Tutorial Django Parte 4: Django admin site

Agora que criamos modelos para o site da LocalLibrary, usaremos o site do Django Admin para adicionar alguns dados de livros "reais". Primeiro, mostraremos como registrar os modelos no site de administração, depois mostraremos como fazer login e criar alguns dados. No final do artigo, mostraremos algumas maneiras de melhorar ainda mais a apresentação do site Admin.

Pré-requisitos:	Primeiro complete: Tutorial Django Parte 3: Usando modelos.
Objetivo:	Para entender os benefícios e limitações do site de administração do Django, use-o para criar alguns registros para nossos modelos.

Visão Geral

O aplicativo de administração do Django pode usar seus modelos para criar automaticamente uma área de site que você possa usar para criar, visualizar, atualizar e excluir registros. Isso pode poupar muito tempo durante o desenvolvimento, tornando muito fácil testar seus modelos e ter uma ideia de se você tem os dados corretos. O aplicativo administrativo também pode ser útil para gerenciar dados em produção, dependendo do tipo de site. O projeto Django o recomenda apenas para gerenciamento interno de dados (ou seja, apenas para uso por administradores ou pessoas internas à sua organização), pois a abordagem centrada no modelo não é necessariamente a melhor interface possível para todos os usuários e expõe muitos detalhes desnecessários sobre os modelos.

Toda a configuração necessária para incluir o aplicativo admin em seu site foi feita automaticamente quando você criou o esqueleto do projeto (para obter informações sobre as dependências reais necessárias, consulte a documentação do Django aqui). Como resultado, tudo o que você precisa fazer para adicionar seus modelos ao aplicativo administrativo é registrá-los. No final deste artigo, forneceremos uma breve demonstração de como você pode configurar

ainda mais a área de administração para exibir melhor nossos dados de modelo.

Depois de registrar os modelos, mostraremos como criar um novo "superusuário", acessar o site e criar alguns livros, autores, instâncias de livros e gêneros. Isso será útil para testar as visualizações e os modelos que começaremos a criar no próximo tutorial.

Registrando modelos

Primeiro, abra o admin.py no aplicativo de catálogo (/locallibrary/catalog/admin.py). Atualmente parece com isso - note que ele já importa `django.contrib.admin`:

```
from django.contrib import admin

# Register your models here.
```

Registre os modelos copiando o seguinte texto na parte inferior do arquivo. Este código simplesmente importa os modelos e, em seguida, chama `admin.site.register` para registrar cada um deles.

```
from catalog.models import Author, Genre, Book, BookInstance

admin.site.register(Book)
admin.site.register(Author)
admin.site.register(Genre)
admin.site.register(BookInstance)
```

> Nota: Se você aceitou o desafio de criar um modelo para representar a linguagem natural de um livro (consulte o artigo do tutorial de modelos), importe-o e registre-o também!

Essa é a maneira mais simples de registrar um modelo ou modelos no site. O site de administração é altamente personalizável e falaremos mais sobre as outras maneiras de registrar seus modelos mais abaixo.

Criando um super usuário

Para fazer login no site de administração, precisamos de uma conta de usuário com o status da equipe ativado. Para visualizar e criar registros, também precisamos que esse usuário tenha

permissões para gerenciar todos os nossos objetos. Você pode criar uma conta "superusuário" que tenha acesso total ao site e todas as permissões necessárias usando **manage.py**.

Chame o seguinte comando, no mesmo diretório que manage.py, para criar o superusuário. Você será solicitado a digitar um nome de usuário, endereço de e-mail e senha forte.

```
python3 manage.py createsuperuser
```

Quando esse comando for concluído, um novo superusuário será adicionado ao banco de dados. Agora reinicie o servidor de desenvolvimento para que possamos testar o login:

```
python3 manage.py runserver
```

Fazendo o login e usando o site

Para fazer login no site, abra o URL */admin* (e.g. http://127.0.0.1:8000/admin) e insira suas novas credenciais de usuário e senha de superusuário (você será redirecionado para a página de login e, em seguida, de volta para o URL */admin* depois de inserir seus detalhes).

Esta parte do site exibe todos os nossos modelos, agrupados por aplicativo instalado. Você pode clicar no nome de um modelo para ir a uma tela que lista todos os seus registros associados e clicar nos registros para editá-los. Você também pode clicar diretamente no link Adicionar ao lado de cada modelo para começar a criar um registro desse tipo.

Clique no link Adicionar à direita de Books para criar um novo livro (isso exibirá um diálogo muito parecido com o abaixo). Observe como os títulos de cada campo, o tipo de widget usado e o `help_text` (se houver) correspondem aos valores especificados no modelo.

Digite valores para os campos. Você pode criar novos autores ou gêneros pressionando o botão + ao lado dos respectivos campos (ou selecione os valores existentes nas listas, se você já os criou). Quando estiver pronto, você pode pressionar **SALVAR, Salvar e adicionar** outro ou S**alvar e continuar editando** para salvar o registro.

.

> Observação: neste ponto, gostaríamos que você passasse algum tempo adicionando alguns livros, autores e gêneros (por exemplo, Fantasia) à sua inscrição. Certifique-se de que cada autor e gênero inclua alguns livros diferentes (isso tornará suas visualizações de lista e
>
> detalhes mais interessantes quando forem implementadas posteriormente na série de artigos).

Quando terminar de adicionar livros, clique no link **Home** no marcador superior para ser levado de volta à página principal do administrador. Então clique no link **Books** para exibir a lista atual de livros (ou em um dos outros links para ver outras listas de modelos). Agora que você adicionou alguns livros, a lista pode ser semelhante à captura de tela abaixo. O título de cada livro é exibido; este é o valor retornado no modelo do livro pelo método `__str__()` que especificamos no último artigo.

.

Nessa lista, você pode excluir livros marcando a caixa de seleção ao lado do livro que não deseja, selecionando a ação excluir ... na lista suspensa Ação e pressionando o botão Ir. Você também pode adicionar novos livros pressionando o botão **ADD BOOK**.

Você pode editar um livro selecionando seu nome no link. A página de edição de um livro, mostrada abaixo, é quase idêntica à página "Adicionar". As principais diferenças são o título da página (Change book) e a adição de botões **Delete**, **HISTORY** e **VIEW ON SITE** (este último botão aparece porque definimos o método `get_absolute_url()` em nosso modelo).

.

Agora navegue de volta para o **Home** page (usando o link Home, a trilha de navegação) e, em seguida, **Author** e listas de **Genre** — você já deve ter criado a partir de quando adicionou os novos livros, mas fique à vontade para adicionar um pouco mais.

O que você não terá é qualquer instância do livro, porque elas não são criadas a partir de livros (embora você possa criar `Book` a partir de `BookInstance` — esta é a natureza da `ForeignKey` field). Navegue de volta para a *Página inicial* e pressione o botão *Adicionar* associado para exibir a tela *Adicionar instância* do livro abaixo. Observe o ID grande e globalmente exclusivo, que pode ser usado para identificar separadamente uma única cópia de um livro na biblioteca.

.

Crie vários desses registros para cada um de seus livros. Defina o status como *Disponível* para pelo menos alguns registros e *Em empréstimo* para outros. Se o status **não** for *Disponível*, defina também uma *data de vencimento* futura.

É isso aí! Agora você aprendeu como configurar e usar o site de administração. Você também criou registros para `Book`, `BookInstance`, `Genre`, e `Author` que poderemos usar assim que criarmos nossas próprias visualizações e modelos.

Configuração Avançada

O Django faz um bom trabalho ao criar um site de administração básico usando as informações dos modelos registrados:

- Cada modelo possui uma lista de registros individuais, identificados pela string criada com o método `__str__()` do modelo, e vinculado a visualizações de views/forms para edição. Por padrão, essa exibição tem um menu de ação na parte superior que você pode usar para executar operações de exclusão em massa nos registros.
- Os formulários de registro de detalhes do modelo para edição e adição de registros contêm todos os campos do modelo, dispostos verticalmente em sua ordem de declaração.

Você pode personalizar ainda mais a interface para torná-la ainda mais fácil de usar. Algumas das coisas que você pode fazer são:

- List views:
  - Adicionar adicional fields/information exibido para cada registro.
  - Adicione filtros para selecionar quais registros são listados, com base na data ou em algum outro valor de seleção (e.g. Book loan status).
  - Adicione opções adicionais ao menu de ações nas exibições de lista e escolha onde esse menu é exibido no formulário.
- Detail views
  - Escolha quais campos exibir (ou excluir), junto com sua ordem, agrupamento, se eles são editáveis, o widget usado, a orientação etc.
  - Adicione campos relacionados a um registro para permitir a edição imediata (por exemplo, adicione a capacidade de adicionar e editar registros de livros enquanto cria

o registro de autor).

In this section we're going to look at a few changes that will improve the interface for our *LocalLibrary*, including adding more information to `Book` and `Author` model lists, and improving the layout of their edit views. We won't change the `Language` and `Genre` model presentation because they only have one field each, so there is no real benefit in doing so!

You can find a complete reference of all the admin site customisation choices in The Django Admin site (Django Docs).

Registrando uma classe ModelAdmin

Para alterar como um modelo é exibido na interface de administração, você define uma classe ModelAdmin (que descreve o layout) e registra-o no modelo.

Vamos começar com o `Author` model. Abra **admin.py** no aplicativo de catálogo (/locallibrary/catalog/admin.py). Comente o seu registro original (prefixo com um #) para o `Author` model:

```
# admin.site.register(Author)
```

Agora adicione um novo `AuthorAdmin` e registre como mostrado abaixo.

```
# Define the admin class
class AuthorAdmin(admin.ModelAdmin):
    pass

# Register the admin class with the associated model
admin.site.register(Author, AuthorAdmin)
```

Agora vamos adicionar as classes `ModelAdmin` para `Book`, e `BookInstance`. Precisamos novamente comentar os registros originais:

```
# admin.site.register(Book)
# admin.site.register(BookInstance)
```

Agora, para criar e registrar os novos modelos; para o propósito desta demonstração, vamos usar o `@register` decorador para registrar os modelos (isso faz exatamente a mesma coisa que

O `@register` decorador para registrar os modelos (isso faz exatamente a mesma coisa que `admin.site.register()` sintaxe):

```
# Register the Admin classes for Book using the decorator
@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    pass

# Register the Admin classes for BookInstance using the decorator
@admin.register(BookInstance)
class BookInstanceAdmin(admin.ModelAdmin):
    pass
```

Atualmente todas as nossas classes administrativas estão vazias (veja `pass`) então o comportamento do administrador não será alterado! Agora podemos estendê-los para definir nosso comportamento administrativo específico do modelo.

Configure list views

A LocalLibrary atualmente lista todos os autores usando o nome do objeto gerado a partir do método `__str__()` do modelo. Isso é bom quando você tem apenas alguns autores, mas quando você tem muitos, você pode acabar tendo duplicatas. Para diferenciá-los, ou apenas porque você quer mostrar informações mais interessantes sobre cada autor, você pode usar list_display para adicionar campos adicionais à vista.

Substitua seu `AuthorAdmin` class com o código abaixo. Os nomes de campo a serem exibidos na lista são declarados em uma *tupla* na ordem requerida, conforme mostrado (esses são os mesmos nomes especificados em seu modelo original).

```
class AuthorAdmin(admin.ModelAdmin):
    list_display = ('last_name', 'first_name', 'date_of_birth', 'date_of
```

Agora navegue até a lista de autores em seu site. Os campos acima devem agora ser exibidos, assim:

.

Para o nosso `Book` model nós vamos adicionalmente exibir o `author` e `genre`. O `author` é uma variável `ForeignKey` (um-para-um) relacionamento, e assim será representado pelo valor

`__str__()` para o registro associado. Substitua o `BookAdmin` class com a versão abaixo.

```
class BookAdmin(admin.ModelAdmin):
    list_display = ('title', 'author', 'display_genre')
```

Infelizmente não podemos especificar diretamente a variável `genre` na `list_display` porque é um `ManyToManyField` (O Django evita isso porque há um grande "custo" de acesso ao banco de dados ao fazer isso). Em vez disso, vamos definir uma função `display_genre` para obter as informações como uma string (esta é a função que chamamos acima; vamos defini-lo abaixo).

> Nota: Obtendo o `genre` pode não ser uma boa ideia aqui, por causa do "custo" da operação do banco de dados. Estamos mostrando como as funções de chamada em seus modelos podem ser muito úteis por outros motivos - por exemplo, para adicionar um link *Apagar* ao lado de cada item da lista.

Adicione o seguinte código ao seu `Book` model (**models.py**). Isso cria uma string a partir dos três primeiros valores da variavel `genre` (se existirem) e cria um `short_description` que pode ser usado no site administrativo para esse método.

```
    def display_genre(self):
        """Create a string for the Genre. This is required to display ge
        return ', '.join(genre.name for genre in self.genre.all()[:3])

    display_genre.short_description = 'Genre'
```

Depois de salvar o modelo e o administrador atualizado, abra o site e vá para a página da lista Livros; você deve ver uma lista de livros como a abaixo:

.

O `Genre` model (e a `Language` model, se você definiu um) ambos têm um único campo, portanto, não faz sentido criar um modelo adicional para exibir campos adicionais.

> Nota: Vale a pena atualizar o `BookInstance` model list para mostrar pelo menos o status e a data de retorno esperada. Nós adicionamos isso como um desafio no final deste artigo!

Adicionando list filters

Uma vez que você tenha muitos itens em uma lista, pode ser útil filtrar quais itens são exibidos. Isso é feito listando os campos no atributo `list_filter`. Substitua sua

atual `BookInstanceAdmin` class com o fragmento de código abaixo.

```
class BookInstanceAdmin(admin.ModelAdmin):
    list_filter = ('status', 'due_back')
```

A visualização de lista agora incluirá uma caixa de filtro à direita. Observe como você pode escolher datas e status para filtrar os valores:

Organizando o layout da detail view

Por padrão, as exibições detalhadas exibem todos os campos verticalmente, em sua ordem de declaração no modelo. Você pode alterar a ordem da declaração, quais campos são exibidos (ou excluídos), se as seções são usadas para organizar as informações, se os campos são exibidos horizontalmente ou verticalmente e até mesmo quais widgets de edição são usados nos formulários admin.

> Nota: Os modelos *LocalLibrary* são relativamente simples, portanto não é necessário alterar o layout; No entanto, faremos algumas alterações, só para mostrar como.

Controlando quais campos são exibidos

Atualize seu `AuthorAdmin` class para adicionar a linha `fields`, como mostrado abaixo (em negrito):

```
class AuthorAdmin(admin.ModelAdmin):
    list_display = ('last_name', 'first_name', 'date_of_birth', 'date_of
    fields = ['first_name', 'last_name', ('date_of_birth', 'date_of_deat
```

O atributo `fields` lista apenas os campos que devem ser exibidos no formulário, em ordem. Os campos são exibidos verticalmente por padrão, mas serão exibidos horizontalmente se você agrupá-los posteriormente em uma tupla (conforme mostrado nos campos "data" acima).

No seu site, acesse a visualização de detalhes do autor. Agora, ele deve aparecer como mostrado abaixo:

.

> Nota: você também pode usar o atributo `exclude` para declarar uma lista de atributos a serem excluídos do formulário (todos os outros atributos no modelo serão exibidos).

Seccionando a detail view

Você pode adicionar "seções" para agrupar informações de modelo relacionadas dentro do formulário detalhado, usando o atributo fieldsets.

Na `BookInstance` model temos informações relacionadas ao que o livro é (i.e. `name`, `imprint`, e `id`) e quando estará disponível (`status`, `due_back`). Podemos adicionar estes em diferentes seções, adicionando o texto em negrito para o nosso `BookInstanceAdmin` class.

```
@admin.register(BookInstance)
class BookInstanceAdmin(admin.ModelAdmin):
    list_filter = ('status', 'due_back')

    fieldsets = (
        (None, {
            'fields': ('book', 'imprint', 'id')
        }),
        ('Availability', {
            'fields': ('status', 'due_back')
        }),
    )
```

Cada seção tem seu próprio título (ou `None`,se você não quiser um título) e uma tupla associada de campos em um dicionário - o formato é complicado de descrever, mas bastante fácil de entender se você olhar o fragmento de código imediatamente acima.

Agora, navegue até uma visualização de instância do livro em seu website; o formulário deve aparecer como mostrado abaixo:

.

Edição inline de registros associados

Às vezes, pode fazer sentido adicionar registros associados ao mesmo tempo. Por exemplo, pode fazer sentido ter as informações do livro e as informações sobre as cópias específicas que você tem na mesma página de detalhes.

Você pode fazer isso declarando inlines, do tipo TabularInline (horizonal layout) or StackedInline (layout vertical, assim como o layout do modelo padrão). Você pode adicionar ao `BookInstance` informações inline para o nosso `Book` detalhe, adicionando as linhas abaixo em negrito perto do seu `BookAdmin`:

```
class BooksInstanceInline(admin.TabularInline):
    model = BookInstance

@admin.register(Book)
class BookAdmin(admin.ModelAdmin):
    list_display = ('title', 'author', 'display_genre')
    inlines = [BooksInstanceInline]
```

Agora navegue até uma view pala um `Book` no seu site - na parte inferior, você verá as instâncias do livro relacionadas a este livro (imediatamente abaixo dos campos de gênero do livro):

.

Nesse caso, tudo o que fizemos foi declarar nossa classe inline tabular, que apenas adiciona todos os campos do modelo embutido. Você pode especificar todos os tipos de informações adicionais para o layout, incluindo os campos a serem exibidos, sua ordem, se eles são somente leitura ou não, etc. (veja TabularInline para maiores informações).

> Nota: Existem alguns limites dolorosos nesta funcionalidade! Na captura de tela acima, temos três instâncias de livros existentes, seguidas de três espaços reservados para novas instâncias de livros (que são muito semelhantes!). Seria melhor não ter instâncias do livro reserva por padrão e apenas adicioná-las com o link **Add another Book instance** , ou poder listar apenas `BookInstance` s como links não legíveis daqui. A primeira opção pode ser feita configurando atributo `extra` para 0 no `BooksInstanceInline` model, tente você mesmo.

Desafie-se

Aprendemos muito nesta seção, então agora é hora de você tentar algumas coisas.

1. Para a *listview* `BookInstance`, adicione o código para exibir o livro, o status, a data de devolução e o id (em vez do texto padrão `__str__()`).
2. Adicione uma listagem *inline* de itens `Book` para a lista detalhada de `Author` usando a mesma abordagem que fizemos para `Book`/`BookInstance`.

Resumo

É isso aí! Agora você aprendeu como configurar o site de administração na sua forma simples e aprimorada, como criar um superusuário, como navegar no site de administração e visualizar, excluir e atualizar registros. Ao longo do caminho você criou um monte de Livros, Instâncias de livros, Gêneros e Autores que poderemos listar e exibir assim que criarmos nossas próprias *views* e *templates*.

Leitura adicional

- Escrevendo seu primeiro app Django, parte 2: Apresentando o Django Admin (Django docs)
- O Django Admin site (Django Docs)

Neste módulo

- Tutorial Django Parte 10: Testando uma aplicação web Django
- Tutorial Django Parte 11: Implantando Django em produção
- Segurança de aplicações Django
- DIY Django mini blog

Last modified: 21 de ago. de 2020, by MDN contributors

Change your language

Português (do Brasil) Change language