

Django Admin Module

COMP 8347

Usama Mir

Usama.Mir@uwindsor.ca



Django Admin Module

Topics

- Admin Site
- ModelAdmin Objects
- InlineModelAdmin Objects



Basic Steps

- Django provides an automatic **admin** interface.
 - It reads model metadata and provides a powerful interface for adding content to the site.
 - It is enabled by default (Django 1.6 onwards)
- Add '**django.contrib.admin**' to INSTALLED_APPS setting.
- Determine which models should be editable in the admin interface.
 - For each of those models, optionally create a **ModelAdmin** class.
 - Tell AdminSite about (**register**) each of your models and ModelAdmin classes.
- Hook the **AdminSite** instance into your **URLconf**.
 - `path('admin/', admin.site.urls)`, --- Already there with Django



ModelAdmin Class

- *ModelAdmin*: representation of a **model** in the **admin** interface.
 - Usually, stored in **admin.py** file in your APP.

```
from django.contrib import admin
from .models import Client
class ClientAdmin(admin.ModelAdmin):
    pass
```

```
admin.site.register(Client, ClientAdmin)
```

- If default interface is sufficient, register model directly
 - `admin.site.register(Client)`



List_display Option

- ***list_display***: controls which fields are displayed on the admin 'change list' page
 - e.g.: `list_display = ('first_name', 'last_name')`
 - Default: admin site displays only the `__str__()` representation of each object.
 - List_display can display the fields, callable, attributes, and so on.
 - Ex.

```
class PersonAdmin(admin.ModelAdmin):  
    list_display = ('first_name', 'last_name')
```



Model Field in list_display - Example

```
from django.contrib import admin
from django.db import models
from .models import Type, Item, Client, OrderItem

class ClientAdmin(admin.ModelAdmin):
    list_display = ('first_name', 'last_name')

# Register your models here.
admin.site.register(Type)
admin.site.register(Item)
admin.site.register(Client, ClientAdmin)
admin.site.register(OrderItem)
```

Action: 0 of 5 selected

<input type="checkbox"/>	FIRST NAME	LAST NAME
<input type="checkbox"/>	Mark	Smith
<input type="checkbox"/>	Prashant	Ranga



Callable in list_display – Example (Code)

```
def firstnameupper(obj):  
    return obj.first_name.upper()
```

```
class ClientAdmin(admin.ModelAdmin):  
    list_display = ['first_name', firstnameupper, 'last_name']
```

```
admin.site.register(Client, ClientAdmin)
```



Callable in list_display – Example (Result)

<input type="checkbox"/>	FIRST NAME	FIRSTNAMEUPPER	LAST NAME
<input type="checkbox"/>	Mark	MARK	Smith
<input type="checkbox"/>	Prashant	PRASHANT	Ranga
<input type="checkbox"/>	Saja	SAJA	Mansouri
<input type="checkbox"/>	Usama	USAMA	Mir
<input type="checkbox"/>	Ziad	ZIAD	Kobti



Model Attribute in list_display – Example (Code)

#models.py under the **Client** model

```
def local_client(self):  
    if self.city == 'WD':  
        return 'Yes'  
    return 'No'
```

#admin.py

```
class ClientAdmin(admin.ModelAdmin):  
    list_display = ['first_name', first_nameupper, 'last_name', 'city', 'local_client']
```

```
admin.site.register(Client, ClientAdmin)
```



Model Attribute in list_display – Example (Result)

Action: 0 of 5 selected

<input type="checkbox"/>	FIRST NAME ▲	FIRST NAME CAPS	LAST NAME	CITY	LOCAL CLIENT
<input type="checkbox"/>	Mark	MARK	Smith	Chatham	No
<input type="checkbox"/>	Prashant	PRASHANT	Ranga	Chatham	No
<input type="checkbox"/>	Saja	SAJA	Mansouri	Chatham	No
<input type="checkbox"/>	Usama	USAMA	Mir	Chatham	No
<input type="checkbox"/>	Ziad	ZIAD	Kobti	Toronto	No

5 Clients



ModelAdmin Actions

- **Actions:** simple functions that get called with a **list of objects** selected on the change list page.
 - Very useful for making same change to many objects at once.
 - The function takes 3 arguments:
 - The current **ModelAdmin**
 - An **HttpRequest** representing the current request,
 - A **QuerySet** containing the set of objects selected by the user.
- Two main steps:
 - Writing actions
 - Adding actions to **ModelAdmin**
- Example: You would like to change the city of a client in your website.



ModelAdmin Actions – Sample Model

- The Client Model from grocery website – ‘models.py’

```
class Client(User):  
    CITY_CHOICES = [  
        ('WD', 'Windsor'),  
        ('TO', 'Toronto'),  
        ('CH', 'Chatham'),  
        ('WL', 'Waterloo'),  
    ]  
    shipping_address = models.CharField(max_length=300, null=True, blank=True)  
    city = models.CharField(max_length=2, choices=CITY_CHOICES)  
    interested_in = models.ManyToManyField(Type)
```



ModelAdmin Actions – Writing Actions

#admin.py

```
from django.contrib import admin
from django.db import models
from .models import Client
```

```
#writing actions
```

```
@admin.action(description='Change your city here')
```

```
def change_city(modeladmin, request, queryset):
    queryset.update(city='CH')
```

- Also, possible to iterate over queryset

Example considering the Item Model:

```
for item in queryset:
    item.price = item.price + 1000
```



ModelAdmin Actions – Adding Actions

- To inform our ModelAdmin of the action.

#admin.py

```
class ClientAdmin(admin.ModelAdmin):  
    list_display = ['first_name', 'city']  
    ordering = ['first_name']  
    actions = [change_city]
```

```
admin.site.register(Client, ClientAdmin)
```



ModelAdmin Actions – Result

Select Client to change

Action: 0 0

<input type="checkbox"/>	FIRS	Delete selected Clients Change your city here
<input type="checkbox"/>	Mark	
<input type="checkbox"/>	Prashant	
<input type="checkbox"/>	Saja	
<input type="checkbox"/>	Usama	
<input type="checkbox"/>	Ziad	

5 Clients



Fields Options

- Used to make simple changes in the layout of fields in the forms.
 - showing a subset of the available fields, modifying their order or grouping them in rows

- Ex. Ordering the fields according to first name

#admin.py

```
class ClientAdmin(admin.ModelAdmin):  
    list_display = ['first_name', 'last_name', 'city']  
    ordering = ['first_name']
```



<input type="checkbox"/>	FIRST NAME
<input type="checkbox"/>	Mark
<input type="checkbox"/>	Prashant
<input type="checkbox"/>	Saja
<input type="checkbox"/>	Usama
<input type="checkbox"/>	Ziad



Inlines

- **Inlines:** Provides admin interface the ability to edit models on the same page as a parent model.
 - Two Subclasses: **TabularInline** and **StackedInline**
 - The difference between these two is merely the template used to render them
 - To edit the cars made by a company on the company page: Add inlines to a model

#admin.py

```
class ItemInline(admin.TabularInline):  
    model = Item  
  
class TypeAdmin(admin.ModelAdmin):  
    inlines = [  
        ItemInline  
    ]  
  
# Register your models here.  
admin.site.register(Type, TypeAdmin)  
admin.site.register(Item)
```



Inlines – Tabular

Result on Admin

Django administration

WELCOM

Home › Myapp1 › Types › Candies and Chocolates

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

MYAPP1

Clients + Add

Items + Add

Order items + Add

Types + Add

Change type

Candies and Chocolates

Name: Candies and Chocolates

ITEMS

NAME	PRICE	STOCK	AVAILABLE	DESCRIPTION
Trident Gum				
<input type="text" value="Trident Gum"/>	<input type="text" value="1.99"/>	<input type="text" value="12"/>	<input checked="" type="checkbox"/>	
<input type="text"/>	<input type="text"/>	<input type="text" value="100"/>	<input checked="" type="checkbox"/>	

Inlines – Stacked

Result on Admin

Django administration

WELCOME, **USAMAMIR**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) › [Myapp1](#) › [Types](#) › Candies and Chocolates

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add
MYAPP1	
Clients	+ Add
Items	+ Add
Order items	+ Add
Types	+ Add

Change type

HISTORY

Candies and Chocolates

Name:

ITEMS

Item: Trident Gum ☐ Delete

Name:

Price:

Stock:

☒ Available

Description:



References

- https://docs.djangoproject.com/en/5.0/ref/contrib/admin/#django.contrib.admin.ModelAdmin.list_display
- <https://docs.djangoproject.com/en/5.0/ref/contrib/admin/>
- <https://docs.djangoproject.com/en/5.0/ref/contrib/admin/actions/>
- <https://simpleisbetterthancomplex.com/tutorial/2017/03/14/how-to-create-django-admin-list-actions.html>
- https://books.agiliq.com/projects/django-admin-cookbook/en/latest/remove_delete_selected.html
- Slides from Dr. Arunita and Dr. Saja
- Python Web Development with Django, by J. Forcier et al.

