Python średnio zaawansowany

Dzień 16

Blok nr 5:

Aplikacja webowa

AGENDA

- Formularze internetowe
- CRUD
- Cross site request forgery
- Tworzenie i walidacja formularzy
- ORM dla Flask

Formularze internetowe

Formularze internetowe

Formularze internetowe pozwalają na interakcję z aplikacją internetową, dzięki nim można:

- tworzyć, (-> create)
- edytować, (-> update)
- usuwać (-> delete)

zasoby przechowywane w systemie.

Formularze internetowe

Przykłady:

- formularz rejestracji nowego użytkownika, (-> create)
- formularz do wypisania się z newslettera, (-> update lub delete)
- formularz zmiany hasła (-> update)

Atrybuty formularza internetowego

Definiując formularz internetowy w HTML należy pamiętać o dwóch ważnych atrybutach *method* oraz *action*:

```
<form method='POST' action='/login'>
    ...
</form>
```

Atrybuty formularza internetowego

method - określa metodę HTTP jaka zostanie wykorzystana przy wysyłaniu formularza

action - adres URL, którego zostanie przesłany formularz; jeżeli został pominięty wówczas adres URL będzie dokładnie ten sam jak strona na, której znajduje się formularz

CRUD

CRUD

Skrót **CRUD** (Create, Read, Update, Delete) często jest wykorzystywany do określenia funkcjonalności aplikacji (lub jej części); oznacza to że funkcjonalność skupia się wokół manipulacji danymi (przetwarzanym modelem) np. panel administratora do zarządzania kontami użytkowników.

Na dzisiejszych slajdach nie oznaczaliśmy operacji „Read”; to podstawowa funkcjonalność umożliwiająca wypełnienie danymi formularza edycji (U) lub usuwania (D).

Cross-site request forgery

Metoda ataku CSRF

Ofiarami CSRF stają się użytkownicy nieświadomie przesyłający do serwera spreparowane żądania przez osoby o wrogich zamiarach.

Celem hackera jest wykorzystanie uprawnień ofiary do wykonania operacji w przeciwnym razie wymagających jej zgody.

Metoda ataku CSRF

Atak ma na celu skłonić użytkownika zalogowanego do serwisu internetowego do tego, aby uruchomił on odnośnik, którego otwarcie wykona w owym serwisie akcję, do której atakujący nie ma domyślnie dostępu.

Metoda ataku CSRF

Szczególnie podatne na ten atak są aplikacje webowe, które wykonują żądania przesłane przez zalogowanych użytkowników bez autoryzacji konkretnej akcji.

Osoba uwierzytelniona na podstawie ciasteczka zapisanego w przeglądarce może nieświadomie wysłać żądanie HTTP do serwera, który jej ufa i wykona niepożądaną akcję w jej imieniu(!)

Obrona przed CSRF

- ***Przesłanie*** (przez serwer) ciasteczka służącego do uwierzytelnienia i porównanie go z wartością ***odesłaną*** (przez klienta) w nagłówku żądania HTTP z tą zapisaną po stronie serwera. Metoda ta opiera swoje bezpieczeństwo na zasadzie ***same origin policy***, która gwarantuje, że wartość ciasteczka uznawana jest jedynie dla skryptów pochodzących z oryginalnej strony.
- Hasła jednorazowe
- Krótki okres ważności zalogowania i dopuszczalny czas bezczynności
- Uzupełnienie kodu stron HTML (w tym formularzy) o tokeny CSRF, na tej samej zasadzie co ciasteczka

Tworzenie i walidacja formularzy

Tworzenie i walidacja formularzy

Moduł WTForms – dojrzały, niezależny od innych frameworków, moduł dla języka Python do obsługi formularzy webowych:

- generowanie kodu HTML dla obiektów formularzy
- obsługa zwróconych danych

https://wtforms.readthedocs.io/en/stable/

Tworzenie i walidacja formularzy

Moduł Flask-WTF realizuje integracje Flaska z WTForms:

- Ułatwia budowanie i obsługę formularzy (parsowanie danych)
- Implementuje tokeny CSRF w formularzach
- Umożliwia zabezpieczenie formularzy przy pomocy reCAPTCHA
- Umożliwia upload plików

https://flask-wtf.readthedocs.io/en/stable/

Tworzenie i walidacja formularzy

Zasada działania:

- Każdy formularz to obiekt klasy Form*. Reprezentacja -> HTML
- Każde pole obiektu odpowiada polu formularza; odpowiedniość typów
- Każde pole ma przypisany walidator
- Funkcja w widoku otrzymuje dane z przeglądarki i waliduje dane
- Współdzieli z Flaskiem sekretny klucz do zabezpieczenia sesji

* widzisz podobieństwo do SQLAlchemy, gdzie każda tabela to też klasa?

Tworzenie i walidacja formularzy

Gdzie znajdę klucz?

```
app.config['SECRET_KEY'] = 'losowa_i_sekretna_wartosc'
```

ORM dla Flask

Flask współpracuje z RDBMS

Moduł Flask-SQLAlchemy realizuje wsparcie SQLAlchemy dla Flaska:

- to dodatkowa warstwa pośrednicząca (wydajność vs użyteczność!)
- umożliwia podejście ORM i SQL
- upraszcza użycie SQLAlchemy (konfiguracja, tworzenie/niszczenie połączeń i sesji, zwłaszcza dla sesji HTTP)
- łatwiejsze wydawanie zapytań

Flask współpracuje z RDBMS

- dostarcza metody wspomagające (np. paginacja)
- dostarcza metodę apply_driver_hacks (wada i zaleta)
- wymaga dostosowania kodu SQLAlchemy

http://flask-sqlalchemy.pocoo.org/2.3/

Różnice i uproszczenia

Najważniejsze z różnic/uproszczeń:

- nazwa **klasy** może być nazwą **tabeli** (nie ma obowiązkowego pola `__tablename__`)
- klasa reprezentująca tabelę dziedziczy po SQLAlchemy.model (nie po declarative_base). Model to „skonfigurowany” declarative_base
- zapytania są prostsze

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
```

Jak zmigrować model SQLAlchemy -> Flask-SQLAlchemy?

1. Zmień klasę nadrzędną:

```
Base = declarative_base()
class Kampanie(Base):
```

```
db = SQLAlchemy(app)
class Kampanie(db.Model):
```

2. Nie usuwaj deklaracji nazw tabel (`__tablename__`) jeśli nie musisz

Jak zmigrować model SQLAlchemy -> Flask-SQLAlchemy?

3. Pozostaw importy* definicji typów danych z sqlalchemy i sqlalchemy.orm lub przemianuj prefiksy typów na (przykładowy) prefiks db:

```
from sqlalchemy import Column, Integer, DateTime

from sqlalchemy.orm import relationship
```

lub

```
data_oferty = Column(DateTime)  ⟶  data_oferty = db.Column(db.DateTime)
```

*Flask-SQLAlchemy to nakładka na SQLAlchemy

Dzięki!