

Steps to Customize the Admin Interface

Change Site Header and Title:

- Use the AdminSite.site_header and AdminSite.site_title attributes to customize the site header and title, respectively.
- These attributes can be set in the admin.py file of your app or project.

Customize Model Display:

- Customize how model data is displayed in the admin interface using the
 list_display attribute in your model's admin class.
- You can specify which fields to display, add custom methods, and format the display as needed.

Learn Django | Georges

Add Search and Filter Capabilities:

- Enhance the admin interface by adding search and filter capabilities using the search_fields and list_filter attributes in your model's admin class.
- search_fields allows users to search for specific records based on specified fields.
- list_filter adds filter options to the right sidebar, allowing users to filter records based on specified fields.

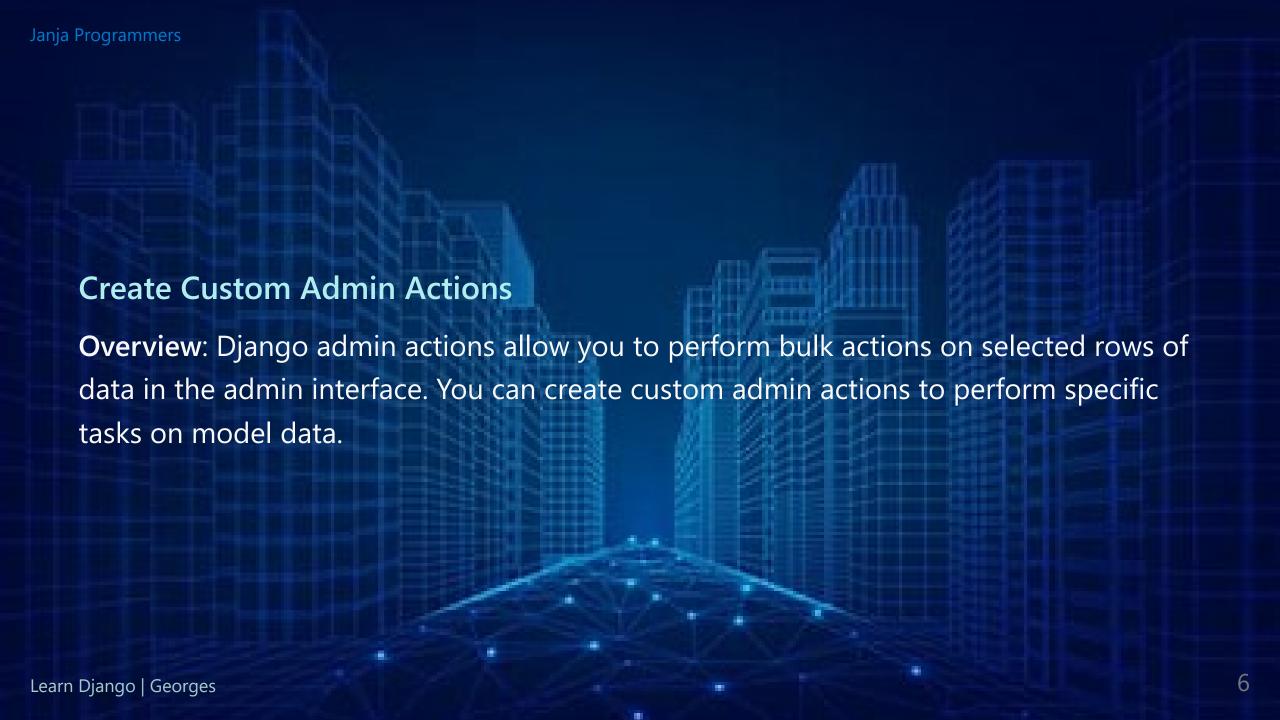
4

Snippet

```
# admin.py
from django.contrib import admin
from .models import Product

class ProductAdmin(admin.ModelAdmin):
    list_display = ('name', 'price', 'quantity')
    search_fields = ('name',)
    list_filter = ('price',)

admin.site.register(Product, ProductAdmin)
admin.site.site_header = 'My Custom Admin'
admin.site.site_title = 'My Admin Site'
```



Steps to Create Custom Admin Actions

1. Define Custom Action Functions:

- Define custom functions in your admin class to perform the desired actions on selected rows of data.
- These functions should accept the model admin, request, and queryset as parameters.

2. Register Custom Actions:

- Register the custom action functions using the actions attribute in your model's admin class.
- Specify a short description and the function name for each custom action.

Learn Django | Georges

Janja Programmers

Admin Actions Snippet:

```
# admin.py
from django.contrib import admin
from .models import Product
class ProductAdmin(admin.ModelAdmin):
    list_display = ('name', 'price', 'quantity')
    def mark as discounted(self, request, queryset):
        queryset.update(discounted=True)
    mark_as_discounted.short_description = 'Mark selected products as discounted'
    actions = ['mark as discounted']
admin.site.register(Product, ProductAdmin)
```

Usage:

- In the admin interface, select one or more rows of data.
- Choose the desired action from the "Actions" dropdown menu and click "Go".

