Flask y htmx

geistesblitze2

Ernesto Rico Schmidt

PyCon Bolivia
Diciembre 2022

Contenido

¿Quién soy yo?

- Ernesto Rico Schmidt
- Ingeniero en Electrotecnia
- Usuario de Linux y Software Libre desde 1994
- Desarrollador remoto políglota de Software
  - Actualmente: Python y Go en el proyecto AURA
  - En el pasado: Objective-C, Swift, Java, Groovy, Kotlin, Scala, Javascript, Dart
  - En el futuro (algún día): Rust
- Traduje El módulo Python 3 de la semana al castellano

Flask

- Framework minimalista
- Broma de Abril en 2010 (Armin Ronacher)
- Un archivo zip (Jinja2 y Werkzeug)
- Un video: Un blog en cinco minutos
- Versión actual 2.2.2 (Agosto 2022)

geistesblitze

- Flask 0.10.1
- Aplicación simple para capturar ideas
- Flask-Bootstrap, Flask-Cors, Flask-Login, Flask-SQLAlchemy, Flask-WTF
- Dos modelos: User, Idea
- Siete rutas: add_idea, get_idea, get_ideas, index, login, logout, register
- Tres formularios: AddIdeaForm, LoginForm, RegisterForm
- Nueve plantillas: 403, 404, add_idea, base, idea, ideas, index, login, register

geistesblitze2

- Una aplicación de una página
- dos rutas adicionales: edit_idea, delete_idea
- una plantilla principal: index
- cuatro plantillas parciales (formularios): add_idea, edit_idea, login, register
- htmx

Flask

geistesblitze2

Modelos: Idea

```
from flask_login import UserMixin
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import check_password_hash, generate_password_hash

db = SQLAlchemy()

class Idea(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(128), unique=False)
    description = db.Column(db.Text(), nullable=True)

    user_id = db.Column(db.Integer, db.ForeignKey("user.id"))
```

geistesblitze2

Modelos: User

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True)
    password_hash = db.Column(db.String(128), unique=False)
    ideas = db.relationship("Idea", backref="user", lazy="dynamic")

    @property
    def password(self):
        raise AttributeError("password is not a readable attribute")

    @password.setter
    def password(self, password):
        self.password_hash = generate_password_hash(password)

    def verify_password(self, password):
        return check_password_hash(self.password_hash, password)
```

geistesblitze2

Formularios: RegistrationForm

```
from flask_wtf import FlaskForm
from wtforms import PasswordField, StringField, SubmitField, ValidationError
from wtforms.validators import DataRequired, EqualTo

from models import User

class RegistrationForm(FlaskForm):
    username = StringField("Username", validators=[DataRequired()])
    password = PasswordField(
        "Password",
        validators=[DataRequired()],
    )
    password2 = PasswordField(
        "Confirm Password",
        validators=[DataRequired(), EqualTo("password", message="Passwords must match.")],
    )
    submit = SubmitField("Register")

    def validate_username(self, field):
        if User.query.filter_by(username=field.data).first():
            raise ValidationError("Username already in use.")
```

geistesblitze2

Formularios: LoginForm e IdeaForm

```
class LoginForm(FlaskForm):
    username = StringField("Username", validators=[DataRequired()])
    password = PasswordField("Password", validators=[DataRequired()])
    submit = SubmitField("Login")

class IdeaForm(FlaskForm):
    name = StringField("Name", validators=[DataRequired()])
    description = StringField("Description", validators=[DataRequired()])
    submit = SubmitField("Save")
```

geistesblitze2

Rutas: register

```
from flask import Flask, redirect, render_template, url_for
from flask_login import LoginManager, current_user, login_user, logout_user, login_required
from forms import IdeaForm, LoginForm, RegistrationForm
from models import Idea, User, db

@app.route("/register", methods=["GET", "POST"])
def register():
    form = RegistrationForm()

    if form.validate_on_submit():
        user = User(username=form.username.data, password=form.password.data)

        db.session.add(user)
        db.session.commit()

        login_user(user, True)

        return redirect(url_for("index"))

    return render_template("partials/register.html", form=form)
```

geistesblitze2

Rutas: login y logout

```
@app.route("/login", methods=["GET", "POST"])
def login():
    form = LoginForm()

    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()

        if user is not None and user.verify_password(form.password.data):
            login_user(user, True)

            return redirect(url_for("index"))

    return render_template("partials/login.html", form=form)

@app.route("/logout")
def logout():
    logout_user()

    return redirect(url_for("index"))
```

geistesblitze2

Rutas: index

```
@app.route("/")
def index():
    if current_user.is_authenticated:
        ideas = Idea.query.filter_by(user=current_user).all()
    else:
        ideas = []

    return render_template("index.html", ideas=ideas)
```

geistesblitze2

Rutas: add_idea

```
@app.route("/add_idea", methods=["GET", "POST"])
@login_required
def add_idea():
    form = IdeaForm()

    if form.validate_on_submit():
        idea = Idea(name=form.name.data, description=form.description.data, user=current_user)

        db.session.add(idea)
        db.session.commit()

        return redirect(url_for("index"))

    return render_template("partials/add_idea.html", form=form)
```

geistesblitze2

Rutas: edit_idea

```
@app.route("/edit_idea/<int:id_>", methods=["GET", "POST"])
@login_required
def edit_idea(id_):
    idea = Idea.query.filter_by(id=id_).first()
    form = IdeaForm()

    if form.validate_on_submit():
        idea.name = form.name.data
        idea.description = form.description.data

        db.session.add(idea)
        db.session.commit()

        return redirect(url_for("index"))

    return render_template("partials/edit_idea.html", idea=idea, form=form)
```

geistesblitze2

Rutas: delete_idea

```
@app.route("/ideas/<int:id_>", methods=["DELETE"])
@login_required
def delete_idea(id_):
    idea = Idea.query.filter_by(id=id_).first()

    db.session.delete(idea)
    db.session.commit()

    return "", 200
```

htmx

htmx

htmx te brinda acceso a AJAX, CSS Transitions, WebSockets y Server Sent Events directamente en HTML, utilizando atributos, para que puedas crear interfaces de usuario modernas con la simplicidad y el poder del hipertexto.

htmx

- Cualquier elemento HTML puede generar un pedido HTTP
- Cualquier evento, no solo un click, puede generar un pedido HTTP
- Cualquier verbo HTTP puede ser usado: GET, POST, PUT, PATCH, DELETE
- Cualquier elemento puede ser remplazado por la respuesta a un pedido
- Atributos HTML
  - hx-get, hx-post, hx-put, hx-patch, hx-delete
  - hx-target: un selector CSS, `this`, `closest`, `find`
  - hx-swap: `innerHTML`, `outerHTML`, `beforebegin`, `afterbegin`, `afterend`, `afterbegin`
  - hx-confirm

geistesblitze2

Plantilla index 1/3

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Geistesblitze</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/missing-1.0.3.min.css') }}">
  <style>
    tr.htmx-swapping td {
      opacity: 0;
      transition: opacity 1s ease-out;
    }
  </style>
</head>
<!-- navbar -->
<body>
<h1><a href="/">Geistesblitze</a></h1>
<div id="idea_form"></div>
<!-- main --->
<script src="{{ url_for('static' , filename='js/htmx-1.8.4.min.js') }}"></script>
</body>
</html>
```

geistesblitze2

Plantilla index 2/3

```
<header class="navbar">
  <nav>
    {% if current_user.is_authenticated %}
      <a hx-get="/add_idea"
         hx-target="#idea_form"
         hx-swap="innerHTML">Add idea</a>
      <a href="/logout">Log out</a>
    {% else %}
      <a hx-get="/login"
         hx-target="#login_form"
         hx-swap="innerHTML">Login</a>
      <a hx-get="/register"
         hx-target="#registration_form"
         hx-swap="innerHTML">Register</a>
    {% endif %}
    <div id="login_form"></div>
    <div id="registration_form"></div>
  </nav>
</header>
```

geistesblitze2

Plantilla index 3/3

```
<div id="main">
  <table class="table">
    <tbody hx-target="closest tr"
           hx-swap="outerHTML swap:1s">
    {% for idea in ideas %}
      <tr>
        <td>
          <a hx-get="/edit_idea/{{ idea.id }}"
             hx-target="#idea_form"
             hx-swap="innerHTML">{{ idea.name }}</a>
        </td>
        <td>
          <button hx-confirm="are you sure?"
                  hx-delete="/ideas/{{ idea.id }}">delete</button>
        </td>
      </tr>
    {% endfor %}
    </tbody>
  </table>
</div>
```

¡Demo!

Hoy aprendí algo...

Hoy aprendí algo...

- Flask ofrece toda la funcionalidad para crear una aplicación web de una página
- No tiene sentido separar una aplicación tan simple en *backend* y *frontend*
- No se necesita una interfaz REST
- No se necesita Javascript
- ¡HTML y htmx son todo lo que se necesita!

Recursos

- Repositorio: `https://git.rico-schmidt.name/ernesto/geistesblitze2`
- Flask: `https://flask.palletsprojects.com/en/2.2.x/`
- htmx: `https://htmx.org`
- missing.css `https://missing.style`
- Mi sitio Web: `https://rico-schmidt.name`
- Mastodon: `https://hachyderm.io/@eigenwijsje`
- El Módulo Python 3 de la semana: `https://rico-schmidt.name/pymotw-3/`