CODING NOMADS

12) Users and User Authentication Lesson

Create a User Registration Form

6 min to complete · By Brandon Gigous

Contents

Even if you've built login functionality into a Python-Flask app, there's no way for you to make new accounts without going into a shell session. In this lesson, you'll make a registration page for new users to sign up for an account.

This registration form will require a lot of validation since you don't want a user to just make an account willy-nilly! That could cause a big mess in your database, or worse, let nefarious users gain access to your system or even cause damage with a well-placed string. No, you'll want your users to follow certain guidelines, like minimum lengths of usernames and only valid email addresses.

Registration Form

The new `RegistrationForm` will go in the `app/auth/views.py` file:

```
from wtforms import PasswordField
from wtforms.validators import Regexp, EqualTo, ValidationError

class RegistrationForm(FlaskForm):
    email = StringField('Email',
```

```
                            validators=[DataRequired(),
                                        Length(1,64),
                                        Email()])
    username = StringField('Username', validators=[
        DataRequired(),
        Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_.]*$', 0,
               'Usernames must have only letters, numbers, dots,
        )])
    password = PasswordField('Password', validators=[
        DataRequired(),
        EqualTo('password_confirm', message='Passwords do not match.
        )])
    password_confirm = PasswordField('Password (confirm):',
                                     validators=[DataRequired()])
    submit = SubmitField('Register')

    def validate_email(self, field):
        if Fan.query.filter_by(email=field.data).first():
            raise ValidationError('Email already registered.')

    def validate_username(self, field):
        if Fan.query.filter_by(username=field.data).first():
            raise ValidationError('Sorry! Username already in use.')
```

The first field is the `email` field, which is limited to 64 characters.

Next is `username`, which is similar but this has a `Regexp` validator to ensure that it matches a certain pattern. Particularly, you don't want the user to use `_`, `.`, or a number

be any combination of letters, numbers, `_`, or `.`. The next two arguments in the `Regexp` validator are the regular expression flags and then the message to display upon failure.

The next field is the `password` field, which can be anything the user desires. The only condition is that the `password` field is `EqualTo` the `password_confirm` field, hence the validator.

Whoa, what are those functions? It turns out that Flask-WTF will understand these as validators. That applies to any `FlaskForm` method that starts with `validate_`, where

the next part of the name is the field that is validated. Flask-WTF does some black magic behind the scenes to ensure that these validators are used for the `email` and `username` fields, respectively. Flask-WTF will even let you access the field through the `field` argument. Why would you want custom validators for these fields? Well, you wouldn't want duplicate values for the `email` or `username` columns in your "users" table. In other words, if an email is already used or a username already taken, you'll want to let the user know that! If either happens, a `ValidationError` is thrown, otherwise the field is successfully validated.

Add A Link To The Login Page

To allow your users to find the registration page, you'll want to add a link to your `login.html` template.

```
{# ... #}
<p>
New user?
<a href="{{ url_for('auth.register') }}">
Click here to register
</a>
</p>
{# ... #}
```

You can even add a link to the navbar if you want to!

Also, make a `auth/register.html` template to render your form. It can be near identical to the `login.html` template you made earlier.

Alright Flask cadet, you're almost out of the user authentication woods. Just one more thing to add: the registration view function. Next lesson!

Summary: How to Create a Flask Registration Form

- The new `RegistrationForm` will go in the `app/auth/views.py` file:

```
from wtforms import PasswordField
from wtforms.validators import Regexp, EqualTo, ValidationError

class RegistrationForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(),
                                    Length(1,64),
                                    Email()])
    username = StringField('Username', validators=[
        DataRequired(),
        Length(1, 64),
        Regexp('^[A-Za-z][A-Za-z0-9_.]*$', 0,
               'Usernames must have only letters, numbers, dot
        )])
    password = PasswordField('Password', validators=[
        DataRequired(),
        EqualTo('password_confirm', message='Passwords do not mat
        )])
    password_confirm = PasswordField('Password (confirm):',
                                     validators=[DataRequired()])
    submit = SubmitField('Register')

    def validate_email(self, field):
        if Fan.query.filter_by(email=field.data).first():
            raise ValidationError('Email already registered.')

    def validate_username(self, field):
        if Fan.query.filter_by(username=field.data).first():
            raise ValidationError('Sorry! Username already in use
```