

```
from django.apps import AppConfig
   class AboutConfig(AppConfig):
       default_auto_field = 'django.db.models.BigAutoField'
6
       name = 'about'
```

### Settings:



#### Results:



```
from django.shortcuts import render
   # Create your views here.
5 def about(request):
       return render(request, 'about/about.html')
6
```



#### Results:



```
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE',
    'fitsphere.settings')

application = get_asgi_application()
```



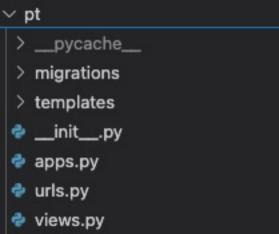
#### Results:



```
from django.conf import settings
def social_links(request):
    return {
        'social_links': settings.SOCIAL_LINKS,
```



#### Results:



```
from django.apps import AppConfig
3
   class PtConfig(AppConfig):
       default_auto_field = 'django.db.models.BigAutoField'
6
       name = 'pt'
```

### Settings:



#### Results:

```
from django.urls import path
   from . import views
3
   urlpatterns = [
       path('', views.pt, name='index'), # Home page route
```

### Settings:







#### Results:



```
1  from django.shortcuts import render, redirect
2
3
4  # Redirect the user to trainers if signed in
5  def pt(request):
6  if request.user.is_authenticated:
7  return redirect('trainers:trainers')
8  return render(request, 'pt/index.html')
9
```



#### Results:

# Code Python Linter

```
from django.urls import path
2
    from . import views
    app_name = 'trainers'
6 - urlpatterns = [
        path('', views.trainers, name='trainers'),
        path('trainer/<int:trainer_id>/',
8
            views.trainer_profile,
9
                name='trainer_profile'),
10
        path('feedback/delete/<int:id>/',
11
12
            views.delete_feedback,
                name='delete_feedback'),
13
14
        path('feedback/edit/<int:feedback_id>/',
            views.edit_feedback,
15
16
                name='edit_feedback'),
        path('trainer/<int:trainer_id>/feedback/',
17
18
            views.submit_feedback,
19
                name='submit_feedback'),
20
```

# Settings:





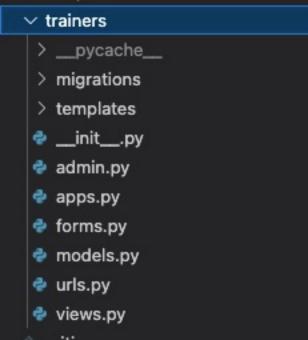


```
# TrainerFeedback Admin with Summernote
42
43 		 class TrainerFeedbackAdmin(SummernoteModelAdmin):
        # List of fields to display in the admin list view
44
        list_display = ('user', 'trainer', 'rating',
45
             'created_at', 'updated_at')
46
47
        # Filter options
        list_filter = ('trainer', 'rating')
48
49
        # Search fields
50
        search_fields = (
51
52
            'user__username',
            'trainer__first_name',
53
            'trainer__last_name',
54
55
        # Summernote for the comment field (rich text
56
            editor)
        summernote_fields = ('comment',)
57
58
        # Read-only fields for timestamps
59
        readonly_fields = ('created_at', 'updated_at')
60
61
        # Ordering feedback by created_at
62
        ordering = ('-created_at',)
63
64
65
    # Register models
66
    admin.site.register(Trainer, TrainerAdmin)
67
    admin.site.register(TrainerFeedback,
68
        TrainerFeedbackAdmin)
69
```

# Settings:



#### Results:



```
from django.apps import AppConfig

class TrainersConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'trainers'
```

# Settings:



### Results:

# Codé Python Linter institute

```
from django import forms
    from .models import TrainerFeedback
 2
 3
    class TrainerFeedbackForm(forms.ModelForm):
        class Meta:
            model = TrainerFeedback
            fields = ['rating', 'comment']
9
10
11 def clean_comment(self):
        comment = self.cleaned_data.get('comment')
12
    # You can add extra validations here if needed
13
14 - if not comment:
            raise forms. ValidationError('Comment cannot be
15
                empty.')
16
            return comment
17
```

# Settings:



### Results:

```
(4, '4'),
72
            (5, '5'),
73
        )
74
75
        user = models.ForeignKey(User, on_delete=models
76
             .CASCADE)
        trainer = models.ForeignKey(
77
78
            Trainer,
            on_delete=models.CASCADE,
79
            related_name="feedbacks"
80
81
        rating = models.IntegerField(choices=STARS)
82
        comment = models.TextField(null=True, blank=True)
83
        created_at = models.DateTimeField(auto_now_add=True
84
            )
        updated_at = models.DateTimeField(auto_now=True)
85
86
87 -
        def __str__(self):
88
            return (
                f"Feedback from {self.user.username} for
89
                    {self.trainer.first_name}"
90
                f"{self.trainer.last_name}"
            )
91
92
93 -
        class Meta:
            ordering = ['-created_at'] # Ordering feedback
94
95 -
            indexes = [
                models.Index(fields=['trainer',
96
                     'created_at'])
            ]
97
98
```

# Settings:



#### Results:

```
request, "There was an error updating
 85
                         your feedback.")
                 return redirect('trainers:trainer_profile'
 86
                      , trainer_id=trainer.id)
87
 88 -
         else:
 89
             form = TrainerFeedbackForm(instance=feedback)
90
         return render(request, 'trainer_profile.html', {
91 -
             'form': form,
92
             'feedback': feedback,
93
             'trainer': trainer,
94
         3)
95
96
97
     def delete_feedback(request, id):
98 -
         feedback = get_object_or_404(TrainerFeedback, id
99
             =id)
         trainer = feedback.trainer
100
101
102 -
         if feedback.user == request.user:
103
             feedback.delete()
             messages.success(request, "Feedback deleted
104
                 successfully!")
105 -
         else:
             messages.error(request, "You can only delete
106
                 your own feedback.")
107
         return redirect('trainers:trainer_profile',
108
             trainer_id=trainer.id)
109
```

# Settings:



#### Results: