AMBIENT

Tying up a loose end

How class-based emails will save your day

Ronny Vedrilla – 25.06.2024

Agenda

What’s going to happen…

+ 🖖 Brief introduction
+ ⚡Motivation
+ 🤓 History lesson
+ 🏗️ Status quo
+ 💌 Why you want to use class-based emails
+ 🧐 Critical review

AMBIENT

AMBIENT

Hello World! ✌

Who am I?

Ronny Vedrilla

- Business Information Systems at Cologne University (Diploma)
- Working with Django since 2012
- Django Software Foundation (DSF) Member
- Organiser Django Meetup Cologne

linkedin.com/in/ronny-vedrilla/

Ambient

Cologne ❤️ Django

- Full-service agency
- Specialised on web and apps
- Weapons of choice: django & reactJS / Vue.js
- Giving me the opportunity to be here

AMBIENT

AMBIENT

Motivation

AMBIENT

Why is creating and maintaining emails such a pain?

— Many developers

Motivation

Why do we talk about emails?

+ Many business applications still rely heavily on emailing when they want to inform the user
+ Even more so, if you have end-customers on your platform
+ More recent approaches like in-app notifications etc. are neat but...

Therefore, emails are an integral part of many web applications 👩‍🚀

AMBIENT

Motivation

Real-life examples

- After working on > 25 Django-based business applications, here are some of my greatest pain-points
  - HTML and plain-text email parts would differ from each other all the time
  - Forgetting to set a “globally” required variable would happen very often □ template won’t complain about an unset variable
  - Emails felt like an external API (which they are somehow) and weren’t unit-tested □ many bugs
  - Convenience features like a unsubscribe link or adding CC or Reply-To headers would be implemented in one project and not in the other

AMBIENT

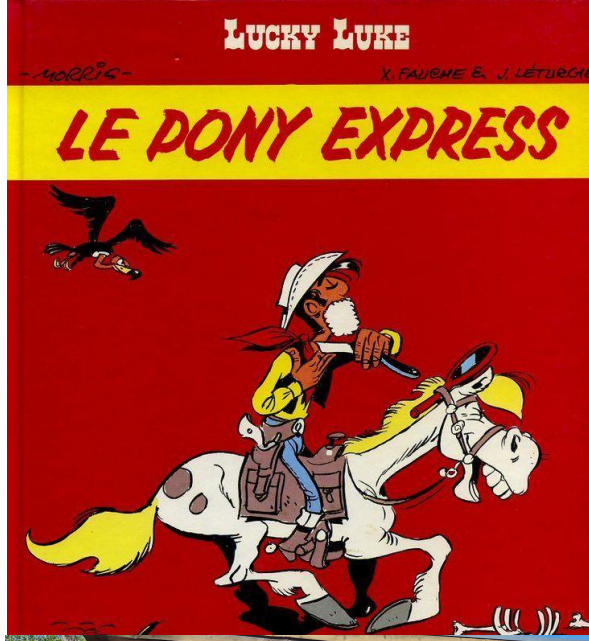
Etymology

Etymology

Why “django pony express”?

+ The pony express was a super-fast mail service in the US in 1860
+ Connected Missouri with new US state California
+ Was superseded by the telegraph in 1861

AMBIENT



AMBIENT

Status quo

Default emailing

Brought to you by “django docs” 🐎

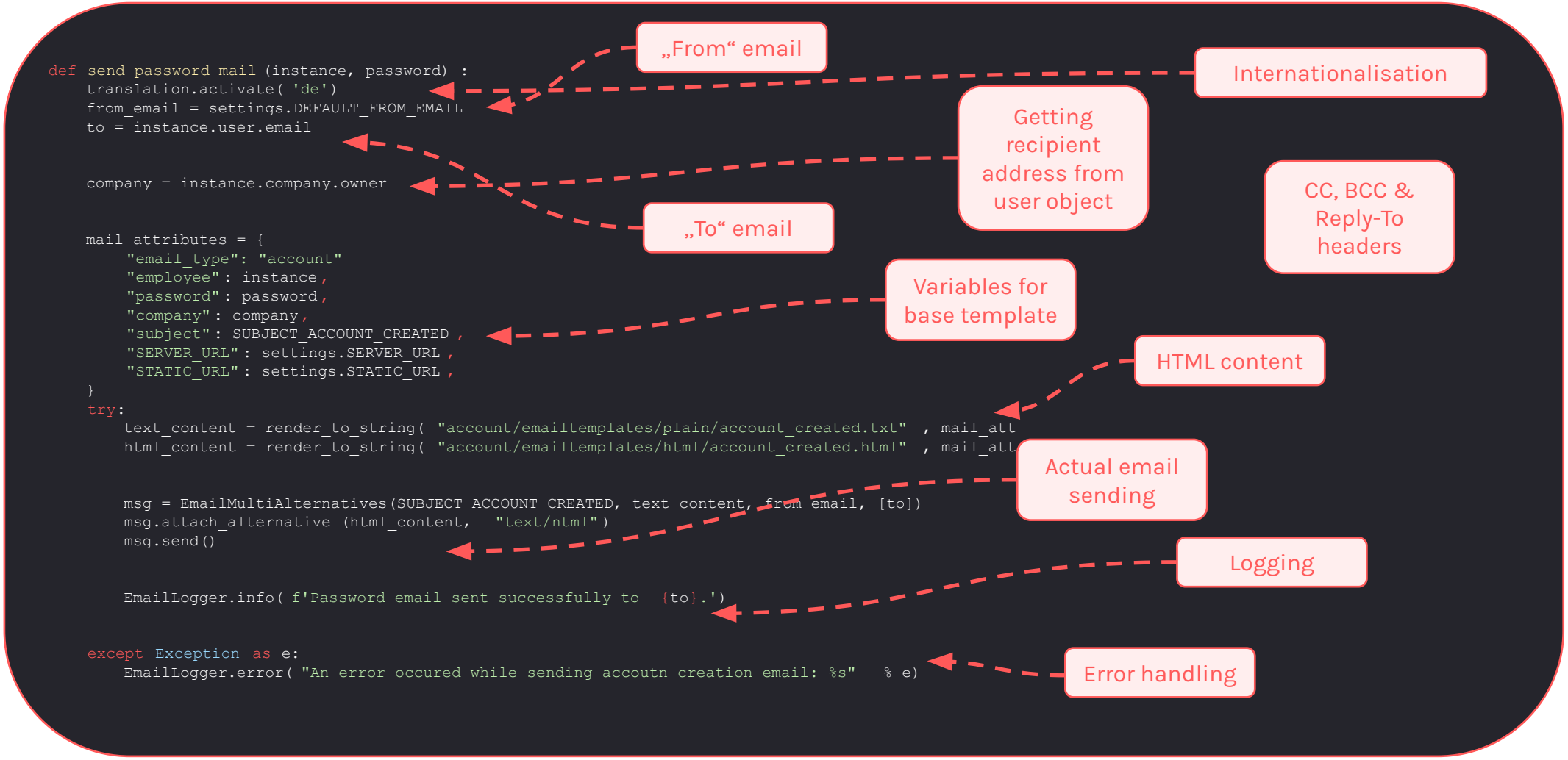
+ Simple case
+ One function call
+ Directly set parameters
+ Doesn’t match real-life-requirements

```python
from django.core.mail import send_mail

send_mail(
    'Subject here',
    'Here is the message.',
    'from@example.com',
    ['to@example.com'],
    fail_silently=False,
)
```

Default emailing

A real-world abomination... ehm... example 🤐



AMBIENT

Default emailing

Drawbacks 🥺

+ Lots of redundant code per email (not DRY (“Don’t Repeat Yourself”))
+ Easy to make a mistakes
+ Duplicate templates for HTML and plain text
+ Unencapsulated email API
+ No inheritance possible
+ Every log message looks a little bit different

AMBIENT

AMBIENT

The pony express

AMBIENT

“

Class-based emails are neat.

— Many developers at Ambient 😌

Class-based emails

Sorry, what? 🐴

+ Similar to class-based views in django
+ “BaseEmailService” provides everything you need
+ Works out-of-the-box
+ Stay DRY, only overwrite what you want to change
+ Fully documented at readthedocs.org

AMBIENT

django-pony-express

Class-based emails

What can they do for you? 🫡

- + No code duplication
- + Don’t worry about how to send an email
- + No duplicated templates
- + Less code
- + Testable 🥳
- + Super-fast creation of new emails
- + Neat 🤓
- + Attention: Take care that we are NOT deriving from “BaseEmailService”

```python
class SendPasswordMail(MyProjectBaseEmailService):
    """
    Email current credentials to requesting user.
    """
    subject = _('Acount created')
    template_name = 'account/email/account_created.html'
```

AMBIENT

Custom base email service

Required setup, simple example 🔧

+ In 99.9% of your cases, you will have a base email template
+ To encapsulate what you need for this, we need a custom base class

```python
class MyProjectBaseEmailService(BaseEmailService):
    SUBJECT_PREFIX = 'Beer Bear Inc.'
    REPLY_TO_ADDRESS = settings.HELLO_EMAIL

    def get_context_data(self) -> dict:
        """
        This method provides the required variables for the base email template. If more variables are required, just override this method
        and make sure, super() is called
        """
        context_data = super().get_context_data()
        context_data.update({
            'backend_url': f'{settings.URL_PROTOCOL}{settings.SERVER_URL}',
            'hello_email': settings.HELLO_EMAIL,
            'subject': self.subject,
            'environment': settings.SENTRY_ENVIRONMENT,
            'prod_environment': 'PROD',
        })
        return context_data
```

AMBIENT

Basic Async support

Threading for the external API

AMBIENT

+ Email providers are an external API
+ Discouraged to communicate with APIs in the main (Python) thread
+ ThreadEmailService creates a new Python thread to send the email with
+ Easy to add your own async backend
    + Celery
    + django-q / django-q2
    + Django Background Workers? 😱

```
class MyThreadBasedEmail(CustomThreadEmailService):
      """
      This will send the email in a Pyhton thread
      """
      subject = _('Thread test email')
      template_name =  'my_app/email/thread_test.html'
```

Custom base email service

Internationalisation 🏴

+ Out-of-the box internationalisation
+ Will fetch language from Django settings
+ Fully customisable

```python
# Customisation example
def get_translation(self) -> Union[str, None]:
    # If we have a recipient, use his/her language
    if self.context_data and 'recipient' in self.context_data and \
            isinstance(self.context_data['recipient'], Employee):
        return self.context_data['recipient']. language
    # If we get the language passed as a value, use it
    elif self.language is not None:
        return self.language
    # Fallback: Use default language
    return super().get_translation()
```

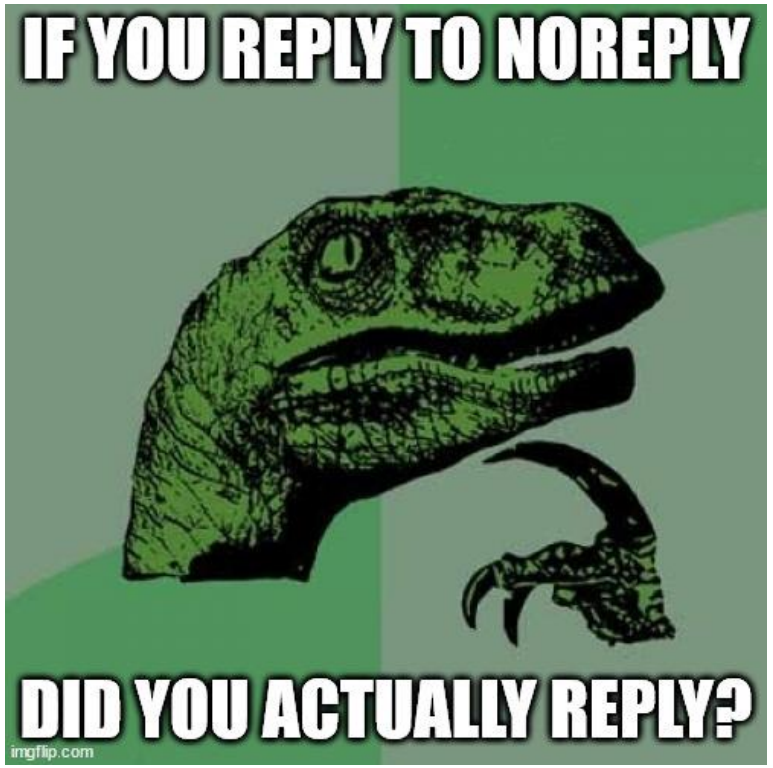
AMBIENT

Custom base email service

The Reply-to dilemma ➡️

AMBIENT

+ Usually, emails are sent from a “noreply@...” email address
+ Unfortunately, sometimes people want to reply
+ Setting “REPLY_TO_ADDRESS” will automatically add a reply-to header to every single email



Custom base email service

Additional candy 🍭

- Under-the-hood HTML to text conversion for plain text part
- “SUBJECT_PREFIX” can be set and ensures that all your different emails have the same subject prefix for a coherent look-and-feel
- Custom validation: If your email setup requires additional steps or data, enforce it!
- Additional error handling: You can customise how your emails should behave in the case of errors (for example silent fail)
- Factories: Create recipient-specific emails
- Basic logging, including a privacy switch to decide if email addresses should be logged
- Attachments

AMBIENT

AMBIENT

Proper unit-testing

Test suite

One suite to test them all 💍

- “EmailTestService” provides helpers for all types of emails, not just class-based
- Enables simple and easy-to-understand way of testing your email content
- Part of this package
- Wrapper for the django test email outbox
- Works similar to Django QuerySets

```
html_content = mail.outbox[0].alternatives[0][0]
```

```
class PeergroupFeedbackServiceTest(BaseTest):


    @classmethod
    def setUpTestData(cls):
        super().setUpTestData()


        # Instantiate email test service
        cls.email_test_service = EmailTestService()
```

AMBIENT

Test suite

Using the helper – typical example 💁

- Count emails in outbox
- Send email
- Validate that email was sent (query by subject)
- Assert recipient
- Assert both (HTML & plain text) contents with helpers
- ☐ More useful stuff in the docs 🙂

```
@freeze_time('2017-02-06')
def test_email_regular(self):
    previous_email_count =  self.email_test_service.filter( subject=self.subject).count()

    # Send email
    my_fancy_custom_code_which_triggers_an_email  ()


    # Get all mails
    list_of_emails =  self.email_test_service.filter( subject=self.subject)


    # We expect an email to be sent to senior (management) and junior (orga)
    self.assertEqual(list_of_emails.count(), previous_email_count + 1)


    # Assertions (we only have one mentor in the database)
    mail_obj = list_of_emails.first()
    self.assertIn(self.senior_dev.user.email, mail_obj.to)
    list_of_emails.assert_body_contains( self.senior_dev.user.first_name)
    list_of_emails.assert_body_contains ( 'Weekly report')
    list_of_emails.assert_body_contains( 'for calender week 5' )
    list_of_emails.assert_body_contains(reverse( 'my-url'))
```

AMBIENT

Outlook

Pun intended 🙃

Outlook

What‘s left to be done? 🔨

- Wait for Background workers and make package compatible
- Add further base classes for async dispatching methods like Celery or django-q

- Write a Trac ticket to move class-based emails to Django core 🙃

Outlook

What has happened since the DjangoCon? 🔨

AMBIENT

#35528 assigned New feature

Erstellt vor 18 Stunden
Zuletzt geändert vor 2 Stunden

Add EmailMultiAlternatives.body_contains() to aid email test assertions

Erstellt von:	**Ronny Vedrilla**	Verantwortlicher:	**Ronny Vedrilla**
Komponente:	Core (Mail)	Version:	dev
Schweregrad:	Normal	Stichworte:	email
Beobachter:		Triage Stage:	Accepted
Has patch:	ja	Needs documentation:	nein
Needs tests:	nein	Patch needs improvement:	nein
Easy pickings:	nein	UI/UX:	nein
Pull Requests:	18278 build:failure		

Beschreibung

Currently, it's very hard and tedious to assert the content of an email object. Therefore, we want to add a method "body_contains()" to "EmailMultiAlternatives" to check a search string in all available text-based alternatives (content parts, like HTML).

↳ Antworten

This method can then be easily asserted in any given unit-test.

There's a forum discussion going on about this topic: ⇨https://forum.djangoproject.com/t/improve-email-unit-testing/32044/1

I've already created a PR for a suggetion: ⇨https://github.com/django/django/pull/18278/files

AMBIENT

Critical review

Critical review

Let’s give the naysayers some attention🧐

+ There are many voices against class-based views – why should we put our money on class-based emails?
+ But what happens if the package becomes discontinued at some point in the future?
+ Why is there a need for a package like this? Shouldn’t be Django provide an easy way to deal with your email setup?

AMBIENT

Thanks!