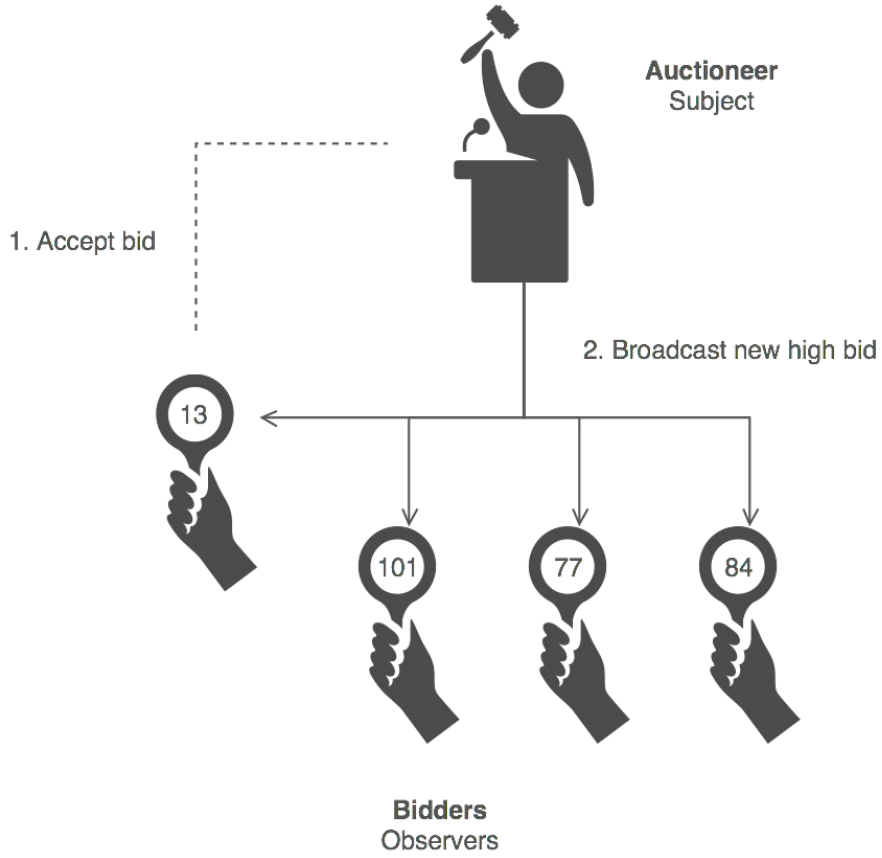
Signals de Fumaça

Desacoplanto rotinas no Django utilizando Signals

@paulo_romano

Signals, Eventos, Observer e afins:

- Dispara-se um determinado sinal em uma rotina A (***sender)***, notificando rotinas B-n (***receivers***)
- Exemplo da casa de leilões:
  - Actioner/Subject/Sender
  - Bidders/Observers/Receivers
- É o mesmo padrão utilizado para desenvolver aplicações GTK + Python;



Django Signals:

- Facilitam o desacoplamento entre aplicações e parte do framework;
- Síncronos.
- Thread-safe.
- Mamãe.

Esperando por sinais (Listening)

```
from django.db.models.signals import post_save


def post_save_handler(sender, **kwargs):
   pass

post_save.connect(post_save_handler, sender=None, weak=True, dispatch_uid=None)
```

- **sender:** Quem está enviando o sinal. Caso informado **apenas** aceitara notificações do objeto indicado
- **weak:** Indica se a referencia é fraca. Por default é fraca, e em determinados casos o garbage collector pode remover o objeto da memoria.
- **dispatch_uid:** String para identificar o receiver, evitando sua duplicidade.

Esperando por sinais (Listening)

```
from django.db.models.signals import post_save
from django.dispatch import receiver


@receiver(post_save)
def post_save_handler(sender, **kwargs):
   pass
```

- Recebe os mesmos parâmetros da Signal.connect, porem aceita uma lista de signals
- Com decorador o código fica mais limpo e legível.

Django Built-in signals

Signals	**Parâmetros**	**Observações**
pre_init	sender, args, **kwargs	args e kwargs = argumentos do modelo;
post_init	sender, instance=, **kwargs	instance = instancia do modelo criada;
pre_save	sender, instance=, raw=, using=, update_fields=, **kwargs	using = qual database alias; update_fields = quais campos serão salvos;
post_save	sender, instance=, created=, raw=, using=, update_fields=, **kwargs	
pre_delete	sender, instance=, using=, **kwargs	
post_delete	sender, instance=, using=, **kwargs	
...		

Django Built-in signals (...)

- Ha muitos outros:
  - https://docs.djangoproject.com/en/1.9/ref/signals/

Exemplo com o post_save

```
from django.db.models.signals import post_save
from django.dispatch import receiver


@receiver(post_save)
def pre_save_handler(sender, instance=None, raw=None, using=None, update_fields=None, **kwargs):
    pass
```

- sender e kwargs são obrigatórios, os nomeados não.

Criando sinais customizados

```
from django.core.signals import Signal

subscription_created = Signal(providing_args=['name'])
```

- **providing_args:** Serve apenas para documentação, não há nenhum tipo de validação.
- O local mais indicado para este código ficar é em um modulo signals.py da aplicação.

Enviando sinais (Dispatching)

Existes duas formas de enviar signals:

1. Signal.send(sender, **kwargs)
2. Signal.send_robust(sender, **kwargs)

A diferença é que Signal.send não trata exceções, simplesmente a propaga de modo que corre o risco de nem todos os ***receivers*** sejam notificados do evento. Já Signal.send_robust trata as exceções e garante que os demais ***receivers*** sejam notificados.

Exemplo de envio

```
from eventex.subscriptions.signals import subscription_created

#...

subscription_created.send(sender=create, name=subscription.name)
#subscription_created.send_robust(sender=create, name=subscription.name)
```

- **sender:** Pode ser uma class ou uma function.

Desconectando receivers

Para desconectar de um sinal:

- Signal.disconect(self, receiver=None, sender=None, weak=None, dispatch_uid=None)

O ***receiver*** pode não ser informado, caso ***dispatch_uid*** seja.

Show-me the code

Para finalizar

- Como deixar o receiver assincrono?
- Signals em python:
  - Melhor biblioteca: Blinker - http://pythonhosted.org/blinker/