1215

2023年12月15日　　上午 12:17

輸入ISBN後，判斷他是否存在 or 不存在於資料庫(Table=BOOK)中

To implement the functionality to check if an ISBN exists in the database (Table=BOOK), you can make an AJAX request to the server when the user enters an ISBN. The server can then check if the ISBN exists in the database and return the result to the client.

```
# views.py

from django.http import JsonResponse
from .models import Book

def check_isbn_exists(request, isbn):
    try:
        book = Book.objects.get(isbn=isbn)
        exists = True
    except Book.DoesNotExist:
        exists = False

    return JsonResponse({'exists': exists})
```

Make sure to add a URL pattern for this view in your urls.py:

```
# urls.py

from django.urls import path
from . import views

urlpatterns = [
    # Other URL patterns...
    path('check_isbn_exists/<str:isbn>/', views.check_isbn_exists,
name='check_isbn_exists'),
]
```

In your JavaScript section of the create_order.html, you can add a function to check the ISBN using AJAX:

```
<!-- Add this script block after your existing script block -->

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

<script>
    // ... Your existing code

    function checkIsbnExists(isbnField) {
        var isbn = $(isbnField).val();
        if (isbn !== "") {
            $.ajax({
                url: `/check_isbn_exists/${isbn}/`,
                type: 'GET',
                success: function (data) {
```

```
                    if (data.exists) {
                        alert('ISBN exists in the database.');
                    } else {
                        alert('ISBN does not exist in the database.');
                    }
                },
                error: function (error) {
                    console.error('Error checking ISBN:', error);
                }
            });
        }
    }

    // Attach this function to the change event of the ISBN field
    $('.book-isbn').on('change', function () {
        checkIsbnExists(this);
    });

    // ... Your existing code
</script>
```

增加分頁的功能，一頁顯示20筆資料

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>NTUSHB 販賣訂單</title>
</head>
<body>
    <div id="title">
        <h1>NTUSHB 販賣訂單</h1>
    </div>

    {% for page in orders.paginator.page_range %}
        <a href="?page={{ page }}">{{ page }}</a>
    {% endfor %}

    {% for order in orders %}
        <h2><a href="{% url 'order:sale_order_detail' user_id=user_id
order_id=order.order_id %}">訂單編號: {{ order.order_id }}</a></h2>
        <ul>
            {% for book in order.books %}
                <li>
                    書籍資訊：
                    ISBN: {{ book.isbn }},
                    價格: {{ book.price }},
                    書名: {{ book.title }}
                </li>
            {% endfor %}
            <li>
                ----------------------------------------------------------
```

```
---------------
            </li>
        </ul>
    {% endfor %}
</body>
</html>
```

在這個例子中，我們使用了Django的分頁模块提供的paginator和page_range屬性。這樣，你就可以在頁面上顯示一連串的頁碼，並透過點擊不同的頁碼來切換頁面。要注意的是，你需要在視圖中確保將分頁功能應用於orders對象，以便在模板中使用。

```
# views.py

from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from django.shortcuts import render
from .models import Order

def order_list(request):
    # 獲取所有訂單對象，你的邏輯可能會有所不同
    all_orders = Order.objects.all()

    # 定義每頁顯示的資料筆數
    items_per_page = 20

    # 將所有訂單對象分頁
    paginator = Paginator(all_orders, items_per_page)

    # 獲取當前頁碼
    page = request.GET.get('page')

    try:
        # 根據當前頁碼獲取對應頁的訂單對象
        orders = paginator.page(page)
    except PageNotAnInteger:
        # 如果page不是整數，將第一頁返回
        orders = paginator.page(1)
    except EmptyPage:
        # 如果page超出範圍，返回最後一頁
        orders = paginator.page(paginator.num_pages)

    return render(request, 'your_template.html', {'orders': orders})
```