Django URLconf 練習解答 (題目1-3)

題目1 解答：基本URL結構

專案結構

```
bookstore/
├── bookstore/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py        # 主要URL配置
│   └── wsgi.py
├── books/
│   ├── __init__.py
│   ├── apps.py
│   ├── models.py
│   ├── views.py
│   └── urls.py        # Books app URL配置
└── manage.py
```

bookstore/urls.py (主專案)

python

```python
"""
主專案URL配置
"""
from django.contrib import admin
from django.urls import path, include
from django.views.generic import TemplateView

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

    # 首頁
    path('', TemplateView.as_view(template_name='home.html'), name='home'),

    # 關於我們
    path('about/', TemplateView.as_view(template_name='about.html'), name='about'),

    # 聯絡我們
    path('contact/', TemplateView.as_view(template_name='contact.html'), name='contact'),

    # 書籍相關URL (使用include包含子應用URL)
    path('books/', include('books.urls')),
]

# 開發環境錯誤處理
from django.conf import settings
if settings.DEBUG:
    from django.conf.urls.static import static
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

books/urls.py (Books app)

```python
"""
Books應用URL配置
"""
from django.urls import path
from . import views

# 設定應用命名空間
app_name = 'books'

urlpatterns = [
    # 書籍列表 - /books/
    path('', views.book_list, name='list'),

    # 也可以使用更明確的路徑
    path('list/', views.book_list, name='book_list'),
]
```

books/views.py

python

```python
"""
Books應用視圖函數
"""
from django.shortcuts import render
from django.http import HttpResponse

def book_list(request):
    """書籍列表頁面"""
    # 模擬書籍資料
    books = [
        {'id': 1, 'title': 'Python程式設計', 'author': '王小明', 'price': 450},
        {'id': 2, 'title': 'Django網頁開發', 'author': '李小華', 'price': 520},
        {'id': 3, 'title': '資料結構與演算法', 'author': '張大同', 'price': 380},
    ]

    context = {
        'books': books,
        'page_title': '書籍列表'
    }
    return render(request, 'books/book_list.html', context)

# 也可以使用類別視圖
from django.views.generic import ListView

class BookListView(ListView):
    """書籍列表類別視圖"""
    template_name = 'books/book_list.html'
    context_object_name = 'books'

    def get_queryset(self):
        # 這裡應該從資料庫取得資料，暫時返回模擬資料
        return [
            {'id': 1, 'title': 'Python程式設計', 'author': '王小明', 'price': 450},
            {'id': 2, 'title': 'Django網頁開發', 'author': '李小華', 'price': 520},
        ]
```

測試URL配置

python

```python
# 在Django shell中測試URL解析
from django.urls import reverse

# 測試URL反向解析
print(reverse('home'))          # 輸出: /
print(reverse('about'))         # 輸出: /about/
print(reverse('books:list'))    # 輸出: /books/
```

題目2 解答：動態URL參數處理

books/urls.py (擴展版本)

python

```python
"""
Books應用URL配置 - 包含動態參數
"""
from django.urls import path, re_path
from . import views

app_name = 'books'

urlpatterns = [
    # 基本路由
    path('', views.book_list, name='list'),

    # 書籍詳細頁面 - 使用int轉換器
    path('<int:book_id>/', views.book_detail, name='detail'),

    # 也可以使用pk作為參數名（更符合Django慣例）
    path('detail/<int:pk>/', views.book_detail_pk, name='detail_pk'),

    # 書籍分類頁面 - 使用str轉換器
    path('category/<str:category_name>/', views.books_by_category, name='category'),

    # 作者頁面 - 使用slug轉換器
    path('author/<slug:author_slug>/', views.books_by_author, name='author'),

    # 搜尋頁面
    path('search/', views.book_search, name='search'),

    # 進階：使用正則表達式匹配ISBN
    re_path(r'^isbn/(?P<isbn>\d{3}-\d{1,5}-\d{1,7}-\d{1,7}-\d{1})/$',
            views.book_by_isbn, name='isbn'),

    # 年份篩選 - 使用自訂轉換器
    path('year/<int:year>/', views.books_by_year, name='year'),

    # 可選參數示例 - 分頁功能
    path('page/<int:page>/', views.book_list_paginated, name='paginated'),
]
```

books/views.py (擴展版本)

python

```python
"""
Books應用視圖函數 - 包含參數處理
"""
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponse, Http404
from django.core.paginator import Paginator
from django.db.models import Q

# 模擬資料
MOCK_BOOKS = [
    {
        'id': 1, 'title': 'Python程式設計', 'author': '王小明',
        'author_slug': 'wang-xiaoming', 'category': 'programming',
        'price': 450, 'isbn': '978-1-234-56789-0', 'year': 2023
    },
    {
        'id': 2, 'title': 'Django網頁開發', 'author': '李小華',
        'author_slug': 'li-xiaohua', 'category': 'web-development',
        'price': 520, 'isbn': '978-1-234-56789-1', 'year': 2024
    },
    {
        'id': 3, 'title': '資料結構與演算法', 'author': '張大同',
        'author_slug': 'zhang-datong', 'category': 'computer-science',
        'price': 380, 'isbn': '978-1-234-56789-2', 'year': 2023
    },
    {
        'id': 4, 'title': '機器學習實戰', 'author': '王小明',
        'author_slug': 'wang-xiaoming', 'category': 'machine-learning',
        'price': 600, 'isbn': '978-1-234-56789-3', 'year': 2024
    },
]

def book_list(request):
    """書籍列表頁面"""
    context = {
        'books': MOCK_BOOKS,
        'page_title': '所有書籍'
    }
    return render(request, 'books/book_list.html', context)

def book_detail(request, book_id):
    """書籍詳細頁面 - 使用book_id參數"""
    try:
        book = next(book for book in MOCK_BOOKS if book['id'] == book_id)
    except StopIteration:
        raise Http404("書籍不存在")
    context = {
        'book': book,
        'page_title': f'書籍詳情 - {book["title"]}'
    }
    return render(request, 'books/book_detail.html', context)

def book_detail_pk(request, pk):
    """書籍詳細頁面 - 使用pk參數（Django慣例）"""
    try:
        book = next(book for book in MOCK_BOOKS if book['id'] == pk)
    except StopIteration:
        raise Http404("書籍不存在")

    context = {
        'book': book,
        'page_title': f'書籍詳情 - {book["title"]}'
    }
    return render(request, 'books/book_detail.html', context)

def books_by_category(request, category_name):
    """依分類顯示書籍"""
    filtered_books = [
        book for book in MOCK_BOOKS
        if book['category'] == category_name
    ]

    if not filtered_books:
        raise Http404(f"分類 '{category_name}' 不存在或無書籍")

    context = {
        'books': filtered_books,
        'category_name': category_name,
        'page_title': f'分類：{category_name}'
    }
    return render(request, 'books/category.html', context)

def books_by_author(request, author_slug):
    """依作者顯示書籍"""
    filtered_books = [
        book for book in MOCK_BOOKS
        if book['author_slug'] == author_slug
    ]

    if not filtered_books:
        raise Http404(f"作者 '{author_slug}' 不存在或無書籍")

    context = {
        'books': filtered_books,
        'author_name': filtered_books[0]['author'],
        'page_title': f'作者：{filtered_books[0]["author"]}'
        'page_title': f'作者: {filtered_books[0]['author']}'
    }
    return render(request, 'books/author.html', context)

def book_search(request):
    """書籍搜尋功能"""
    query = request.GET.get('q', '')
    books = []

    if query:
        # 搜尋書名或作者
        books = [
            book for book in MOCK_BOOKS
            if query.lower() in book['title'].lower() or
               query.lower() in book['author'].lower()
        ]

    context = {
        'books': books,
        'query': query,
        'page_title': f'搜尋結果: {query}' if query else '書籍搜尋'
    }
    return render(request, 'books/search.html', context)

def book_by_isbn(request, isbn):
    """使用ISBN查詢書籍"""
    try:
        book = next(book for book in MOCK_BOOKS if book['isbn'] == isbn)
    except StopIteration:
        raise Http404(f"ISBN '{isbn}' 的書籍不存在")

    context = {
        'book': book,
        'page_title': f'ISBN: {isbn}'
    }
    return render(request, 'books/book_detail.html', context)

def books_by_year(request, year):
    """依出版年份顯示書籍"""
    filtered_books = [
        book for book in MOCK_BOOKS
        if book['year'] == year
    ]

    context = {
        'books': filtered_books,
        'year': year,
        'page_title': f'{year}年出版書籍'
    }
    return render(request, 'books/year.html', context)
```

```python
def book_list_paginated(request, page=1):
    """分頁書籍列表"""
    paginator = Paginator(MOCK_BOOKS, 2)  # 每頁2本書

    try:
        books_page = paginator.page(page)
    except:
        raise Http404("頁面不存在")

    context = {
        'books_page': books_page,
        'page_title': f'書籍列表 - 第{page}頁'
    }
    return render(request, 'books/book_list_paginated.html', context)
```

URL測試範例

```python
# 在視圖中使用URL反向解析
from django.urls import reverse
from django.shortcuts import redirect

def redirect_to_book(request, book_id):
    """重定向到書籍詳細頁面的範例"""
    return redirect('books:detail', book_id=book_id)

def redirect_to_category(request, category):
    """重定向到分類頁面的範例"""
    return redirect('books:category', category_name=category)

# 在模板中使用URL標籤
# {% url 'books:detail' book.id %}
# {% url 'books:category' 'programming' %}
# {% url 'books:author' 'wang-xiaoming' %}
```

題目3 解答：RESTful API風格URL

api/init.py

```python
# 建立API應用
```

api/urls.py

python

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

# API版本1
app_name = 'api_v1'

# 書籍相關API端點
book_patterns = [
    path('', views.BookListCreateView.as_view(), name='book-list-create'),
    path('<int:pk>/', views.BookDetailView.as_view(), name='book-detail'),
    path('<int:book_id>/reviews/', views.ReviewListCreateView.as_view(), name='review-list-create'),
    path('<int:book_id>/reviews/<int:pk>/', views.ReviewDetailView.as_view(), name='review-detail'),
]

# 主要URL模式
urlpatterns = [
    # API根路徑
    path('', views.api_root, name='api-root'),

    # 書籍端點
    path('books/', include(book_patterns)),

    # 分類端點
    path('categories/', views.CategoryListView.as_view(), name='category-list'),
    path('categories/<int:pk>/', views.CategoryDetailView.as_view(), name='category-detail'),

    # 作者端點
    path('authors/', views.AuthorListView.as_view(), name='author-list'),
    path('authors/<int:pk>/', views.AuthorDetailView.as_view(), name='author-detail'),
    path('authors/<int:author_id>/books/', views.AuthorBooksView.as_view(), name='author-books'),

    # 購物車端點
    path('cart/', views.CartView.as_view(), name='cart'),
    path('cart/items/', views.CartItemListView.as_view(), name='cart-items'),
    path('cart/items/<int:pk>/', views.CartItemDetailView.as_view(), name='cart-item-detail'),

    # 訂單端點
    path('orders/', views.OrderListCreateView.as_view(), name='order-list-create'),
    path('orders/<int:pk>/', views.OrderDetailView.as_view(), name='order-detail'),

    # 搜尋端點
    path('search/', views.SearchView.as_view(), name='search'),
]
```

bookstore/urls.py (更新主URL配置)

```python
"""
主專案URL配置 - 包含API版本控制
"""
from django.contrib import admin
from django.urls import path, include
from django.views.generic import TemplateView

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

    # 基本頁面
    path('', TemplateView.as_view(template_name='home.html'), name='home'),
    path('about/', TemplateView.as_view(template_name='about.html'), name='about'),
    path('contact/', TemplateView.as_view(template_name='contact.html'), name='contact'),

    # 書籍頁面
    path('books/', include('books.urls')),

    # API端點 - 版本控制
    path('api/v1/', include('api.urls', namespace='api_v1')),

    # API文件 (可選)
    path('api/docs/', TemplateView.as_view(template_name='api/docs.html'), name='api-docs'),
]
```

api/views.py

python

```python
"""
API視圖 - RESTful設計
"""
from django.http import JsonResponse, Http404
from django.views import View
from django.views.generic import TemplateView
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
from django.urls import reverse
import json

# 模擬資料
MOCK_BOOKS = [
    {'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},
]

MOCK_CATEGORIES = [
    {'id': 1, 'name': 'Programming'},
    {'id': 2, 'name': 'Computer Science'},
]

MOCK_REVIEWS = [
    {'id': 1, 'book_id': 1, 'rating': 5, 'comment': '很棒的書！'},
    {'id': 2, 'book_id': 1, 'rating': 4, 'comment': '內容豐富'},
]

def api_root(request):
    """API根端點 - 提供所有端點資訊"""
    endpoints = {
        'books': request.build_absolute_uri(reverse('api_v1:book-list-create')),
        'categories': request.build_absolute_uri(reverse('api_v1:category-list')),
        'authors': request.build_absolute_uri(reverse('api_v1:author-list')),
        'orders': request.build_absolute_uri(reverse('api_v1:order-list-create')),
        'search': request.build_absolute_uri(reverse('api_v1:search')),
    }
    return JsonResponse({
        'message': 'Welcome to Bookstore API v1',
        'endpoints': endpoints
    })

@method_decorator(csrf_exempt, name='dispatch')
class BookListCreateView(View):
    """書籍列表和建立端點"""

    def get(self, request):
        """GET /api/v1/books/ - 取得書籍列表"""
        # 處理查詢參數
        category = request.GET.get('category')
        search = request.GET.get('search')

        books = MOCK_BOOKS.copy()

        # 篩選邏輯
        if category:
            books = [b for b in books if b['category_id'] == int(category)]
        if search:
            books = [b for b in books if search.lower() in b['title'].lower()]

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

    def post(self, request):
        """POST /api/v1/books/ - 建立新書籍"""
        try:
            data = json.loads(request.body)

            # 簡單驗證
            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 (模擬)
            new_book = {
                'id': len(MOCK_BOOKS) + 1,
                'title': data['title'],
                'author': data['author'],
                'price': data['price'],
                'category_id': data.get('category_id', 1)
            }
            MOCK_BOOKS.append(new_book)

            return JsonResponse(new_book, status=201)

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

@method_decorator(csrf_exempt, name='dispatch')
class BookDetailView(View):
    """書籍詳細資訊端點"""
    def get_book(self, pk):
        """取得單本書籍"""
        try:
            return next(book for book in MOCK_BOOKS if book['id'] == pk)
        except StopIteration:
            raise Http404("Book not found")

    def get(self, request, pk):
        """GET /api/v1/books/{id}/ - 取得特定書籍"""
        book = self.get_book(pk)
        return JsonResponse(book)

    def put(self, request, pk):
        """PUT /api/v1/books/{id}/ - 更新書籍"""
        book = self.get_book(pk)

        try:
            data = json.loads(request.body)

            # 更新資料
            book.update(data)

            return JsonResponse(book)

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

    def delete(self, request, pk):
        """DELETE /api/v1/books/{id}/ - 刪除書籍"""
        book = self.get_book(pk)

        # 從列表中移除 (模擬刪除)
        MOCK_BOOKS.remove(book)

        return JsonResponse({'message': 'Book deleted successfully'}, status=204)

@method_decorator(csrf_exempt, name='dispatch')
class ReviewListCreateView(View):
    """書籍評論列表和建立端點"""

    def get(self, request, book_id):
        """GET /api/v1/books/{id}/reviews/ - 取得書籍評論"""
        reviews = [r for r in MOCK_REVIEWS if r['book_id'] == book_id]
        return JsonResponse({
            'count': len(reviews),
            'results': reviews
        })

    def post(self, request, book_id):
```

```python
        """POST /api/v1/books/{id}/reviews/ - 新增評論"""
        # 檢查書籍是否存在
        try:
            next(book for book in MOCK_BOOKS if book['id'] == book_id)
        except StopIteration:
            return JsonResponse({'error': 'Book not found'}, status=404)

        try:
            data = json.loads(request.body)

            new_review = {
                'id': len(MOCK_REVIEWS) + 1,
                'book_id': book_id,
                'rating': data.get('rating', 5),
                'comment': data.get('comment', '')
            }
            MOCK_REVIEWS.append(new_review)

            return JsonResponse(new_review, status=201)

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

class ReviewDetailView(View):
    """評論詳細資訊端點"""

    def get(self, request, book_id, pk):
        """GET /api/v1/books/{book_id}/reviews/{id}/ - 取得特定評論"""
        try:
            review = next(r for r in MOCK_REVIEWS
                         if r['id'] == pk and r['book_id'] == book_id)
            return JsonResponse(review)
        except StopIteration:
            raise Http404("Review not found")

class CategoryListView(View):
    """分類列表端點"""

    def get(self, request):
        """GET /api/v1/categories/ - 取得所有分類"""
        return JsonResponse({
            'count': len(MOCK_CATEGORIES),
            'results': MOCK_CATEGORIES
        })

class CategoryDetailView(View):
    """分類詳細資訊端點"""

    def get(self, request, pk):
        """GET /api/v1/categories/{id}/ - 取得特定分類"""
        """GET /api/v1/categories/{id}/ - 取得特定分類"""
        try:
            category = next(c for c in MOCK_CATEGORIES if c['id'] == pk)
            # 包含該分類的書籍
            books = [b for b in MOCK_BOOKS if b['category_id'] == pk]
            category['books'] = books
            return JsonResponse(category)
        except StopIteration:
            raise Http404("Category not found")

class AuthorListView(View):
    """作者列表端點"""

    def get(self, request):
        """GET /api/v1/authors/ - 取得所有作者"""
        authors = list(set(book['author'] for book in MOCK_BOOKS))
        author_list = [{'id': i+1, 'name': author} for i, author in enumerate(authors)]
        return JsonResponse({
            'count': len(author_list),
            'results': author_list
        })

class AuthorDetailView(View):
    """作者詳細資訊端點"""

    def get(self, request, pk):
        """GET /api/v1/authors/{id}/ - 取得特定作者"""
        authors = list(set(book['author'] for book in MOCK_BOOKS))
        try:
            author_name = authors[pk-1]
            return JsonResponse({
                'id': pk,
                'name': author_name
            })
        except IndexError:
            raise Http404("Author not found")

class AuthorBooksView(View):
    """作者書籍端點"""

    def get(self, request, author_id):
        """GET /api/v1/authors/{id}/books/ - 取得作者的所有書籍"""
        authors = list(set(book['author'] for book in MOCK_BOOKS))
        try:
            author_name = authors[author_id-1]
            books = [b for b in MOCK_BOOKS if b['author'] == author_name]
            return JsonResponse({
                'author': author_name,
                'count': len(books),
                'books': books
                'books': books
            })
        except IndexError:
            raise Http404("Author not found")

class CartView(View):
    """購物車端點"""

    def get(self, request):
        """GET /api/v1/cart/ - 取得購物車內容"""
        # 模擬購物車資料
        cart = {
            'items': [
                {'book_id': 1, 'quantity': 2, 'price': 450},
                {'book_id': 2, 'quantity': 1, 'price': 520},
            ],
            'total': 1420
        }
        return JsonResponse(cart)

class CartItemListView(View):
    """購物車項目列表端點"""

    def get(self, request):
        """GET /api/v1/cart/items/ - 取得購物車項目"""
        return JsonResponse({
            'message': 'Cart items endpoint'
        })

class CartItemDetailView(View):
    """購物車項目詳細端點"""

    def get(self, request, pk):
        """GET /api/v1/cart/items/{id}/ - 取得特定購物車項目"""
        return JsonResponse({
            'message': f'Cart item {pk} endpoint'
        })

class OrderListCreateView(View):
    """訂單列表和建立端點"""

    def get(self, request):
        """GET /api/v1/orders/ - 取得訂單列表"""
        return JsonResponse({
            'message': 'Orders list endpoint'
        })

    def post(self, request):
        """POST /api/v1/orders/ - 建立新訂單"""
        return JsonResponse({
            'message': 'Create order endpoint'
        })

class OrderDetailView(View):
    """訂單詳細資訊端點"""

    def get(self, request, pk):
        """GET /api/v1/orders/{id}/ - 取得特定訂單"""
        return JsonResponse({
            'message': f'Order {pk} detail endpoint'
        })

class SearchView(View):
    """搜尋端點"""

    def get(self, request):
        """GET /api/v1/search/ - 搜尋功能"""
        query = request.GET.get('q', '')
        if not query:
            return JsonResponse({
                'message': 'Please provide search query with ?q=keyword'
            })

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

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

解答重點說明

題目1 重點

- **URL結構組織**：使用`include()`將URL分層管理
- **命名空間**：使用`app_name`避免URL命名衝突
- **視圖函數**：基本的view函數撰寫
- **URL反向解析**：使用`reverse()`和`{% url %}`

題目2 重點

- **路徑參數**：`<int:pk>`, `<str:name>`, `<slug:slug>`
- **正則表達式**：使用`re_path()`處理複雜模式
- **參數驗證**：在view中處理參數錯誤
- **查詢字串**：使用`request.GET.get()`處理搜尋參數
- **錯誤處理**：適當使用`Http404`處理不存在的資源

題目3 重點

- **RESTful設計**：HTTP方法與CRUD操作對應
- **API版本控制**：使用URL路徑進行版本管理
- **巢狀資源**：書籍評論等相關資源的URL設計
- **JSON回應**：使用`JsonResponse`返回API資料
- **CSRF豁免**：API端點使用`@csrf_exempt`

模板檔案範例

templates/books/book_list.html

```html
<!DOCTYPE html>
<html lang="zh-TW">
<head>
    <meta charset="UTF-8">
    <title>{{ page_title }}</title>
    <style>
        .book-card {
            border: 1px solid #ddd;
            padding: 15px;
            margin: 10px 0;
            border-radius: 5px;
        }
        .book-title { font-weight: bold; }
        .book-author { color: #666; }
        .book-price { color: #e74c3c; font-weight: bold; }
    </style>
</head>
<body>
    <h1>{{ page_title }}</h1>

    <div class="book-list">
        {% for book in books %}
            <div class="book-card">
                <div class="book-title">
                    <a href="{% url 'books:detail' book.id %}">{{ book.title }}</a>
                </div>
                <div class="book-author">作者：{{ book.author }}</div>
                <div class="book-price">價格：NT$ {{ book.price }}</div>
            </div>
        {% empty %}
            <p>目前沒有書籍資料</p>
        {% endfor %}
    </div>

    <div class="navigation">
        <a href="{% url 'home' %}">回首頁</a>
    </div>
</body>
</html>
```

templates/books/book_detail.html

html

```html
<!DOCTYPE html>
<html lang="zh-TW">
<head>
    <meta charset="UTF-8">
    <title>{{ page_title }}</title>
    <style>
        .book-detail {
            max-width: 600px;
            margin: 20px auto;
            padding: 20px;
            border: 1px solid #ddd;
            border-radius: 8px;
        }
        .book-info { margin-bottom: 15px; }
        .label { font-weight: bold; }
        .navigation { margin-top: 20px; }
        .btn {
            padding: 8px 16px;
            margin-right: 10px;
            text-decoration: none;
            background: #007bff;
            color: white;
            border-radius: 4px;
        }
    </style>
</head>
<body>
    <div class="book-detail">
        <h1>{{ book.title }}</h1>

        <div class="book-info">
            <span class="label">作者：</span>{{ book.author }}
        </div>

        <div class="book-info">
            <span class="label">價格：</span>NT$ {{ book.price }}
        </div>

        {% if book.isbn %}
        <div class="book-info">
            <span class="label">ISBN：</span>{{ book.isbn }}
        </div>
        {% endif %}

        {% if book.category %}
        <div class="book-info">
            <span class="label">分類：</span>
            <a href="{% url 'books:category' book.category %}">{{ book.category }}</a>
        </div>
        {% endif %}

        <div class="navigation">
            <a href="{% url 'books:list' %}" class="btn">回書籍列表</a>
            <a href="{% url 'home' %}" class="btn">回首頁</a>
        </div>
    </div>
</body>
</html>
```

templates/books/search.html

```html
<!DOCTYPE html>
<html lang="zh-TW">
<head>
  <meta charset="UTF-8">
  <title>{{ page_title }}</title>
</head>
<body>
  <h1>書籍搜尋</h1>

  <form method="get" action="{% url 'books:search' %}">
    <input type="text" name="q" value="{{ query }}" placeholder="輸入書名或作者...">
    <button type="submit">搜尋</button>
  </form>

  {% if query %}
    <h2>搜尋結果：「{{ query }}」</h2>

    {% if books %}
      <div class="search-results">
        {% for book in books %}
          <div class="book-card">
            <h3><a href="{% url 'books:detail' book.id %}">{{ book.title }}</a></h3>
            <p>作者：{{ book.author }}</p>
            <p>價格：NT$ {{ book.price }}</p>
          </div>
        {% endfor %}
      </div>
    {% else %}
      <p>沒有找到相關書籍</p>
    {% endif %}
  {% endif %}

  <div class="navigation">
    <a href="{% url 'books:list' %}">瀏覽所有書籍</a> |
    <a href="{% url 'home' %}">回首頁</a>
  </div>
</body>
</html>
```

API測試範例

使用curl測試API端點

```bash
# 取得所有書籍
curl -X GET http://localhost:8000/api/v1/books/

# 取得特定書籍
curl -X GET http://localhost:8000/api/v1/books/1/

# 建立新書籍
curl -X POST http://localhost:8000/api/v1/books/ \
  -H "Content-Type: application/json" \
  -d '{"title": "新書", "author": "新作者", "price": 399}'

# 更新書籍
curl -X PUT http://localhost:8000/api/v1/books/1/ \
  -H "Content-Type: application/json" \
  -d '{"title": "更新的書名", "author": "王小明", "price": 499}'

# 刪除書籍
curl -X DELETE http://localhost:8000/api/v1/books/1/

# 取得書籍評論
curl -X GET http://localhost:8000/api/v1/books/1/reviews/

# 新增評論
curl -X POST http://localhost:8000/api/v1/books/1/reviews/ \
  -H "Content-Type: application/json" \
  -d '{"rating": 5, "comment": "非常好的書！"}'

# 搜尋功能
curl -X GET "http://localhost:8000/api/v1/search/?q=Python"
```

使用Python requests測試API

python

```python
import requests
import json

base_url = "http://localhost:8000/api/v1"

# 測試書籍列表
response = requests.get(f"{base_url}/books/")
print("書籍列表:", response.json())

# 測試建立書籍
book_data = {
    "title": "測試書籍",
    "author": "測試作者",
    "price": 299
}
response = requests.post(f"{base_url}/books/", json=book_data)
print("建立書籍:", response.json())

# 測試搜尋
response = requests.get(f"{base_url}/search/?q=Python")
print("搜尋結果:", response.json())
```

URL除錯技巧

使用Django shell除錯URL

python

```python
# 啟動Django shell
python manage.py shell

# 測試URL解析
from django.urls import reverse, resolve

# 正向解析（name -> URL）
print(reverse('books:detail', args=[1]))  # /books/1/
print(reverse('books:category', args=['programming']))  # /books/category/programming/

# 反向解析（URL -> view）
from django.test import RequestFactory
factory = RequestFactory()

resolver = resolve('/books/1/')
print(f"View: {resolver.func}")
print(f"Args: {resolver.args}")
print(f"Kwargs: {resolver.kwargs}")

# 測試URL模式
from django.core.management import execute_from_command_line
execute_from_command_line(['manage.py', 'show_urls'])  # 需要安裝django-extensions
```

常見錯誤和解決方法

```python
# 1. NoReverseMatch 錯誤
# 錯誤：{% url 'books:detail' %}
# 正確：{% url 'books:detail' book.id %}

# 2. 參數類型不匹配
# 錯誤：path('<str:book_id>/', views.book_detail)  # view期望int
# 正確：path('<int:book_id>/', views.book_detail)

# 3. URL模式順序問題
urlpatterns = [
    path('books/<int:pk>/', views.book_detail),  # 具體的模式要放在前面
    path('books/<str:action>/', views.book_action),  # 通用的模式放在後面
]

# 4. 命名空間衝突
# 使用app_name避免衝突
app_name = 'books'

# 5. include()路徑問題
# 錯誤：path('books', include('books.urls'))  # 缺少結尾斜線
# 正確：path('books/', include('books.urls'))
```

效能優化建議

URL設計最佳實踐

```python
# 1. 使用適當的參數類型
path('<int:pk>/', views.detail)  # 比 <str:pk> 更有效率

# 2. 避免過度複雜的正則表達式
# 較差：re_path(r'^books/(?P<category>\w+)/(?P<year>\d{4})/(?P<month>\d{2})/')
# 較好：path('books/<str:category>/<int:year>/<int:month>/')

# 3. 合理使用include()
# 將相關URL組織在一起，避免在根URL配置中放太多路由

# 4. URL快取
from django.views.decorators.cache import cache_page

urlpatterns = [
    path('books/', cache_page(60 * 15)(views.book_list)),  # 快取15分鐘
]
```

監控和日誌

```python
# settings.py
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': 'django_urls.log',
        },
    },
    'loggers': {
        'django.request': {
            'handlers': ['file'],
            'level': 'INFO',
            'propagate': True,
        },
    },
}

# 在view中記錄URL存取
import logging
logger = logging.getLogger(__name__)

def book_detail(request, book_id):
    logger.info(f"Accessing book detail for ID: {book_id}")
    # ... view logic
```

學習要點總結

核心概念掌握

1. **URL模式設計**：從簡單到複雜的URL結構
2. **參數處理**：路徑參數、查詢參數的正確使用
3. **命名空間**：避免URL名稱衝突的方法
4. **反向解析**：在Python和模板中的URL生成

RESTful API設計

1. **HTTP方法對應**：GET、POST、PUT、DELETE的正確使用
2. **資源設計**：RESTful URL結構的最佳實踐
3. **版本控制**：API版本管理策略
4. **錯誤處理**：適當的HTTP狀態碼使用

除錯和測試

1. **URL除錯工具**：Django shell和第三方工具
2. **測試策略**：URL測試的完整方法
3. **效能監控**：URL存取效能的監控方法
4. **日誌記錄**：URL存取日誌的配置

這些解答涵蓋了Django URLconf的核心概念和實際應用，從基礎到進階，提供了完整的學習路徑。