Kurs rozszerzony języka Python

Środowisko Django, cz. 3

Marcin Młotkowski

1 lutego 2022

Plan wykładu

Plan wykładu

Autentykacja

django oferuje domyślnie

- system użytkowników;
- system uprawnień;
- formularze i widoki do logowania.

Użytkownicy

Utworzenie użytkowników

Z panelu administracyjnego

Użytkownicy

Utworzenie użytkowników

Z panelu administracyjnego

podstawowe atrybuty modelu User

- username
- password
- email
- first_name, last_name

Hasła

Hasła są przechowywane w postaci hasza, z losową solą i informacją o algorytmie haszowania.

Zmiana hasła

```
user = User.objects.get(...)
user.set_password('noweHaslo')
user.save()
```

Weryfikacja użytkownika

Autentykacja

```
from django.contrib.auth import authenticate
user = authenticate(username='mm', password='123456')
if user is not None:
    ...
```

Weryfikacja użytkownika

Autentykacja

```
from django.contrib.auth import authenticate
user = authenticate(username='mm', password='123456')
if user is not None:
    ...
```

Weryfikacja

```
if request.user.is_authenticated():
    # zrób coś
else :
    # anonimowy użytkownik
```

logowanie

Autentykacja tylko weryfikuje, ale nie tworzy sesji dla użytkownika.

logowanie

Autentykacja tylko weryfikuje, ale nie tworzy sesji dla użytkownika.

```
from django.contrib.auth import authenticate, login

def login_view(request):
    username = request.POST['username']
    password = request.POST['password']
    user = authenticate(username=username, password=password)
    if user is not None:
        if user.is_active:
            login(request, user)
            # Przekieruj na odpowiednią stronę.
        else :
            # informacja o zablokowanym koncie
        else :
            # Zgłoś błąd logowania.
```

Wylogowanie

```
from django.contrib.auth import logout
def logout_view(request):
    logout(request)
```

Rozwiązanie proste

```
from django.shortcuts import redirect

def tajne_view(request):
    if not request.user.is_authenticated():
        return redirect('/login/?next=%s' % request.path)
```

Inne rozwiązanie

wersja z dekoratorem

```
from django.contrib.auth.decorators import login_required

@login_required
def tajne_view(request):
```

W razie braku autentykacji przekierowuje na stronę określoną w `settings.LOGIN_URL`.

Dodatkowe domyślne mechanizmy

- system grup użytkowników;
- system uprawnień.

Na koniec

Czego nie ma w standardowym django

- sprawdzanie siły hasła
- wsparcie dla innych systemów użytkowników;
- ograniczenie prób nieudanego logowania.

Plan wykładu

Rodzaje ataków

- *Cross site scripting (XSS)*: osadzenie na stronie obcego kodu (np. JavaScript);
- *Cross site request forgery (CSRF)*: próba podstępnego wymuszenia wykonania akcji przez zalogowanego użytkownika;
- *SQL injection*: wstrzyknięcie obcego kodu do zapytania SQL;
- *Clickjacking*: Oryginalna strona jest w sposób ukryty osadzana jako ramka na stronie na złośliwym serwerze.

HTML escaping

Przykład szablonu

```
Hello, {{ name }}
```

gdzie zmienna `name` ma wartość

```
Hello, <script>alert('hello')</script>
```

Remedium: HTML escaping

Zamiana tagów html w szablonach:
< na `<`
> na `>`
' na `&#39;`
& na `&`

Wsparcie Django:
zawsze wyświetlane wartości zmiennych w szablonach są konwertowane (chyba że jawnie zezwolimy na wyświetlanie wprost).

Cross site scripting (XSS)

Ktoś rejestruje się jako użytkownik

```
<img src="http://zapisy/admin/addUser?login=devil&pass=1234
```

Obrona

Dodanie do formularzy ukrytego pola z losowym tokenem:

```
<form action="." method="post">{% csrf_token %}
```

i zabezpieczenie widoku

```
from django.views.decorators.csrf import csrf_protect
from django.shortcuts import render

@csrf_protect
def widok(request):
```

SQL injection

Wyszukiwarka na stronie:

```
SELECT * FROM some_table WHERE title LIKE '%{query}%';
```

SQL injection

Wyszukiwarka na stronie:

```
SELECT * FROM some_table WHERE title LIKE '%{query}%';
```

Atakujący "szuka": "'; DELETE FROM some_table"

SQL injection

Wyszukiwarka na stronie:

```
SELECT * FROM some_table WHERE title LIKE '%{query}%';
```

Atakujący "szuka": "'; DELETE FROM some_table"

```
SELECT * FROM some_table WHERE title LIKE '%%';
    DELETE FROM some_table;
```

SQL injection

Wyszukiwarka na stronie:

```
SELECT * FROM some_table WHERE title LIKE '%{query}%';
```

Atakujący "szuka": "'; DELETE FROM some_table"

```
SELECT * FROM some_table WHERE title LIKE '%%';
    DELETE FROM some_table;
```

Obrona: użycie dedykowanych funkcji do zapytań zamiast zapytań „surowych":

```
cursor.execute(
        "SELECT * FROM some_table WHERE title LIKE '%?%'",
        [request.GET['q']]
        )
```

Clickjacking

Opis ataku

Oryginalna strona jest w sposób ukryty osadzana jako ramka na stronie na złośliwym serwerze.

Obrona

Zakazanie osadzania strony poprzez dodaniu taga `X-Frame-Options: deny` w odpowiedzi serwera:

```
X_FRAME_OPTIONS = 'DENY'
```

Plan wykładu

Hosting Django

- `www.heroku.com`: hobbistyczne projekty: free, obsługa Ruby, Java, Scala, Python, Django, ...
- `www.pythonanywhere.com`: od 0$/miesiąc;
- `www.webfaction.com`: Django, PHP, Perl, Rails, ...
- `https://www.linode.com/`: Linux na wirtualnej maszynie

Hosting Django

- `www.heroku.com`: hobbistyczne projekty: free, obsługa Ruby, Java, Scala, Python, Django, ...
- `www.pythonanywhere.com`: od 0$/miesiąc;
- `www.webfaction.com`: Django, PHP, Perl, Rails, ...
- `https://www.linode.com/`: Linux na wirtualnej maszynie

http://djangohosting.com/

Strona z odnośnikami do Djangowych hostingów z komentarzami.

Plan wykładu