**4 Szkolenie z Flaska**

**Flashowanie wiadomości**

**Czym jest flashing?**

Konieczne jest, by dobrze napisana aplikacja, była intuicyjna i łatwa w obsłudze. Ponadto powinna ona definiować jasny UX oraz bazować na skutecznym zawiadomieniu użytkownika w przypadku zmian w jej działaniu.   
  
Zapewnienie feedback-u użytkownikowi, w momencie gdy, np. proces logowania powiódł się prawidłowo, aplikacja utraciła połączenie z Internetem czy zmienił się status profilu, umożliwia **message flashing**.

Flashing daje Nam możliwość tworzenia wiadomości **na końcu jednego requestu** oraz jej odtwarzanie **na początku następnego**.

Przykładowo więc, w momencie gdy użytkownik zatwierdzi wpisane do formularza dane **przyciskiem Submit**, w zależności od tego, czy został on poprawnie uwierzytelniony, flashować będziemy stosowny komunikat. Następnie możemy go wyświetlić, na nowej już podstronie z informacją, np. **“You’ve been successfully logged in”.**

**Realizacja w kodzie.**

Chcąc zaimplementować system flashowania wiadomości, wystarczy tak naprawdę:

* Wywołać funkcję flash w odpowiedniej części logicznej Naszej aplikacji
* Odczytać zaflashowaną wiadomość z poziomu szablonu

Przykład: **Dodaj komunikaty do przykładu ze wcześniejszego szkolenia, tak aby wyświetlały się powiadomienia:***“You have been successfully logged in”****,*** *“You have been successfully logged out”****,*** *“Already logged in!”***w odpowiednich sytuacjach (np. gdy użytkownik będzie chciał się ponownie zalogować przed wylogowaniem).**

Aby zrealizować powyższe założenia, wystarczy tak naprawdę zaimportować funkcję **flash**, dodać proces flashowania wiadomości do stworzonych funkcji logout, dashboard i login oraz na końcu zmodyfikować nieco szablon bazowy base.html.

| from flask import render\_template, redirect, url\_for, Blueprint, request, session, flash ### 0  login\_blueprint = Blueprint("login", \_\_name\_\_) dashboard\_blueprint = Blueprint("dashboard", \_\_name\_\_) logout\_blueprint = Blueprint("logout", \_\_name\_\_)   @login\_blueprint.route('/', methods=["POST", "GET"]) def login():  if request.method == "POST":  session.update({"nick" : request.form['nickname']})  flash("You've been successfully logged in.") ### 1  elif request.method == "GET" and "nick" not in session:  return render\_template("login.html")  elif request.method == "GET" and "nick" in session: ### 2  flash("Already logged in!") ### 2.1  return redirect(url\_for("dashboard.dashboard"))   @logout\_blueprint.route('/logout') def logout():  if "nick" in session:  session.pop("nick")  flash("You have been logged out!") ### 3  else: ### 4   flash("You are not logged in!") ### 4.1   return redirect(url\_for("login.login"))   @dashboard\_blueprint.route('/dashboard') def dashboard():  if "nick" in session:  return render\_template("dashboard.html", nickname=session["nick"])  else:  flash("You are not logged in!") ### 5  return redirect(url\_for("login.login")) |
| --- |

Plik main.py

| ...  </**li**>  </**ul**>  </**div**>  </**div**> </**nav**>  {% with messages = get\_flashed\_messages() %}  {% for msg in messages %}  <**p**>{{ msg }}</**p**>  {% endfor %} {% endwith %}  {% block content %} {% endblock %}  ... |
| --- |

*Plik base.html*

**Wyjaśnienie:**

flash(wiadomość) - dodanie odpowiedniego komunikatu do bufora flashującego

**Plik base.html:**

- {% with messages = get\_flashed\_messages() %} - umożliwia pobranie wszystkie aktualnie zaflashowanych wiadomości

- {% for msg in messages %} - przechodzenie po każdej pojedynczej wiadomości…

- <p> {{ msg }} </p> - ...i jej wyświetlanie

**Kategorie wiadomości**

Rozwijając nieco funkcjonalność flashowania wiadomości, okazuje się, że możemy dodawać do wiadomości, tzw. **kategorie**.

Istnieje, rzecz jasna, pełna dowolność w tworzeniu własnych grup wiadomości, dlatego też w przykładzie, który rozważymy, wyodrębnimy 2 rodzaje powiadomień:

* wiadomości mówiące o poprawnym wyborze (**success**)
* wiadomości, które powstają gdy dana operacja nie może zostać przeprowadzona (**warning**)

Aby utworzyć wiadomości według konkretnej kategorii, należy zastosować zapis:

| flash(wiadomość, kategoria) |
| --- |

I tak po zmienieniu pliku main.py, otrzymamy m.in:

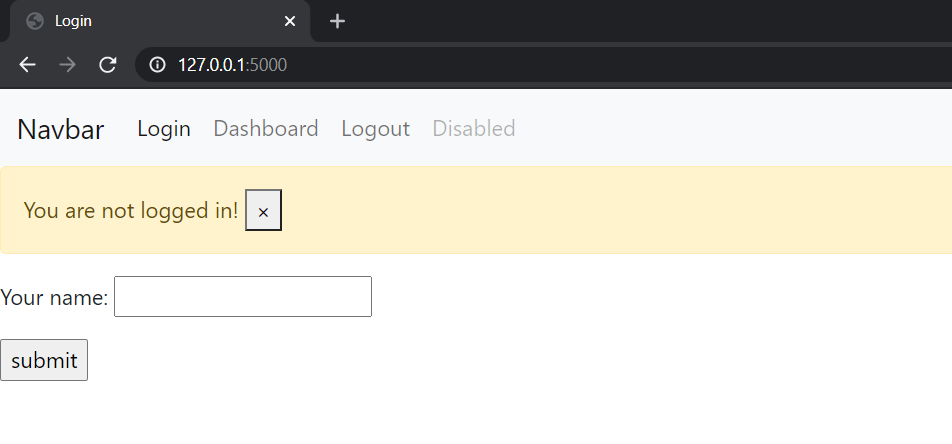
| flash("You've been successfully logged in.", "success")  flash("Already logged in!", "warning") |
| --- |

**Po co dodawać kategorie?**

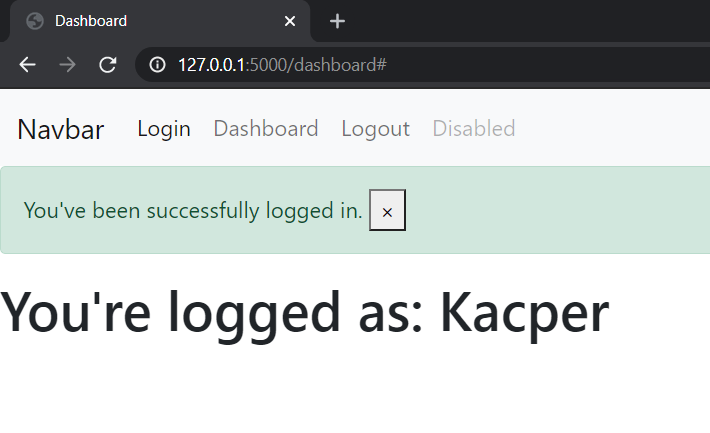
Dzięki odpowiedniemu grupowaniu powiadomień, możliwe jest, np. **łatwe formatowanie wyświetlanego tekstu w zależności od kategorii, do której należy** (np. kolorem czerwonym oznaczać będziemy errory, a kolorem zielonym wiadomości typu success).

Zmodyfikujmy nieco szablon base.html, aby dostosować go pod rodzaj wiadomości.

| {% with messages = get\_flashed\_messages(with\_categories=true) %}  {% for category, message in messages %}  <**div** class="alert alert-{{ category }} alert-dismissible fade show" role="alert">  <**span**>{{ message }}</**span**>  <**button** type="button" class="close" data-dismiss="alert" aria-label="Close">  <**span** aria-hidden="true">&times;</**span**>  </**button**>  </**div**>  {% endfor %} {% endwith %} |
| --- |



*Komunikat z kategorii* ***warning***



*Komunikat z kategorii* ***success***