set to Yes, this item will accrue additional shipping costs, separate from other items in the same cart.

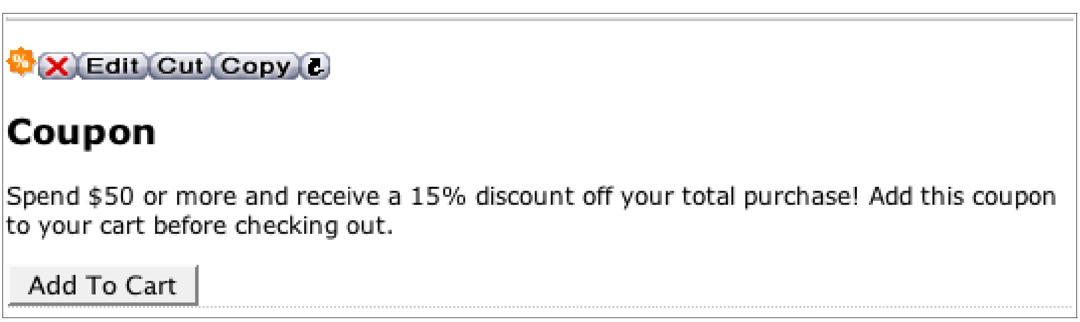
D. The “Must Spend” field allows you to set a monetary amount that must be spent in your shop before this discount will apply.

E. The next two fields, “Percentage Discount,” and “Price Discount,” allow you to set the amount this coupon is worth. If the coupon takes a percentage of the total price off the purchase, enter that percentage in the “Percentage Discount” field. If the coupon takes a flat monetary amount off the purchase, enter that amount in the “Price Discount” field.

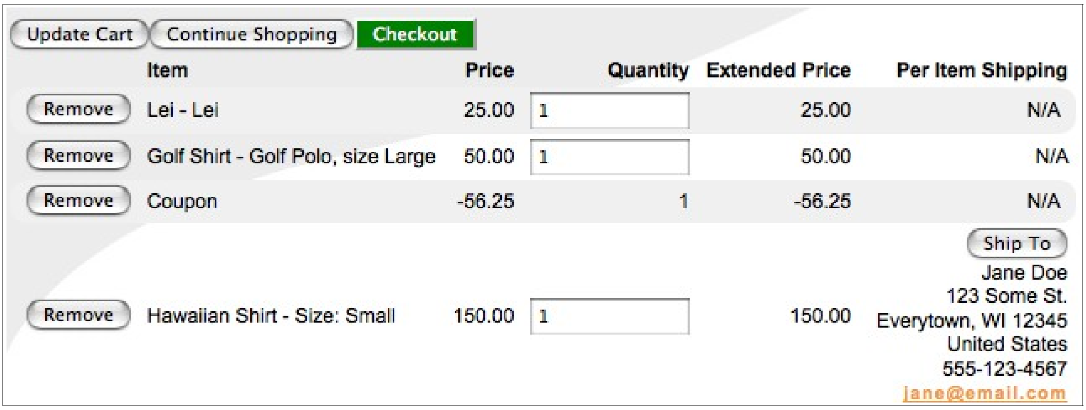
F. The “Override tax rate?” field allows you to override the default tax rate for this item. This is useful in locations where there are special tax rates, or tax is not charged, on life-essential items such as food and clothing.

G. If the “Override tax rate?” is set to Yes, enter the amount of the override in the “Tax Rate Override” field. In other words, enter the new percentage of tax to apply to this item. This probably doesn't apply to a coupon.

11. Click save to place the coupon on the page.



Users simply click the Add to Cart button to add this coupon to their carts, where the discount will be applied during checkout, as seen below.



You can take advantage of WebGUI's versioning system when using coupons to create coupons that appear on your site for a limited amount of time. For example, create a version tag in which you create a Flat Discount Coupon to give shoppers a special discount that only applies to purchases made on a single day. You may want the coupon to be only thing contained in the version tag.

Enter the Version Tag that contains this Flat Discount Coupon asset.

In the “Start Time” and “End Time” fields at the top of the screen, set the time period during which the coupon will be available for use in your Shop. WebGUI will publish the coupon at the start time indicated, and when the savings are no longer available, the coupon will be removed from the site at the end time indicated. Users will need to make sure they checkout during this time frame, or the coupon will also disappear from their carts. The “End Time” field is also a good way to impose expiration dates on coupons.

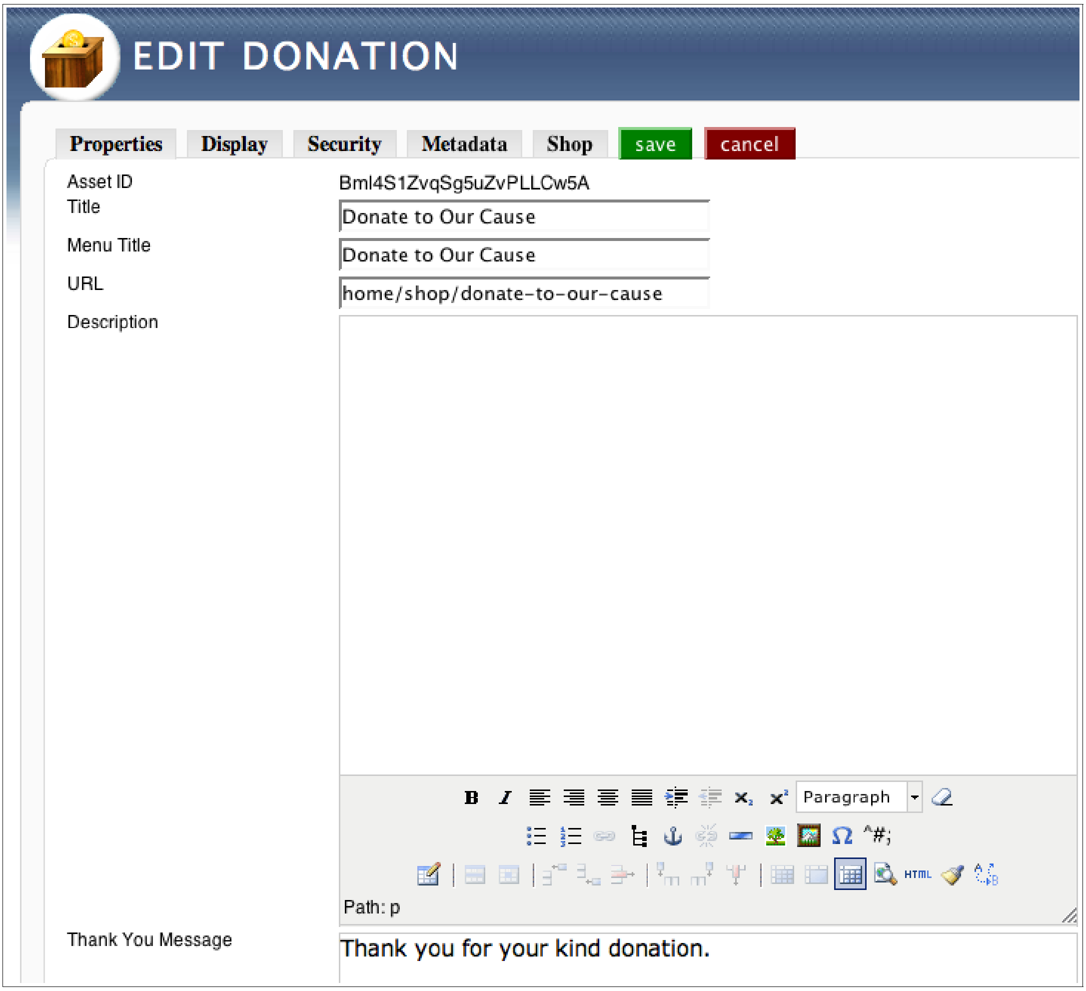
Also, remember that the Flat Discount Coupon is an asset. Therefore, it has viewing privileges just like other assets. You may want to use this to your advantage to provide special savings to a specific group of users. In the Flat Discount Coupon asset's Security tab, set the “Who can view?” field to the group of users you want to allow to use this coupon. This could be a group of users who are subscribed to a service on the site. Perhaps you'd like to offer them a discount if they renew their subscriptions. Or, you can set the who can view privileges to correspond with a group of users subscribed to your newsletter.

You can also place the coupon asset on a container asset that is hidden from the site's navigation, keeping it hidden from a normal site user's view. For example, in a Shelf or Page Layout asset's Display tab, set the “Hide from navigation?” field to Yes. Then, place a Flat Discount Coupon on the page or Shelf, and include the URL of the coupon in a newsletter to allow readers to easily access their savings, and allow only them to redeem the coupon.

Donation

Many non-profit organizations can benefit from allowing people to make online donations. The Donation asset is an easy way for users to do so, and allows the organization to suggest a default donation amount, while still allowing the user to edit that amount.

1. Select Donation from the New Content menu to open the Add/Edit Donation screen.



2. In the “Title” field, enter a title for this asset. This could be the name of the charity to donate to, or some other indication of what this donation is for.

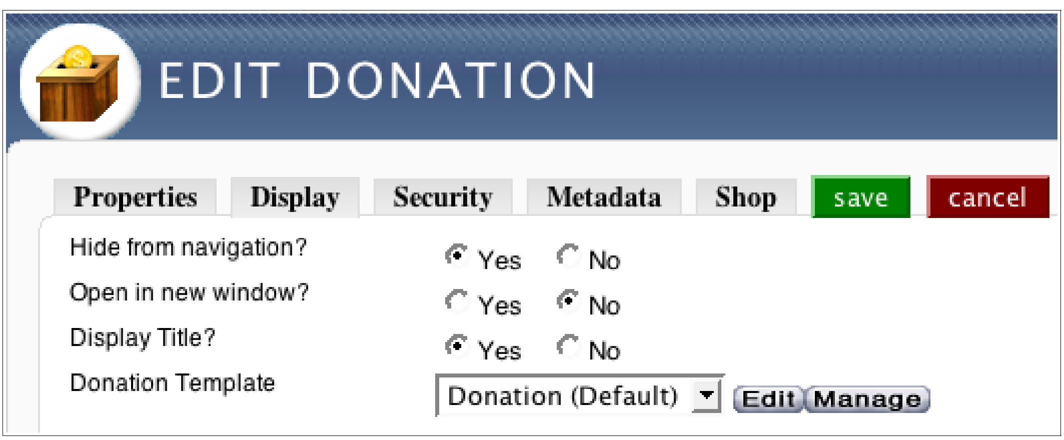
3. The “Menu Title” is the title of this donation asset as it will appear in the site navigation. You can enter a menu title, or leave it blank and WebGUI will use the title in the “Title” field for this field as well.

4. The “URL” field is the URL for this asset. You can enter a URL, or you can leave this blank and WebGUI will create one for you.

5. In the “Description” field you can enter text related to this donation. For example, you can explain why this donation is needed, direct people to additional information, or use the WebGUI Collateral Image Manager to include an image.

6. The “Thank You Message” is displayed to users once they have added their donations to their carts.

7. Set display options in the Display tab.



A. If the “Hide from navigation?” field is set to Yes, the Menu Title of this asset will not appear in the site navigation.

B. If “Open in new window?” is set to Yes, this asset will open in a new browser window when viewed directly.

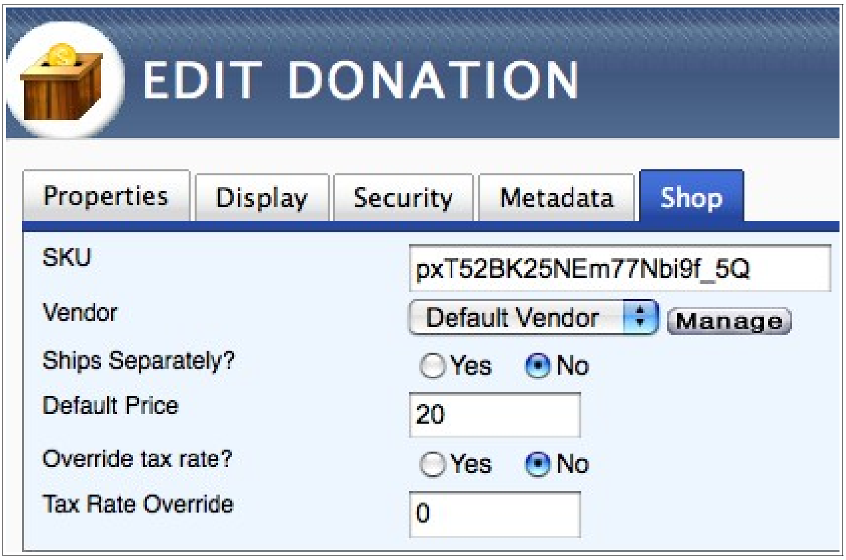
C. If “Display Title?” is set to No, the title of this asset will not appear on the page.

D. The “Donation Template” is the template used for displaying this asset. If you have a customized donation template, you would choose it here.

8. The Security tab allows you to set the group of users allowed to edit and view this asset. The Owner can always edit and view.

9. The Metadata tab is the default metadata tab common amongst most assets. If you need to add metadata to this asset do so here. Like other SKU assets, metadata keywords may be used to pull this asset from anywhere on the site into a Shelf asset with matching keyword(s) for display. See the chapter on Shelf for more information.

10. In the Shop tab, set the commerce settings for this Donation asset.



A. The SKU is the stock keeping unit for this donation asset. You can enter a specific SKU, or WebGUI will randomly generate one for you.

B. The “Vendor” field allows you to select a Vendor, as entered in the admin Shop screen, to which credit can be given for this item.

C. If “Ships Separately?” is set to Yes, this item will accrue shipping costs separate from other items in the cart. Since this is a donation, and not a physical item, this field can be left at no.

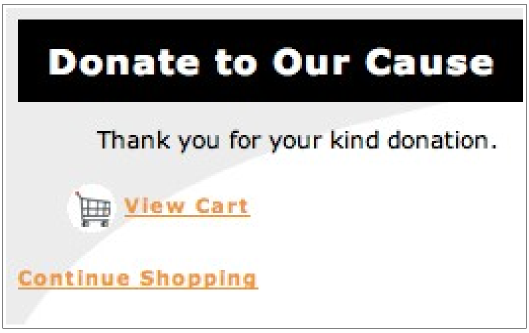
D. The “Default Price” field allows you to enter a default value that will be entered in the donation field when users view this asset. This can be used to display a suggested donation amount. Users will be able to edit this field and enter their own donation amount if they wish.

E. The “Override tax rate?” field allows to override the default tax rate for this item. In some areas, tax might not be applicable for charitable donations.

F. If the “Override tax rate?” field is set to Yes, the “Tax Rate Override” field is where you enter the new percentage used to calculate tax for this item.

11. Click save to place the donation asset on the site.

When users visit the site, they can enter the amount of their donations in the donation field. You can see that the default donation amount will already be displayed. Clicking the Add Donation To Cart button will direct the user to a screen displaying the Thank You message.



Users can click on the View Cart link to be directed to their shopping carts where they can complete the checkout process. The Continue Shopping link will direct users back to the Donation asset.

The Donation asset's functionality extends beyond donations to non-profit organizations. In fact, it doesn't have to be used for donations at all. For example, consider some of the popular online auction sites. The Donation asset provides a way for users to make arbitrary payments. If you have goods for sale via auction on your site, this asset is a good way to provide buyers a mechanism to pay you. Change the asset's Title to something more appropriate, simply “Pay for Items” would suffice, and then alter the Thank You Message to also relate to the asset's new purpose.

In addition, allowing users to make arbitrary credit card payments is helpful for businesses who complete custom projects for clients. For example, an artist may be commissioned to complete a work. Perhaps a deposit is made up front, and then full payment is required once the piece is delivered. Place the Donation asset on a private URL (a Page Layout hidden from the navigation would work) by which to access the Donation asset, and change the asset's title to something like “Pay Your Bill.” Provide the client an invoice and include the private URL on it. Then, the client can come out to the site and pay via credit card.

Subscriptions

Subscriptions are a great way to offer products or services on your site that otherwise would be unavailable to the average user. Often, subscriptions are used to provide access to online support services that need to be purchased on a cyclical basis, for example every year.

A Subscription asset could be used to provide access to special material on your site. You might run an online magazine service. The content for the magazine would be made viewable to only users who have subscribed to the magazine. The Subscription asset is how those users would purchase a subscription.

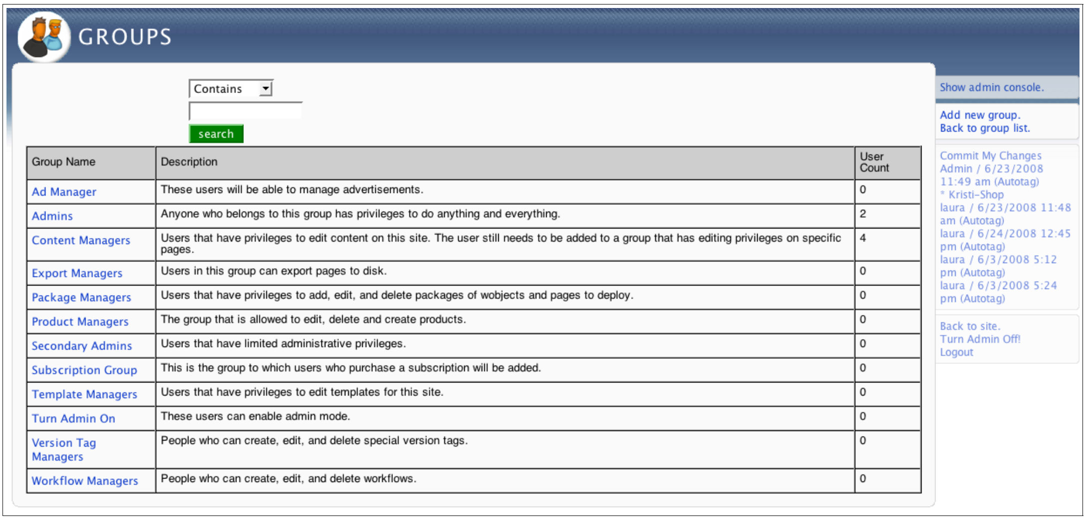
Another example is a digital download service. Perhaps you want to sell music, movies, software or other media through your site, and you want to allow users to download the media from a special area. Selling a subscription to these items allows the user to access the items and come back to the site any time during the subscription term to download more items. In a case such as software downloads, the user might also have access to the latest versions as they are released.

Setting up a subscription is a combination of Shop and Group set-up.

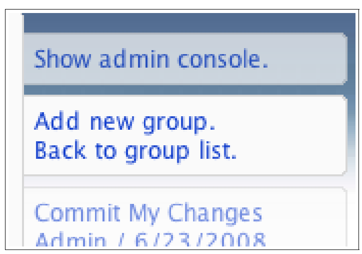
Create a Subscription Group

You will first need to establish a group subscribers will become members of upon purchasing a subscription. This is done through Groups in the Admin Console.

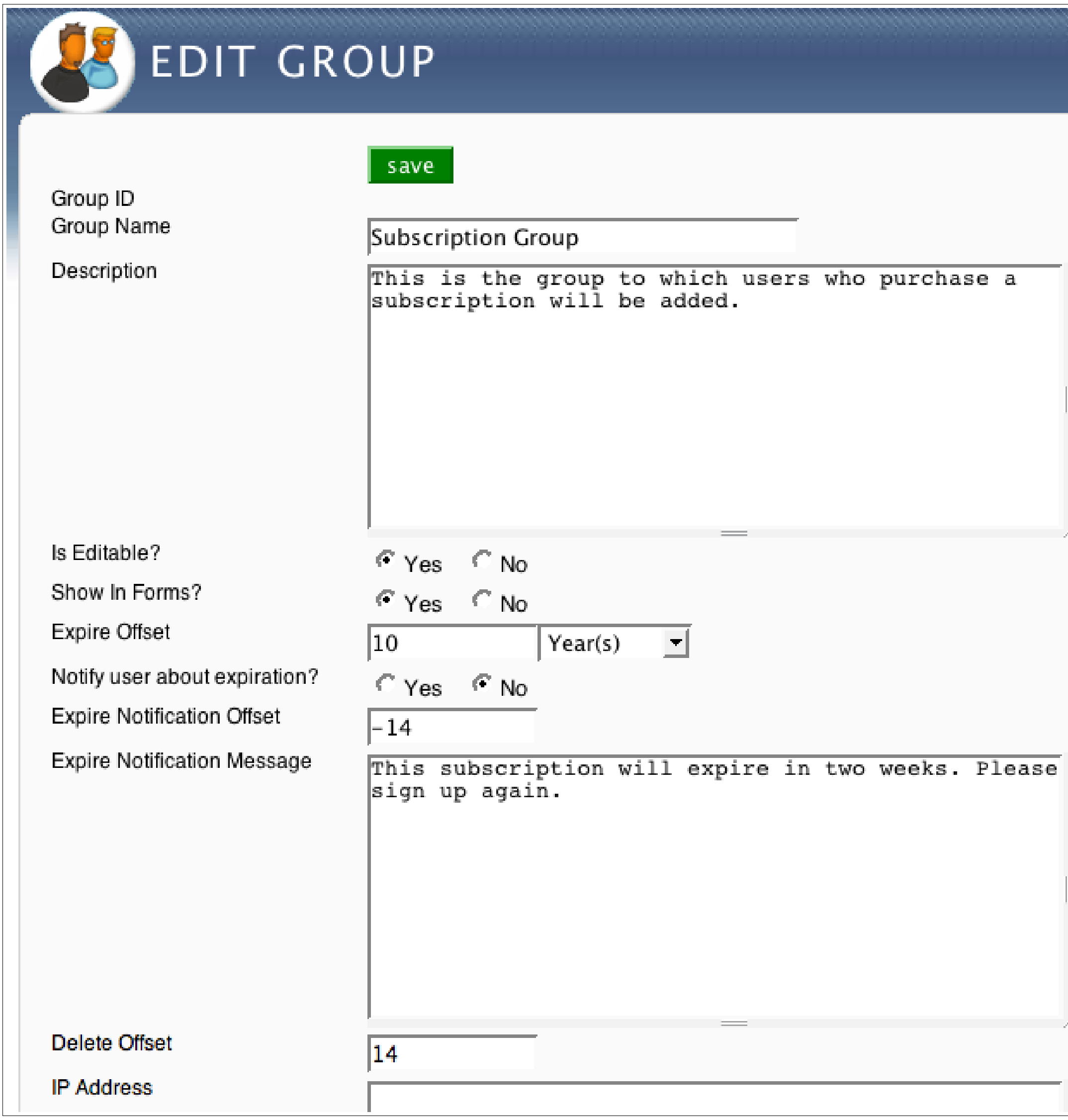
In the Admin Console, click on Groups to open the Groups screen. Upon entering the Groups screen you will see all the existing groups listed that are available to manage.



In the Groups screen, click the "Add a new group" link on the far right hand side of the screen to open the Edit Group screen.

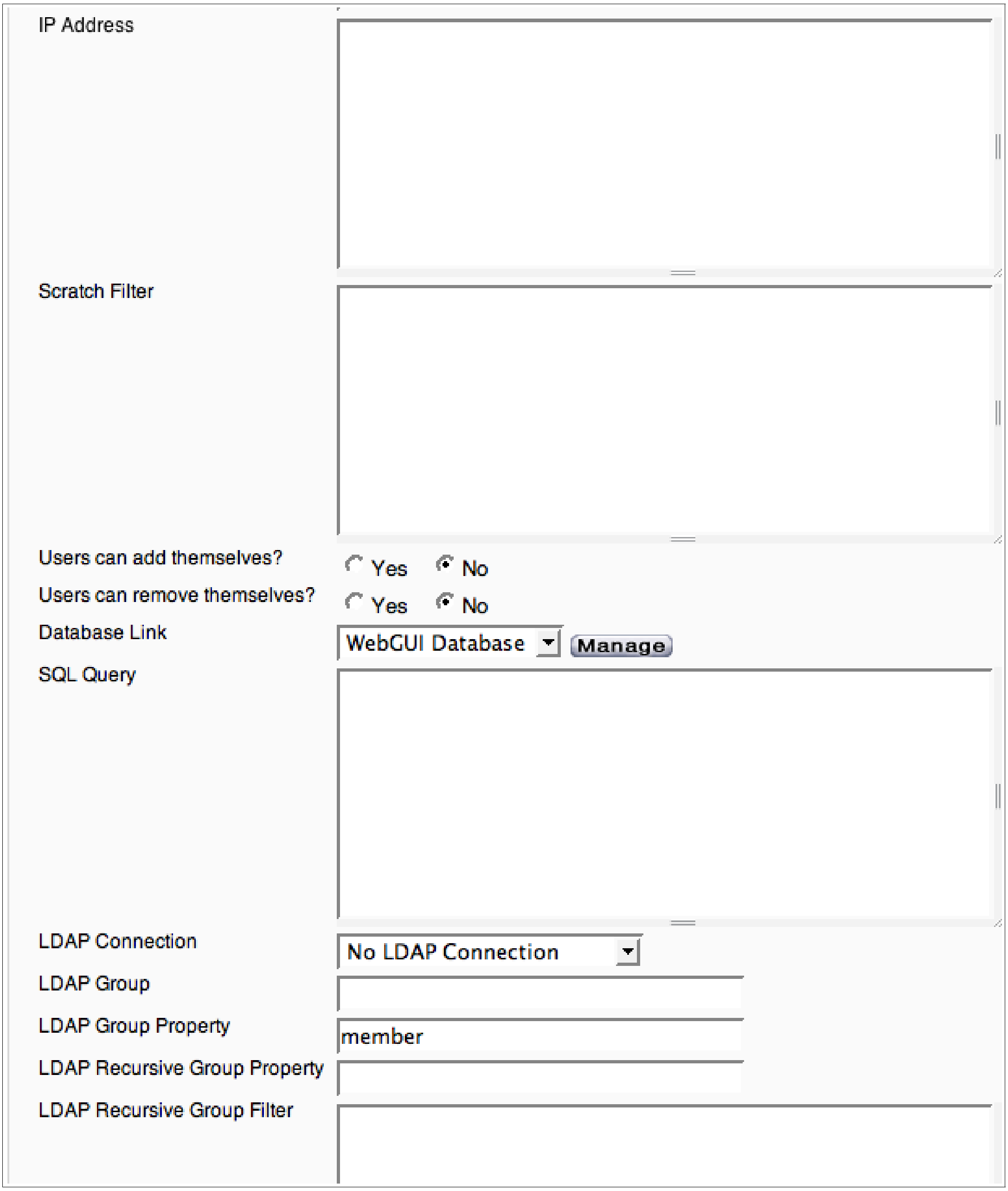


In the Edit Group screen, enter the necessary information regarding this group. This can be as simple as a name and description, or more involved depending on your needs.



- Group Name: a name or title for this group.
- Description: a description of the what this group is for.
- Is Editable?: if set to No, you will not be able to edit this group again after its creation, and it will not appear in the list of manageable groups.
- Show in Forms?: should this group show up in fields where you can select groups for the purposes of assigning privileges. For example, in an asset's Security tab for the who can edit or view fields.

- Expire Offset: the amount of time that can pass before the group access expires. If this is a group that you want to remain active for a very long time set this to a large number.
- Notify user about expiration?: set this field to Yes if you want WebGUI to notify users when they are about to be expired from this group.
- Expire Notification Offset: determines how much advance notice a group member will receive before group membership expires.
- Expire Notification Message: enter a message the user will receive via email upon group expiration.
- Delete Offset: the number of days that will pass between expiration notification and the group's actual deletion from the system.
- IP Address: if this is specified, any users visiting the site from that IP range will automatically be a member of this group. This is great for intranets where you want to grant access to some information, but only if the users are inside your network.



- Scratch Filter: users can be dynamically bound to a group by a scratch variable in their sessions.

- Users can add/remove themselves?: set if users can add or remove themselves from a group. For example, if a user clicks on a Subscribe link in a forum, the user is electing to add him/herself to that group.

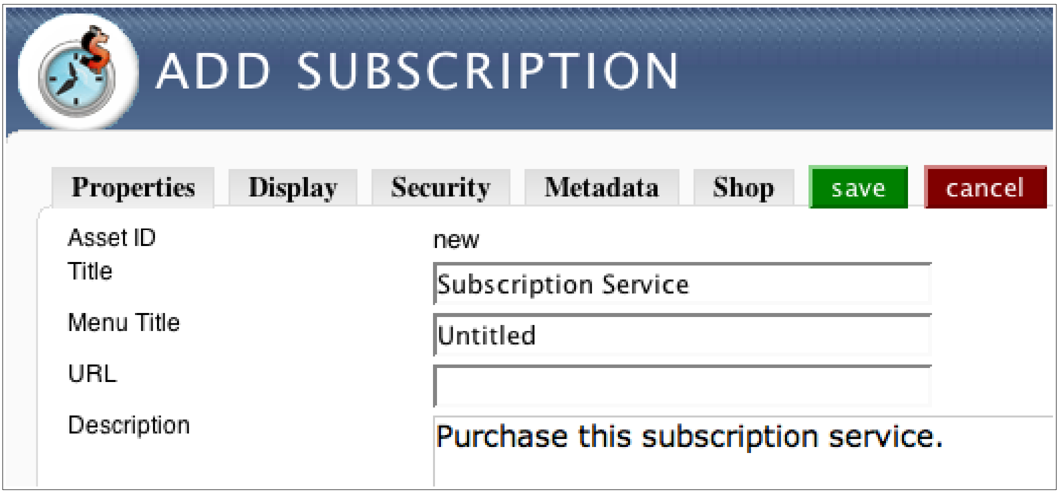
- Database Link: if you'd like to have this group validate users using an external database, select the database to use.
- SQL Query: to validate users against an external database you may construct an SQL statement that will return a list of WebGUI userids for users in the group.
- The LDAP fields can be used to bind users in a group to an existing LDAP directory.
- Cache groups for how long?: large sites using external group data will make many calls to the external database. To help reduce the load, you may select how long you'd like to cache the results of the external database query within the WebGUI database.

When the group is configured as you wish, click save to create the group. You will be returned to the Groups screen. From there, click the Back to Site link on the far right hand side of the screen to return to the site.

Create a Subscription

Subscription creation is done through the Subscription asset, located in the New Content menu.

1. Navigate to the location on the site where you would like to add the Subscription.
2. Click on Subscription in the New Content menu to open the “Add/Edit Subscription” screen.



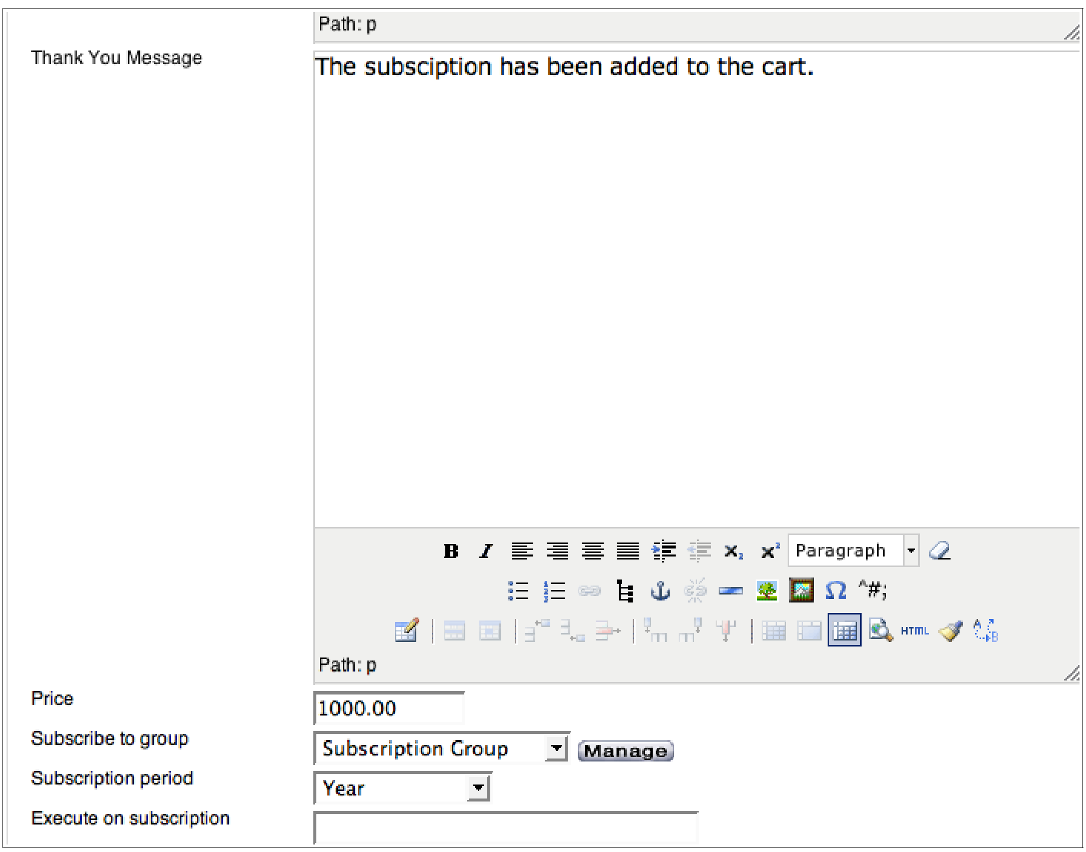
3. In the “Title” field, enter a title for this asset.

4. The “Menu Title” is the title of this asset as it appears in the site navigation. You can enter one yourself, or you can leave it blank and WebGUI will use the title entered in the “Title” field here.

5. The “URL” is the URL for this asset. You can create one, or leave this field blank and WebGUI will create one for you.

6. In the “Description” field you can enter text related to this subscription service. This could be instructions, explanation of the services provided by this subscription, contact information, or any important information you want the user to know.

7. The “Thank You Message” is the message users will see upon adding this Subscription to their shopping carts.



8. The “Subscribe to group” is the group which the user will become a member of upon purchasing this subscription. In the example above,

you can see that the Subscription Group created at the beginning of this chapter was selected. Use the dropdown menu to select a group from your system. Make sure that the “Who can view?” field in the Security tab of any asset related to this subscription corresponds to the “Subscribe to group” field in the Properties tab of this Subscription asset.

9. The “Subscription period” is the length of a single subscription term. So, how much time will pass before users must pay to renew their subscriptions? If a subscription is non-renewing (meaning the subscriber doesn't get billed on a recurring basis), then purchasing two subscriptions will cause the length of the subscription to double. Likewise, the length of the subscription will be extended if a subscriber returns and purchases a new subscription before the current subscription expires.

10. The “Execute on subscription” field is an area in which a Perl script can be entered. This script will be executed when the user has subscribed and paid. This can be used if WebGUI is being used for registration of an external system, in which case WebGUI is only the commerce system. For example, you may be selling “support” on the site, but actually conducting that through an external trouble ticketing system. You will need to give privileges to the external system, so a Perl script is run to do that. It can also be used to set up notifications to staff or trigger processes on other servers, such as a shipping system to send out merchandise associated with this subscription.

11. Set display settings in the Display tab.

A. If “Hide from navigation?” is set to Yes, the menu title of this asset will be hidden from the site's navigation.

B. If “Open in new window?” is set to Yes, this asset will open in a new browser window when viewed directly.

C. If “Display Title?” is set to No, the asset's title will not appear on the page.

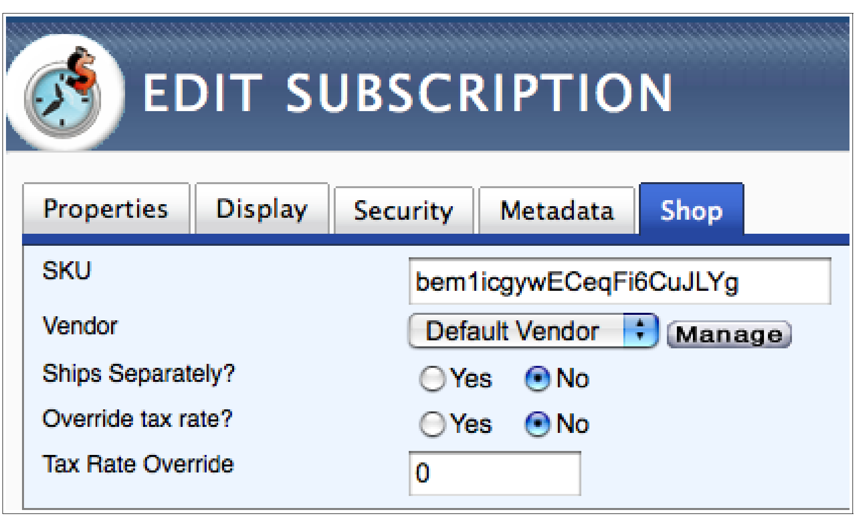
D. The “Subscription Template” is the template applied to this Subscription asset. If you have a custom Subscription template, you select it here.

E. The “Redeem Subscription template” is the template used to display the screen in which users redeem subscription codes.

12. In the Security tab you can select the group of users who can view this asset on the site. For example, if this subscription is only available to registered users, you could select that group for the “Who can view?” field. The “Who can Edit?” field is the group of users allowed to edit this specific asset. The Owner of the asset always has both editing and viewing rights.

13. The Metadata tab contains the standard metadata fields common amongst most assets. If you want to assign metadata to this subscription asset, do so here.

14. The Shop tab contains some commerce-related fields.



A. The SKU is the stock keeping unit for this Subscription asset. You can create a SKU of your own, or let WebGUI generate a random SKU for you.

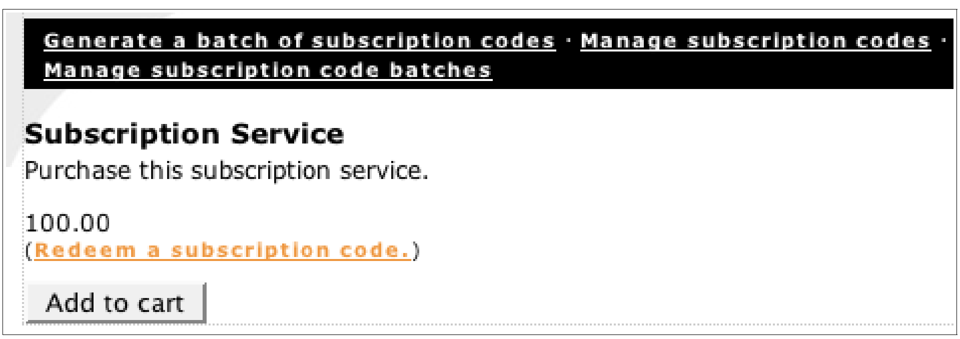
B. The "Vendor" field allows you to select a vendor who should get credit for this item. This is optional.

C. If "Ships Separately?" is set to Yes, this item will accrue shipping costs separate from other items in the cart. Since a subscription is not a physical item, you can probably leave this field set to No.

D. The "Override tax rate?" field allows you to override the default tax rate for this item.

E. If you are overriding the tax rate, enter the new tax percentage to be applied to this item in the "Tax Rate Override" field.

15. Click save to place the Subscription asset on the site.



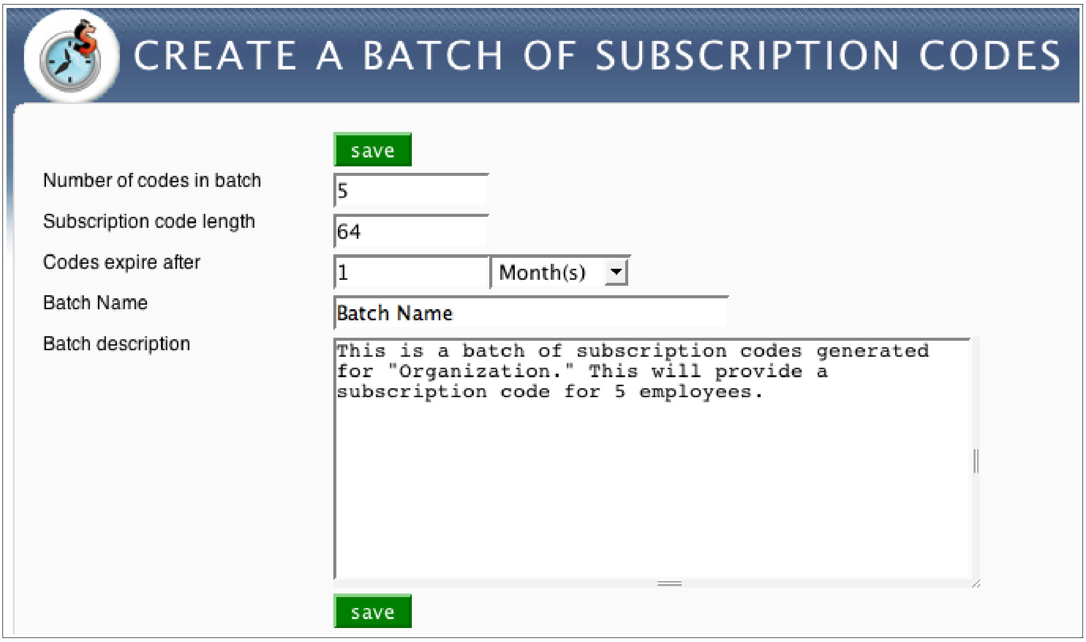
Above is the Subscription asset as seen in Admin mode. While in Admin mode, users with the appropriate privileges will see a number of links for creating and managing subscription codes. Normal site visitors will not see the black bar containing the additional links. Users purchasing this subscription click on the "Add to cart" button, and proceed to their carts to complete the purchase. You can learn more about the shopping cart in the chapter devoted to that subject later in this book.

Subscription Codes

It is possible for users to access subscription content without actually going through the purchase process. This is done by providing the user a subscription code. For example, an organization may purchase a subscription service for an entire department within that organization. Instead of purchasing a separate subscription for each person, a single purchase can be made, and you, or whoever manages the Subscription asset on your site, can generate a batch of subscription codes to provide the organization. The organization can then distribute the subscription codes, and each user within the organization can then redeem that code on your site to access the subscription.

In order to provide a user with a subscription code, you must generate a batch of subscription codes. While in Admin mode, a black bar containing a number of links will appear at the top of the Subscription asset.

Click on the “Generate a batch of subscription codes” link. This will open the Create a Batch of Subscription Codes screen.



1. In the "Number of codes in batch" field enter the number of subscription codes you need to generate. Once a code is redeemed it will become inactive, so codes can not be shared.

2. The "Subscription code length" field allows you to determine how many characters each code should contain. The default value is 64, and the minimum number of characters allowed is 10.

3. The "Codes expire after" field allows you to set an expiration for the subscription codes. So, if codes are not redeemed by the end of one month, they will no longer work. You can alter the length of time to fit your needs.

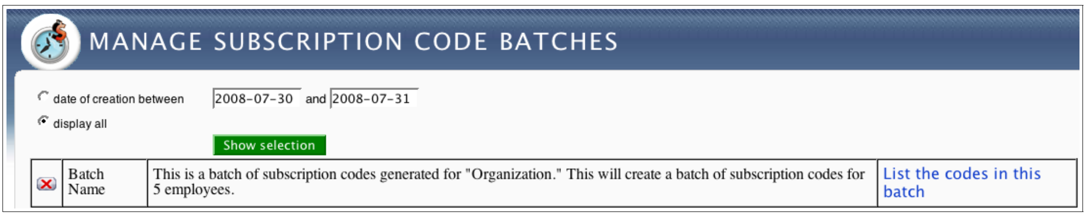
4. The "Batch Name" field is where you enter a name to assign to this batch of subscription codes.

5. The "Batch description" field is a place where you can enter notes about why this batch is being created.

6. Click save to generate the subscription codes. You will be directed to the Manage Subscription Code Batches screen.

Manage Subscription Code Batches

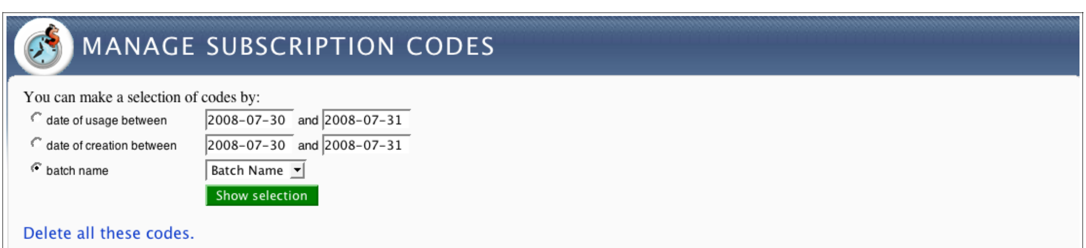
The Manage Subscription Code Batches screen allows you manage the code batches that currently exist on your site. This screen is where you will be directed upon creating a batch of subscription codes, and can also be accessed through the links available while viewing the Subscription asset while in Admin mode. A Subscription Code Batch will appear in the screen until all codes in a given batch are redeemed, expired, or manually deleted by you.



The screen displays the Batch Name and description of each subscription code batch, along with a red X by which to delete a batch. The “List the codes in this batch” link will direct you to the Manage Subscription Codes screen, where you can retrieve the codes.

Manage Subscription Codes

This screen can be accessed via the links that appear when viewing the Subscription asset in Admin mode or by clicking on “List the codes in this batch” in the Manage Subscription Code Batches screen.



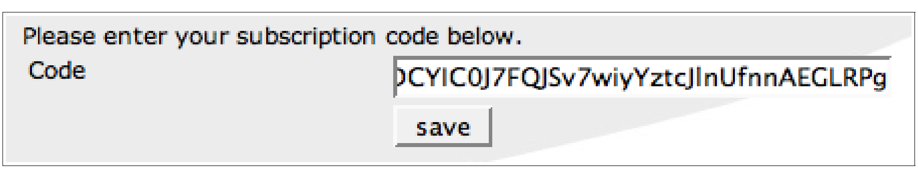
	BatchId	Code	Creation date	Date of usage	Status
vjfqupKj5MQdMVt3equKQg	SELC6g2Nodx5ZwQURM3Xndm1yhLoADCYIC0J7FQJSv7wiyYztcJlnUfnnAEGLRPg	6/26/2008 12:56 pm	7/21/2008 11:24 am	Used	
vjfqupKj5MQdMVt3equKQg	JQN6IhiOdpXtH2s50j83-HTjwA-KW5CUPqFYuOlr6CAR-FnSxx165VJM6zp7aXg8	6/26/2008 12:56 pm		Expired	
vjfqupKj5MQdMVt3equKQg	ZpFtwIw66z21WNdmaGHYSV12I4FO-nHc5z8F1k4gp7AD4p8UdmO1sSFaQEjOkXnz	6/26/2008 12:56 pm		Expired	
vjfqupKj5MQdMVt3equKQg	nS-rJ1tSnAduUCqecp9xIsgBI8LewSuV6Fym3vF6qKT5yh8YsjowwoEAprYj4wOV	6/26/2008 12:56 pm		Expired	
vjfqupKj5MQdMVt3equKQg	XuuPEhR13tuSD3fStRYUDka5hB6i3-K0-DCwOFn9IkpgHK6SEKpWA8yldpTY68lp	6/26/2008 12:56 pm		Expired	

At the top of the screen are some search fields by which to locate batch of subscription codes. You may limit the results on this screen by searching for a group of codes within a range of usage dates or creation dates, or you can select the batch of codes to display by selecting a specific batch's description from the dropdown menu. The codes displayed on the screen will reflect what you input into the search fields at the top of the screen. You can see the status of each individual code, used, unused, or expired on the far right hand side of the screen. To delete all subscription codes currently viewed on the screen, use the “Delete all of these codes.” link above the list of codes on the screen.

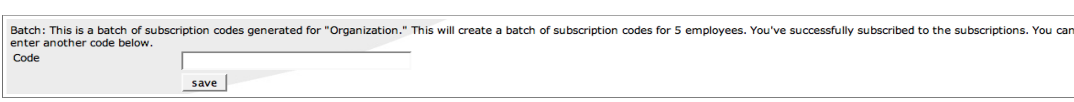
Redeem Subscription Code

Once a batch of subscription codes has been generated, a link will appear within the Subscription asset the codes correspond to.

Users click on the “Redeem a subscription code.” link to be directed to the screen in which their codes can be redeemed.



Users simply enter the subscription code into the “Code” field, click save, and they will be granted access to the site content related to this subscription. A confirmation message will be displayed upon saving.



Thingy and ThingyRecord

Thingy is WebGUI's point and click application builder. It allows you to easily build complex data forms without any programming knowledge. Thingy's possibilities are endless, and this chapter will give some examples of how you might use it.

In addition, Shop contains a SKU asset called ThingyRecord, which allows you to sell records in a Thing. Using ThingyRecord, you can build things like an online classifieds section where people can purchase a record, or classified ad, on your site. This example will be illustrated in this chapter.

Since ThingyRecord requires that you understand how to use Thingy first, both assets are covered here.

Thingy

Thingy allows you to build complex web-based applications without any knowledge of Perl, HTML, CSS, or Javascript. Just point and click! Thingy is useful for building employee directories, contact lists, inventory databases and more.

Creating a Thingy is a multi-step process. First, you create the Thingy asset. Then, you add “things” inside the asset to develop the application you want. Multiple “things” can be created and cross referenced to link data stored within things to each other.

To create a Thingy:

1. Click on Thingy in the New Content menu to open the Add/Edit Thingy screen.

2. Give the Thingy a title in the “Title” field.

3. The menu title is the title of this Thingy asset as it appears in the site navigation. If left blank, the title entered in the “Title” field will be used; otherwise, specify a different menu title here.

4. The URL is the URL of this specific asset. You can specify a URL, or you can leave it blank and WebGUI will make one for you.

5. The “Description” field can be used to enter content that will display as static content on the page to the application user. This can be something that describes the purpose of the application you are building, or provides instructions for how to use it.

6. Set display settings in the Display tab.

A. **Hide from navigation?**: if set to Yes, this asset will be hidden in the site's navigation.

B. **Open in new window?**: if set to yes, this asset will open in a new browser window when independently viewed.

C. **Display the title?**: if set to No, the title of this asset will not appear on the page.

D. **Style Template**: this is a style that the asset is displayed in; if the asset is viewed on a container asset, such as a page layout or folder, the container asset's style will override this.

E. **Printable Style**: the template used for the printable version of this asset.

F. **Thingy Template**: the template that is used for this specific asset.

G. **Default Thing**: the thing that is displayed by default upon entering the application. This can be a screen for entering data, or for searching existing data. This field will remain empty until you create things, so may need to come back and edit it later.

7. The Security tab contains the standard security settings. You can select an Owner of the asset, and select the group to edit and group

to view.

8. The Metadata tab is the standard Metadata screen common amongst all assets. If you need to handle metadata for the purposes of content profiling you can do so here.

9. Click save to create a Thingy asset.

Upon initially creating the Thingy asset, two options will appear. To create a thing(s) within this asset, click on the Add Thing link below the asset title. To create an application, you need to create a series of things from which the application is built. A thing can be anything from a simple text field, to an entire contact form, to something containing assets and templates.... the sky is the limit. Once you have created these things, you can then manage them.

The “Back to Thingy default view” link will return you to the default view of this Thingy asset, based on the Default Thing field in the Thingy's Display tab.

Add Things

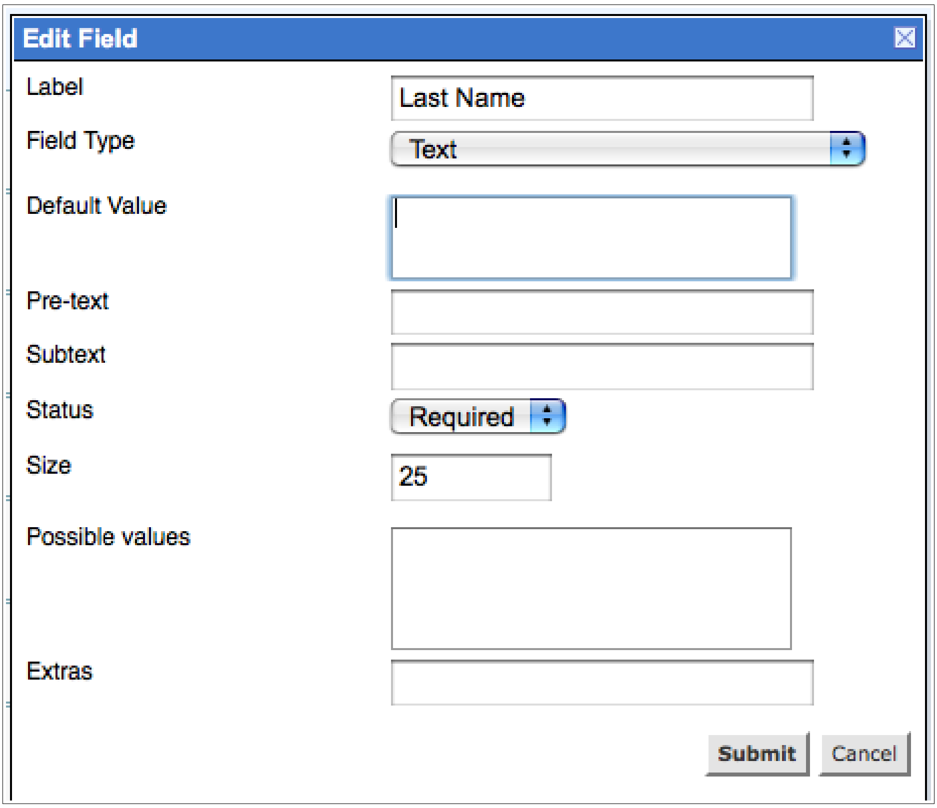
Things need to be added to the Thingy asset. These are the “things” from which your web application is built. The type of thing(s) you build is completely dependent upon the purpose of your application.

1. Click the Add Thing link located below the Thingy asset's title. This will open the Edit/Add Thing screen. The Fields tab is the default view of this screen, and is where you create the fields the user will see and, if necessary, complete. Once a Thingy asset is populated with things, it may be necessary to enter the Manage Things screen in order to access the Add Thing link.

2. Give this thing a title in the “Thing Name” field.

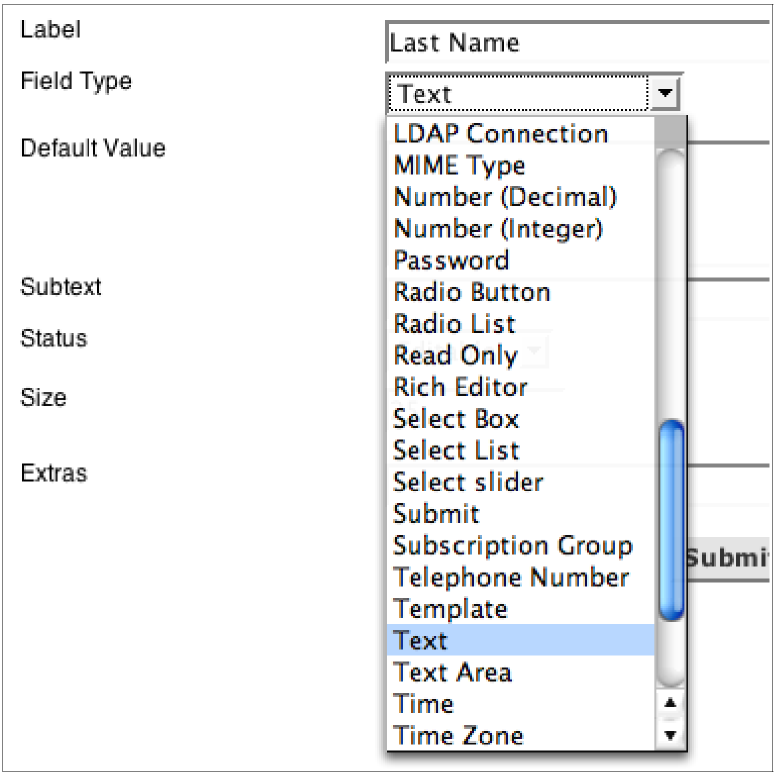
3. Click on the Add Field button to begin adding fields to this thing. In this example, a form is being made to collect employee contact information, so information such as employee name, email address, and telephone number need to be included.

4. Upon clicking on the Add Field button, the Add Field screen will pop up. On this screen you configure the field you are adding.



A. The "Label" field is the label for this field as the user is going to see it displayed on the page.

B. From the dropdown menu, select the "Field Type," which is the form element for this field. It will also be used to validate that the user has input the correct type of data if necessary. This is a lengthy list, so be sure to look it over before selecting. Some of the selections are unique, such as the ability to select an asset or workflow. Once other things are created, you will find that they will appear in this list as well. In this way, things can be contained within other things and data can be cross referenced.



C. For any type of field in which the user is making a selection, such as Yes/No or select box, enter the default value for the field in the "Default Value" field. For example, if you want a Yes/No field to be set to Yes by default, enter Yes in this field. If you select a field type such as Select Box, Select List or Radio List you will need to enter the possible values.

You can fill in the "Possible Values" field in a normal manner, such as:

Nothing
50 Percent
Full Amount

Or, you can do key/value pairs:

0|Nothing
50|50 Percent
100|Full Amount

An example of this using hex color codes would look like:

#ff0000|Red
#00ff00|Green
#0000ff|Blue

You can also do the value/label pairs like this:

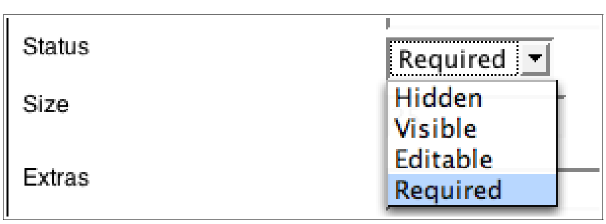
0|American Indian/Alaskan Native
1|Asian
2|Black or African American
3|Native Hawaiian/Other Pacific Islander
4|White
5|American Indian/Alaskan Native and White
6|Asian and White
7|Black/African American and White

When using key/value pairs, the first value (before the pipe) is what gets stored in the database, but the second value (after the pipe) is what gets displayed to the user.

D. Content entered in the “Pre-text” field will be displayed above/before the field. It is useful for providing instructions to the user about what should be entered.

E. The “Subtext” field is an extension of the label. You can use this to give additional information or direction for what should be entered in the field you are creating. This text will appear under/after the field.

F. Set the “Status” of this field:



- Hidden: hidden fields are not visible to the user. Make sure you enter a default value for this field.
- Visible: can be seen by the user, but not modified. Make sure you enter a default value for this field.

- Editable: can be seen by the user and can be filled out by the user.
- Required: must be filled out by the user.

F. The “Size” field allows you to set the width of the field by indicating how many characters this field would contain on average.

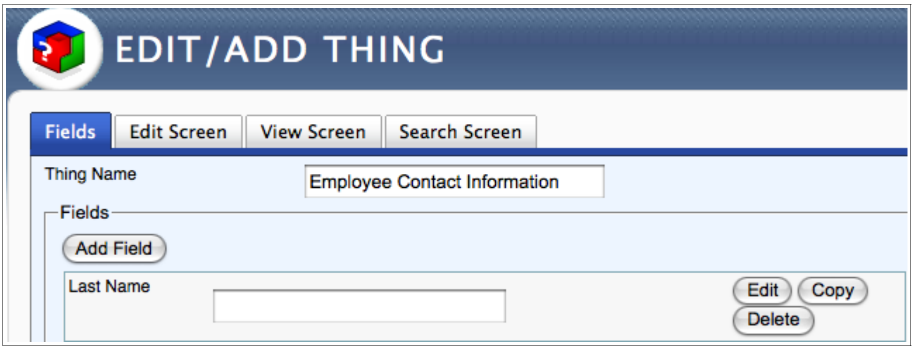
G. The “Possible values” field is used for list types. Enter the values you want to appear, one per line. If you want a different label, other than the value entered in this field, the possible values list should be formatted as:

```
{
    “key1”=>”value1”,
    “key2”=>”value2”,
    “Key3”=>”value3”
    ...
}
```

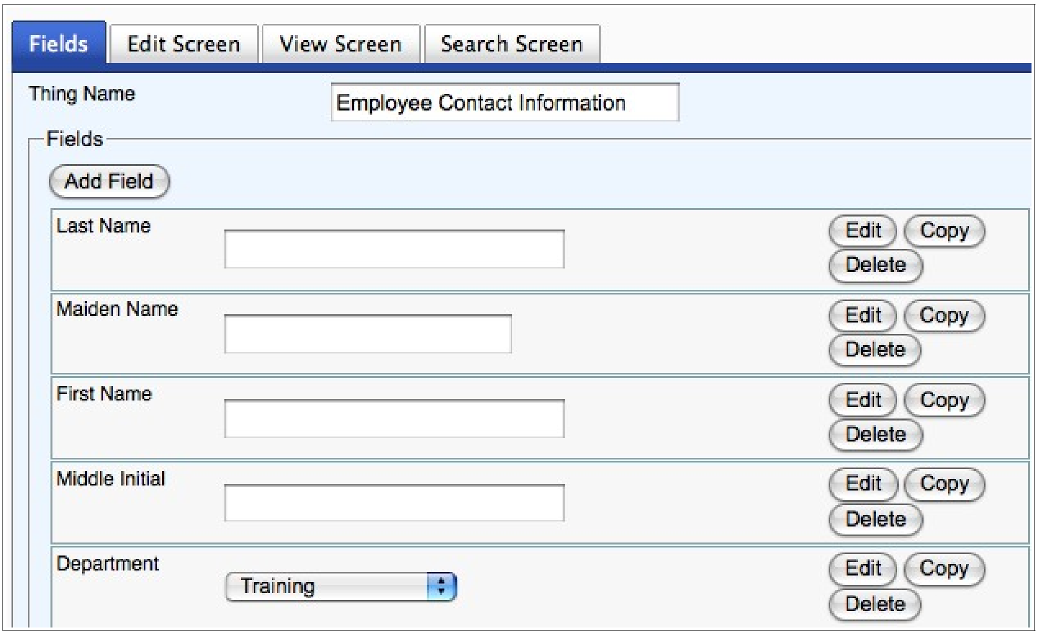
Make sure you use the braced, quotes etc. Simply replace “key1” and “value1” with your own name/value pairs.

H. The “Extras” field is available to enter additional tag properties for the field tag. For example, 'class=”myClass”'.

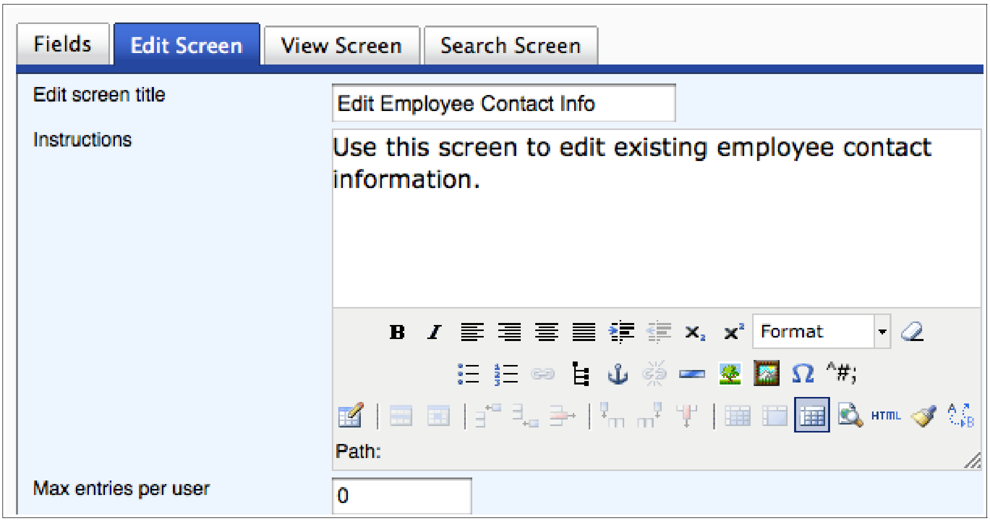
I. When all fields are complete, click the Submit button.



The field will appear on the Edit/Add Thing screen, and it will have buttons to either edit, delete or copy. Click the Add Field button to continue adding fields for this thing.



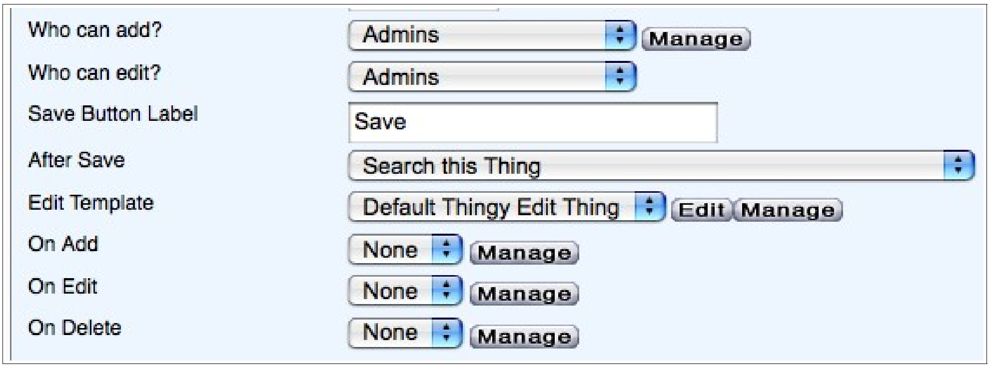
5. The Edit Screen tab allows you to customize the appearance of the edit screen for this thing.



A. Give the edit screen a title in the "Edit screen title" field.

B. The "Instructions" field can be used to enter instructions for users using this screen.

C. In the “Max entries per user” field indicate the maximum number of entries, of this thing, a user can add.



D. The “Who can add?” field allows you to select the group of users who are allowed to add this thing.

E. The “Who can edit?” field allows you to determine the group of users who will be allowed to use this edit screen.

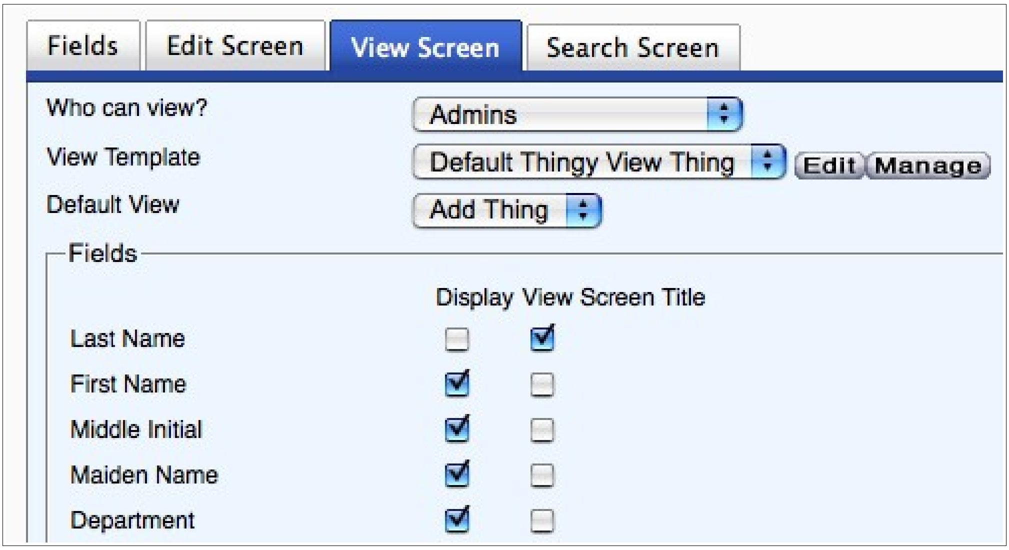
F. The “Save Button Label” allows you to determine what the form submit button should say; this is the label that users will see on the screen.

G. The “After Save” field allows you to select the screen shown to the user after saving/submitting this thing.

H. The “Edit Template” is the template used for this edit screen.

I. The On Add, On Edit, and On Delete fields allow you to select a workflow to be triggered once each of these actions takes place. For example, upon adding a new employee, you may want to trigger a workflow that automatically emails a colleague alerting him/her of this action.

6. The View Screen tab allows you to customize the view of records contained in this thing. This is the form used to display the information that is input in the Edit Screen for this thing. So, if a user performs a search in this thing, this is the screen used to display/view data related to a specific search result.



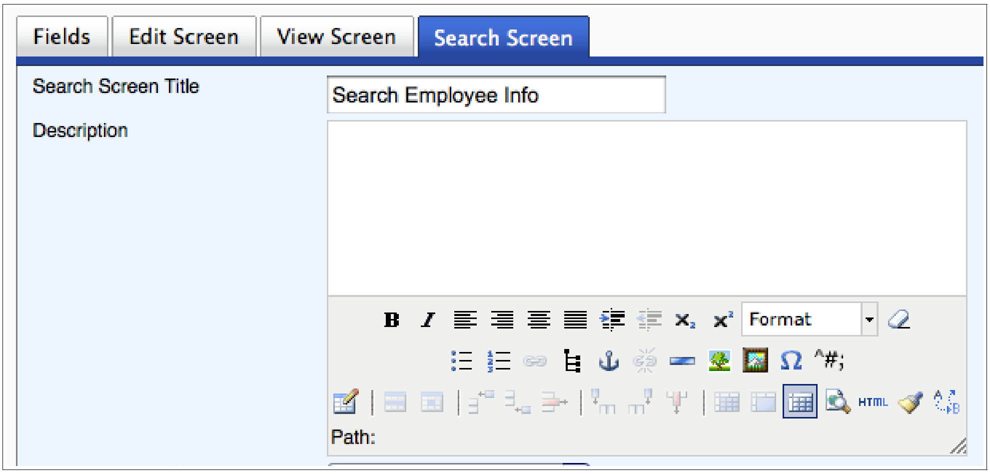
A. The "Who can view?" field allows you to select the group of users allowed to view this thing.

B. The "View Template" is the template used to display this thing.

C. The "Default View" field allows you to select the default view of this thing. You can select Search or Add Thing. This is the screen users will see upon entering the application.

D. The Fields area allows you to select which fields (created in the Fields tab by you) you want displayed in the default view of this thing. The "View Screen Title" will concentrate all the fields that you select into a list, with each field labels separated by a comma. It will then place that string into a template variable called viewScreenTitle that is available on the view screen.

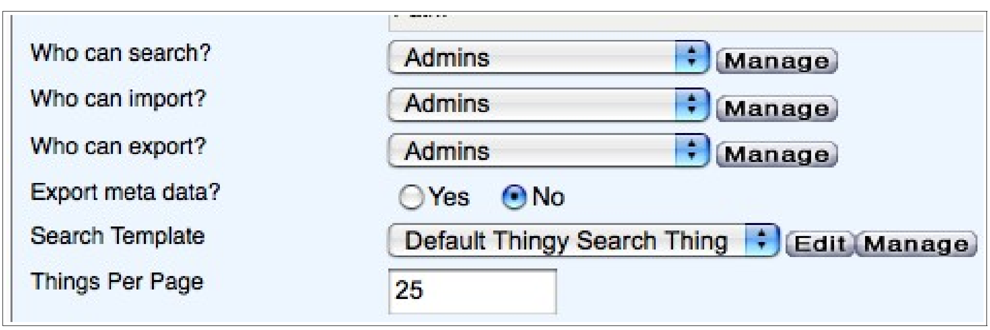
7. The Search Screen tab allows you to customize the criteria by which this thing can be searched.



A. Give the search screen a title in the “Search Screen Title” field.

B. In the “Description” field you can enter information or instructions regarding this search screen.

C. The “Who can search?” field allows you to select the group of users allowed to search this thing.



D. The “Who can import?” field allows you to select the group of users allowed to import data from an external database into this thing.

E. The “Who can export?” field allows you to select the group of users allowed to export data from this thing to an external database.

F. If “Export meta data?” is set to Yes, metadata fields will be included in the export (thingDataId, lastUpdated, ipAddress etc).

G. The "Search Template" is the template used to display the search screen for this thing.

H. The "Things Per Page" field allows you to determine how many things will be displayed on the page in the search results before pagination.

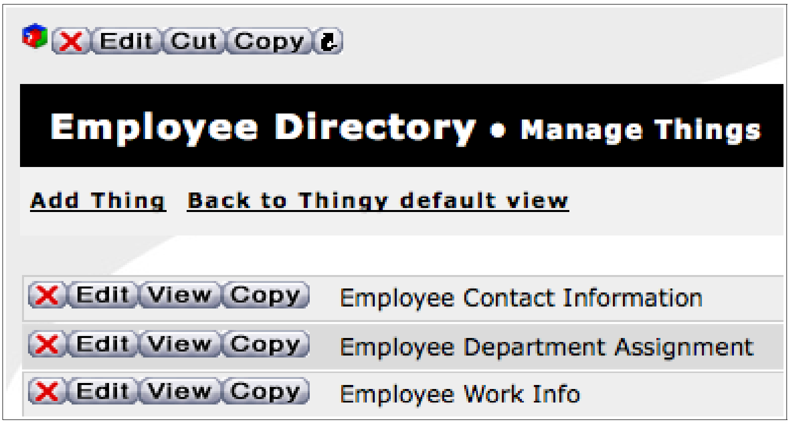
I. The "Fields" field allows you to select what fields users are allowed to search this thing by, and what fields will be displayed in the search results for this thing. It also allows you to determine the criteria by which search results will be organized.

8. When all tabs of the Edit/Add Thing screen are complete, click the save button at the top of the screen to save your new thing. Click Add Thing to create a new thing and repeat the steps above until you have created as many things as necessary.

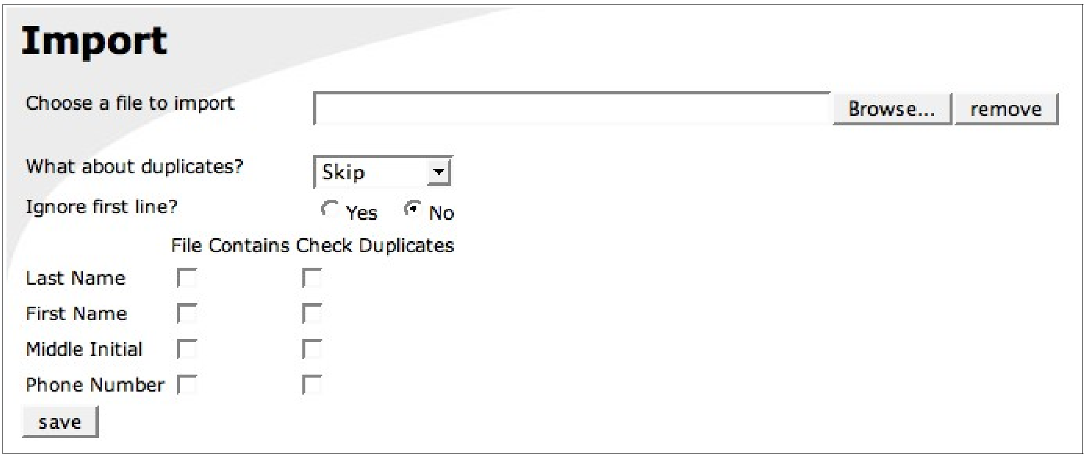
Upon entering the Thingy asset, users will see the screen that you set as the default. In this example, the Search screen for this thing.

Users with the appropriate privileges will also see some additional links. These will differ among users depending upon individual privileges, and depending upon the things you have built.

- Manage Things: takes you to a screen on which all things contained within the Thingy asset are displayed. Each thing has an asset toolbar from which to manage it. Also in this screen is a link from which additional things can be added. This is the Add Thing link located below this screen's title.



- Import: allows you to import data into the thing.



The user can browse his/her workstation and upload a .csv file. The fields near the bottom of this screen are the fields contained in this thing, and the user may designate which fields are included in the data being imported, and determine what to do with duplicate data.

- After a search a link will be made available from which to export data contained in this thing. Clicking this link will generate a csv file for download.
- A link is also available to direct users to the edit screen for this thing; in this example, the Add/Edit Employee Contact Info screen.

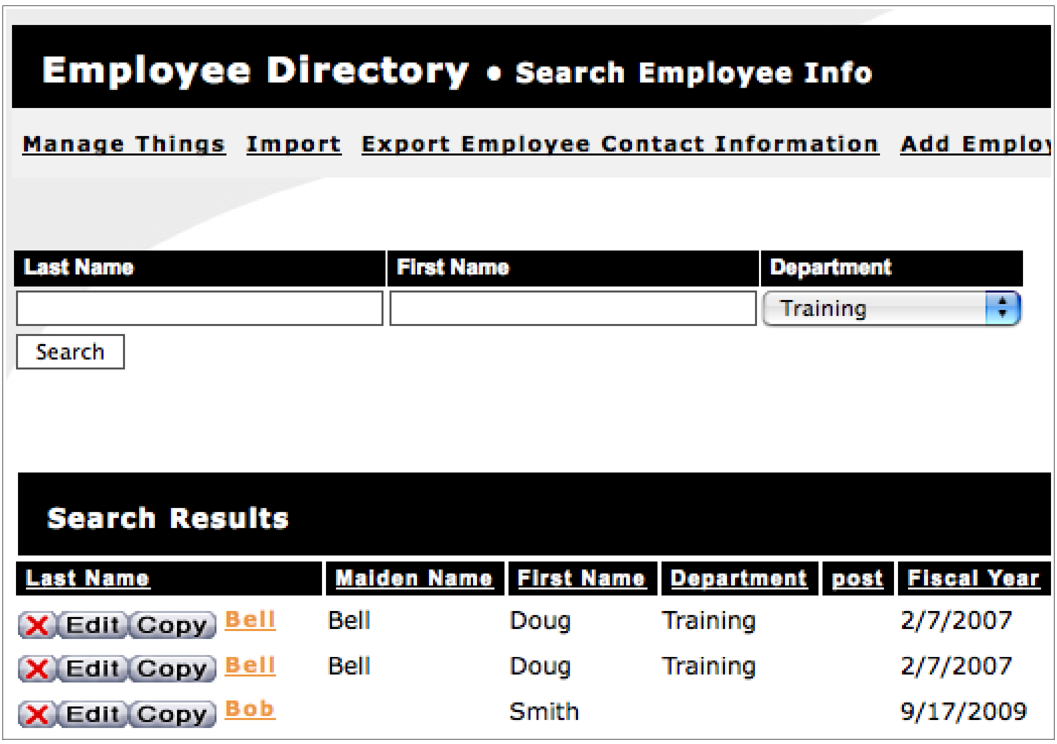
- The the right of these links is a faded dropdown menu. You can use this dropdown menu to select a different thing to view.

In this example, users can perform a search upon entering the application.

If data needs to be input, the user can select the “Add Employee Contact Information” link from the links at the top. This will open the Edit Employee Contact Info screen.

In this screen users can add an employee to the database. Fields with an asterisk are required. Upon saving, the user is directed back to the default thing, which in this case is the search.

To perform a search, users simply enter search criteria in the available search fields. The search is a limiting search. Leaving the search fields blank and clicking on Search will return all data, in this case all employees, by default. Searching for last name Doe will return all users with the last name Doe; results will be further limited if a first name is included. Search returns will be displayed directly below the search fields.

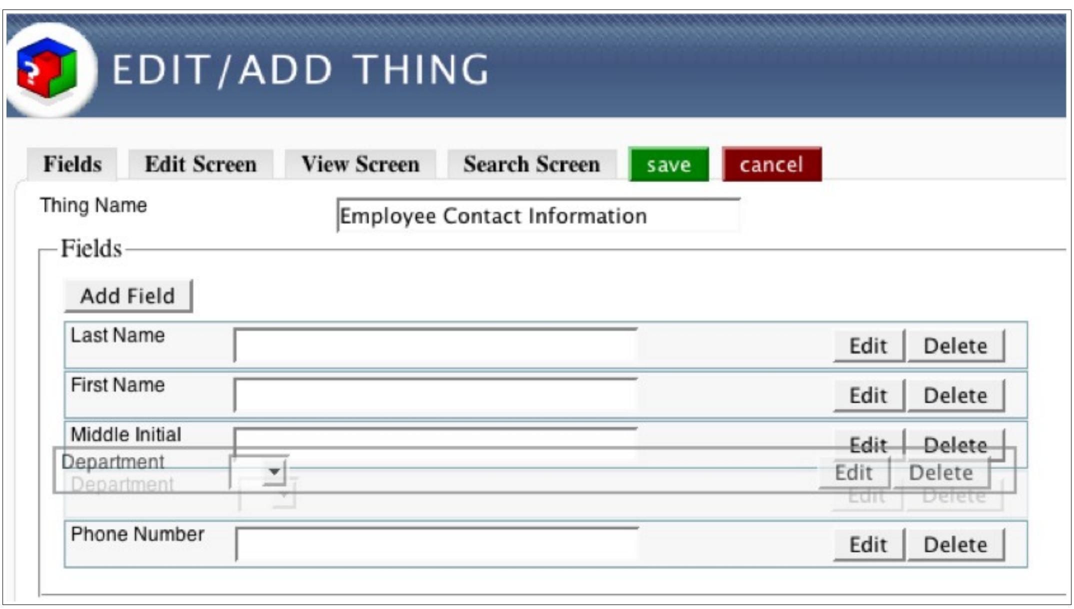


Search results will be sorted by the criteria designated when the search screen was configured for this thing, in this case, alphabetical. To sort by other criteria, such as first name or telephone number, click on the title of that column. Each search result has a red X by which a result can be deleted. The user will be prompted to confirm that this delete should take place. Each result also has an edit button. Clicking on the edit button will direct the user to the edit screen for this thing; in this example, the Edit Employee Contact Info screen. The Copy button will allow an individual record to be copied.

Clicking on the matching search criteria of a specific result, in this case the last name Doe, will open the view screen displaying just that result's data.

Rearrange Fields in a Thing

The field display order within a Thing is easily rearranged. While in the Edit/Add Thing screen, simply grab a field with your mouse, drag it to a new position, and drop it in place. In the image below, the Department field is being moved.



Examples

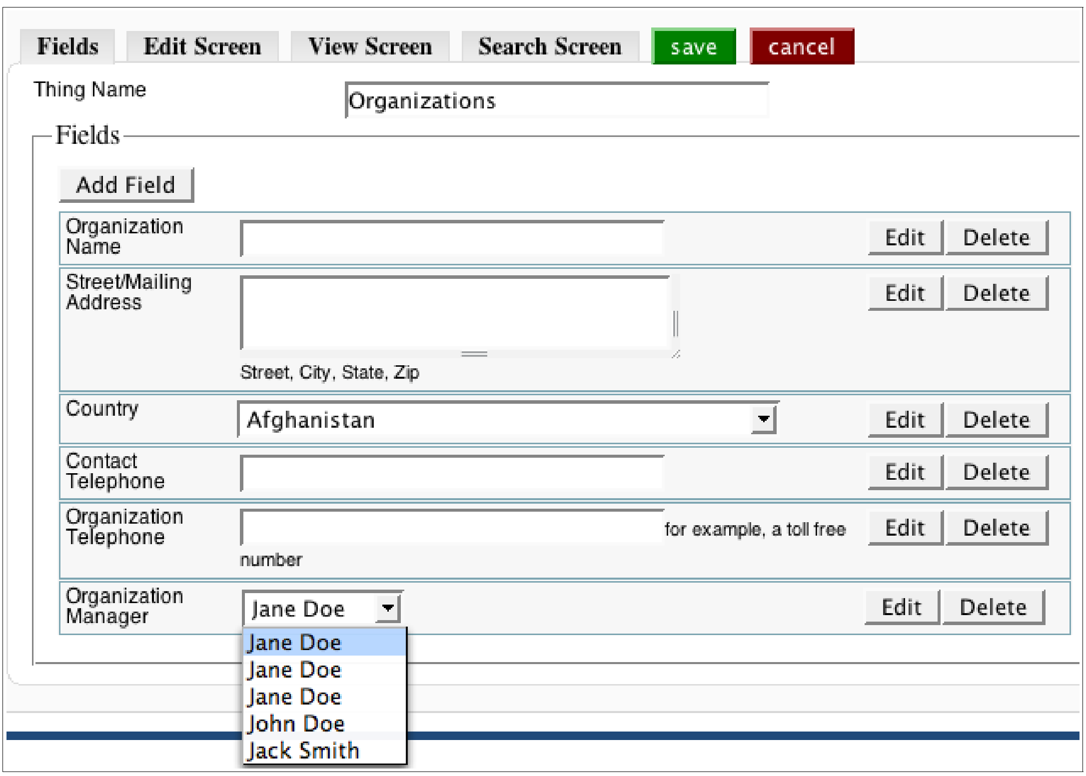
Here are some samples to give you some ideas. These are fairly simple examples, so you will want to take your organization's specific needs into consideration and experiment to see what best meets those needs.

Rolodex

Try using Thingy for a rolodex type of application. In this example, two things were added to the Thingy asset: one to enter organizations, and one to enter members of organizations. The two things were then cross-linked so when an organization is added a member may be selected as that organization's contact, and when a member is created, s/he may be added to an existing organization.

First Thing: Organizations

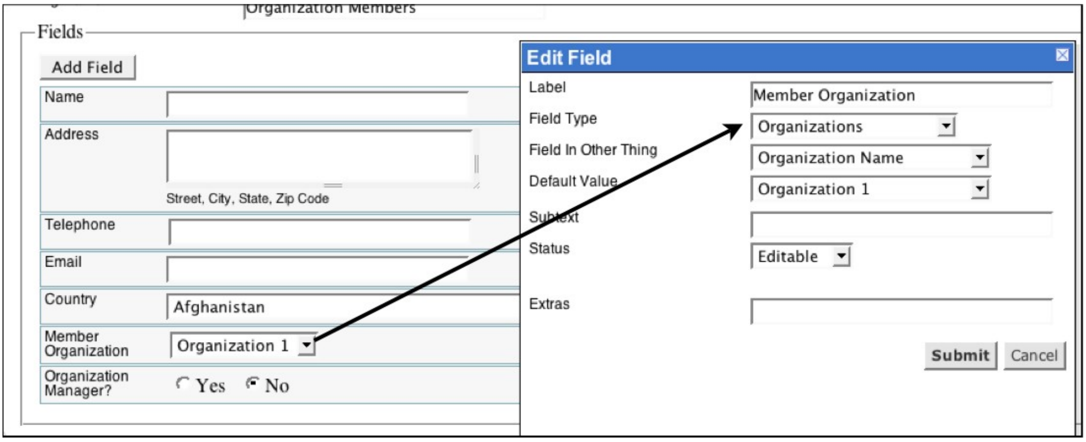
The Fields tab contains the basic contact information for the organization. The Organization Manager field is populated by selecting a person from the Organization Members database (next Thing in the Thingy asset). The Organization Manager field was created by selecting the Organization Members thing for the field type.



This thing can be searched by organization name.

Second Thing: Organization Members

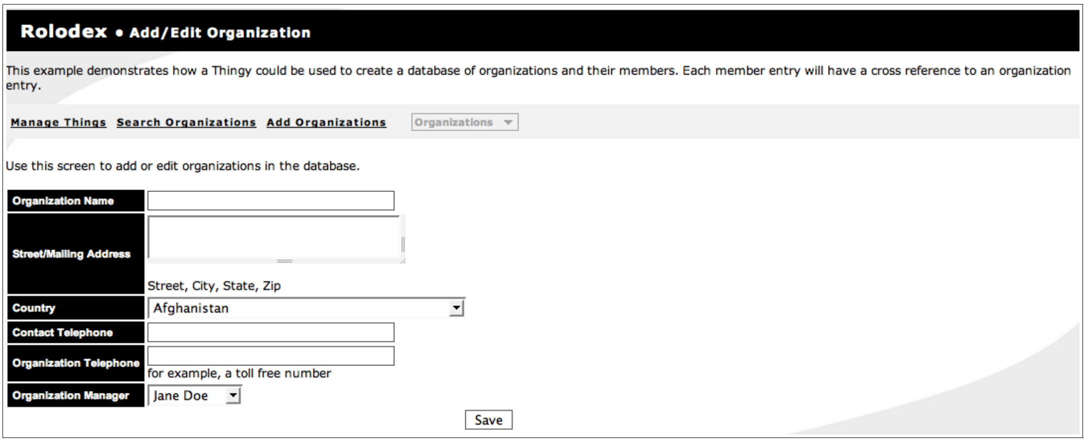
The Organization Members Fields tab contains members' contact information and organization affiliation. The Member Organization field was created by selecting the Organizations thing as the field type.



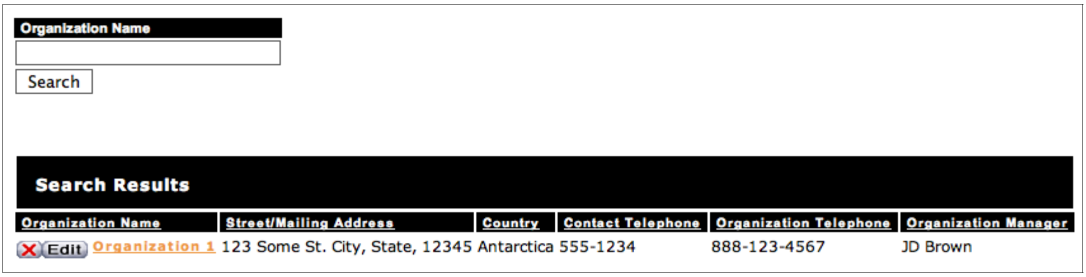
This thing was then configured to be searched by either the member's name and/or organization.

Results

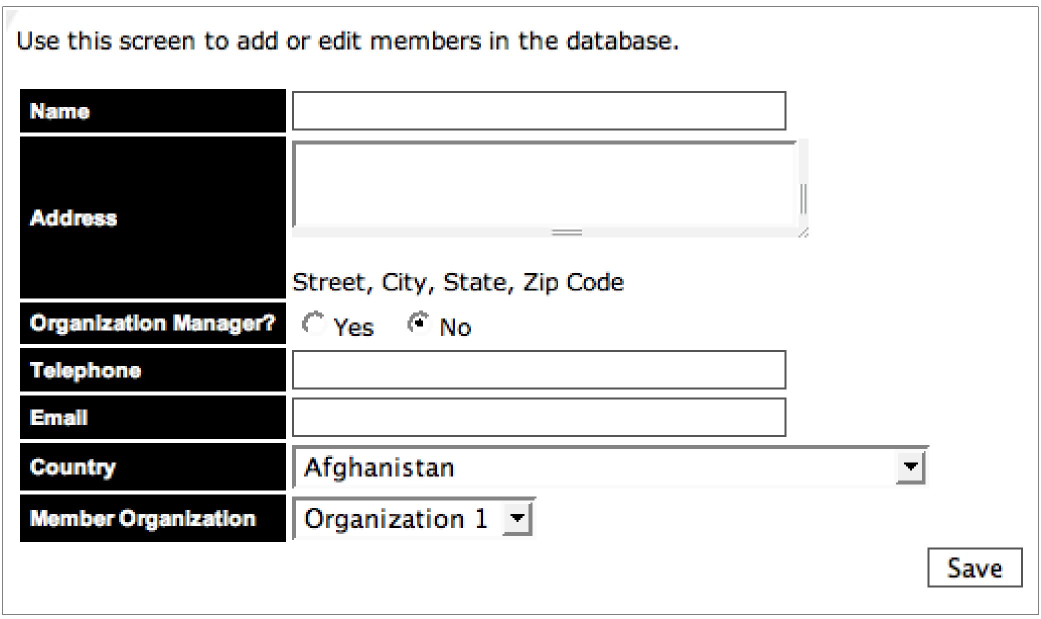
The Add/Edit Organization screen contains the contact information for the organization, and the Organization Manager field contains a dropdown menu of organization members from which to choose the manager.



The Organizations search screen displays a field to search for the organization you wish to find. In the example, Organization 1 was searched for. The search results display the organization's name, address, telephone number, and organization manager.



The Organization Members thing contains a screen in which to Add/Edit Members.



This screen allows new members to be added to the database and existing members' information to be edited. Included in this screen is the Member Organization dropdown menu, which allows each member in the system to be affiliated with the organization s/he is a member of. This field is populated with the organizations created in the Organizations thing. At the bottom is a field that indicates if this member is the organization manager.

The Search Members screen allows you to search each organization separately by making a selection from the organization dropdown menu, or to search for a specific member within a specific organization. In the example below, the name field was left blank, and the Member Organization field was set to Organization 1. So, all members from Organization 1 are returned.

Name	Member Organization
	Organization 1

Search

Search Results

	Name	Address	Organization Manager?	Telephone	Email	Country	Member Organization
Edit	Bob Brown	435 Same St. City, State, 45643	0	555-1256	bob@someurl.com	Afghanistan	Organization 1
Edit	Jane Doe	123 Some St. Anywhere, State, 54632	0	789-555-1234	jane@someurl.com	Afghanistan	Organization 1
Edit	JD Brown	567 Same St. City, State, 54678	1	123-555-0987	jd@someurl.com	Belarus	Organization 1

The results list all members in Organization 1. In the Organization Manager? field at the far right you can see that JD Brown is the manager of Organization 1. Each search result has a red X, by which to delete a member, and an Edit button by which to edit a member's information. Clicking on a member's name directs you to that member's full information.

Address	567 Same St. City, State, 54678
Organization Manager?	1
Telephone	123-555-0987
Email	jd@someurl.com
Country	Belarus
Member Organization	Organization 1

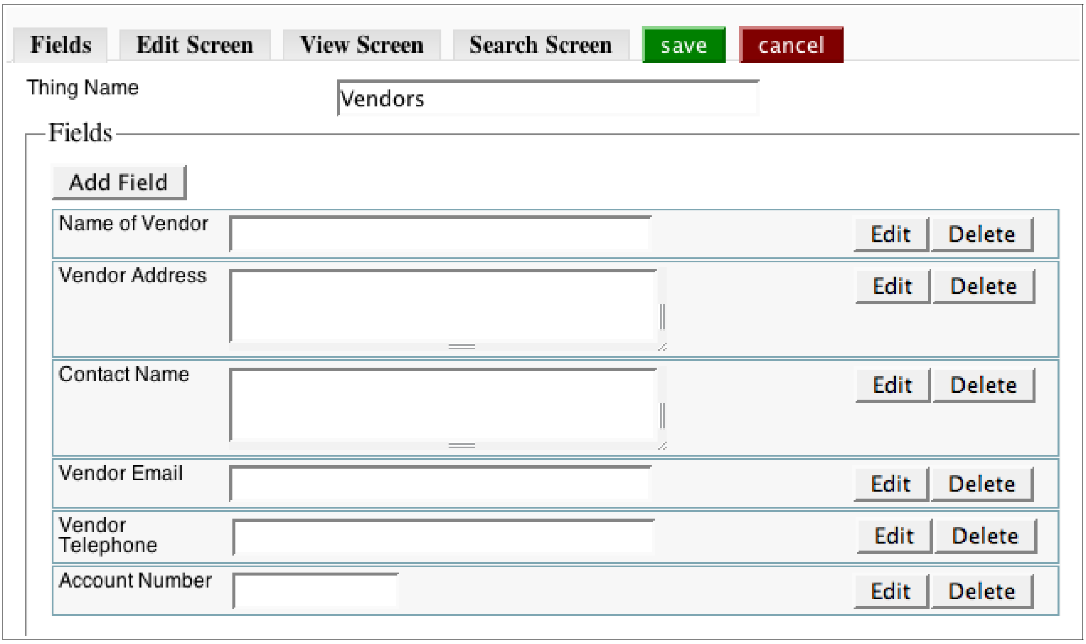
Because JD Brown is a member of Organization 1, you can see the Member Organization field contains a hyperlink to Organization 1. Clicking on this link directs you to the view screen for Organization 1, below, in which you will see JD Brown listed as the manager, and a link back to his information.

Organization Name	Organization 1
Street/Mailing Address	123 Some St. City, State, 12345
Country	Antarctica
Contact Telephone	555-1234
Organization Telephone	888-123-4567
Organization Manager	JD Brown

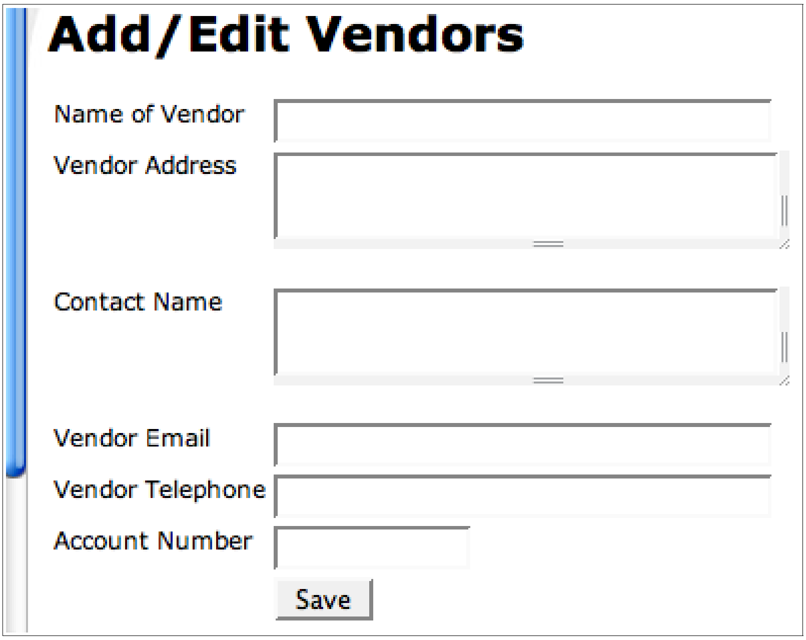
Inventory Manager

Consider using Thingy to keep track of product inventory and vendors. Link the two together so you can can access vendor information directly from a specific product's information.

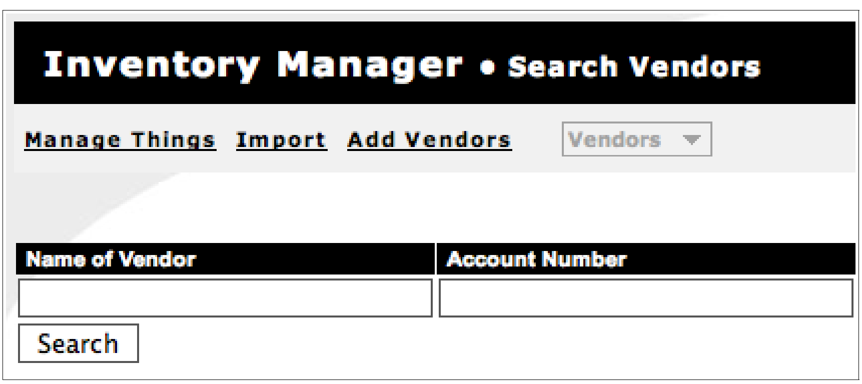
First Thing: Vendor Information



A thing was created to store vendor information. Using this form, users can input information, such as contact information and account numbers, for their product vendors.

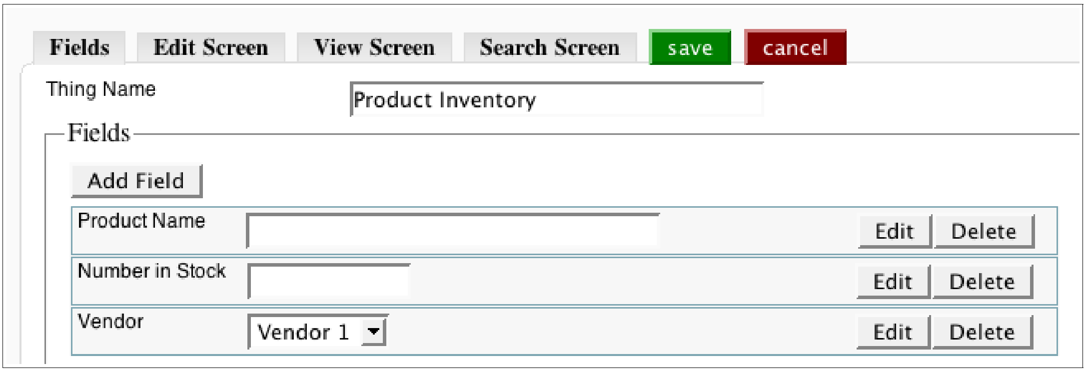


This thing allows users to search the vendor database by vendor name or by account number.

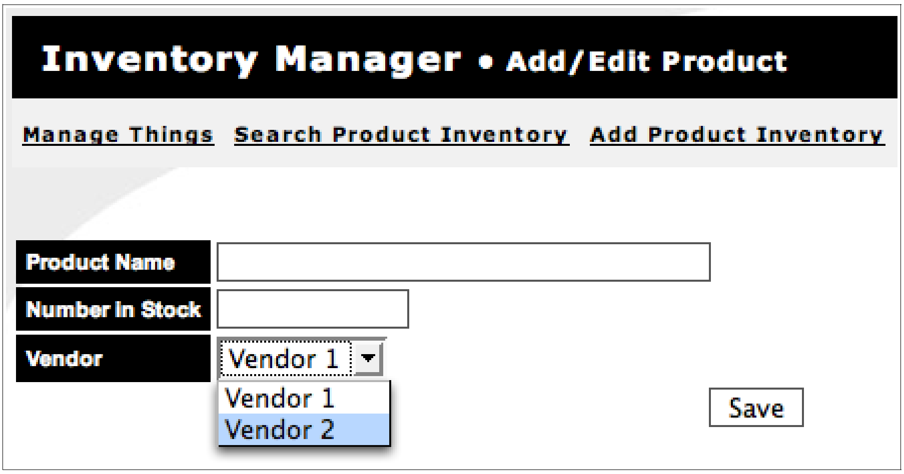


Second Thing: Product Inventory

Then, another thing was created to keep track of types of products, and the number of a product in stock. The Vendor thing was selected as the field type of the Vendor field. So, each product entered can be associated with a vendor that exists in the vendor database.

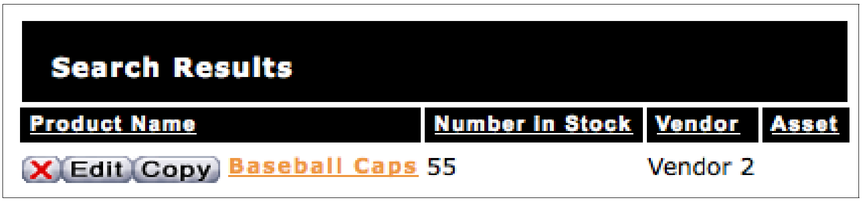


This is what the the Add/Edit Product screen looks like. Each product can have a vendor associated with it from the Vendor dropdown menu.

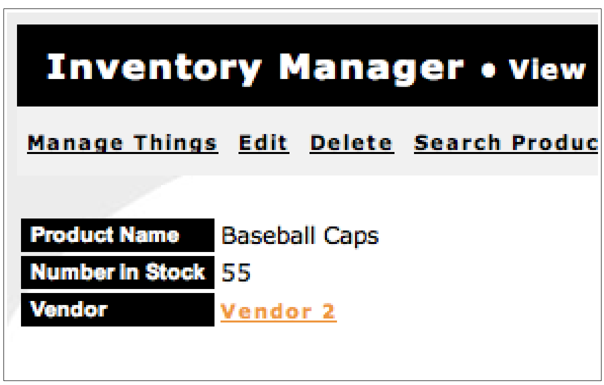


For search purposes, products can be searched according to product name or vendor.

In this example, baseball caps supplied by Vendor 2 was searched.

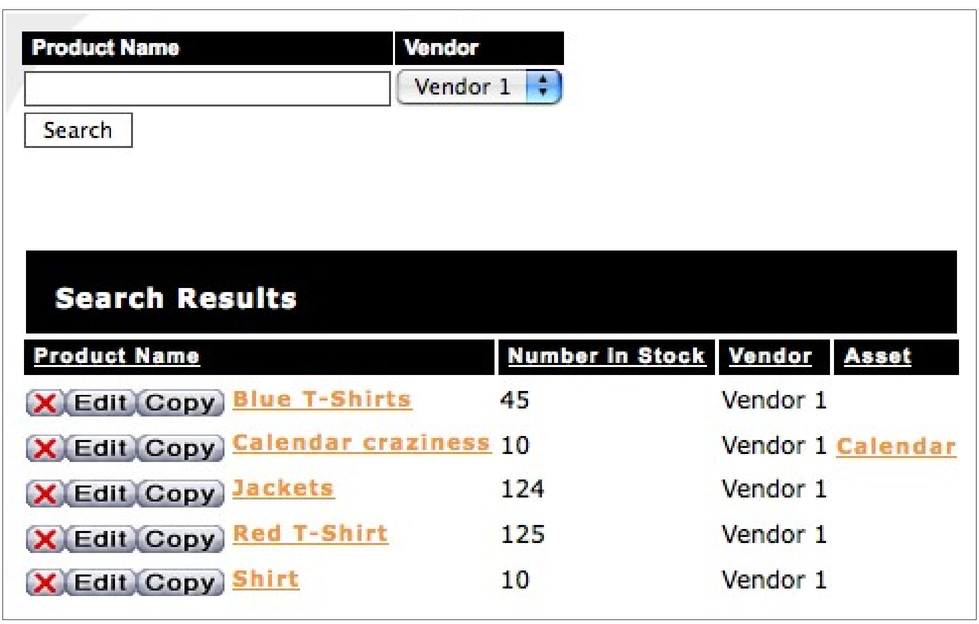


The search results indicate that 55 baseball caps remain in stock. To edit the number of caps, the edit button next to the product name, Baseball Caps, can be clicked to be directed to the Add/Edit Product screen for baseball caps. If more caps need to be ordered, clicking on the product name, Baseball Caps, will direct the user to the view screen.

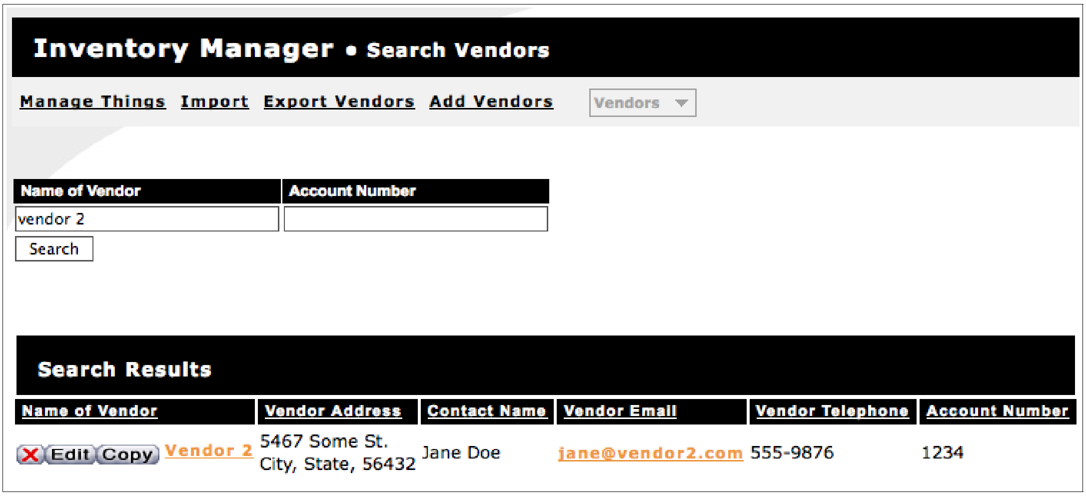


The view screen contains a link to Vendor 2, the vendor for this particular product. Clicking on the Vendor will direct you to the view screen, which contains full vendor contact information as entered in the Vendor Information thing.

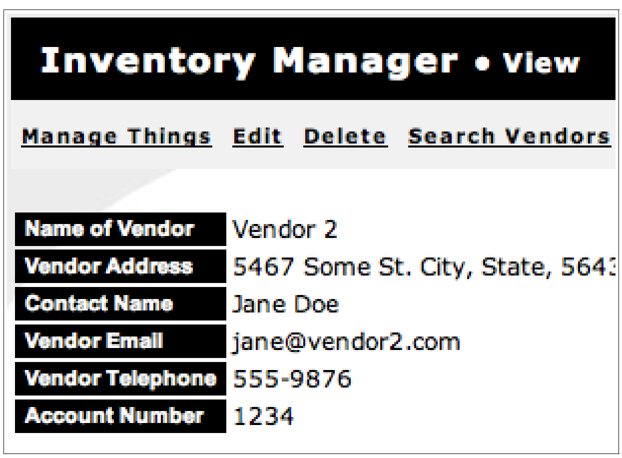
Or, the user can search for all products by a given vendor by leaving the Product Name search field blank, and selecting the desired vendor from the Vendor dropdown menu. In this example, Vendor 1 was selected, and all products associated with Vendor 1 are returned.



To access vendor information directly, users can search the vendor thing by vendor name or account number. In the example below, Vendor 2 was entered in the Search field.



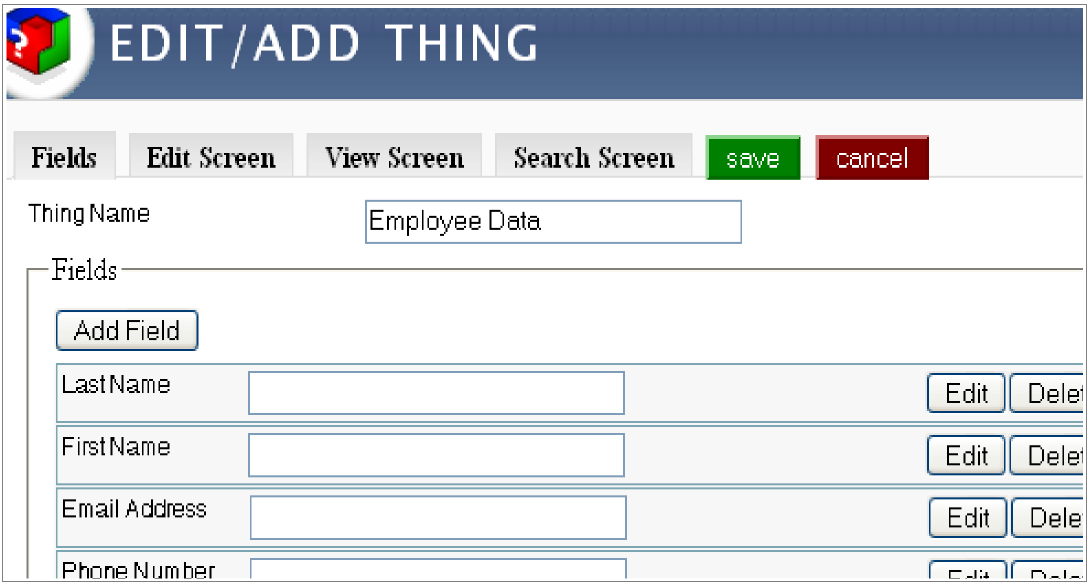
Using the edit button, information regarding this vendor can be altered. Clicking on the vendor name will direct the user to the view screen for that vendor.



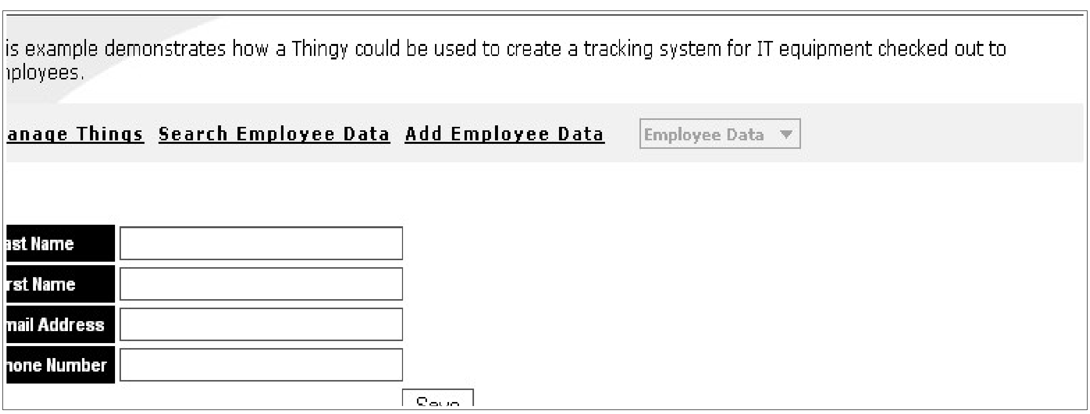
Tracking Equipment

Try using Thingy as a tracking system for IT equipment that is checked out to employees. In this simple example, two things are added to the Thingy asset: one to enter IT equipment, and one to enter employees checking out this equipment. The two things are then cross-linked, so when an employee checks out IT equipment his or her name attaches to that item.

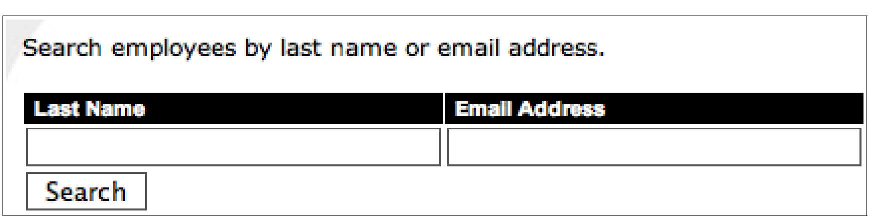
First Thing: Employee Data



A thing is created to store employee data. Using this form, users can input employee data, including last name, first name, email address, and phone number.

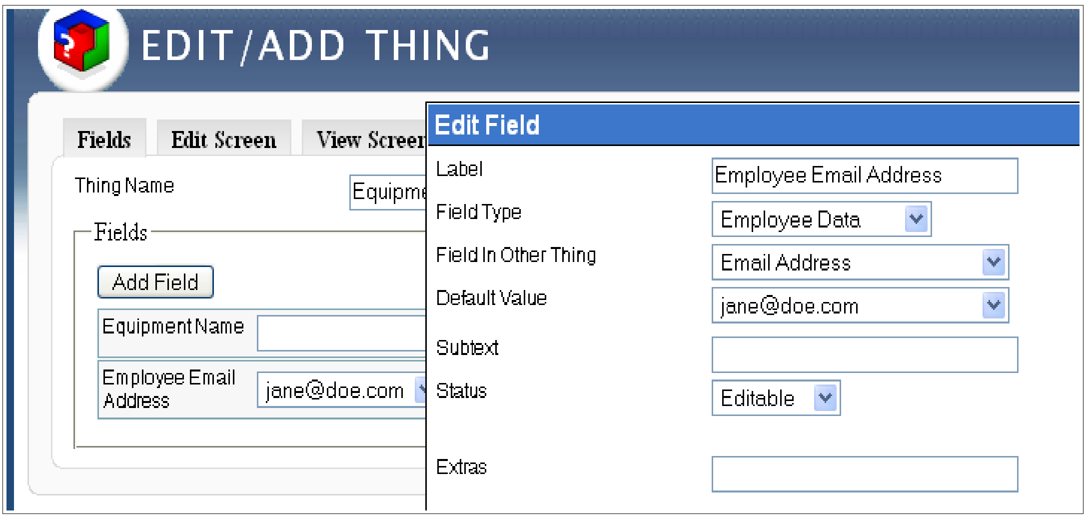


The search screen for this thing allows users to search the employee database by an employee's last name or email address.

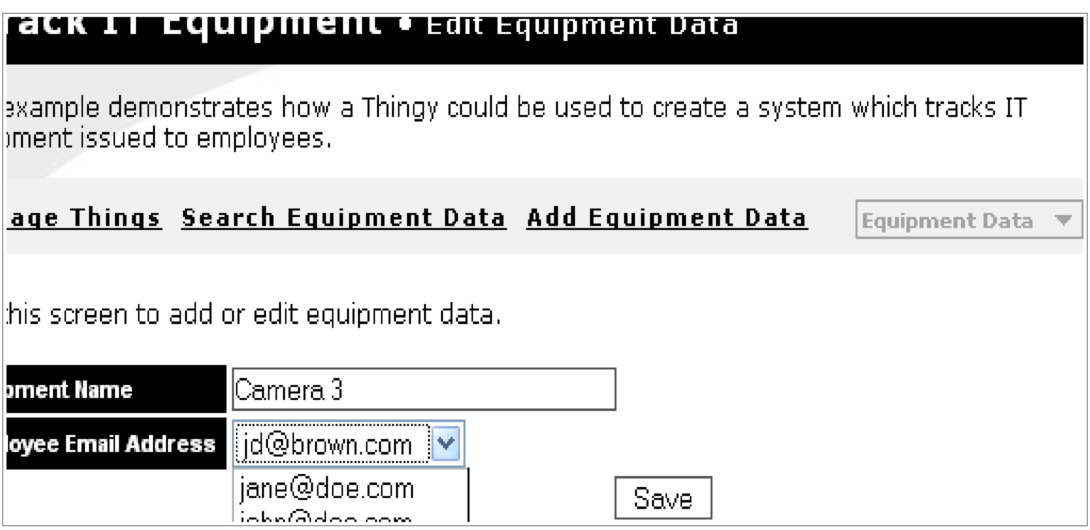


Second Thing: Equipment Data

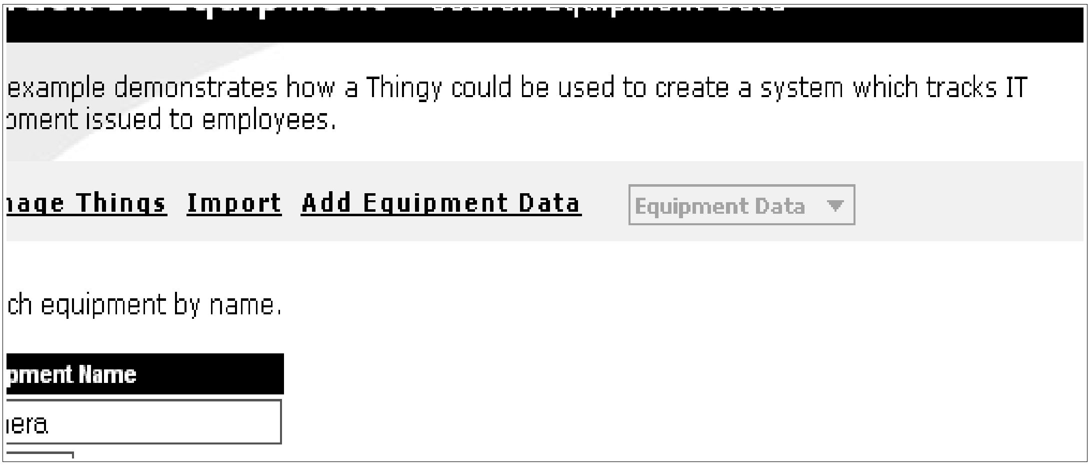
Then, another thing is created to store equipment data. The Employee Data thing is selected as the field type of the Employee Email Address field. So, each piece of equipment entered can be associated with an employee that exists in the employee database. Because some employees might have a common last name, the field chosen from the Employee Data thing is selected as the Email Address field.



This is what the Add/Edit Equipment screen looks like. Each piece of equipment can have an employee associated with it from the Employee Email Address dropdown menu. An employee may be associated with more than one piece of equipment, but a piece of equipment may only be checked out to one person at a time.

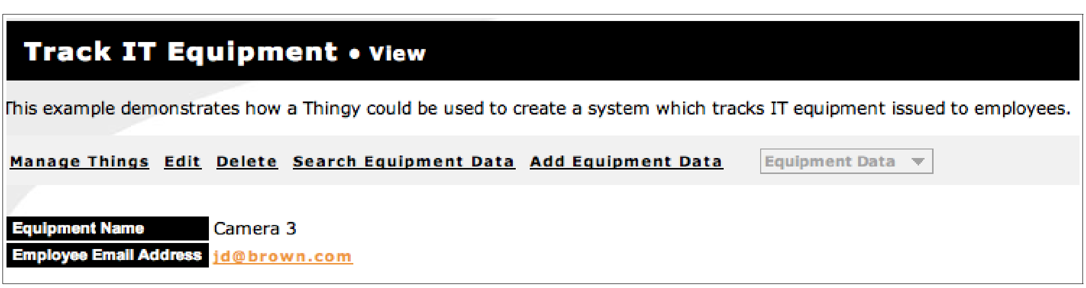


For search purposes, equipment can be searched by name using search terms such as “camera,” “scanner,” “phone,” or “computer.”



In this example, the term “camera” was searched.

The search results indicate that Camera 1 and Camera 2 are not checked out, while Camera 3 is associated with the email address jd@brown.com. To view more information about this, click the link to Camera 3.



The view screen contains a link to jd@brown.com, the employee email address associated with Camera 3. Users can email the employee directly based on this screen, or click the email address for more information about this employee, including his last name, first name, and phone number as entered in the Employee Data thing.

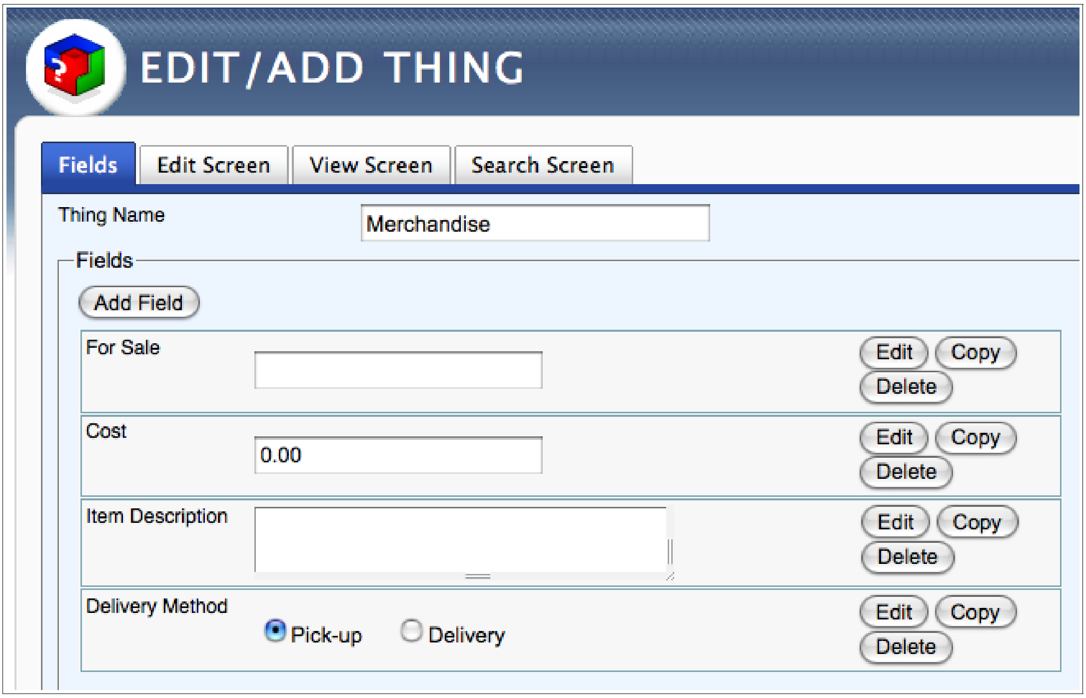
ThingyRecord

ThingyRecord links WebGUI's Thingy asset with WebGUI Shop. It allows you to sell a record inside a Thing. This allows users to build things like an online classifieds system. A Thing inside a Thingy asset is built to house data. Then, a ThingyRecord is built, and in ThingyRecord you indicate the fields you want users to fill out. Once they complete those fields, they purchase the record. Upon doing so, the data they provided appears in the Thing.

In this example, an online classifieds system will be built. Before beginning, make sure that WebGUI Shop is set up and configured for your site.

To begin, a Thingy asset to house the classifieds system was created. In the Thingy and Thing was built for the classifieds category “Merchandise”. Additional Things could be created for other classifieds categories such as “Pets” or “Real Estate.”

In the Merchandise Thing, the basic fields for a classifieds ad were created.



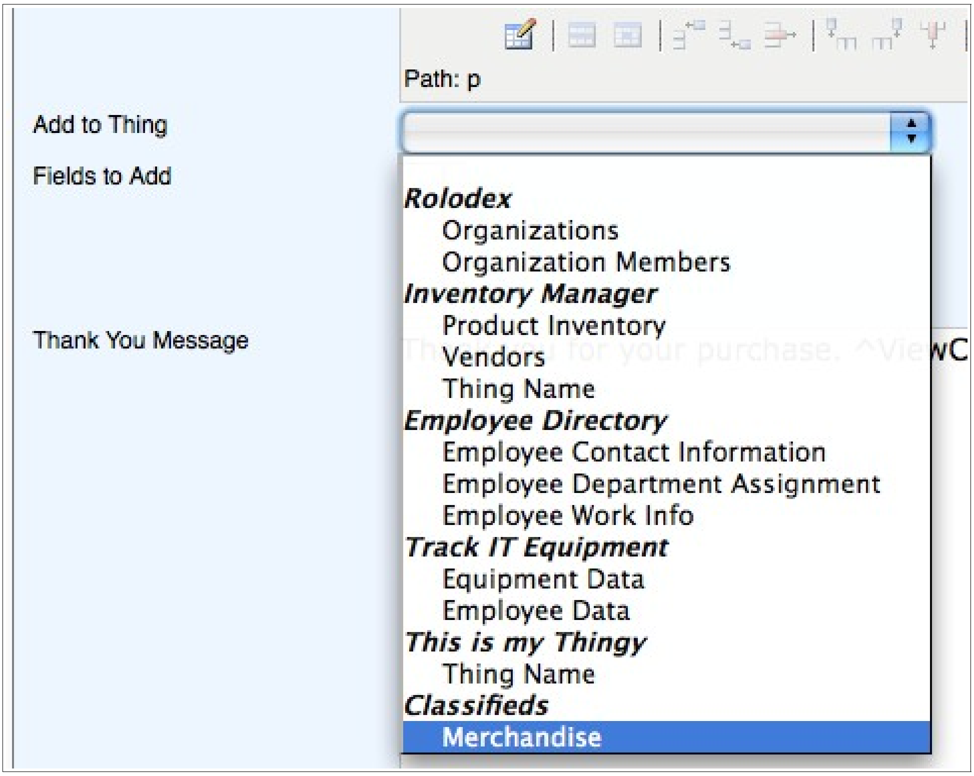
The Default view of the Thing is the Search screen, and it has been set to allow site visitors to search by item name, which is the the "For Sale" field.

Once the Thing has been created, the ThingyRecord can be created to allow people to advertise their merchandise in your classifieds ad system. You can use ThingyRecord to charge a cost for placing the ad.

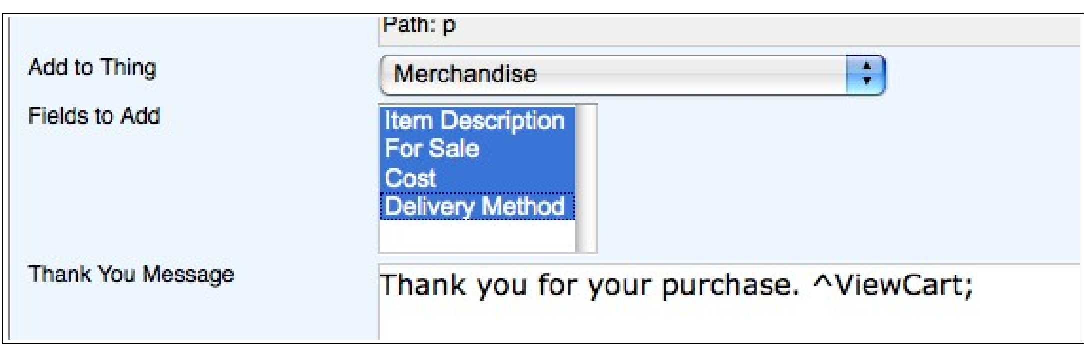
To create a ThingyRecord asset:

1. Navigate to the page you wish to place ThingyRecord on.
2. Select ThingyRecord from the New Content menu. The "Add/Edit Thingy Record" screen will open.

3. Enter a title for the ThingyRecord in the "Title" field.
4. If necessary, enter a different Menu Title. This is the ThingyRecord title for the site navigation. By default, this asset is hidden from the navigation.
5. The URL field can be left blank, or you can enter one.
6. In the Description field enter text related to this item of content. In this example a brief description of the purpose of the asset has been entered.
7. Below the Description field is the "Add to Thing" drop down menu. From this menu you can select the Thing whose fields you are using in ThingyRecord. In this example, the Merchandise thing from the Classifieds Thingy was selected.



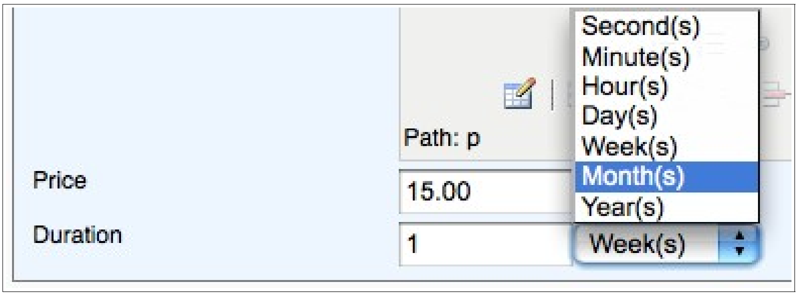
8. The “Fields to Add” field lists the fields the Thing (selected in number 7) contains. Select the fields you want users to complete when purchasing a ThingyRecord. In this example, users must complete all four fields for their classifieds ad when purchasing the ad, so all fields have been selected.



9. The “Thank You Message” field displays the thank you message people will see upon submitting the ThingyRecord. You can change this text. The ^ViewCart; macro is what displays a link that allows users to link to their shopping carts and pay for this record. You will

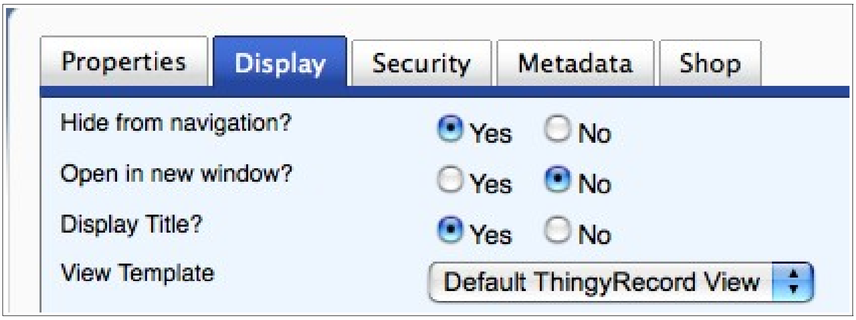
most likely want to leave the macro in place.

10. In the “Price” field, enter the amount that people will be charged for purchasing this ThingyRecord.

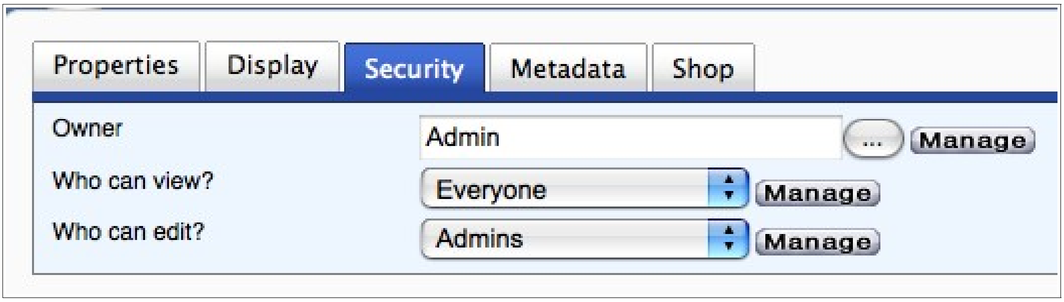


11. The “Duration” field allows you to indicate how long the record will appear in the Thing. For example, if set to 1 Month, the person's classifieds ad will appear in the Merchandise Thing for one month.

12. Set Display options in the Display tab. This is the standard Display tab.

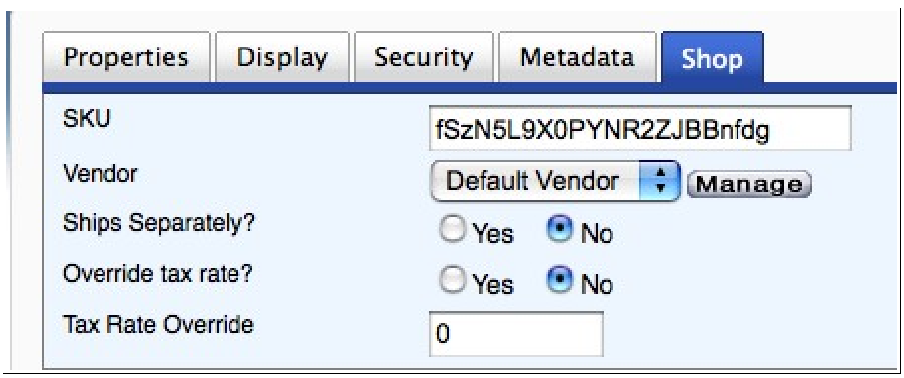


13. Set Security permissions in the Security tab. Again, this is the standard Security tab.



14. If you wish to enter Metadata for this item, do so in the Metadata tab. The tab contains all the standard Metadata fields. In WebGUI Shop, you can dynamically pull in SKU assets for display on a Shelf using Keyword association. For example, I can enter a Keyword of Merchandise for a ThingyRecord asset. I can then create a Shelf asset in another location on the site. If I also give the Shelf asset a Keyword of Merchandise, the Shelf will pull in and display all ThingyRecord assets and other SKU assets that have that same keyword in their Metadata tabs.

15. The Shop tab contains items related to WebGUI Shop commerce.



A. “SKU” stands for Stock Keeping Unit. It is basically a special product code or inventory unit. WebGUI will create a unique SKU for you, or you can enter your own.

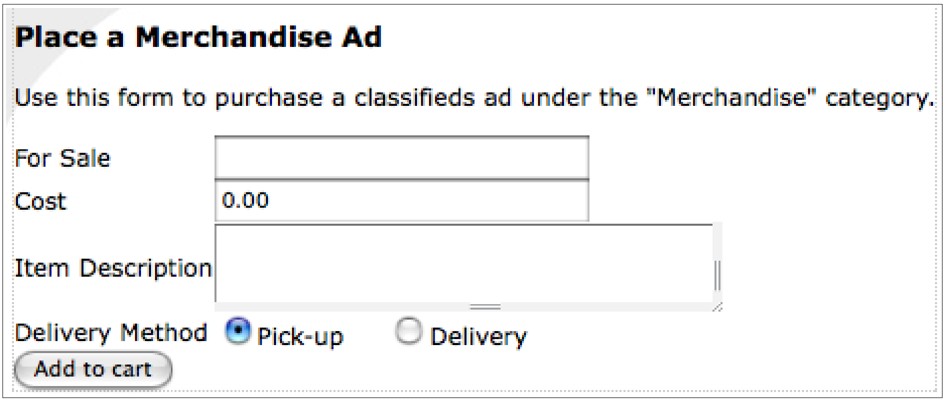
B. “Vendor” is the person or company who gets credit for selling this item. Vendors are set up in the Admin Console Shop screen. This is used for sites where outside vendors sell items on your site and receive a portion of the sale.

C. “Ships Separately?” allows you to indicate if this item should accrue additional (separate) shipping costs. In the case of ThingyRecord you can probably leave this field set to No.

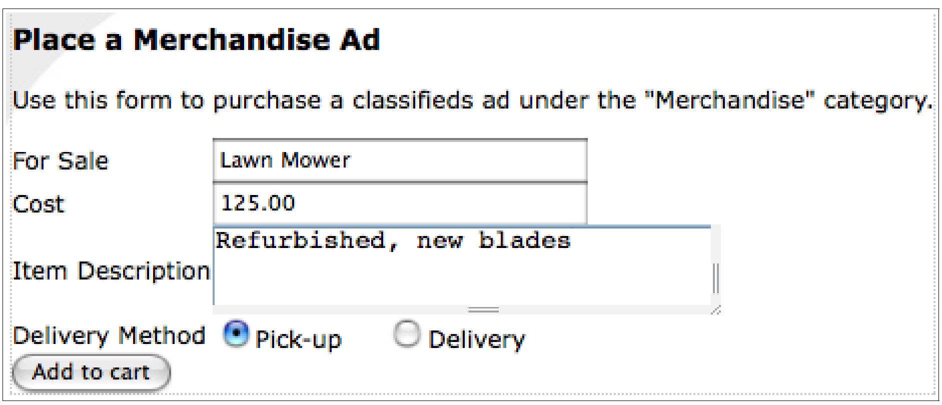
D. “Override tax rate?” allows you to override the default tax rate for an area. This is most often used in areas of the world where essential items, like food and clothing, are exempt from taxes. In the case of ThingyRecord you can probably leave this set to No.

E. If “Override tax rate?” is set to Yes, “Tax Rate Override” is where you enter the new tax rate to calculate on this item.

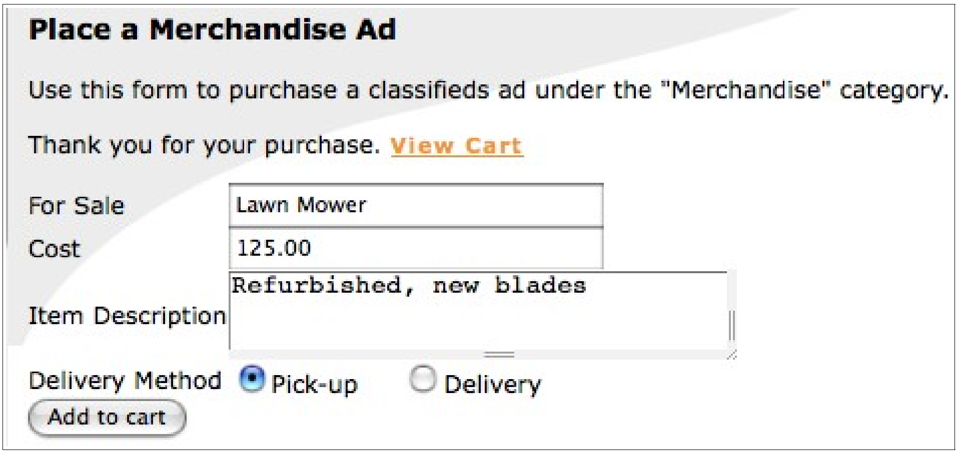
16. Click the save button to create your ThingyRecord asset.



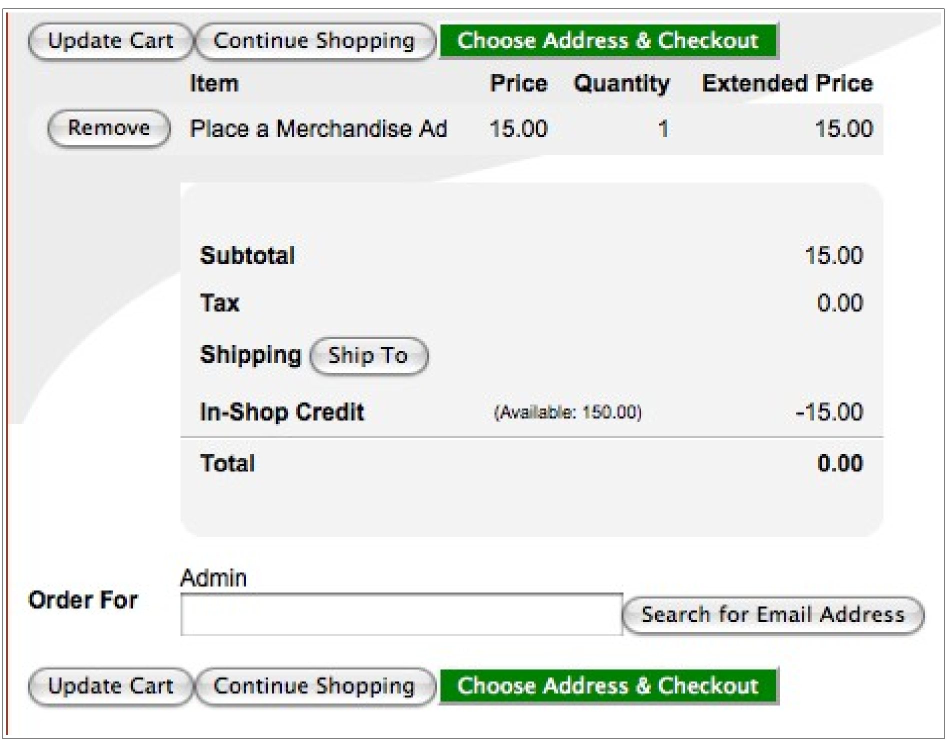
When site users visit the site they fill in the fields in the ThingyRecord. Then, they click the “Add to cart” button.



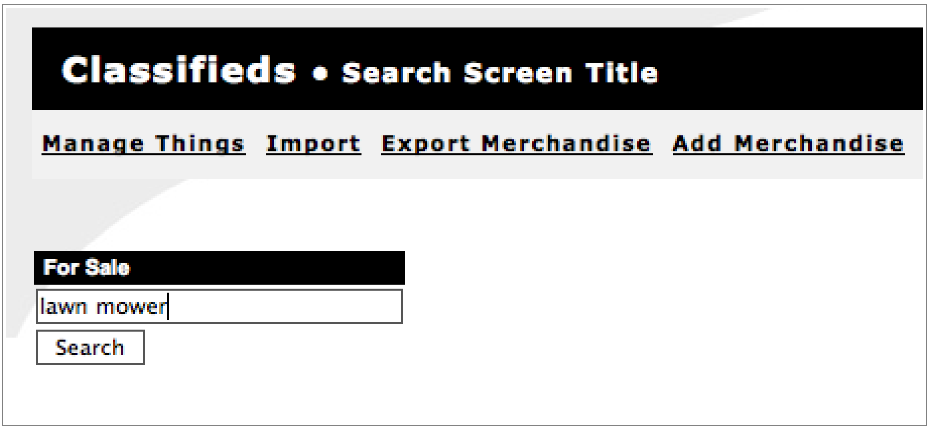
Upon submitting the form, the user will be shown (by default) a thank you message, the form with their data filled in, and a link to view their cart.



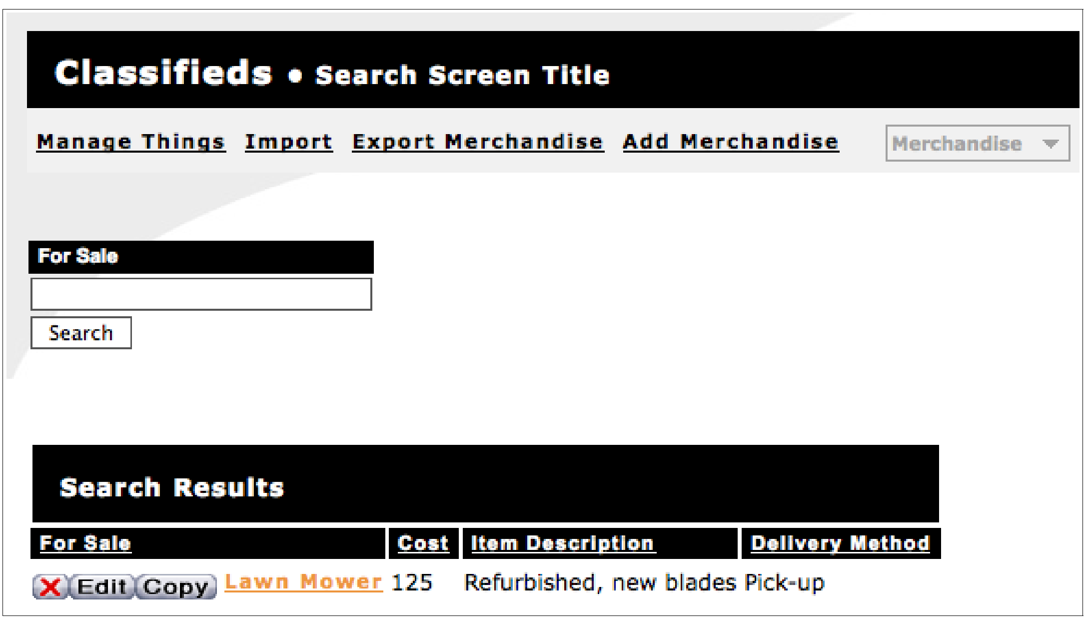
The user can click on the View Cart link to proceed to the shopping cart and finish the check out process.



Upon visiting the original Classifieds Thingy, users can search for items they would like to purchase.



If an item matches, it will appear in the search results. In the example below, you can see the classified ad purchased through ThingyRecord appears.



Advertising and Ad Sales

The advertising module in WebGUI gives you an easy way to put advertisements into a rotation on your web site. You may be thinking to yourself, “Why is that useful? I get the code from my ad partner and put it into my HTML. I don't need no stinking advertising system.” You're only half right if that's what you're thinking. Depending upon what your site is about, you may be able to make more money by selling ads directly to advertisers. In addition, you may be able to make more money by using more than one ad network. Using the ad system allows you to distribute ads for multiple ad networks in the same space, and allows you to throw in your own advertisements as well.

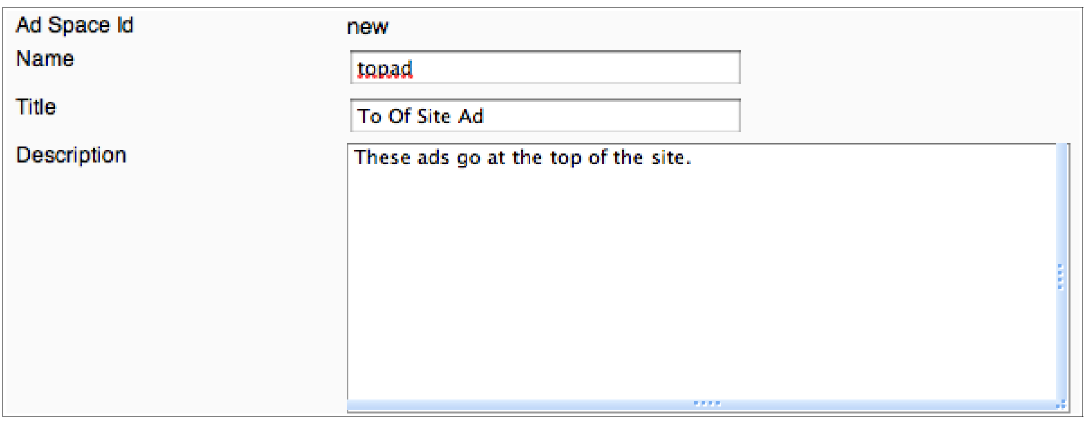
Beyond selling advertising on your site, have you ever thought about how you could use the popular areas of your site to increase awareness of the newer or less frequented areas of your site? You can do this too using the ad management system. Instead of placing advertisements, you're placing your own site elements into the rotation. You can see this in use on the front page of plainblack.com:

Creating Ad Spaces

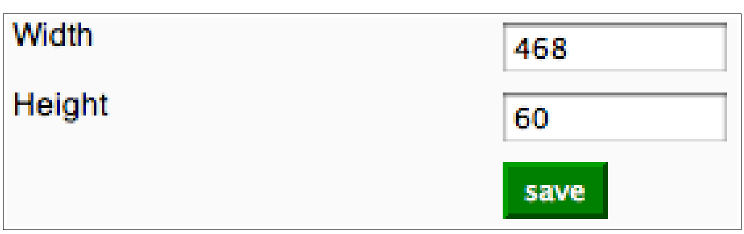
An Ad space is a placeholder for where you will put advertisements. With it you can define the size of ads allowed, and where they are placed on your site.

To create an ad space go to Admin Console > Advertising. Then choose "Add ad space." on the right side.

Give a name title and description to your ad space. These can each be anything you want them to be, but the name should be unique amongst all ad spaces defined on your system.



You can also define how large of an ad space this will be. The ad system will enforce this so that ads can't disrupt the look and feel of your site even if someone ads and ad that is too big for the ad space.



Hit save when you're done editing this new ad space. You'll now see a box appear that shows you how to use the ad space.

You'll note that an ad space is placed using nothing more than a macro. It takes the form of ^AdSpace(ad space name); You can define as many ad spaces as you want on a site, and place as many as you want into a given page. Normally you'd place the AdSpace into a style template.

Create Ads

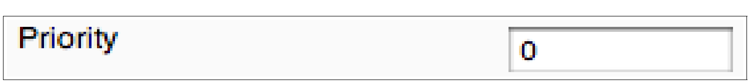
Now that we have an ad space, let's create an ad. From the ad space screen click on “Add an ad.”

Enable the ad. You can disable the ad if you ever want to stop running it, but don't wish to delete it.

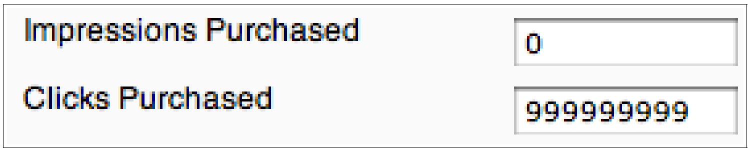
Then fill in the title. The title is displayed in text ads, and is used as the alt text in image based ads.

Now fill in the URL. This is where the user will be directed to if they click on text based or image based ads.

If you wish you can set the priority of the ad. The closer the priority is to 0 the more often the ad will be displayed. This is not an exact ratio, but rather a scaling mechanism. The more impressions per minute your site displays the larger the impact of the scale. If your site only displays one impression per minute, then priority will have no effect.



Now enter the number of impressions or clicks to be displayed for this ad. The ad will only be displayed that number of clicks or impressions. If you want the ad to run indefinitely then enter 999999999 clicks. Note that you can only use impressions as a counter for rich media ads.



Finally, choose the type of ad that you wish to display. You can choose from text, image, and rich media ads.

Text Ads

Text based ads generate the ad using just text and a simple color palette. First enter the text of the add, which will appear under the title. Then pick the colors you'd like the ad to use.

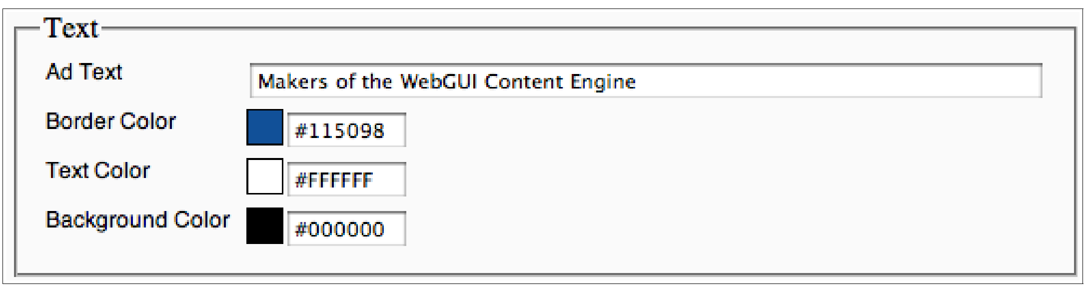
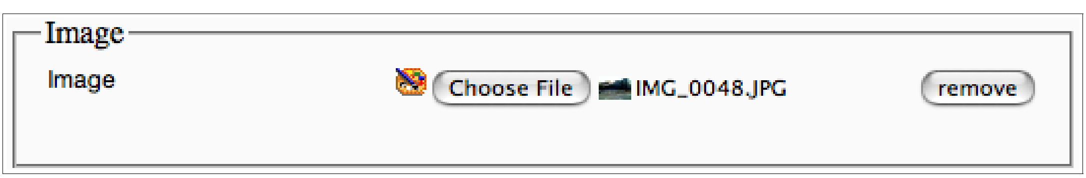


Image Ads

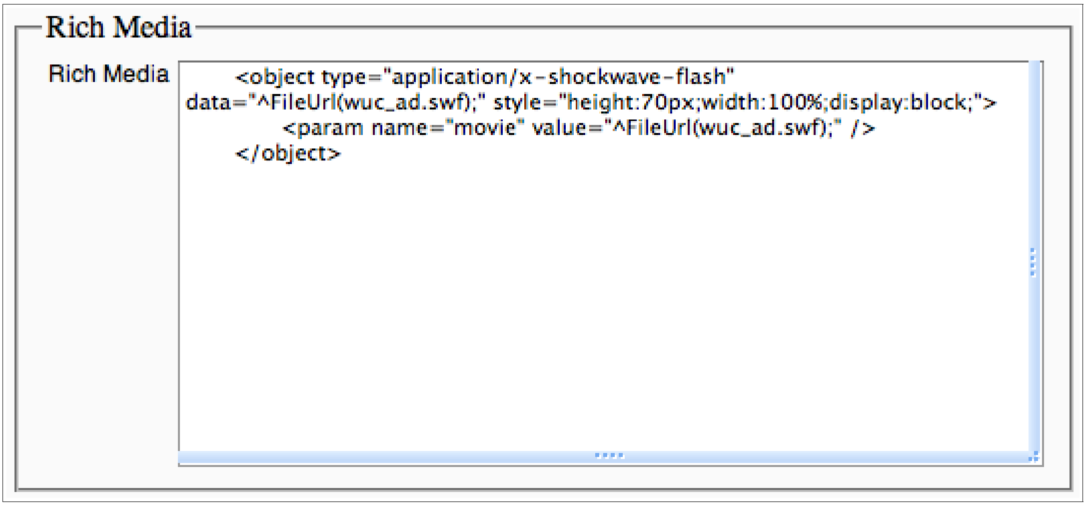
Image based ads are the more traditional banner ads that you see around the world wide web. If you selected image based ad then you can upload an image to use here.



Rich Media Ads

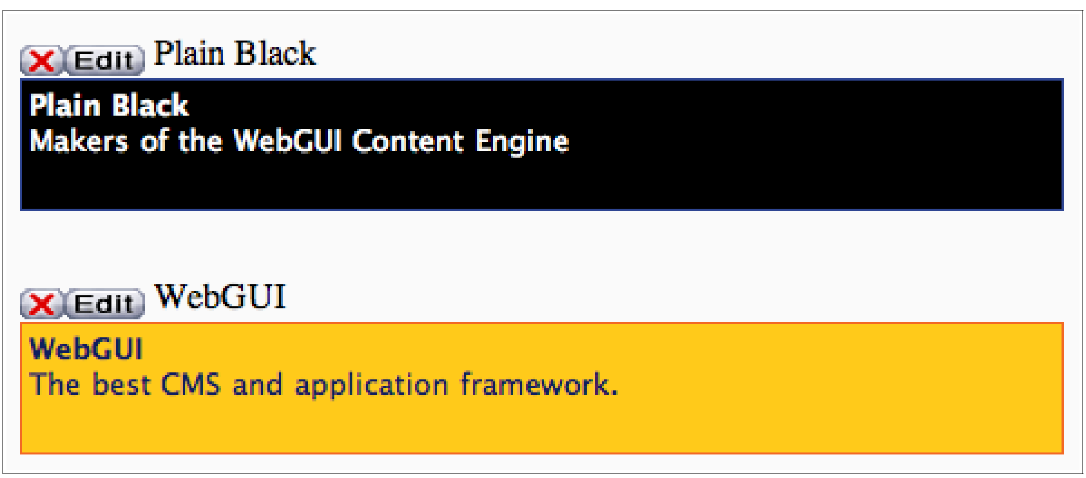
Rich Media ads are ads that allow you to place HTML tags in them. Using this you can make ads from complex HTML, video files, or flash movies.

Simply place the HTML into the rich media field.



Viewing Ads

If you edit an ad space after you've defined ads you'll see a list of the ads assigned to this ad space.



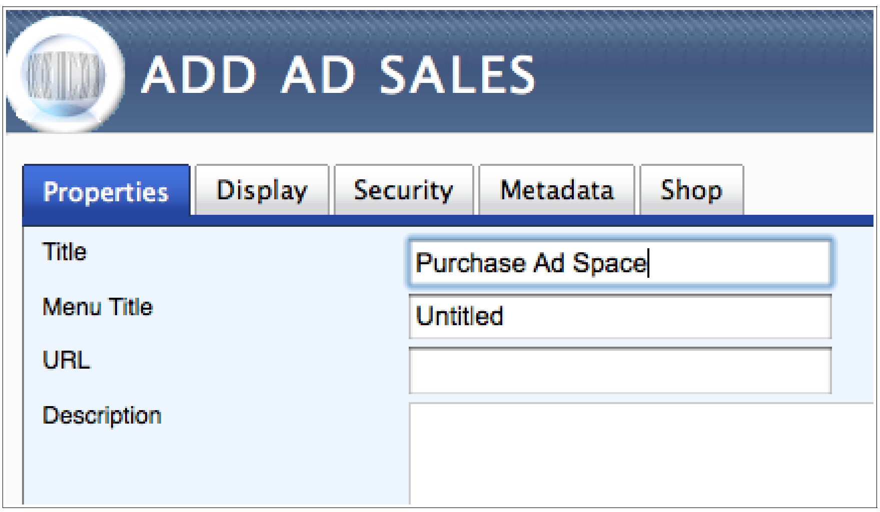
If you edit one of those ads, you'll be able to see how many impressions and clicks it's had.

Ad Sales

The Ad Sales asset allows you to sell ad space on your website. Prices can be set on a per click or per impression basis.

To sell ad space on your site:

1. Select “Ad Sales” from the New Content menu. The “Add Ad Sales” screen will open.



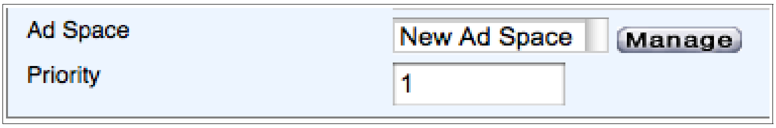
2. Enter a title for the asset in the “Title” field.

3. If you would like, a “Menu Title” can be entered. This is the title as it will appear in the site navigation and the asset manager. If left blank, the asset's title will be used here as well.

4. The URL field is the URL for this specific asset. It an be left blank, and WebGUI will create one, or you can enter one.

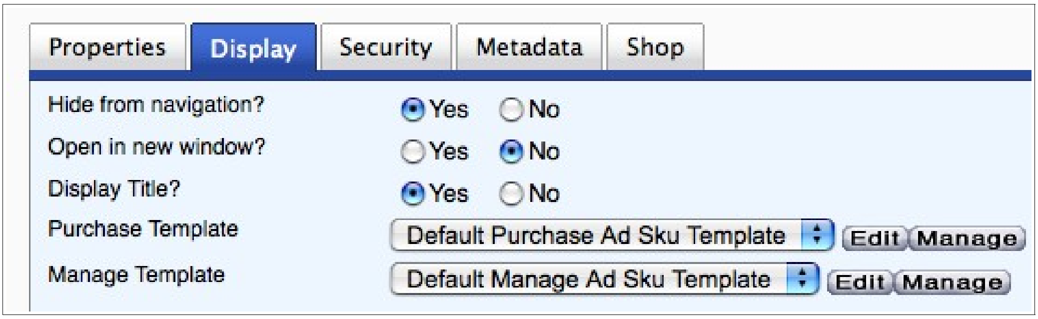
5. In the “Description” field you can enter text. This could be a description of what this asset is used for, or a set of instructions for your site users. This would be a good place to enter things such as the size of images that may be uploaded for use in an ad.

6. In the “Ad Space” field, select the ad space, from WebGUI's Advertising administrative interface, that would you like ads purchased through this asset to be displayed under.



7. The “Priority” field indicates the level of priority (10 highest, 1 lowest) ads purchased through this asset will hold. Ads with a higher priority level are displayed more often. If all assets are to hold the same level of priority, you can leave this field set to one.

8. Set Display options in the Display tab.



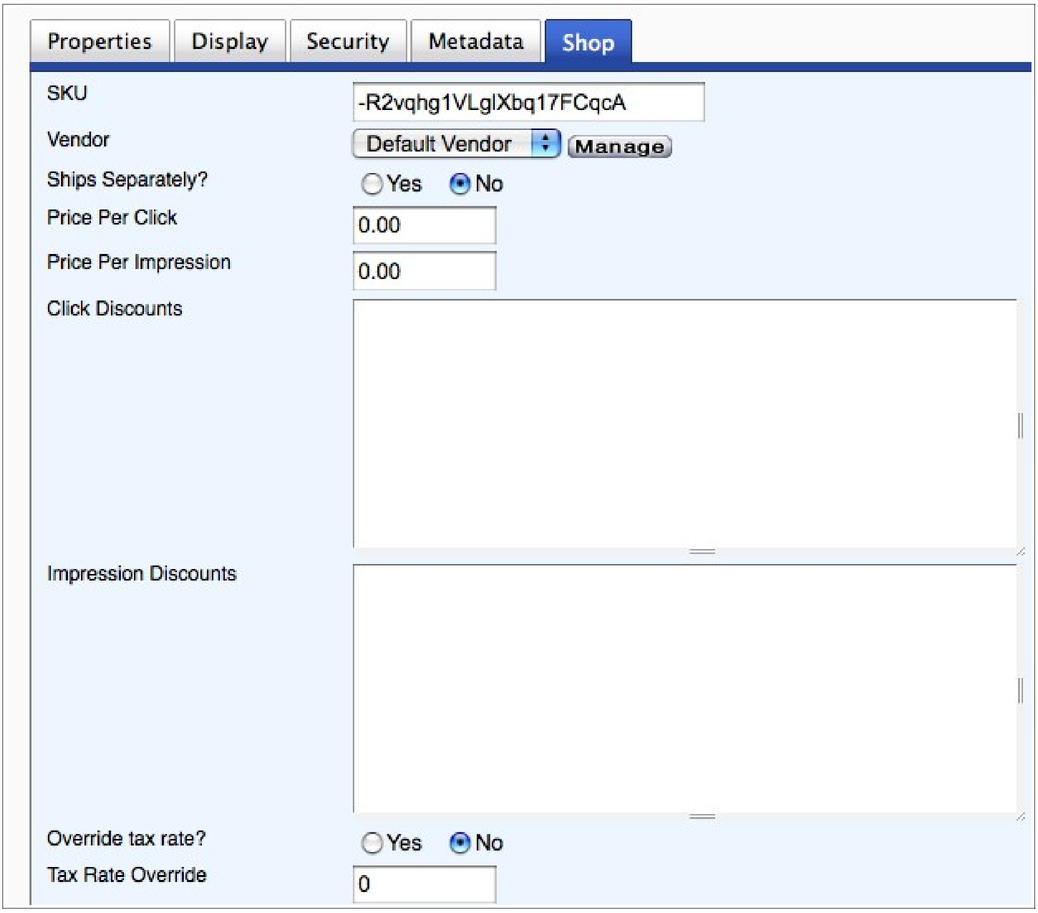
A. The first three fields are the standard in the Display tab.

B. The template fields relate to specific templates used to display the ad purchase screen and ad manage screen of this asset. If you have a custom template, you could choose it in these fields, or just leave them at the defaults.

9. Set Security options in the Security tab. These are the standard Security fields common amongst most assets.

10. Manage optional Metadata fields in the Metadata tab. These are the standard Metadata fields common amongst most assets.

11. Set Shop-related properties in the Shop tab.



A. SKU: stands for stock keeping unit, and is a unique value assigned to this asset. You can edit this field to something different if you would like.

B. Vendor: this asset can be associated with a specific vendor that has been set up in the Shop's Vendor screen. This is helpful if you are selling outside vendor items on your site and need to keep track of those sales.

C. Ships Separately?: indicate if this item ships separately from other items in a transaction. If it does, separate shipping costs will be charged for this item.

D. Price Per Click: you can charge for ads on a per-click basis. When people by ad space on your site, they will indicate the number of clicks they would like to purchase.

E. Price Per Impression: you can charge for ads on a per-impression basis. When people by ad space on your site, they will indicate the number of impressions they would like to purchase.

F. Click Discounts: enter discounts one per line at the start of the line. Extra text is ignored so you can add comments to the discounts. Each discount consists of two numbers separated by '@' with no spaces. The first number is the percent discount(no decimal point) the second number is the number of items purchased. So 5@1000 indicates a 5% discount for 1000 or more clicks purchased.

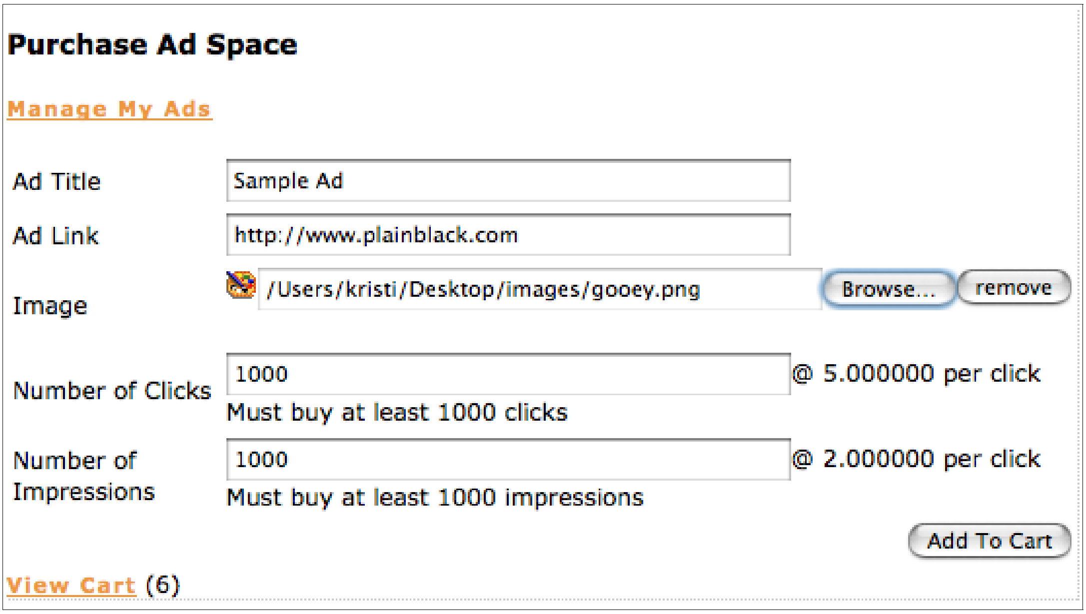
G. Impression Discounts: Enter discounts one per line at the start of the line. Extra text is ignored so you can add comments to the discounts. Each discount consists of two numbers separated by '@' with no spaces. The first number is the percent discount(no decimal point) the second number is the number of items purchased. So 5@1000 indicates a 5% discount for 1000 or more impressions purchased.

H. Override tax rate?:"Override tax rate?" allows you to override the default tax rate for an area. This is most often used in areas of the world where essential items, like food and clothing, are exempt from taxes. In the case of Ad Sales, you can probably leave this set to No.

I. If "Override tax rate?" is set to Yes, "Tax Rate Override" is where you enter the new tax rate to calculate on this item.

12. Click save to create the Ad Sales asset.

When site visitors come to this location, they can purchase an ad space on your site by filling in this form, and an purchasing the ad through Shop's cart.



The fields in the Ad Sales purchase form are:

Title: the title of the ad being purchased. This will be used in the manage screen to identify this ad.

Ad Link: the location people will be linked to if they click on an ad.

Image: the ad image that will be displayed.

Number of clicks: the number of clicks being purchased (at the price indicated).

Number of impressions: the number of ad impressions being purchased (at the price indicated).

Then, users click on the Add to Cart button to add the ad to their shopping carts and complete the check out process from there. The ad the user purchased will be added to the appropriate ad space, and displayed in the location on the site configured to display advertisements. Once the number of clicks/impressions has been reached, the ad will no longer appear in the rotation.

Event Manager

The Event Manager is useful for selling admission to a convention or similar event. The Event Manager allows you to sell badges to an event, create tickets for admission to individual events taking place at the main event, create ribbons to redeem at the event for discounts, and create tokens to use as currency at the event.

1. Select Event Manager from the New Content menu of the Admin Bar.

2. The "Add/Edit Event Manager" screen will open.

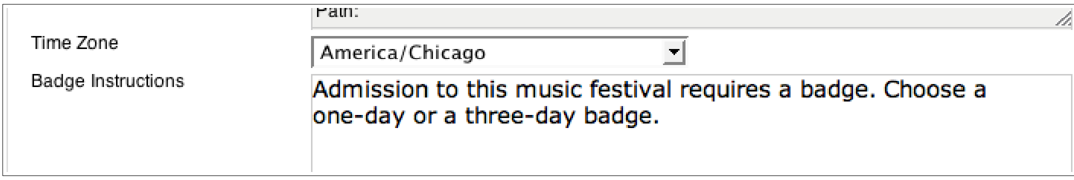
3. Enter a title for your event in the "Title" field.

4. The "Menu Title" is the title as it appears in the site navigation and site maps. You can leave this blank and WebGUI will insert a menu title for you (by default, WebGUI will use the "Title").

5. The URL can also be left blank and WebGUI will generate a URL for you.

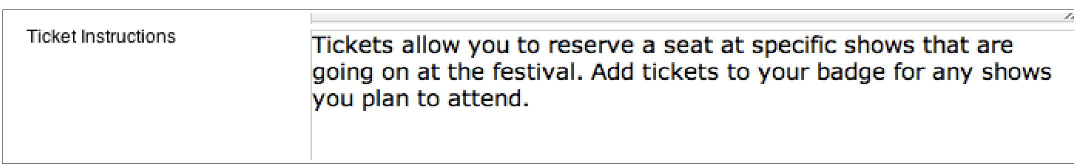
6. In the "Description" field, enter a short description of the event. This will remain as static text and will appear beneath the event title and above badges available for purchase.

7. Select the "Time Zone" in which this event will take place.

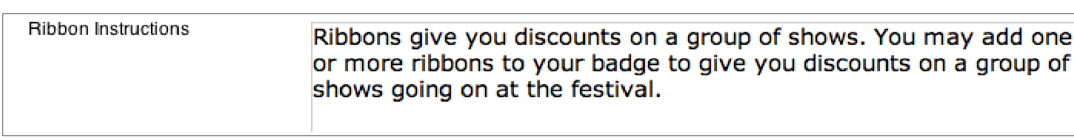


8. In the “Badge Instructions” field, enter instructions for the registrant regarding the purchase and use of a badge. The badge allows admission to the event.

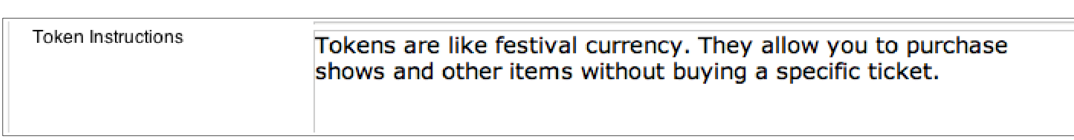
9. In the “Ticket Instructions” field, enter instructions for the user regarding the purchase and use of a ticket. The ticket is purchased in addition to the badge, and allows a registrant to reserve a seat/spot at events, such as a special speaker or workshop, that are going on at the main event. You will want to use tickets for events that have limited attendance.



10. In the “Ribbon Instructions” field, enter instructions for the registrant regarding the use of a ribbon. Ribbons can be purchased, and in return the registrant will receive a discount off of any related tickets purchased. See the section on creating ribbons for more information.



11. In the “Token Instructions” field, enter instructions for the registrant regarding the purchase and use of tokens. Tokens can be redeemed at the event for physical goods, or put towards purchasing access to events once onsite if seats remain. See the section on creating tickets for more information.



12. Choose your display settings from the “Display” tab.

A. If the “Hide from navigation?” field is set to “Yes,” your \Event Manager “Menu Title” will not appear in the site navigation.

B. If “Open in new window?” is set to “Yes,” the Event Manager will open in a new browser window when viewed directly.

C. If “Display the title?” is set to “No,” the title of the Event Manager entered in the “Properties” tab will not be displayed to users.

D. You can choose a “Style Template” to be applied to the Event Manager if it is to be viewed directly. If viewed on a Page Layout, the layout style template will be used instead.

E. You can choose a “Printable Style” for the page.

F. Under “Main Template,” you can choose the template to use for the main screen which lists the badges for purchase. This is the screen registrants see upon initially viewing the asset.

G. The "Schedule Template" is the template used to display the schedule for events.

H. The "Schedule Locations Per Page" field allows you to set the maximum number of event locations that are displayed on each page of the Schedule. If you only have a few events, you can just set this field to the number of events. If you have around 10-20 events, then you might want to set this number to around half the number of total events. If you have many events you can leave this field at the default value.

I. Under "Badge Builder Template," you can choose the template to use for the badge builder screen.

J. Under "Lookup Registrant Template," you can select the template to use for the screen on which the Registration Staff Group can look up a registrant's badge.

K. Under "Print Badge Template," you can select the template to use for printing badges.

L. Under "Print Ticket Template," you can choose the template to use for printing tickets.

13. Choose your security settings from the "Security" tab.

A. The "Owner" has full editing and viewing rights for this asset. Usually, the owner is the person who created the asset.

B. The “Who can view?” field allows you to limit who can view this asset on the site. Select a group of users allowed to view this asset from the dropdown menu. If there are no viewing restrictions, leave the menu at Everyone. The Owner can always view the asset.

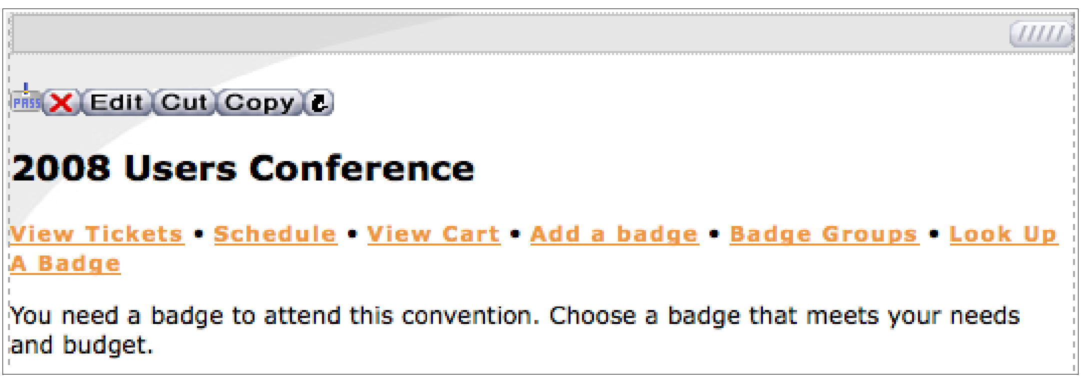
C. Select who has editing rights for this asset from the “Who can edit?” dropdown menu. The Owner can always edit.

D. From the “Registration Staff Group” dropdown menu, select the group of users that will handle registration. These users can look up and manage the badge registrations for any attendee.

The screenshots in this chapter are taken as a user in the Registration Staff Group.

14. If you wish, enter metadata in the “Metadata” tab. The metadata tab is the standard tab seen amongst most assets.

15. When all your settings are complete, click save.



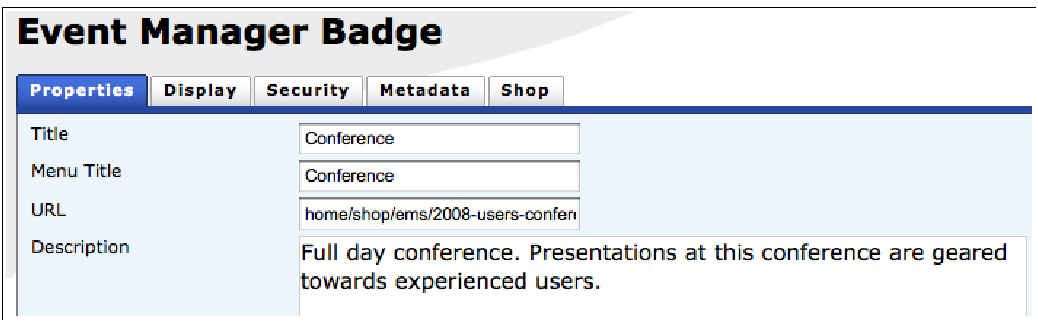
Add a Badge

Badges are used to purchase admission to an event. Once a badge is created, additional tickets to individual events being held at this main event can be created. So, you may have a conference (the badge), and at the conference there may be special workshops or presentations that require additional admission (tickets).

1. Click the link to “Add a badge” located beneath the Event Manager asset title and description.

2. The “Event Manager Badge” screen will open. Registrants will purchase a badge to gain admission to an event. You can have multiple badges for an event if you want to give users options. For example, a three day event may have a badge for each individual day, and perhaps a discounted badge for attendees who chose to attend all three days or who register early.

You can create Badge Groups that will allow you to associate individual badges with tickets to events. When a user buys a badge, s/he will be directed to a screen showing all tickets available for purchase at the event. Badge Groups are how you make the association between badges and tickets. See the section on Badge Groups for more information.

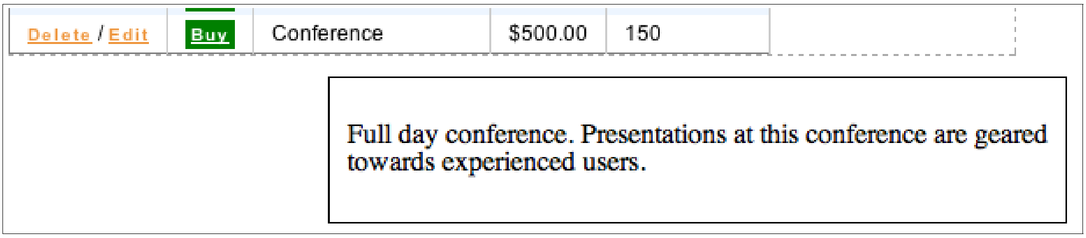


3. Enter a title for this badge in the “Title” field.

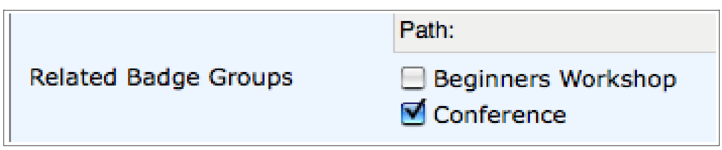
4. The “Menu Title” is the title as it appears in site navigation. You can leave this blank and WebGUI will insert a menu title for you (by default, WebGUI will use the “Title”).

5. The URL can also be left blank and WebGUI will generate a URL for you.

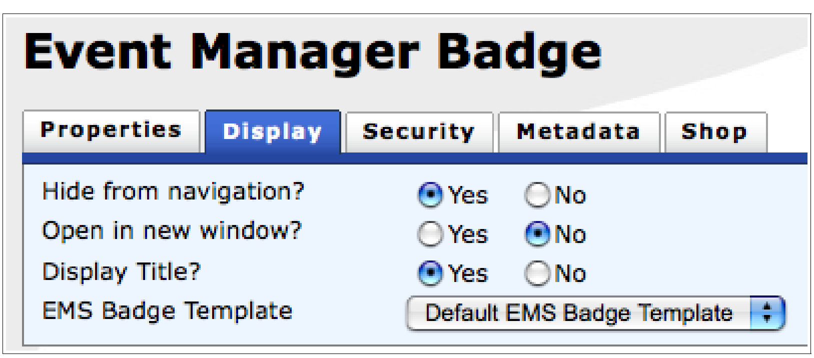
6. In the “Description” field, enter a short description of the badge. Once this badge is created, this text will appear as hover help when the registrant's cursor moves over the badge title in the Event Manager screen, shown below.



7. At the bottom of the screen is the “Related Badge Groups” field. Badge groups are used to associate badges with tickets. Badges can be related to a badge group, and when tickets to events within the badge are created, the tickets are related to the same badge group. When registrants select a badge to purchase they will be directed to a screen that contains all tickets to events in the same badge group. If a registrant purchases a badge, and tickets, and then removes the badge from his/her cart, all the tickets related to that badge will also be removed. See the section on Badge Groups for more information.



8. Choose your display settings for this badge in the “Display” tab. These are the standard display settings common amongst most assets.



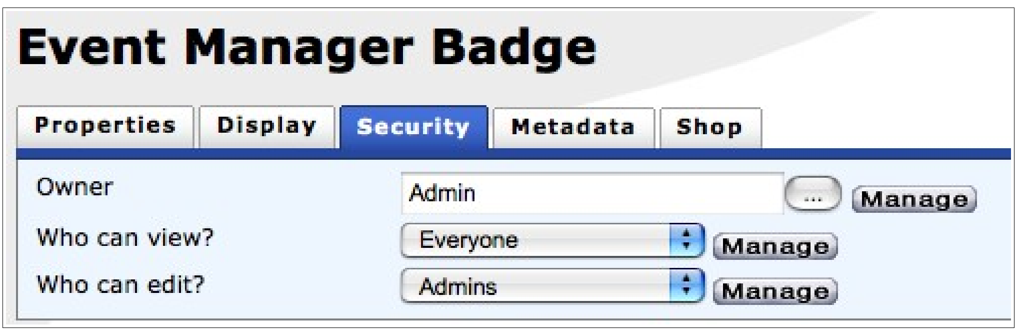
A. If the “Hide from navigation?” field is set to “Yes,” the badge “Menu Title” will not appear in the site navigation or site maps.

B. If “Open in new window?” is set to “Yes,” the badge will open in a new browser window when viewed directly.

C. If “Display Title?” is set to “No,” the title of this badge entered in the “Properties” tab will not be displayed on the web page.

D. The EMS Badge Template is the template used to display badges in the Event Manager. If you have a custom badge template, you apply it here.

9. Choose your security settings from the “Security” tab. Again, these are the standard security settings.



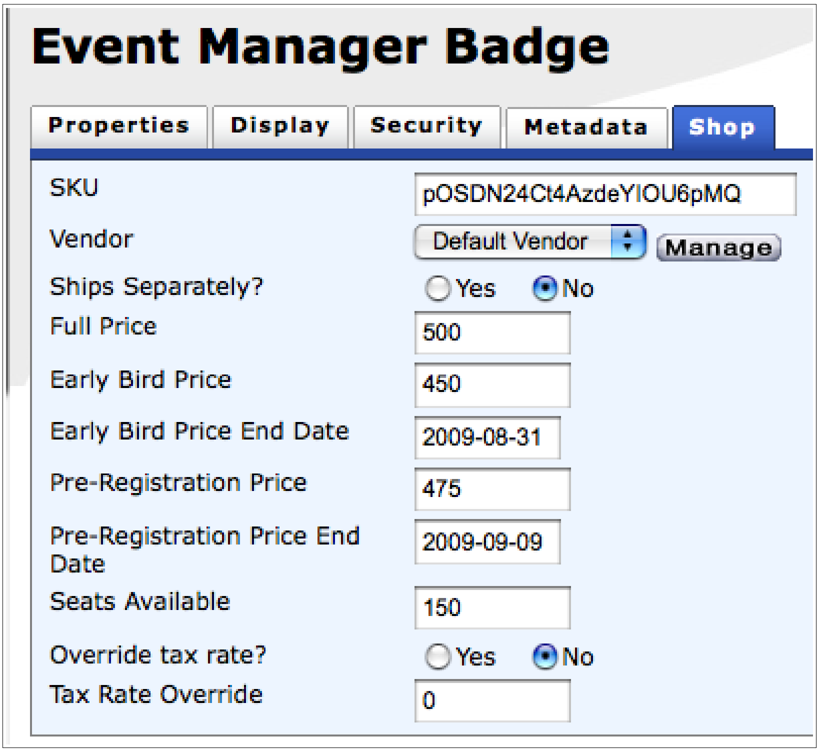
A. The Owner has full editing and viewing rights for this badge. Usually, the Owner is the user who created the badge.

B. The “Who can view?” field allows you to limit the group of users who can view this badge on the site. Select a group of users allowed to view this badge from the dropdown menu. If there are no viewing restrictions, select Everyone in the menu. The Owner can always view the badge. This can be helpful if you want to make badges for the same event available at different prices for certain users, like a discounted price for registered users as opposed to the general public.

C. Select the group who has editing rights for this badge from the “Who can edit?” dropdown menu. The Owner can always edit.

10. If you wish, enter metadata in the “Metadata” tab. The metadata tab is the standard tab seen amongst most assets.

11. Choose your inventory and price settings from the “Shop” tab.



A. “SKU” stands for Stock Keeping Unit. In this field, you can enter an inventory code or product number for this badge, or leave the value created for you.

B. Under the “Vendor” dropdown menu, you can choose a person or company defined in the commerce system to receive credit for selling this badge.

C. If the “Ships Separately?” field is set to Yes, this item will accrue shipping costs separately from other items in your cart. Since a badge is not a physical item to ship, you can probably leave this field set to No.

D. In the “Full Price” field, you can enter the amount to be charged for this badge. This covers admission to this event; tickets will be charged separately.

E. If you are offering an early bird discount, you can enter the early bird price in the “Early Bird Price” field.

F. If you are offering an early bird discount, enter the date on which the discount ends in the “Early Bird Price End Date” field.

G. If you are offering a discounted price for pre-registration to this badge, you can enter that price in the “Pre-Registration Price” field.

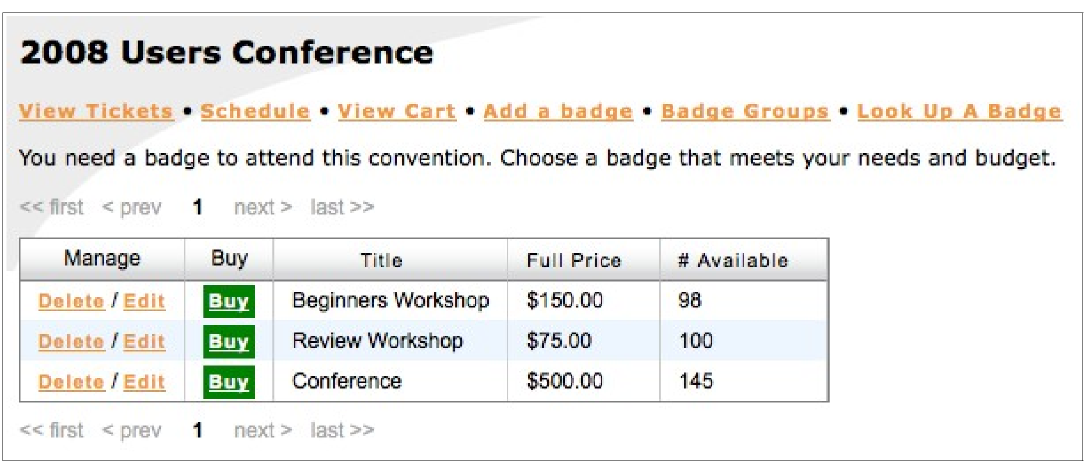
H. If you are offering a discounted price for pre-registration to this badge, enter the date that this offer ends in the “Pre-Registration Price End Date” field.

I. In the “Seats Available” field, you can enter the maximum number of seats to be sold for this event. Each badge that is sold will deduct one seat from this number in the Event Manager screen.

J. If “Override tax rate?” is set to “Yes,” you can override the default tax rate set in the Shop administrative screen. This allows you to apply a different percentage of tax for this badge than you would other items sold through your Shop.

K. In “Tax Rate Override,” enter the new percentage to be used in calculating the tax rate. The rate entered here will only be applied to the badge if “Override tax rate?” is set to “Yes.”

12. When all settings are complete, click “save.” You will be returned to the Event Manager where the badge you just created will be displayed. Members of the Registration Staff Group will see links by which to delete or edit this badge. The green Buy link is what registrants click to purchase this badge. Below, you can see an Event Manager that contains three badges.

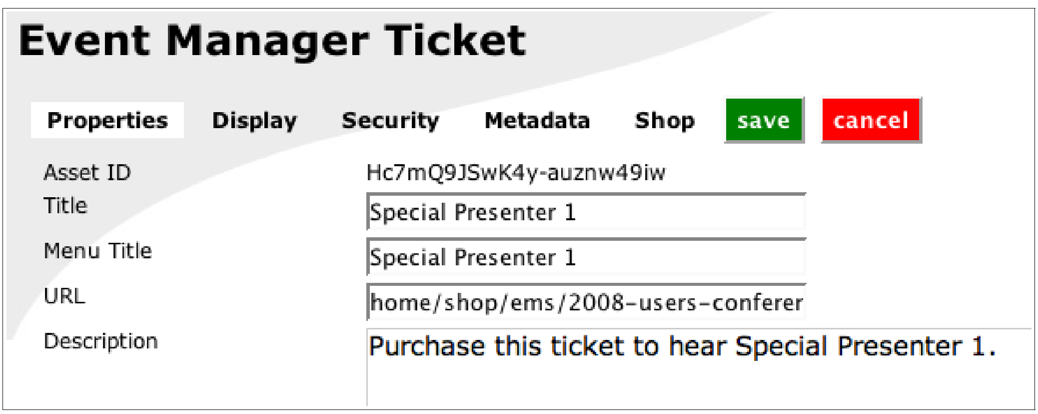


Create Tickets

Tickets are used to sell admission to individual events within a badge. For example, you might have a main event (badge), and at the event are a number of special speakers for which you need to charge admission. In addition, those speakers' presentations may have limited seating. Tickets allow you to sell a limited number of seats to a specific presentation.

1. In the main Event Manager screen, click the link to “View Tickets,” located beneath the asset title and description. The “Tickets” tab will open. The tickets tab is part of a screen called the Badge Builder. A registrant selects a badge to purchase, and then “builds” that badge by making additional purchases from the tabs in this screen.

2. Click the link in the upper left hand corner to “Add a ticket.” The “Event Manager Ticket” screen will open.



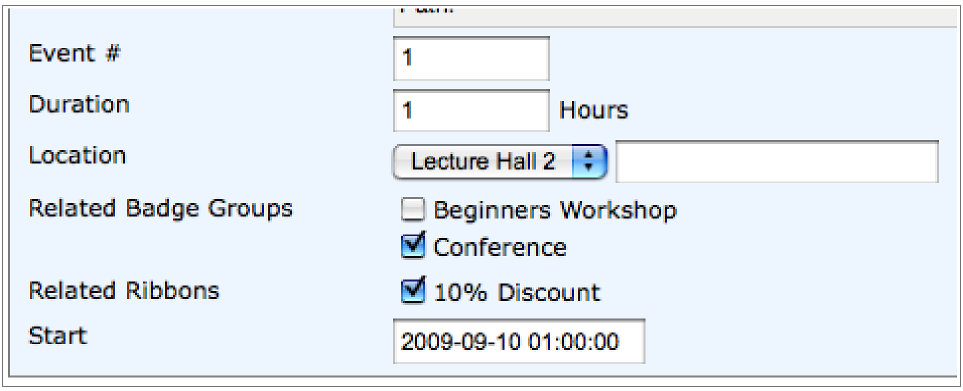
3. Enter a title for this ticket in the “Title” field.

4. The “Menu Title” is the title as it appears in the site navigation. You can leave this blank and WebGUI will insert a menu title for you (by default, WebGUI will use the “Title”).

5. The URL can also be left blank and WebGUI will generate a URL for you.

6. In the “Description” field enter a short description of the ticket. Here is where you could provide instructions for the registrant or a description of the event. This text will appear as hover help when the registrant's cursor moves over the ticket title in the “Tickets” tab.

7. At the bottom of the screen, there is a field to enter an “Event #”. In this field, you can enter a number to represent this event. Each event number should be unique, as it is useful for referring to the event in a catalog or program.



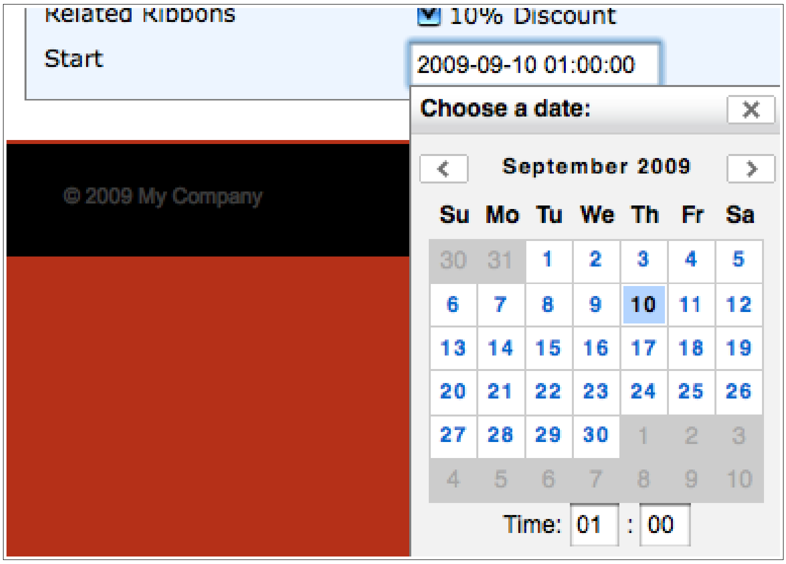
8. In the “Duration” field, you can enter the number of hours the event will last.

9. From the “Location” dropdown menu, select the location where the event will take place. If the event's location is not listed in the dropdown menu, you can enter a new location in the text field to the right of the menu. This new location will appear in future “Event Manager Ticket” screens, under the “Location” dropdown menu.

10. The “Related Badge Groups” field allows you to associate this ticket to a badge group. Tickets can be placed in the same group as a badge. Then, when a user purchases a badge, s/he will be shown the tickets that are related. See the section on Badge Groups for more information.

11. If Ribbons have been created for tickets at this event, they will be listed under the “Related Ribbons” field. Check the box next to a ribbon that can provide a discount for this ticket.

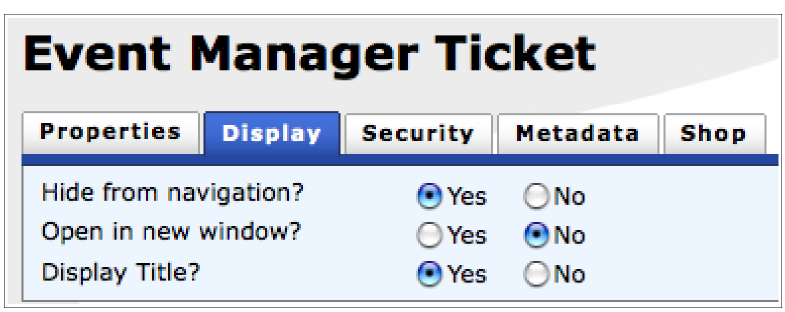
12. In the “Start” field, you can enter the date and time that the event will begin. You may either manually enter the start date and time or select a date and time from the calendar picker.



A. To manually enter the date and time into the “Start” text field, you can enter the date using the format YYYY-MM-DD and the time using the 24-hour clock format HH:MM:SS.

B. To select a date and time from the calendar picker, click anywhere in the “Start” text field, and a calendar will open. Click the arrows to the left or right of the month name to select a month and year, then click to select a date for the event to begin. In the “Time” field located beneath the calendar, you can enter a time for the event to begin, using the 24-hour clock format HH:MM. When you have finished selecting a start date and time, click the “X” located in the upper right corner of the picker.

13. Choose display settings from the “Display” tab.

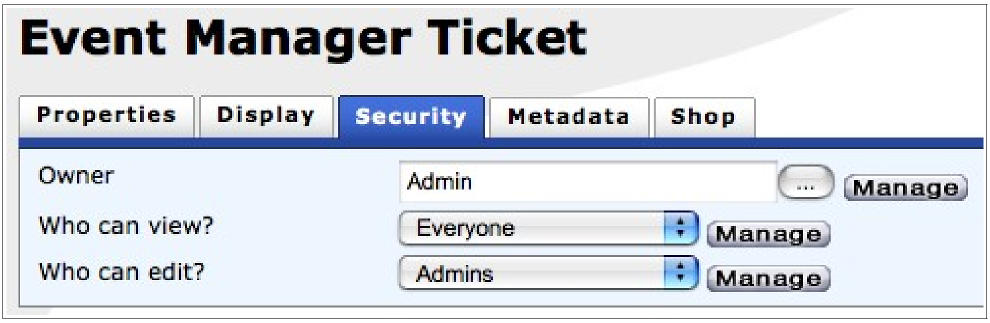


A. If “Hide from navigation?” field is set to “Yes,” the ticket “Menu Title” will not appear in the site navigation or site maps.

B. If “Open in new window?” is set to “Yes,” the ticket will open in a new browser window when viewed independently.

C. If “Display Title?” is set to “No,” the title of the ticket entered in the “Properties” tab will not be displayed to registrants.

14. Choose security settings from the “Security” tab.



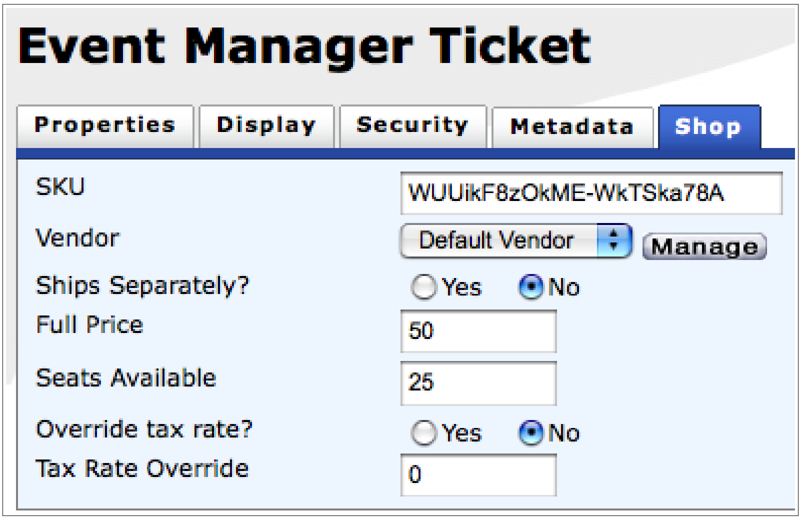
A. The Owner has full editing and viewing rights for this ticket. Usually, the owner is the user who created the ticket.

B. The “Who can view?” field allows you to limit who can view this ticket. Select a group of users allowed to view this ticket from the dropdown menu. If there are no viewing restrictions, leave the menu at Everyone. The Owner can always view the ticket.

C. Select who has editing rights for this ticket from the “Who can edit?” dropdown menu. The Owner can always edit.

15. If you wish, enter metadata in the “Metadata” tab. The metadata tab is the standard tab seen amongst all assets. Tickets in an Event Manager can also be assigned their own Meta Fields. Those fields will be displayed at the bottom of the Metadata tab. See the section on Ticket Meta Fields for more information.

16. Choose your inventory and price settings from the “Shop” tab.



A. "SKU" stands for Stock Keeping Unit. In this field, you can enter an inventory code or product number for this ticket.

B. Under the "Vendor" dropdown menu, you can choose a person or company defined in the commerce system to receive credit for selling this ticket.

C. If "Ships Separately?" is set to Yes, this item will accrue separate shipping costs in addition to other items in your cart. Since this is not a physical item to ship, this field can be left at No.

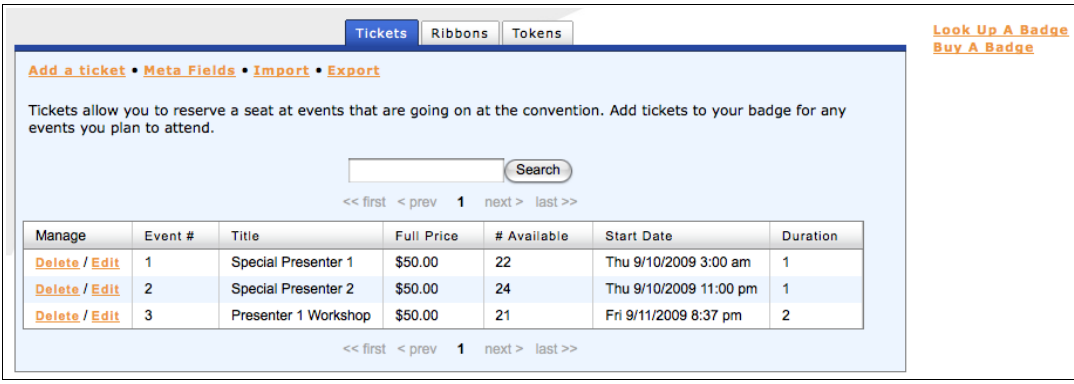
D. In the "Full Price" field, you can enter the amount to be charged for this ticket.

E. In the "Seats Available" field, you can enter the maximum number of seats to be sold for this ticket. Each ticket that is sold will deduct one available seat from this ticket. If a ticket is returned, the seat will be replenished.

F. If "Override tax rate?" is set to "Yes," you can override the default tax rate set in the Shop administrative screen. This allows you to apply a different percentage of tax for this ticket than you would other items sold through your Shop.

G. In "Tax Rate Override," enter the new percentage to be used in calculating the tax rate. The rate entered here will only be applied to the ticket if "Override tax rate?" is set to "Yes."

17. When all settings are complete, click save. After saving, you will be taken back to the “Tickets” tab where you can continue creating additional tickets.

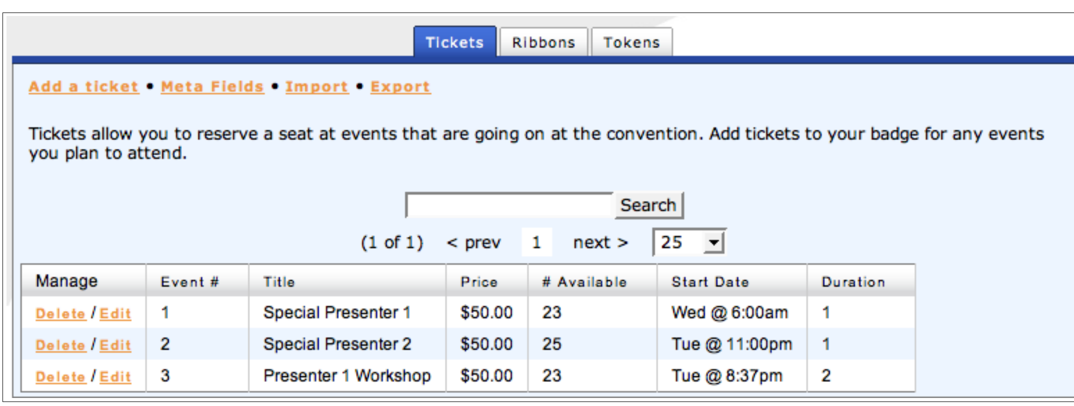


Manage	Event #	Title	Full Price	# Available	Start Date	Duration
Delete / Edit	1	Special Presenter 1	$50.00	22	Thu 9/10/2009 3:00 am	1
Delete / Edit	2	Special Presenter 2	$50.00	24	Thu 9/10/2009 11:00 pm	1
Delete / Edit	3	Presenter 1 Workshop	$50.00	21	Fri 9/11/2009 8:37 pm	2

Ticket Meta Fields

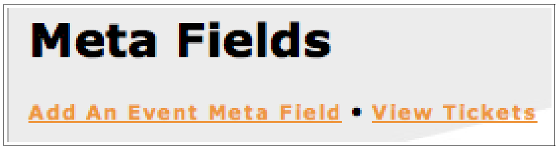
Upon entering the Tickets tab, a link will be provided by which you may create Meta Fields for the purpose of assigning metadata to tickets in this Event Manager. Meta fields created in this way will only appear for Ticket assets. To create Meta Fields for tickets:

1. Enter the Tickets screen by selecting View Tickets in the Event Manager.

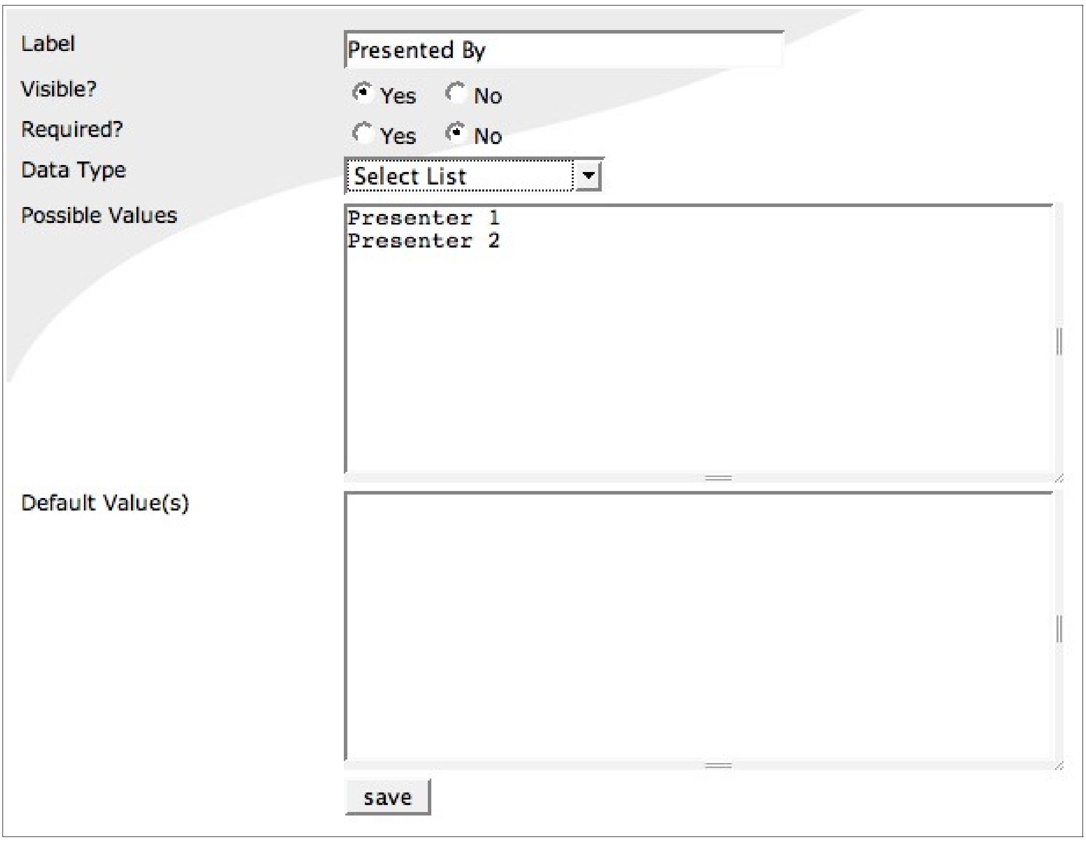


Manage	Event #	Title	Price	# Available	Start Date	Duration
Delete / Edit	1	Special Presenter 1	$50.00	23	Wed @ 6:00am	1
Delete / Edit	2	Special Presenter 2	$50.00	25	Tue @ 11:00pm	1
Delete / Edit	3	Presenter 1 Workshop	$50.00	23	Tue @ 8:37pm	2

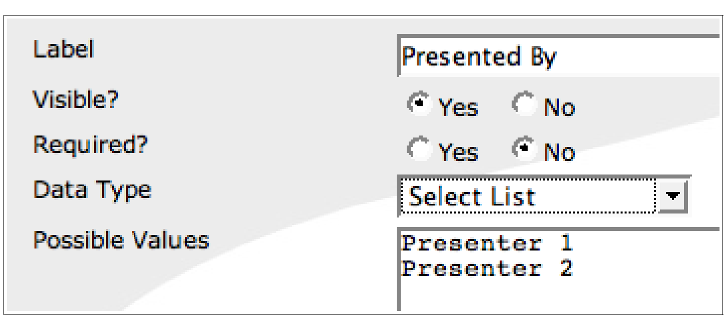
2. At the top of the Tickets tab are a number of links. The second from the left is Meta Fields. Click on the Meta Fields link to open the Meta Fields screen.



3. Click on the “Add An Event Meta Field” link to open the screen on which you can create a new meta field.



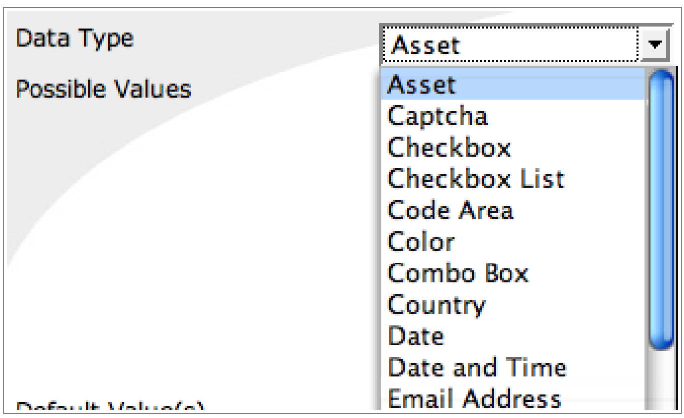
4. In the “Label” field, enter the label for this field. This is the label the user will see on the screen.



5. Set the “Visible?” toggle to Yes or No to determine if this field will be visible.

6. Set the “Required?” toggle to Yes or No to determine if this meta field will be required.

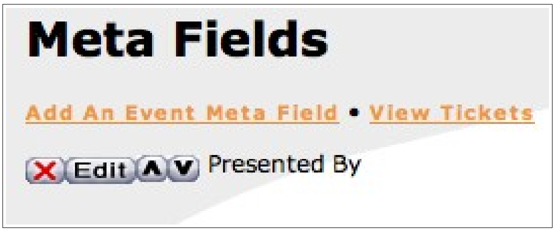
7. Select the “Data Type” for this field from the dropdown menu. WebGUI will validate the data input by the user to ensure it matches the data type you select. For example, if “Email Address” is selected, and the user enters something other than an email address, an error message will be displayed.



8. The “Possible Values” field is where you enter the values from which users may select. Each value is separated by a carriage return. This field is only used with the following Data Types: Checkbox List, Combo Box, Hidden List, Radio List, Select Box, and Select List.

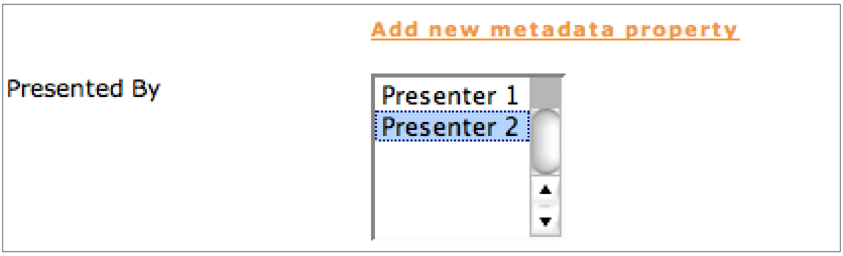
9. If a Data Type was chosen that requires a list of values for the user to choose from, you may wish to set a “Default Value.” This is the value that will be pre-selected for the user upon viewing the meta field. This is useful if there is one value that is far more common than the others, as it will be automatically selected for you. A default value must directly reference one or more of the values entered in the Possible Values field.

10. Click save to create the meta field. You will be returned to the Meta Fields screen, where the meta field you just created will be displayed, along with a toolbar from which you may delete the field or edit it. The “View Tickets” screen will take you back to the Tickets tab.



To apply this meta field to a Ticket, enter a ticket's Metadata tab. This can be done at the time of Ticket creation, or by editing an existing Ticket.

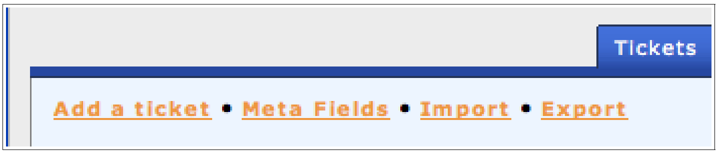
The meta fields you created will be displayed at the bottom of the Ticket's Metadata tab.



Import and Export Tickets

At times, you may find it necessary to manage tickets in an external spreadsheet application. The Event Manager will allow you to export existing Ticket data, and then import it again. Upon importing the data, the Event Manager will update the existing Tickets on the site to reflect any changes in the imported data.

Upon entering the View Tickets screen, you will see an Import and Export link at the top.

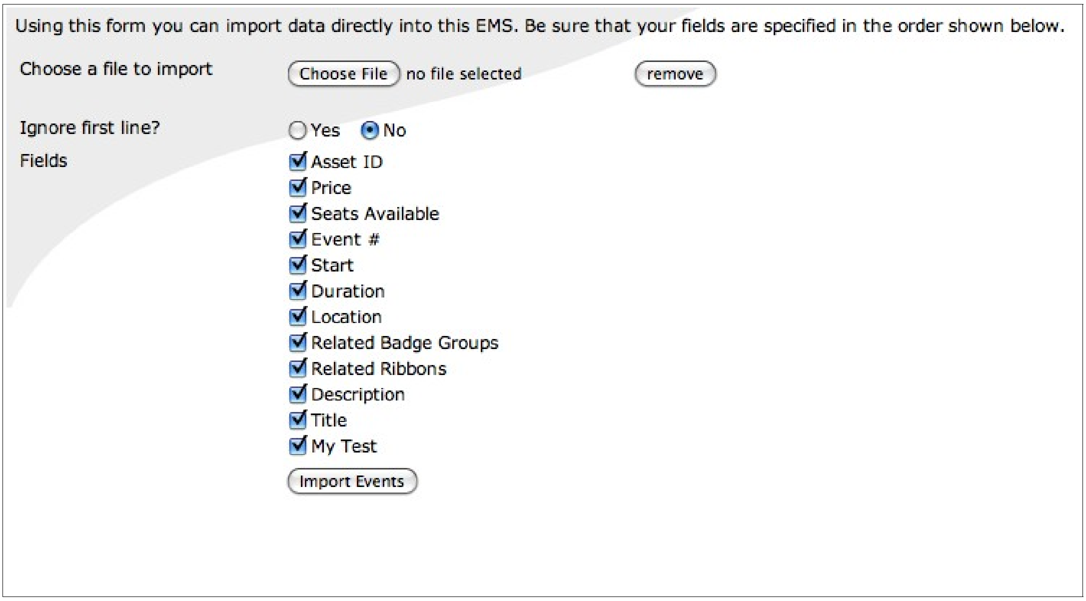


To export existing Ticket data, click on the Export link. The Event Manager will generate a .csv file for you to save on your computer, seen below.

```
"Asset ID",Price,"Seats Available","Event #",Start,Duration,Location,"Related Badge Groups","Related Ribbon
IQU25_7HAI9zT2KU-DGxgw,142,25,0,"2008-07-27 22:12:13",1,,,,"<p>my description</p>","This is",0
Uonf_i0LUyTGCuTAltbTWg,50,80,45,"2008-07-27 10:00:00",1.5,"The Big Room",,,"Big Dog","Like It",0
```

This data can then be used in an another application. Using the export function will give you a base import file to work from in an external application.

To import Ticket data back into the Event Manager, click on the Import link in the View Tickets screen. You will be presented with a screen that allows you to configure the import.



If no "Asset ID" is specified for a particular row in the import file, then a new Ticket will be created. If the assetId is specified, then an existing ticket will be updated to reflect any changes in data. You can optionally leave out fields from your file if you only want to import data for certain fields.

Badge Groups

Badge Groups are used to relate events to each other. Specifically, a Badge Group allows you to relate tickets to a specific badge in the Event Manager. A Badge Group is created, and a specific badge and a number of related tickets are associated with that group. When registrants purchase a badge, they are shown all tickets available for purchase at that event. The Event Manager knows which tickets to show the user based on their common association with a Badge Group.

1. Click the “Badge Groups” link located beneath the Event Manager's title and description. The “Badge Groups” screen will open.

2. Click the link to “Add a Badge Group.”

3. In the “Badge Group Name” field, enter a name for this badge group. The Badge Group Name will appear in the “Related Badge Groups” field when creating Tickets and Badges.

4. You will be taken back to the main “Badge Groups” screen. The Badge Group you created will be displayed, along with links by which to delete and edit existing Badge Groups. From this screen, you may either “Add A Badge Group,” or “View Badges” you've created.

   A. If you choose to add another badge group, you will be taken to a “Badge Groups” screen with a fresh “Badge Group Name” field. Repeat the same process to add a badge group to your Event Manager.

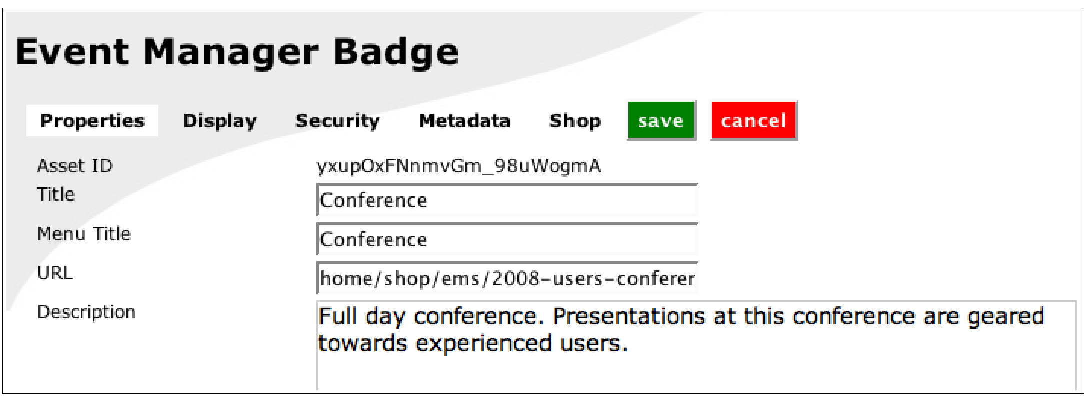
   B. If you do not wish to add any more badge groups to your Event

Manager, choose “View Badges.” You will be returned to the main Event Manager screen, where you can view the badges you have created.

Associate Badges and Tickets with a Badge Group

Now that a badge group has been created, you can associate badges and tickets within your Event Manager with this badge group. If you know you need to associate tickets and badges early on, you may want to create all necessary badge groups first, then move on to creating badges and tickets.

1. Click the link to “Add a badge,” located beneath the Event Manager asset title and description. Or, click the Edit link to edit an existing badge. In this example, an existing badge was edited.



2. The “Event Manager Badge” screen will open. If this is a new badge, enter the appropriate information; if this is an existing badge, proceed to number 3.

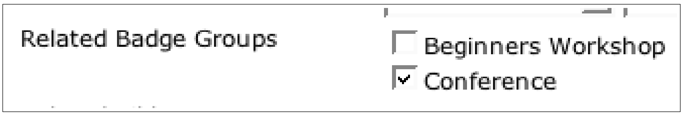
3. At the bottom of the screen, in the “Properties” tab, is the “Related Badge Groups” field. Now that a badge group(s) has been added to your event, it will be displayed here. You may check the box to add this badge to a badge group you have created. Continue through the remaining tabs in the “Event Manager Badge” screen as instructed in the previous section.

4. After saving, you will return to the Event Manager. Next, you need to associate tickets with the same Badge Group as this badge.

5. Click on the “View Tickets” link near the top of the Event Manager asset.

6. Upon entering the Tickets tab, click the “Add a ticket” link to create a new ticket, or click the Edit link of an existing ticket. See the section on creating Tickets to learn more about Tickets.

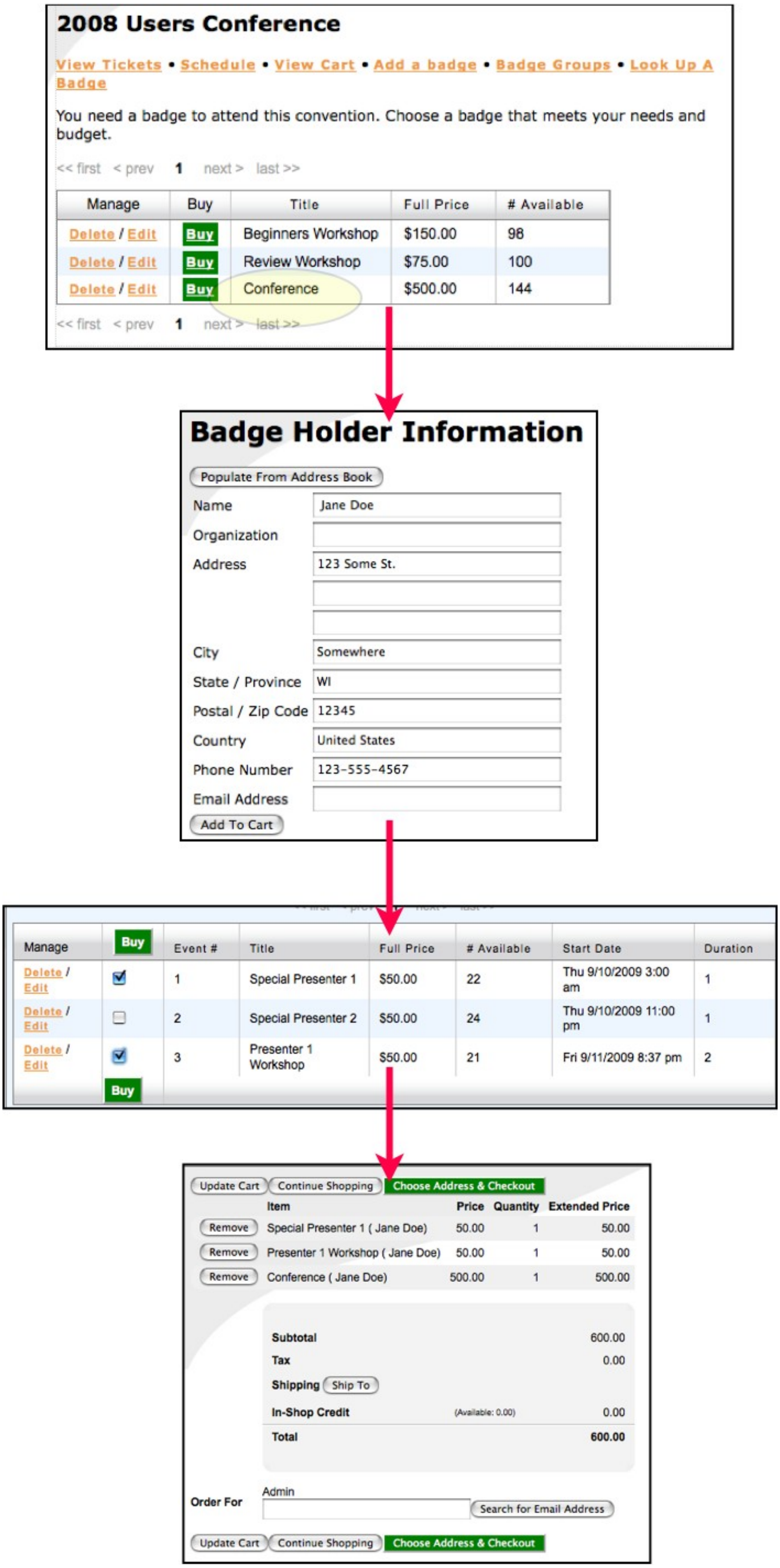
7. In the Event Manager Ticket screen, select the appropriate badge in the “Related Badge Groups” screen.



8. Save the ticket.

In this example, a number of tickets were created, and each was assigned “Conference” in the “Related Badge Groups” field. In addition, the Conference badge was assigned the same related badge group.

If a registrant selects the Conference badge for purchase, s/he is asked to fill in some personal information. After clicking the Add to Cart button, the registrant is then directed to a screen on which all tickets contained within the same badge group are displayed. From the tickets screen, the registrant selects which individual tickets s/he wants to add to the purchase and clicks the green Buy button. After doing so, the Conference badge, along with the individual tickets, are added to the registrant's shopping cart. The images below show this process.

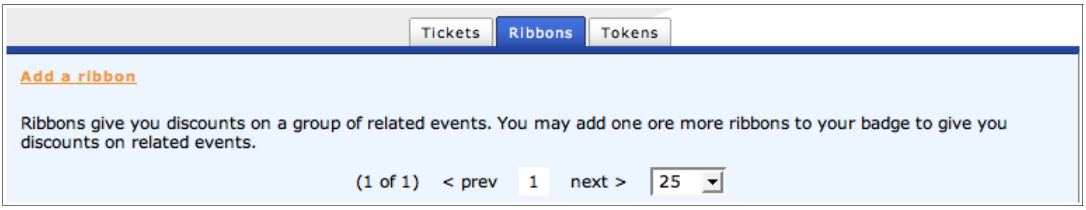

2008 Users Conference
View Tickets • Schedule • View Cart • Add a badge • Badge Groups • Look Up A Badge
You need a badge to attend this convention. Choose a badge that meets your needs and budget.
<< first < prev 1 next > last >>
Manage | Buy | Title | Full Price | # Available
Delete / Edit | Buy | Beginners Workshop | $150.00 | 98
Delete / Edit | Buy | Review Workshop | $75.00 | 100
Delete / Edit | Buy | Conference | $500.00 | 144
<< first < prev 1 next > last >>
Badge Holder Information
Populate From Address Book
Name: Jane Doe
Organization
Address: 123 Some St.
City: Somewhere
State / Province: WI
Postal / Zip Code: 12345
Country: United States
Phone Number: 123-555-4567
Email Address
Add To Cart
Manage | Buy | Event # | Title | Full Price | # Available | Start Date | Duration
Delete / Edit | 1 | Special Presenter 1 | $50.00 | 22 | Thu 9/10/2009 3:00 am | 1
Delete / Edit | 2 | Special Presenter 2 | $50.00 | 24 | Thu 9/10/2009 11:00 pm | 1
Delete / Edit | 3 | Presenter 1 Workshop | $50.00 | 21 | Fri 9/11/2009 8:37 pm | 2
Buy
Update Cart | Continue Shopping | Choose Address & Checkout
Item | Price | Quantity | Extended Price
Remove | Special Presenter 1 (Jane Doe) | 50.00 | 1 | 50.00
Remove | Presenter 1 Workshop (Jane Doe) | 50.00 | 1 | 50.00
Remove | Conference (Jane Doe) | 500.00 | 1 | 500.00
Subtotal 600.00
Tax 0.00
Shipping Ship To
In-Shop Credit (Available: 0.00) 0.00
Total 600.00
Order For Admin
Search for Email Address
Update Cart | Continue Shopping | Choose Address & Checkout

A user can remove a ticket from the cart, and the seat being taken up by that ticket will be replaced and made available once again for purchase. If a user removes a badge from the cart, all related tickets will be removed as well. So, a user can not attend an event purchased through a ticket without purchasing a badge for full event admission as well. In essence, the badge is a prerequisite to the ticket.

Create Ribbons

Ribbons allow discounts to be applied to tickets. A ribbon is created, and then associated with tickets in a ticket's Properties tab. When tickets and a corresponding ribbon are placed in the registrant's shopping cart, the ribbon's discount will be applied to the price of the tickets.

1. Select the “Ribbons” tab located to the right of the “Tickets” tab.
2. The “Ribbons” tab will be displayed.



3. Click the link to “Add a ribbon.” The “Event Manager Ribbon” screen will open.

4. Enter a title for the ribbon in the “Title” field.
5. The “Menu Title” is the title as it appears in the site navigation. You can leave this blank and WebGUI will insert a menu title for you (by

default, WebGUI will use the "Title").

6. The URL can also be left blank and WebGUI will generate a URL for you.

7. In the "Description" field, enter a short description of the ribbon. Here is where you could provide instructions for the registrant. This text will appear as hover help when the registrant's cursor moves over the ribbon title in the "Ribbons" tab.

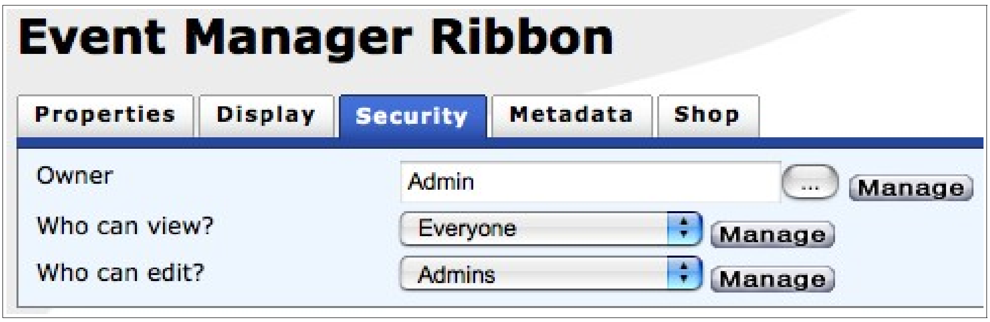
8. Choose display settings in the "Display" tab.

   A. If the "Hide from navigation?" field is set to "Yes," the ribbon "Menu Title" will not appear in the navigation.

   B. If "Open in new window?" is set to "Yes," the ribbon will open in a new browser window when directly viewed.

   C. If "Display Title?" is set to "No," the title of the ribbon entered in the "Properties" tab will not be displayed.

9. Choose security settings from the "Security" tab.



   A. The Owner has full editing and viewing rights for this ribbon.

Usually, the Owner is the user who created the ribbon.

B. The “Who can view?” field allows you to limit who can view this ribbon. Select a group of users allowed to view this ribbon from the dropdown menu. If there are no viewing restrictions, leave the menu at Everyone.

C. Select who has editing rights for this ribbon asset from the “Who can edit?” dropdown menu. The Owner can always edit.

10. If you wish, enter metadata in the “Metadata” tab. The metadata tab is the standard tab seen amongst most assets.

11. Choose your inventory and price settings from the “Shop” tab.

A. “SKU” stands for Stock Keeping Unit. In this field, you can enter an inventory code or product number for this ribbon.

B. Under the “Vendor” dropdown menu, you can choose a person or company defined in the commerce system to receive credit for selling this ribbon.

C. If the “Ships Separately?” field is set to Yes, this item will accrue additional shipping charges, separate from other items in the cart. Since this is not a physical item to ship, this can be left at No.

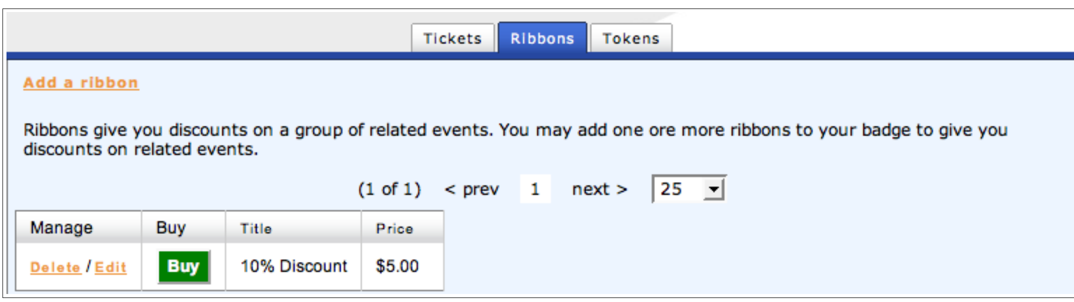
D. In the “Full Price” field, you can enter the amount to be charged for this ribbon.

E. In the “Percentage discount” field, enter the percentage discount to be applied to tickets if the user adds this ribbon to the cart.

F. If “Override tax rate?” is set to “Yes,” you can override the default tax rate set in the Shop administrative screen. This would allow you to apply a different percentage of tax for this ribbon than you would other items sold through your Shop.

G. In “Tax Rate Override,” enter the new percentage to be used in calculating the tax rate. The rate entered here will only be applied to the ribbon if “Override tax rate?” is set to “Yes.”

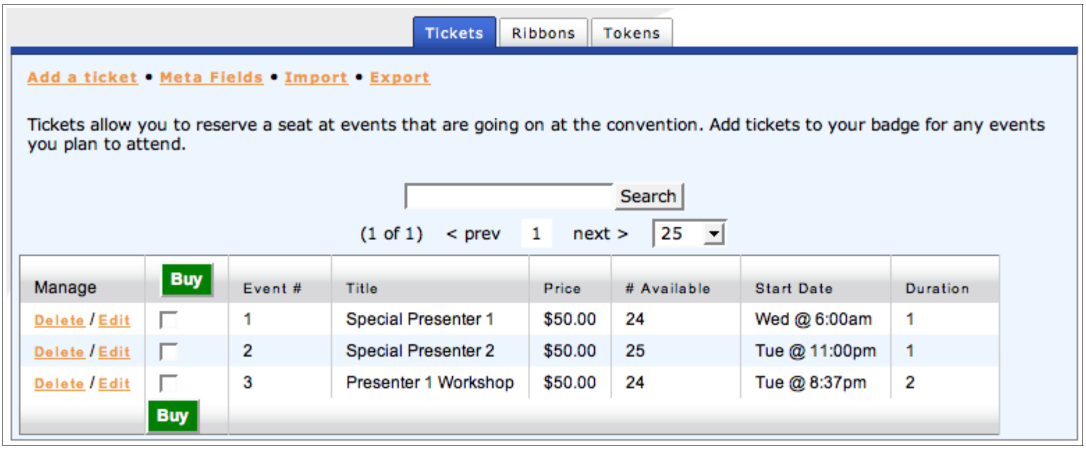
12. When all settings are complete, click save. After saving, you will be taken back to the “Ribbons” tab. A registrant will click a ribbon's green Buy button to place the ribbon in his/her cart.



Associate Ribbons with Tickets

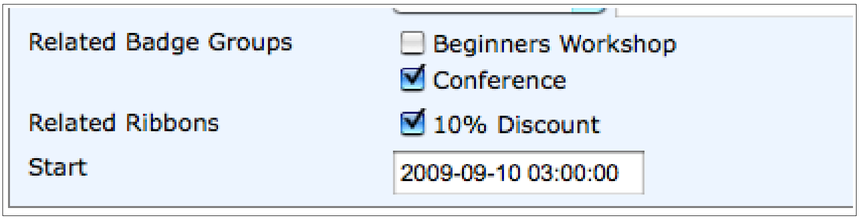
Now that a ribbon has been created for this event, you can offer discounts on a group of tickets by associating the ribbon with tickets in the “Tickets” tab.

1. Click the “Tickets” tab, located to the left of the “Ribbons” tab. The “Tickets” tab will open.



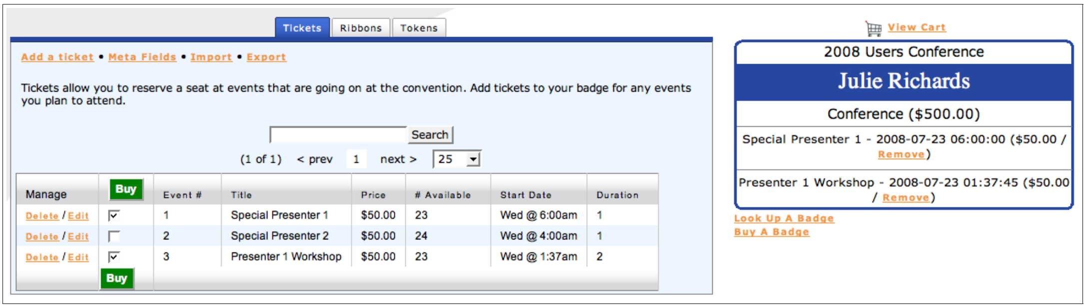
2. Click the link to "Add a ticket." The "Event Manager Ticket" screen will open. Or, edit an existing ticket. See the section on creating tickets for more information.

3. At the bottom of the ticket's "Properties" tab is the "Related Ribbons" field. Ribbons created for tickets at this event will be displayed here. You may check the box to associate a ribbon with this ticket.

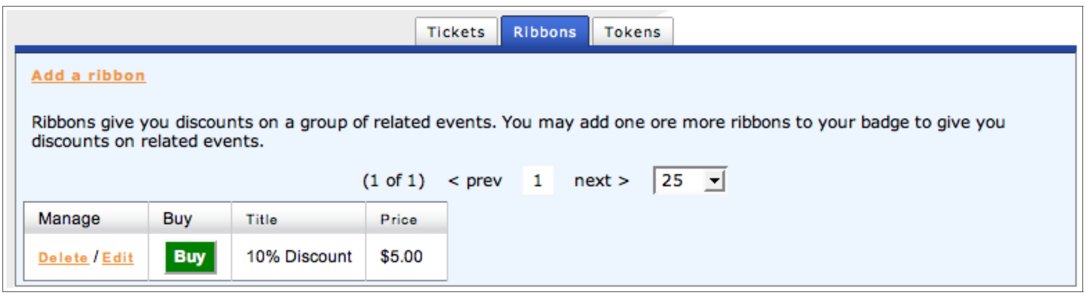


4. When all your settings are complete, click save. After saving, you will be taken back to the "Tickets" tab.

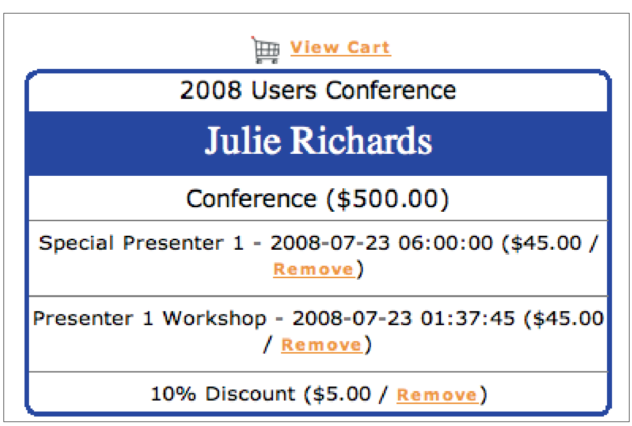
After registrants place badges in their carts, they will be directed to the tickets screen. Registrants can then select which tickets they wish to purchase. In the screenshot below, the registrant has selected two tickets, each priced at $50. The registrant's badge is displayed along with the ticket purchases on the right hand side of the screen.



Then, the registrant enters the Ribbons tab to purchase a Ribbon. The registrant clicks on the green Buy button to place the ribbon in her cart.



Once the ribbon has been added to the registrant's cart, the ticket amounts will be adjusted to reflect the discount provided by the ribbon. In the screenshot below, you can see that the registrant has added the ribbon to the badge, and the ticket prices have each been reduced by 10% ($5 dollars off the $50 ticket prices, for a final cost of $45 dollars a piece). In this case, the registrant purchased a $5 ribbon in order to receive $10 off her total cost, for an overall savings of $5.



Create Tokens

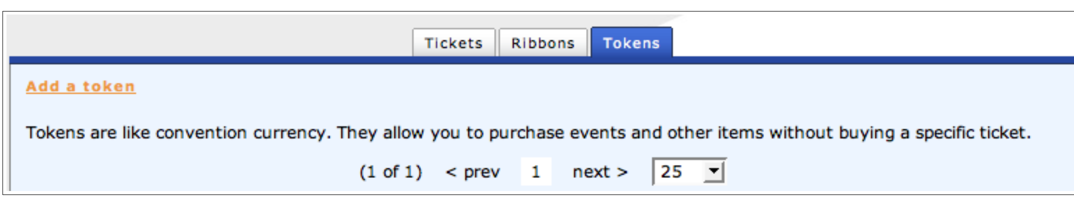
Tokens can serve two main purposes:

- Registrants can buy tokens during registration. Upon checking in at the event, the registrant will be provided physical tokens by the hosting organization. These tokens can then be used like currency throughout the event to make purchases, or to buy access to ticketed events that still have available seats.

- A Token asset can be created to represent a physical good associated with this event. For example, this event may have a special T-shirt that is only available for paying attendees. A Token asset is created, and given an appropriate title, such as 2008 Convention T-Shirt. When the registrant adds the convention badge to his/her cart, s/he will be directed to the badge building screen where tickets can be purchased. When the registrant enters the Tokens tab, the T-shirt will be displayed for sale.

To create Tokens:

1. While viewing the tickets for sale for an event, click the “Tokens” tab. The “Tokens” tab will be displayed.



2. Click the link to “Add a token.” The “Event Manager Token” screen will open.

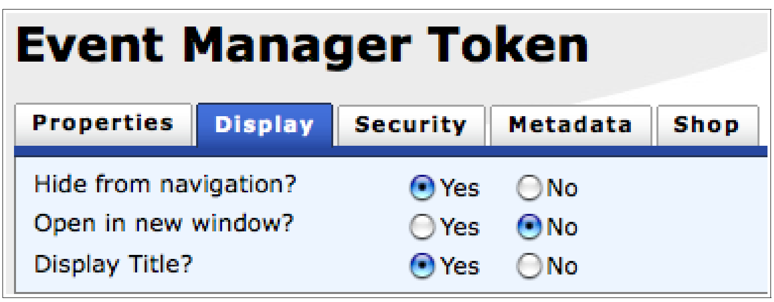
3. Enter a title for this token in the “Title” field. This is the name of the token as the registrant will see it in the Tokens tab. If using this for event currency, you can simply label this “Tokens”; however, if you want to sell a physical good (T-Shirts, totes, key chains, coffee mugs, bobble heads...) associated with the event, put the item's name in the title field.

4. The “Menu Title” is the title as it appears in the site navigation. You can leave this blank and WebGUI will insert a menu title for you (by default, WebGUI will use the “Title”).

5. The URL can also be left blank and WebGUI will generate a URL for you.

6. In the “Description” field, enter a short description of the token. Here is where you could provide instructions for the user. This text will appear as hover help when the user's cursor moves over the token title in the “Tokens” tab.

7. Choose display settings from the “Display” tab.

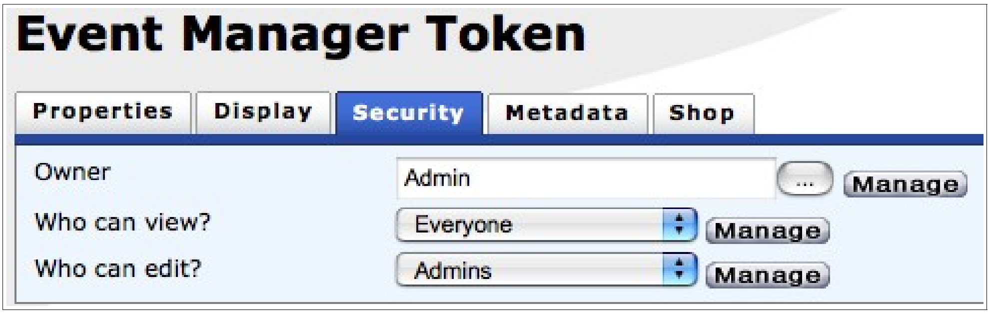


   A. If the “Hide from navigation?” field is set to “Yes,” the token's “Menu Title” will not appear in the site navigation.

   B. If “Open in new window?” is set to “Yes,” the token will open in a new browser window when directly viewed.

   C. If “Display Title?” is set to “No,” the title of this token will not be displayed.

8. Choose security settings in the “Security” tab.



   A. The owner has full editing and viewing rights for this token. Usually, the owner is the user who created the token.

   B. The “Who can view?” field allows you to limit the group of users who can view this token. Select a group of users allowed to view this token from the dropdown menu. If there are no viewing restrictions, leave the menu at Everyone. The Owner can always view the token.

   C. Select who has editing rights for this token from the “Who can edit?” dropdown menu. The Owner can always edit.

9. If you wish, enter metadata in the “Metadata” tab. The metadata tab is the standard tab seen amongst most assets.

10. Choose your inventory and price settings in the “Shop” tab.

   A. “SKU” stands for Stock Keeping Unit. In this field, you can enter an inventory code or product number for this token.

B. Under the “Vendor” dropdown menu, you can choose a person or company defined in the commerce system to receive credit for selling this token.

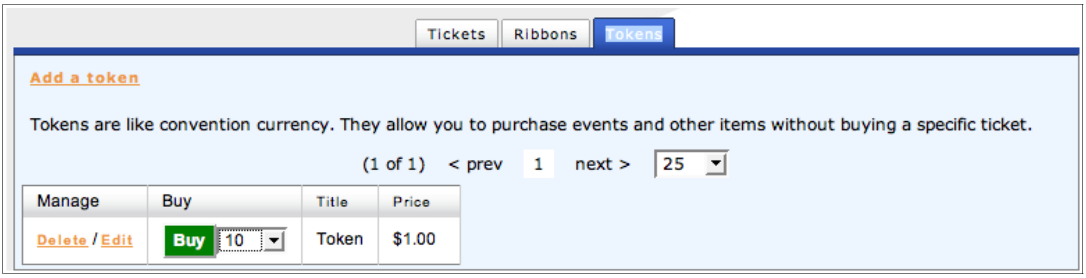
C. If “Ships Separately?” is set to Yes, this item will accrue additional shipping costs, separate from other items in the cart. Most organizations hand out tokens at event registration, so this can probably be set at No.

D. In the “Full Price” field, you can enter the amount to be charged for this token.

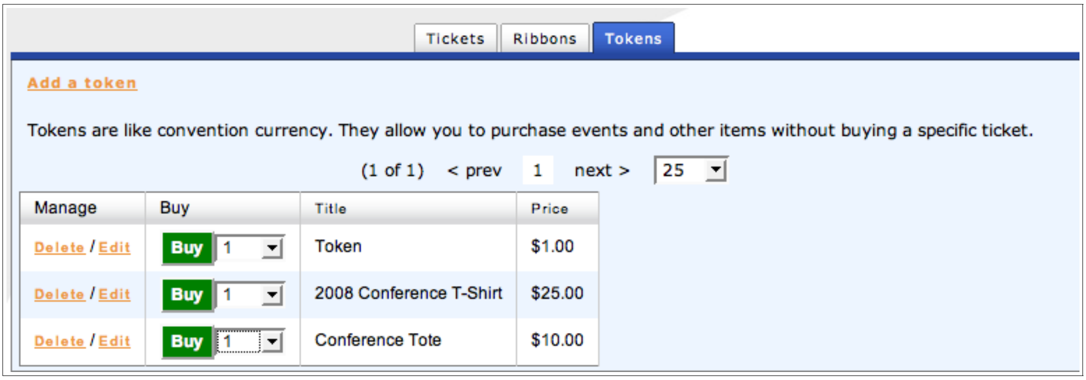
E. If “Override tax rate?” is set to “Yes,” you can override the default tax rate set in the Shop administrative screen. This would allow you to apply a different percentage of tax for this token than you would other items sold through your Shop.

F. In “Tax Rate Override,” enter the new percentage to be used in calculating the tax rate. The rate entered here will only be applied to the token if “Override tax rate?” is set to “Yes.”

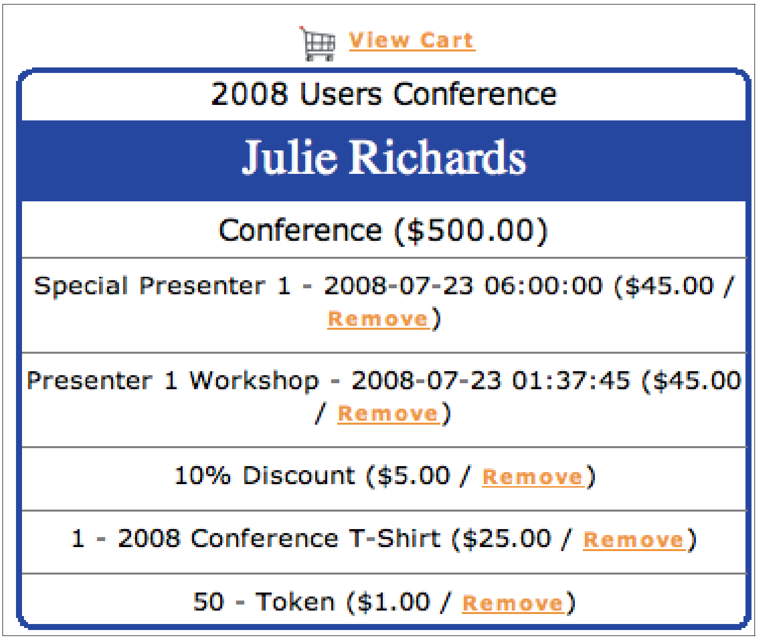
11. When all your settings are complete, click “save.” After saving, you will be taken back to the “Tokens” tab.



Repeat these steps to create additional Token assets. The screenshot below shows that registrants can purchase actual tokens, which will be provided at the event by the hosting organization to use as currency, as well a number of physical goods available for purchase with this badge.



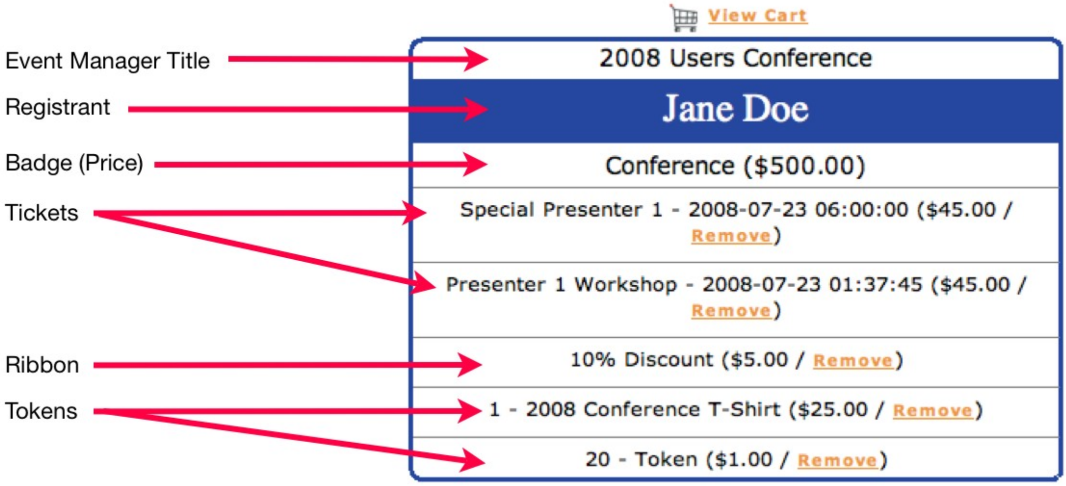
When registrants purchase a badge, they will be directed to a screen displaying tickets associated with that badge and available for purchase. The registrant can then enter the Tokens tab to purchase tokens. To purchase, the user simply selects the number of tokens s/he would like to purchase from the dropdown menu in the Buy column. In the example above, the registrant has chosen to purchase fifty tokens and one T-Shirt. After selecting the amount to buy, the user clicks on the green Buy button to place the tokens in the shopping cart. In the screenshot of the registrant's badge, below, you can see that Julie Richards is attending the 2008 Users Conference badge, and has purchased two tickets and one ribbon. In addition, Julie is purchasing fifty tokens and one T-Shirt.



Registrants may purchase more than one badge. After completing the badge builder screen (Tickets, Ribbons and Tokens), the registrant can click on the Buy A Badge link located under the current badge on the right hand side of the screen. When the registrant has made all his/her selections, s/he simply clicks the View Cart link to complete the full purchase.

Anatomy of a Badge

As a user adds items to the cart in the Event Manager, a special badge displays on the page. This badge essentially displays the registrant's current badge, along with all items purchased in that badge.



At the top is the traditional “View Cart” link to direct the registrant to his/her shopping cart to complete the purchase. The main body of the badge shows the name of the Badge followed by the registrant. In this example, 2008 Users Conference and Jane Doe. Below the registrant name is the name of the Badge Group with the price of the Badge in parentheses. In this example, the 2008 Users Conference badge costs $500. The 2008 Users Conference badge, Special Presenter 1 ticket and Presenter 1 Workshop ticket are all part of the Conference badge group. Registrants may remove an individual ticket from their carts by clicking on its Remove link in the badge. This registrant has also purchased a Ribbon, which was related to the Special Presenter 1 ticket only, indicated by the ticket's reduced price. Tokens have been placed in the cart, and the hosting organization will provide the registrant physical tokens to use as currency at this event. These may also be removed by using the Remove link.

When the registrant is ready, s/he can click on the View Cart link to be directed to the shopping cart where the purchase can be completed. See the chapter on the shopping cart for more information.

Manage Registrants

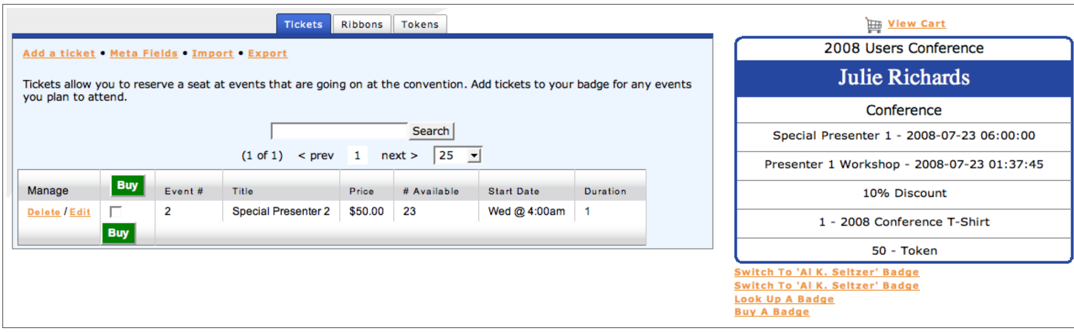
Members of the Staff Registration Group will be allowed to manage registrants for the purpose of checking them in, refunding purchases, adding additional events, or printing badges. To manage individual registrants, begin by looking up a registrant's badge.

Look Up A Badge

Look up a badge to find a specific registrant's information, as well as manage that individual registrant. To access registrant badges, click on the "Look Up A Badge" link in the Event Manager. In the main Event Manager screen this is located along with a number of other links at the top of the asset. In the View Tickets screen, this link is located on the right hand side of the screen.

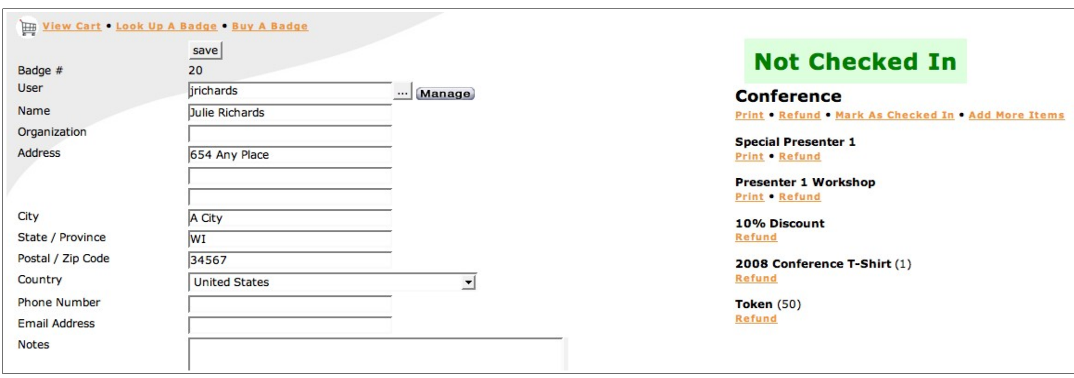
Upon entering the badge look-up screen, all registrants will be listed. In this example, very few registrants exist. In the event that you have many registrants, results will be paginated. The Search field in the upper right hand corner can be used to limit the results displayed on the screen. You can enter a registrant's first or last name to narrow down the registrants listed. The number menu, below the Search field, allows you to set how many registrants you want displayed.

The second column to the right is the Badge # column. Clicking on a registrant's badge number will direct you to the badge builder screen for that registrant.



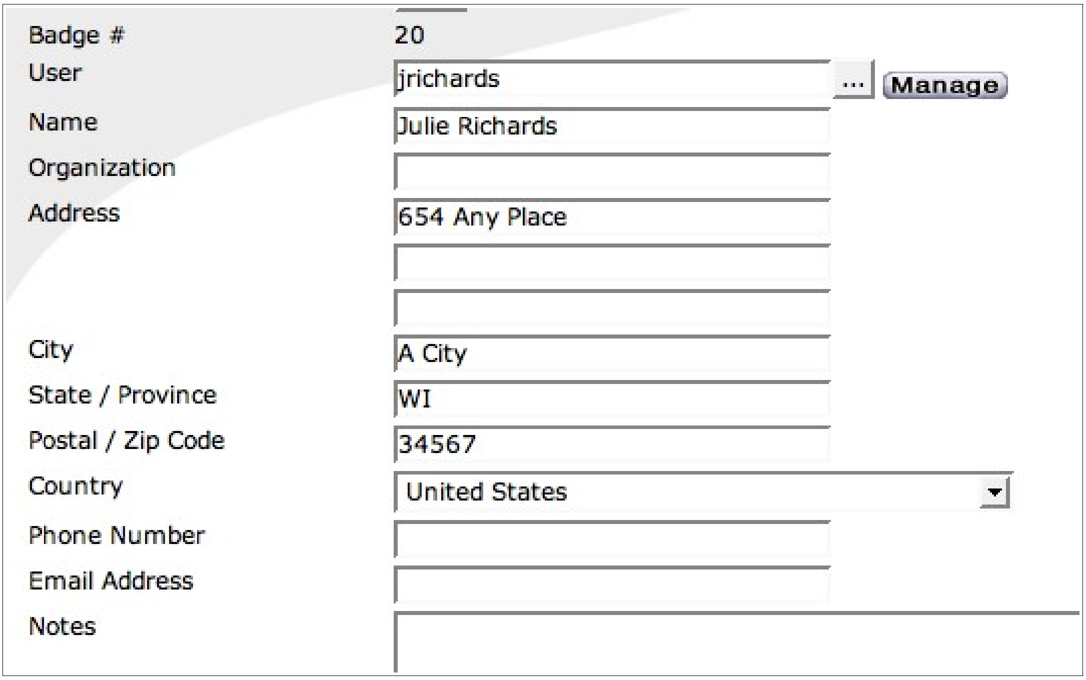
From here you can add additional tickets, ribbons or tokens to this registrant's badge. If there are other open badges in the system, links will appear below the badge to switch to another. You can click in the Look Up A Badge link to return to the list of registrant badges.

The far left column is the Manage column. Clicking the Manage link for any registrant's badge will direct you to the manage screen for that registrant.



On the left of the screen is the registrant's personal information as provided at the time of registration. On the right side of the screen are options that allow you to manage the items in this registrant's badge.

If the registrant's personal information has changed, or a there is a misspelling you can correct it.



- Badge #: each badge purchased in the Event Manager is assigned a badge number. This is for reference, and can not be edited.
- User: this is the username of the registrant. Registrants can be associated with an existing WebGUI user account. Click the gray box with the ... in it to select a different username.
- Name: the name of the registrant as entered when the badge was selected for purchase.
- Organization: an organization that this registrant is associated with, such as a business name.
- Address: the street address for this registrant.
- City: the city in which the resident resides or works.
- State/Province: the state/province in which the registrant resides or works.
- Country: the country in which this registrant resides or works.
- Phone Number: a telephone number by which this registrant can be contacted.

- Email Address: an email address by which this registrant can be contacted.

- Notes: a field in which event staff can enter notes about this registrant.

Once any necessary changes have been made, click the Save button.

On the right hand side of the registrant's manage screen are additional management options.

- Checked In status: on the day of the event, registrants will check in. At this time tokens will be provided, and the registrant might receive a name tag. Before the registrant checks in, a green “Not Checked In” message will display. To check in a registrant, simply click on this green message. A new message will display in red indicating that the check in process is complete.

- Badge Name: below the checked in status is the name of the badge this registrant purchased.

Event staff have the following options listed below the badge name:

- Print: the print link directs the user to the template view used to print badges. This is often used for name tags at an event to indicate that a person is a paid attendee. The default template displays the badge name, registrant's name, registrant's city and state, and the badge number.

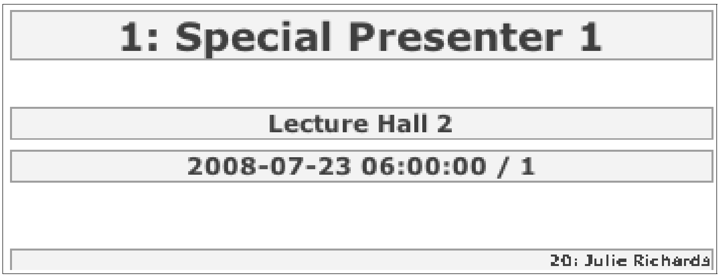
- Refund: clicking on Refund will refund the price of this badge to the registrant. In addition, it will replenish this Badge in the Event Manager, allowing someone else to purchase it. If this badge is part of a Badge Group with some tickets, refunding the badge will also refund and return all related tickets.

- Mark as Not Checked In: this reverses the Checked In status to not checked in.

- Add More Items: directs event staff back to the badge builder screen for this registrant, where additional Tickets, Ribbons, and Tokens can be added to the badge. The registrant will have to complete the check out process to pay for any additional items added.

Below the name of the badge this registrant purchased are listed all the additional items added to the badge during registration. These include Tickets, Ribbons and Tokens.

Each item has one or two links: Print and Refund. Tickets will have Print link which will direct the user to the template used to print that item. In the case of tickets, this would be used to provide the registrant a physical paper ticket to present at that ticket's event.



The screenshot above shows the print template for a ticket. At the top is the name of the ticket. Below it are listed the location of this ticket's event followed by the date and time this event occurs. In the lower right hand corner is this ticket's corresponding badge number and registrant's name.

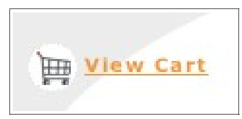
Every item in the badge has a Refund link. Clicking the refund link will remove that item from the badge, refund the registrant the cost of that item, and place the item back in the Event Manager so another person can purchase it.

Schedule

Upon entering the EMS, you will see a Schedule link located in the list of links at the top of the asset. In order to view a schedule, you must have tickets set up inside a badge. Clicking on the Schedule link will display a schedule of events.

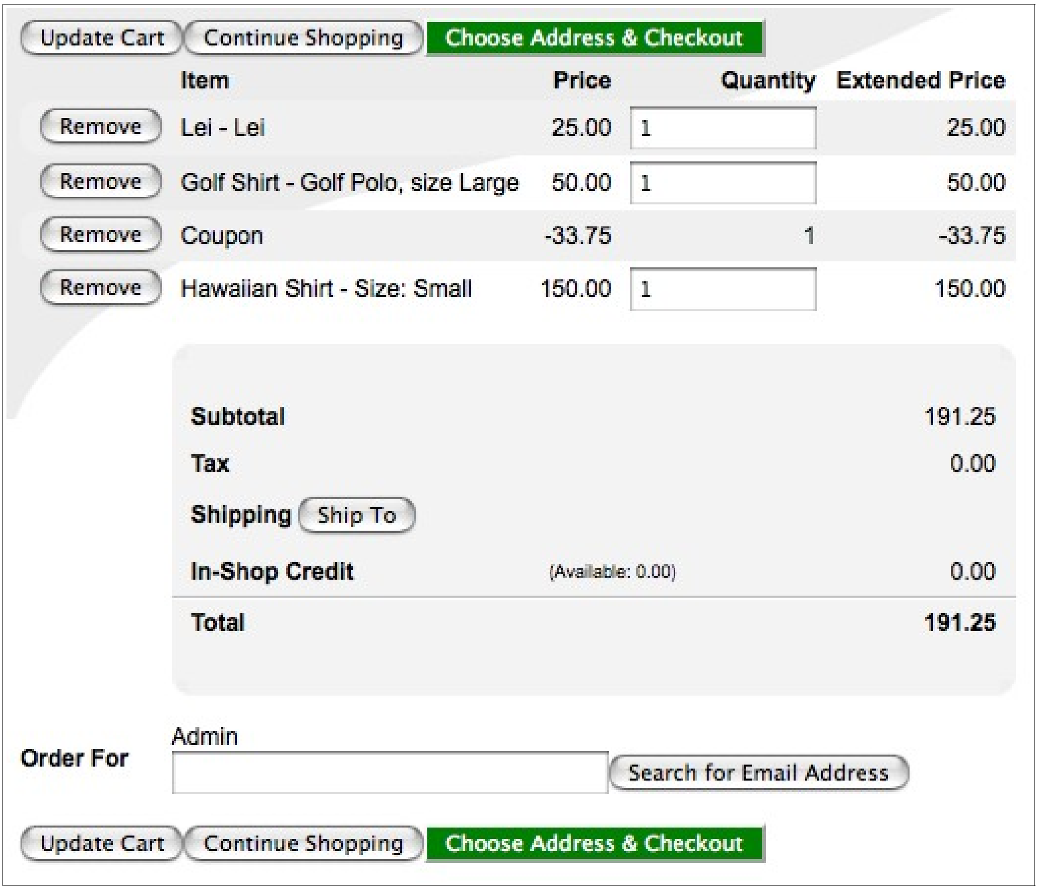
Shopping Cart

After selecting something to purchase- Product, Donation, Subscription or Event Manager related assets- the user will need to go to the shopping cart to complete the transaction. The shopping cart is accessed through the View Cart link. This link can be located in a number of different locations on the page, depending on how it is being displayed and on the templates being used to display it. Generally speaking, the user will see one of the following on the page:



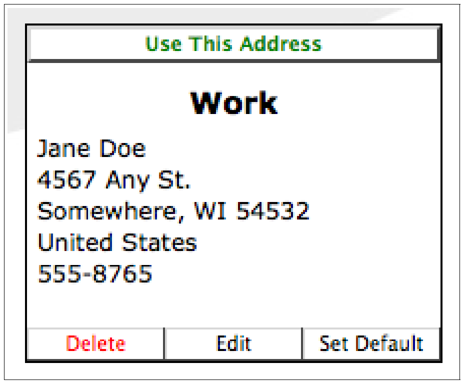
To access their shopping carts, users click on "View Cart." This opens the shopping cart, where users can remove items, change the number of a single item being purchased, set a shipping address and complete the payment process.

You can not purchase recurring items (such as a Subscription) and non-recurring items in the same Cart. Each type of transaction needs to be in its own Cart.



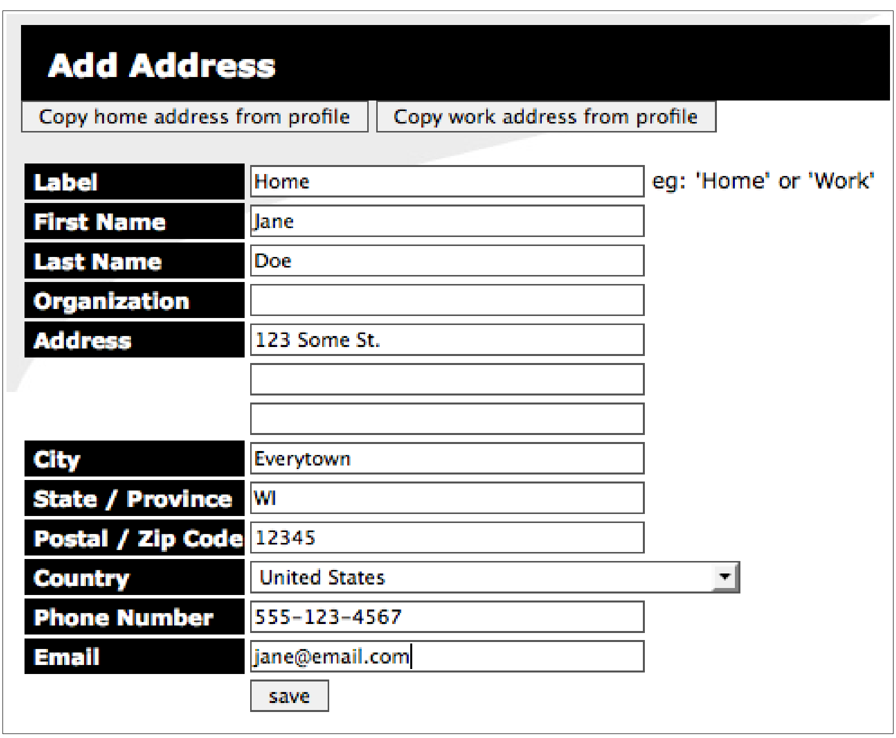
At the top and bottom of the shopping cart are three buttons: Update Cart, Continue Shopping, and Choose Shipping Address.

- Update Cart: if an item is removed or its quantity is changed, clicking Update Cart will update the data to reflect those changes.
- Continue Shopping: updates the cart and directs the user back to the shop to continue shopping.
- Choose Address & Checkout: clicking on this directs the user to the Address book screen, where shipping address can be created and stored. Once created, the user can select an existing address.



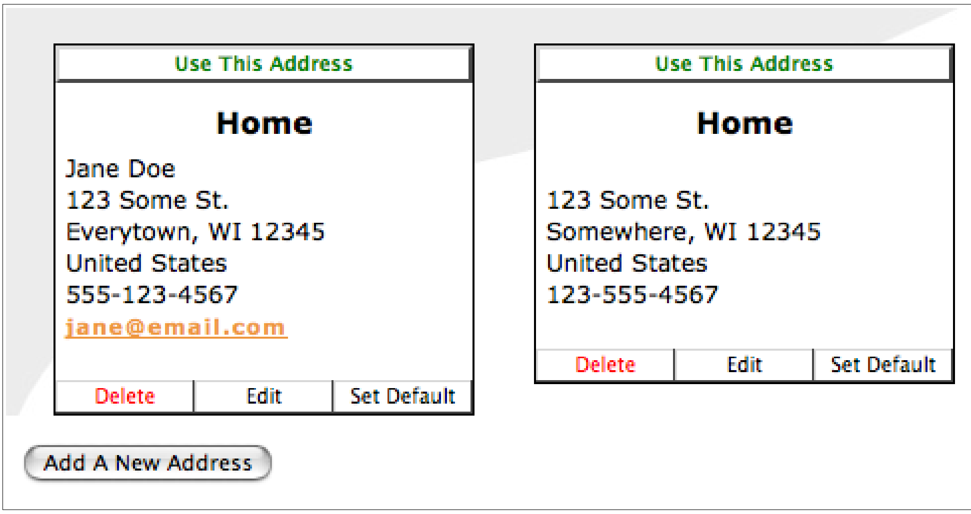
If a shipping address already exists for this user, that address will display, as well as a button by which to add a new shipping address.

Clicking the "Add a New Address" button opens the Add Address form. From here users can click on the "Copy home address from profile," or "Copy work address from profile" buttons to populate this form with the data already entered in the user's user profile. Or, a user can manually create an address by filling in the fields displayed on this screen.

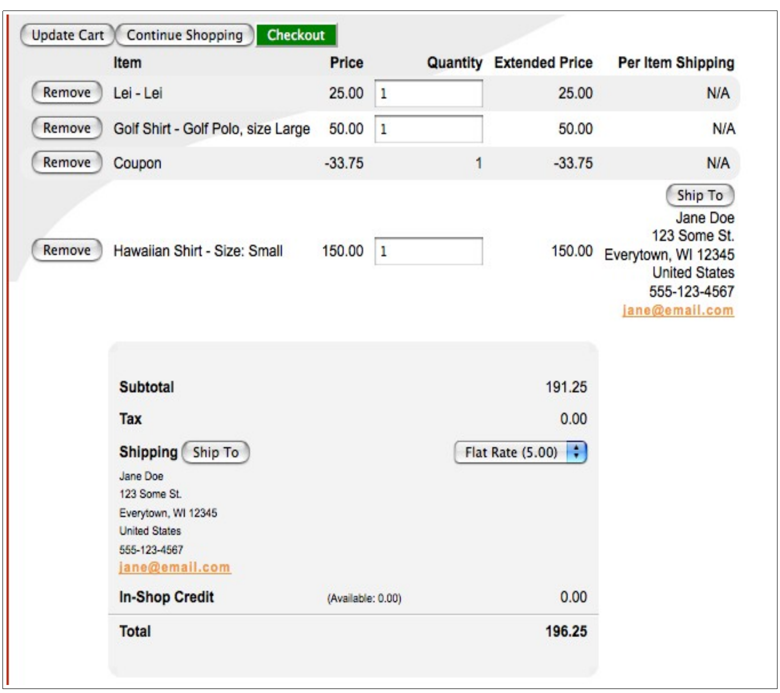


- In the “Label” field enter a label by which to identify this shipping address in the Shipping Address screen.
- In the “First Name” field enter the first name of person to whom this purchase is being shipped.
- In the “Last Name” field enter the last name of person to whom this purchase is being shipped.
- In the “Organization” field you can enter an organization, such as a business or non-profit, to associate with this shipping address. This could be something like the workplace of the recipient.
- In the “Address”field, enter the street address, including apartment or office number if applicable, for this shipping address.
- In the “City” field, enter the city where this shipping address is located.
- In the “State/Province” field, enter the state of province in which this shipping address is located.
- In the “Postal/Zip Code” field, enter the postal or zip code of this shipping address.
- In the “Country” field, select the name of the country where this shipping address is located.
- In the “Phone Number” field, enter a telephone number by which to contact someone at this shipping address.
- Enter the recipient's email address in the “Email” field.

When the Shipping Address form is complete, click on save to add it to the Shipping Address screen.



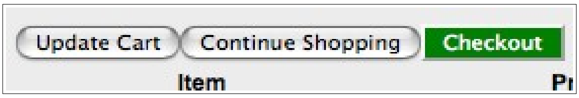
Users may click on the green “Use This Address” link above a shipping address to select it for use with this purchase. At the bottom of each address is a red delete link to delete an address, and and Edit link to edit an address. “Set Default” will make the selected address the default shipping address used every time this user enters the shopping cart screen.



When an address is chosen as the shipping address, it will appear in the main shopping cart with the label “Shipping” next to it. The “Ship To” button next to it will allow users to select a different shipping address.

In the gray area of the screen is a small dropdown menu that allows the user to select the shipping driver for this purchase (for example, free shipping).

Notice that once a shipping address is selected, the Choose Address & Checkout button changes to “Checkout.”

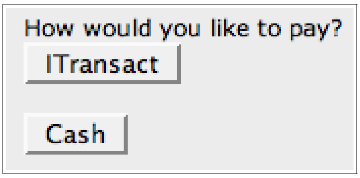


The cart may also contain a Per Item Shipping column. If an item is shippable (such as a Product asset), the shopper can choose to have each item shipped to a different destination. A small link will appear next to the item in the cart to allow a new shipping address to be specified.

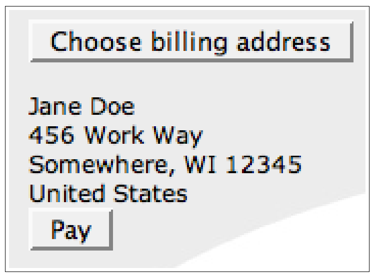
- Next to each item in the shopping cart is a “Remove” button. Clicking this button will remove this item from the shopping cart and make it available again to purchase. WebGUI Shop will know that this item has been removed from the cart; therefore, the Product's, or other asset's, stock amount will be adjusted accordingly. In the case of the Event Manager, removing a Badge or Ticket will replenish the seats at those events. If a user removes a Badge that is part of a Badge Group, all related tickets will be automatically removed as well.

- The “Quantity” field allows users to change the quantity of an item in the cart. Clicking the “Update Cart” button after changing the quantity of an item will update the data on the screen to reflect this change.

- The “Checkout” button directs the user to a screen prompting the user to select a payment method.



When a payment option is selected, the user is directed to a screen in which to enter billing information. If a user selected to pay by credit card, the user would first need to fill in credit card information before proceeding. By default, the address selected as the shipping address will appear as the billing address. If the user wants to use the shipping address as the billing address as well, the user can go ahead and click the Pay button.



Clicking on the Choose Billing address button will open Shop's address book, where the user can select a billing address, or create a new address. Users then click Pay to complete the process.

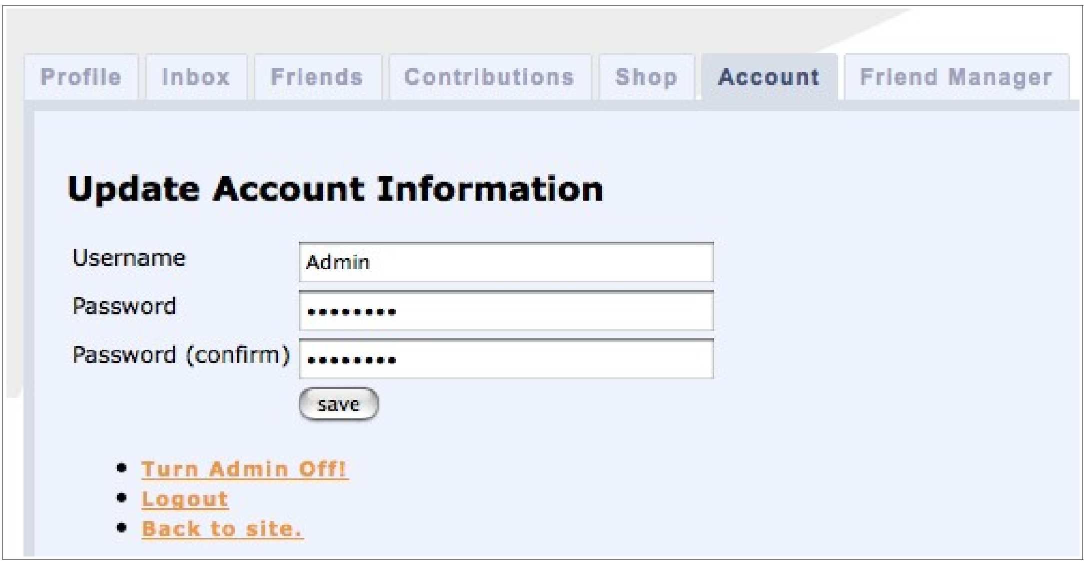
Other payment drivers may be set up on your site. You may need to fill out forms slightly different than what is shown here.

Shop administrators will also see an “Order For” field at the bottom of the shopping cart screen.

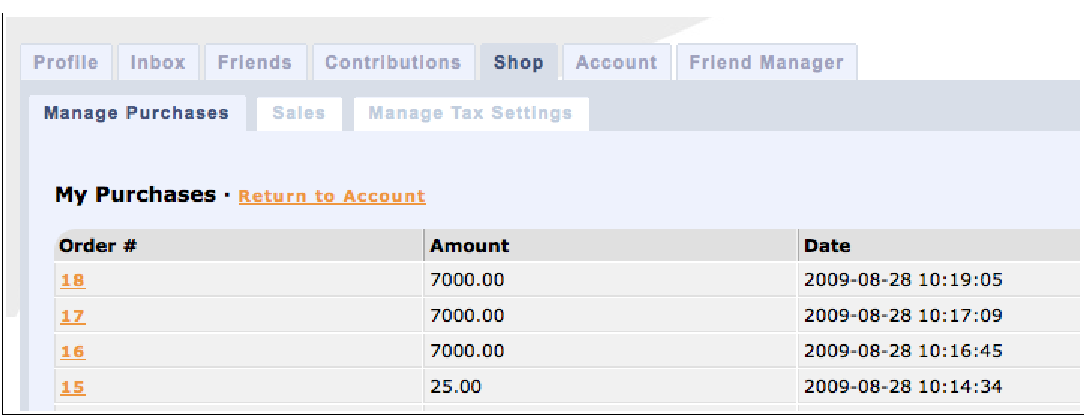
A shop administrator can search for users based on email address. Once the appropriate user is found, the shop administrator can complete a purchase on behalf of that user. Users must have an email address in their user profile for this to work.

My Purchases

WebGUI Shop allows users to view purchases made through the site. To see their purchases, users enter the Update Account Information screen. In the user account system there is a Shop tab.

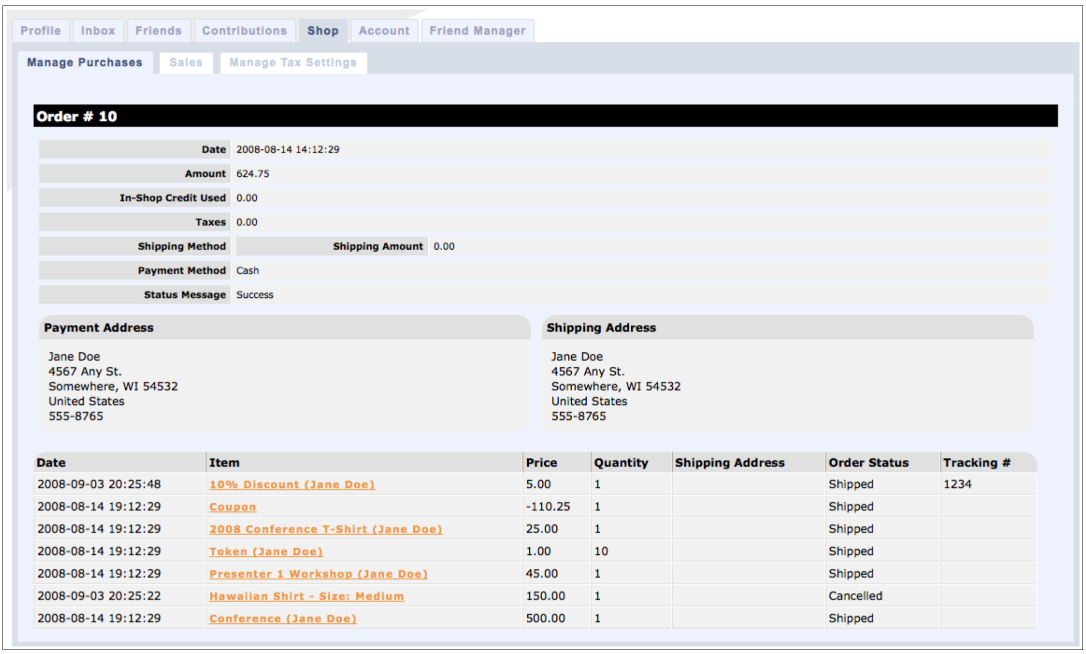


Below the username and password fields are a number of links. Clicking on the “My Purchases” screen directs the user to a screen displaying all the purchases that user has made on the site.



Down the left hand side of the screen, in the Order # column, are listed the order numbers for each purchase. The middle column, Amount, displays the total amount paid for each purchase. The column on the far right, Date, displays the date and time of purchase.

Clicking on an order number will direct the user to details about that purchase. This screen is similar to what Shop managers see when accessing an order through the Transactions screen.



The date, purchase amounts and other applicable charges are listed in the top left hand area of the screen. Near the middle is the payment address and shipping address this purchase was sent to, and at the very bottom of the screen is an itemized list of the item(s) purchased. Clicking on an item will direct the user to that item on the site. Users can also see the price of each item, the quantity of each item purchased, a separate shipping address for the item if one was used, the order status and its tracking number.

More information about the Sales and Manage Tax Settings tabs will be included in later editions of the Shop guide.

Writing SKUs

This chapter shows how to write SKUs, subclasses of WebGUI::Asset::Sku, which are assets that tie into the WebGUI Shop. Creating your own SKUs provides your customers a custom shopping experience.

Learn more about writing assets in the Assets chapter of the WebGUI Developers Guide.

WebGUI Shop comes with many SKUs that you can use as examples to guide you in writing your own SKUs. The Event Manager has four SKUs of its own: EMSTicket, EMSBadge, EMSRibbon, and EMSToken. In addition, there are the Product and Subscription SKU', which can be used to generically sell any kind of products or recurring memberships. There are also some specialty SKUs, specifically the Donation, Ad Sales, Thingy Record and Flat Discount Coupon. It's wise to examine these SKUs before developing your own to get a feel for the effort involved, and to give you ideas of how to build your own.

API Highlights

The rich SKU API provides the power and flexibility needed to develop your own subclasses.

addToCart()

The most important method in all SKUs is the addToCart() method, because this is where the transaction begins. In some cases you may not need to modify the addToCart() method for your SKU, but in most cases you will probably either modify it or, at the very least, write a wrapper around it that calls it.

The main instance where you would want to modify the addToCart() method is if you have some book keeping functions to perform. For example, if this is a product with an inventory, you'll likely want to subtract the item out of inventory so that it can't be oversold.

The base addToCart() method accepts a hash reference of configuration properties and then calls the methods necessary to put this product into the cart. The configuration properties are used if you need to keep track of how the product is configured. For example, if it's a donation, you could put the donation amount in the properties. If it's a configurable product, like a computer, then you could store all of those configuration options in there. Whenever the cart, or other parts of the shop, interact with this SKU, it will restore this SKU with these options using the applyOptions() and getOptions() methods.

The addToCart() method is generally never called by any WebGUI subsystems. It's there for you to write a wrapper around and to call. Generally, you might create a www_addToCart() method that would be called by a user clicking on a button. How that happens is up to you.

Event Handlers

The SKU provides a series of event handlers which help manage the purchase process. There are some that are specific to the type of SKU that you're building. Others are useful in all SKUs: onCompletePurchase(), onRefund(), and onRemoveFromCart().

- onCompletePurchase() is called when a transaction has finalized and payment has been made. Use onCompletePurchase() to handle book keeping tasks like giving privileges, accounting for inventory, etc.

- onRefund() is called when a shop manager clicks the Refund button in the transaction manager. Use onRefund() to take away privileges, restock inventory that was previously purchased, etc.

- onRemoveFromCart() is called when an item is removed from the shopping cart before purchase. Use onRemoveFromCart() to restock items that you took out of inventory with the addToCart() method.

The Numbers

In a sale it all comes down to the numbers, so there are several methods that help you manage the numbers for each SKU.

- The getMaxAllowedInCart() method should be overridden in the event you're selling a unique item (for instance a ticket that has been customized to a user).

- The getPrice() method should be overridden to return the price of the SKU, and should calculate the price in the case of a configurable product.

- The getQuantityAvailable() method should be overridden if there are a limited number of the SKU available for sale.

Display Helpers

There are a couple of methods that should be overridden to help visually integrate your SKU with the Shop.

- getConfiguredTitle() should return a descriptive title, such as “Red XL T-Shirt” rather than just “T-Shirt”.

- getThumbnailUrl() should return the URL to a thumbnail of a picture of the SKU.

SKUs as Products

The most common archetype of a SKU is a product: a physical good set up for sale. There is a generic Product SKU, that comes with WebGUI, that can handle the needs of most users. However, there are many scenarios in which using a SKU to do just the right job makes sense. That's where you come in to write a new product SKU.

Special Methods

There are a few special methods that you'll need to know when writing this type of SKU.

- The getWeight() method should be overridden to reflect the item's shipping weight.

- The isShippingRequired() method should be overridden to return 1 if the good is non-digital (a book rather than a PDF), or if the item will be picked up rather than shipped (like will-call tickets at a box office).

- The onAdjustQuantityInCart() method is an event handler, which is called when the user updates the quantity of an item in the cart. Use onAdjustQuantityInCart() to adjust inventory levels so that you don't over-sell an item.

Book Example

This shows how to write a SKU as a product archetype through the example of selling a book. Books have several properties that are unique to each book. A normal Product SKU could do the job, but not as well as a SKU written specifically for selling books. This example uses the special properties of ISBN, Author, and Edition.

The examples in this chapter only cover Shop specific information. Learn more in the Assets chapter of the WebGUI Developers Guide

As with any asset, start by creating the definition(). Add the author, ISBN, and edition properties, as well as the weight and price of the book.

```
sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		isbn => {
			tab		=> "properties",
			fieldType	=> "text",
			defaultValue	=> undef,
			label		=> 'ISBN',
			hoverHelp	=> 'The International Standard Book Number.',
			},
		author => {
			tab		=> "properties",
			fieldType	=> "text",
			defaultValue	=> undef,
			label		=> 'Author',
			hoverHelp	=> 'Author\'s name or pen name.',
			},
		edition => {
			tab		=> "properties",
			fieldType	=> "text",
			defaultValue	=> 'First Edition',
			label		=> 'Edition',
			hoverHelp	=> 'The printing number or name of the book.',
			},
		price => {
			tab		=> "shop",
			fieldType	=> "float",
			defaultValue	=> 0,
			label		=> 'Price',
			hoverHelp	=> 'The amount this book sells for.',
			},
```

```
            weight => {
                    tab                 => "shop",
                    fieldType           => "float",
                    defaultValue        => 0,
                    label               => 'Weight',
                    hoverHelp           => 'How much in lbs does this book weigh?',
                    },
        );
        push @{$definition}, {
            assetName               => 'Book',
            icon                    => 'assets.gif',
            autoGenerateForms       => 1,
            tableName               => 'Book',
            className               => 'WebGUI::Asset::Sku::Book',
            properties              => \%properties
        };
        return $class->SUPER::definition($session, $definition);
}
```

Because you're building a product archetype, fill out the getPrice() and getWeight() methods. Note that the price formatted is returned using sprintf. This saves you from having to do it everywhere that you want to display the price.

```
sub getPrice {
        my ( $self ) = @_;
        return sprintf "%.2f", $self->get('price');
}

sub getWeight {
        my ( $self ) = @_;
        return $self->get('weight');
}
```

Then, define your view() method. Normally, you would want to template this, but to keep it simple this example has left out that step. Instead, all the properties are displayed using a standard HTML Form.

```
sub view {
        my ( $self ) = @_;
        my $session = $self->session;
        my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
```

```
        $f->readOnly(
                label  => 'Title',
                value  => $self->get('title'),
                );
        $f->readOnly(
                label  => 'Edition',
                value  => $self->get('edition'),
                );
        $f->readOnly(
                label  => 'Description',
                value  => $self->get('description'),
                );
        $f->readOnly(
                label  => 'Author',
                value  => $self->get('author'),
                );
        $f->readOnly(
                label  => 'ISBN',
                value  => $self->get('isbn'),
                );
        $f->readOnly(
                label  => 'Price',
                value  => $self->getPrice,
                );
        $f->hidden(
                name => "func",
                value  => "buy",
                );
        $f->submit( value => 'Add To Cart');
        return $f->print;
}
```

In view() you defined a reference to a www_buy() method, so you must also create that. This is the wrapper that will call addToCart() for you.

```
sub www_buy {
        my ( $self ) = @_;
        my $session = $self->session;
        return $session->privilege->noAccess() unless ($self->canView);
        $self->addToCart;
        return $self->getParent->www_view;
}
```

Note that you want www_buy() to follow the same view privileges as all other assets. If you can't view it, you shouldn't be able to buy it. In this way you can make some books available to only certain groups of purchasers.

That gives you a final class that looks similar to the one below. This is also uploaded to the add-ons area of webgui.org, so you can work directly from this code if you plan on making something similar.

```
package WebGUI::Asset::Sku::Book;
use strict;
use Tie::IxHash;
use base 'WebGUI::Asset::Sku';
use WebGUI::Utility;
use WebGUI::HTMLForm;

=head1 NAME
Package WebGUI::Asset::Sku::Book
=head1 DESCRIPTION
This sku allows you to sell books on your site with complete information.
=head1 SYNOPSIS
use WebGUI::Asset::Sku::Book;

=head1 METHODS
These methods are available from this class:
=cut

#-------------------------------------------------------------------
=head2 definition ( session, definition )
Adds isbn, author, and edition fields.

=head3 session

=head3 definition

A hash reference passed in from a subclass definition.
=cut

sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		isbn => {
			tab			=> "properties",
			fieldType		=> "text",
```

```
                defaultValue        => undef,
                label               => 'ISBN',
                hoverHelp           => 'The International Standard Book Number.',
                },
        author => {
                tab                 => "properties",
                fieldType           => "text",
                defaultValue        => undef,
                label               => 'Author',
                hoverHelp           => 'Author\'s name or pen name.',
                },
        edition => {
                tab                 => "properties",
                fieldType           => "text",
                defaultValue        => 'First Edition',
                label               => 'Edition',
                hoverHelp           => 'The printing number or name of the book.',
                },
        price => {
                tab                 => "shop",
                fieldType           => "float",
                defaultValue        => 0,
                label               => 'Price',
                hoverHelp           => 'The amount this book sells for.',
                },
        weight => {
                tab                 => "shop",
                fieldType           => "float",
                defaultValue        => 0,
                label               => 'Weight',
                hoverHelp           => 'How much in lbs does this book weigh?',
                },
    );
    push @{$definition}, {
        assetName               => 'Book',
        icon                    => 'assets.gif',
        autoGenerateForms       => 1,
        tableName               => 'Book',
        className               => 'WebGUI::Asset::Sku::Book',
        properties              => \%properties
    };
    return $class->SUPER::definition($session, $definition);
}
```

```
#-------------------------------------------------------------------

=head2 getPrice ()

Returns the value of the price field formatted as currency.

=cut

sub getPrice {
	my ( $self ) = @_;
	return sprintf "%.2f", $self->get('price');
}

#-------------------------------------------------------------------

=head2 getWeight ()

Returns the value of the price field formatted as currency.

=cut

sub getWeight {
	my ( $self ) = @_;
	return $self->get('weight');
}

#-------------------------------------------------------------------

=head2 indexContent ( )

Adding search keywords. See WebGUI::Asset::indexContent() for additonal details.

=cut

sub indexContent {
	my ( $self ) = @_;
	my $indexer = $self->SUPER::indexContent;
	$indexer->addKeywords($self->get('author'), $self->get('isbn'), $self->get('edition'));
}

#-------------------------------------------------------------------
=head2 view ( )
```

```
method called by the container www_view method.

=cut

sub view {
	my ( $self ) = @_;
	my $session = $self->session;
	my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
	$f->readOnly(
		label  => 'Title',
		value => $self->get('title'),
		);
	$f->readOnly(
		label  => 'Edition',
		value => $self->get('edition'),
		);
	$f->readOnly(
		label  => 'Description',
		value => $self->get('description'),
		);
	$f->readOnly(
		label  => 'Author',
		value => $self->get('author'),
		);
	$f->readOnly(
		label  => 'ISBN',
		value => $self->get('isbn'),
		);
	$f->readOnly(
		label  => 'Price',
		value => $self->getPrice,
		);
	$f->hidden(
		name => "func",
		value => "buy",
		);
	$f->submit( value => 'Add To Cart');
	return $f->print;
}

#-------------------------------------------------------------------
```

```
=head2 www_buy ()

Adds the book to the cart.

=cut

sub www_buy {
	my ( $self ) = @_;
	my $session = $self->session;
	return $session->privilege->noAccess() unless ($self->canView);
	$self->addToCart;
	return $self->getParent->www_view;
}

1;
```

Recurring SKU's

Recurring SKU's are often thought of as subscriptions or memberships. You pay every so often to see a support board, to read some articles about your favorite sports team, or to pay your dues to a club. In many of these cases the Subscription SKU that comes with WebGUI will serve the purpose perfectly. However, you may want to collect some data along with the subscription, so you'd create your own recurring SKU.

Special Methods

There are a few special methods you need to be aware of when writing a recurring SKU archetype.

- The getRecurInterval() method should be overridden so that the Shop knows how often to process the recurring transaction.

- The isRecurring() method should be overridden to return 1 so that the Shop knows this is a recurring SKU.

- The onCancelRecurring() method is an event handler, which is called when a user or administrator cancels a recurring transaction. Use this to make privilege adjustments, send out emails, etc. However, be careful not to take away a user's privileges too early. This cancels the next transaction, but if you're working with a subscription, and that subscription has not already expired, then the user needs to maintain its current privileges, and lose them when the next transaction fires.

Association Dues Example

In clubs and associations, it's often necessary to collect dues and members' information. You may want to collect these simultaneously the first time, and then allow your members to update their data periodically thereafter. A great way to achieve this is to display a form of the data you want to capture with the add to cart button. For this example, data is collected and placed in the user's profile. That way the user can use the profile system to update the data going forward.

Start with the definition() method, which defines the memberGroupId (the group that holds your members) and the price fields.

```
sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		memberGroupId => {
			tab		=> "properties",
			fieldType	=> "group",
			defaultValue	=> '3',
			label		=> 'Member Group',
			hoverHelp	=> 'The group holding your membership.',
			},
		price => {
			tab		=> "shop",
			fieldType	=> "float",
			defaultValue	=> 0,
			label		=> 'Price',
			hoverHelp	=> 'The annual fee for this membership.',
			},
	);
	push @{$definition}, {
		assetName		=> 'Member Dues',
		icon			=> 'assets.gif',
		autoGenerateForms	=> 1,
		tableName		=> 'MemberDues',
		className		=> 'WebGUI::Asset::Sku::MemberDues',
		properties		=> \%properties
	};
	return $class->SUPER::definition($session, $definition);
}
```

Then, you have to publish your price.

```
sub getPrice {
	my ( $self ) = @_;
	return sprintf "%.2f", $self->get('price');
}
```

Because this is a recurring SKU, you have to override the appropriate method to let the Shop know that. By defining isRecurring to return 1, getMaxAllowedInCart() will automatically also return 1. In this case, you're building a SKU that recurs annually, so you can hard code getRecurInterval(). In some cases you may want to make this a select box property so that the user can select the recur interval.

```
sub getRecurInterval {
	return 'Yearly';
}

sub isRecurring {
	return 1;
}
```

You need to set up the privileges on each recurrence, but you only want to update the membership data on the first recurrence. Do that with the onCompletePurchase() method.

```
sub onCompletePurchase {
	my ($self, $item) = @_;
	my $session = $self->session;
	my $transaction = $item->transaction;
	my $group = WebGUI::Group->new($session, $self->get('memberGroupId'));
	$group->addUsers([$transaction->get('userId')],60*60*24*365);
	if ($transaction->isFirst) { # this is the first instance of this transaction
		my $user = WebGUI::User->new($session, $transaction->get('userId'));
		my $options = $self->getOptions;
		$user->update( {
			email		=> $options->{email},
			firstName	=> $options->{firstName},
			lastName	=> $options->{lastName},
			workPhone	=> $options->{workPhone},
		);
	}
```

```
}
```

Then, you can build out the view() method with your registration form.

```
sub view {
	my ( $self ) = @_;
	my $session = $self->session;
	my ( $user ) = $session->quick(qw{ user });
	my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
	$f->readOnly(
		label  => 'Price',
		value  => $self->getPrice,
		);
	$f->hidden(
		name => "func",
		value => "buy",
		);
	$f->text(
		name		=> 'firstName',
		defaultValue	=> $user->profileField('firstName'),
		label		=> 'First Name',
		);
	$f->text(
		name		=> 'lastName',
		defaultValue	=> $user->profileField('lastName'),
		label		=> 'Last Name',
		);
	$f->email(
		name		=> 'email',
		defaultValue	=> $user->profileField('email'),
		label		=> 'Email Address',
		);
	$f->phone(
		name		=> 'workPhone',
		defaultValue	=> $user->profileField('workPhone'),
		label		=> 'Telephone Number',
		);
	$f->submit( value => 'Add To Cart');
	my $output = sprintf '<h3>%s</h3><p>%s</p><p>%s</p>',
		$self->getTitle, $self->get('description'), $f->print;
	return $output;
}
```

Just like in the Book product, you have to create the www_buy() method, but this time you have some configuration options to store with the item. Note the use of that in the addToCart() method.

```
sub www_buy {
	my ( $self ) = @_;
	my $session = $self->session;
	my ( $form ) = $session->quick(qw{ form });
	return $session->privilege->noAccess() unless ($self->canView);
	$self->addToCart({
		workPhone  => $form->get('workPhone','phone'),
		email      => $form->get('email','email'),
		firstName  => $form->get('firstName','text'),
		lastName   => $form->get('lastName','text'),
		});
	return $self->getParent->www_view;
}
```

All of the methods together give you an asset that looks like this:

```
package WebGUI::Asset::Sku::MemberDues;

use strict;
use Tie::IxHash;
use base 'WebGUI::Asset::Sku';
use WebGUI::Utility;
use WebGUI::HTMLForm;
use WebGUI::Group;


=head1 NAME

Package WebGUI::Asset::Sku::MemberDues

=head1 DESCRIPTION

This sku allows you to register new members to your organization.

=head1 SYNOPSIS

use WebGUI::Asset::Sku::MemberDues;
```

```
=head1 METHODS

These methods are available from this class:

=cut

#-------------------------------------------------------------------

=head2 definition ( session, definition )

Adds  memberGroupId and price fields.

=head3 session

=head3 definition

A hash reference passed in from a subclass definition.

=cut

sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		memberGroupId => {
			tab		=> "properties",
			fieldType	=> "group",
			defaultValue	=> '3',
			label		=> 'Member Group',
			hoverHelp	=> 'The group holding your membership.',
			},
		price => {
			tab		=> "shop",
			fieldType	=> "float",
			defaultValue	=> 0,
			label		=> 'Price',
			hoverHelp	=> 'The annual fee for this membership.',
			},
	);
	push @{$definition}, {
		assetName		=> 'Member Dues',
		icon			=> 'assets.gif',
		autoGenerateForms	=> 1,
```

```
        tableName               => 'MemberDues',
        className               => 'WebGUI::Asset::Sku::MemberDues',
        properties              => \%properties
    };
    return $class->SUPER::definition($session, $definition);
}

#-------------------------------------------------------------------

=head2 getPrice ()

Returns the value of the price field formatted as currency.

=cut

sub getPrice {
    my ( $self ) = @_;
    return sprintf "%.2f", $self->get('price');
}

#-------------------------------------------------------------------

=head2 getRecurInterval ( )

Returns 'Yearly'.

=cut

sub getRecurInterval {
    return 'Yearly';
}

#-------------------------------------------------------------------

=head2 isRecurring ( )

Returns 1.

=cut

sub isRecurring {
    return 1;
}
```

```
sub onCompletePurchase {
	my ($self, $item) = @_;
	my $session = $self->session;
	my $transaction = $item->transaction;
	my $group = WebGUI::Group->new($session, $self->get('memberGroupId'));
	$group->addUsers([$transaction->get('userId')],60*60*24*365);
	if ($transaction->isFirst) { # this is the first instance of this transaction
		my $user = WebGUI::User->new($session, $transaction->get('userId'));
		my $options = $self->getOptions;
		$user->update( {
			email		=> $options->{email},
			firstName	=> $options->{firstName},
			lastName	=> $options->{lastName},
			workPhone	=> $options->{workPhone},
		);
	}
}

#-------------------------------------------------------------------

sub view {
	my ( $self ) = @_;
	my $session = $self->session;
	my ( $user ) = $session->quick(qw{ user });
	my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
	$f->readOnly(
		label	=> 'Price',
		value	=> $self->getPrice,
		);
	$f->hidden(
		name => "func",
		value => "buy",
		);
	$f->text(
		name		=> 'firstName',
		defaultValue	=> $user->profileField('firstName'),
		label		=> 'First Name',
		);
	$f->text(
		name		=> 'lastName',
		defaultValue	=> $user->profileField('lastName'),
		label		=> 'Last Name',
```

```
        );
    $f->email(
        name        => 'email',
        defaultValue=> $user->profileField('email'),
        label       => 'Email Address',
        );
    $f->phone(
        name        => 'workPhone',
        defaultValue=> $user->profileField('workPhone'),
        label       => 'Telephone Number',
        );
    $f->submit( value => 'Add To Cart');
    my $output = sprintf '<h3>%s</h3><p>%s</p><p>%s</p>',
        $self->getTitle, $self->get('description'), $f->print;
    return $output;
}

#-------------------------------------------------------------------

=head2 www_buy ()

Adds the book to the cart.

=cut

sub www_buy {
    my ( $self ) = @_;
    my $session = $self->session;
    my ( $form ) = $session->quick(qw{ form });
    return $session->privilege->noAccess() unless ($self->canView);
    $self->addToCart({
        workPhone   => $form->get('workPhone','phone'),
        email       => $form->get('email','email'),
        firstName   => $form->get('firstName','text'),
        lastName    => $form->get('lastName','text'),
        });
    return $self->getParent->www_view;
}

1;
```

SKUs as Coupons

Another powerful aspect of SKUs is that they know what else is in the cart and can adjust the price accordingly. This allows SKUs to be used as coupons. For example, you might detect that the user has placed an order for 10 packs of tube socks in the cart. You just happen to be running a special on tube socks, which gives the user a 20% discount if s/he buys 5 or more packages. Your coupon can look at the cart, determine if the user has met the criteria, and then return a negative price, thus creating a discount.

Special Methods

There are a couple of methods you should be aware of when creating a SKU as coupon archetype.

- The getCart() method gives you a reference to the shopping cart (WebGUI::Shop::Cart) so that you can see what else is in the cart, to make your judgments about discounts to give.

- You also need to override isCoupon() to return a 1, because some coupons may want to know if the user has other coupons in the cart, and only allow one coupon per cart.

Member Discount Example

Let's say that you have an online shop, which sells training materials for realtors, and that you also run a realtor association from your web site. You may want to give a discount to all the realtors that also belong to your association. So, create a coupon that your members can add to their carts that gives them their association discount.

Similar to the Member Dues example, start out your definition() method with a memberGroupId property. This time, instead of price, you have a discount field.

```
sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		memberGroupId => {
			tab		=> "properties",
			fieldType	=> "group",
			defaultValue	=> '3',
			label		=> 'Member Group',
			hoverHelp	=> 'The group holding your membership.',
			},
```

```
            discount => {
                tab             => "shop",
                fieldType       => "integer",
                defaultValue    => 10,
                label           => 'Percentage Discount',
                hoverHelp       => 'The amount of discount in percent.',
                },
    );
    push @{$definition}, {
            assetName           => 'Member Discount',
            icon                => 'assets.gif',
            autoGenerateForms   => 1,
            tableName           => 'MemberDiscount',
            className           => 'WebGUI::Asset::Sku::MemberDiscount',
            properties          => \%properties
    };
    return $class->SUPER::definition($session, $definition);
}
```

With the coupon archetype most of the work happens in the getPrice() method. Only give the discount if the user is a member of your group. Also note that when checking the items in the cart you skip yourself. If you didn't do this you would create an infinite loop.

```
sub getPrice {
    my ( $self ) = @_;
    my $session = $self->session;
    my ( $user ) = $session->quick(qw{ user });
    my $discount = 0;
    if ($user->isInGroup($self->get('memberGroupId'))) {
        foreach my $item (@{$self->getCart->getItems}) {
            next if ($item->get('assetId') eq $self->getId); # prevent infinite loop
            $discount += $item->sku->getPrice * $item->get('quantity') * $self-
>get('discount') * -1 / 100;
        }
    }
    return sprintf "%.2f", $discount;
}
```

Because this is a coupon you have to tell the Shop that.

```
sub isCoupon {
```

```
        return 1;
}
```

And your view() method looks very simple this time.

```
sub view {
        my ( $self ) = @_;
        my $session = $self->session;
        my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
        $f->hidden(
                name => "func",
                value  => "addToCart",
                );
        $f->submit( value => 'Add To Cart');
        my $output = sprintf '<h3>%s</h3><p>%s</p><p>%s</p>',
                $self->getTitle, $self->get('description'), $f->print;
        return $output;
}
```

But the www_addToCart() method looks pretty similar to the www_buy() method in the Book example.

```
sub www_addToCart {
        my ( $self ) = @_;
        return $self->session->privilege->noAccess() unless ($self->canView);
        $self->addToCart;
        return $self->getParent->www_view;
}
```

If you put all that together, you get an asset that looks similar to this:

```
package WebGUI::Asset::Sku::MemberDiscount;

use strict;
use Tie::IxHash;
use base 'WebGUI::Asset::Sku';
use WebGUI::Utility;
use WebGUI::HTMLForm;



=head1 NAME
```

```
Package WebGUI::Asset::Sku::MemberDiscount

=head1 DESCRIPTION

This sku gives members of a specific group a discount.

=head1 SYNOPSIS

use WebGUI::Asset::Sku::MemberDiscount;


=head1 METHODS

These methods are available from this class:

=cut

#-------------------------------------------------------------------

=head2 definition ( session, definition )

Adds memberGroupId and discount fields.

=head3 session

=head3 definition

A hash reference passed in from a subclass definition.

=cut

sub definition {
	my ( $class, $session, $definition ) = @_;
	tie my %properties, 'Tie::IxHash', (
		memberGroupId => {
			tab		=> "properties",
			fieldType	=> "group",
			defaultValue	=> '3',
			label		=> 'Member Group',
			hoverHelp	=> 'The group holding your membership.',
			},
		discount => {
```

```
                        tab                     => "shop",
                        fieldType               => "integer",
                        defaultValue            => 10,
                        label                   => 'Percentage Discount',
                        hoverHelp               => 'The amount of discount in percent.',
                        },
        );
        push @{$definition}, {
                assetName               => 'Member Discount',
                icon                    => 'assets.gif',
                autoGenerateForms       => 1,
                tableName               => 'MemberDiscount',
                className               => 'WebGUI::Asset::Sku::MemberDiscount',
                properties              => \%properties
        };
        return $class->SUPER::definition($session, $definition);
}

#-------------------------------------------------------------------

=head2 getPrice ()

Returns the discount formatted as currency.

=cut

sub getPrice {
        my ( $self ) = @_;
        my $session = $self->session;
        my ( $user ) = $session->quick(qw{ user });
        my $discount = 0;
        if ($user->isInGroup($self->get('memberGroupId'))) {
                foreach my $item (@{$self->getCart->getItems}) {
                        next if ($item->get('assetId') eq $self->getId); # prevent infinite loop
                        $discount += $item->sku->getPrice * $item->get('quantity') * $self-
>get('discount') * -1 / 100;
                }
        }
        return sprintf "%.2f", $discount;
}

#-------------------------------------------------------------------
```

```
=head2 isCoupon ()

Returns 1.

=cut

sub isCoupon {
	return 1;
}

#-------------------------------------------------------------------

=head2 view ( )

method called by the container www_view method.

=cut

sub view {
	my ( $self ) = @_;
	my $session = $self->session;
	my $f = WebGUI::HTMLForm->new($session, action=>$self->getUrl);
	$f->hidden(
		name => "func",
		value  => "addToCart",
		);
	$f->submit( value => 'Add To Cart');
	my $output = sprintf '<h3>%s</h3><p>%s</p><p>%s</p>',
		$self->getTitle, $self->get('description'), $f->print;
	return $output;
}

#-------------------------------------------------------------------

=head2 www_addToCart ()

Adds the coupon to the cart.

=cut

sub www_addToCart {
	my ( $self ) = @_;
	return $self->session->privilege->noAccess() unless ($self->canView);
```

```
        $self->addToCart;
        return $self->getParent->www_view;
}

1;
```

Writing Payment Drivers

Payment drivers are plugins to WebGUI Shop that accept payment for an order. They may tie in to a traditional credit card payment gateway, or an e-check service, or to a payment service like Pay Pal© or Google Checkout©, or even to a custom billing system. No matter what they connect to, the idea is the same: verify and accept payment for an order purchased through WebGUI Shop.

API Highlights

WebGUI comes with many payment drivers. Cash is good for point of sale type orders. iTransact is a traditional credit card payment gateway service. PayPal is the ubiquitous web payment gateway. Ogone is a popular European payment gateway. All are great reference material that you should have a look at and take the time to understand before you start writing your own payment driver. When you are ready to write your own payment driver, there are several parts of the API of which you should be aware.

The Master Classes

Before you start writing your own payment driver subclass, check out WebGUI::Shop::Pay and WebGUI::Shop::PayDriver. Pay is the management interface for all PayDrivers. PayDriver is the master class that you'll be subclassing to create your own.

Pay hands off web requests to the PayDrivers. It does this through its www_do() method. Along the URL it looks like:

/home?shop=pay;method=do;driverId=XXXXX;do=**someMethod**

Note the bolded "someMethod" part of the URL. That refers to a www_**someMethod**() method in your PayDriver subclass. This allows you to build web accessible content through your payment methods, and use that to integrate your payment module with any payment gateway you want to over HTTP.

PayDriver Basics

There are a few methods you're going to need to know in order to write your own payment driver.

The first is the definition() method. If you've written an asset or workflow activity then you're familiar with how this works. In definition() you specify the properties of the driver and its human readable name.

You'll need to override the getButton() method and return an HTML button that the user will click on to select this payment method. This is generally the entrance point into a series of www_ methods that you'll create. However, for external payment systems like PayPal, this may redirect the user to the external system.

You'll then need to override the processPayment() method, which will be called by the processTransaction() method. The processPayment() method should actually perform the payment request to the payment gateway. This needs to be an atomic (pass/fail) type of transaction; either the whole thing succeeds or the whole thing fails.

Recurring Methods

If your payment gateway can be used to handle recurring payments (most can't), then there are a few additional methods of which you'll need to be aware.

You'll need to override the cancelRecurringPayment() method, which should make a call to the payment gateway requesting that the recurring transaction be terminated. Like processPayment() this request must be atomic.

You'll also need to override the handlesRecurring() method and make it return 1. This way WebGUI Shop knows this driver is capable of handling recurring payments.

Karma Payment Example

This example creates a payment gateway that uses Karma, instead of money, to pay for goods. This is not only an interesting concept, but it requires nothing other than WebGUI to use it. This means you'll be able to test it without getting involved with some external payment gateway service, which can be a daunting task if you haven't done it before.

First, create your definition() method, where you'll define a special property used as a currency conversion ratio. In other words, how many points of karma it takes to purchase an item worth $1.

```
sub definition {
    my ( $class, $session, $definition ) = shift;

    tie my %fields, 'Tie::IxHash', (
        conversionRatio => {
            fieldType      => 'integer',
            label          => 'Conversion Ratio',
            hoverHelp      => 'The amount of karma it takes to equal 1 of the shop\'s currency
(dollars, yen, euros, etc)',
            defaultValue    => 500,
            subtext        => 'karma : 1 currency'
        },
        );

    push @{ $definition }, {
        name        => 'Karma',
        properties  => \%fields,
    };

    return $class->SUPER::definition($session, $definition);
}
```

Now, define the getButton() method. Note the use of getDoFormTags(), which is just a helper that adds the shop, method, do, and payId variables to the form as hidden input tags. Redirect to the www_pay() method here.

```
sub getButton {
    my ( $self ) = @_;
    my $session = $self->session;
    my $payForm = WebGUI::Form::formHeader($session)
        . $self->getDoFormTags('pay')
        . WebGUI::Form::submit($session, {value => 'Karma' })
        . WebGUI::Form::formFooter($session);

    return $payForm;
}
```

Now you can write the payment processing method, which checks to see that the user has enough karma, then subtracts it for the payment.

```
sub processPayment {
   my ($self, $transaction) = @_;
   my $session = $self->session;
   my $user = WebGUI::User->new($session, $transaction->get('userId'));
   if ($user->karma / $self->get('conversionRatio') > $transaction->get('amount')) {
      $user->karma(
        $transaction->get('amount') * -1,
        'Shop',
        'Purchase of Order #'.$transaction->get('orderNumber')
        );
      return (1,          # success
            undef,      # transaction code
            1,          # status code
            'Success'); # status message
   }
   return (0,                    # success
         undef,                # transaction code
         0,                    # status code
         'Insufficient Karma');  # status message
}
```

Finish up the driver with the method that processes the payment and displays the “thank you” page.

```
sub www_pay {
   my ( $self ) = @_;
   my $cart    = $self->getCart;

   # Make sure you can checkout the cart
   return "" unless $cart->readyForCheckout;

   # Complete the transaction
   my $transaction = $self->processTransaction( );
   return $transaction->thankYou();
}
```

The finished class will look something like the following:

```
package WebGUI::Shop::PayDriver::Karma;

use strict;

use WebGUI::Shop::PayDriver;
use WebGUI::Exception;

use base qw/WebGUI::Shop::PayDriver/;

#-------------------------------------------------------------------

sub definition {
    my ( $class, $session, $definition ) = shift;

    tie my %fields, 'Tie::IxHash', (
        conversionRatio => {
            fieldType       => 'integer',
            label           => 'Conversion Ratio',
            hoverHelp       => 'The amount of karma it takes to equal 1 of the shop\'s currency
(dollars, yen, euros, etc)',
            defaultValue    => 500,
            subtext         => 'karma : 1 currency'
        },
        );

    push @{ $definition }, {
        name        => 'Karma',
        properties  => \%fields,
    };

    return $class->SUPER::definition($session, $definition);
}

#-------------------------------------------------------------------

sub getButton {
    my ( $self ) = @_;
    my $session = $self->session;
    my $payForm = WebGUI::Form::formHeader($session)
        . $self->getDoFormTags('pay')
        . WebGUI::Form::submit($session, {value => 'Karma' })
        . WebGUI::Form::formFooter($session);
```

```
    return $payForm;
}

#-------------------------------------------------------------------

sub processPayment {
    my ($self, $transaction) = @_;
    my $session = $self->session;
    my $user = WebGUI::User->new($session, $transaction->get('userId'));
    if ($user->karma / $self->get('conversionRatio') > $transaction->get('amount')) {
        $user->karma(
          $transaction->get('amount') * -1,
          'Shop',
          'Purchase of Order #'.$transaction->get('orderNumber')
          );
        return (1,          # success
                undef,      # transaction code
                1,          # status code
                'Success'); # status message
    }
    return (0,                    # success
            undef,                # transaction code
            0,                    # status code
            'Insufficient Karma');  # status message
}

#-------------------------------------------------------------------

sub www_pay {
    my ( $self ) = @_;
    my $cart    = $self->getCart;

    # Make sure you can checkout the cart
    return "" unless $cart->readyForCheckout;

    # Complete the transaction
    my $transaction = $self->processTransaction( );
    return $transaction->thankYou();
}


1;
```

Writing Shipping Drivers

Shipping drivers allow you to calculate and charge for shipping costs, and in some cases tie into third-party shipping providers such as UPS or FedEx.

API Highlights

WebGUI comes with two shipping drivers. FlatRate can calculate most basic shipping costs, but does not tie into any specific shipping provider. USPS interacts with the United States Postal Service APIs to calculate shipping costs based on weight and shipping address. They are both great reference material that you should have a look at and take the time to understand before you start writing your own shipping driver. When you are ready to write your own shipping driver, there are several parts of the API of which you should be aware.

The Master Classes

Just like with writing payment drivers, before you start writing your own shipping driver subclass, check out WebGUI::Shop::Ship and WebGUI::Shop::ShipDriver. Ship is the management interface for all ShipDrivers. ShipDriver is the master class that you'll be subclassing to create your own.

Ship hands off web requests to the ShipDrivers. It does this through its www_do() method. Along the URL it looks like:

/home?shop=ship;method=do;driverId=XXXXX;do=**someMethod**

Note the bolded “someMethod” part of the URL. That refers to a www_**someMethod**() method in your ShipDriver subclass. Generally speaking, you won't need this ability, but it's there if you do.

The Basics

There are really only two methods you should need to fill out in your shipping drivers, unless you do shipment tracking integration with a third party service. They are calculate() and definition().

- The calculate method is the true heart of a ShipDriver. This method gets a reference to the cart, and from that calculates the shipping cost on the products in the cart.

- The definition() method works just like the definition method in PayDrivers and is very similar to the definition() methods in Assets and Workflow Activities.

Free Shipping on Big Orders Example

Most shops would love it if their average order price were greater than $100. You can actually encourage that behavior using a custom shipping driver. Let's build a driver where the users will pay a base shipping fee of X and Y per item, unless the transaction is worth Z. For example, they will pay a base fee of $5 + $2 per item unless their transaction is > $100.

First, build the definition() method so you can define X, Y, and Z.

```
sub definition {
    my ( $class, $session, $definition ) = @_;
    WebGUI::Error::InvalidParam->throw(error => q{Must provide a session variable})
        unless ref $session eq 'WebGUI::Session';
    $definition ||= [];
    tie my %fields, 'Tie::IxHash', (
        baseFee => {
            fieldType    => 'float',
            label        => 'Base Fee',
            hoverHelp    => 'The amount of shipping applied to all orders.',
            defaultValue => 5.00,
        },
        perItemFee => {
            fieldType    => 'float',
            label        => 'Per Item Fee',
            hoverHelp    => 'The additional shipping charge per item.',
            defaultValue => 2.00,
        },
        freeThreshold => {
            fieldType    => 'float',
            label        => 'Free Threshold',
            hoverHelp    => 'The total amount of the order before shipping is free.',
            defaultValue => 100,
        },
    );
```

```
    push @{ $definition }, {
        name       => 'Free Shipping On Big Orders',
        properties  => \%fields,
    };
    return $class->SUPER::definition($session, $definition);
}
```

Now that you have the properties to work with, you can define your calculate() method.

```
sub calculate {
    my ($self, $cart) = @_;
        my $quantityOfShippableItems = 0;
        foreach my $item (@{$cart->getItems}) {
                my $sku = $item->getSku;
                if ($sku->isShippingRequired) {
                        $quantityOfShippableItems += $item->get('quantity');
                }
        }
        my $cost = 0;
        if ($quantityOfShippableItems > 0) {
                $cost = $self->get('baseFee') + ($quantityOfShippableItems * $self-
>get('perItemFee'));
        }
        if ($cost > $self->get('freeThreshold')) {
                return '0.00';
        }
    return sprintf "%.2f", $cost;
}
```

The finished module should then look something like this:

```
package WebGUI::Shop::ShipDriver::FreeShippingOnBigOrders;

use strict;
use base qw/WebGUI::Shop::ShipDriver/;
use WebGUI::Exception;

=head1 NAME

Package WebGUI::Shop::ShipDriver::FreeShippingOnBigOrders

```

```
=head1 DESCRIPTION

This driver allows free shipping on big orders.

=head1 SYNOPSIS

=head1 METHODS

See the master class, WebGUI::Shop::ShipDriver for information about
base methods.  These methods are customized in this class:

=cut

#-------------------------------------------------------------------

=head2 calculate ( $cart )

Returns a shipping price. Calculates the shipping price using the following formula:

    baseFee + (perItemFee * item quantity) || 0 if (cart > freeThreshold)

=head3 $cart

A WebGUI::Shop::Cart object.  The contents of the cart are analyzed to calculate the
shipping costs.  If no items in the cart require shipping, then no shipping
costs are assessed.

=cut

sub calculate {
    my ($self, $cart) = @_;
        my $quantityOfShippableItems = 0;
        foreach my $item (@{$cart->getItems}) {
                my $sku = $item->getSku;
                if ($sku->isShippingRequired) {
                        $quantityOfShippableItems += $item->get('quantity');
                }
        }
        my $cost = 0;
        if ($quantityOfShippableItems > 0) {
                $cost = $self->get('baseFee') + ($quantityOfShippableItems * $self-
>get('perItemFee'));
        }
```

```
        if ($cost > $self->get('freeThreshold')) {
                return '0.00';
        }
    return sprintf "%.2f", $cost;
}

#-------------------------------------------------------------------

=head2 definition ( $session )

Add baseFee, perItemFee, and freeThreshold properties.

=cut

sub definition {
    my ( $class, $session, $definition ) = @_;
    WebGUI::Error::InvalidParam->throw(error => q{Must provide a session variable})
        unless ref $session eq 'WebGUI::Session';
    $definition ||= [];
    tie my %fields, 'Tie::IxHash', (
        baseFee => {
            fieldType    => 'float',
            label        => 'Base Fee',
            hoverHelp    => 'The amount of shipping applied to all orders.',
            defaultValue => 5.00,
        },
        perItemFee => {
            fieldType    => 'float',
            label        => 'Per Item Fee',
            hoverHelp    => 'The additional shipping charge per item.',
            defaultValue => 2.00,
        },
        freeThreshold => {
            fieldType    => 'float',
            label        => 'Free Threshold',
            hoverHelp    => 'The total amount of the order before shipping is free.',
            defaultValue => 100,
        },
    );
    push @{ $definition }, {
        name       => 'Free Shipping On Big Orders',
        properties => \%fields,
    };
```

```
    return $class->SUPER::definition($session, $definition);
}

1;
```

Writing Tax Drivers

Like payment and shipping drivers, tax drivers allow you to calculate the amount of tax to add to an order. WebGUI already comes with two tax drivers that will handle almost any need in the US and Europe, but since taxes can be even more confusing than payment and shipping, WebGUI allows you to write your own tax driver.

API Highlights

WebGUI comes with two tax drivers. The Generic driver takes the cart's shipping address and applies tax based on a configured tax table. The EU driver handles Europe's VAT system and connects to an external system to validate VAT numbers. These are both excellent examples of fully-featured tax handling systems.

The Master Classes

Before you begin to write your own tax driver, you'll want to take a look at WebGUI::Shop::Tax and WebGUI::Shop::TaxDriver. As before, WebGUI::Shop::Tax is the management interface to tax drivers, and WebGUI::Shop::TaxDriver is the class your driver must inherit from.

Also as before, tax drivers are allowed to have their own www_ methods that can be accessed through a URL like this:

/home?shop=tax;method=do;do=**someMethod**

The **someMethod** refers to the www_**someMethod** in your tax driver.T

The Basics

There is only one important method in a tax driver: **getTaxRate**. Given a WebGUI::Asset::Sku and a WebGUI::Shop::Address, it returns a tax rate in percents (19 is 19% tax).

Since it is called on each SKU individually, you can customize the tax rate for every SKU (like some places where food is not taxed unless it is prepared for you).

There are three types of configuration for taxes. First, the driver has its own configuration in **getConfigurationScreen**. Second, the driver can expose a form for every SKU to have some tax configuration options of its own with **skuFormDefinition**. Last, the driver can expose configuration options tied to a user account with **getUserScreen**.

A Simple Tax Driver Example

This example demonstrates how to write a tax driver that lets you choose one tax rate for everything, and allows individual SKUs to be marked as tax-exempt.

To start, make the driver a subclass of the base TaxDriver.

```
package WebGUI::Shop::TaxDriver::Simple;

use strict;
use base 'WebGUI::Shop::TaxDriver';

# Return the class name so WebGUI knows what to look for
sub className { return __PACKAGE__; }
```

Next, set up a configuration screen for the shop admin to enter the tax rate.

```
sub getConfigurationScreen {
        my ( $self ) = @_;
        my $session = $self->session;
        my $hf = WebGUI::HTMLForm->new( $session );
```

You also need to handle the saving of the form, so make the form target the www_saveConfiguration method of your driver. This will make a URL like *...?shop=tax;method=do;do=saveConfiguration*.

```
        $hf->hidden( name => 'shop', value => 'tax' );
        $hf->hidden( name => 'method', value => 'do' );
        $hf->hidden( name => 'do', value => 'saveConfiguration' );
```

Finally, add the field to input the tax rate you want. Like other places in WebGUI, you can use the get() method to get a configuration value.

```
        $hf->text(
```

```
            name => 'taxRate',
            label => 'Tax Rate',
            value => $self->get('taxRate'),
    );
    $hf->submit();
    return $hf->print;
}
```

Of course, we need to make a www_saveConfiguration method to save our configuration.

```
sub www_saveConfiguration {
    my ( $self ) = @_;
    my $session = $self->session;
    my ( $form ) = $session->quick(qw{ form });
    $self->update({ taxRate => $form->get('taxRate') });
    return ''; # Send us back to the config screen
}
```

You may need to be able to mark individual SKUs as tax-exempt. To do this, override the **skuFormDefinition** method to give a hash reference of field configuration.

```
sub skuFormDefinition {
    my ( $self ) = @_;
    return {
        isTaxExempt => {
            fieldType   => 'yesNo',
            label       => “Is Tax Exempt?”
        },
    };
}
```

Finally, write the method that will return the tax rate for the item.

```
sub getTaxRate {
    my ( $self, $sku, $address ) = @_;
    if ( $sku->getTaxConfiguration( $self->className )->{ isTaxExempt } ) {
        return 0;
    }
    else {
```

```
        return $self->get('taxRate');
    }
}
```

Now, you just need to add your class to the “taxDrivers” configuration in the site configuration file and restart the server. Once a driver is chosen, the configuration forms will appear and you will be able to enter tax information.

The full module should look like this:

```
package WebGUI::Shop::TaxDriver::Simple;

use strict;
use base 'WebGUI::Shop::TaxDriver';

# Return the class name so WebGUI knows what to look for
sub className { return __PACKAGE__; }

sub getConfigurationScreen {
    my ( $self ) = @_;
    my $session = $self->session;
    my $hf = WebGUI::HTMLForm->new( $session );
    $hf->hidden( name => 'shop', value => 'tax' );
    $hf->hidden( name => 'method', value => 'do' );
    $hf->hidden( name => 'do', value => 'saveConfiguration' );
    $hf->text(
        name => 'taxRate',
        label => 'Tax Rate',
        value => $self->get('taxRate'),
    );
    $hf->submit();
    return $hf->print;
}

sub www_saveConfiguration {
    my ( $self ) = @_;
    my $session = $self->session;
    my ( $form ) = $session->quick(qw{ form });
    $self->update({ taxRate => $form->get('taxRate') });
    return '';
}
```

```
sub getTaxRate {
	my ( $self, $sku, $address ) = @_;
	if ( $sku->getTaxConfiguration( $self->className )->{ isTaxExempt } ) {
		return 0;
	}
	else {
		return $self->get('taxRate');
	}
}

sub skuFormDefinition {
	my ( $self ) = @_;
	return {
		isTaxExempt => {
			fieldType	=> 'yesNo',
			label		=> "Is Tax Exempt",
		},
	};
}

1;
```

Index