Customizing Your Satchmo Store

Chris Moffitt

Wednesday,September 9th 2009
Djangocon 2009

www.satchmoproject.com

- Introduction
- The ”Satchmo Way” aka How to Customize
  - Settings
  - Templates
  - Functions, Tags & Filters
  - Custom Shipping
  - Custom Payment
  - Signals
- Questions

What is Satchmo?

- Open Source *framework* for creating online stores
  - BSD Licensed
  - Uses Django framework
  - In development for several years (DOB April 12, 2006)
  - Dozens of live shops in production
  - Actively being developed
  - ***Designed to be extended***

Features

- Fully template driven
- Multiple payment options
  - Authorize.net, cybersource, paypal, google, etc
  - Gift Certificate support
- Multiple shipping options
  - USPS, UPS, Fedex, etc
- Supports translations
- Multi-site
- Multiple product types
  - Downloadable, subscription, variations, etc
- Tax/VAT
- Product comments/ratings
- Configurable checkout process
- Myriad of discount options
- PDF Invoice, receipt and shipping docs

The ”Satchmo Way”

How to think about extending and enhancing Satchmo the right way...

Or

”How to protect yourself so upgrades don't hurt”

First - Setting up the store

Localsite application ->

Follow recommendations
in the docs!

templates ->

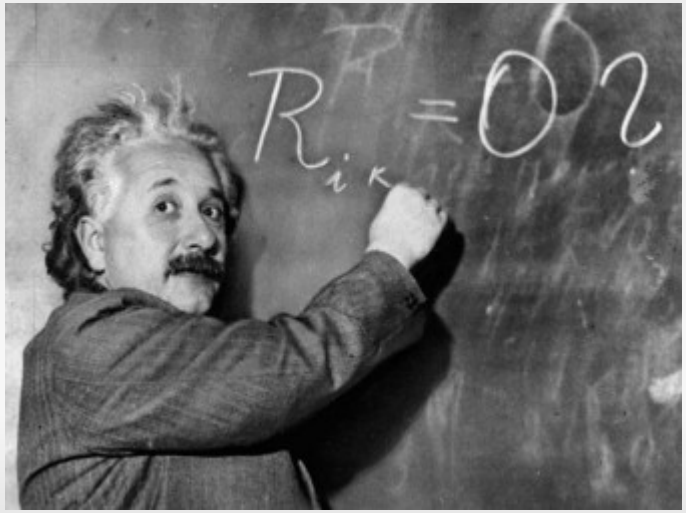
```
chris@1610-laptop:~/src/simple_store$ tree
.
|-- __init__.py
|-- local_settings.py
|-- localsite
|   |-- __init__.py
|   |-- listeners.py
|   |-- models.py
|   |-- templatetags
|   |   |-- __init__.py
|   |   `-- updated_category.py
|   |-- urls.py
|   `-- views.py
|-- manage.py
|-- settings.py
|-- simple.db
|-- static
|   |-- css
|   |   |-- blackbird.css
|   |   |-- jquery.autocomplete.css
|   |   `-- style.css
|   |-- images
|   |   |-- blackbird_icons.png
|   |   |-- blackbird_panel.png
|   |   |-- productimage-picture-default.jpg
|   |   `-- sample-logo.bmp
|   |-- js
|   |   |-- blackbird.js
|   |   `-- satchmo_store.js
|   `-- test
|-- templates
|   |-- localsite
|   |   `-- example.html
|   |-- registration
|   |   `-- new-user.txt
|   `-- shop
|       |-- base.html
|       `-- cart.html
`-- urls.py
```

Modifying Satchmo in a smart way

- Settings
- Templates
- Tags and Filters
- Custom Shipping
- Custom Payment
- Custom Signal Events

Easiest

Most complex



Settings: Rodney Dangerfield of Configuration Options

- settings.py - ”Don't get no respect”
  - Configure base url
  - Url overrrides
  - Enable various satchmo apps
  - Set logging levels
  - Lock down livesettings
  - Enable custom modules
- Also, the basis for other enhancements

SATCHMO_SETTINGS

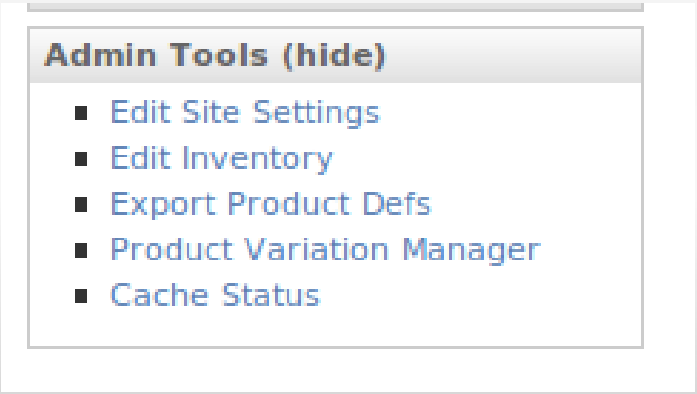
Example from settings.py file

```
119  from django.conf.urls.defaults import patterns, include
120  SATCHMO_SETTINGS = {
121                      'SHOP_BASE': '',
122                      'MULTISHOP': False,
123                      'SHOP_URLS' : patterns('', (r'^i18n/', include('l10n.urls'))),
124                      'CUSTOM_NEWSLETTER_MODULES': [],
125                      'CUSTOM_PAYMENT_MODULES': [],
126                      'CUSTOM_SHIPPING_MODULES': [],
127                      'CUSTOM_PRODUCT_MODULES': [],
128                      'CUSTOM_TAX_MODULES': [],
129  }
```

Livesettings: /settings

- Registration process
- Cart behavior
- Login required for checkout?
- Require certain info for checkout
- Default currency symbol
- Allow translations
- Collect discount code
- Set minimum order size
- Slugs for category and products

- Allow for checkout with 0 inventory
- Capture cc info
- Show featured products?
- Hide shipping choice if only 1 option?
- Tax settings
- Shipping settings
- Verbose logging



Locking down livesettings

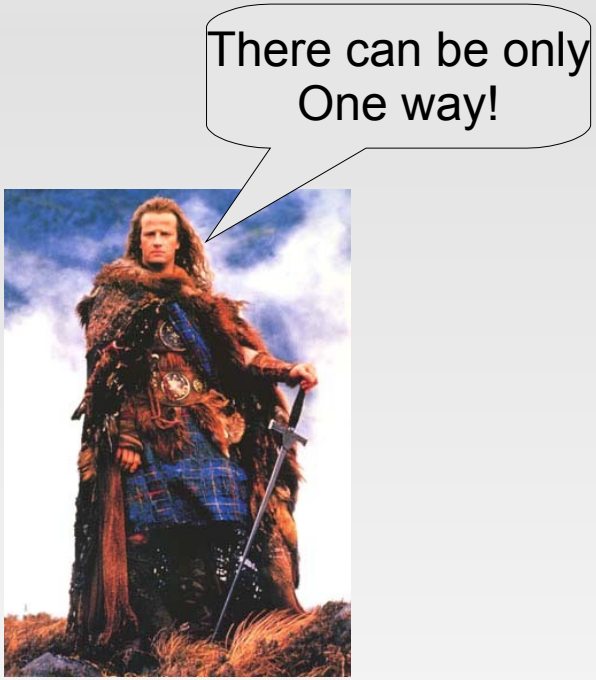
- Ability to lock down the admin settings and control from settings.py file

- Example from settings.py:

```
LIVESETTINGS_OPTIONS = {
        1 : {
                'DB' : False,
                'SETTINGS' : {
                    'PRODUCT' : {'TRACK_INVENTORY': 'True', 'CATEGORY_SLUG': 'niftycat' },
                }
            }
        }
```

Templates – Modifying Effectively

- Templates used everywhere: 140+ for HTML
- 2 Ways to modify templates



Modify the templates directly in the Satchmo source or copying over all templates

Copy only the ones you need to your app directory

Template change scenario - Add a thumbnail to the cart

- From this -

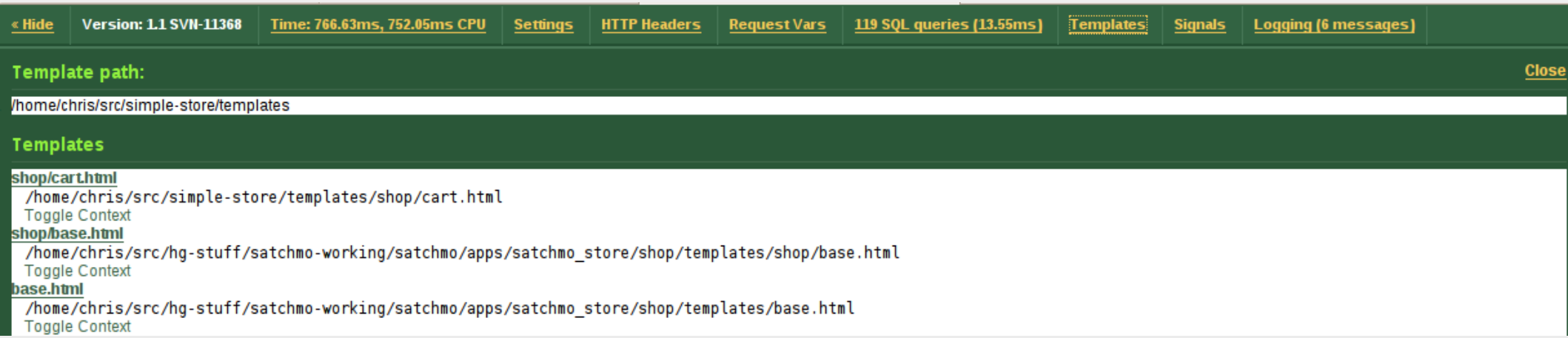
- To this -

- Without modifying Satchmo's code base

Add a thumbnail to the cart

- Setup a local application (as mentioned earlier)
  - Satchmo apps included
  - Add a template directory (in your local app):

    TEMPLATE_DIRS = (os.path.join(DIRNAME, "templates"),)
- Figure out which template to modify
  - Debug toolbar is your friend



Add a thumbnail to the cart

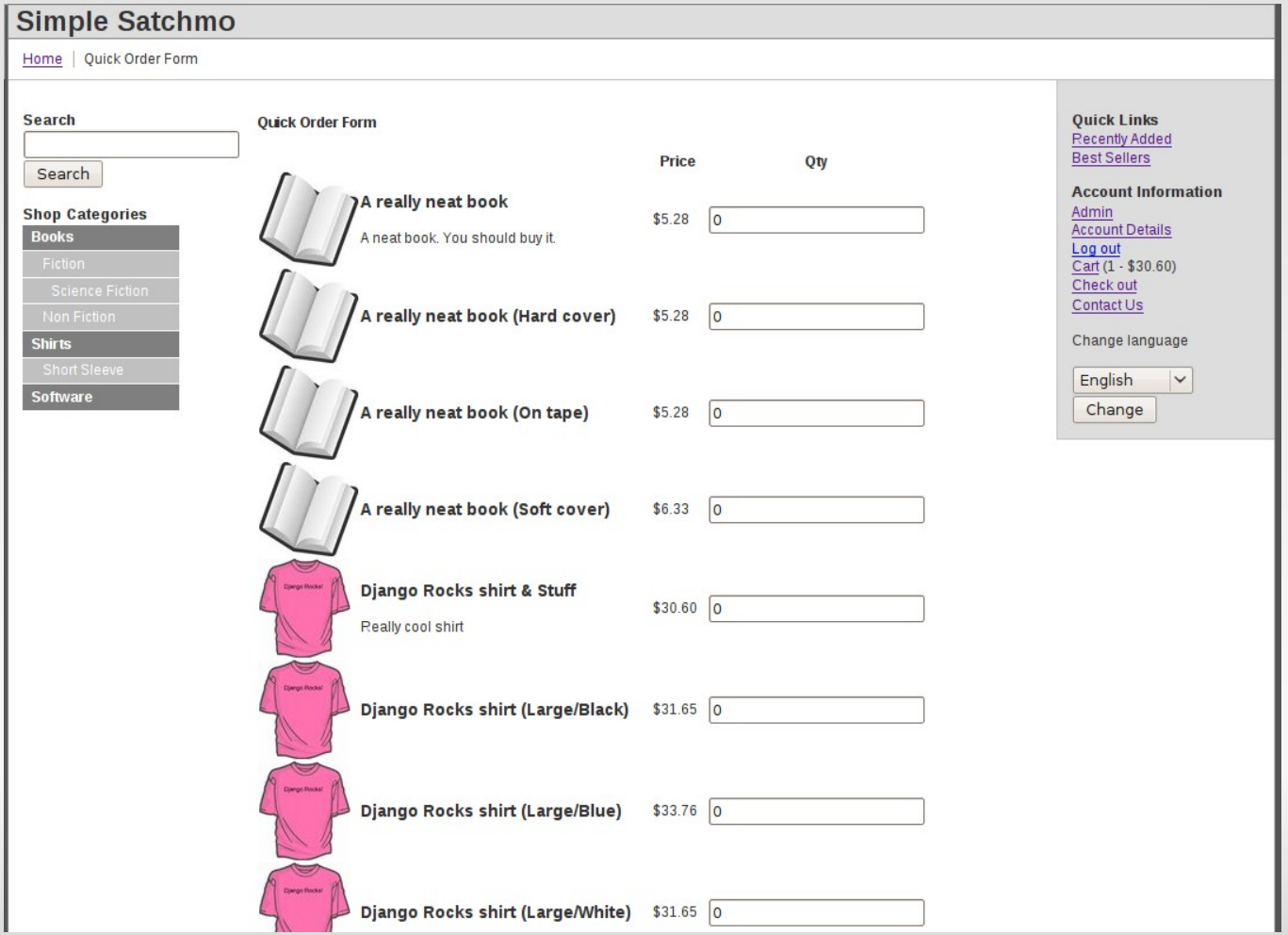
- Create /path/to/simple_store/templates/shop
- Copy cart.html from satchmo source to this directory
- Modify cart.html to load and show thumbnail

```
51
52          {% thumbnail cartitem.product.main_image.picture 25x25 as image %}
53          <td><img src="{{ image }}" width="{{ image.width }}" height="{{ image.height }}" /></td>
54          <td><a href="{{cartitem.product.get_absolute_url}}">{{ cartitem.description }}</a></td>
```

- **Now you only need to track change to 1 template!**

Select a single view to override

- Modify the quick order view (/quickorder)
  - Currently would display all products but we only want to display featured products



Quickorder default implementation

- /satchmo_store/shop/urls.py

```
10  urlpatterns += patterns('satchmo_store.shop.views',
11      (r'^$','home.home', {}, 'satchmo_shop_home'),
12      (r'^add/$', 'smart.smart_add', {}, 'satchmo_smart_add'),
13      (r'^cart/$', 'cart.display', {}, 'satchmo_cart'),
14      (r'^cart/accept/$', 'cart.agree_terms', {}, 'satchmo_cart_accept_terms'),
15      (r'^cart/add/$', 'cart.add', {}, 'satchmo_cart_add'),
16      (r'^cart/add/ajax/$', 'cart.add_ajax', {}, 'satchmo_cart_add_ajax'),
17      (r'^cart/qty/$', 'cart.set_quantity', {}, 'satchmo_cart_set_qty'),
18      (r'^cart/qty/ajax/$', 'cart.set_quantity_ajax', {}, 'satchmo_cart_set_qty_ajax'),
19      (r'^cart/remove/$', 'cart.remove', {}, 'satchmo_cart_remove'),
20      (r'^cart/remove/ajax/$', 'cart.remove_ajax', {}, 'satchmo_cart_remove_ajax'),
21      (r'^checkout/', include('payment.urls')),
22      (r'^contact/$', 'contact.form', {}, 'satchmo_contact'),
23      (r'^history/$', 'orders.order_history', {}, 'satchmo_order_history'),
24      (r'^quickorder/$', 'cart.add_multiple', {}, 'satchmo_quick_order'),
25      (r'^tracking/(?P<order_id>\d+)/$', 'orders.order_tracking', {}, 'satchmo_order_tracking'),
26      (r'^search/$', 'search.search_view', {}, 'satchmo_search'),
27
28      # Used for downloadable products.
29      (r'^download/process/(?P<download_key>\w+)/$', 'download.process', {}, 'satchmo_download_process'),
30      (r'^download/send/(?P<download_key>\w+)/$', 'download.send_file', {}, 'satchmo_download_send'),
31  )
```

Quickorder default implementation

- /satchmo_store/shop/views/cart.py

```
273 def add_multiple(request, redirect_to='satchmo_cart', products=None, template="shop/multiple_product_form.html"):
274     """Add multiple items to the cart.
275     """
276     if request.method == "POST":
277         log.debug('FORM: %s', request.POST)
278         formdata = request.POST.copy()
279         form = forms.MultipleProductForm(formdata, products=products)
280 
281         if form.is_valid():
282             cart = Cart.objects.from_request(request, create=True)
283             form.save(cart, request)
284             satchmo_cart_changed.send(cart, cart=cart, request=request)
285 
286             url = urlresolvers.reverse(redirect_to)
287             return HttpResponseRedirect(url)
288     else:
289         form = forms.MultipleProductForm(products=products)
290 
291     return render_to_response(template, RequestContext(request, {'form' : form}))
```

- Could replace entire view if we needed to but for this example, we just need to call it with a list of products

Select a single view to override

- Use replace_urlpattern from satchmo_utils.urlhelper to selectively swap out 1 url
- Sample project urls.py file-

```
1  from django.conf.urls.defaults import *
2  from satchmo_utils.urlhelper import replace_urlpattern
3
4  from satchmo_store.urls import urlpatterns
5  from product.models import Product
6
7  product_list = Product.objects.filter(featured=True)
8
9  replacement = url(r'^quickorder/$', 'satchmo_store.shop.views.cart.add_multiple',
10                   {'products': product_list}, 'satchmo_quick_order')
11 replace_urlpattern(urlpatterns, replacement)
12
```

New quickorder screen

Simple Satchmo

Home | Quick Order Form

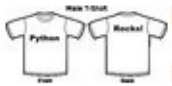
Search

Search

Shop Categories

- Books
  - Fiction
    - Science Fiction
  - Non Fiction
- Shirts
  - Short Sleeve
- Software

Quick Order Form

	Price	Qty
 **A really neat book** A neat book. You should buy it.	$5.28	0
 **Django Rocks shirt & Stuff** Really cool shirt	$30.60	0
 **Python Rocks shirt** Really cool python shirt - One Size Fits All	$20.58	0
 **Robots Attack!** Robots try to take over the world.	$8.43	0

Add to Cart

Quick Links

Recently Added

Best Sellers

Account Information

Admin

Account Details

Log out

Cart (1 - $30.60)

Check out

Contact Us

Change language

English

Change

Powered by Satchmo.

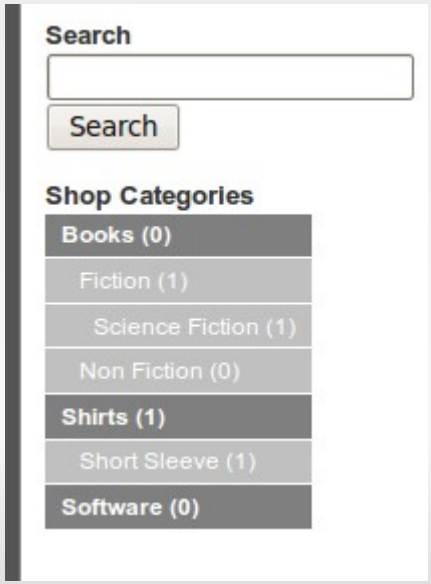
Improved functionality with no changes to the Satchmo code base!

Custom Template Tags Are Another Useful Approach

- Let's add a product count to the category list:

Steps

- Copy over the base.html template to the new template directory (/path/to/simple_store/templates/shop)
- Create a templatetags directory in the local app
- Create the new templatetag (updated_category.py)
- View the final result




```
chris@l610-laptop:~/src/simple_store$ tree
.
|-- __init__.py
|-- local_settings.py
|-- localsite
|   |-- __init__.py
|   |-- listeners.py
|   |-- models.py
|   |-- templatetags
|   |   |-- __init__.py
|   |   `-- updated_category.py
|   |-- urls.py
|   `-- views.py
|-- manage.py
|-- settings.py
|-- simple.db
|-- static
|   |-- css
|   |   |-- blackbird.css
|   |   |-- jquery.autocomplete.css
|   |   `-- style.css
|   |-- images
|   |   |-- blackbird_icons.png
|   |   |-- blackbird_panel.png
|   |   |-- productimage-picture-default.jpg
|   |   `-- sample-logo.bmp
|   |-- js
|   |   |-- blackbird.js
|   |   `-- satchmo_store.js
|   `-- test
|-- templates
|   |-- localsite
|   |   `-- example.html
|   |-- registration
|   |   `-- new-user.txt
|   `-- shop
|       |-- base.html
|       `-- cart.html
`-- urls.py
```

New Templatetag

- Slight modification of satchmo_category.py

```
1  from django.template import Library, Node, TemplateSyntaxError
2  from product.models import Category
3  from satchmo_utils.templatetags import get_filter_args
4  import logging
5
6  try:
7      from xml.etree.ElementTree import Element, SubElement, tostring
8  except ImportError:
9      from elementtree.ElementTree import Element, SubElement, tostring
10
11 register = Library()
12
13 def recurse_for_children(current_node, parent_node, active_cat, show_empty=True):
14     child_count = current_node.child.active().count()
15
16     if show_empty or child_count > 0 or current_node.product_set.count() > 0:
17         temp_parent = SubElement(parent_node, 'li')
18         attrs = {'href': current_node.get_absolute_url()}
19         if current_node == active_cat:
20             attrs["class"] = "current"
21         link = SubElement(temp_parent, 'a', attrs)
22         link.text = current_node.translated_name() + " (" + str(current_node.active_products().count()) + ")"
23
24         if child_count > 0:
25             new_parent = SubElement(temp_parent, 'ul')
26             children = current_node.child.active()
27             for child in children:
28                 recurse_for_children(child, new_parent, active_cat)
29
30 def category_tree_count(id=None):
60 register.simple_tag(category_tree_count)
61
```

Modify new base.html template

```
31          {% block sidebar %}
32          <h3>{% trans "Shop Categories" %}</h3>
33          {% block sidebar-categories %}
34          {% load updated_category %}
35          <div id="menu_container">
36              {% if category.id %}
37                  {% category_tree_count category.id %}
38              {% else %}
39                  {% if product.get_category %}
40                      {% category_tree_count product.get_category.id %}
41                  {% else %}
42                      {% category_tree_count %}
43                  {% endif %}
44              {% endif %}
45          </div>
46          {% endblock sidebar-categories %}
```

The new category tag will be displayed throughout the site.

Future changes or upgrades will be much easier to track now.

Custom Shipping

Wide variety of modules are available :

- Very simple:
  - Dummy
  - Flat
  - No
  - Per
- XML integration:
  - UPS
  - USPS
  - Fedex
  - Canada Post
- Database driven:
  - Tiered quantity
  - Tiered weight
  - Tiered price
  - Product

Use these as a basis for your custom needs

Custom Payment Modules

- Like shipping, wide variety of modules are available to show how to customize the entire checkout process:

Self contained	Simple Messaging	More complex Messaging
Autosuccess	Authorize.net	Google Checkout
COD	Cybersource	Sermepa
Dummy	Protx	Paypal IPN
Purchase Order	Trustcommerce	

- The Giftcertificate module is a unique hybrid between a product and a payment

Signals

- Signals provide integration points for altering store behavior or integrating with other apps.
  - Price query
  - Cart add
  - Contact change
  - New registration
  - See full list: http://www.satchmoproject.com/docs/svn/signals.html

Example Signal Usage

- Scenario: Send an email to the store admin, every time a new user registers.
- Alternative 1: Modify Satchmo's code to call an email function.
- Alternative 2: Use the satchmo_registration_verified signal to send the email

Making the signal work

- Create your new function in localsite/listeners.py:

```
from django.template import Context
from django.template import loader
from django.utils.translation import ugettext
from satchmo_store.shop.models import Config
from socket import error as SocketError
import logging

from django.core.mail import send_mail

log = logging.getLogger('localsite.process_registration')

def process_registration(sender, contact=None, **kwargs):
    """
    Sample showing how to do something when a registration is complete.
    In this case, we send an email to the store admin.
    """
    t = loader.get_template('registration/new-user.txt')
    shop_config = Config.objects.get_current()
    shop_email = shop_config.store_email
    subject = ugettext("New User Registration")
    c = Context({
        'first_name': contact.first_name,
        'last_name': contact.last_name,
    })
    body = t.render(c)
    try:
        send_mail(subject, body, shop_email, [shop_email], fail_silently=False)
    except SocketError, e:
        log.fatal('Error sending mail: %s' % e)
        raise IOError('Could not send email, please make sure your email settings are correct.')
```

Making the signal work

- Create your email template in templates/registration:

  {{ first_name }} {{ last_name }} has been registered on the site.

- Connect the signal with process_registration in models.py:

```
1  from localsite.listeners import process_registration
2  from satchmo_store.accounts.signals import satchmo_registration_verified
3
4  satchmo_registration_verified.connect(process_registration, sender=None)
5
```

- Wait for the emails to come in!

Summary

Following the ”Satchmo Way” and being smart about your store customizations will save you a lot of headaches when you need to upgrade your site.

When confronted with the need to customize think through all the alternatives before modifying Satchmo's code base directly.

QUESTIONS?

Image credits

- http://thehelplessdancer.files.wordpress.com/2008/12/louis-armstrong.jpg
- http://failblog.org
- http://img.timeinc.net/recipes/i/recipes/ck/03/11/pumpkin-pie-ck-549931-l.jpg
- http://palscience.com/wp-content/uploads/2009/05/albert-einstein.jpg
- http://larrycutrone.com/db1/00067/larrycutrone.com/_uimages/legends_rodne
- http://veritasatis.files.wordpress.com/2008/04/wrong-way.jpg
- http://freethoughtsociety.files.wordpress.com/2009/02/highlander.jpg
- http://detroitsportreport.com/images/Borat%20Thumbs%20Up.jpg
- http://affordablehousinginstitute.org/blogs/us/wp-content/uploads/bueller_stein_small.jpg