django-konfera Documentation

Release 0.1

SPy o.z.

Mar 21, 2017

Contents

Contents:

CHAPTER 1

Installation

Using Pip

```
$ pip install django-konfera
```

Using the Source

Get a source tarball from pypi, unpack, then install with:

```
$ python setup.py install
```

Note: As an alternative, if you don't want to mess with any packaging tool, unpack the tarball and copy/move the modeltranslation directory to a path listed in your `PYTHONPATH` environment variable.

CHAPTER 2

Setup

TODO

CHAPTER 3

Configuration

Available Settings

Configuration options available, to modify application according your needs. Knowing this list of settings can save you a lot of time. You can define any of this `settings.py` in you project's (local) settings file.

Here's a full list of all available settings, and their default values. All settings described here can be found in `konfera/settings.py`.

GOOGLE_ANALYTICS

Default: `None`

OPTIONAL setting. Define your Google analytics code and it will be generated on all pages.

Note: Google analytics code can be overwritten per event, in event details.

Example:

```
GOOGLE_ANALYTICS = 'UA-XXXXXXXX-X'
```

NAVIGATION_ENABLED

Default: `False`

NAVIGATION_BRAND

Default: `'Konfera'`

NAVIGATION_URL

Default: `'/'`

NAVIGATION_LOGO

Default: `None`

Application supports django-sitetree navigation support, weather it should be passed to template.

CURRENCY

Default: `('€', 'EUR')`

Currency used in the application. (Currently support just one currency). Defined as tuple of Currency Symbol (Unicode block) and Currency code (ISO 4217)

TALK_LANGUAGE

Default: `(('SK', _('Slovak')), ('CZ', _('Czech')), ('EN', _('English')),)`

TALK_LANGUAGE_DEFAULT

Default: `EN`

TALK_DURATION

Default: `((5, _('5 min')), (30, _('30 min')), (45, _('45 min')),)`

LANDING_PAGE

Default: `latest_conference`

Setting is a composite of two keywords: *<timewise>_<event>*

- *<timewise>* can be: latest or earliest
- *<event>* can be: conference or meetup

possible combinations:

- latest_conference (DEFAULT)
- latest_meetup
- earliest_conference
- earliest_meetup

ORDER_REDIRECT

Default: `order_detail`

Specify url, where user will be redirected after registering the ticket.

REGISTER_EMAIL_NOTIFY

Default: `False`

Register email notification.

PROPOSAL_EMAIL_NOTIFY

Default: `False`

Notify after submitting proposal

EMAIL_NOTIFY_BCC

Default value: `[]`

Universal BCC for all notifications, MUST be empty list OR list of valid email adresses

UNPAID_ORDER_NOTIFICATION_REPEAT

Default value: `3`

How many times we should repeat the email notification

UNPAID_ORDER_NOTIFICATION_REPEAT_DELAY

Default: `5`

How long should we wait to notify about missing payment

SITE_URL

Default: `'https://www.pycon.sk'`

Absolute url base with protocol, should not contain trailing slash (/) at the end

ENABLE_ORDER_PDF_GENERATION

Default: `False`

Enable ability to store order as PDF. In order to make this functionality work, make sure django-wkhtmltopdf, with wkhtmltopdf binary.

CHAPTER 4

Contributors Guide

We are happy with any volunteers involvement in django-konfera app. If you would like to help us, there are multiple ways to do so. Depending on your skills and type of work you would like to do (doesn't have to be development), we encourage you to start with any of the following:

Write a blog, get involved on social media or make a talk

You can help out by spreading the word about django-konfera , or joining mailing list: django-konfera@pycon.sk or Gitter (if there is Slovak chatter, don't worry just start in English) to help others or share your ideas and experiences with people in community.

Update documentation

GitHub wiki is used to guide users and developers the right way. If you don't know how to do something, we probably missed it in our wiki. Documentation is a never ending process so we welcome any improvement suggestions, feel free to create issues in our bug tracker.

If you feel that our documentation needs to be modified or we missed something, feel free to submit PR, or get in touch with us at our mailing list: django-konfera@pycon.sk or Gitter (if there is Slovak chatter, don't worry just start in English).

Suggest an improvement or report bug

All issues are handled by GitHub issue tracker, if you've found a bug please create an issue for it. For security related issues please send GPG encrypted email to richard.kellner (at) pycon.sk. Public GPG key

If there is something you are missing, and wish to be implemented in django-konfera, feel free to create an issue and mark it as an enhancement.

Update django-konfera

All development is done on GitHub. If you decide to work on existing issue, **please mention in the issue comment that you are working on it so other people do not work on the same issue**. Create your fork and **in new branch update code**. Once you are happy with your changes create pull request and we will review and merge it as soon as we can. To make the life easier please do all your work in a separate branch (if there are multiple commits we do squash merge), if there is a issue for your change should include the issue number in the branch name and merge request description so they are linked on GitHub. We encourage you to write tests for your code (however this is not required), as we have continuous integration in place. Once you request merge request on GitHub, there will be an automated test run, please make sure each test passes. Sometimes it take few minutes for the Travis CI to start tests, so please be patient. Once the tests are successful, we do run test coverage, that calculates how much of the code is covered with unit tests, if you add new code without test you lower the coverage and pull request is marked that some checks were not successful. We encourage you to do the best practice and write the tests, but event if you don't we will accept the pull request.

Write a test

We realize that there is never too much testing, so you can help us by creating any form of automated testing. You will improve our continuous integration and make the project harder to break.

Getting help

If you look for help, visit our monthly meetups in Bratislava or give us a shout at mailing list: django-konfera@pycon.sk or Gitter (if there is Slovak chatter, don't worry just start in English).

CHAPTER 5

Developer's HowTo

Development standards

- We do use standard PEP8, with extended line to 119 characters.
- Each pull request is tested against our automated test suite (yes, PEP8 is one of the tests).
- Writing automated tests for the new code is preferred, but not required.

Development setup

You can either follow guide in example directory, but that is most just for testing the app from cloned repo.

This is reusable django app, which means you have to create project first. Create directory and run the following commands (in Linux, or Mac).

1. `pyvenv envs3` this will create virtual environments for you, where you can install all requirements needed
2. `source envs3/bin/activate` activate virtual environments
3. `pip install django` install out main dependency
4. `django-admin startproject pyconsk` start your own django project (feel free to name it differently)
5. `git clone git@github.com:YOUR-GITHUB-ACCOUNT/django-konfera.git` make a clone of your fork of django-konfera
6. `cd pyconsk` lets go inside the project directory
7. `ln -s ../django-konfera/konfera .` create a symbolic link to it is in PYTHONPATH and app can be found by Django
8. in pyconsk/settings.py add `konfera` into INSTALLED APPS
9. in pyconsk/settings.py add `konfera.utils.collect_view_data` into TEMPLATES, context_processors

10. `url(r'', include('konfera.urls'))` include konfera urls in project's pyconsk/urls.py file (don't forget from django.conf.ulrs import include)
11. `python manage.py migrate` execute migration so it will pre-populate the DB structure
12. `python manage.py loaddata konfera/fixtures/test_data.json` insert dummy data into DB
13. `python manage.py runserver` start development server, and check the app in browser

Development methodology

1. You create a fork of the project (you do this only once. Afterwards you already have it in your GitHub, it is your repo in which you are doing all the development).
2. Clone your fork locally `git clone git@github.com:YOUR-GITHUB-ACCOUNT/django-konfera.git` add upstream remote to be able to download updated into your fork `git remote add upstream https://github.com/pyconsk/django-konfera.git`. You don't have the right to push to upstream, but do regularly pull and push to your fork to keep it up-to-date and prevent conflicts.
3. Pick up a issue, and make a comment that you are working on it.
4. In your local git copy you create a branch: `git checkout -b XX-new-feature` (where XX is issue number).
5. Coding time:
   - Do commit how often you need. At this point doesn't matter if code is broken between commits.
   - Store your change in your repo at GitHub. You can push to server how many times you want: `git push origin XX-new-feature`.
   - Merge the code from upstream as often as you can: `git pull upstream master`. At this point we don't care about merge message, or rebase to get rid of it. We will do squash merge (in upstream master it will looks like one commit).
   - Anytime during development execute `python manage.py test konfera` to run all tests, make sure all passes before creating PR.
6. Once you are happy with your code, you click on pull request button, and select master branch in upstream and XX-new-feature branch from your repo. At this point automated tests will be run if everything is OK, if you see some errors please fix them and push your fix into your branch. This way the pull request is updated with fixes and tests are run again.
7. In case reviewer asks for changes you can do all the things mentioned in point 5. Once happy with the changes make a note in pull request to review again.
8. Your feature is approved and merged to master of upstream, so you can check out master at your local copy: `git checkout master` and pull the newly approved changes from upstream `git pull upstream master`. Pull from upstream will download your work (as one commit into master) that has been done in branch. Now you can delete your local branch `git branch --delete XX-new-feature`, and also remote one `git push origin :XX-new-feature`

Continuous Integration

Once developer changes create pull request we do automated test for supported Python and Django versions and execute all unit tests in our Travis CI. Once the pull request is merged to the master staging server is updated automatically, so you can see your changes in project on server immediately.

CHAPTER 6

Options

How to setup Google Analytics?

Django-konfera has an option to add tracking code for your application just simply use option:

GOOGLE_ANALYTICS = ‘UA-XXXXXXXX-X’

Enabling Google Analytics will autoamtically track certain events. The Projct manager can click on sponsors banner or clicking on outbound links. If you add outbound links tracking make sure you have *onclick=”ga(‘send’, ‘event’, ‘sponsor’, ‘click’, ‘sponsors-panel-{{ sponsor|slugify }}’); trackOutboundLink(‘{{ sponsor.url }}’);*

For implementing ecommerce tracking with django-konfera, first you need to enable it in your Google Analitcs settings for your account and afterwards just enable option:

GOOGLE_ANALYTICS_ECOMMERCE = True

CHAPTER 7

Talks

TODO: Describe model and functionality

Lifecycle of the talk can have multiple stages. Only talks that are marked as *Published* are displayed on the frontend of the application. Rest of the statuses are for organizers information only.

Talk statuses: * *Call For Proposals* - Submited talk proposal by the speaker. (Default status when submitting via frontend). * *Draft* - Talk added by admin, details of the talk are not finished, or speaker is not confirmed yet. * *Approved* - Talk ready to be published, but organizers do not want to display it on the frontend. * *Published* - Displayed talks on the frontend. * *Rejected* - Talk proposal did not pass. * *Withdrawn* - Speaker has to cancel the talk.

CHAPTER 8

Orders

Once user decide to visit event, ha can register ticket, which create order. Order has to paid in order to be valid ticket.

Order statuses: * *Awaiting payment* - Newly created order that hasnt been paid off yet. *Default* status for all new orders. * *Paid* - Order has been paid for, or if it is grant, it has been approved. * *Partly Paid* - Order has been paid for, but the amount of received money was less that the price. * *Expired* - Order hasn't been paid for a long time and it has expired. * *Cancelled* - Order was voided by user.

Management commands

Orders can be managed by projects *manage.py* commands.

- *show_overdue_orders* - Management command will just show orders, that are overdue (after due date with enaought notifications) and will be expired on next run of command *email_unpaid_notifications*
- *show_unpaid_orders* - Management command will just show orders, that are not paid and are after due date, or last notification is after due date and user will be notified on next run of command *email_unpaid_notifications*
- *email_unpaid_notifications* - Management command will send notifications, to orders that were set to expired, and also will notify orders that after due date, or last notification is after due date.

Sample command

```
$ ./manage.py email_unpaid_notifications
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

CHAPTER 10

Attendee Check In

1. Create a group in admin, called *Checkin*.
2. Add the users who should be able to access the check-in system to the group.

The users in *Checkin* group can now access the check-in system at */event_slug/checkin/*.

In the check-in system, you can use the search for either *First name*, *Last name* or *Email*. However, not all at once.