zipfelchappe documentation Documentation

Release 0.3.5

Stefan Reinhard

August 20, 2016

Contents

Zipfelchappe is a simple open source crowdfunding platform based on the Django framework and FeinCMS. This project is inpired by various crowdfunding pages like:

- kickstarter.com
- indiegogo.com
- wemakeit.ch

The main advantage of zipfelchappe: it's free, customizable, styleable and can be integrated in your existing website. Free means not only that you are allowed to install and use the software freely, you also decide where the money goes! Zipfelchappe does not cut a share of your pledges or raise a monthly fee.

See http://beiss-den-hai.ch for an example page running with zipfelchappe.

CHAPTER 1

Installation

1.1 Base Setup

Zipfelchappe requires at least:

- Django
- FeinCMS v1.10
- requests v1.0

Note: Zipfelchappe is used as a FeinCMS ApplicationContent. You need to have FeinCMS set up and working before you continue.

An authentication app is required as well and the sites framework must be active.

It is recommended to install zipfelchappe via pip:

```
pip install zipfelchappe
```

Add zipfelchappe to your INSTALLED_APPS:

```
INSTALLED_APPS = (
    ...
    'feincms',

    'zipfelchappe',
    'zipfelchappe.translations',
)
```

`zipfelchappe.translations` is only required if you have a multilingual setup.

Now, add zipfelchappe to your feincms application content modules. This is usually done in your projects `models.py` file:

```
Page.create_content_type(ApplicationContent, APPLICATIONS=(
    ('zipfelchappe.urls', _('Zipfelchappe projects')),
))
```

The next step is to define the content types you want use:

```
from zipfelchappe.models import Project
```

```
Project.create_content_type(RichTextContent)
Project.create_content_type(MediaFileContent)
```

Content types are FeinCMS building blocks, that allow you to use any kind of content in your projects.

1.2 Richtext Editor

Zipfelchappe has been tested with TinyMCE 4. Use the correct init template:

```
FEINCMS_RICHTEXT_INIT_TEMPLATE = 'admin/content/richtext/init_tinymce4.html'
```

1.3 Payment providers

Depending on which payment provider you plan to use, add the following modules to your `INSTALLED_APPS`:

```
'zipfelchappe.paypal',
'zipfelchappe.postfinace',
```

Payment modules also require you to add some urls to your root urls:

```
urlpatterns += patterns('',
    url(r'^paypal/', include('zipfelchappe.paypal.urls')),
    url(r'^postfinance/', include('zipfelchappe.postfinance.urls')),
)
```

If you wish to automatically charge your successfully funded projects you need to make sure that some periodic tasks get executed. You can do this with cronjobs, Celery or whatever you like.

Usually running these tasks every hour is a good idea.

Cronjobs will need to execute these commands:

```
./manage.py paypal_payments

./manage.py postfinance_payments
./manage.py postfinance_updates
```

The task are also available as pure python function if you use Celery:

```
zipfelchappe.paypal.tasks.process_payments

zipfelchappe.postfinance.tasks.process_payments
zipfelchappe.postfinance.tasks.update_payments
```

1.4 Configuration

The last step is to configure zipfelchappe according to your needs. Here is a full example with all available configuration options, you can tailor them to suit your needs:

```
# The currencies you can choose for projects (only 1 per project)
ZIPFELCHAPPE_CURRENCIES = ('CHF', 'EUR', 'USD')

# Will try to use django comments if set to None
```

```
ZIPFELCHAPPE_DISQUS_SHORTNAME = None

# Number of projects per page in project list
ZIPFELCHAPPE_PAGINATE_BY = 10

# The URL for the terms and conditions
ZIPFELCHAPPE_TERMS_URL = '/terms-and-conditions/'

# Offers a flag if someone does not wish to appear on the backer list
ZIPFELCHAPPE_ALLOW_ANONYMOUS_PLEDGES = True

# Similar to django user profiles, this allows you to store additional data
# to the backer model.
ZIPFELCHAPPE_BACKER_PROFILE = 'mybackerprofile.BackerProfileModel'

# The receivers for system emails
# Defaults to settings.MANAGERS
ZIPFELCHAPPE_MANAGERS = (('Name', 'info@my-project.com'), )

# Paypal provider settings
ZIPFELCHAPPE_PAYPAL = {
    'USERID': '',
    'PASSWORD': '',
    'SIGNATURE': '',
    'APPLICATIONID': '', # not required for testing
    'LIVE': False,
    'RECEIVERS': [{
        'email': 'whogetsthemoney@mommy.com',
        'percent': 100,
    }]
}

# Postfinance provider settings
ZIPFELCHAPPE_POSTFINANCE = {
    'PSPID': '',
    'LIVE': False,
    'SHA1_IN': '',
    'SHA1_OUT': '',
    'USERID': '', # This is the Postfinance Direct Link API user
    'PSWD': '',   # and his password
}

# If a custom user model is used, define field names for first name,
# last name and email
ZIPFELCHAPPE_USER_EMAIL_FIELD = 'email'  # defaults to 'email'
ZIPFELCHAPPE_USER_FIRST_NAME_FIELD = 'given_name'  # defaults to 'first_name'
ZIPFELCHAPPE_USER_LAST_NAME_FIELD = 'last_name'  # defaults to 'last_name'
```

CHAPTER 2

Integration

2.1 Project

Register the available regions in your models.py:

```
from zipfelchappe.models import Project

Project.register_regions(
    ('main', _('Content')),
    ('thankyou', _('Thank you')),
)
```

Add the content types like in FeinCMS:

```
Project.create_content_type(RichTextContent)
Project.create_content_type(
    MediaFileContent,
    TYPE_CHOICES=(
        ('default', _('default')),
    ))
```

2.2 Templates

To integrate zipfelchappe into your page, it's often advisable to override the base template `zipfelchappe/base.html`. When you do this, take care to support the three required blocks for zipfelchappe:

- maincontent
- sidebar
- javascript

You also need to have the feincms_page object in your template context even for the payment views which don't use the application content. Either that or make sure the payment templates don't inherit from your common feincms base template.

The simplest way to add the feincms_page object to the template context is to add the context processor:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    "feincms.context_processors.add_page_if_missing",
)
```

2.3 Migrations

Zipfelchappe does not come with any sort of migrations itself. This would be a bad idea anyway because of it's highly configurable nature. Howerver, there is no reason why you shouldn't use South or whatever you like in your project.

If you are using South, you will have to define a custom migration path for zipfelchappe. Something like:

```
SOUTH_MIGRATION_MODULES = {
    'zipfelchappe': 'yourproject.migrate.zipfelchappe',
}
```

2.4 Extensions

If you need to add some custom fields to your projects, you can leverage the FeinCMS extension mechanism. Take a look at some of the built-in exenstion and read the documentation at feincms.org.

Zipfelchappe provides a `categories` extension.

2.5 BackerProfile

To be added.

CHAPTER 3

Crowdfunding workflow

3.1 User roles

- Platform owner: Owns and operates the platform (website).
- Project owner: Created a project featured on the platform. The project owner receives the funds on successful completion of the project.
- Visitor: A visitor to the website.
- Backer: A visitor who has backed a project becomes a backer.

3.2 Project setup

1. A project owner creates an account on the plattform.
2. He creates a project.
3. The platform owner reviews and approves the project.
4. The project is open for bidding.

3.3 Bidding phase

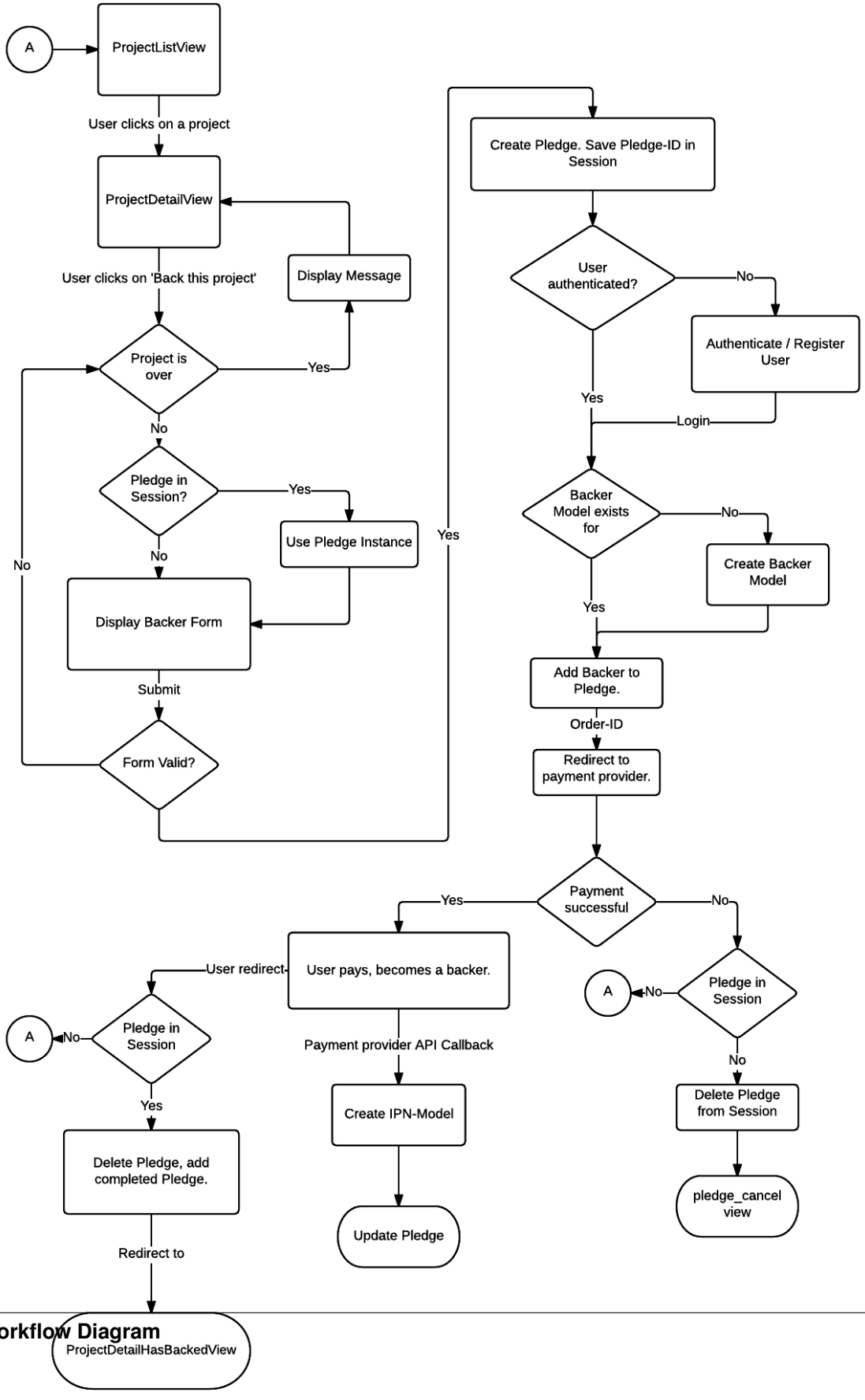
1. A visitor visits the website and looks at the project.
2. He decides to support it by clicking on the 'Back this project' button.
3. He is redirected to the backing page. On this page he has to enter the amount he wishes to back and select the reward.
4. He gets redirected to the payment provider site where he enter his payment details.
5. After successful completion he gets redirected to the thank you page which is the project detail page with 'thank_you' and the pledge id added as a query string. He is now a backer. If the visitor cancels the payment he is redirected to the detail page again. A message informs him that he had cancelled the transfer.

3.4 End phase

1. After the project end date has passed, the bidding has ended.
2. The system sends out a message to all backers informing them if the funding has succeeded or not.
3. If the project did not succeed, the preapprovals are cancelled. No money is deducted.
4. If the project did succeed, the money is transferred to the project owner.

3.5 Workflow Diagram

The diagram can be viewed (and edited) on LucidChart.


A
ProjectListView
User clicks on a project
ProjectDetailView
User clicks on 'Back this project'
Display Message
Project is over
Yes
No
Pledge in Session?
Yes
Use Pledge Instance
No
No
Display Backer Form
Submit
Form Valid?
Yes
Create Pledge. Save Pledge-ID in Session
User authenticated?
No
Authenticate / Register User
Yes
Login
Backer Model exists for
No
Create Backer Model
Yes
Add Backer to Pledge.
Order-ID
Redirect to payment provider.
Payment successful
Yes
No
User pays, becomes a backer.
Payment provider API Callback
Create IPN-Model
Update Pledge
A
No
Pledge in Session
No
Delete Pledge from Session
pledge_cancel view
User redirect
A
No
Pledge in Session
Yes
Delete Pledge, add completed Pledge.
Redirect to
ProjectDetailHasBackedView

CHAPTER 4

Event Handlers

CHAPTER 5

Payment provider

5.1 Paypal

This provider uses paypals Adaptive Payments API. Documentation can be found here.

To start developing, log on to https://developer.paypal.com and follow the steps there to get a sandbox testing account.

Once you want to go live you need to submit your application to http://x.com. This process requires a lot of infos and may take some time. You need to have your site already online and working on the sandbox before you should submit to x.com

more paypal

5.2 Postfinance

Postfinance is not a classical crowdfunding payment provider but very common in Switzerland. You find their offers https://www.postfinance.ch/en/biz/prod/eserv/epay.html

Postfinance only offers Postcard by default. To be able to accept credit cards with Postfinance, you need an additional contract with a credit card provider such as Aduno.

When you configure this provider, make sure that:

- Default operation mode ist "Authorisation"
- SHA-1 is the active hash algorithm
- You have set the correct SHA1-IN and SHA1-OUT hashes in your settings
- The Direct HTTP server-to-server urls point to the providers IPN view
- The request method for IPN messages is POST

If you want to automatically request payments for successfully funded projects in the background, you will need to purchase the optional "DirectLink" package by postfinance. This is available on the Basic and the Professional plan.

more postfinance

5.3 Custom

If you want to implement a custom payment provider, you need to follow a few quick rules:

- Provide a 'zipfelchappe_<providername>_payment' named view to you start the payment process. You can get the pledge to pay from the session
- If the payment has been authorized, set the pledge status to AUTHORIZED and redirect to the 'zipfelchappe_pledge_thankyou' view.
- If payment was canceled, redirect to the 'zipfelchappe_pledge_canceled' view
- When payment is collected, set pledge status to PAID
- If payment failed multiple times, set pledge status to FAILED

Take a look at the exisiting payment providers to get further insights.

CHAPTER 6

Using the app

6.1 Adding the fundraising module to your website

In the Dajngo admin interface, create a new page. Add an application content and select “Zipfelchappe projects”.

If you have a multilingual website, you have to do this for every language.

6.2 Adding project teasers

There are two content types which let you promote selected projects on a web page. `ProjectTeaserContent` which promotes a single project and `ProjectTeaserRowContent` to promote three projects.

To make those available in the admin interface, register them with the `Page` module:

```
from zipfelchappe.content import ProjectTeaserContent, ProjectTeaserRowContent

Page.create_content_type(ProjectTeaserContent, regions=('main',))
Page.create_content_type(ProjectTeaserRowContent, regions=('main',))
```

You have to migrate your database after adding the content types. Once the content types are registered, you can select them in the admin interface.

They use the template `zipfelchappe/project_teaser.html` and `zipfelchappe/project_teaser_row.html` respectively.

CHAPTER 7

Contributing

7.1 Set-up

Create a virtualenv and install the dependencies specified in example/requirements.txt.

7.2 Testing

To run the tests, activate the virtualenv and run `./manage.py test tests`.