URL-Routing und Query-Parameter in Django

Diese Anleitung zeigt, wie man URL-Parameter und Query-Parameter in Django auslesen kann.

Beispiel 1: URL-Parameter

Schritt-für-Schritt-Anleitung: URL-Parameter in Django

1. **Projekt- und App-Vorbereitung**:
   - Stelle sicher, dass du bereits ein Django-Projekt und eine App erstellt hast. Falls noch nicht, erstelle eine App (z. B. `myapp`) mit:

     ```
     python manage.py startapp myapp
     ```

2. **App in den Projekteinstellungen registrieren**:
   - Öffne die Datei `settings.py` und füge deine App zur Liste der `INSTALLED_APPS` hinzu:

     ```
     INSTALLED_APPS = [
         ...,
         'myapp',
     ]
     ```

3. **URL-Routing einrichten**:
   - Erstelle eine Datei `urls.py` in deinem `myapp`-Ordner, falls noch nicht vorhanden, um die spezifischen Routen für die App zu verwalten.

4. **Beispiel-URL-Muster mit Parametern hinzufügen**:
   - In `myapp/urls.py` definieren wir eine Route, die einen Parameter in der URL erwartet:

     ```
     from django.urls import path
     from . import views

     urlpatterns = [
         path('greet/<str:name>/', views.greet_user, name='greet_user'),  # String-
     Parameter
         path('multiply/<int:a>/<int:b>/', views.multiply, name='multiply'),  # Integer-
     Parameter
     ]
     ```

   - Hier verwenden wir `<str:name>` und `<int:a>/<int:b>` als Platzhalter für URL-Parameter:
     - `<str:name>`: Extrahiert den `name`-Parameter als String.
     - `<int:a>` und `<int:b>`: Extrahieren die `a` und `b` Parameter als Ganzzahlen.

5. **URL-Parameter in Views auslesen**:
   - In `views.py` erstellen wir die View-Funktionen und lesen die Parameter aus der URL.

     ```
     from django.http import HttpResponse

     # View für die Begrüßung
     def greet_user(request, name):
         response = f"Hallo, {name}! Willkommen auf unserer Seite!"
     ```

```
    return HttpResponse(response)

# View für Multiplikation
def multiply(request, a, b):
    result = a * b
    response = f"Das Ergebnis von {a} * {b} ist {result}."
    return HttpResponse(response)
```

6. **Haupt-URL-Konfiguration**:
   - Verknüpfe die URLs der App mit der Haupt-URL-Konfiguration des Projekts. Öffne `myproject/urls.py` und füge einen `include`-Pfad hinzu:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myapp/', include('myapp.urls')),  # Verknüpft die URLs von myapp
]
```

7. **Server starten und testen**:
   - Starte den Entwicklungsserver:

```
python manage.py runserver
```

   - Besuche die folgenden URLs, um die Funktionen zu testen:
     - Begrüßung: `http://127.0.0.1:8000/myapp/greet/Alex/`
     - Multiplikation: `http://127.0.0.1:8000/myapp/multiply/5/3/`

8. **Warum, weshalb, wieso?**:
   - **URL-Parameter**: Durch die Platzhalter (`<str:name>`, `<int:a>`, etc.) in Django können URL-Parameter dynamisch in Views weitergeleitet werden.
   - **Modularisierung durch Apps**: Die App-spezifischen Routen in einer eigenen `urls.py` zu definieren und dann in die Haupt-URL-Konfiguration einzubinden, macht das Projekt übersichtlicher.

Beispiel 2: Query-Parameter

Schritt-für-Schritt-Anleitung: Query-Parameter in Django auslesen

1. **Voraussetzung: Projekt- und App-Struktur**
   - Stelle sicher, dass du ein Django-Projekt und eine App (z. B. `myapp`) erstellt und registriert hast.

2. **Route für die View einrichten**
   - In `myapp/urls.py` erstellen wir eine Route, die Query-Parameter verarbeiten kann:

```
from django.urls import path
from . import views

urlpatterns = [
    path('search/', views.search, name='search'),  # Route für die Suche
]
```

3. **Query-Parameter in der View verarbeiten**

- In `myapp/views.py` erstellen wir die View-Funktion `search`, die Query-Parameter aus der URL extrahiert und verarbeitet:

```
from django.http import HttpResponse

def search(request):
    # Extrahiert den Wert für das Keyword "q" aus der URL
    query = request.GET.get('q', '')
    sort_order = request.GET.get('sort', 'asc')  # Standardwert "asc", falls nicht
angegeben

    response_text = f"Suchergebnis für: {query}. Sortierung: {sort_order}."
    return HttpResponse(response_text)
```

4. **URLs konfigurieren**
   - In der Haupt-URL-Konfiguration (`myproject/urls.py`) binde die URLs der App ein:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('myapp/', include('myapp.urls')),
]
```

5. **Server starten und Query-Parameter testen**
   - Starte den Server:

     ```
     python manage.py runserver
     ```

   - Rufe die Seite im Browser auf, um die verschiedenen Query-Parameter zu testen:
     - `http://127.0.0.1:8000/myapp/search/?q=django&sort=desc`
     - `http://127.0.0.1:8000/myapp/search/?q=python`
     - `http://127.0.0.1:8000/myapp/search/`

6. **Erklärung der Funktionsweise**
   - **Warum Query-Parameter?** Query-Parameter sind hilfreich, um optionale und dynamische Daten an die URL anzuhängen, z. B. Suchbegriffe, Filter oder Sortierkriterien.
   - **Flexibilität durch `request.GET`**: Mit `request.GET` kann Django Query-Parameter einfach abfragen, ohne die URL-Struktur anpassen zu müssen.

Mit dieser Anleitung können deine Schüler lernen, wie sie flexible, datengetriebene URLs in Django nutzen, um interaktive Webanwendungen zu entwickeln.