



Accelerating Information Technology Innovation

<http://aiti.mit.edu>

Nigeria Summer 2012
Lecture DJ3 – Django Admin

Django Admin

- Used for inputting, editing, and deleting data from your application
- Saves you from manually creating admin forms
- Automatically generated based on your models
- Customizable through `admin.py`

Most important rule

- Django **admin**
- Not Django user

Django Admin Nevers

- Never:
 - Give normal users access to django admin
 - Give anybody access that you don't 100% trust

Blog example

- If you are the only blogger, you can use the admin interface
- If you provide a blogging service, you need to make a user interface
- You **MUST** create a separate interface for users to add comments
- You can use Django Admin to clean up comments

Admin pages

- Home (All Models that are registered)
 - List (all objects of that Model)
 - Details (all attributes of that object)

Ex:

- Home
 - Blogs
 - Blog post

Default admin

```
class Book(models.Model):
    title = models.CharField(max_length=100)
    authors = models.ManyToManyField(Author)
    publisher = models.ForeignKey(Publisher)
    publication_date = models.DateField()
    def __unicode__(self):
        return self.title

admin.site.register(Book)
```

Extended Admin

```
class Book(models.Model):
    title = models.CharField(max_length=100)
    authors = models.ManyToManyField(Author)
    publisher = models.ForeignKey(Publisher)
    publication_date = models.DateField()
    def __unicode__(self):
        return self.title
```

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

```
Admin.site.register(Book, BookAdmin)
```


Extended Admin Example

```
Class BookAdmin(admin.ModelAdmin):  
    list_display = ( 'title' , ' publisher' ,  
                    ' publication_date' )  
    list_filter = ( 'publisher' ,  
                   ' publication_date' )  
    search_fields = ( 'title' , 'publisher' )  
    ordering = ( 'title' , '-publication_date' )
```

Extending your model

- Goal: display the first 10 letters of a book title
- Solution:

In your model, create a method:

```
def title_first_10(self):  
    return self.title[:10]
```

In the admin class, add:

```
list_display = ( 'title_first_10' )
```

Inlines

- On the admin pages, you may want to see all the Book objects that relate to one Author.
- Django Admin lets you put this all on one page with minimal effort

Inline Syntax

```
class BookInline(admin.TabularInline):  
    model = Book  
  
class AuthorAdmin(admin.ModelAdmin):  
    inlines = [BookInline]
```

(You can use either TabularInline or StackedInline)

We want this:

Django administration Welcome, **austin**. [Change password](#) / [Log out](#)

[Home](#) > [Blog](#) > [Blogs](#)

Select blog to change Add blog +

Search

Action: 0 of 1 selected

<input type="checkbox"/>	Title	Created	Updated
<input type="checkbox"/>	First Blog Post	June 21, 2011, 9:42 p.m.	June 21, 2011, 9:42 p.m.

1 blog

Filter
By created
[Any date](#)
[Today](#)
[Past 7 days](#)
[This month](#)
[This year](#)

And this:

Change blog

History

Title:

Body:

Comments

Body	Author	Delete?
<p>That blog sucked</p> <input type="text" value="That blog sucked"/>	<input type="text" value="austin"/>	<input type="checkbox"/>
<p>Seconded ;lkj ;lkj lkjlkjlkjksd falskdjf alskdjf lasdkjf alsdkjf alkdsjf laksjdf lajds fikas jdfik</p> <p>Seconded ;lkj ;lkj lkjlkjlkjksd falskdjf alskdjf lasdkjf alsdkjf alkdsjf laksjdf lajds fikas idfkajds fikais dfkaj sdtkajdsfka isdkjf alkdsf jalskdjf laksdjf laksd ifakd ifaksd if</p>	<input type="text" value="not austin"/>	<input type="checkbox"/>