



Accelerating Information Technology

<http://aiti.mit.edu>

Ghana 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

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
```

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

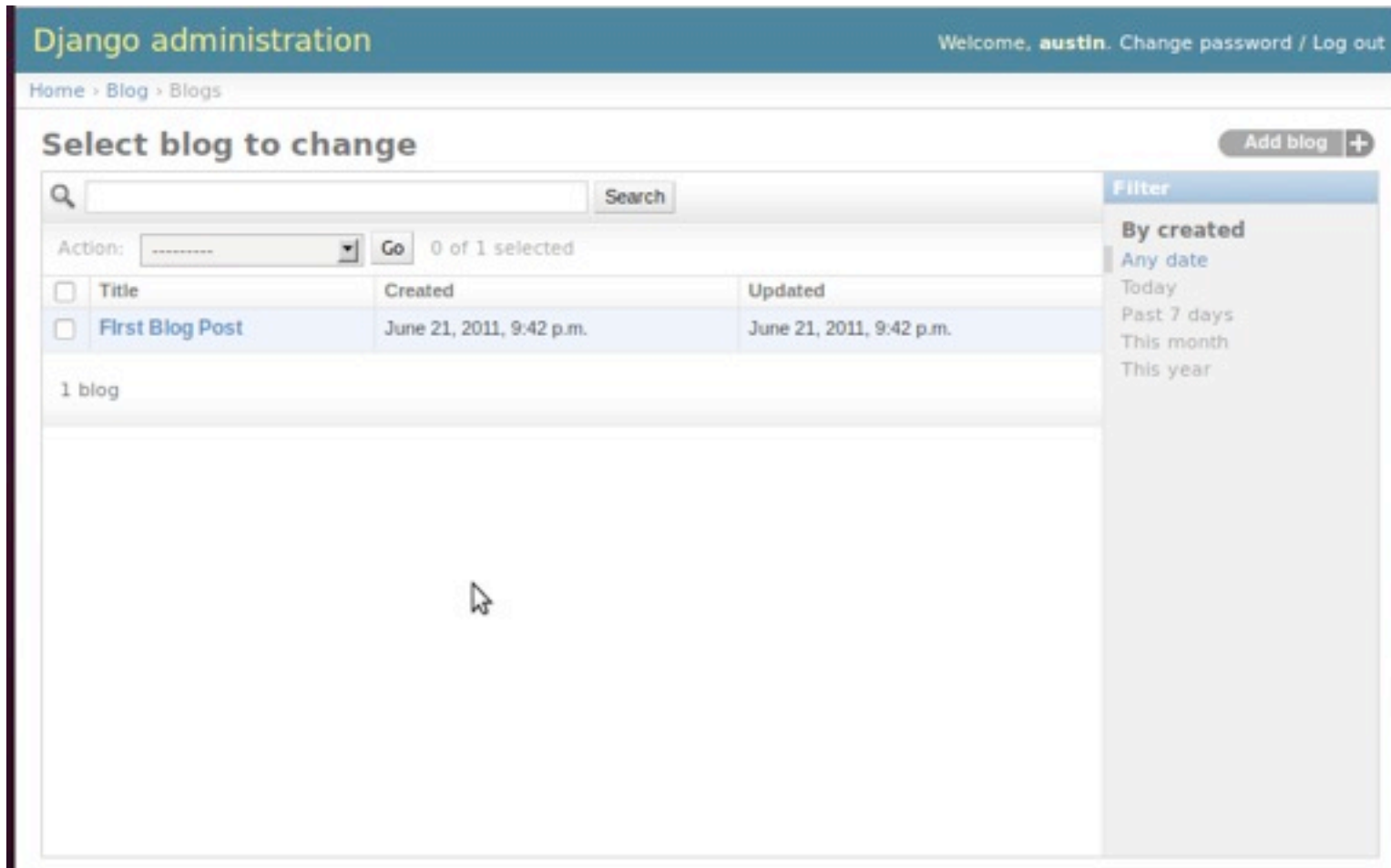
- On the admin pages, you may want to see all the Book objects that relate to one Author.
- Django Admin lets you put this all on one page with minimal effort

Inline Syntax

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

(You can use either TabularInline or StackedInline)

We want this:



The screenshot shows the Django administration interface for a 'Blog' application. The page title is 'Django administration' and the user is logged in as 'austin'. The breadcrumb trail is 'Home > Blog > Blogs'. The main heading is 'Select blog to change'. There is a search bar and a 'Search' button. Below the search bar is an 'Action:' dropdown menu and a 'Go' button. The table below shows one blog entry:

<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.

Below the table, it says '1 blog'. On the right side, there is a 'Filter' sidebar with the following options:

- By created
- Any date
- Today
- Past 7 days
- This month
- This year

At the bottom of the page, there is a mouse cursor pointing to the text '1 blog'.

And this:

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

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

Change blog History

Title:

Body:

Comments

Body	Author	Delete?
<input type="text" value="That blog sucked"/>	<input type="text" value="austin"/>	<input type="checkbox"/>
<input type="text" value="Seconded :lk :lk lkjlkjlkjfd fskdjf alskdjf lsdkjf alsdkjf alskdjf laksjdf lajsdf fkas jdfk"/>	<input type="text" value="not austin"/>	<input type="checkbox"/>