Django Template Language: Step-by-Step Anleitung mit vollständigen Beispielen

Einführung

Django ermöglicht Entwicklern, dynamische Webseiteninhalte effizient zu erstellen. Die Django Template Language (DTL) verbindet Datenquellen mit statischem HTML und bietet dynamische Inhalte mit Platzhaltern, Tags und Filtern.

---

1. Einrichten des Projekts

Schritt 1: Django-Projekt erstellen

```
django-admin startproject dynamic_site
cd dynamic_site
python manage.py startapp menu_app
```

Schritt 2: App registrieren

In `settings.py`:

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

Schritt 3: Ordnerstruktur

Erstelle folgende Ordnerstruktur:

```
dynamic_site/
    menu_app/
        templates/
            menu_app/
                menu_card.html
```

---

2. Modelle: Datenquelle für dynamische Inhalte

Erstelle ein Modell `Menu`, um Daten wie Menüname und Preis zu speichern.

Code: `models.py`

```
from django.db import models

class Menu(models.Model):
    name = models.CharField(max_length=100)
    price = models.IntegerField()

    def __str__(self):
        return self.name
```

Migration

```
python manage.py makemigrations
python manage.py migrate
```

---

3. Views: Daten für das Template bereitstellen

Code: `views.py`

```
from django.shortcuts import render
from .models import Menu

def menu_view(request):
    menu_items = Menu.objects.all()
    context = {'menu': menu_items}
    return render(request, 'menu_app/menu_card.html', context)
```

---

4. URLs: View mit einer URL verknüpfen

Code: `urls.py`

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

urlpatterns = [
    path('menu/', views.menu_view, name='menu'),
]
```

---

5. Templates: Inhalte dynamisch anzeigen

Template: `menu_card.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Speisekarte</title>
</head>
<body>
    <h1>Unsere Speisekarte</h1>
    {% if menu %}
        <ul>
            {% for item in menu %}
                <li>{{ item.name }} - {{ item.price }} €</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>Keine Menüeinträge verfügbar.</p>
    {% endif %}
</body>
</html>
```

---

6. Variablen in Templates

Zugriff auf Variablen

```
<h1>Willkommen, {{ name }}</h1>
```

Dot-Notation für komplexe Daten

Dictionaries

```
<p>Name: {{ person.name }}</p>
<p>Beruf: {{ person.profession }}</p>
```

Listen

```
<p>Erstes Element: {{ my_list.0 }}</p>
```

---

7. Tags in Templates

Bedingungen: `{% if %}`

```
{% if user == "admin" %}
    <h1>Willkommen, Admin</h1>
{% else %}
    <h1>Willkommen, Gast</h1>
{% endif %}
```

Schleifen: `{% for %}`

Liste durchlaufen

```
<ul>
    {% for lang in langs %}
        <li>{{ lang }}</li>
    {% endfor %}
</ul>
```

Schleifen-Helper

```
<ul>
    {% for lang in langs %}
        <li>{{ forloop.counter }}: {{ lang }}</li>
    {% endfor %}
</ul>
```

---

8. Filters

Filters verändern die Darstellung von Variablen.

Beispiele

- `{{ name | upper }}` → Wandelt den Namen in Großbuchstaben um.
- `{{ name | lower }}` → Wandelt den Namen in Kleinbuchstaben um.
- `{{ name | title }}` → Kapitalisiert den Namen.

- `{{ items | length }}` → Gibt die Anzahl der Elemente zurück.
- `{{ my_list | first }}` → Gibt das erste Element der Liste zurück.
- `{{ my_list | last }}` → Gibt das letzte Element der Liste zurück.
- `{{ words | join:", " }}` → Verbindet die Wörter mit einem Komma.
- `{{ string | wordcount }}` → Zählt die Wörter im String.
- `{{ numbers | slice:"1:3" }}` → Gibt eine Teilliste zurück.

Django Template Filters: Praxisbeispiele

1. Beispiel-Dictionary in der View

Definiere in der View ein Dictionary, das alle benötigten Daten enthält, um die Filterbeispiele im Template darzustellen.

Code für `views.py`

```python
from django.shortcuts import render

def filter_examples_view(request):
    context = {
        "name": "John Doe",
        "items": ["apple", "banana", "cherry"],
        "my_list": ["first_item", "second_item", "third_item"],
        "words": ["Django", "Templates", "Filters", "are", "powerful"],
        "string": "Django Templates Filters are powerful",
        "numbers": [1, 2, 3, 4, 5, 6]
    }
    return render(request, 'filters_example.html', context)
```

---

2. HTML-Vorlage mit Filter-Beispielen

In der Vorlage werden alle Filter mit den Daten aus dem Dictionary angewendet. Jede Sektion zeigt ein spezifisches Beispiel.

Code für `filters_example.html`

```html
<!DOCTYPE html>
<html>
<head>
    <title>Django Template Filters</title>
</head>
<body>
    <h1>Django Template Filters Beispiele</h1>

    <section>
        <h2>1. Großbuchstaben (upper)</h2>
        <p>Original: {{ name }}</p>
        <p>Upper: {{ name | upper }}</p>
    </section>

    <section>
        <h2>2. Kleinbuchstaben (lower)</h2>
        <p>Original: {{ name }}</p>
        <p>Lower: {{ name | lower }}</p>
    </section>

    <section>
        <h2>3. Kapitalisierung (title)</h2>
        <p>Original: {{ name }}</p>
        <p>Title: {{ name | title }}</p>
    </section>
```

```
    <section>
        <h2>4. Länge einer Liste (length)</h2>
        <p>Liste: {{ items }}</p>
        <p>Länge: {{ items | length }}</p>
    </section>

    <section>
        <h2>5. Erstes Element (first)</h2>
        <p>Liste: {{ my_list }}</p>
        <p>Erstes Element: {{ my_list | first }}</p>
    </section>

    <section>
        <h2>6. Letztes Element (last)</h2>
        <p>Liste: {{ my_list }}</p>
        <p>Letztes Element: {{ my_list | last }}</p>
    </section>

    <section>
        <h2>7. Liste verbinden (join)</h2>
        <p>Wörter: {{ words }}</p>
        <p>Verbunden: {{ words | join:", " }}</p>
    </section>

    <section>
        <h2>8. Wortanzahl (wordcount)</h2>
        <p>String: "{{ string }}"</p>
        <p>Wörter: {{ string | wordcount }}</p>
    </section>

    <section>
        <h2>9. Liste slicen (slice)</h2>
        <p>Nummern: {{ numbers }}</p>
        <p>Slice [1:3]: {{ numbers | slice:"1:3" }}</p>
    </section>
</body>
</html>
```

---

3. Zusammenfassung

Diese Vorlage zeigt:

1. Die Verwendung von Django-Filtern.
2. Dynamisches Laden von Daten aus der View.
3. Strukturierte Abschnitte zur Präsentation im Unterricht.

- **URL-Konfiguration**: Verknüpfe die View mit einer URL.

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

urlpatterns = [
    path('filters/', filter_examples_view, name='filter_examples'),
]
```

Nutze diese Beispiele im Unterricht, um die Leistungsfähigkeit von Django Template Filters zu demonstrieren.

---

9. Erweiterte Features

Kommentare

```
{% comment "Hier ist ein Kommentar" %}
<p>Dies wird nicht angezeigt.</p>
{% endcomment %}
```

Template-Vererbung

Erstelle eine Basisvorlage (`base.html`):

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}Meine Seite{% endblock %}</title>
</head>
<body>
    <header>
        <h1>Willkommen</h1>
    </header>
    <main>
        {% block content %}{% endblock %}
    </main>
    <footer>
        <p>© 2024</p>
    </footer>
</body>
</html>
```

Erstelle eine abgeleitete Vorlage:

```
{% extends "base.html" %}

{% block title %}Über uns{% endblock %}

{% block content %}
<p>Wir sind ein modernes Unternehmen.</p>
{% endblock %}
```

---

10. Projektzusammenfassung

Ein komplettes Beispielprojekt, das alle obigen Punkte zusammenfasst.

Finales View

```
def menu_view(request):
    menu_items = Menu.objects.all()
    context = {'menu': menu_items}
    return render(request, 'menu_app/menu_card.html', context)
```

Finales Template

```
<!DOCTYPE html>
<html>
<head>
    <title>Speisekarte</title>
</head>
<body>
    <h1>Unsere Speisekarte</h1>
```

```
    {% if menu %}
        <ul>
            {% for item in menu %}
                <li>{{ item.name | upper }} - {{ item.price }} €</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>Keine Menüeinträge verfügbar.</p>
    {% endif %}
</body>
</html>
```

Zusammenfassung

In dieser Anleitung haben wir die Verwendung von:

1. Variablen,
2. Tags,
3. Filtern und
4. Template-Vererbung behandelt.

Nutze dieses Projekt als Grundlage, um dynamische Inhalte mit Django effektiv zu vermitteln.