Django Template und Model Aufgaben: Schritt-für-Schritt

Diese Aufgaben führen dich durch den Prozess der Erstellung einer Django-Webanwendung. Sie umfassen Template-Vererbung, das Arbeiten mit Modellen, Formularen und dynamischen Daten.

Aufgabe 1: Erstellung eines Grundgerüsts mit Template-Inheritance und Navigation

Ziel: Erstelle eine Django-Anwendung mit einer `base.html`, die Header, Navigation und Footer enthält. Verwende Template-Inheritance, um spezifische Seiten wie eine Startseite und eine Produktseite zu erstellen.

Anforderungen:

1. Erstelle ein Template `base.html` mit folgenden Bestandteilen:
    - **Header**: Enthält den Titel der Webseite.
    - **Navigation**: Links zu "Startseite" und "Produkte".
    - **Footer**: Enthält allgemeine Informationen.
2. Erstelle zwei Seiten:
    - `home.html`: Zeigt eine Willkommensnachricht.
    - `products.html`: Platzhalter für eine zukünftige Produktliste.
3. Nutze `{% block %}`, `{% include %}`, und Vererbung.

HTML-Grundgerüst für `base.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}My Website{% endblock %}</title>
</head>
<body>
    <header>
        <h1>My Website</h1>
        <nav>
            <ul>
                <li><a href="#">Startseite</a></li>
                <li><a href="#">Produkte</a></li>
            </ul>
        </nav>
    </header>

    <main>
        {% block content %}{% endblock %}
    </main>

    <footer>
        <p>© 2024 My Website. Alle Rechte vorbehalten.</p>
    </footer>
</body>
</html>
```

Aufgabe 2: Produkte erstellen und anzeigen

Ziel: Füge Produkte in der Datenbank hinzu und zeige sie in einer Produktliste auf der Seite `products.html` an.

Anforderungen:

1. Definiere ein Modell für ein Produkt.
2. Erstelle eine Datenbankmigration und füge Beispielprodukte hinzu.
3. Erstelle eine View-Funktion, die alle Produkte an die Seite `products.html` übergibt.
4. Zeige die Produktliste auf der Seite an. Nutze dafür:
    - Eine `{% for %}`-Schleife.
    - Einen Include für die Darstellung jedes Produkts.

HTML-Grundgerüst für `products.html`:

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

{% block title %}Produkte{% endblock %}

{% block content %}
<h2>Unsere Produkte</h2>
<div class="product-list">
    <!-- Hier sollen die Produkte angezeigt werden -->
</div>
{% endblock %}
```

Aufgabe 3: Produktdetails anzeigen

Ziel: Implementiere eine Detailansicht für Produkte. Erstelle eine Seite, die Informationen zu einem Produkt anzeigt, wenn ein Nutzer auf ein Produkt in der Liste klickt.

Anforderungen:

1. Füge in der Produktliste Links zu den Detailseiten hinzu.
2. Erstelle eine Detailseite `product_detail.html`, die:
    - Den Namen, die Beschreibung und den Preis eines Produkts anzeigt.
    - Das Erstellungsdatum formatiert ausgibt.

HTML-Grundgerüst für `product_detail.html`:

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

{% block title %}Produktdetails{% endblock %}

{% block content %}
<h2>Produkt: {{ product.name }}</h2>
<p>Beschreibung: {{ product.description }}</p>
<p>Preis: {{ product.price }} €</p>
<p>Erstellt am: {{ product.created_at }}</p>
{% endblock %}
```

Aufgabe 4: Produkte über ein Formular hinzufügen

Ziel: Erstelle ein Formular, mit dem Nutzer neue Produkte hinzufügen können. Zeige das Formular auf einer eigenen Seite und leite Nutzer nach dem Hinzufügen auf die Produktliste weiter.

Anforderungen:

1. Erstelle ein Formular für Produkte.
2. Zeige das Formular auf einer neuen Seite an.
3. Verarbeite die Daten und speichere das Produkt in der Datenbank.
4. Leite die Nutzer nach dem Speichern zurück zur Produktliste.

HTML-Grundgerüst für die Formularseite:

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

{% block title %}Produkt hinzufügen{% endblock %}

{% block content %}
<h2>Produkt hinzufügen</h2>
<form method="post">
    {% csrf_token %}
    <!-- Hier das Formular anzeigen -->
    <button type="submit">Speichern</button>
</form>
{% endblock %}
```

Aufgabe 5: Design und Bilder hinzufügen

Ziel: Verbessere das Design der Webseite und füge Produktbilder hinzu, die auf der Produktliste und der Detailseite angezeigt werden.

Anforderungen:

1. Füge dem Produktmodell ein Feld für ein Bild hinzu.
2. Zeige Bilder in der Produktliste und auf den Detailseiten an.
3. Erstelle eine CSS-Datei, um die Seiten optisch zu verbessern.

Beispiel-CSS für `styles.css`:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header, footer {
    background-color: #f4f4f4;
    padding: 10px;
    text-align: center;
}

.product-list {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
}

.product-item {
    border: 1px solid #ccc;
    padding: 10px;
    text-align: center;
}
```

HTML-Grundgerüst für die Produktliste mit Bildern:

```
{% for product in products %}
<div class="product-item">
    <img src="{{ product.image.url }}" alt="{{ product.name }}">
    <h3>{{ product.name }}</h3>
    <p>{{ product.price }} €</p>
    <a href="#">Details</a>
```

```
</div>
{% endfor %}
```

Hinweise

- Überlege selbstständig, wie du die Datenstruktur aufbaust und die Anforderungen umsetzt.
- Nutze Template-Vererbung, Filter, Schleifen und andere Konzepte, um dynamische und strukturierte Seiten zu erstellen.