django-spreedly Documentation

Release 2.0

James Rivett-Carnac

July 27, 2015

Contents

These are a first run on documents, They are mostly built from doc strings, which are generally accurate. The more meta level stuff - conf and use - are slightly out of date - you are warned.

Contents:

CHAPTER 1

Configuration

1.1 Dependencies

:: pyspreedly>=2.0

1.2 Settings

1. In your settings file, add spreedly to *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'spreedly',
    ...
    )
```

2. Add your auth token and site name as well:

```
SPREEDLY_AUTH_TOKEN = 'super-secret-token'
SPREEDLY_SITE_NAME = 'site-name'
```

3. The following are in the process of being reviewed for being removal. They can also be added, they are optiona:

```
# this string will be used as the url for returning users from spreedly.
# this defaults to '/thanks/'
SPREEDLY_RETURN_URL = '/welcome/'

# the base subscription url (where users will be redirected when their subscriptions expire)
# this defaults to '/subscriptions/' if you don't add a value to your settings.
SPREEDLY_URL ='/register/'

# if you want to restrict access to your entire site based to only users with an active subscrip
# this defaults to False
SPREEDLY_USERS_ONLY = True

# URL paths that a user without a subscription can vist without being redirected to the subscrip
# these can be single pages ('/some/page/') of full directories ('/directory')
SPREEDLY_ALLOWED_PATHS = ['/login', '/logout']

# This template will be used when checking to make sure the user is using a valid email
# this default to 'confirm_email.txt' Be sure to include {{ spreedly_url }} in your template
SPREEDLY_CONFIRM_EMAIL = 'path/to/your/template.txt'
```

```
    # This subject will be used for confirmation emails
    # this defaults to "'complete your subscription to %s' % Site.object.get(id=settings.SITE_ID).na
    SPREEDLY_CONFIRM_EMAIL_SUBJECT = 'This is a new subject'

    # Where a user is directed after signing up.
    # this defaults to 'email_sent.html'
    SPREEDLY_EMAIL_SENT_TEMPLATE = 'path/to/your/template.html'

    # this is the email that will be sent to the user recieving the gift subscription
    # this default to 'gift_email.txt' Be sure to include {{ spreedly_url }} in your template
    SPREEDLY_GIFT_EMAIL = 'path/to/your/template.txt'

    # the subject for the gift confirm email
    # this defaults to 'gift subscription to %s' % Site.objects.get(id=settings.SITE_ID).name
    SPREEDLY_GIFT_EMAIL_SUBJECT = 'This is a new subject'

    # the base url for your site to be used when returning users from spreedly.
    # this default to Site.objects.get(id=settings.SITE_ID) from the django Site app.
    SPREEDLY_SITE_URL = 'something.com'
```

4. Add the middleware to your *settings.py* MIDDLEWARE_CLASSES. This will be turned into a decorator in a later version:

```
    'spreedly.middleware.SpreedlyMiddleware'
```

5. Add the following to urlpatterns in *urls.py*:

```
    import spreedly.settings as spreedly_settings
    (r'^spreedly', include('spreedly.urls')),
```

1.3 Syncdb

spreedly uses **:py:module:‘South‘** to manage database migrations. So after running *syncdb*, you must run *manage.py migrate spreedly*.

CHAPTER 2

spreedly

2.1 Models

2.1.1 models Module

class spreedly.models.**Fee**(**args*, ***kwargs*)
A Fee for a given Plan.

Attr plan ForeignKey(Plan)

Attr name CharField(max_length=100)

Attr group ForeignKey(FeeGroup)

Attr default_amount DecimalField(default=0)

add_fee(*user*, *description*, *amount=None*)
add a fee to the given user, with description and amount. if amount is not passed, then it will use *default_amount* if it is greater than 0.

if 404 or 422 are returned, the default action is not to save the line item to the db, this can be overriden with the setting SPREEDLY_SAVE_ON_FAIL, but it is not recomended as who knows what will happen.

Parameters

- **user** – the user to bill for the fee. they must be subscribed to *self.plan*
- **description** – The description of the fee to appear on the invoice
- **amount** – The amount to bill or *None*

Raises ValueError if the user is not subscribed to the plan or is subscribed to a different plan.

Raises Http404 if spreedly can't find the plan, user, etc.

Raises HttpUnprocessableEntity if spreedly raised 422 for some reason.

class spreedly.models.**FeeGroup**(*name*)

class spreedly.models.**LineItem**(**args*, ***kwargs*)
This is an instance of a fee

class spreedly.models.**Plan**(**args*, ***kwargs*)
Subscription plan

start_trial(*user*)
Check if a user is eligibile for a trial on this plan, and if so, start a plan.

Parameters **user** – user object to check

Returns py:class:*Subscription*

Raises py:exc:*Plan.NotEligibile* if the user is not eligibile

trial_eligible(*user*)
Is a customer/user eligibile for a trial? :param user: `auth.User`

class `spreedly.models.`**PlanManager**
Manager that handles syncing plans and finding enabled plans

enabled()

Returns Returns all enabled `Plans`

sync_plans()
Gets a full list of plans from spreedly, and updates the local db to match it

class `spreedly.models.`**Subscription**(**args*, ***kwargs*)
Class that manages the details for a specific `auth.User`'s subscription to a plan. Since a user can only have one subscription, this is sometimes treated as a user profile class.

add_fee(*fee*, *units*, *description*)
Add a fee to the subscription

Parameters

- **fee** – *Fee* to add to the linked user
- **units** – the number of units the charge is for (100kb, 4 nights, etc.)
- **description** – a description of the charge

Returns None

Raises Http404 if incorrect subscriber, HttpUnprocessableEntity for any other 422 error

allow_free_trial()
Allow a free Trial

Returns *Subscription*

Raises `Exception` (of some kind) if bad juju

create_complimentary_subscription(*time*, *unit*, *feature_level*)

Raises `NotImplementedError` cause it isn't implemented

ending_this_month
Will this plan end within the next 30 days

subscription_active
gets the status based on current active status and active_until

update_subscription(*data=None*)
update a subscription with supplied data

2.2 Views

2.2.1 `views` Module

class `spreedly.views.`**`EmailSent`**`(**kwargs)`
A thankyou page for after registration saying an email has been sent

class `spreedly.views.`**`PlanList`**`(**kwargs)`
inherits from `ListView` and `FormMixin`, hybrid list and subscription entry view. default template name is *spreedly_plan_list.html*, object_list name is *plans* cache's plans for 24 hours

`get``(*args, **kwargs)`
Gets the form and object list and returns a rendered template

`get_context_data``(object_list, **kwargs)`
Adds form and object list plus whatever else is passed as a kwarg to the context. :param object_list: list of :py:class:‘Plan‘s (actually queryset)

`get_queryset``()`
Gets and caches the plan list for 1 day

`model`
alias of `Plan`

class `spreedly.views.`**`SubscribeMixin`**
inherits from FormMixin, handles, get_success_url, form valid, invalid and post. Needs to be integerated into get context data and get_success_url

class `spreedly.views.`**`SubscriptionDetails`**`(**kwargs)`
view to see subscription details. takes subscription id as an optional parameter. if it is not there return the user's subscription if available, if not 404. if user.is_staff() - then you can see any Subscription details.

`model`
alias of `Subscription`

CHAPTER 3

Use

After the app is installed, you can start creating subscriptions!

The app is designed to work with the following flow:

- new user enters user information and chooses a plan
- inactive user object is created and the user is sent an email with a link to spreedly to pay for plan
- after successful payment, user is directed back to your site
- the app will check with spreedly for users status
- if the user has an active subscription, the user object will be set to active and the user will be given a login url

If you want to make your site subscription only you can set the SPREEDLY_USERS_ONLY to True. This will redirect any anonymous user (or user with an inactive subscription) who visits a page not in the SPREEDLY_ALLOWED_PATHS list to your SPREEDLY_URL

3.1 Some Important Notes

Spreedly is sent a redirect url that will check and see if the user has signed up and activate their account. **A user may not click on this link** and in that case their account won't be active, unless:

Spreedly will ping a url with subscriptions change, and django-spreedly is setup to listen for this.

in your spreedly setting is the following: 'Subscribers Changed Notification URL'

if you changed SPREEDLY_URL, you'll need to substitute that for subscriptions.

If you want to add a fallback, you can also add the following to your login view after a user is logged in (but before you check if they are active):

```
from spreedly.functions import get_subscription

if not user.is_active: get_subscription(user)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

S