題目3 簡化版解答：RESTful API風格URL

專案結構

```
bookstore/
├── bookstore/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py        # 主要URL配置
│   └── wsgi.py
├── api/
│   ├── __init__.py
│   ├── views.py        # API視圖函數
│   └── urls.py        # API URL配置
└── manage.py
```

---

bookstore/urls.py (主專案)

```python
"""
主專案URL配置 - 簡化版本
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    # 管理後台
    path('admin/', admin.site.urls),

    # API端點 - 直接包含，不使用版本控制
    path('api/', include('api.urls')),
]
```

---

api/urls.py (API URL配置)

python

```python
"""
API URL配置 - 簡化版RESTful設計
"""
from django.urls import path
from . import views

urlpatterns = [
    # API首頁
    path('', views.api_home, name='api_home'),

    # ===== 書籍相關端點 =====
    # GET /api/books/ - 取得書籍列表
    # POST /api/books/ - 新增書籍
    path('books/', views.books_list, name='books_list'),

    # GET /api/books/1/ - 取得特定書籍
    # PUT /api/books/1/ - 更新書籍
    # DELETE /api/books/1/ - 刪除書籍
    path('books/<int:book_id>/', views.book_detail, name='book_detail'),

    # ===== 書籍評論端點 =====
    # GET /api/books/1/reviews/ - 取得書籍評論
    # POST /api/books/1/reviews/ - 新增評論
    path('books/<int:book_id>/reviews/', views.book_reviews, name='book_reviews'),

    # GET /api/books/1/reviews/1/ - 取得特定評論
    # PUT /api/books/1/reviews/1/ - 更新評論
    # DELETE /api/books/1/reviews/1/ - 刪除評論
    path('books/<int:book_id>/reviews/<int:review_id>/', views.review_detail, name='review_detail'),

    # ===== 分類端點 =====
    # GET /api/categories/ - 取得所有分類
    path('categories/', views.categories_list, name='categories_list'),

    # GET /api/categories/1/ - 取得特定分類
    path('categories/<int:category_id>/', views.category_detail, name='category_detail'),

    # ===== 作者端點 =====
    # GET /api/authors/ - 取得所有作者
    path('authors/', views.authors_list, name='authors_list'),

    # GET /api/authors/1/ - 取得特定作者
    path('authors/<int:author_id>/', views.author_detail, name='author_detail'),

    # GET /api/authors/1/books/ - 取得作者的書籍
    path('authors/<int:author_id>/books/', views.author_books, name='author_books'),
    # ===== 購物車端點 =====
    # GET /api/cart/ - 取得購物車
    # POST /api/cart/ - 更新購物車
    path('cart/', views.cart, name='cart'),

    # ===== 訂單端點 =====
    # GET /api/orders/ - 取得訂單列表
    # POST /api/orders/ - 建立訂單
    path('orders/', views.orders_list, name='orders_list'),

    # GET /api/orders/1/ - 取得特定訂單
    path('orders/<int:order_id>/', views.order_detail, name='order_detail'),

    # ===== 搜尋端點 =====
    # GET /api/search/?q=keyword - 搜尋功能
    path('search/', views.search, name='search'),
]
```

---

api/views.py (API視圖函數)

python

```python
"""
API視圖函數 - 簡化版RESTful設計
"""
from django.http import JsonResponse, Http404
from django.views.decorators.csrf import csrf_exempt
from django.views.decorators.http import require_http_methods
import json

# ===== 模擬資料 =====
BOOKS_DATA = [
    {'id': 1, 'title': 'Python程式設計', 'author': '王小明', 'price': 450, 'category_id': 1},
    {'id': 2, 'title': 'Django網頁開發', 'author': '李小華', 'price': 520, 'category_id': 1},
    {'id': 3, 'title': '資料結構與演算法', 'author': '張大同', 'price': 380, 'category_id': 2},
    {'id': 4, 'title': '機器學習入門', 'author': '陳小美', 'price': 600, 'category_id': 3},
]

CATEGORIES_DATA = [
    {'id': 1, 'name': 'Programming', 'description': '程式設計相關書籍'},
    {'id': 2, 'name': 'Computer Science', 'description': '計算機科學相關書籍'},
    {'id': 3, 'name': 'Machine Learning', 'description': '機器學習相關書籍'},
]

REVIEWS_DATA = [
    {'id': 1, 'book_id': 1, 'rating': 5, 'comment': '很棒的書！', 'user': '讀者A'},
    {'id': 2, 'book_id': 1, 'rating': 4, 'comment': '內容豐富', 'user': '讀者B'},
    {'id': 3, 'book_id': 2, 'rating': 5, 'comment': 'Django入門首選', 'user': '讀者C'},
]

# ===== 輔助函數 =====
def find_book_by_id(book_id):
    """根據ID尋找書籍"""
    for book in BOOKS_DATA:
        if book['id'] == book_id:
            return book
    return None

def find_category_by_id(category_id):
    """根據ID尋找分類"""
    for category in CATEGORIES_DATA:
        if category['id'] == category_id:
            return category
    return None

def get_reviews_by_book_id(book_id):
    """取得特定書籍的所有評論"""
    return [review for review in REVIEWS_DATA if review['book_id'] == book_id]
```

```
def get_next_id(data_list):
    """取得下一個可用的ID"""
    if not data_list:
        return 1
    return max(item['id'] for item in data_list) + 1

# ===== API視圖函數 =====

def api_home(request):
    """API首頁 - 顯示可用的端點"""
    return JsonResponse({
        'message': 'Welcome to Bookstore API',
        'version': '1.0',
        'endpoints': {
            'books': '/api/books/',
            'categories': '/api/categories/',
            'authors': '/api/authors/',
            'cart': '/api/cart/',
            'orders': '/api/orders/',
            'search': '/api/search/?q=keyword'
        }
    })

@csrf_exempt
def books_list(request):
    """書籍列表端點"""
    if request.method == 'GET':
        # 處理查詢參數
        category = request.GET.get('category')
        search = request.GET.get('search')

        books = BOOKS_DATA.copy()

        # 篩選邏輯
        if category:
            try:
                category_id = int(category)
                books = [book for book in books if book['category_id'] == category_id]
            except ValueError:
                return JsonResponse({'error': 'Invalid category ID'}, status=400)

        if search:
            books = [book for book in books
                    if search.lower() in book['title'].lower() or
                       search.lower() in book['author'].lower()]

        return JsonResponse({
            'count': len(books),
            'books': books
        })
        })

    elif request.method == 'POST':
        try:
            data = json.loads(request.body)

            # 驗證必要欄位
            required_fields = ['title', 'author', 'price']
            for field in required_fields:
                if field not in data:
                    return JsonResponse({
                        'error': f'Missing required field: {field}'
                    }, status=400)

            # 建立新書籍
            new_book = {
                'id': get_next_id(BOOKS_DATA),
                'title': data['title'],
                'author': data['author'],
                'price': data['price'],
                'category_id': data.get('category_id', 1)
            }

            BOOKS_DATA.append(new_book)

            return JsonResponse({
                'message': 'Book created successfully',
                'book': new_book
            }, status=201)

        except json.JSONDecodeError:
            return JsonResponse({'error': 'Invalid JSON format'}, status=400)

    else:
        return JsonResponse({'error': 'Method not allowed'}, status=405)

@csrf_exempt
def book_detail(request, book_id):
    """單本書籍詳細資訊端點"""
    book = find_book_by_id(book_id)
    if not book:
        return JsonResponse({'error': 'Book not found'}, status=404)

    if request.method == 'GET':
        return JsonResponse({'book': book})

    elif request.method == 'PUT':
        try:
            data = json.loads(request.body)
            # 更新書籍資料
            book.update(data)

            return JsonResponse({
                'message': 'Book updated successfully',
                'book': book
            })

        except json.JSONDecodeError:
            return JsonResponse({'error': 'Invalid JSON format'}, status=400)

    elif request.method == 'DELETE':
        BOOKS_DATA.remove(book)
        return JsonResponse({'message': 'Book deleted successfully'}, status=204)

    else:
        return JsonResponse({'error': 'Method not allowed'}, status=405)

@csrf_exempt
def book_reviews(request, book_id):
    """書籍評論端點"""
    # 檢查書籍是否存在
    book = find_book_by_id(book_id)
    if not book:
        return JsonResponse({'error': 'Book not found'}, status=404)

    if request.method == 'GET':
        reviews = get_reviews_by_book_id(book_id)
        return JsonResponse({
            'book_title': book['title'],
            'count': len(reviews),
            'reviews': reviews
        })

    elif request.method == 'POST':
        try:
            data = json.loads(request.body)

            new_review = {
                'id': get_next_id(REVIEWS_DATA),
                'book_id': book_id,
                'rating': data.get('rating', 5),
                'comment': data.get('comment', ''),
                'user': data.get('user', 'Anonymous')
            }

            REVIEWS_DATA.append(new_review)

            return JsonResponse({
```

```
            'message': 'Review created successfully',
            'review': new_review
        }, status=201)

    except json.JSONDecodeError:
        return JsonResponse({'error': 'Invalid JSON format'}, status=400)

else:
    return JsonResponse({'error': 'Method not allowed'}, status=405)

@csrf_exempt
def review_detail(request, book_id, review_id):
    """單個評論詳細資訊端點"""
    # 找到評論
    review = None
    for r in REVIEWS_DATA:
        if r['id'] == review_id and r['book_id'] == book_id:
            review = r
            break

    if not review:
        return JsonResponse({'error': 'Review not found'}, status=404)

    if request.method == 'GET':
        return JsonResponse({'review': review})

    elif request.method == 'PUT':
        try:
            data = json.loads(request.body)
            review.update(data)
            return JsonResponse({
                'message': 'Review updated successfully',
                'review': review
            })
        except json.JSONDecodeError:
            return JsonResponse({'error': 'Invalid JSON format'}, status=400)

    elif request.method == 'DELETE':
        REVIEWS_DATA.remove(review)
        return JsonResponse({'message': 'Review deleted successfully'}, status=204)

    else:
        return JsonResponse({'error': 'Method not allowed'}, status=405)

def categories_list(request):
    """分類列表端點"""
    return JsonResponse({
        'count': len(CATEGORIES_DATA),
        'categories': CATEGORIES_DATA
    })
```

```python
    })

def category_detail(request, category_id):
    """分類詳細資訊端點"""
    category = find_category_by_id(category_id)
    if not category:
        return JsonResponse({'error': 'Category not found'}, status=404)

    # 取得該分類的書籍
    books = [book for book in BOOKS_DATA if book['category_id'] == category_id]

    return JsonResponse({
        'category': category,
        'books_count': len(books),
        'books': books
    })

def authors_list(request):
    """作者列表端點"""
    # 從書籍資料中提取唯一的作者
    authors = {}
    author_id = 1

    for book in BOOKS_DATA:
        author_name = book['author']
        if author_name not in authors:
            authors[author_name] = {
                'id': author_id,
                'name': author_name,
                'books_count': 0
            }
            author_id += 1
        authors[author_name]['books_count'] += 1

    return JsonResponse({
        'count': len(authors),
        'authors': list(authors.values())
    })

def author_detail(request, author_id):
    """作者詳細資訊端點"""
    # 建立作者映射
    authors = {}
    current_id = 1

    for book in BOOKS_DATA:
        author_name = book['author']
        if author_name not in authors:
            authors[author_name] = {
                'id': current_id,
                'name': author_name
            }
            current_id += 1

    # 找到對應的作者
    author = None
    for author_data in authors.values():
        if author_data['id'] == author_id:
            author = author_data
            break

    if not author:
        return JsonResponse({'error': 'Author not found'}, status=404)

    # 取得作者的書籍
    author_books = [book for book in BOOKS_DATA if book['author'] == author['name']]

    return JsonResponse({
        'author': author,
        'books_count': len(author_books),
        'books': author_books
    })

def author_books(request, author_id):
    """作者書籍端點"""
    # 這裡重複使用 author_detail 的邏輯
    return author_detail(request, author_id)

def cart(request):
    """購物車端點"""
    if request.method == 'GET':
        # 模擬購物車資料
        cart_data = {
            'items': [
                {
                    'book_id': 1,
                    'book_title': 'Python程式設計',
                    'quantity': 2,
                    'unit_price': 450,
                    'subtotal': 900
                },
                {
                    'book_id': 2,
                    'book_title': 'Django網頁開發',
                    'quantity': 1,
                    'unit_price': 520,
                    'subtotal': 520
                }
            ],
            'total_items': 3,
            'total_amount': 1420
        }
        return JsonResponse(cart_data)

    elif request.method == 'POST':
        # 模擬更新購物車
        return JsonResponse({
            'message': 'Cart updated successfully'
        })

    else:
        return JsonResponse({'error': 'Method not allowed'}, status=405)

@csrf_exempt
def orders_list(request):
    """訂單列表端點"""
    if request.method == 'GET':
        # 模擬訂單資料
        orders = [
            {
                'id': 1,
                'order_date': '2024-01-15',
                'total_amount': 970,
                'status': 'completed'
            },
            {
                'id': 2,
                'order_date': '2024-01-20',
                'total_amount': 1420,
                'status': 'processing'
            }
        ]
        return JsonResponse({
            'count': len(orders),
            'orders': orders
        })

    elif request.method == 'POST':
        try:
            data = json.loads(request.body)
            new_order = {
                'id': 3,
                'order_date': '2024-01-25',
                'total_amount': data.get('total_amount', 0),
                'status': 'pending'
            }
            return JsonResponse({
                'message': 'Order created successfully',
                'order': new_order
```

```python
            }, status=201)
        except json.JSONDecodeError:
            return JsonResponse({'error': 'Invalid JSON format'}, status=400)

    else:
        return JsonResponse({'error': 'Method not allowed'}, status=405)

def order_detail(request, order_id):
    """訂單詳細資訊端點"""
    # 模擬訂單詳細資料
    if order_id == 1:
        order = {
            'id': 1,
            'order_date': '2024-01-15',
            'status': 'completed',
            'items': [
                {'book_id': 1, 'title': 'Python程式設計', 'quantity': 1, 'price': 450},
                {'book_id': 3, 'title': '資料結構與演算法', 'quantity': 1, 'price': 380}
            ],
            'total_amount': 830
        }
    elif order_id == 2:
        order = {
            'id': 2,
            'order_date': '2024-01-20',
            'status': 'processing',
            'items': [
                {'book_id': 2, 'title': 'Django網頁開發', 'quantity': 1, 'price': 520},
                {'book_id': 4, 'title': '機器學習入門', 'quantity': 1, 'price': 600}
            ],
            'total_amount': 1120
        }
    else:
        return JsonResponse({'error': 'Order not found'}, status=404)

    return JsonResponse({'order': order})

def search(request):
    """搜尋端點"""
    query = request.GET.get('q', '').strip()

    if not query:
        return JsonResponse({
            'error': 'Please provide search query with ?q=keyword'
        }, status=400)

    # 搜尋書籍
    results = []
    for book in BOOKS_DATA:
        if (query.lower() in book['title'].lower() or
            query.lower() in book['author'].lower()):
            results.append(book)

    return JsonResponse({
        'query': query,
        'count': len(results),
        'results': results
    })
```

API測試範例

使用curl測試

```bash
# 1. API首頁
curl -X GET http://localhost:8000/api/

# 2. 取得所有書籍
curl -X GET http://localhost:8000/api/books/

# 3. 取得特定書籍
curl -X GET http://localhost:8000/api/books/1/

# 4. 新增書籍
curl -X POST http://localhost:8000/api/books/ \
  -H "Content-Type: application/json" \
  -d '{"title": "新書", "author": "新作者", "price": 399}'

# 5. 更新書籍
curl -X PUT http://localhost:8000/api/books/1/ \
  -H "Content-Type: application/json" \
  -d '{"title": "更新的書名", "price": 499}'

# 6. 刪除書籍
curl -X DELETE http://localhost:8000/api/books/1/

# 7. 取得書籍評論
curl -X GET http://localhost:8000/api/books/1/reviews/

# 8. 新增評論
curl -X POST http://localhost:8000/api/books/1/reviews/ \
  -H "Content-Type: application/json" \
  -d '{"rating": 5, "comment": "很棒的書！", "user": "測試用戶"}'

# 9. 搜尋書籍
curl -X GET "http://localhost:8000/api/search/?q=Python"

# 10. 取得分類
curl -X GET http://localhost:8000/api/categories/

# 11. 取得購物車
curl -X GET http://localhost:8000/api/cart/

# 12. 取得訂單
curl -X GET http://localhost:8000/api/orders/
```

使用瀏覽器測試

直接在瀏覽器中訪問以下URL：

- `http://localhost:8000/api/` - API首頁
- `http://localhost:8000/api/books/` - 書籍列表
- `http://localhost:8000/api/books/1/` - 特定書籍
- `http://localhost:8000/api/categories/` - 分類列表
- `http://localhost:8000/api/search/?q=Python` - 搜尋結果

主要特點

✅ 簡化的設計

- **只使用函數視圖**：沒有複雜的類別視圖
- **直接URL配置**：不使用命名空間和app_name
- **基本RESTful**：遵循REST原則但保持簡單
- **清晰的結構**：易於理解和維護

✅ RESTful設計原則

- **HTTP方法對應**：GET(查詢)、POST(新增)、PUT(更新)、DELETE(刪除)
- **資源導向URL**：`/books/`、`/books/1/`、`/books/1/reviews/`
- **統一回應格式**：JSON格式回應
- **適當的狀態碼**：200、201、400、404等

✅ 實用功能

- **搜尋功能**：支援關鍵字搜尋
- **篩選功能**：支援分類篩選
- **巢狀資源**：書籍評論、作者書籍
- **錯誤處理**：適當的錯誤訊息

這個簡化版本保持了RESTful API的核心概念，但使用最基本的Django功能，非常適合初學者理解和學習。