

# *DJANGO*

**DANE OD UŻYTKOWNIKA**

# PLAN

Metody przekazywania danych od użytkownika:

- w adresie url
  - wbudowane konwertery funkcji path
  - wyrażenia regularne - funkcja re\_path
  - własne konwertery funkcji path
- metoda GET
- metoda POST
- formularze
- ciasteczka

Bezpieczeństwo aplikacji sieciowych

- Atak typu phishing
- Podatność CSRF



*ALWAYS REMEMBER TO  
SANITIZE YOUR INPUT AND  
ESCAPE YOUR OUTPUT*

*PRZEKAZYWANIE  
DANYCH W ADRESIE URL  
(ENDPOINT)*

# *DANE PRZEKAZYWANE W URL*

Django udostępnia trzy metody do pobierania danych z adresu url:

1. Wbudowane konwertery funkcji path
2. Wyrażenia regularne w funkcji re\_path (patrz dodatki na końcu tej prezentacji)
3. Własne konwertery funkcji path (patrz dodatki na końcu tej prezentacji)

*WBUDOWANE  
KONWERTERY  
FUNKCJI PATH*

Project

```
urls.py ×  
from django.urls import path  
  
from hello_app import views  
  
urlpatterns = [  
    path('hello-world/<str:name>',  
        views.hello_world  
),
```

DB Browser

```
hello_world.html ×  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Powitanie</title>  
</head>  
<body>  
    Cześć, {{ username }}!  
</body>  
</html>
```

Favorites

1

```
views.py ×  
from django.shortcuts import render  
from datetime import datetime  
  
def hello_world(request, name):  
    return render(  
        request,  
        'hello_app/hello_world.html',  
        context={  
            'username': name,  
        })
```

2



Cześć, adam!

3

## WBUDOWANE KONWERTERY FUNKCJI PATH

### str

Dowolny, niepusty ciąg znaków z wyłączeniem znaku '/', np. 'inactive'.  
Przekazywany do widoku jako łańcuch znaków (str).  
Odpowiednik regex: '[^/]+'.

### int

Dodatnia wartość całkowita większa lub równa 0, np. '7463'.  
Przekazywany do widoku jako liczba całkowita (int).  
Odpowiednik regex: '[0-9]+' lub '|\d|+'.

### slug

Ciąg znaków, który może składać się z liter, cyfr oraz znaków '-' i '\_', np. 'czterna-torebka-z-naszywka'.  
Przekazywany do widoku jako łańcuch znaków (str).  
Odpowiednik regex: '[-a-zA-Z0-9\_]+'

### uuid

Ciąg znaków odpowiadający sformatowanemu UUID (z kreskami i małymi literami), np. 'd8a6f1b2-adca-4b11-ad42-f99d8ffccc26'.  
Przekazywany do widoku jako instancja UUID (uuid.UUID).  
Odpowiednik regex: '[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}'

### path

Dowolny niepusty ciąg znaków włączając znak '/', np. 'path/to/file/pdf'.  
Przekazywany do widoku jako łańcuch znaków (str).  
Odpowiednik regex: '.+'

dj URL dispatcher | Django document +

← → 🔍 https://docs.djangoproject.com/en/3.2/topics/http/urls/#path-converters

## Path converters

The following path converters are available by default:

- **str** - Matches any non-empty string, excluding the path separator, '/'. This is the default if a converter isn't included in the expression.
- **int** - Matches zero or any positive integer. Returns an **int**.
- **slug** - Matches any slug string consisting of ASCII letters or numbers, plus the hyphen and underscore characters. For example, **building-your-1st-django-site**.
- **uuid** - Matches a formatted UUID. To prevent multiple URLs from mapping to the same page, dashes must be included and letters must be lowercase. For example, **075194d3-6885-417e-a8a8-6c931e272f00**. Returns a **UUID** instance.
- **path** - Matches any non-empty string, including the path separator, '/'. This allows you to match against a complete URL path rather than a segment of a URL path as with **str**.

---

## Registering custom path converters

For more complex matching requirements, you can define your own path converters.

A converter is a class that includes the following:

- A **regex** class attribute, as a string.
- A **to\_python(self, value)** method, which handles converting the matched string into the type that should be passed to the view function. It should raise **ValueError** if it can't convert the given value. A

WBUDOWANE  
KONWERTERY  
FUNKCJI PATH

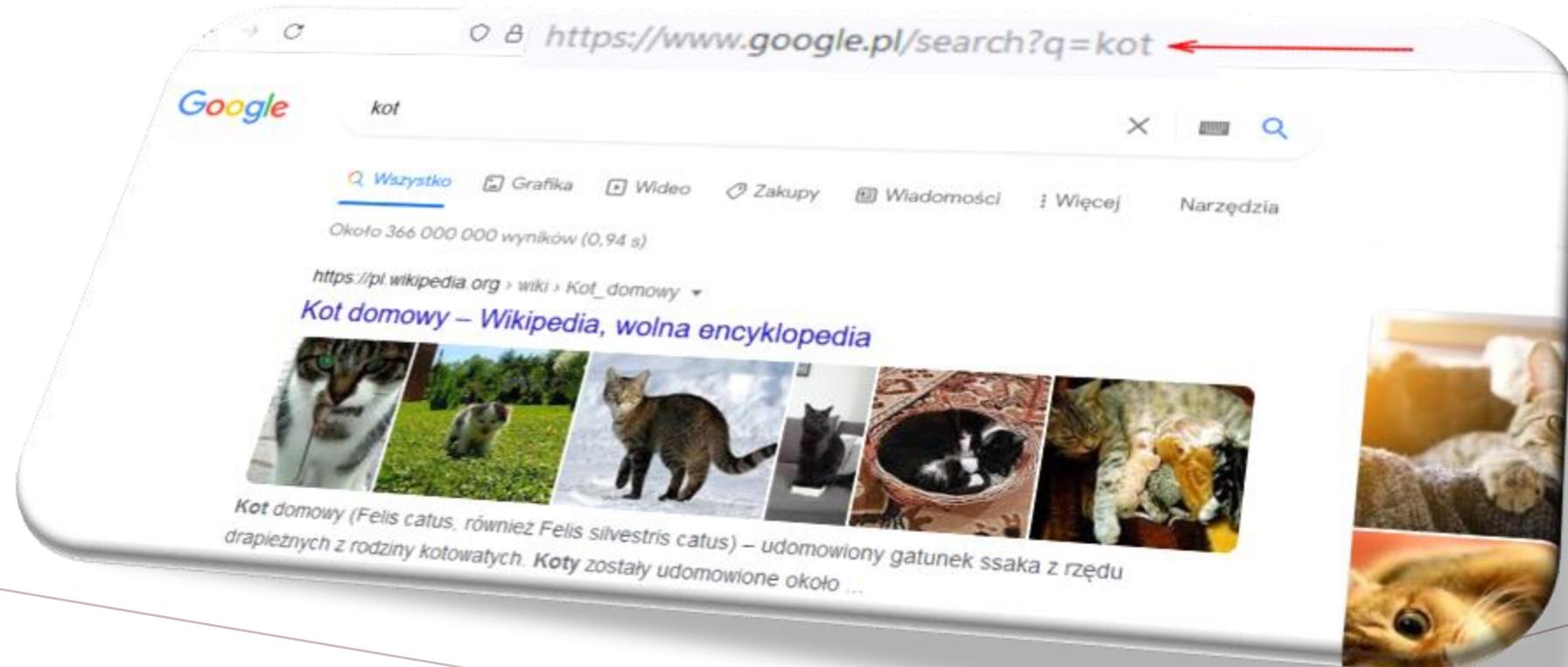


*METHODY HTTP*

# *METODA GET*

# DANE GET

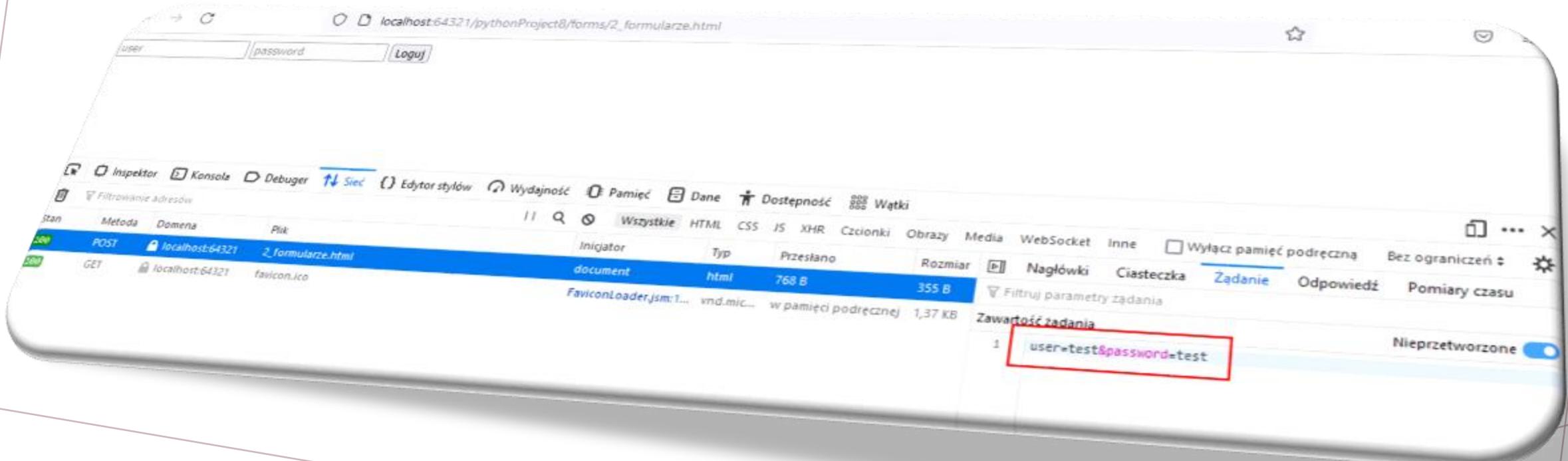
Dane przekazywane są w **adresie URL**, w postaci tzw. **parametrów GET**. Parametry GET **NIE** budują endpointu. Są dodatkowymi informacjami umieszczonymi **ZA** endpointem.



*METODA POST*

# DANE POST

Dane przekazywane są w ciele żądania, w postaci tzw. parametrów POST.



# *METODA GET VS METODA POST*

*Metodę POST uważa się za bezpieczniejszą od metody GET. W związku z tym przyjęto się w sytuacjach gdy przesyłane dane zmieniają stan bazy (tj. kiedy serwer po odebraniu danych wykonuje jedną z operacji niebezpiecznych - **create/update/delete**) wysyłać te dane metodą POST. Przyjęcie takiego założenia niesie ze sobą szereg konsekwencji. Najważniejsze z nich to ochrona przed podatnością csrf oraz przekierowania. Oba zagadnienia zostaną omówione w dalszej części.*

*Metodę GET jako mniej bezpieczną wykorzystuje się tylko do przesyłania danych, na podstawie których wykonywane są bezpieczne operacje na bazie, tj. takie które nie zmieniają stanu bazy (**read – odczyt**).*



*METHODY HTTP W  
DJANGO*

# *METODY HTTP W DJANGO*

*Django przechowuje informacje o tym jaką metodą zostało przesłane zapytanie na serwer w atrybucie `method` obiektu `request`*

*`request.method`*

*Wartość atrybutu to string z nazwą metody, np. "POST", "GET". Sprawdzenie realizujemy za pomocą prostego porównania, np.*

*`if request.method == "POST":`*

*...*

test\_08 &gt; formapp &gt; views.py

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, DB Navigator, and Help. The current file is 'views.py' under the 'test\_08' project. The code in 'views.py' is:

```
from django.shortcuts import HttpResponse

def index(request):
    if request.method == "GET":
        return HttpResponse("Zapytanie GET")

    elif request.method == "POST":
        return HttpResponse("Zapytanie POST")
```

A red rectangular box highlights the conditional logic from line 5 to line 9. The code editor has a light gray background with syntax highlighting. The left sidebar shows the project structure with 'formapp' selected, containing files like \_\_init\_\_.py, asgi.py, settings.py, urls.py, wsgi.py, and form.html. Below the project tree is a terminal window showing session output.

sessions.

Run 'python manage.py migrate' to apply them.

May 01, 2022 - 08:47:47

Django version 4.0.4, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*METODA GET  
(DJANGO)*

test\_08 &gt; formapp &gt; views.py

Project

Commit

Pull Requests

DB Browser

Structure

Bookmarks

```
from django.shortcuts import HttpResponse

def index(request):
    if request.method == "GET":
        return HttpResponse("Zapytanie GET")

    elif request.method == "POST":
        return HttpResponse("Zapytanie POST")
```

Microsoft Windows [Version 10.0.19044.1645]  
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

```
C: |Users|jerem>curl http://127.0.0.1:8000/
Zapytanie GET
```

```
ess>C: |Users|jerem>
Run
```

```
Ma
```

```
Django Ver
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CTRL-BREAK.
```

METODA GET  
(DJANGO)

test\_08 &gt; formapp &gt; views.py

Project

Commit

Pull Requests

DB Browser

Structure

Bookmarks

```
from django.shortcuts import HttpResponse

def index(request):
    if request.method == "GET":
        return HttpResponse("Zapytanie GET")

    elif request.method == "POST":
        return HttpResponse("Zapytanie POST")
```

Terminal: Local +

sessions.

Run 'python manage.py migrate' to apply them.

May 01, 2022 - 08:47:47

Django version 4.0.4, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

*METODA POST  
(DJANGO)*

test\_08 &gt; formapp &gt; views.py

Project

Commit

Pull Requests

DB Browser

&gt;

venv library

.gitignore

db.sqlite3

manage.py

README

Terminal:

Loc:

sessions

Run 'p'

May 0

Django

Starting

Bookmarks

Bookmarks

 test\_08 C:\Users\jerem\Py  
config  
  \_\_init\_\_.py  
  asgi.py  
  settings.py  
  urls.py  
  wsgi.py  
formapp  
  migrations  
  templates  
    formapp  
      form.html  
    \_\_init\_\_.py  
    admin.py  
    apps.py  
    models.py  
    tests.py  
    urls.py  
    views.py  
venv library  
  .gitignore  
  db.sqlite3  
  manage.py  
  README views.py  
from django.shortcuts import HttpResponseRedirect  
  
def index(request):  
 if request.method == "GET":  
 return HttpResponseRedirect("Zapytanie GET")  
  
 elif request.method == "POST":  
 return HttpResponseRedirect("Zapytanie POST") Microsoft Windows [Version 10.0.19044.1645]  
c) Microsoft Corporation. Wszelkie prawa zastrzezone.  
C:\Users\jerem>curl --request POST http://127.0.0.1:8000/  
...  
  
<li>The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the page with the form, because the token is rotated after a login.</li>  
  
<p>You're seeing the help section of this page because you have <code>DEBUG = True</code> in your Django settings file. Change that to <code>False</code>, and only the initial error message will be displayed. </p>  
Run 'p'</div>  
May 0<br/></body></html>  
Starting  
Quit the server with CTRL-BREAK.

## METODA POST (ODRZUCENIE ZAPYTANIA ZE WZGLEDU NA PODATNOSC CSRF)

test\_08 &gt; config &gt; settings.py

```
Project . . . + - settings.py x views.py x
test_08 C:\Users\jerem\Py
  config
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  formapp
    migrations
    templates
      formapp
        form.html
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        urls.py
        views.py
  venv library root
    .gitignore
    db.sqlite3
    manage.py
    README.md

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    # 'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Terminal: Local +

sessions.

Run 'python manage.py migrate' to apply them.

May 01, 2022 - 08:50:28

Django version 4.0.4, using settings 'config.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

METODA POST  
(CHWILOWE WYLACZENIE  
ZABEZPIECZENIA CSRF)

test\_08 &gt; formapp &gt; views.py

Project

Commit

Pull Requests

DB Browser

Wiersz poleceń

&gt;

venv library/r

.gitignore

db.sqlite3

manage.py

README

Terminal: Local

sessions

Zapytanie GET

Run 'p

C: |Users |jerem|&gt;

May 01

Django version 4.0.4, using settings 'test\_08.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CTRL-BREAK.

```
from django.shortcuts import HttpResponse

def index(request):
    if request.method == "GET":
        return HttpResponse("Zapytanie GET")

    elif request.method == "POST":
        return HttpResponse("Zapytanie POST")
```

**METODA POST  
(DJANGO)**

# *FORMULARZ HTML*

*GRAFICZNY ELEMENT HTML  
DO WPROWADZANIA DANYCH*

# FORMULARZE HTML

*Formularze to elementy HTML, których celem jest udostępnienie użytkownikowi wygodnego sposobu na wprowadzenie danych. Dane wprowadzone w formularzu mogą być wysłane na serwer jedną z dwóch poznanych metod: GET lub POST.*

*Formularz jako elementem HTML posiada atrybuty. Trzy podstawowe atrybuty formularza to:*

- ***action*** – definiuje url (adres) na jaki zostaną przesłane wprowadzone dane. Domyślana wartość atrybutu *action* to ten sam adres, na którym wyświetlony został formularz.
- ***method*** – definiuje metodę, którą wprowadzone dane zostaną przesłane. Domyślana wartość atrybutu *method* to "GET".
- ***target*** – definiuje miejsce przeglądarki, w którym po przesłaniu danych zostanie wyświetlona odpowiedź serwera. Domyślana wartość atrybutu *target* to "*\_self*" oznaczająca, że odpowiedź zostanie wyświetlona w tej samej zakładce przeglądarki, w której wyświetlony był formularz. Dopuszczalne wartości to: *\_blank, \_self, \_parent, \_top, <framename>*.

# *STRUKUTURA FORMULARZA HTML*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Formularz</title>
</head>
<body>
    <form action="https://www.google.pl/search" method="get">
        <input type="text" name="q">
        <input type="submit" value="Szukaj w Google">
    </form>
</body>
</html>
```

*Formularz html zaczyna się znacznikiem `<form>` i kończy znacznikiem `</form>`.*

*Wewnątrz formularza umieszczane są odpowiednie pola (tzw. elementy formularza) do wprowadzenia wszystkich potrzebnych danych (np. `input`, `select`, `button`, ...)*

*FORMULARZ GET*

pythonProject8 &gt; forms &gt; 1\_formularze.html

Project

- pythonProject8 C:\Users\jerem  
  forms  
    1\_formularze.html
- intro
- relationships
- user\_auth
- venv library root  
  db.sqlite3
- manage.py

External Libraries

Scratches and Consoles

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Formularz 1</title>
</head>
<body>
    <form action="https://www.google.pl/search" method="get">
        <input type="text" name="q">
        <input type="submit" value="Szukaj w Google">
    </form>
</body>
</html>
```

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject8&gt;

FORMULARZ GET

TODO

Problems

DB Execution Console

Terminal

Python Console

pythonProject8 &gt; forms &gt; 1\_formularze.html

Project

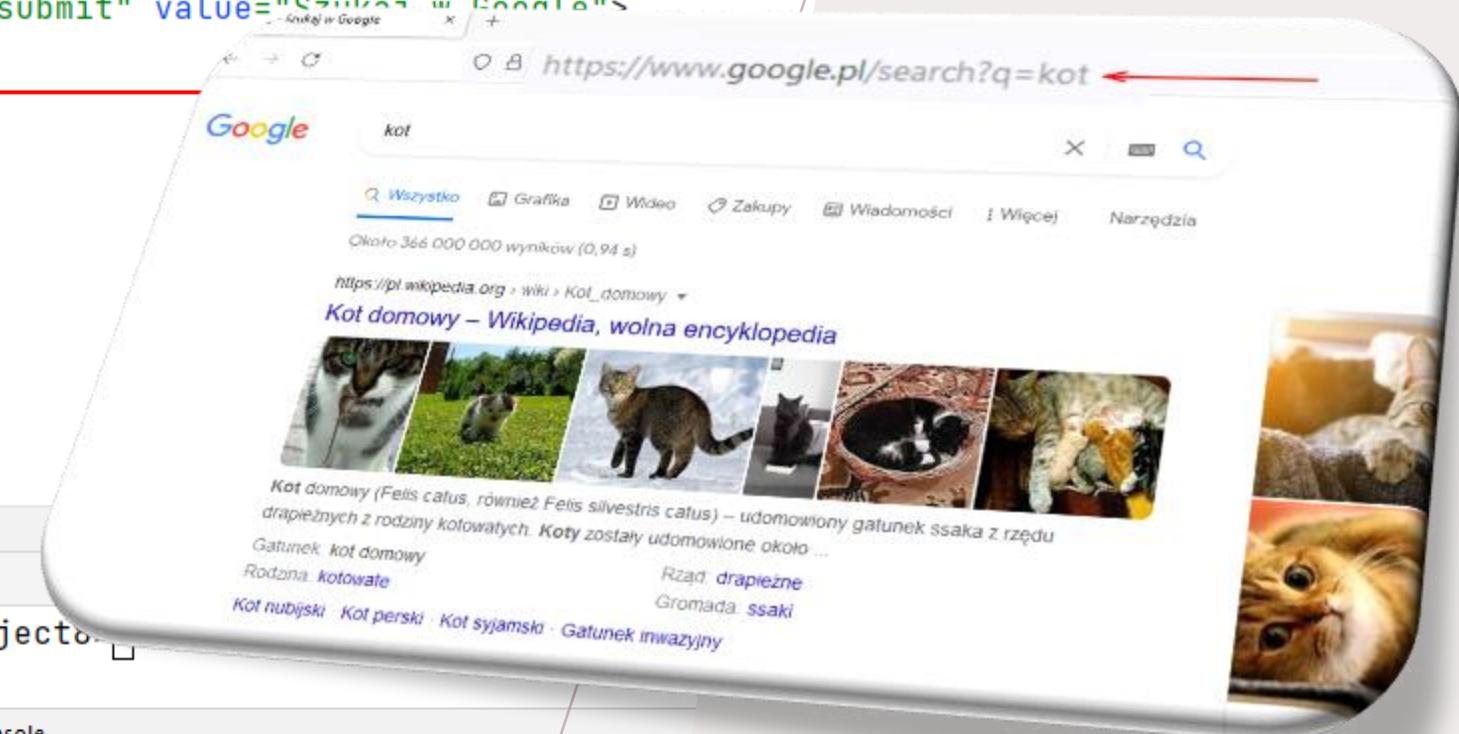
- pythonProject8 C:\Users\jerem  
  forms  
    1\_formularze.html  
  intro  
  relationships  
  user\_auth  
  venv library root  
    db.sqlite3  
  manage.py
- External Libraries
- Scratches and Consoles

DB Browser

Structure

Favorites

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Formularz 1</title>
</head>
<body>
    <form action="https://www.google.pl/search" method="get">
        <input type="text" name="q">
        <input type="submit" value="Szukaj w Google" />
    </form>
</body>
</html>
```



Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject8&gt;

TODO

Problems

DB Execution Console

Terminal

Python Console

*FORMULARZ POST*

pythonProject8 &gt; forms &gt; 2\_formularze.html

Project DB Browser Structure Favorites

pythonProject8 C:\Users\jerem\PycharmProjects\pythonProject8

forms

1\_formularze.html 2\_formularze.html

intro relationships user\_auth venv library root db.sqlite3 manage.py External Libraries Scratches and Consoles

2\_formularze.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Formularz 2</title>
    </head>
    <body>
        <form method="post">
            <input type="text" name="user" placeholder="user">
            <input type="password" name="password" placeholder="pas">
            <input type="submit", value="Loguj">
        </form>
    </body>
</html>
```

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject8>

TODO Problems DB Execution Console Terminal Python Console

FORMULARZ POST

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help pythonProject8 - 2\_formularze.html

pythonProject8 > forms > 2\_formularze.html

Project DB Browser Structure Favorites

pythonProject8 C:\Users\jerem  
forms  
1\_formularze.html  
2\_formularze.html  
intro  
relationships  
user\_auth  
venv library root  
db.sqlite3  
manage.py  
External Libraries  
Scratches and Consoles

2\_formularze.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Formularz 2</title>
    </head>
    <body>
        <form method="post">
            <input type="text" name="user" placeholder="user">
            <input type="password" name="password" placeholder="pas">
            <input type="submit" value="Loguj">
        </form>
    </body>

```

localhost:64321/pythonProject8/forms/2\_formularze.html

Inspektor Konsola Debugger Sieci Edytor stylów Wydajność Pamięć Dane Dostępność Wątki

Stan	Metoda	Domena	Plik	Initiator	Typ	Przesłano	Rozmiar	Inne
POST	localhost:64321	2_formularze.html		document	HTML	768 B	355 B	Nagłówki Ciasteczka Zadanie Odpowiedź Pomiary czasu
GET	localhost:64321	favicon.ico		FaviconLoader.jsm:1	vrnd.mic...	w pamięci podręcznej	1,37 KB	Zawartość zadania

Terminal: Local +

(venv) C:\Users\jerem\PycharmProjects\pythonProject8

TODO Problems DB Execution Console Terminal Python Console

user=test&password=test Nieprzetworzone



# *ELEMENTY FORMULARZA*

# *ELEMENTY FORMULARZA*

*Elementy formularza to elementy html, z których budujemy formularz.*

*Zadaniem elementów formularza jest wyświetlenie wygodnych w użyciu widgetów (elementów graficznych) do wprowadzenia danych (np. kalendarz w polu do wprowadzenia daty). Ponadto elementy formularza posiadają wbudowane walidatory danych.*

*Wyróżniamy 5 elementów formularza:*

- *element <select>*
- *element <textarea>*
- *element <button>*
- *element <datalist>*
- *element <input>*

*Poszczególne elementy są omówione w dodatkach, na końcu tej prezentacji.*

*OBSŁUGA DANYCH PO  
STRONIE DJANGO*



# *OBSŁUGA DANYCH GET W DJANGO*

# *PARAMETRY GET W DJANGO*

*Parametry przesłane metodą get są przechowywane w Django w atrybucie GET obiektu request*

*request.GET*

*request.GET to obiekt klasy QueryDict (klasa rozszerzająca możliwości tradycyjnego słownika pythonowego). Posiada metody słownikowe. W szczególności, odwołanie do elementu obiektu klasy QueryDict ma postać:*

*request.GET.get('<key>')*

*OBSŁUGA DANYCH  
POST W DJANGO*

# *PARAMETRY POST W DJANGO*

*Parametry przesłane metodą post są przechowywane w Django w atrybucie POST obiektu request*

*request.POST*

*request.POST również jest obiektem klasy QueryDict:*

*request.POST.get('<key>')*

Ale sama zmiana wartości atrybutu 'method' formularza na "post" nie wystarczy ze względu na wbudowane w Django zabezpieczenie przed atakiem csrf. Podatność CSRF rozpatrywana jest dopiero na etapie formularza POST, ponieważ zakłada się, że modyfikować stan bazy można wyłącznie na podstawie danych przesyłanych metodą POST.



# *CSRF*

*(CROSS-SITE REQUEST FORGERY)*

# *PODATNOSC CSRF - WSTEP*

*Mówimy, że aplikacja posiada podatność **cross-site request forgery** kiedy nie potrafi rozróżnić czy dane, które przyszły na serwer pochodzą z formularza wygenerowanego przez tą aplikację, czy z **obcego**, niewygenerowanego przez tą aplikację formularza.*

# *CSRF*

Request Forgery



Cross-Site



Phishing



# *PODATNOSC CSRF - OCHRONA*

*Najprostszy sposób ochrony przed atakiem csrf polega na oznaczaniu każdego generowanego w aplikacji formularza **unikalnym identyfikatorem** i zapisywaniu wygenerowanego identyfikatora w bazie. Następnie, po przyjściu danych metodą POST aplikacja powinna sprawdzać, czy wśród przesyłanych danych formularza znajduje się identyfikator i jeżeli istnieje to porównać go z identyfikatorem zapisanym w bazie. Dane powinny być dalej przetwarzane wtedy i tylko wtedy gdy oba identyfikatory są identyczne.*

*Django posiada wbudowane rozwiązanie, dzięki któremu nie musimy samodzielnie implementować całego mechanizmu ochrony. Wystarczy w szablonie, w formularzu (post) umieścić znacznik **csrf\_token**. Django domyślnie odrzuca wszystkie dane post wśród których nie ma takiego znacznika (csrftoken).*

*ODRZUCENIE DANYCH POST ZE  
WZGLEDUNA BRAK  
IDENTYFIKATORA CSRF TOKEN  
WDJANGO*

name.html

views.py

```
1 from django.shortcuts import render
2
3
4 def name(request):
5
6     print(request.GET)
7
8     return render(
9         request,
10        'formapp/name.html',
11    )
12
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8     <form method="POST">
9       <input type="text" name="n">
10      <input type="submit" value="Wyślij">
11    </form>
12  </body>
13 </html>
```

/favicon.ico HTTP/1.1" 404 2750

/formapp/ HTTP/1.1" 404 2806

/formapp/name/ HTTP/1.1" 200 234

g.): /formapp/name/

/formapp/name/ HTTP/1.1" 403 2506

*FORMULARZ POST*

Title

127.0.0.1:8000/formapp/name/

Wyślij

Elements Console Network CSS Overview Sources Performance Mem

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <form method="POST"> == $0
      <input type="text" name="n">
      <input type="submit" value="Wyślij">
    </form>
  </body>
</html>
```

html body form

Console Issues +

## FORMULARZ POST (WYSWIETLENIE FORMULARZA)

403 Forbidden

127.0.0.1:8000/formapp/name/

# Forbidden (403)

CSRF verification failed. Request aborted.

## Help

Reason given for failure:  
CSRF token missing.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when [Django's CSRF mechanism](#) has not been used correctly. For P

- Your browser is accepting cookies.
- The view function passes a `request` to the template's `render` method.
- In the template, there is a `{% csrf_token %}` template tag inside each POST form that targets an internal URL.
- If you are not using `CsrfViewMiddleware`, then you must use `csrf_protect` on any views that use the `csrf_token` template tag, as well as those th
- The form has a valid CSRF token. After logging in in another browser tab or hitting the back button after a login, you may need to reload the

You're seeing the help section of this page because you have `DEBUG = True` in your Django settings file. Change that to `False`, and only the initial error

You can customize this page using the `CSRF_FAILURE_VIEW` setting.

*KOMUNIKAT O  
BRAKU CSRFTOKEN  
(PRZESŁANIE DANYCH)*

*OCHRONA PRZED  
PODATNOSCIA CSRF  
WDJANGO*

## ZDPYTp46\_backend &gt; config &gt; settings.py

Project Commit Pull Requests External Libraries Scratches and Consoles Structure Bookmarks

```
Project views.py settings.py
46     'linkapp.apps.LinkappConfig',
47     'formapp.apps.FormappConfig',
48 ]
49
50 MIDDLEWARE = [
51     'django.middleware.security.SecurityMiddleware',
52     'django.contrib.sessions.middleware.SessionMiddleware',
53     'django.middleware.common.CommonMiddleware',
54     'django.middleware.csrf.CsrfViewMiddleware', # Line 54
55     'django.contrib.auth.middleware.AuthenticationMiddleware',
56     'django.contrib.messages.middleware.MessageMiddleware',
57     'django.middleware.clickjacking.XFrameOptionsMiddleware'
58 ]
59
60 ROOT_URLCONF = 'config.urls'
61
62 TEMPLATES = [
63     {
64         'BACKEND': 'django.template.backends.django.DjangoTemplates',
65         'DIRS': [],
66         'APP_DIRS': True,
67         'OPTIONS': {
68             'context_processors': [

```

CSRF  
MIDDLEWARE  
(WBUDOWANA)



```
views.py x
1 from django.shortcuts import render
2
3
4 def name(request):
5
6     print(request.GET)
7
8     return render(
9         request,
10        'formapp/name.html',
11    )
12
```

```
name.html x
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8     <form method="POST">{% csrf_token %}
9       <input type="text" name="n">
10      <input type="submit" value="Wyślij" />
11    </form>
12  </body>
13 </html>
```

.ico HTTP/1.1" 404 2750

/ HTTP/1.1" 404 2806

/name/ HTTP/1.1" 200 234

rmapp/name/

p/name/ HTTP/1.1" 403 2506

***FORMULARZ POST  
Z TOKENEM CSRF***

Formularz POST z tokenem CSRF – TU CSRFMIDDLEWARETOKEN (WYSWIETLENIE FORMULARZA)

The screenshot shows a browser window with the URL `127.0.0.1:8000/formapp/name/`. The page contains a single button labeled "Wyślij". The browser's developer tools are open, specifically the "Elements" tab, which displays the HTML source code of the page. A red box highlights the `<input type="hidden" name="csrfmiddlewaretoken" value="RHiYcdoz9p1tvalueWOF712bj...</input>` line, indicating the presence of a CSRF token in the form.

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    ... <form method="POST"> -- ⚡
      <input type="hidden" name="csrfmiddlewaretoken" value="RHiYcdoz9p1tvalueWOF712bj...</input>
      <input type="text" name="n">
      <input type="submit" value="Wyślij">
    </form>
  </body>
</html>
```

*FORMULARZ POST Z  
TOKENEM CSRF – TU  
CSRFMIDDLEWARETOKEN  
(WYSWIETLENIE FORMULARZA)*

# *PRZEKIEROWANIA*

## *(FUNKCJA REDIRECT)*

# *DANE POST - PRZEKIEROWANIE*

*Przyjmuje się, że w wyniku przesłania danych metodą post na danych przechowywanych w aplikacji zostanie wykonana jedna z operacji niebezpiecznych (C, U lub D z CRUD). Przy takim założeniu odświeżenie strony (czyli ponowne wysłanie identycznego zapytania) spowoduje ponowne wykonanie niebezpiecznej operacji.*

*Jeżeli w wyniku zapytania został usunięty jakiś wpis, to ponowne wysłanie takiego zapytania spowoduje próbę usunięcia nieistniejącego już wpisu. Podobne błędy pojawią się, jeżeli w wyniku zapytania post coś powinno zostać dodane lub zmodyfikowane.*

*Przeglądarki posiadają wbudowane ostrzeżenie przed realizacją takiego scenariuszem.*

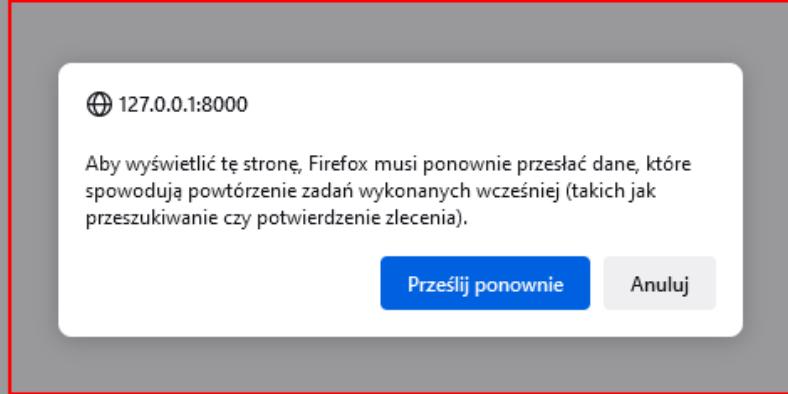
views.py x

```
def name_post(request):
    name = request.POST.get('name')
    if name:
        return render(
            request,
            'formapp/hello.html',
            context={
                'name': name,
            }
        )
    return render(
        request,
        'formapp/name_post.html',
    )
```

Brak  
przekierowania

*METODA POST  
BRAK PRZEKIEROWANIA*

+

 127.0.0.1:8000/form/name-post/

*METODA POST  
BRAK PRZEKIEROWANIA –  
OSTRZEZENIE PRZEGŁADARKI*

# *PRZEKIEROWANIA – FUNKCJA REDIRECT*

*Żeby uniknąć ponownego wykonywania niebezpiecznej operacji, po obsłużeniu danych **POST** użytkownika należy **ZAWSZE PRZEKIEROWYWAĆ** na odpowiedni endpoint.*

*Do przekierowywania używamy funkcji **redirect** z modułu `django.shortcuts`*

```
redirect('<nazwa_mapowania>')
```

*Jeżeli endpoint, na który przekierowujemy posiada dodatkowe parametry należy umieścić je po przecinku:*

```
redirect('<nazwa_mapowania>', <nazwa_parametru>=<wartość parametru>')
```



views.py

```
20
21     def name_post(request):
22         name = request.POST.get('name')
23         if name:
24             return redirect('formapp:hello', name=name)
25
26         return render(
27             request,
28             'formapp/name_post.html',
29         )
30
31
32     def hello(request, name):
33         return render(
34             request,
35             'formapp/hello.html',
36             context={
37                 'name': name,
38             }
39         )
40
41     |
```

Dodatkowe parametry  
(jeżeli potrzebne)

*METODA POST  
PRZEKIEROWANIE*



*WALIDACJA DANYCH FORMULARZA*

# *WALIDACJA DANYCH FORMULARZA*

*Walidacją danych formularza nazywamy proces sprawdzenia czy dane wprowadzone w formularzu mają poprawny format.*

*Dobrą praktyką jest sprawdzanie poprawności danych w dwóch miejscach – na frontendzie i na backendzie. Zawsze powinniśmy pamiętać o sprawdzeniu poprawności danych na backendzie. Sprawdzenie poprawności danych na frontendzie nie jest obowiązkowe, ale sprawia, że aplikacja jest przyjemniejsza w użyciu.*

*WALIDACJA DANYCH  
FORMULARZA  
(FRONTEND)*

*WALIDACJA DANYCH  
FORMULARZA  
(BACKEND)*

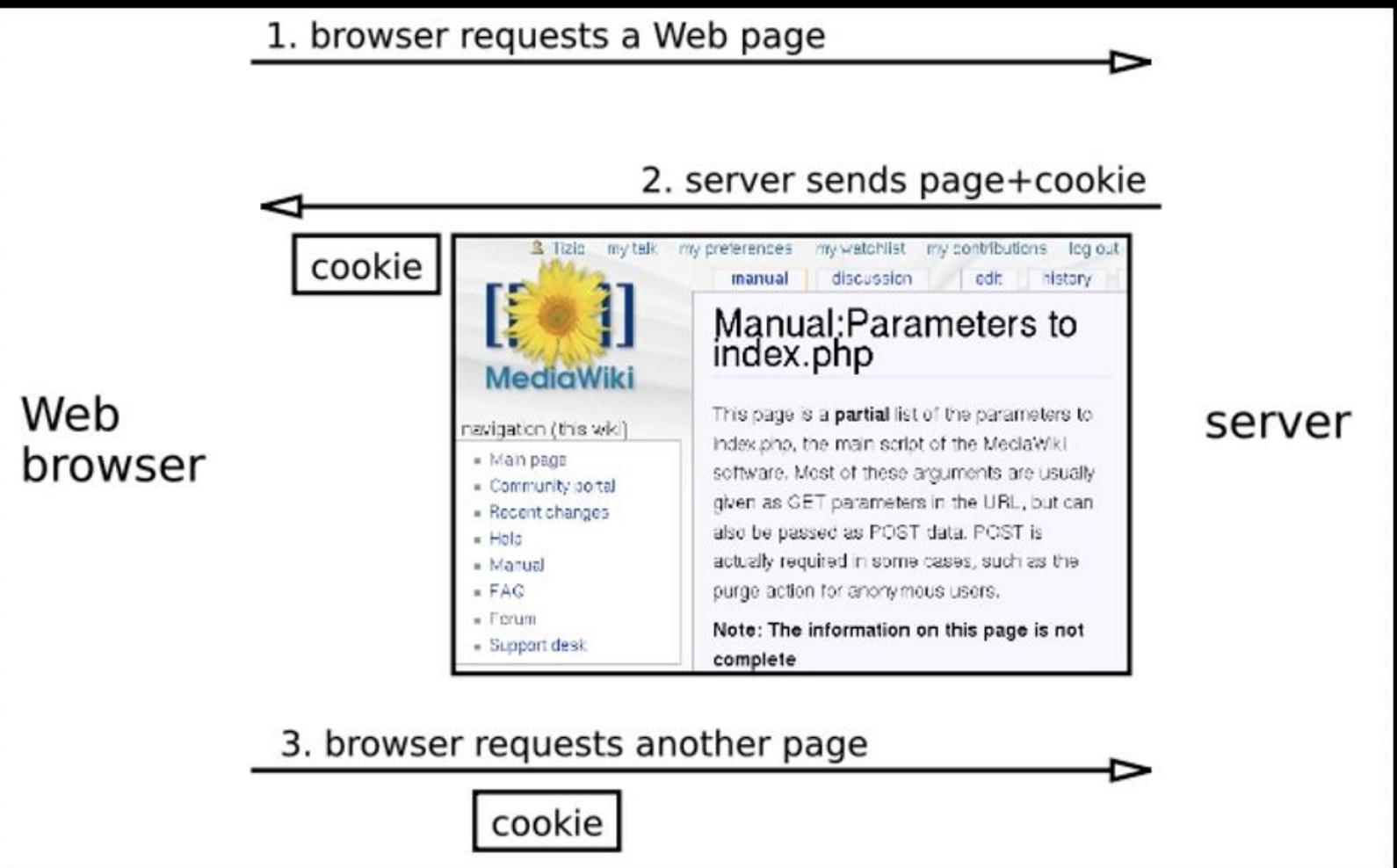


*FORMULARZE W*  
*DJANGO*

# *WDJANGO WYROZNIAMY 3 RODZAJE FORMULARZY*

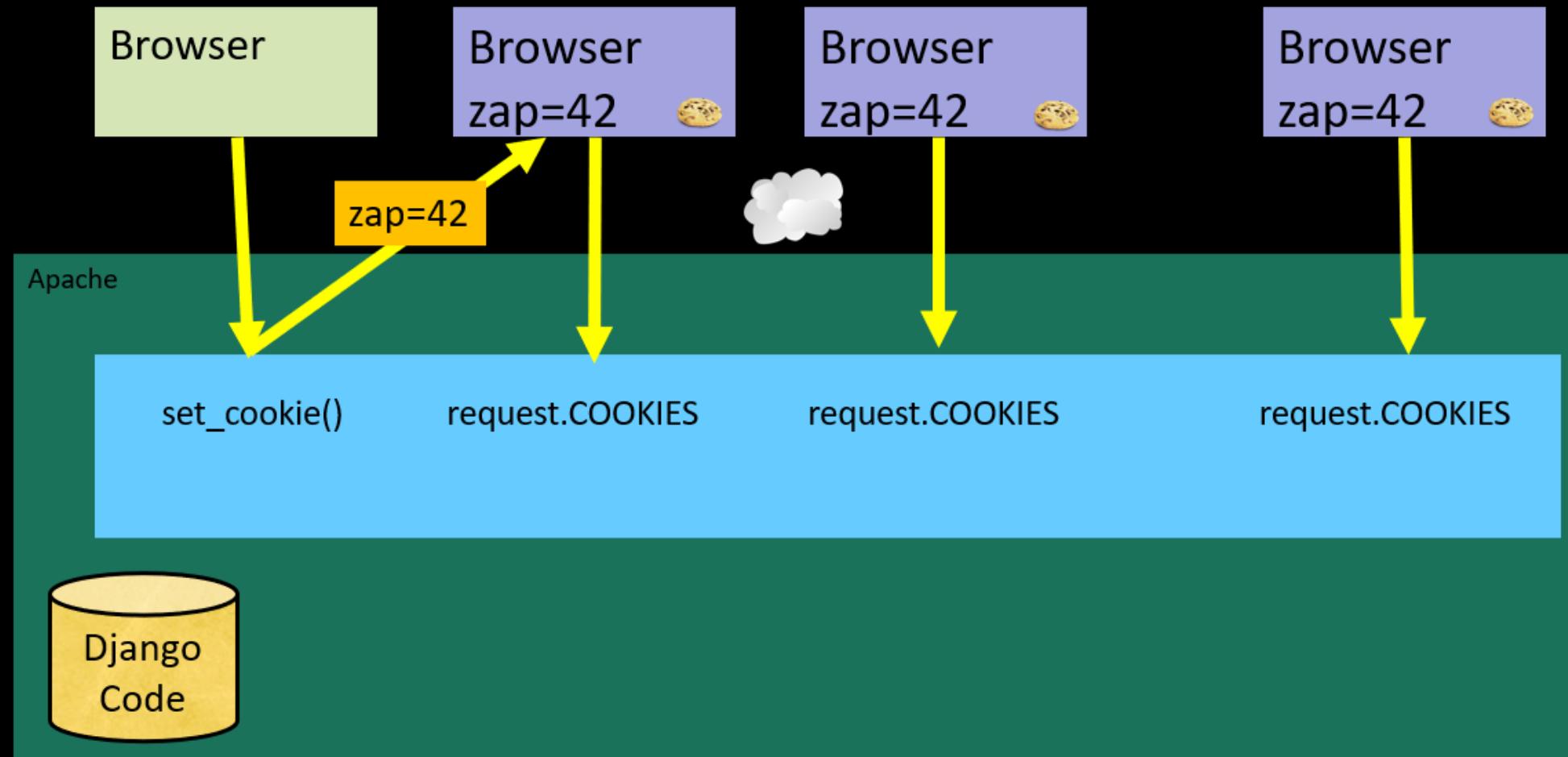
1. Formularz HTML
2. Formularz Django (omówione w kolejnych prezentacjach)
3. Formularz modelu Django (omówione w kolejnych prezentacjach)

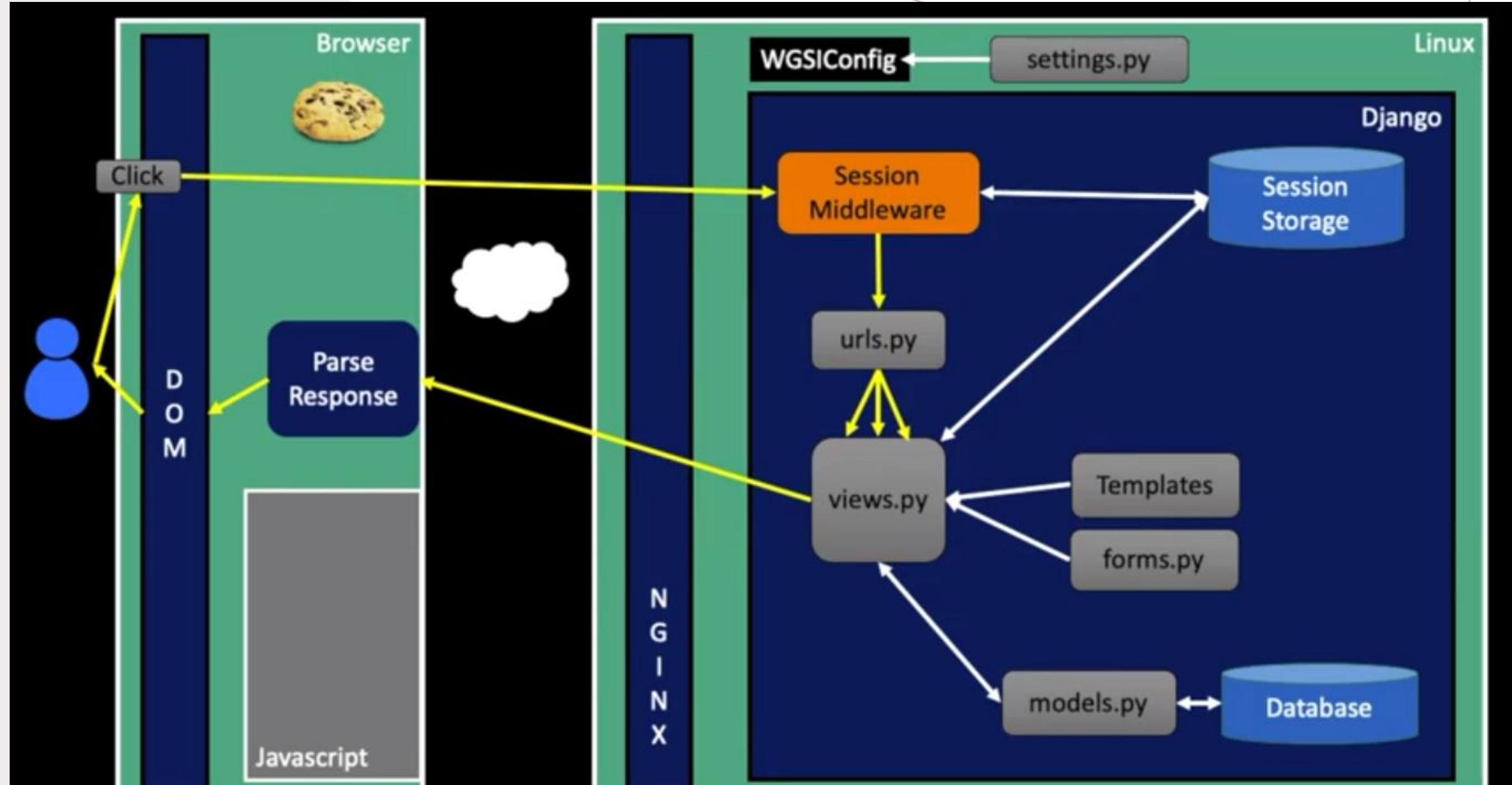
*CIASTECZKA*



[http://en.wikipedia.org/wiki/HTTP\\_cookie](http://en.wikipedia.org/wiki/HTTP_cookie)

Time





# *LOKALIZACJA*

# *LOKALIZACJA CIASTECZEK*

W zależności od systemu operacyjnego, przeglądarki internetowej oraz jej wersji ciasteczka przechowywane są w innym miejscu. Najczęściej przeglądarki zapisują swoje ciasteczka w bazie sqlite.



Plik Narzędzia główne Udosłupnianie Widok



C:\Users\jerem\AppData\Roaming\Mozilla\Firefox\Profiles\vap90txz.default-release

Szybki dostęp

- Pulpit
- Pobrane
- Dokumenty
- Obrazy
- Nowy folder (3)
- django
- Nowy folder (2)
- Nowy folder (3)
- ZDPYTp0l46\_backend

OneDrive - Personal

- Dokumenty
- Obrazy
- Pulpit

Ten komputer

- Dokumenty
- Muzyka
- Obiekty 3D
- Obrazy
- Pobrane
- Pulpit
- Wideo

Dysk lokalny (C:)

Sieć

Nazwa	Data modyfikacji	Typ	Rozmiar
storage.cache	08.10.2021 09:22	Folder plików	
storage	04.12.2021 19:58	Folder plików	
weave			
activity-stream.discovery_stream.json	17.04.2022 17:11	JSON Source File	3 KB
addons.json	17.04.2022 22:26	JSON Source File	1 KB
addonStartup.json.lz4	14.04.2022 18:33	Plik LZ4	5 KB
AlternateServices.txt	18.04.2022 10:45	Dokument tekstowy	127 KB
broadcast-listeners.json	18.04.2022 05:04	JSON Source File	1 KB
cert9.db	10.03.2022 20:22	Data Base File	288 KB
compatibility.ini	14.04.2022 18:33	Ustawienia konfig...	1 KB
containers.json	08.10.2021 09:22	JSON Source File	1 KB
content-prefs.sqlite	06.04.2022 16:41	Plik SQLITE	224 KB
<b>cookies.sqlite</b>	<b>18.04.2022 10:44</b>	<b>Plik SQLITE</b>	<b>1 024 KB</b>
cookies.sqlite-shm	14.04.2022 18:33	Plik SQLITE-SHM	32 KB
cookies.sqlite-wal	18.04.2022 10:44	Plik SQLITE-WAL	641 KB
enumerate_devices.txt	02.04.2022 19:12	Dokument tekstowy	1 KB
extension-preferences.json	24.03.2022 18:46	JSON Source File	2 KB
extensions.json	17.04.2022 22:28	JSON Source File	50 KB
favicons.sqlite	18.04.2022 10:45	Plik SQLITE	1 376 KB
formhistory.sqlite	17.04.2022 21:38	Plik SQLITE	256 KB
handlers.json	31.03.2022 21:13	JSON Source File	1 KB
key4.db	29.11.2021 19:53	Data Base File	288 KB
logins.json	10.03.2022 12:27	JSON Source File	2 KB
logins-backup.json	29.11.2021 19:53	JSON Source File	1 KB
notificationstore.json	16.04.2022 15:36	JSON Source File	1 KB
parent.lock	14.04.2022 18:33	Plik LOCK	0 KB
permissions.sqlite	18.04.2022 01:35	Plik SQLITE	96 KB
pkcs11.txt	08.10.2021 09:22	Dokument tekstowy	1 KB
places.sqlite	18.04.2022 10:45	Plik SQLITE	5 120 KB
<b>prefs.js</b>	<b>18.04.2022 10:45</b>	<b>Plik JavaScript</b>	<b>20 KB</b>
protections.sqlite	18.04.2022 10:44	Plik SQLITE	64 KB
search.json.mozlz4	14.04.2022 18:33	Plik MOZLZ4	1 KB
serviceworker.txt	16.04.2022 22:22	Dokument tekstowy	12 KB
sessionCheckpoints.json	18.04.2022 10:45	JSON Source File	1 KB
sessionstore.jsonlz4	18.04.2022 10:45	Plik JSONLZ4	37 KB

Elementy: 54 1 zaznaczony element. 1,00 MB

WINDOWS 10  
FIREFOX 99.0.1

SQL Commit Rollback Auto cookies.sqlite < N/A >

Enter a part of object name

- census.sqlite
- census.sqlite 2
- census.sqlite 3
- coderslab\_vm - localhost:
- cookies.sqlite**
- Tables
  - moz\_cookies
- Views
- Indexes
- Sequences
- Table Triggers
- Data Types

exercises - localhost:5432

postgres - localhost:5432

Properties Data ER Diagram

moz\_cookies Enter a SQL expression to filter results (use Ctrl+Space)

Grid	123	ABC	oi	ABC name	ABC value	ABC host	ABC pa	123 expiry	123 lastAccessed
1	5			CONSENT	YES-shp.gws-20220323-0-RC3.pl+FX+191	.google.pl	/	1,711,991,576	1,650,238,503,727,000
2	6			CONSENT	YES-shp.gws-20220323-0-RC3.pl+FX+861	.youtube.com	/	1,711,991,576	1,650,140,539,051,000
3	8			CONSENT	YES-shp.gws-20220323-0-RC3.pl+FX+731	.google.com	/	1,711,991,576	1,650,238,503,724,00
4	18			ppwp_wp_session	4185a0883a663c6c3c94bdf60979e003%7C%7C	mpress.pl	/	1,648,921,388	1,648,919,588,995,00
5	19			_ga	GA1.2.45905036.1648919589	.mpress.pl	/	1,711,991,589	1,648,919,589,122,0
6	20			_gid	GA1.2.200713500.1648919589	.mpress.pl	/	1,649,005,989	1,648,919,589,123,0
7	21			_gat_gtag_UA_126083467_7	1	.mpress.pl	/	1,648,919,649	1,648,919,589,126,
8	22			_gads	ID=1646bb35f4c5de7a-228c0fda6bcd0029:T=	.mpress.pl	/	1,682,615,588	1,648,919,589,715
9	25			uuid	ca26ea9e-dff1-4def-abc1-6df9b31b5e7f-2022	.innovid.com	/	1,656,695,594	1,648,919,594,190
10	29			cto_bundle	bsfqiV9hNUI1U2xZdTVhVFBNeTNKeVNkNFRF	.mpress.pl	/	1,682,615,599	1,648,919,599,68
11	34			trc_cookie_storage	taboola%2520global%253Auser-id%3D27e3c	.mpress.pl	/	1,680,455,661	1,648,919,661,7
12	36			tb_push_test	active	ladygreat.com	/	1,649,006,072	1,648,919,867,7
13	37			cn_test	c1n	ladygreat.com	/	1,649,006,072	1,648,919,867,
14	38			cmpoct_test	cmpagg	ladygreat.com	/	1,649,006,072	1,648,919,867,
15	39			pbtol_test	pb3900	ladygreat.com	/	1,649,006,072	1,648,919,867
16	40			ch_test	ctl	ladygreat.com	/	1,649,006,072	1,648,919,86
17	41			amztam_test	high	ladygreat.com	/	1,649,006,072	1,648,919,86
18	42			BO_test	B4O	ladygreat.com	/	1,649,006,072	1,648,919,8
19	43			utm_source	taboola	ladygreat.com	/	1,648,926,872	1,648,919,8
20	44			utm_medium	marcinp-mpress	ladygreat.com	/	1,648,926,872	1,648,919,
21	45			utm_campaign	11231003	ladygreat.com	/	1,648,926,872	1,648,919
22	46			utm_content	<a href="http://cdn.taboola.com/libtrc/static/thun">http://cdn.taboola.com/libtrc/static/thun</a>	ladygreat.com	/	1,648,926,872	1,648,919
23	47			utm_term	Pamiętasz ją? Weź głęboki oddech zanim zob	ladygreat.com	/	1,648,926,872	1,648,91
24	48			_gcl_au	1.1.576554710.1648919673	.ladygreat.com	/	1,656,695,673	1,648,91
25	49			advanced_ads_browser_wic	1583	ladygreat.com	/	1,651,511,673	1,648,9
26	50			_ga	GA1.2.1881200418.1648919674	.ladygreat.com	/	1,711,991,674	1,648,
27	51			_gid	GA1.2.802795425.1648919674	.ladygreat.com	/	1,649,006,074	1,648,
28	53			_gads	ID=19fcfbe890da57e2:T=1648919674:S=ALNL	.ladygreat.com	/	1,682,615,674	1,648
29	54			outbrain_cid_fetch	true	ladygreat.com	/	1,648,919,975	1,64
30	56			euconsent-v2	CPW1SzTPW1SzTAKAVAENCJCsAP_AAH_AAI	.ladygreat.com	/	1,682,615,676	1,64
31	57			addtl_consent	1~39.4.3.9.6.9.13.6.4.15.9.5.2.7.4.1.7.1.3.2.10.3.	.ladygreat.com	/	1,682,615,676	1,6
32	90			_qca	P0-930687547-1648919676041	.ladygreat.com	/	1,682,788,477	1,6

< Save Cancel Script | 200 1,254 Rows: 1 1254 row(s)

WINDOWS 10  
FIREFOX 99.0.1



Network

Plik Narzędzia główne Udostępnianie Widok

← → ⌂ ⌃ C:\Users\jerem\AppData\Local\Google\Chrome\User Data\Default\Network

## Szybki dostęp

- Pulpit
- Pobrane
- Dokumenty
- Obrazy
- Nowy folder (3)
- django
- Nowy folder (2)
- Nowy folder (3)
- ZDPYTp0l46\_backend

## OneDrive - Personal

- Dokumenty
- Obrazy
- Pulpit

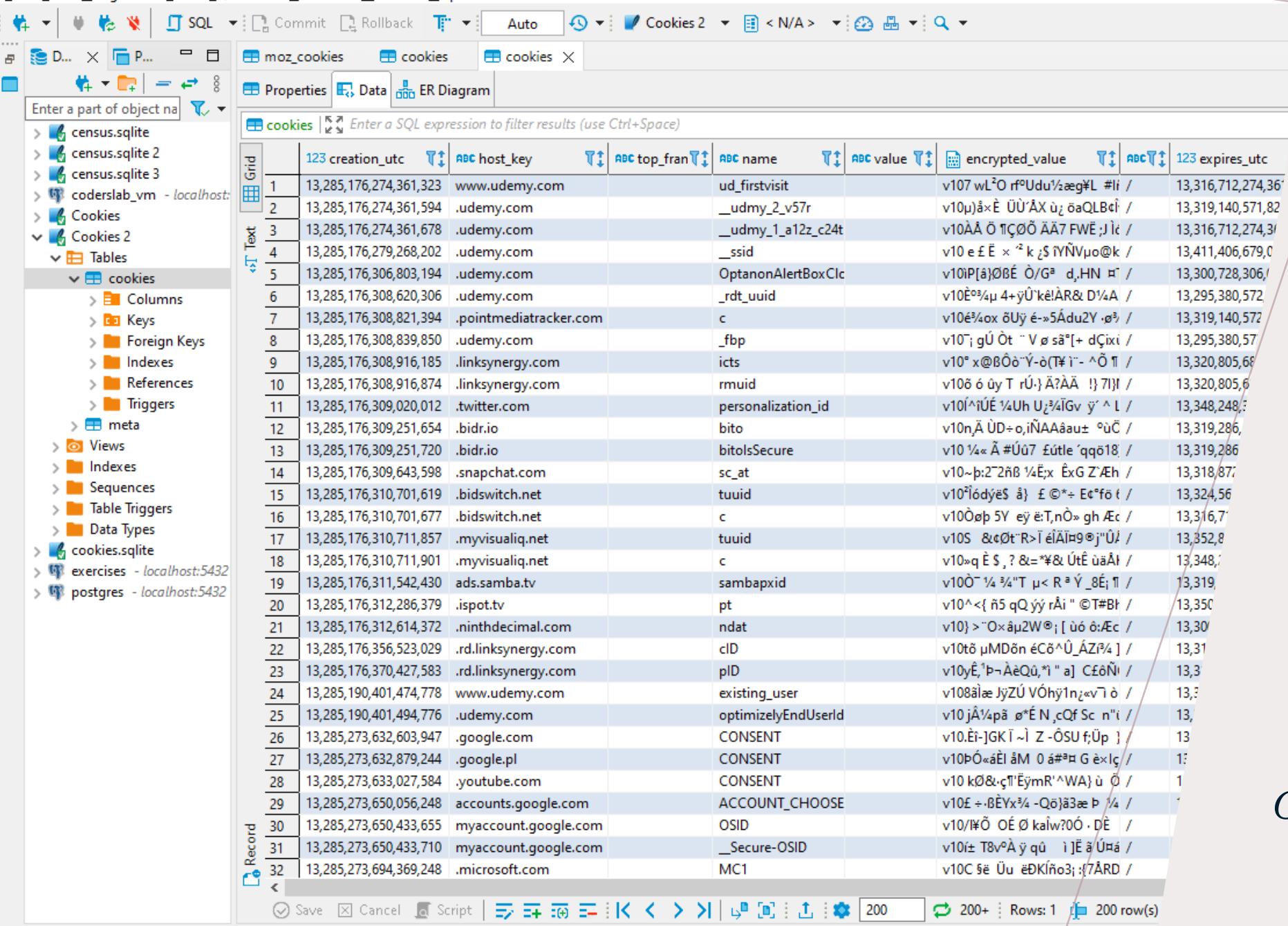
## Ten komputer

- Dokumenty
- Muzyka
- Obiekty 3D
- Obrazy
- Pobrane
- Pulpit
- Wideo
- Dysk lokalny (C:)

## Sieć

Nazwa	Data modyfikacji	Typ	Rozmiar
Cookies	18.04.2022 11:15	Plik	1 856 KB
Cookies-journal	18.04.2022 11:15	Plik	0 KB
Network Persistent State	18.04.2022 11:15	Plik	68 KB
NetworkDataMigrated	23.12.2021 14:01	Plik	0 KB
Reporting and NEL	18.04.2022 11:14	Plik	512 KB
Reporting and NEL-journal	18.04.2022 11:14	Plik	0 KB
TransportSecurity	18.04.2022 11:15	Plik	120 KB
Trust Tokens	17.04.2022 22:06	Plik	52 KB
Trust Tokens-journal	17.04.2022 22:06	Plik	0 KB

WINDOWS 10  
CHROME 100.0.4896



*WINDOWS 10  
CHROME 100.0.4896*

## Ustawienia

Przeszukaj ustawienia

Ty i Google

Autouzupełnianie

Prywatność i  
bezpieczeństwo

Wygląd

Wyszukiwarka

Domyślna przeglądarka

Po uruchomieniu

Zaawansowane

Rozszerzenia

Chrome – informacje

← Wszystkie witryny

Szukaj

Sortuj według Najczęstsze

Całkowita pamięć używana przez strony: 31,6 MB

[Wyczyść wszystkie dane](#)

live.com  
1 062 kB · 137 plików cookie

google.pl  
7,2 kB · 19 plików cookie

stackoverflow.com  
84 B · 6 plików cookie

google.com  
16,7 MB · 40 plików cookie

theguardian.com  
162 kB · 18 plików cookie

ayz.pl  
760 B · 1 plik cookie

mozilla.org  
1 757 B · 1 plik cookie

youtube.com  
1 193 B · 16 plików cookie

cinema-city.pl  
89 B · 13 plików cookie

bbc.com

*WINDOWS 10  
CHROME 100.0.4896*



Plik Narzędzia główne Udoskonalanie Widok

← → ⌂ ⌃ ⌄ C:\Users\jerem\AppData\Local\Microsoft\Edge\User Data\Default\Network

Szybki dostęp

▀ Pulpit  
▀ Pobrane  
▀ Dokumenty  
▀ Obrazy  
▀ Nowy folder (3)  
▀ django  
▀ Nowy folder (2)  
▀ Nowy folder (3)  
▀ ZDPYTpol46\_backend

▀ OneDrive - Personal  
▀ Dokumenty  
▀ Obrazy  
▀ Pulpit

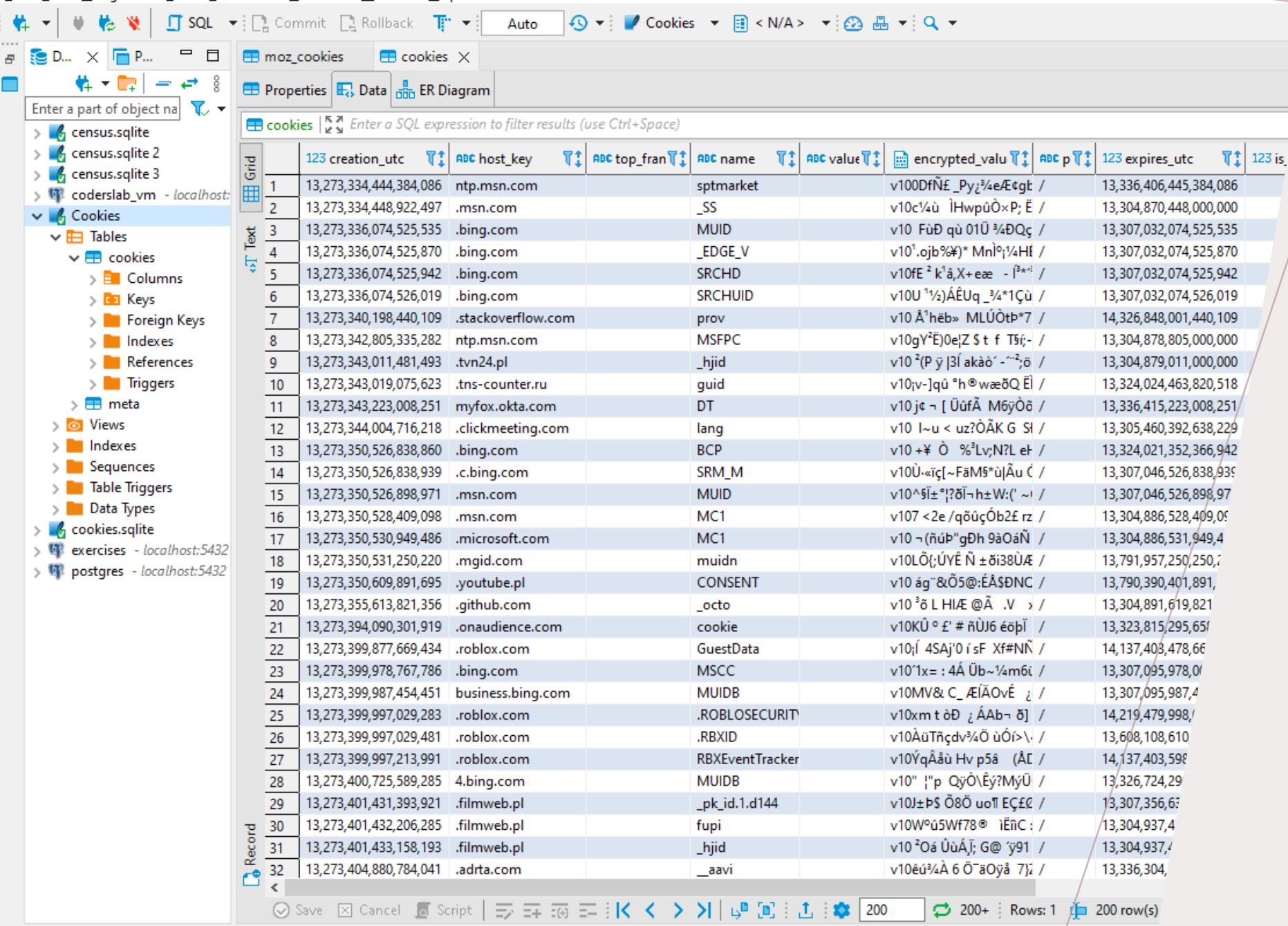
▀ Ten komputer

▀ Dokumenty  
▀ Muzyka  
▀ Obiekty 3D  
▀ Obrazy  
▀ Pobrane  
▀ Pulpit  
▀ Wideo  
▀ Dysk lokalny (C:)

▀ Sieć

Nazwa	Data modyfikacji	Typ	Rozmiar
Cookies	18.04.2022 11:03	Plik	2 048 KB
Cookies-journal	18.04.2022 11:03	Plik	0 KB
Network Persistent State	18.04.2022 11:03	Plik	74 KB
NetworkDataMigrated	05.02.2022 15:51	Plik	0 KB
Reporting and NEL	18.04.2022 11:02	Plik	864 KB
Reporting and NEL-journal	18.04.2022 11:02	Plik	0 KB
Token Bindings	21.12.2020 15:38	Plik	20 KB
Token Bindings-journal	21.12.2020 15:38	Plik	0 KB
TransportSecurity	18.04.2022 11:01	Plik	431 KB

WINDOWS 10  
EDGE 100.0.1185



WINDOWS 10  
EDGE 100.0.1185

Ustawienia

← Pliki cookie i dane witryn / Wszystkie pliki cookie i dane witryn

Wyszukaj w ustawieniach

Profile

Prywatność, wyszukiwanie i usługi

Wygląd

Ekran startowy strona główna i nowe karty

Udostępnianie, kopiowanie i wklejanie

**Pliki cookie i uprawnienia witryny**

Przeglądarka domyślna

Pobrane

Bezpieczeństwo rodzinne

Pasek przeglądarki Edge

Języki

Drukarki

System i wydajność

Resetuj ustawienia

Telefon i inne urządzenia

Ułatwienia dostępu

Microsoft Edge — informacje

Wyszukaj

Usuń wszystko

Usuń pliki cookie

Liczba bajtów: 51 • 2 pliki cookie

123movieshub.tc  
3 KB • 2 pliki cookie

1dapp.news  
2 KB • 2 pliki cookie

1dmp.io  
Liczba bajtów: 0 • 1 plik cookie

2embed.ru  
1 KB • 0 plików cookie

2win.pl  
Liczba bajtów: 0 • 1 plik cookie

300gospodarka.pl  
155 KB • 0 plików cookie

3destate.pl  
Liczba bajtów: 65 • 0 plików cookie

42heilbronn.de

WINDOWS 10  
EDGE 100.0.1185

## *INNE PRZEGŁADARKI (WINDOWS 10)*

### **Opera 85.0.4341**

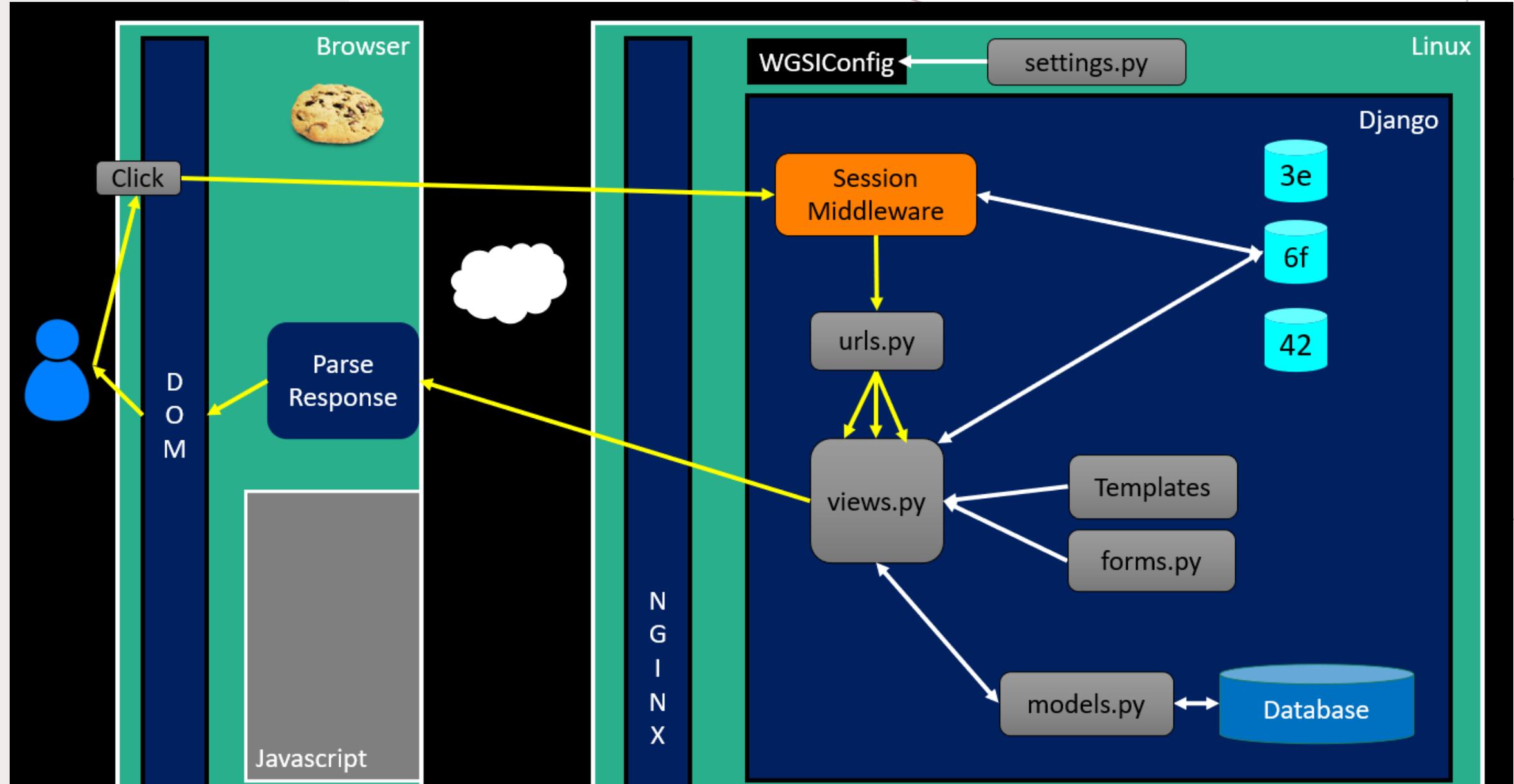
C:\Users\<nazwa\_uzytkownika>\AppData\Local\Opera Software\Opera Stable\Cookies

### **Brave 1.37.116**

C:\Users\<nazwa\_uzytkownika>\AppData\Local\BraveSoftware\Brave-Browser\User Data\Default\Network\Cookies

### **Safari**

*SESJA*



A diagram showing three separate browser sessions at the top, each containing a unique session ID (S=A123, S=B345, S=C678). A yellow double-headed arrow labeled "Space" spans the width of the three sessions. Below them is a large green rectangular area representing the "Django Session Middleware". Inside this middleware layer are three white cylinder icons labeled "A123", "B345", and "C678", which correspond to the session IDs in the browser sessions above. The entire setup is set against a black background.

Space

Browser  
S=A123

Browser  
S=B345

Browser  
S=C678

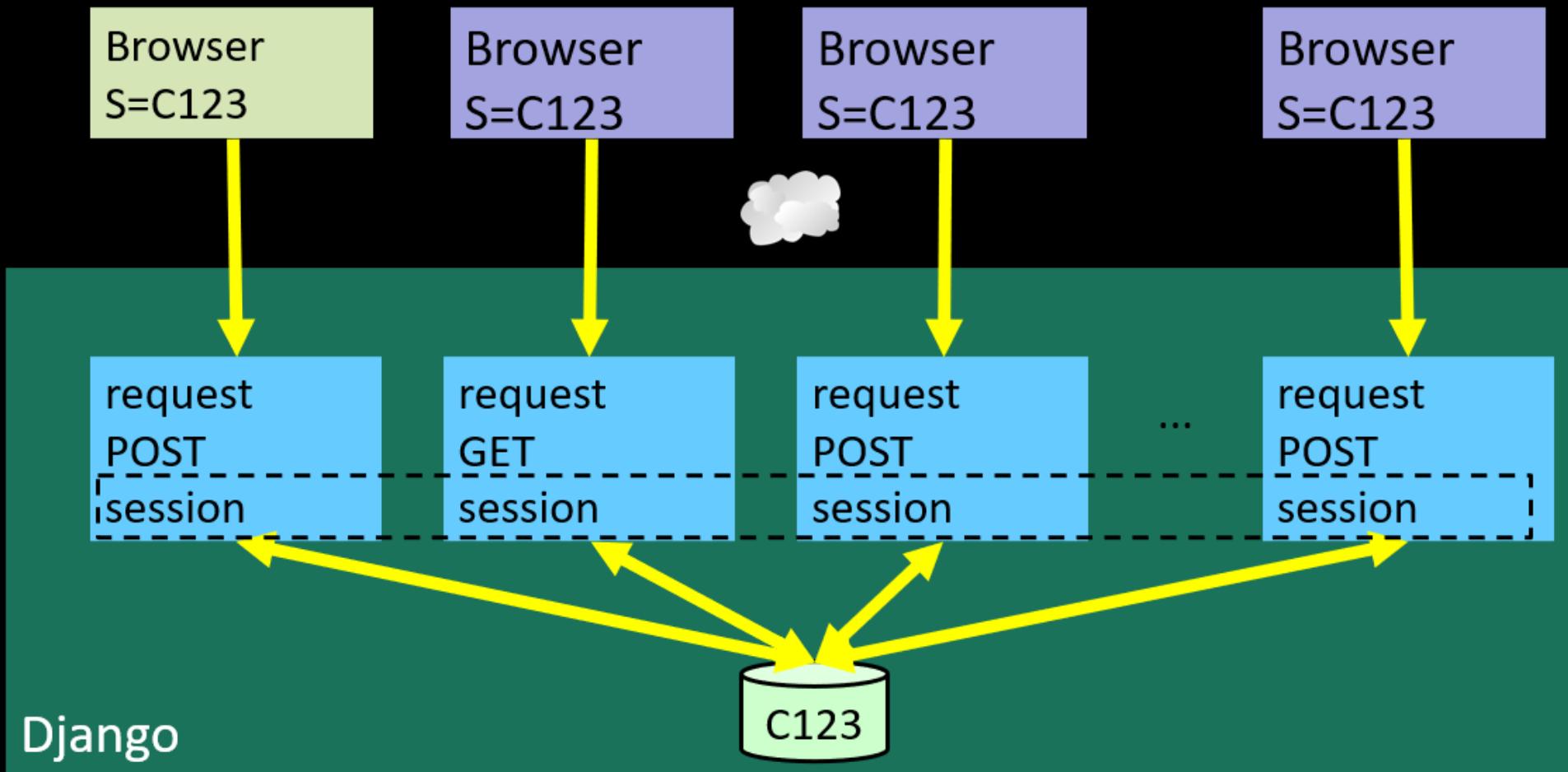
Django Session Middleware

A123 B345

C678

Web Server

Time



# *PYTANIA*

# PYTANIE I

Czym jest obiekt `request.GET` ?

- A. pythonowym słownikiem
- B. pythonową listą
- C. pythonowym obiektem słownikopodobnym
- D. żadne z powyższych

## PYTAÑIE II

W której z podatności atakujący wstrzykuje szkodliwy kod do linku pochodzącego z zaufanego źródła ?

- A. Cross-site scripting
- B. Command injection
- C. Cross-site request forgery
- D. żadne z powyższych

# PYTAŃIE III

Atrybut action elementu html mówi:

- A. jaką metodą dane formularza mają zostać wysłane
- B. gdzie mają zostać przesłane dane formularza
- C. kiedy mają być przesłane dane formularza
- D. żadne z powyższych

# PYTANIE IV

Osoba spoglądająca na ekran nigdy nie zobaczy przekazywanych danych jeżeli w formularzu html użyjemy tagu `<form method="get">`.

- A. Prawda
- B. Fałsz

# PYTANIE V

Które z poniższych twierdzeń dotyczących bezpieczeństwa formularzy Django jest prawdziwe?

- A. Token CSRF jest wstawiany za pomocą tagu `{%csrf_token%}` wewnątrz widoku w celu uniemożliwienia ataku Cross-Site Request Forgery.
- B. Django automatycznie zapewnia bezpieczeństwo formularzy.
- C. Walidacja formularzy zabezpiecza przed wstrzykiwaniem sql.
- D. Token csrf jest generowany tylko podczas pierwszego wyświetlania strony.



*MATERIALY*  
*POMOCNICZE*

# Django for Everybody

This web site is building a set of free materials, lectures, and assignments to help students learn the Django web development framework. You can take this course and receive a certificate at:

- Coursera: [Django for Everybody Specialization ↗](#)
- edX: [Django for Everybody XSeries Program ↗](#)
- FutureLearn: [Django for Everybody ↗](#)
- FreeCodeCamp: [Django for Everybody ↗](#)
- Free certificates for University of Michigan students and staff [↗](#)



We use the free [PythonAnywhere ↗](#) hosting environment to deploy and test our Django projects and applications. You can keep using this hosting environment to develop and deploy your Django applications after you complete the course.

## Technology

This site uses [Tsugi ↗](#) framework to embed a learning management system into this site and handle the autograders. If you are interested in collaborating to build these kinds of sites for yourself, please see the [tsugi.org ↗](#) website.

DJ4E

## Copyright

# *DODATKI*

*FUNKCJA RE\_PATH I  
WYRAZENIA REGULARNE  
WE WZORCACH URL*

The screenshot shows the PyCharm IDE interface with three code editors open:

- urls.py**: Contains a URL pattern for a flight list view. A red number '1' is placed next to the 're\_path' line.

```
from django.urls import path
from django.urls import re_path

from user_data import views

urlpatterns = [
    re_path(
        'flights/(?P<year>[0-1][0-9]{2})/$',
        views.flights_list,
    ), # regex
]
```
- views.py**: Contains a view function for the flight list. A red number '1' is placed next to the 'year' parameter in the context dictionary. A yellow status bar at the bottom indicates 'A 1 ^ v'.

```
from django.shortcuts import render

def flights_list(request, year):
    return render(
        request,
        'user_data/flights_list.html',
        context={
            'year': int(year), # year jest stringiem
        }
)
```
- flights\_list.html**: Contains an HTML template for displaying the flight list. It includes a title and a heading with a placeholder for the year.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista lotów</title>
</head>
<body>
    <h3>Lista lotów na rok {{ year }}:</h3>
    ...
</body>
</html>
```

*WYRAZENIA REGULARNE*

The screenshot shows a PyCharm IDE interface with three open files:

- urls.py**: Contains URL patterns for flights. A red arrow points from the regex pattern in the code to the corresponding placeholder in the **flights\_list.html** template.
- views.py**: Contains a function `flights_list` that renders a template. A red arrow points from the `year` parameter in the function call to its definition in the template.
- flights\_list.html**: An HTML template with a title and a header section containing the placeholder `{{ year }}`.

The IDE's navigation sidebar on the left includes tabs for Project, DB Browser, Pull Requests, and Favorites. The status bar at the bottom displays the text "WYRAZENIA REGULARNE".

```
from django.urls import path
from django.urls import re_path
from user_data import views

urlpatterns = [
    re_path(
        'flights/(?P<year>2[0-1][0-9]{2})/$',
        views.flights_list,
    ), # regex
]
```

```
from django.shortcuts import render

def flights_list(request, year):
    return render(
        request,
        'user_data/flights_list.html',
        context={
            'year': int(year), # year jest stringiem
        }
)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista lotów</title>
</head>
<body>
    <h3>Lista lotów na rok {{ year }}:</h3>
    ...
</body>
</html>
```

*WYRAZENIA REGULARNE*

The screenshot shows the PyCharm IDE interface with three code editors open:

- urls.py**: Contains URL patterns for a 'flights' app, including a regex path for 'flights/2[0-1][0-9]{2}/\$' pointing to the 'flights\_list' view.
- views.py**: Contains a 'flights\_list' view function that renders 'flights\_list.html' with a context variable 'year'.
- flights\_list.html**: An HTML template with a title and a header section containing the text "Lista lotów na rok {{ year }}".

A red arrow points from the 'year' variable in the Python code to the corresponding placeholder in the HTML template. A red number '3' is placed near the arrow, likely indicating a step or part of a diagram.

```
from django.urls import path
from django.urls import re_path
from user_data import views

urlpatterns = [
    re_path(
        'flights/(?P<year>2[0-1][0-9]{2})/$',
        views.flights_list,
    ), # regex
]
```

```
from django.shortcuts import render

def flights_list(request, year):
    return render(
        request,
        'user_data/flights_list.html',
        context={
            'year': int(year), # year jest stringiem
        }
)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista lotów</title>
</head>
<body>
    <h3>Lista lotów na rok {{ year }}:</h3>
    ...
</body>
</html>
```

*WYRAZENIA REGULARNE*

backend\_zdpytpol40\_2 > user\_data > views.py

Project

DB Browser

Pull Requests

Favorites

Structure

urls.py

```
from django.urls import path
from django.urls import re_path
from user_data import views

urlpatterns = [
    re_path(
        'flights/(?P<year>\d{4})',
        views.flights_list,
    ), # regex
]
```

Lista lotów

+

views.py

```
from django.shortcuts import render
```

127.0.0.1:8000/user-data/flights/2011/

***Lista lotów na rok 2011:***

flights\_list.htm

```
<!DOCTYPE html>
<html lang="pl">
    <head>
        <meta charset="UTF-8">
        <title>Lista lotów</title>
    </head>
    <body>
        <h3>Lista lotów na rok {{ year }}:</h3>
        ...
    </body>
</html>
```

*WYRAZENIA REGULARNE*

## Using regular expressions

If the paths and converters syntax isn't sufficient for defining your URL patterns, you can also use regular expressions. To do so, use `re_path()` instead of `path()`.

In Python regular expressions, the syntax for named regular expression groups is `(?P<name>pattern)`, where `name` is the name of the group and `pattern` is some pattern to match.

Here's the example URLconf from earlier, rewritten using regular expressions:

```
from django.urls import path, re_path

from . import views

urlpatterns = [
    path('articles/2003/', views.special_case_2003),
    re_path(r'^articles/(?P<year>[0-9]{4})/$', views.year_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$',
        views.month_archive),
    re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>
        [\w-]+)/$', views.article_detail),
]
```

*WYRAZENIA  
REGULARNE*

This accomplishes roughly the same thing as the previous example, except:

- The exact URLs that will match are slightly more constrained. For example, the year 10000 will no longer match since the year integers are constrained to be exactly four digits long.

*WLASNE  
KONWERTERY  
FUNKCJI PATH*

The screenshot shows a PyCharm IDE interface with the following files open:

- urls.py**:

```
from django.urls import path
from django.urls import register_converter

from user_data import views
from user_data import converters

register_converter(converters.YearConverter, 'yyyy')

app_name = 'user_data'

urlpatterns = [
    path(
        'tasks/<yyyy:year>',
        views.tasks_list,
    ), # custom path converters
]
```
- views.py**:

```
from django.shortcuts import render

def tasks_list(request, year):
    return render(
        request,
        'user_data/tasks_list.html',
        context={
            'year': year, # year już jest intem (YearConverter)
        }
)
```
- converters.py**:

```
class YearConverter:
    regex = '2[0-1][0-9]{2}'

    def to_python(self, value):
        # co robimy z dopasowanym stringiem
        return int(value)

    def to_url(self, value):
        # co robimy ze zmienne kiedy znajdzie
        # się w tagu url jako parametr

        return f"{value}"
```
- tasks\_list.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista zadań</title>
</head>
<body>
    <h3>Lista zadań na rok {{ year }}</h3>
    ...
</body>
</html>
```

A red box highlights the `YearConverter` class definition in `converters.py`. A large red number "1" is positioned to the right of the code block.

*WŁASNE KONWERTERY FUNKCJI PATH*

The screenshot shows the PyCharm IDE interface with several open files:

- urls.py**:

```
from django.urls import path
from django.urls import register_converter

from user_data import views
from user_data import converters

register_converter(converters.YearConverter, 'yyyy')

app_name = 'user_data'

urlpatterns = [
    path(
        'tasks/<yyyy:year>',
        views.tasks_list,
    ), # custom path converters
]
```
- views.py**:

```
from django.shortcuts import render

def tasks_list(request, year):
    return render(
        request,
        'user_data/tasks_list.html',
        context={
            'year': year, # year już jest intem (YearConverter)
        }
)
```
- converters.py**:

```
class YearConverter:
    regex = '2[0-1][0-9]{2}'

    def to_python(self, value):
        # co robimy z dopasowanym stringiem
        return int(value)

    def to_url(self, value):
        # co robimy ze zmienią kiedy znajdzie
        # się w tagu url jako parametr

        return f"{value}"
```
- tasks\_list.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista zadań</title>
</head>
<body>
    <h3>Lista zadań na rok {{ year }}</h3>
    ...
</body>
```

A red arrow points from the line `register_converter(converters.YearConverter, 'yyyy')` in **urls.py** to the definition of `YearConverter` in **converters.py**. A large red number **2** is positioned above the **urls.py** tab.

*WŁASNE KONWERTERY FUNKCJI PATH*

The screenshot shows the PyCharm IDE interface with four open files:

- urls.py**:

```
from django.urls import path
from django.urls import register_converter

from user_data import views
from user_data import converters

register_converter(converters.YearConverter, 'yyyy')

app_name = 'user_data'

urlpatterns = [
    path(
        'tasks/<yyyy:year>',
        views.tasks_list,
    ), # custom path converters
]
```
- views.py**:

```
from django.shortcuts import render

def tasks_list(request, year):
    return render(
        request,
        'user_data/tasks_list.html',
        context={
            'year': year, # year już jest intem (YearConverter)
        }
)
```

A red arrow points from the 'year' parameter in the `tasks_list` view to the `YearConverter` definition in `converters.py`, with the number '3' written above it.
- converters.py**:

```
class YearConverter:
    regex = '2[0-1][0-9]{2}'

    def to_python(self, value):
        # co robimy z dopasowanym stringiem
        return int(value)

    def to_url(self, value):
        # co robimy ze zmienią kiedy znajdzie
        # się w tagu url jako parametr

        return f"{value}"
```
- tasks\_list.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Lista zadań</title>
</head>
<body>
    <h3>Lista zadań na rok {{ year }}</h3>
    ...
</body>
```

*WŁASNE KONWERTERY FUNKCJI PATH*

The screenshot shows a PyCharm IDE interface with four open files:

- urls.py**: Contains URL patterns for a 'user\_data' app, including a path converter for the year.
- views.py**: Contains a view function 'tasks\_list' that uses the 'render' shortcut to return a template response.
- converters.py**: Contains a custom path converter class 'YearConverter' with methods 'to\_python' and 'to\_url'.
- tasks\_list.html**: A simple HTML template for displaying a task list for a given year.

A red arrow points from the 'year' placeholder in the 'tasks\_list.html' template to the 'YearConverter' class in 'converters.py'. Another red arrow points from the 'year' parameter in the 'tasks\_list' view function to the same 'YearConverter' class. The PyCharm editor highlights the 'YearConverter' class in all four files, indicating a strong reference or association between them.

# WLASNE KONWERTERY FUNKCJI PATH

The screenshot shows the PyCharm IDE interface with the following components:

- Top Bar:** PC, File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, DB Navigator, Help, backend\_zdpytpol40\_2 - converters.py
- Project Tree:** backend\_zdpytpol40\_2 > user\_data > converters.py
- Code Editors:**
  - urls.py:** from django.urls import path  
from django.urls import register\_converter  
  
from user\_data import views  
from user\_data import converters  
  
register\_converter(converters.YearsConverter, 'year')  
  
app\_name = 'user\_data'  
  
urlpatterns = [  
 path('tasks/', views.tasks\_list), # custom converter]
  - views.py:** from django.shortcuts import render  
  
def tasks\_list(request, year):  
 return render(  
 request,
- Browser Preview:** A modal window titled "Lista zadań" displays the text "Lista zadań na rok 2011".
- Bottom Editor:** tasks\_list.html (Structure view)

```
<!DOCTYPE html>  
<html lang="pl-PL">  
  <head>  
    <meta charset="UTF-8">  
    <title>Lista zadań</title>  
  </head>  
  <body>  
    <h3>Lista zadań na rok {{ year }}</h3>  
    ...  
  </body>
```
- Status Bar:** Add Configuration..., Git: ✓ ✓ ✓ ✓ ✓ ✓

*WŁASNE KONWERTERY FUNKCJI PATH*

URL dispatcher | Django document +

← → ⌂ https://docs.djangoproject.com/en/3.2/topics/http/urls/#registering-custom-path-converters

## Registering custom path converters

For more complex matching requirements, you can define your own path converters.

A converter is a class that includes the following:

- A `regex` class attribute, as a string.
- A `to_python(self, value)` method, which handles converting the matched string into the type that should be passed to the view function. It should raise `ValueError` if it can't convert the given value. A `ValueError` is interpreted as no match and as a consequence a 404 response is sent to the user unless another URL pattern matches.
- A `to_url(self, value)` method, which handles converting the Python type into a string to be used in the URL. It should raise `ValueError` if it can't convert the given value. A `ValueError` is interpreted as no match and as a consequence `reverse()` will raise `NoReverseMatch` unless another URL pattern matches.

**Changed in Django 3.1:**

Support for raising `ValueError` to indicate no match was added.

For example:

```
class FourDigitYearConverter:  
    regex = '[0-9]{4}'  
  
    def to_python(self, value):
```

*WLASNE  
KONWERTERY  
FUNKCJI PATH*



# *KLASA*

## *HTTPRESPONSEREDIRECT*

# *ALTERNatywna metoda przekierowywania klasa HttpResponseRedirect*

*Poza funkcją redirect do przekierowywania można użyć również klasy HttpResponseRedirect z modułu django.http*

*HttpResponseRedirect('<endpoint>')*

*W przypadku klasy HttpResponseRedirect, w celu uniknięcia 'zahardkodowywania' endpointów w kodzie aplikacji należy użyć funkcji resolve\_url z modułu django.shortcuts*

*HttpResponseRedirect( resolve\_url('<nazwa\_mapowowania>') )*

*W przypadku dodatkowych parametrów endpointu postępujemy analogicznie jak w przypadku funkcji redirect – umieszczamy parametry po przecinku*

*HttpResponseRedirect( resolve\_url('<nazwa\_mapowowania>'), <nazwa\_parametru>=<wartość\_parametru> )*



*ELEMENTY  
FORMULARZA HTML 5*

# *ELEMENTY FORMULARZA*

*Elementy formularza (aka pola formularza) to elementy html, z których budujemy formularz.*

*Zadaniem elementów formularza jest udostępnienie miejsca do wprowadzania danych. Dzięki elementom formularza użytkownik może wprowadzić informację do specjalnie przygotowanego pola, a formularz sam umieści wprowadzone w tym polu dane w odpowiednim miejscu podczas wysyłania zapytania (w miejscu zgodnym ze wskazaną w formularzu metodą http).*

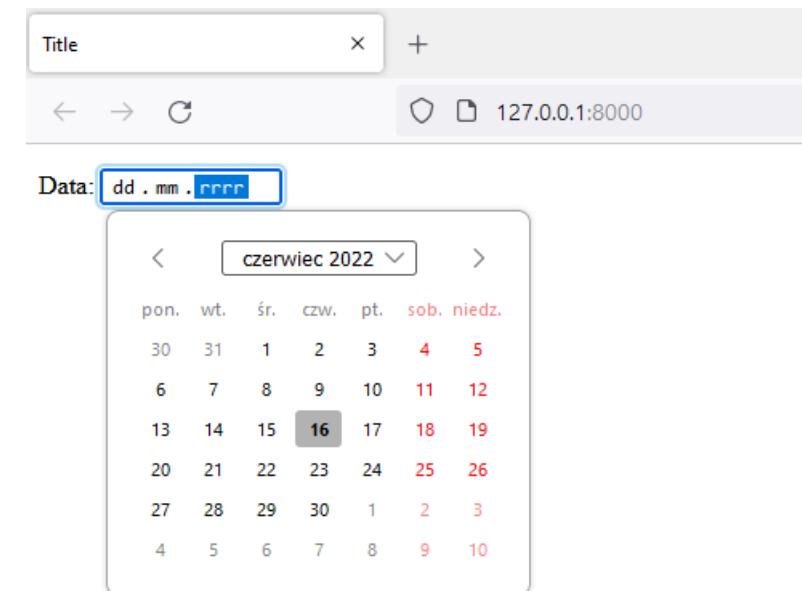
*Wyróżniamy 5 podstawowych elementów formularza:*

- *element <input>*
- *element <button>*
- *element <select>*
- *element <textarea>*
- *element <datalist>*

# ELEMENTY FORMULARZA - WIDGET

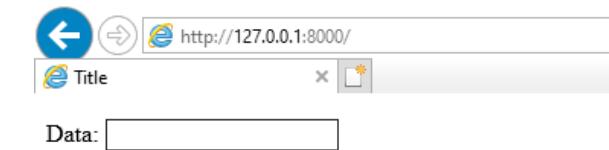
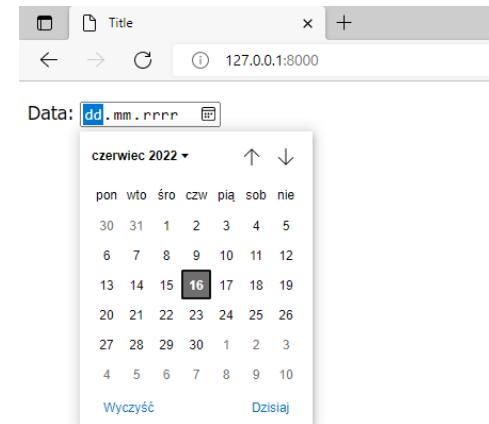
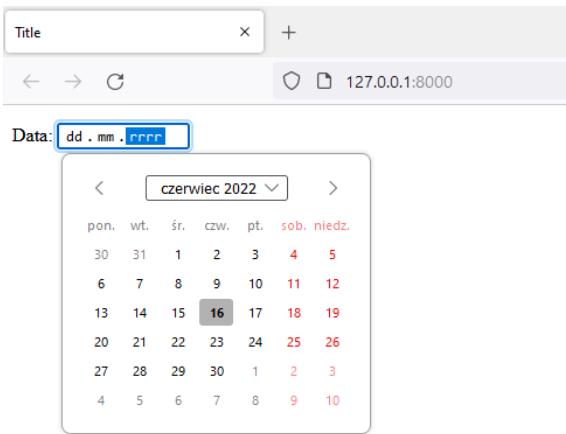
Przeglądarki internetowe, w celu zwiększenia własnej atrakcyjności wyposażają elementy formularza w wygodne **widżety** (elementy graficzne) do wprowadzania danych (np. kalendarz do wprowadzanie informacji o dacie).

```
<form>
    <p>Data: <input type="date" name="reservationDate"></p>
</form>
```



# ELEMENTY FORMULARZA - WIDGET

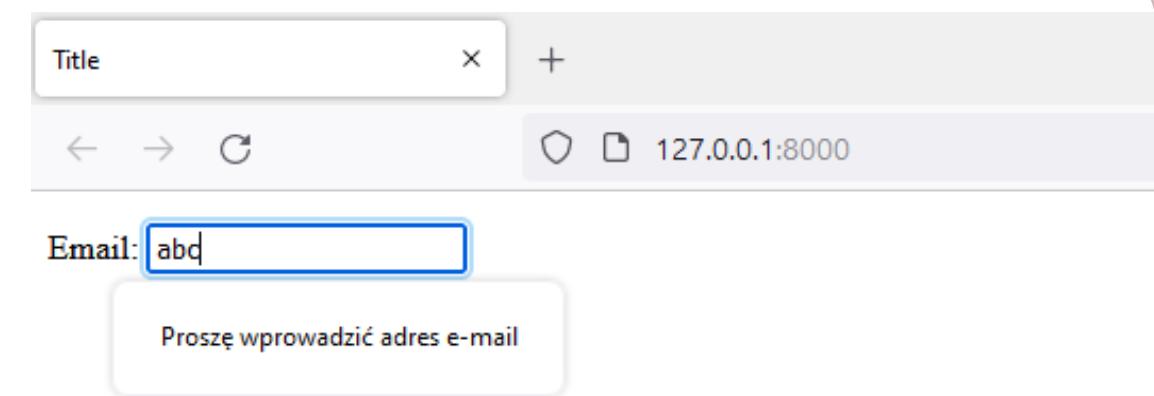
Widgety różnią się pomiędzy przeglądarkami, a niektóre przeglądarki mogą w ogóle nie implementować niektórych widgetów (np. brak dedykowanego widget-u dla elementu `input` typu `date` w IE).



# ELEMENTY FORMULARZA - WALIDATORY

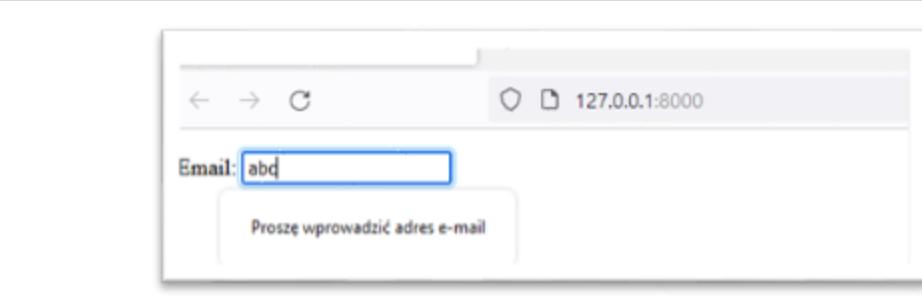
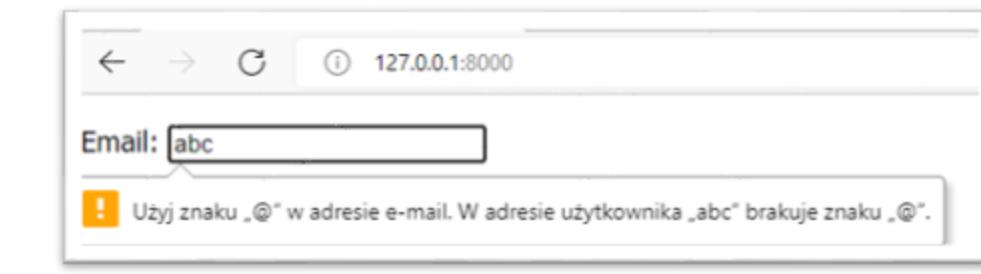
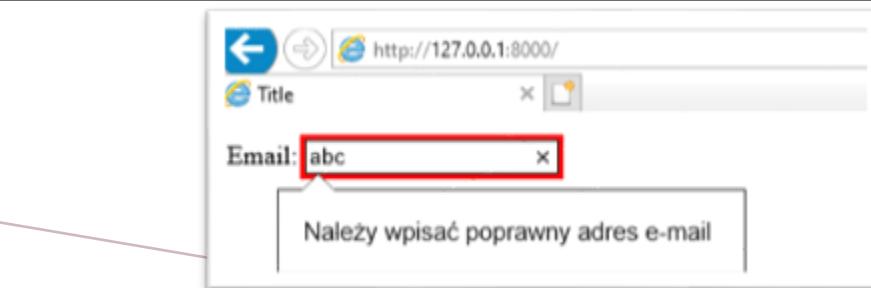
Przeglądarki internetowe, w celu zwiększenia własnej atrakcyjności wyposażają elementy formularza w **walidatory danych**. Walidatory przed wysłaniem zapytania sprawdzają, czy dane umieszczone w elemencie formularza mają format zgodny z typem elementu. W przypadku błędного formatu dane nie są wysyłane, a użytkownikowi zostaje wyświetlony odpowiedni komunikat.

```
<form>
  <p>Email: <input type="email" name="userEmail"></p>
</form>
```



# ELEMENTY FORMULARZA - WALIDATORY

Sposób działania validatorów może różnić się pomiędzy przeglądarkami.

*INPUT*

# INPUT

Element `<input>` to podstawowy element html służący do wprowadzania wartości takich jak tekst, data, wartość logiczna, itp.

Title  x +

← → C 127.0.0.1:8000

Username:  login <input type="text">

Password:  ..... <input type="password">

Gender: Male  Female  <input type="radio">

Enable features: Feature 1  Feature 2  <input type="checkbox">

submit <input type="submit">

reset <input type="reset">

dd, mm , yyyy <input type="date">

czerwiec 2022 <input type="button" value="<"/> <input type="button" value=">"/>

pon.	wt.	śr.	czw.	pt.	sob.	niedz.
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

# *TYPY ELEMENTU INPUT (HTML 5)*

<code>&lt;input type="text"&gt;</code>	<code>&lt;input type="tel"&gt;</code>	<code>&lt;input type="month"&gt;</code>	<code>&lt;input type="file"&gt;</code>	<code>&lt;input type="button"&gt;</code>
<code>&lt;input type="password"&gt;</code>	<code>&lt;input type="email"&gt;</code>	<code>&lt;input type="date"&gt;</code>	<code>&lt;input type="image"&gt;</code>	<code>&lt;input type="submit"&gt;</code>
<code>&lt;input type="hidden"&gt;</code>	<code>&lt;input type="url"&gt;</code>	<code>&lt;input type="datetime-local"&gt;</code>	<code>&lt;input type="radio"&gt;</code>	
<code>&lt;input type="number"&gt;</code>	<code>&lt;input type="time"&gt;</code>	<code>&lt;input type="search"&gt;</code>	<code>&lt;input type="checkbox"&gt;</code>	
<code>&lt;input type="range"&gt;</code>	<code>&lt;input type="week"&gt;</code>	<code>&lt;input type="color"&gt;</code>	<code>&lt;input type="reset"&gt;</code>	

## *PODSTAWOWA STRUKTURA ELEMENTU INPUT*

```
<input type="inputType" name="inputName" value="InitialValue"/>
```

*Element input posiada trzy podstawowe atrybuty:*

- *type – atrybut reprezentujący typ elementu (np. text, password, radio)*
- *name – atrybut reprezentujący unikalną nazwę elementu*
- *value - atrybut reprezentujący początkową wartość elementu*

# DODATKOWE ATRYBUTY ELEMENTU INPUT

Elementy input posiadają 10 wspólnych, dodatkowych atrybutów. Elementy input typu checkbox i radio posiadają ponadto atrybut checked. Rolą dodatkowych atrybutów elementu input jest przeważnie rozszerzenie logiki walidującej wprowadzane dane.

disabled	min	required	value
max	pattern	size	checked (tylko dla checkbox i radio)
maxlength	readonly	step	

*SELECT*

```
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form>
        <select name="fruit">
            <option value="0">Arbus</option>
            <option value="1">Pomarańcza</option>
            <option value="2">Jabłko</option>
        </select>
        <input type="submit" value="Wyślij">
    </form>
</body>
```

## SELECT

Select to element formularza implementujący listę rozwijalną.

## *SELECT - WIDGET*

A screenshot of a web browser window titled "title". The address bar shows "127.0.0.1:8000". The page content displays a dropdown menu with the following options:

- Arbus (selected)
- Arbus
- Pomarańcza
- Jabłko

Next to the dropdown is a button labeled "Wyślij".

*Select to element formularza  
implementujący listę rozwijalną*

# *SELECT*

```
<select name="fruit">  
    <option value="0">Arbus</option>  
</select>
```

Nazwa zmiennej  
Wartość zmiennej      Etykieta

*Struktura html/*

# ATRYBUTY ELEMENTU SELECT

- *selected* - początkowo wybrana etykieta
- *size* – liczba widocznych wpisów. Domyślna wartość to 1. Dla wartości 1 po kliknięciu w element rozwija się lista. Dla wartości większej od 1 lista nie jest rozwijalna. Zamiast tego wyświetla się liczba wpisów odpowiadająca wartości atrybutu size z paskiem przewijania po prawej stronie.
- *multiple* - możliwość wybrania więcej niż jednej wartości (domyślnie z listy można wybrać tylko jedną wartość)

The screenshot shows a PyCharm IDE interface with two code editors and a terminal window.

**Code Editors:**

- views.py**: Python code defining an `index` view that prints the value of the `'fruit'` parameter from the GET request. A red underline highlights the `request.GET.get('fruit')` call, labeled with a red number **2**.
- form.html**: HTML form code containing a dropdown menu with three options: `<option value="0">Arbus</option>`, `<option value="1">Pomarańcza</option>`, and `<option value="2">Jabłko</option>`. An input button is also present. This code is labeled with a red number **1**.

**Terminal:**

```
[01/May/2022 12:42:50] "GET /?fruit=0 HTTP/1.1" 200 386  
None  
[01/May/2022 12:43:05] "GET / HTTP/1.1" 200 386  
0 3  
[01/May/2022 12:43:07] "GET /?fruit=0 HTTP/1.1" 200 386
```

*SELECT  
(WDJANGO)*

## *SELECT Z ATRYBUTEM MULTIPLE*

A screenshot of a web browser window. The title bar says "Title". Below it is a toolbar with back, forward, and refresh buttons. To the right of the toolbar is the URL "127.0.0.1:8000". On the left, there is a dropdown menu with three items: "Arbus", "Pomarańcza", and "Jabłko". The item "Jabłko" is highlighted with a blue background. At the bottom right of the dropdown is a "Wyślij" button.

*Atrybut multiple umożliwia wybór wielu elementów*

The screenshot shows the PyCharm IDE interface with two code editors and a terminal window.

**Left Editor (views.py):**

```
from django.shortcuts import render, redirect
from django.http import HttpResponse

def index(request):
    if request.method == "GET":
        print(request.GET.get('fruit'))

    return render(
        request,
        'formapp/form.html'
    )
```

**Right Editor (form.html):**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form>
        <select name="fruit" multiple>
            <option value="0">Arbus</option>
            <option value="1">Pomarańcza</option>
            <option value="2">Jabłko</option>
        </select>
        <input type="submit" value="Wyślij">
    </form>
</body>
```

**Terminal Output:**

```
[01/May/2022 14:25:42] "GET /?fruit=0 HTTP/1.1" 200 395
None
[01/May/2022 14:26:38] "GET / HTTP/1.1" 200 395
2
[01/May/2022 14:26:41] "GET /?fruit=0&fruit=2 HTTP/1.1" 200 395
```

*SELECT Z ATRYBUTEM MULTIPLE - GET  
(METODA GET ZWRACA OSTATNI ELEMENT LISTY)*

The screenshot shows the PyCharm IDE interface with two files open:

- views.py** (left panel):

```
from django.shortcuts import render, redirect
from django.http import HttpResponse

def index(request):
    if request.method == "GET":
        print(request.GET.getlist('fruit'))

    return render(
        request,
        'formapp/form.html'
    )
```
- form.html** (right panel):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form>
        <select name="fruit" multiple>
            <option value="0">Arbus</option>
            <option value="1">Pomarańcza</option>
            <option value="2">Jabłko</option>
        </select>
        <input type="submit" value="Wyślij">
    </form>
</body>
```

The terminal at the bottom shows the server output:

```
Quit the server with CTRL-BREAK.
[]
[01/May/2022 14:28:02] "GET / HTTP/1.1" 200 395
['0', '2']
[01/May/2022 14:28:06] "GET /?fruit=0&fruit=2 HTTP/1.1" 200 395
```

***SELECT Z ATRYBUTEM MULTIPLE - GETLIST  
(METODA GETLIST ZWRACA WSZYSTKIE ZAZNACZONE ELEMENTY)***

*TEXTAREA*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form>
        <textarea name="foo"></textarea>
        <input type="submit" value="Wyślij">
    </form>
</body>
</html>
```

## TEXTAREA

*<textarea> to element formularza implementujący wielolinijkowe pole tekstowe.*

## *TEXTAREA*

A screenshot of a web browser window. The title bar says "Title". The address bar shows "127.0.0.1:8000". The main content area contains a text area with the following text:

```
To jest mój  
wielolinijkowy napis
```

The word "wielolinijkowy" is underlined with a red wavy line, indicating it might be misspelled. There is a small icon of a document with a diagonal line through it next to the text area. To the right of the text area is a blue button labeled "Wyślij".

*<textarea>* to element formularza implementujący wielolinijkowe pole tekstowe.

The screenshot shows a PyCharm IDE interface with two code editors and a terminal window.

**Code Editors:**

- views.py**: Python code defining a view named `index`. It prints the value of the `'foo'` parameter from the GET request.
- form.html**: HTML template containing a form with a text area and a submit button labeled "Wyślij".

**Terminal Output:**

```
None
[01/May/2022 15:03:26] "GET / HTTP/1.1" 200 236
To jest mój
wielolinijkowy napis
```

A red box highlights the text "To jest mój wielolinijkowy napis" in the terminal output, which corresponds to the value printed from the `request.GET.get('foo')` line in the views.py code.

# TEXTAREA (WDJANGO)

# *ATRYBUTY ELEMENTU TEXTAREA*

- *rows* - liczba wierszy pola tekstowego (wysokość pola). Domyślna wartość atrybutu rows różni się pomiędzy przeglądarkami.
- *cols* – liczba kolumn pola tekstowego (szerokość pola). Domyślna wartość atrybutu cols różni się pomiędzy przeglądarkami.

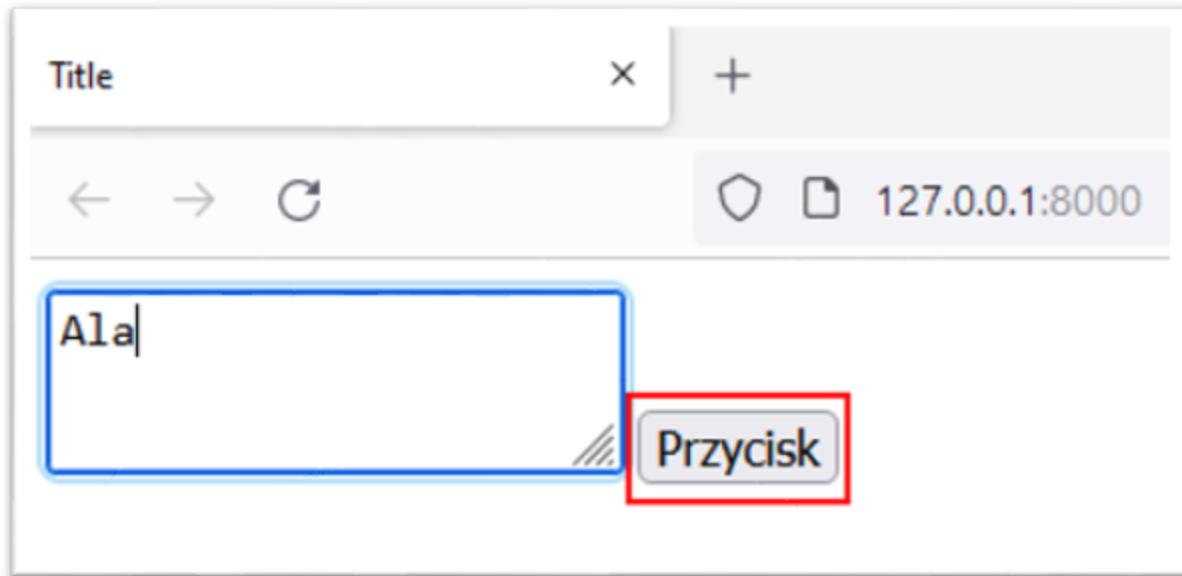
*BUTTON*

# BUTTON

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form>
        <textarea name="my_str"></textarea>
        <button name="foo" value="pressed">Przycisk</button>
    </form>
</body>
</html>
```

*<button>* to element formularza implementujący "klikalny" przycisk w formularzu.

## BUTTON



*<button>* to element formularza implementujący "klikalny" przycisk w formularzu.

The screenshot shows the PyCharm IDE interface with two code editors and a terminal window.

**Project Structure:**

- test\_08
- config
- formapp
  - migrations
  - templates
    - formapp
      - form.html
  - views.py
- venv library root
- .gitignore
- db.sqlite3
- manage.py
- README.md
- requirements.txt

**Code Editors:**

- views.py:** Contains Python code for a view named `index` which handles GET requests and prints the `request.GET` object.
- form.html:** Contains an HTML template with a form containing a text area and a button.

**Terminal:**

```
Quit the server with CTRL-BREAK.  
<QueryDict: {}>  
[01/May/2022 20:29:08] "GET / HTTP/1.1" 200 254  
<QueryDict: {'my_str': ['Ala'], 'foo': ['pressed']}>  
[01/May/2022 20:29:12] "GET /?my_str=Ala&foo=pressed HTTP/1.1" 200 254
```

*BUTTON*  
(*WDJANGO*)

# ATRYBUTY ELEMENTU BUTTON

- *type - typ przycisku. Domyślna wartość atrybutu type to "submit". Dostępne wartości atrybutu to: "button", "submit", "reset".*
  - *submit - przycisk wysyłający na serwer wartości wszystkich pól formularza.*
  - *reset - przycisk resetujący wszystkie wartości wprowadzone w polach formularza (bez komunikacji z serwerem).*
  - *button - typ domyślny. Przycisk do podpinania pod jego zdarzenia odpowiednich funkcji js. Domyślnie po kliknięciu w <button type="button"> nic się nie dzieje.*

# *DATALIST*

```
<title>Title</title>
</head>
<body>
<form>
    <input list="colors" name="color">
    <datalist id="colors">
        <option>czerwony</option>
        <option>niebieski</option>
        <option>żółty</option>
        <option>zielony</option>
        <option>czarny</option>
    </datalist>
    <input type="submit" value="Wyślij">
</form>
</body>
</html>
```

## DATALIST

*<datalist>* to element formularza łączący w sobie element *<input>* oraz *<select>*. Dodatkowym atutem elementu jest wyświetlanie na liście wyłącznie tych wartości, które zawierają wprowadzoną w elemencie *<input>* frazę.

## DATALIST

A screenshot of a web browser window titled "Title". The address bar shows "127.0.0.1:8000". Below the address bar is a search input field containing "czerwony" and a "Wyślij" button. To the right of the search input is a dropdown menu with the following options: "czerwony", "niebieski", "żółty", "zielony", and "czarny". The "czerwony" option is highlighted with a blue border.

`<datalist>` to element formularza łączący w sobie element `<input>` oraz `<select>`. Dodatkowym atutem elementu jest wyświetlanie na liście wyłącznie tych wartości, które zawierają wprowadzoną w elemencie `<input>` frazę.

## DATALIST

The screenshot shows a web browser window with a title bar labeled "Title". Below the title bar is a toolbar with navigation icons (back, forward, search) and a status bar showing the URL "127.0.0.1:8000". The main content area contains a form. On the left is a text input field with the placeholder "cz" and a dropdown menu below it listing "czerwony" and "czarny". To the right of the input field is a button labeled "Wyślij".

`<datalist>` to element formularza łączący w sobie element `<input>` oraz `<select>`. Dodatkowym atutem elementu jest wyświetlanie na liście wyłącznie tych wartości, które zawierają wprowadzoną w elemencie `<input>` frazę.

The screenshot shows the PyCharm IDE interface with two open files: `views.py` and `form.html`.

**views.py:**

```
from django.shortcuts import render

def index(request):
    if request.method == "GET":
        print(request.GET)

    return render(
        request,
        'formapp/form.html'
    )
```

**form.html:**

```
<title>Title</title>
</head>
<body>
<form>
    <input list="colors" name="color">
    <datalist id="colors">
        <option>czerwony</option>
        <option>niebieski</option>
        <option>żółty</option>
        <option>zielony</option>
        <option>czarny</option>
    </datalist>
    <input type="submit" value="Wyślij">
</form>
</body>
</html>
```

A red box highlights the `<form>` block in the `form.html` file.

**Terminal:**

```
Not Found: /favicon.ico
[01/May/2022 21:22:00] "GET /favicon.ico HTTP/1.1" 404 2271
<QueryDict: {'color': ['czerwony']}>
[01/May/2022 21:22:50] "GET /?color=czerwony HTTP/1.1" 200 477
```

*BUTTON  
(WDJANGO)*

## *SELECT*

*Select to element formularza implementujący listę rozwijalną.*

A screenshot of a web browser window titled "title". The address bar shows "127.0.0.1:8000". Below the address bar is a dropdown menu with four items: "Arbus", "Arbus", "Pomarańcza", and "Jabłko". The first item is highlighted. To the right of the dropdown is a button labeled "Wyślij".

# *ATRYBUTY ELEMENTU DATALIST*

- *list - atrybut elementu <input> wskazujący na listę powiązaną z danym inputem.*  
*Wartość atrybutu powinna być identyczna z wartością atrybutu id elementu <datalist>, który chcemy powiązać z inputem*
- *id – unikalny identyfikator elementu <datalist>*

# *DATALIST VS SELECT*

- *Element <datalist> w odróżnieniu od elementu <select> pozwala na wprowadzenie i przesłanie wartości nie znajdującej się na liście. Jeżeli chcemy ograniczyć użytkownikowi zbiór wartości to należy użyć elementu <select>.*
- *Element <datalist> w odróżnieniu od elementu <select> ogranicza listę wyświetlanych wartości do wartości odpowiadających wprowadzanej przez użytkownika frazie.*



SERWER



Klient

*ARCHITEKTURA  
Klient-Serwer*