31/10/2024

Django forms and form validations

1. Create a Simple Form and Handle It in a View

Objective: Create a form for collecting user feedback with fields for name, email, and message.

Tasks:

1, Create a new form class in forms.py called FeedbackForm.

2, Define fields for name, email, and message using Django form field types.

forms.py

```python
from django import forms


class FeedbackForm(forms.Form):
    name = forms.CharField(label='Name',max_length=100)
    email = forms.EmailField(label='Email')
    message = forms.CharField(label='Message', widget=forms.Textarea)

```

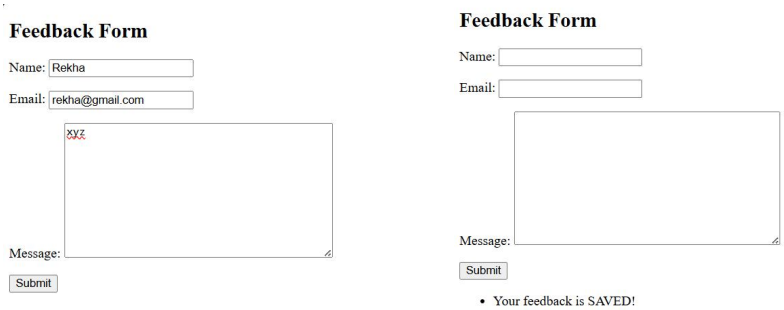
3, Create a view to render the form and handle form submission.

views.py

```python
from django.shortcuts import render, redirect
from .forms import FeedbackForm
from django.contrib import messages

def feedback_view(request):
    if request.method == 'POST':
        form = FeedbackForm(request.POST)
        if form.is_valid():
            messages.success(request, 'Your feedback is SAVED!')
            return redirect('feedback')
    else:
        form = FeedbackForm()
    return render(request, 'feedback.html', {'form': form})

```

urls.py

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

urlpatterns = [
    path('feedback/', feedback_view, name='feedback'),
]

```



2. Add Validation to Ensure Proper Data Entry

Objective: Implement both field-specific and form-wide validation.

Tasks:

1, Add a custom validator to check that the email is from a specific domain

forms.py

```python
from django import forms
from django.core.exceptions import ValidationError


def validate_email_domain(value):
    allowed_domains = [
        "@yahoo.com", "@google.com", "@example.com", "@hotmail.com", "@outlook.com",
        "@aol.com", "@msn.com", "@live.com", "@icloud.com", "@protonmail.com",
        "@zoho.com", "@mail.com", "@gmx.com", "@yandex.com", "@inbox.com",
        "@fastmail.com", "@hey.com", "@hushmail.com", "@tutanota.com", "@me.com"
    ]

    if not any(value.endswith(domain) for domain in allowed_domains):
        allowed_domains_str = ", ".join(allowed_domains)
        raise ValidationError(f"Enter an email address with a permitted domain: {allowed_domains_str}.")



class FeedbackForm(forms.Form):
    name = forms.CharField(label='Name',max_length=100)
    email = forms.EmailField(label='Email',validators=[validate_email_domain])
    message = forms.CharField(label='Message', widget=forms.Textarea)

```

2, Add form-wide validation to ensure that the message doesn’t contain offensive words.

```python
from django import forms
from django.core.exceptions import ValidationError


def validate_email_domain(value):
    allowed_domains = [
        "@yahoo.com", "@google.com", "@example.com", "@hotmail.com", "@outlook.com",
        "@aol.com", "@msn.com", "@live.com", "@icloud.com", "@protonmail.com",
        "@zoho.com", "@mail.com", "@gmx.com", "@yandex.com", "@inbox.com",
        "@fastmail.com", "@hey.com", "@hushmail.com", "@tutanota.com", "@me.com"
    ]

    if not any(value.endswith(domain) for domain in allowed_domains):
        allowed_domains_str = ", ".join(allowed_domains)
        raise ValidationError(f"Enter an email address with a permitted domain: {allowed_domains_str}.")

offensive_words = ['bloody', 'hell', 'in shit','damn','nigra']

def validate_no_offensive_words(value):
    for word in offensive_words:
        if word in value.lower():
            raise ValidationError(f"Your message contains offensive words. Please remove: {word}.")



class FeedbackForm(forms.Form):
    name = forms.CharField(label='Name',max_length=100)
    email = forms.EmailField(label='Email',validators=[validate_email_domain])
    message = forms.CharField(label='Message', widget=forms.Textarea, validators=[validate_no_offensive_words])

```

3. Handle Form Errors and Display Them in the Template

Objective: Display error messages directly under each form field.

Tasks:

1. Ensure that error messages are shown when the user submits invalid input.

Feedback Form

Name: Rekha
Email: rekha@yahoo.com
Message: The door creaked open bloody and we were in a long entrance hall. The dust on the carpet was visible and a thick film of it coated a hall table holding a dead, brittle succulent.

- Your message contains offensive words. Please remove: bloody.

Submit

2. Display non-field errors at the top of the form if any form-wide validation fails.

```
class FeedbackForm(forms.Form):
    name = forms.CharField(label='Name',max_length=100)
    email = forms.EmailField(label='Email',validators=[validate_email_domain])
    message = forms.CharField(label='Message', widget=forms.Textarea)

    def clean(self):
        cleaned_data = super().clean()
        message = cleaned_data.get('message')
        offensive_words = ['bloody', 'damn', 'in shit','bitch','nigga']
        if message:
            for word in offensive_words:
                if word in message.lower():
                    self.add_error(None, "Your message contains offensive words.")
                    break
        return cleaned_data
```

Feedback Form

Your message contains offensive words.
Name: Rekha
Email: rekha@gmail.com
Message: xyz bloody
Submit

4. Extend to ModelForm for Database Interaction

Objective: Convert the FeedbackForm to a ModelForm that saves feedback data directly to the database.

Tasks:

1, Create a model called Feedback in models.py with fields for name, email, and message.

models.py

```
from django.db import models

class Feedback(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    message = models.TextField()

    def __str__(self):
        return f"{self.name} - {self.email}"

```

2. Update the form to use ModelForm and save data on valid form submissions.

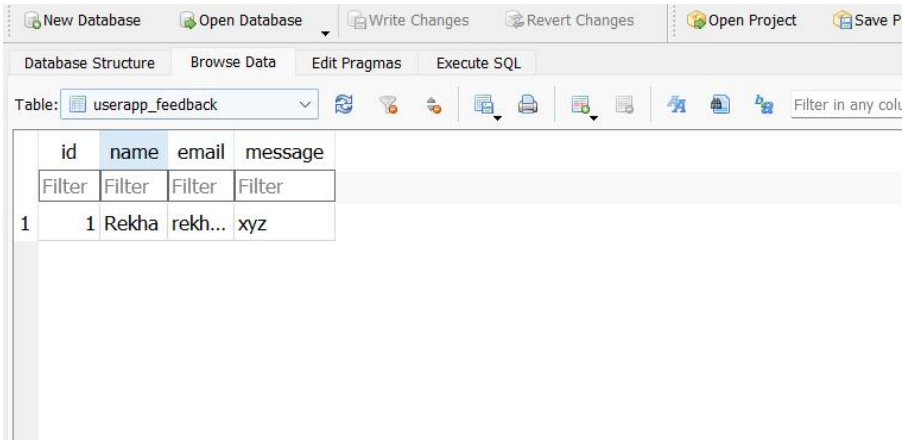
views.py

```python
from django.shortcuts import render, redirect
from .forms import FeedbackForm
from django.contrib import messages


def feedback_view(request):
    if request.method == 'POST':
        form = FeedbackForm(request.POST)
        if form.is_valid():
            feedback = form.save()
            messages.success(request, 'Your feedback is SAVED!')
            return redirect('feedback')
        else:

            for error in form.non_field_errors():
                messages.error(request, error)

    else:
        form = FeedbackForm()

    return render(request, 'feedback.html', {'form': form})

def index(request):
    return render(request, 'index.html')

```

forms.py

```python
from django import forms
from django.core.exceptions import ValidationError
from .models import Feedback

def validate_email_domain(value):
    allowed_domains = [
        "@yahoo.com", "@google.com", "@gmail.com", "@hotmail.com", "@outlook.com",
        "@aol.com", "@msn.com", "@live.com", "@icloud.com", "@protonmail.com",
        "@zoho.com", "@mail.com", "@gmx.com", "@yandex.com", "@inbox.com",
        "@fastmail.com", "@hey.com", "@hushmail.com", "@tutanota.com", "@me.com"
    ]

    if not any(value.endswith(domain) for domain in allowed_domains):
        allowed_domains_str = ", ".join(allowed_domains)
        raise ValidationError(f"Enter an email address with a permitted domain: {allowed_domains_str}.")

class FeedbackForm(forms.ModelForm):
    class Meta:
        model = Feedback
        fields = ['name', 'email', 'message']

    def clean(self):
        cleaned_data = super().clean()
        message = cleaned_data.get('message')
        offensive_words = ['bloody', 'damn', 'in shit','bitch','nigga']
        if message:
            for word in offensive_words:
                if word in message.lower():
                    self.add_error(None, "Your message contains offensive words.")
                    break
        return cleaned_data
```

Database



feedback.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Feedback Form</title>
</head>
<body>
    <h2>Feedback Form</h2>
    <form method="post">
        {% csrf_token %}
        {% if form.non_field_errors %}
            <div style="color: red;">
                {% for error in form.non_field_errors %}
                    <div>{{ error }}</div>
                {% endfor %}
            </div>
        {% endif %}
        <div>
            <label for="id_name">Name:</label>
            {{ form.name }}
            {% if form.name.errors %}
                <div style="color: red;">{{ form.name.errors }}</div>
            {% endif %}
        </div>
        <div>
            <label for="id_email">Email:</label>
            {{ form.email }}
            {% if form.email.errors %}
                <div style="color: red;">{{ form.email.errors }}</div>
            {% endif %}
        </div>
        <div>
            <label for="id_message">Message:</label>
            {{ form.message }}
        </div>
        <button type="submit">Submit</button>
    </form>
    {% if messages %}
        <ul>
            {% for message in messages %}
                <li>{{ message }}</li>
            {% endfor %}
        </ul>
    {% endif %}
</body>
</html>
```