

Django Admin Module

COMP 8347

Slides prepared by Dr. Arunita Jaekel
arunita@uwindSOR.ca



Django Admin Module

- Topics
 - Admin Site
 - ModelAdmin Objects
 - Registering objects
 - Options and methods
 - InlineModelAdmin Objects
 - Options



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(r'admin/', admin.site.urls),`



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 Author, Book
class AuthorAdmin(admin.ModelAdmin):
 pass
admin.site.register(Author, AuthorAdmin)
```
- If default interface is sufficient, register model directly
    - `admin.site.register(Book)`



# 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 a only the `__str__()` representation of each object.
  - Four possible values can be used in ***list\_display***
    - 1.A **field** of the corresponding model
    - 2.A **callable** that accepts one parameter for the model instance.
    - 3.A string representing an **attribute** on the ModelAdmin.
    - 4.A string representing an **attribute** on the model.

| MYAPP   |                       |                          |
|---------|-----------------------|--------------------------|
| Authors | <a href="#">+ Add</a> | <a href="#">✎ Change</a> |
| Books   | <a href="#">+ Add</a> | <a href="#">✎ Change</a> |

Action:

0 of 4 selected

- ☐ BOOK
- ☐ Travel Stories
- ☐ Pride and Prejudice



# Model Field in list\_display

## models.py

```
class Employee(models.Model):
 name =
models.CharField(max_length=50)
 age = models.IntegerField()
 email =
models.EmailField(max_length=100)
 start_date = models.DateField()

 def __str__(self):
 return self.name
```

## admin.py

```
class EmployeeAdmin(admin.ModelAdmin):
 list_display = ('name', 'age')
```

Action:

1 of 7 selected

|                          |            |
|--------------------------|------------|
| <input type="checkbox"/> | EMPLOYEE   |
| <input type="checkbox"/> |            |
| <input type="checkbox"/> | Anne Jones |
| <input type="checkbox"/> | Bill King  |
| <input type="checkbox"/> | Mary Smith |

|                          |            |     |
|--------------------------|------------|-----|
| <input type="checkbox"/> | NAME       | AGE |
| <input type="checkbox"/> |            | 42  |
| <input type="checkbox"/> | Anne Jones | 30  |
| <input type="checkbox"/> | Bill King  | 26  |
| <input type="checkbox"/> | Mary Smith | 24  |



# Callable in list\_display

**admin.py**

```
def upper_case_name(obj):
 return obj.name.upper()
```

```
class EmployeeAdmin(admin.ModelAdmin):
 list_display = ('name', upper_case_name, 'age',)
```

```
class AuthorAdmin(admin.ModelAdmin):
 list_display = ('name', upper_case_name, 'city')

Register your models here.
admin.site.register(Author, AuthorAdmin)
admin.site.register(Employee, EmployeeAdmin)
```



# ModelAdmin Attribute in list\_display

```
class EmployeeAdmin(admin.ModelAdmin):
 list_display = ('name', uppper_case_name, 'age', 'status)

 def status(self, obj):
 if datetime.date.today() - obj.start_date < datetime.timedelta(180):
 return 'Trainee'
 else:
 return 'Regular'
 status.short_description = 'Employee Status'

Register your models here.

...
```





# Model Attribute in list\_display

## models.py

```
class Author(models.Model):
 name = models.CharField(max_length=50)
 city = models.CharField(max_length=20, default='Windsor')
 books = models.ManyToManyField(Book)

 def local_author(self):
 if self.city == 'Windsor':
 return 'Yes'
 return 'No'
```

## admin.py

```
class AuthorAdmin(admin.ModelAdmin):
 list_display = (name, uppper_case_name, 'local_author')
```

*# Register your models here.*



# 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 want to update **status** of several articles at once.
  - Relevant models and functions defined in following slides.

Action:  ▼

0 of 4 selected

|                          |                     |
|--------------------------|---------------------|
| <input type="checkbox"/> | BOOK                |
| <input type="checkbox"/> | Travel Stories      |
| <input type="checkbox"/> | Pride and Prejudice |



# Example

# Book model

```
class Book(models.Model):
 STATUS_CHOICES = (
 (0, 'In stock'),
 (1, 'Available soon'),
 (2, 'Not Available'),
)
 title = models.CharField(max_length=100)
 length = models.IntegerField()
 pub_date = models.DateField()
 status =
models.IntegerField(choices=STATUS_CHOICES)
```



# Writing Actions

# Book model

```
class Book(models.Model):
 STATUS_CHOICES = (
 (0, 'In stock'),
 (1, 'Available soon'),
 (2, 'Not Available'),
)
 title =
models.CharField(max_lengt
h=100)
 length =
models.IntegerField()
 pub_date =
models.DateField()
 status =
models.IntegerField(choices=
STATUS_CHOICES)
```

```
def make_available(modeladmin, request,
queryset):
```

```
 queryset.update(status=0)
```

- Default name in action list → “Make available”

- Also possible to iterate over queryset

```
for obj in queryset:
```

```
 obj.status = 0
```

- Provide friendly description in action list.

```
def make_available (modeladmin, request,
queryset):
```

```
 queryset.update(status=0)
```

```
 make_available.short_description = "Mark as
available"
```



# Adding Actions

- To inform our ModelAdmin of the action:
  - Add the action to the list of available actions for the object.
  - “delete selected objects” action available to all models

```
from django.contrib import admin
from models import Book
```

```
def make_available(modeladmin, request, queryset):
 queryset.update(status=0)
 return
make_available.short_description = 'Mark as available' #
```

```
class BookAdmin(admin.ModelAdmin):
 list_display = ('title', 'status')
 actions = [make_available]
```

```
Register your models here.
admin.site.register(Book, BookAdmin)
```



# Fields Option

- 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
- CAUTION: If a **required** field is excluded, it will cause ERROR when trying to save the object.
  - If only blank=True in model → ERROR
  - If a default value is provided in the model → OK.

```
class BookAdmin(admin.ModelAdmin):
```

```
 fields = ('title', 'length', 'pub_date')
```

- displays only the above three fields for the model, in the order specified.
- To show multiple fields on the same line, wrap those fields in their own tuple

```
class BookAdmin(admin.ModelAdmin):
```

```
 fields = ('title', ('length', 'pub_date'))
```



# 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

# Suppose two models are defined  
from django.db import models

```
class Company(models.Model):
 co_name =
 models.CharField(max_length=50)
```

```
class Car(models.Model):
 type =
 models.CharField(max_length=20)
 company =
 models.ForeignKey(Company)
```

from django.contrib import admin

```
class CarInline(admin.TabularInline):
 model = Car
```

```
class
CompanyAdmin(admin.ModelAdmin):
 inlines = [
 CarInline,
]
```



# Summary

- Admin Site
- ModelAdmin Objects
  - Registering objects
- Options and methods
  - Actions, fields and fieldsets
- InlineModelAdmin Objects
  - TabularInline
  - StackedInline





- [1] <https://docs.djangoproject.com/en//ref/contrib/admin/>

