TDD in Python

"Why", but mostly "how"

Hi, I’m Tudor

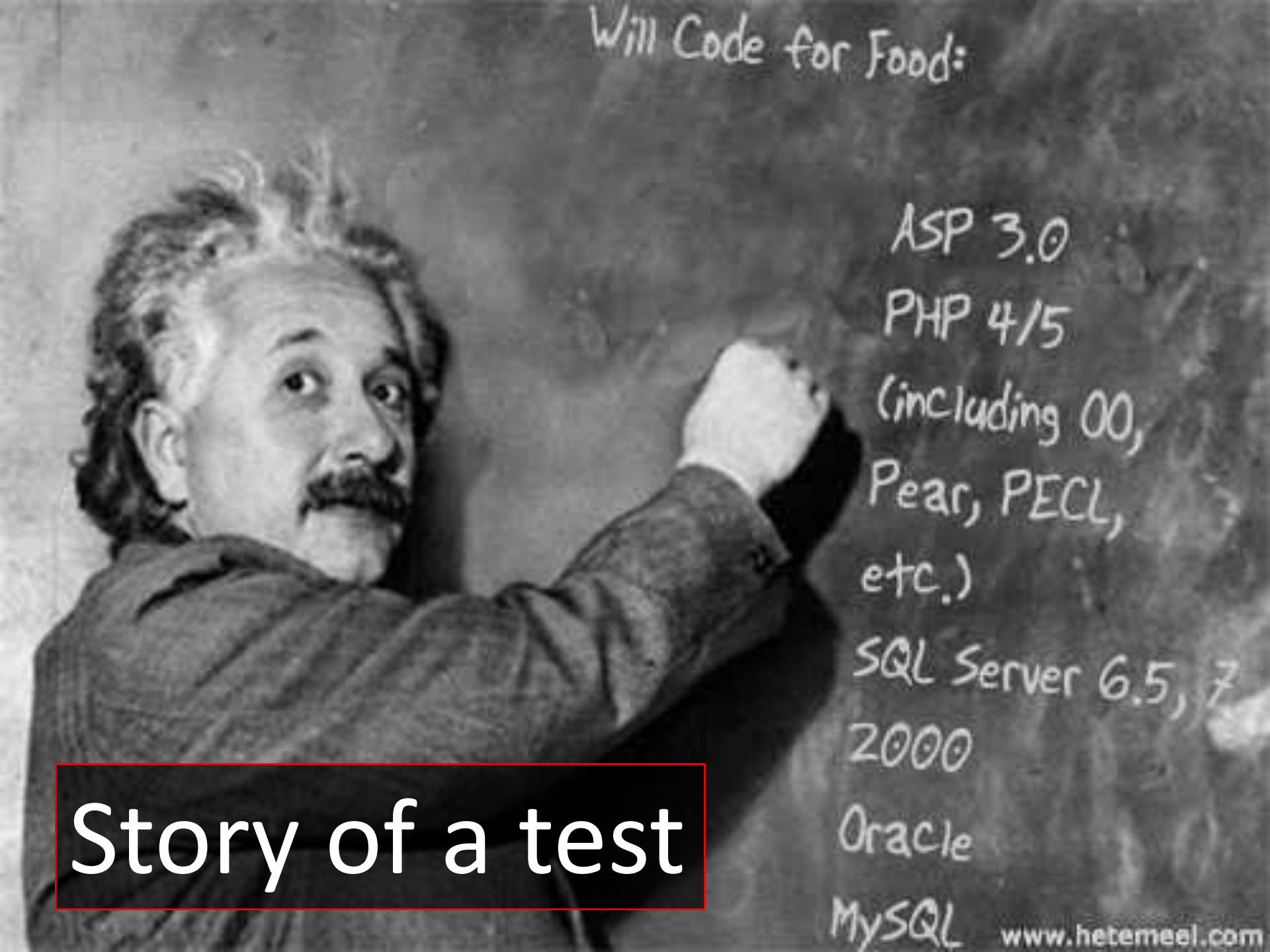
Pythonista

Odeon Consulting Group Pte Ltd – www.od-eon.com
Grok Media Pte Ltd – www.grokprojects.com

Developing web applications in
Django (because it's awesome!),
but mostly because it's Python.

"Test-driven development is not about testing. Test-driven development is about development (and design), specifically improving the quality and design of code. The resulting unit tests are just an extremely useful by-product.”

by Jason Diamond
http://bit.ly/python-tdd



Story of a test

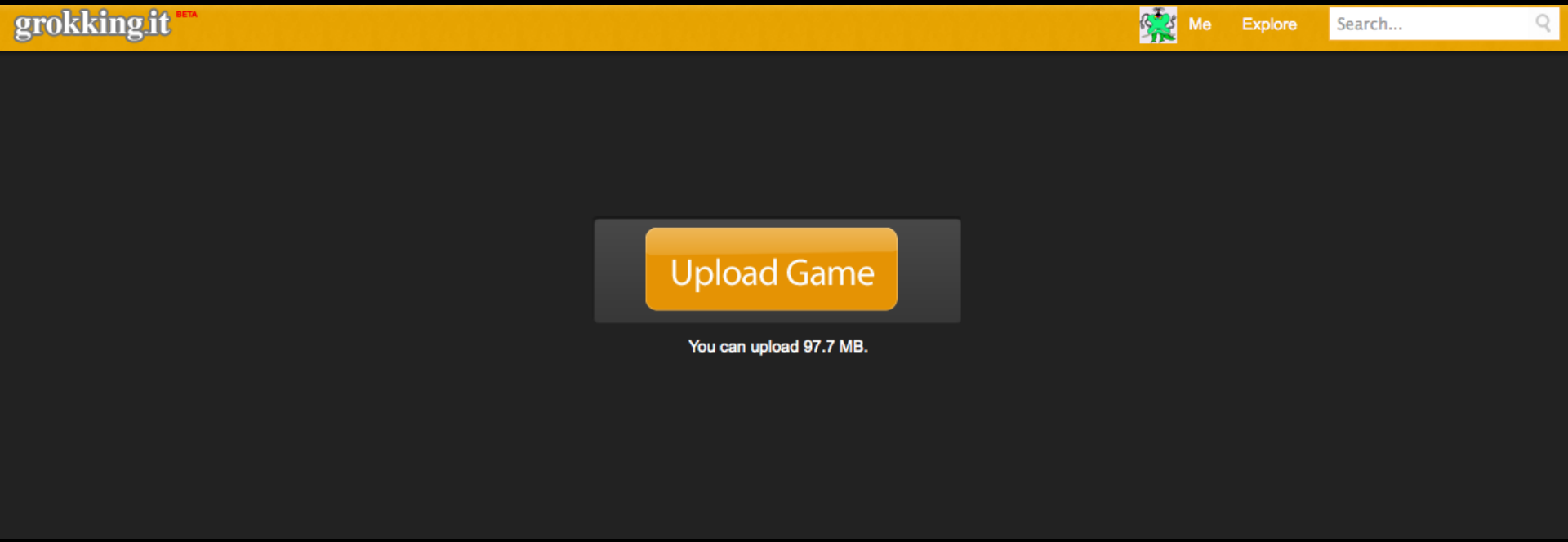
```
# create a generic user (Steve is born)
# add this user to the business group (he becomes a business owner)
# create an Organisation
# set permision for the organisation
# Steve becomes part of the organisation
# Steve defines a start and end date for the contest
# Steve creates a contest and check if it was saved (URL check)
# check if appears in organisation admin panel (URL check)
# Steve tries to score points
# gotcha! The owner of the of the contest can't score points
# Mary signs up
# Mary tries to score
# no luck. The contest is not live yet
# make the contest live
# Mary scores 3 times, trough the AJAX URL
# Stephanie is playing too
# Susan want's to play, but the contest has reached the players maximum so she can't
score
# Mary, being a simple-minded user, shouldn't have access to the contest list page (URL
permissions check)
```

extracted from the code-base of Grokking.it, http://grokking.it/

It’s important to test what users can and can NOT do, otherwise you might be surprised by loopholes in the logic.

Testing Files

- images, media content
- archives
- etc.


grokking.it
BETA
Me
Explore
Search...
Upload Game
You can upload 97.7 MB.

In Python file handling:

- built-in file-handling
- PIL for images
- zipfile for archive
- pisa for .pdf
- plethora of other libs for various filetype

In Django file handling:

- files are associated with database entries
- models.FileField()
- models.ImageField()
- ImageKit for file processing http://bit.ly/django-imagekit

```
def _get_image_for_model(path):
    """
    returns SimpleUploadedFile from a `path`.
    Reads mimetype and filecontent
    """
    from django.core.files.uploadedfile import SimpleUploadedFile
    import mimetypes as mime
    im = open(path)
    c = im.read()
    im.close()
    m = mime.guess_type(path)
    uf = SimpleUploadedFile(path, c, m)
    return uf
```

```
image_path = settings.MEDIA_ROOT + 'testdata/mega_fox.png'
image = _get_image_for_model(image_path)
megan_fox = Hottie.objects.create(name = 'Megan Fox', image =
                                  image, movie = 'Transformers')
```

Django

```
class ProfileTest(TestCase):

    def tearDown(self):
        for profile in Profile.objects.all():
            profile.delete();
```

Python

```
import os
os.remove(path)
```


TDD for better, faster
and leaner apps

```
def _upload_file(request):
    """
    Upload file to the server.
    """
    if request.method == 'POST':
        #print request.POST
        #get the user who is posting
        user_id = request.POST.get('check')
        user = User.objects.get(pk=user_id)
        title = request.POST.get('title')

        if not user.get_profile().is_business:
            print "404 Reason: Not a business user!"
            raise Http404()

        # check the number of uploaded books vs plans
        left_catalogs = left_books(user)
        if left_catalogs <= 0:
            print "404 Reason: user reached plan's catalog number limit"
            raise Http404()

        folder = request.POST.get('folder')
        if request.FILES:

            filedata = request.FILES['Filedata']
            file_name, file_ext = os.path.splitext(filedata.name)
            #file_name, file_ext = filedata.name.split('.') #might need improvment for pdf files with . in names

            # folder name is the organisation slug
            org = getUsersOrganisation(user)
            if not org:
                print "404 Reason: The user is not a member of an Organisation"
                raise Http404()
            folder_name = os.path.join(org.slug, file_name)

            # Check if tmp exists
            tmp_dir = os.path.join(settings.MEDIA_ROOT, 'tmp')
            if not os.path.isdir(tmp_dir):
                os.makedirs(tmp_dir)

            # create the book object
            b = Book(title = file_name, original = filedata, path = folder_name, user = user, organisation = org)
            b.save()

            #send the pdf file for processing
            ConvertBook.objects.create(book=b)


            #print b.pk
    # This is the tricky part, we need to send a response str back to the flash object
    # to trigger the onComplete function.
    return HttpResponse(str(b.pk))
```

Code was:

- dirty
- long
- hard to change
- prone to bugs and long debugging sessions

views.py

```
def buy(request, item_id = None):
    item = get_object_or_404(Item, id = item_id)
    if request.method == 'POST':
        form = BuyItemForm(request.POST)
        if form.is_valid():
            cd = form.cleaned_data
            user, profile = Profile.objects.create_user(cd)
            buyer = Buyer.objects.create(user = user, item = item)
            EmailValidation.objects.add(user = user, email = user.email)
            return HttpResponseRedirect(reverse('item_pay', args = [buyer.id]))
    else:
        form = BuyItemForm(initial = {'country': 'Singapore', 'city': "444"})
    return render_to_response('items/buy.html', locals(), context_instance=RequestContext(request))
```

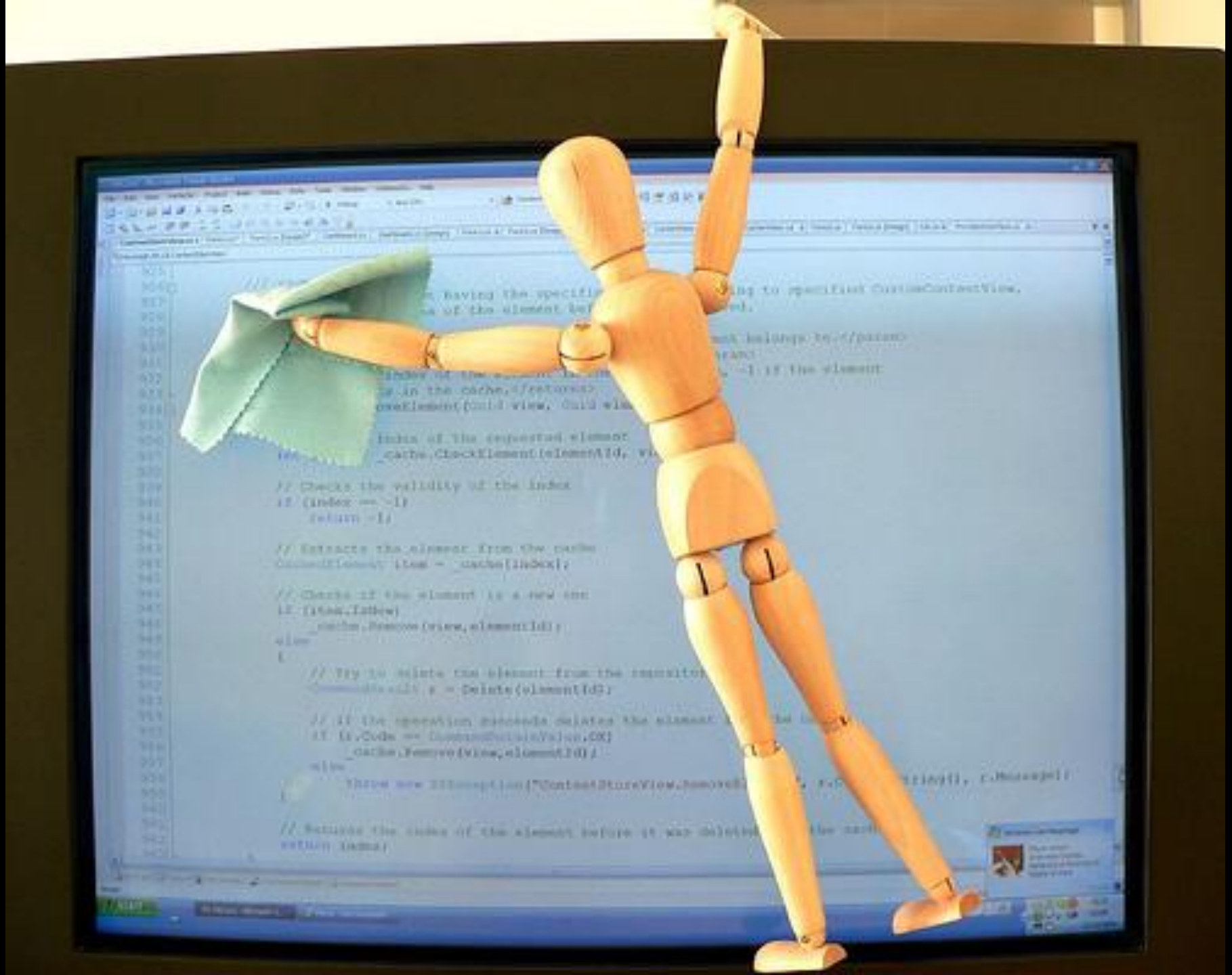
tests.py

```
def test_accepting_item(self):
        item = _create_item(self)
        user = User.objects.create_user('john', 'john@son.com', '123456')
        buyer = Buyer.objects.create(user = user, item = item)
        self.assertEquals(User.objects.count(), 1)
        self.assertEquals(Buyer.objects.count(), 1)
        self.assertEquals(item.buyer_set.count(), 1)
        self.assertEquals(item.get_percentage_sold(), 0.0)

    def test_max_buyers(self):
        item = _create_item(self)
        item.max_available = 10
        item.minimum_for_item = 5
        item.save()
        resp = self.client.get('/')
        self.assertContains(resp, 'more to go')
        for i in range(1, 12):
            user = User.objects.create_user('john%s' % i, 'john%s@son.com' % i, '123456')
            buyer = Buyer.objects.create(user = user, item = item, payed = True)
            if i < item.minimum_for_item:
                resp = self.client.get('/')
                self.assertContains(resp, 'more to go')
            elif i == item.minimum_for_item:
                self.assertTrue(item.minimum_reached())
                resp = self.client.get('/')
                self.assertContains(resp, 'is ON!')
            elif i > item.minimum_for_item and i < item.max_available:
                resp = self.client.get('/')
                self.assertContains(resp, 'is ON!')

        resp = self.client.get('/')
        self.assertContains(resp, 'Ding Ding Ding')
        self.assertEquals(Buyer.objects.count(), 10)
        self.assertEquals(Item.objects.count(), 2)
        self.assertTrue(item.is_closed())
        # double check number of people who
        # bought AND payed
        self.assertEquals(item.count_sold(), 10)
        # item buy page should be off limits now
        resp = self.client.get(reverse('buy_item', args = [item.id]))
        self.assertRedirects(resp, reverse('closed_item'))


    def test_fill_buy_form(self):
        item = _create_item(self)
        resp = self._get_and_check(reverse('buy_item', args = [item.id]))
        data = dict(email = 'cheryl@cole.com', first_name = 'Cheryl',
                    last_name = 'Cole', address = "55 Main Londong Stree",
                    zip = '123456', mobile = '12345678')
        resp = self.client.post(reverse('buy_item', args = [item.id]), data = data)
        self.assertEquals(Profile.objects.count(), 0)
        self.assertEquals(User.objects.count(), 0)
        self.assertEquals(Buyer.objects.count(), 0)
        data['city'] = '282'
        data['country'] = 'United Kingdom'
        resp = self.client.post(reverse('buy_item', args = [item.id]), data = data)
        self.assertEquals(Profile.objects.count(), 1)
        self.assertEquals(User.objects.count(), 1)
        self.assertEquals(Buyer.objects.count(), 1)
        self.assertEquals(User.objects.all()[0].first_name, "Cheryl")
        self.assertEquals(User.objects.all()[0].last_name, "Cole")
        buyer = Buyer.objects.all()[0]
        self.assertRedirects(resp, reverse('pay_item', args = [buyer.id]))
        self.assertEquals(len(mail.outbox), 1)
        # check the source of paypal redirect page
        resp = self.client.get(reverse('pay_item', args = [buyer.id]))
        #print resp.content
        # check the thank you page
        resp = self.client.get(reverse('thank_you', args = [item.id]))
        self.assertContains(resp, 'Thank You')
        self.assertContains(resp, item.count_items_left())
```



TDD enables to write lean code.

What is that?

Faster to change and with a smaller
chance of breaking other parts.

Thanks for your time!

tudor.munteanu@od-eon.com
tudor@grokprojects.com
twitter.com/tudorizer