به مستندات کتابخانه خوش آمدید!

فهرست:

پکیج حساب کاربری

زیر ماژول ها

account.admin ماژول

این فایل شامل مدیریت مدل ها در پنل مدیریت جنگو می باشد.

class account.admin.**ProfileAdmin**(*model*, *admin_site*)

پایه ها: `ModelAdmin`

لیستی از فیلدهایی که در لیست نمایش در پنل مدیریت نمایش داده می شوند.

list_display = *('user',)*

property **media**

account.apps ماژول

ماژول پیکربندی برای برنامه "حساب کاربری".

این ماژول شامل پیکربندی برنامه «حساب کاربری»، از جمله نام برنامه و تنظیمات فیلد خودکار پیش‌فرض است.

class account.apps.**AccountConfig**(*app_name*, *app_module*)

پایه ها: `AppConfig`

کلاس پیکربندی برای برنامه «حساب کاربری».

'default_auto_field = *'django.db.models.BigAutoField*

'name = *'account*

account.forms ماژول

این فایل حاوی تعریف فرم هایی برای مدیریت ورودی داده های کاربر است.

class account.forms.**EditProfileFrom**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

پایه ها: `ModelForm`

فرم مشخصات کاربر

class **Meta**

پایه ها: `object`

fields = *('profileImage',)*

model

نام جانشین `Profile`

widgets = *{'profileImage': <django.forms.widgets.FileInput object>}*

base_fields = *{'profileImage': <django.forms.fields.ImageField object>}*

declared_fields = *{'profileImage': <django.forms.fields.ImageField object>}*

property **media**

.Return all media required to render the widgets on this form

class account.forms.**EditUserForm**(**args*, ***kwargs*)

پایه ها: `UserChangeForm`

فرم اطلاعات کاربر

class **Meta**

پایه ها: `Meta`

fields = *('first_name', 'last_name', 'email')*

widgets = *{'email': <django.forms.widgets.TextInput object>, 'first_name': <django.forms.widgets.TextInput object>, 'last_name': <django.forms.widgets.TextInput object>}*

base_fields = *{'email': <django.forms.fields.EmailField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>}*

{} = **declared_fields**

property **media**

.Return all media required to render the widgets on this form

password = *None*

class account.forms.**LoginUserForm**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, field_order=None, use_required_attribute=None, renderer=None*)

پایه ها: `Form`

فرم ورود کاربر

base_fields = *{'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**

.Return all media required to render the widgets on this form

class account.forms.**RegisterUserForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

پایه ها: `Form`

فرم ثبت نام کاربران جدید

base_fields = *{'confirm_password': <django.forms.fields.CharField object>, 'email': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

declared_fields = *{'confirm_password': <django.forms.fields.CharField object>, 'email': <django.forms.fields.CharField object>, 'first_name': <django.forms.fields.CharField object>, 'last_name': <django.forms.fields.CharField object>, 'password': <django.forms.fields.CharField object>, 'username': <django.forms.fields.CharField object>}*

property **media**

.Return all media required to render the widgets on this form

ماژول account.models

این فایل شامل تعریف مدل های پایگاه داده استفاده شده در برنامه ما می باشد.

class account.models.**Profile**(**args*, ***kwargs*)

پایه ها: `Model`

این فیلد با مدل کاربری خود جنگو رابطه یک به یک دارد و اطلاعات مربوط به پروفایل را ذخیره می کند.

exception **DoesNotExist**

پایه ها: `ObjectDoesNotExist`

exception **MultipleObjectsReturned**

پایه ها: `MultipleObjectsReturned`

id

A wrapper for a deferred-loading field. When the value is read from this .object the first time, the query is executed

<objects = <django.db.models.manager.Manager object

profileImage

نام کاربری مرتبط با نمایه را برگردانید تا به صورت رشته نمایش داده شود.

user

عکس پروفایل کاربر را ذخیره می کند. مسیر ذخیره سازی و امکان خالی یا خالی بودن فیلد تصویر مشخص شده است.

user_id

account.urls ماژول

این فایل حاوی تنظیمات مسیریابی صفحه برای نماهای مربوط به حساب کاربری است.

account.urls.urlpatterns ***= [<URLPattern 'sign-up/' [name='register']>, <URLPattern 'login/' [name='login']>, <URLPattern 'logout/' [name='logout']>, <URLPattern 'profile/' [name='profile']>, <URLPattern 'profile/delete-photo/' [name='delete_photo']>]***

توضیحات الگوهای صفحه:

- 'sign-up/': الگوی صفحه برای ثبت نام کاربر.
- 'login/': الگوی صفحه برای ورود کاربر.
- 'logout/': الگوی صفحه برای خروج کاربر.
- 'profile/': الگوی صفحه برای نمای نمایه کاربر.
- 'profile/delete-photo/': الگوی صفحه برای حذف عکس پروفایل کاربر.

ماژول account.views

این فایل شامل نماها و منطق برای احراز هویت کاربر و مدیریت پروفایل است.

account.views.delete_photo(*request*)

مشاهده برای ویرایش مشخصات کاربر.

> فرمی را برای کاربر نمایش می دهد تا اطلاعات نمایه خود را ویرایش کند. ارسال فرم ویرایش نمایه را انجام می دهد و در صورت معتبر بودن جزئیات نمایه را به روز می کند.

account.views.loginUser(*request*)

مشاهده برای ورود کاربر

> فرآیند ورود کاربر، از جمله اعتبار سنجی فرم و احراز هویت را مدیریت می کند. کاربر را پس از ورود موفقیت آمیز به صفحه مناسب هدایت می کند یا یک پیام خطا نمایش می دهد.

account.views.logoutUser(*request*)

مشاهده برای خروج کاربر

> کاربر احراز هویت شده را از سیستم خارج می کند، یک پیام تشکر نشان می دهد و به صفحه تغییر مسیر ورود به سیستم هدایت می شود.

account.views.profileEdit(*request*)

عملکرد ویرایش نمایه را برای یک کاربر وارد شده مدیریت می کند.

> **ارگمان ها:**
>
> درخواست: درخواست HTTP.
>
> **خروجی:**
>
> پاسخ HTML ارائه شده برای صفحه ویرایش نمایه.

account.views.profileUser(*request*)

مشاهده صفحه نمایه کاربر

> اطلاعات نمایه کاربر از جمله گزینه ویرایش نمایه را نمایش می دهد. ارسال فرم ویرایش نمایه را انجام می دهد و در صورت معتبر بودن جزئیات نمایه را به روز می کند.

account.views.register(*request*)

مشاهده برای ثبت نام کاربر

> فرآیند ثبت نام کاربر، از جمله اعتبار سنجی فرم و ایجاد حساب را مدیریت می کند. پس از ثبت نام موفقیت آمیز به صفحه مناسب هدایت می شود یا یک پیام خطا نمایش می دهد.

پکیج کتاب

زیر ماژول ها

book.admin ماژول

این فایل شامل پیکربندی برای رابط مدیریت برای مدیریت مدل‌های کتاب، ژانر، نویسنده و نمونه کتاب است.

class book.admin.**AuthorAdmin**(*model*, *admin_site*)

پایه ها: `ModelAdmin`

کلاس مدیریت برای مدیریت مدل نویسنده.

> این کلاس گزینه های نمایش، فیلدهای جستجو و فیلترها را برای مدل نویسنده در رابط مدیریت تعریف می کند.

list_display = *('id', 'last_name', 'first_name')*

list_filter = *('last_name', 'first_name')*

property **media**

search_fields = *('last_name', 'first_name')*

class book.admin.**BookAdmin**(*model*, *admin_site*)

پایه ها: `ModelAdmin`

کلاس مدیریت برای مدیریت مدل کتاب.

این کلاس گزینه های نمایش، فیلدهای جستجو و فیلترها را برای مدل کتاب در رابط مدیریت تعریف می کند.

همچنین یک روش سفارشی برای نمایش ژانرهای مرتبط با هر کتاب تعریف می کند.

class **Meta**

پایه ها: `object`

ordering = *('genre',)*

display_genre(*obj*)

روش سفارشی برای نمایش ژانرها برای هر کتاب.

list_display = *('id', 'name', 'display_genre', 'author', 'status')*

list_filter = *('author', 'genre')*

property **media**

search_fields = *('name', 'author', 'genre')*

class book.admin.**BookInstanceAdmin**(*model*, *admin_site*)

پایه ها: `ModelAdmin`

کلاس مدیریت برای مدیریت مدل نمونه کتاب.

> این کلاس گزینه های نمایش، فیلدهای جستجو و فیلترها را برای مدل نمونه کتاب در رابط مدیریت تعریف می کند.

exclude = *('id',)*

list_display = *('book', 'borrower', 'due_back', 'status')*

list_filter = *('due_back',)*

property **media**

search_fields = *('book',)*

class book.admin.**GenreAdmin**(*model*, *admin_site*)

پایه ها: `ModelAdmin`

کلاس مدیریت برای مدیریت مدل ژانر.

> این کلاس گزینه های نمایش، فیلدهای جستجو و فیلترها را برای مدل ژانر در رابط مدیریت تعریف می کند.

```
list_display = ('id', 'title')

list_filter = ('title',)

property media

search_fields = ('title',)
```

book.apps ماژول

ماژول پیکربندی برای برنامه "کتاب".

این ماژول شامل پیکربندی برنامه «کتاب»، از جمله نام برنامه و تنظیمات فیلد خودکار پیش‌فرض است.

```
class book.apps.BookConfig(app_name, app_module)
```

پایه ها: `AppConfig`

کلاس پیکربندی برای برنامه "کتاب".

```
'default_auto_field = 'django.db.models.BigAutoField

'name = 'book
```

book.forms ماژول

این فایل حاوی تعریف فرم هایی برای مدیریت ورودی داده های کتاب است.

```
class book.forms.AddBookInstanceForm(data=None, files=None,
auto_id='id_%s', prefix=None, initial=None, error_class=<class
'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False,
instance=None, use_required_attribute=None, renderer=None)
```

پایه ها: `ModelForm`

فرم برای افزودن یک نمونه کتاب جدید.

> فیلدی برای تعیین تاریخ سررسید برای بازگرداندن نمونه کتاب فراهم می کند.

***class* Meta**

پایه ها: `object`

fields = *('due_back',)*

model

نام جانشین `BookInstance`

widgets = *{'due_back': <django.forms.widgets.DateInput object>}*

base_fields = *{'due_back': <django.forms.fields.DateField object>}*

{} = **declared_fields**

***property* media**

.Return all media required to render the widgets on this form

class book.forms.**EditBookForm**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None, renderer=None*)

پایه ها: `ModelForm`

فرم ویرایش کتاب موجود

> زمینه هایی را برای ویرایش نام، خلاصه، ژانر، نویسنده و تصویر کتاب فراهم می کند.

***class* Meta**

پایه ها: `object`

fields = *{'author', 'bookImage', 'genre', 'name', 'summary'}*

model

نام جانشین `Book`

widgets = {'bookImage': <django.forms.widgets.FileInput object>, 'name': <django.forms.widgets.TextInput object>, 'summary': <django.forms.widgets.Textarea object>}

base_fields = {'author': <django.forms.models.ModelChoiceField object>, 'bookImage': <django.forms.fields.ImageField object>, 'genre': <django.forms.models.ModelMultipleChoiceField object>, 'name': <django.forms.fields.CharField object>, 'summary': <django.forms.fields.CharField object>}

{} = **declared_fields**

property **media**

.Return all media required to render the widgets on this form

class book.forms.**InsertBookForm**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None, renderer=None*)

پایه ها: `ModelForm`

فرم درج کتاب جدید.

> زمینه هایی را برای افزودن نام، خلاصه، ژانر، نویسنده و تصویر کتاب جدید ارائه می دهد.

class **Meta**

پایه ها: `object`

fields = {'author', 'bookImage', 'genre', 'name', 'summary'}

model

نام جانشین `Book`

widgets = *{'bookImage': <django.forms.widgets.FileInput object>, 'name': <django.forms.widgets.TextInput object>, 'summary': <django.forms.widgets.Textarea object>}*

base_fields = *{'author': <django.forms.models.ModelChoiceField object>, 'bookImage': <django.forms.fields.ImageField object>, 'genre': <django.forms.models.ModelMultipleChoiceField object>, 'name': <django.forms.fields.CharField object>, 'summary': <django.forms.fields.CharField object>}*

{} = **declared_fields**

property **media**

.Return all media required to render the widgets on this form

class book.forms.**SearchBoxForm**(*data=None, files=None, auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None, empty_permitted=False, field_order=None, use_required_attribute=None, renderer=None*)

پایه ها: `Form`

فرم برای کادر جستجو

> به کاربران امکان می دهد کلمات کلیدی جستجو را برای جستجوی کتاب وارد کنند.

base_fields = *{'search': <django.forms.fields.CharField object>}*

declared_fields = *{'search': <django.forms.fields.CharField object>}*

property **media**

.Return all media required to render the widgets on this form

book.models ماژول

این ماژول مدل های پایگاه داده را برای برنامه مدیریت کتاب تعریف می کند.

این شامل مدل‌های ژانر، نویسنده و کتاب است که به ترتیب ژانرها، نویسندگان، کتاب‌ها و نمونه کتاب را نشان می‌دهند.

class book.models.**Author**(**args*, ***kwargs*)

پایه ها: `Model`

مدلی که نماینده یک نویسنده است.

> **زمینه های:**
>
> first_name (CharField): نام کوچک نویسنده. last_name (CharField): نام خانوادگی نویسنده.

exception **DoesNotExist**

پایه ها: `ObjectDoesNotExist`

exception **MultipleObjectsReturned**

پایه ها: `MultipleObjectsReturned`

book_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

first_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

last_name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

class book.models.Book(*args, **kwargs)

پایه ها: `Model`

مدلی که یک کتاب را نشان می دهد.

زمینه های:

LOAN_STATUS (دوگانه): گزینه‌هایی برای وضعیت امانت کتاب. name (CharField): نام کتاب. summary (TextField): شرح مختصری از کتاب. genre (ManyToManyField): ژانرهای مرتبط با کتاب. author (ForeignKey): نویسنده کتاب. status (CharField): وضعیت امانت کتاب. bookImage (ImageField): تصویری از کتاب. quantity (IntegerField): مقدار کتاب.

exception DoesNotExist

پایه ها: `ObjectDoesNotExist`

LOAN_STATUS = (('o', 'On Loan'), ('a', 'Available'))

exception MultipleObjectsReturned

پایه ها: `MultipleObjectsReturned`

author

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
:class Child(Model)
    parent = ForeignKey(Parent, related_name='children')
```

.`Child.parent` is a `ForwardManyToOneDescriptor` instance

author_id

bookImage

Just like the FileDescriptor, but for ImageFields. The only difference is .assigning the width/height to the width_field/height_field, if appropriate

bookinstance_set

Accessor to the related objects manager on the reverse side of a many-to-.one relation

:In the example

```
:class Child(Model)
    parent = ForeignKey(Parent, related_name='children')
```

.`Parent.children` is a `ReverseManyToOneDescriptor` instance

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined .below

genre

Accessor to the related objects manager on the forward and reverse sides .of a many-to-many relation

:In the example

```
:class Pizza(Model)
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` .instances

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_status_display(*, *field=<django.db.models.fields.CharField: status>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

name

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = *<django.db.models.manager.Manager object*

quantity

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

status

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

summary

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class book.models.**BookInstance**(**args*, ***kwargs*)

پایه ها: `Model`

مدلی که نمونه ای از کتاب موجود برای امانت را نشان می دهد.

زمینه های:

LOAN_STATUS (دوگانه): گزینه هایی برای وضعیت امانت نمونه کتاب.
TODAY (datetime): تاریخ فعلی. DUE_DATE (تاریخ): تاریخ سررسید برای بازگرداندن نمونه کتاب. id (UUIDField): شناسه منحصر به فرد برای نمونه

کتاب. (ForeignKey) book: کتاب مرتبط برای این نمونه. due_back
(DateField): تاریخ سررسید برای بازگرداندن نمونه کتاب. borrower
(ForeignKey): کاربری که نمونه کتاب را قرض می گیرد. status (CharField):
وضعیت امانت نمونه کتاب.

DUE_DATE = *datetime.datetime(2023, 8, 18, 11, 53, 3, 255092)*

exception **DoesNotExist**

پایه ها: `ObjectDoesNotExist`

LOAN_STATUS = *(('o', 'On Loan'), ('a', 'Available'))*

exception **MultipleObjectsReturned**

پایه ها: `MultipleObjectsReturned`

TODAY = *datetime.datetime(2023, 8, 11, 11, 53, 3, 255092)*

book

Accessor to the related object on the forward side of a many-to-one or
.one-to-one (via ForwardOneToOneDescriptor subclass) relation

:In the example

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

book_id

borrower

Accessor to the related object on the forward side of a many-to-one or
.one-to-one (via ForwardOneToOneDescriptor subclass) relation

:In the example

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

borrower_id

()default_user

due_back

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_due_back(*, *field=<django.db.models.fields.DateField: due_back>*, *is_next=True*, ***kwargs*)

get_previous_by_due_back(*, *field=<django.db.models.fields.DateField: due_back>*, *is_next=False*, ***kwargs*)

get_status_display(*, *field=<django.db.models.fields.CharField: status>*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

<objects = <django.db.models.manager.Manager object

status

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

***class* book.models.Genre(**args*, ***kwargs*)**

پایه ها: `Model`

مدلی که یک ژانر کتاب را نشان می دهد.

زمینه های:

title (CharField): عنوان ژانر.

exception **DoesNotExist**

پایه ها: `ObjectDoesNotExist`

exception **MultipleObjectsReturned**

پایه ها: `MultipleObjectsReturned`

book_set

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

<objects = <django.db.models.manager.Manager object

title

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

book.urls ماژول

این ماژول الگوهای صفحه را برای برنامه مدیریت کتاب تعریف می کند.

این شامل الگوهای صفحه برای نماهای مختلف مانند خانه، افزودن کتاب، مشاهده جزئیات کتاب، ویرایش جزئیات کتاب، حذف یک کتاب،

نمایش همه ژانرها، نمایش همه نویسندگان، نمایش همه نمونه های کتاب، و یک صفحه درباره ما است.

book.urls.urlpatterns = *[<URLPattern '' [name='home']>, <URLPattern 'insert-book/' [name='addBook']>, <URLPattern 'detail/<int:pk>/' [name='detail']>, <URLPattern 'detail/<int:pk>/edit' [name='editBook']>, <URLPattern 'detail/<int:pk>/delete' [name='deleteBook']>, <URLPattern 'all-genres/' [name='allGenres']>, <URLPattern 'all-authors/' [name='allAuthors']>, <URLPattern 'all-instances/' [name='allInstances']>, <URLPattern 'about/' [name='about']>]*

توضیحات الگوهای صفحه:

- ' ' (home): صفحه اصلی پیش فرض برنامه مدیریت کتاب.
- 'insert-book/': مشاهده برای افزودن یک کتاب جدید به کتابخانه.
- 'detail/<int:pk>/': نمایش برای نمایش جزئیات یک کتاب خاص با استفاده از کلید اصلی آن.
- 'detail/<int:pk>/edit': مشاهده برای ویرایش جزئیات یک کتاب خاص با استفاده از کلید اصلی آن.
- 'detail/<int:pk>/delete': نمایش برای حذف یک کتاب خاص با استفاده از کلید اصلی آن.
- 'all-genres/': نمایش برای نمایش همه ژانرهای کتاب موجود.
- 'all-authors/': نمایش برای نمایش همه نویسندگان مرتبط با کتاب ها.
- 'all-instances/': نمایش برای نمایش همه نمونه های کتاب در کتابخانه.
- 'about/': صفحه درباره ارائه اطلاعات در مورد برنامه.

book.views ماژول

این ماژول شامل نماهایی برای عملکردهای مرتبط با کتاب از جمله نمایش صفحه اصلی، صفحه جزئیات برای نمونه های کتاب، افزودن،

ویرایش، و حذف کتاب ها، نمایش ژانرها و نویسندگان، و نمایش نمونه های کتاب کاربر است.

book.views.about(*request*)

صفحه "درباره" را ارائه می دهد و اطلاعاتی در مورد برنامه ارائه می دهد.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML برای صفحه «درباره» ارائه شده است.

`book.views.addBook(request)`

اضافه کردن یک کتاب جدید به پایگاه داده را کنترل می کند.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML ارائه شده برای صفحه "افزودن کتاب".

`book.views.deleteBook(request, pk)`

حذف یک کتاب از پایگاه داده را کنترل می کند.

ارگمان ها:

درخواست: درخواست HTTP. pk: کلید اصلی کتاب.

خروجی:

پس از حذف موفقیت آمیز به صفحه اصلی هدایت می شود.

`book.views.detail(request, pk)`

صفحه جزئیات کتاب را رندر می کند و با افزودن نمونه های کتاب مدیریت می کند.

ارگمان ها:

درخواست: درخواست HTTP. pk: کلید اصلی کتاب.

خروجی:

پاسخ HTML ارائه شده برای صفحه جزئیات کتاب.

`book.views.editBook(request, pk)`

ویرایش یک کتاب در پایگاه داده را کنترل می کند.

ارگمان ها:

درخواست: درخواست HTTP. pk: کلید اصلی کتاب.

خروجی:

پاسخ HTML ارائه شده برای صفحه 'ویرایش جزئیات کتاب'.

`book.views.home(request)`

صفحه اصلی را ارائه می دهد و فهرستی از کتاب های موجود را با قابلیت صفحه بندی و جستجو نمایش می دهد.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML ارائه شده برای صفحه اصلی.

`book.views.showAuthors(request)`

صفحه را نمایش می دهد که تمام نویسندگان موجود را نمایش می دهد.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML ارائه شده برای صفحه "نمایش نویسندگان".

`book.views.showBookInstance(request)`

صفحه ای را نمایش می دهد که تمام نمونه های کتاب قرض گرفته شده توسط کاربر وارد شده را نمایش می دهد.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML ارائه شده برای صفحه "نمایش نمونه کتاب ها".

`book.views.showGenres(request)`

صفحه را نمایش می دهد که همه ژانرهای کتاب موجود را نشان می دهد.

ارگمان ها:

درخواست: درخواست HTTP.

خروجی:

پاسخ HTML ارائه شده برای صفحه "نمایش ژانرها".

پکیج کتابخانه

زیر ماژول ها

library.urls ماژول

پیکربندی صفحه کتابخانه

این فایل شامل تنظیمات مسیریابی صفحه برای کل پروژه کتابخانه است.

فهرست «urlpatterns» صفحه ها را به نماها هدایت می کند. برای اطلاعات بیشتر لطفا لینک زیر را ببینید:

/https://docs.djangoproject.com/en/4.1/topics/http/urls

library.views ماژول

```
library.views.goHome(request)
```