**4\_301: Testing in Django**

# pip install factory\_boy==3.0.1

**4\_302: Factory Boy**

**model\_factories.py**

import factory

from django.test import TestCase

from django.conf import settings

from django.core.files import File

from .models import \*

class ECFactory(factory.django.DjangoModelFactory):

ec\_name = "transferase"

class Meta:

model = EC

class SequencingFactory(factory.django.DjangoModelFactory):

sequencing\_factory = "Sanger"

factory\_location = "UK"

class Meta:

model = Sequencing

class GeneFactory(factory.django.DjangoModelFactory):

gene\_id = "GeneX"

entity = "Plasmid"

start = 12

stop = 100

sense = "+"

start\_codon = "M"

sequencing = factory.SubFactory(SequencingFactory)

ec = factory.SubFactory(ECFactory)

access = 0

class Meta:

model = Gene

**4\_303: Writing API test**

**urls.py**

path('api/genes', views.GeneList.as\_view(), name='genes\_api'),

path('api/gene/<int:pk>/', api.GeneDetail.as\_view(), name='gene\_api'),

**tests.py**

import json

from django.test import TestCase

from django.urls import reverse

from django.urls import reverse\_lazy

from rest\_framework.test import APIRequestFactory

from rest\_framework.test import APITestCase

from .model\_factories import \*

from .serializers import \*

class GeneTest(APITestCase):

def test\_geneDetailReturnsSuccess(self):

gene = GeneFactory.create(pk=1, gene\_id="gene1")

url = reverse('gene\_api', kwargs={'pk': 1})

response = self.client.get(url)

response.render()

self.assertEqual(response.status\_code, 200)

def test\_geneDetailReturnFailOnBadPk(self):

gene = GeneFactory.create(pk=2, gene\_id="gene2")

url = "/api/gene/H/"

response = self.client.get(url)

self.assertEqual(response.status\_code, 404)

**tests.py**

gene1 = None

gene2 = None

good\_url = ''

bad\_url = ''

def setUp(self):

self.gene1 = GeneFactory.create(pk=1, gene\_id="gene1")

self.gene1 = GeneFactory.create(pk=2, gene\_id="gene2")

self.good\_url = reverse('gene\_api', kwargs={'pk': 1})

self.bad\_url = "/api/gene/H/"

def tearDown(self):

EC.objects.all().delete()

Sequencing.objects.all().delete()

Gene.objects.all().delete()

ECFactory.reset\_sequence(0)

SequencingFactory.reset\_sequence(0)

GeneFactory.reset\_sequence(0)

def test\_geneDetailReturnsSuccess(self):

response = self.client.get(self.good\_url, format='json')

response.render()

self.assertEqual(response.status\_code, 200)

data = json.loads(response.content)

self.assertTrue('entity' in data)

self.assertEqual(data['entity'], 'Plasmid')

def test\_geneDetailReturnFailOnBadPk(self):

response = self.client.get(self.bad\_url, format='json')

self.assertEqual(response.status\_code, 404)

**tests.py**

delete\_url = ''

self.gene3 = GeneFactory.create(pk=3, gene\_id="gene3")

self.delete\_url = reverse('gene\_api', kwargs={'pk': 3})

def test\_geneDetailDeleteIsSuccessful(self):

response = self.client.delete(self.delete\_url, format='json')

self.assertEqual(response.status\_code, 204)

**4.304 Serialiser tests**

**tests.py**

from .serializers import \*

class GeneSerialiserTest(APITestCase):

gene1 = None

geneserializer = None

def setUp(self):

self.gene1 = GeneFactory.create(pk=1, gene\_id="gene1")

self.geneserializer = GeneSerializer(instance=self.gene1)

def tearDown(self):

EC.objects.all().delete()

Sequencing.objects.all().delete()

Gene.objects.all().delete()

ECFactory.reset\_sequence(0)

SequencingFactory.reset\_sequence(0)

GeneFactory.reset\_sequence(0)

def test\_geneSerilaiserHasCorrectFields(self):

data = self.geneserializer.data

self.assertEqual(set(data.keys()), set(['gene\_id', 'sequencing',

'sense', 'start', 'stop',

'entity', 'ec',

'start\_codon']))

def test\_geneSerilaiserGeneIDIsHasCorrectData(self):

data = self.geneserializer.data

self.assertEqual(data['gene\_id'], "gene1")

**model\_factories.py**

from random import randint

class GeneFactory(factory.django.DjangoModelFactory):

gene\_id = factory.Sequence(lambda n: 'gene%d' % n+str(1))

entity = factory.Faker('sentence', nb\_words=1)

start = randint(1, 100000)

stop = start+randint(1, 10000)

**model\_factories.py**

from random import choice

entity = choice([‘Plasmid’, ‘Chromosome’])